Introducción a la Minería de Datos

Resumen de Prácticas

Juan José Méndez Torrero
31018712-S
i42metoj@uco.es

Universidad de Córdoba

29 de diciembre de 2018

Resumen

In the following document, we will solve the assignments for the course *Introducción a la Minería de Datos*. In order to do this, we will use the program Weka in order to see the evaluation of differents datasets all over the course. Moreover, we will use different configurations, all depending on the filter that we will need to solve the assignments.

Índice

1. Práctica 1: Preprocesamiento y exploración de datos **4**
1.1. Ejercicio 1: 4
1.2. Ejercicio 2: 6

2. Práctica 2: Clasificación básica **8**
2.1. Ejercicio 1: 8
2.2. Ejercicio 2: 8
2.3. Ejercicio 3: 11
2.4. Ejercicio 4: 12
2.5. Ejercicio 5: 12

3. Práctica 3: Clasificación avanzada **13**
3.1. Ejercicio 1: 13
3.2. Ejercicio 2: 13
3.3. Ejercicio 3: 14
3.4. Ejercicio 4: 14
3.5. Ejercicio 5: 15
3.6. Ejercicio 7: 15

4. Práctica 4: Clasificación usando método multiclase **16**
4.1. Ejercicio 1: 16
4.2. Ejercicio 2: 16
4.3. Ejercicio 3: 17
4.4. Ejercicio 4: 17
4.5. Ejercicio 5: 17

5. Práctica 5: Reglas de asociación **18**
5.1. Ejercicio 1: 18
5.2. Ejercicio 2: 18
5.3. Ejercicio 3: 18

6. Práctica 6: Agrupación y evaluación de resultados **20**
6.1. Ejercicio 1: 20
6.2. Ejercicio 2: 20
6.3. Ejercicio 4: 21
6.4. Ejercicio 5: 21

7. Bibliografía **28**

Índice de cuadros

1. Tabla recopilatoria con los CCR y RMSE de los filtros aplicados . . . 5
2. Tabla recopilatoria con los CCR y RMSE de los filtros aplicados para el conjunto de datos diabetes . . . 6
3. Tabla recopilatoria con los CCR y RMSE de los filtros aplicados para el conjunto de datos glass . . . 7
4. Tabla recopilatoria con los CCR y RMSE de los filtros aplicados para el conjunto de datos iris . . . 7
5. Tabla recopilatoria con los CCR y RMSE de los clasificadores aplicados al conjunto de datos diabetes . . . 8
6. Tabla recopilatoria con los CCR y RMSE de los clasificadores aplicados al conjunto de datos glass . . . 9
7. Tabla recopilatoria con los CCR y RMSE de los clasificadores aplicados al conjunto de datos iris . . . 9
8. Tabla recopilatoria con los CCR y RMSE de los clasificadores aplicados al conjunto de datos labor . . . 9
9. Tabla recopilatoria con los CCR y RMSE de los clasificadores aplicados al conjunto de datos vote . . . 9
10. Gráfico del error cometido . . . 10
11. Rango medio de los clasificadores . . . 10
12. Test de Wilconxon sobre los algoritmos MLP e IB1 . . . 11
13. Test de Friedman para los tres algoritmos . . . 11
14. Clasificador J48 aplicado a 10 datasets distintos . . . 13
15. Clasificador Bagging aplicado a 10 datasets distintos . . . 14
16. Algoritmo AdaBoostM1 aplicado a 10 datasets distintos . . . 14
17. Algoritmo LogitBoost aplicado a 10 datasets distintos . . . 15
18. Algoritmo J48 aplicado a datasets multiclase . . . 16
19. Métodos multiclase OvO, OvA y ECOC aplicados a 10 conjuntos de datos . . . 17
20. Método a priori con el conjunto de datos weather.nominal . . . 18
21. Método a priori con el conjunto de datos weather.nominal cambiando el parámetro minMetric . . . 19
22. Centroides y SSE obtenidos con el conjunto de datos glass . . . 22
23. Nº de cluster/SSE conjunto de datos glass . . . 22
24. Centroides y SSE obtenidos con el conjunto de datos ionosphere . . . 23
25. Nº de cluster/SSE conjunto de datos ionosphere . . . 24
26. Centroides y SSE obtenidos con el conjunto de datos poker . . . 24
27. Nº de cluster/SSE conjunto de datos poker . . . 25
28. Centroides y SSE obtenidos con el conjunto de datos sonar . . . 26
29. Nº de cluster/SSE conjunto de datos sonar . . . 26
30. Centroides y SSE obtenidos con el conjunto de datos iris . . . 27
31. Nº de cluster/SSE conjunto de datos iris . . . 27

1. Práctica 1: Preprocesamiento y exploración de datos

1.1. Ejercicio 1:

Usando como clasificador el método J48 compruebe el efecto de los dos filtros estudiados sobre el error de clasificación en uno de los conjuntos de datos de la UCI MLR.

Para dar una solución a este ejercicio, hemos elegido utilizar el conjunto de datos *diabetes*, descargado de la página web UCI MLR. Este conjunto de datos consta de 768 instancias y de 9 atributos de tipo numérico, exceptuando el atributo clase que es de tipo nominal. Primeramente, aplicando el clasificador J48 con los valores por defecto y sin ningún filtro sobre el error de clasificación podemos observar, en la Figura 1, el árbol resultante.

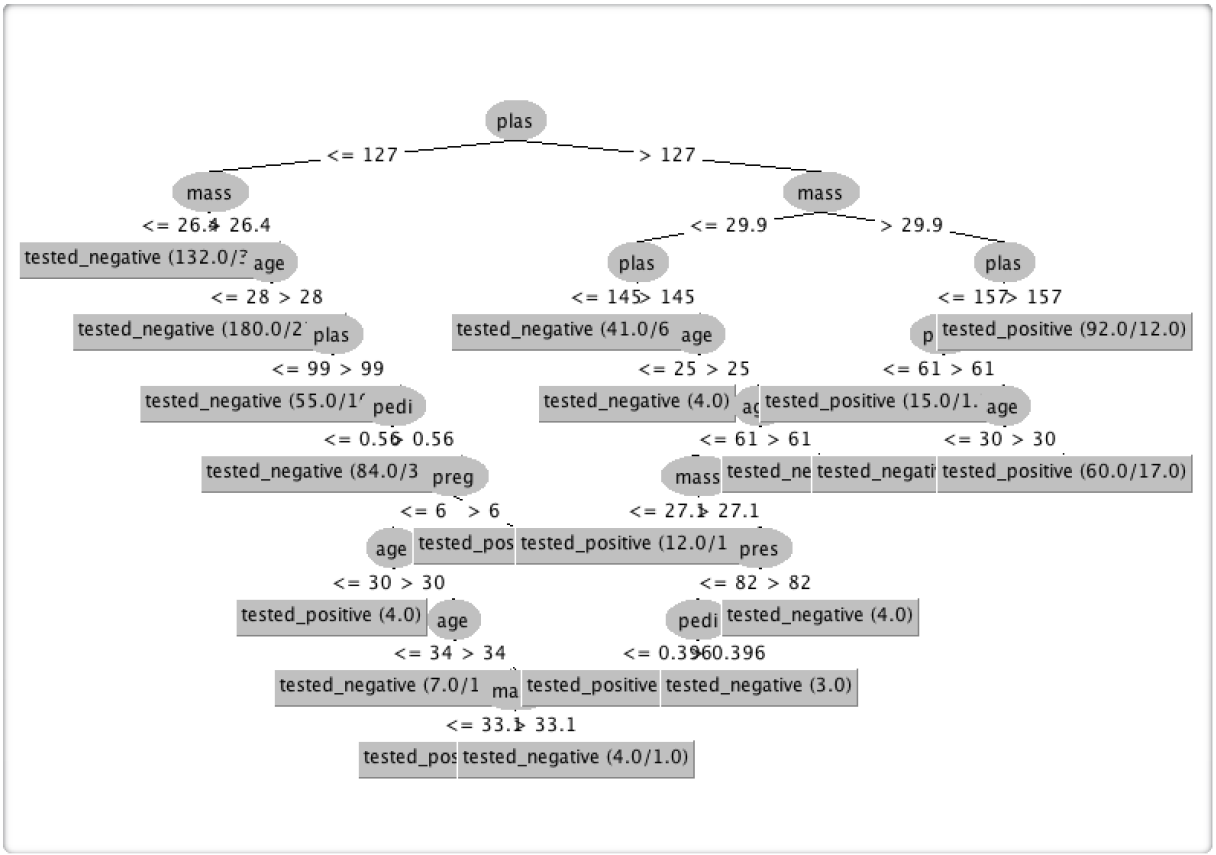


Figura 1: Árbol generado con el método J48 en el conjunto de datos diabetes

A continuación, en la Figura 2, podremos observar el árbol generado por el método de clasificación J48 pero esta vez aplicando, previamente, el filtro no supervisado *RandomProjection* con los valores por defecto.

Finalmente, hemos aplicado el filtro no supervisado *RemoveUseless*, que elimina los atributos que no varían demasiado o los que varían mucho. El problema con este

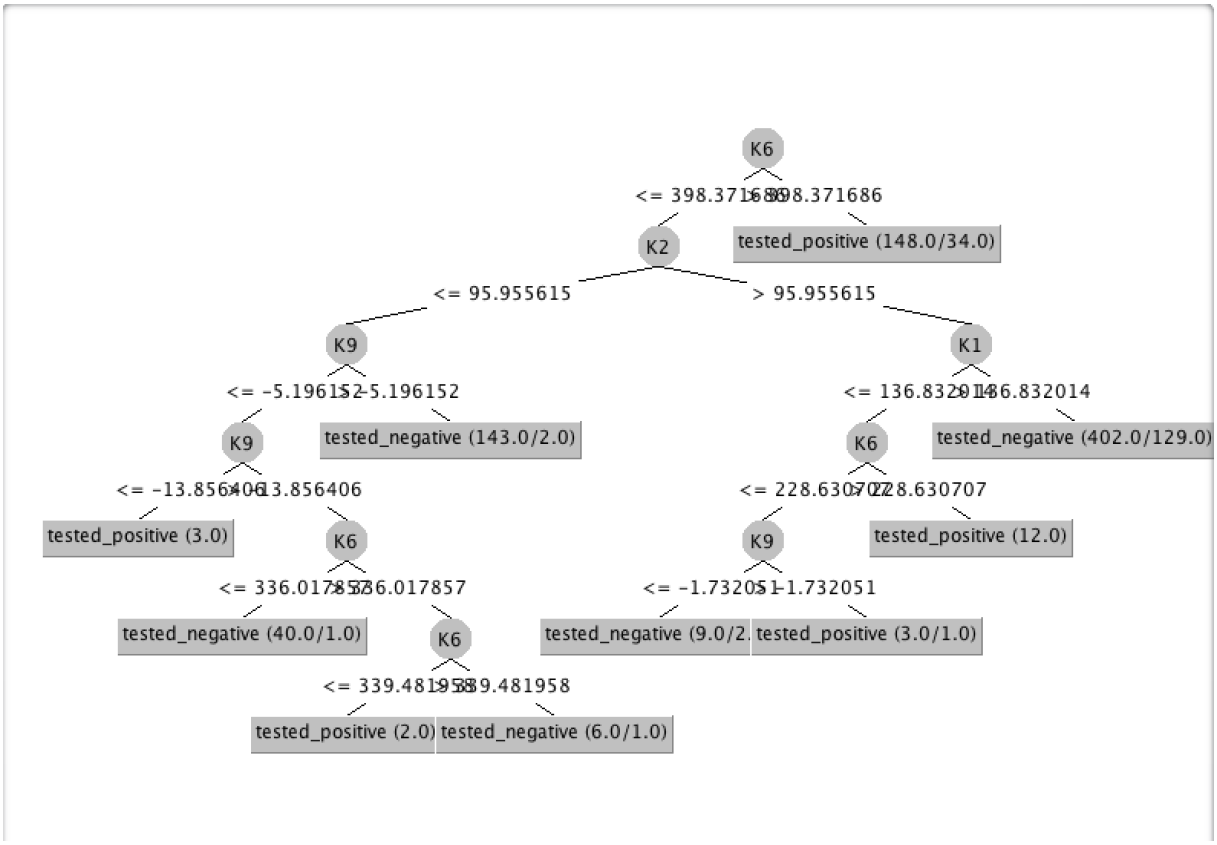


Figura 2: Árbol generado con el método J48 en el conjunto de datos diabetes con filtro Random-Proyection

filtro es que sólo es aplicable a variables de tipo nominal, siendo no este nuestro caso. Por consecuente, hemos decidido aplicar previamente el filtro no supervisado *NumericToNominal*, el cuál convierte las instancias de tipo numérico en tipo nominal. En este caso no mostraremos el árbol generado al sólo constar de un nodo.

Los valores de CCR y RMSE pueden ser observados en la Tabla 1.

Dataset: diabetes.arff	Correctly Classified Instances	Incorrectly Classified Instances	RMSE
Sin filtros	73,8281 %	26,1719 %	0.4463
RemoveUseless	65.1042 %	34.8958 %	0.4766
RandomProjection	74,2188 %	25,7813 %	0.4254

Cuadro 1: Tabla recopilatoria con los CCR y RMSE de los filtros aplicados

Como podemos observar, los mejores resultados son obtenidos aplicando el filtro no supervisado a los atributos *RandomProyection*. Como nota, estos resultados pueden ser mejorados si cambiamos la configuración por defecto de este filtro.

1.2. Ejercicio 2:

Usando como clasificador el método J48 compruebe el efecto de la normalización, la estandarización y el análisis en componentes principales sobre el error de clasificación en los conjuntos de datos.
En este ejercicio hemos decicido utilizar los siguientes conjuntos de datos:

- *diabetes*
- *glass*
- *iris*

A continuación vamos a ir explicando el efecto de la normalización, estadarización y análisis de componentes para cada uno de los conjuntos de datos.

1. **Conjunto de datos diabetes**: Como hemos explicado en el ejercicio anterior, este conjunto de datos consta de 768 instancias recogidas en 9 atributos de tipo numérico, exceptuando la clase, que es de tipo nominal. Seguidamente vamos a ir explicando el efecto de cada uno de los filtros comentados anteriormente y al final se expondrá una tabla con los resultados obtenidos.

   - **Normalización**: Como este conjunto de datos es tipo numérico, este filtro podría hacer que nuestros resultados varíen, pero éste no es el caso al devolver el mismo CCR y RMSE. Lo único que hace es discretizar los valores entre 0 y 1.
   - **Estadarización**: Como antes, al tener un conjunto de datos numéricos, sólo cambiaría el valor de estas instancias, pero no cambiaría ni el CCR ni el RMSE. Además, hace que las medias de las instancias sea 0 y que la desviación típica sea 1.
   - **Análisis de componentes**: Con este filtro, a diferencia de los dos anteriores, hace que nuestro CCR y RMSE cambien, y este cambio es peor que aplicando los dos filtros anteriores.

   Finalmente, en la Tabla 2 podemos ver cómo varían los resultados dependiendo del filtro no supervisado aplicado.

Dataset: diabetes.arff	Correctly Classified Instances	Incorrectly Classified Instances	RMSE
Sin filtros	73,8281 %	26,1719 %	0.4463
Normalize	73,8281 %	26,1719 %	0.4463
Standarize	73,8281 %	26,1719 %	0.4463
PrincipalComponent	70.7031 %	29.2969 %	0.4531

Cuadro 2: Tabla recopilatoria con los CCR y RMSE de los filtros aplicados para el conjunto de datos diabetes

2. **Conjunto de datos glass**: Este conjunto de datos consta de 214 instancias recogidas en 10 atributos de tipo numérico, excepto por el atributo clase que

es de tipo nominal.

Para no extender mucho, como este conjunto de datos es prácticamente igual que el anterior, es decir, los atributos son de tipo numérico, no vamos a explicar el efecto de los filtros no supervisados aplicados, ya que es el mismo que antes. Los resultados son observables en la Tabla 3.

Dataset: glass.arff	Correctly Classified Instances	Incorrectly Classified Instances	RMSE
Sin filtros	66.8224 %	33.1776 %	0.2897
Normalize	66.8224 %	33.1776 %	0.2904
Standarize	66.8224 %	33.1776 %	0.2904
PrincipalComponent	65.8879 %	34.1121 %	0.2862

Cuadro 3: Tabla recopilatoria con los CCR y RMSE de los filtros aplicados para el conjunto de datos glass

Como vemos, los peores resultados, como antes, los obtenemos aplicando el filtro no supervisado *PrincipalComponent*. Esto es debido a que el número de atributos es reducido de 10 a 7.

3. **Conjunto de datos iris**: Este conjunto de datos consta de 150 instancias agrupadas en 5 atributos que, como hasta ahora, son todos de tipos numéricos excepto por el atributo clase, que es de tipo nominal.

   El efecto de los filtros no supervisados para este conjunto de datos es el mismo que hasta ahora, con lo que no vamos a volver a explicarlos. La única diferencia notables es que con el filtro no supervisado *PrincipalComponent*, el número de atributos es reducido de 5 a 3. Los resultados obtenidos para el CCR y RMSE se pueden observar en la Tabla 4.

Dataset: iris.arff	Correctly Classified Instances	Incorrectly Classified Instances	RMSE
Sin filtros	96 %	4 %	0.1586
Normalize	96 %	4 %	0.1586
Standarize	96 %	4 %	0.1586
PrincipalComponent	85.3333 %	14.6667 %	0.2841

Cuadro 4: Tabla recopilatoria con los CCR y RMSE de los filtros aplicados para el conjunto de datos iris

Como nota, cabe destacar que con estos conjuntos de datos el peor filtro no supervisado que podemos aplicar es el de *PrincipalCompenent*, pero podría darse el caso en el que los resultados tras aplicar este filtro mejorasen.

2. Práctica 2: Clasificación básica

2.1. Ejercicio 1:

Seleccione 3 clasificadores dentro de los disponibles en Weka y 5 conjuntos de datos. No use combinaciones de modelos. Para la resolución de este ejercicio hemos elegido los siguientes conjuntos de datos:

- *diabetes*.
- *glass*.
- *iris*.
- *labor*.
- *vote*.

Los clasificadores elegidos para esta práctica han sido los siguientes:

- MultilayerPerceptron.
- IBK (con $K = 1$).
- NaiveBayes.

2.2. Ejercicio 2:

Use como método para obtener el error la validación cruzada de 10 particiones. Ejecute para cada clasificador seleccionado un entrenamiento y anote el error. Represente gráficamente el error obtenido con cada uno de los métodos de clasificación.

En este ejercicio mostraremos los resultados para cada conjunto de datos y veremos qué clasificador nos ofrece mejores resultados.

1. **Conjunto de datos diabetes**:

Dataset: diabetes.arff	Correctly Classified Instances	Incorrectly Classified Instances	RMSE
MultilayerPerceptron	75.3906 % (2)	24.6094 %	0.4215
IB1	70.1823 % (3)	29.8177 %	0.5453
NaiveBayes	76.3021 % (1)	23.6979 %	0.4168

Cuadro 5: Tabla recopilatoria con los CCR y RMSE de los clasificadores aplicados al conjunto de datos diabetes

2. **Conjunto de datos glass**:

Dataset: glass.arff	Correctly Classified Instances	Incorrectly Classified Instances	RMSE
MultilayerPerceptron	67.757 % (2)	32.243 %	0.2627
IB1	70.5607 % (1)	29.4393 %	0.2852
NaiveBayes	48.5981 % (3)	51.4019 %	0.3399

Cuadro 6: Tabla recopilatoria con los CCR y RMSE de los clasificadores aplicados al conjunto de datos glass

3. **Conjunto de datos iris**:

Dataset: iris.arff	Correctly Classified Instances	Incorrectly Classified Instances	RMSE
MultilayerPerceptron	97.3333 % (1)	2.6667 %	0.1291
IB1	95.3333 % (3)	4.6667 %	0.1747
NaiveBayes	96 % (2)	4 %	0.1550

Cuadro 7: Tabla recopilatoria con los CCR y RMSE de los clasificadores aplicados al conjunto de datos iris

4. **Conjunto de datos labor**:

Dataset: labor.arff	Correctly Classified Instances	Incorrectly Classified Instances	RMSE
MultilayerPerceptron	85.9649 % (2)	14.0351 %	0.3370
IB1	82.4561 % (3)	17.5439 %	0.4113
NaiveBayes	89.4737 % (1)	10.5263 %	0.2637

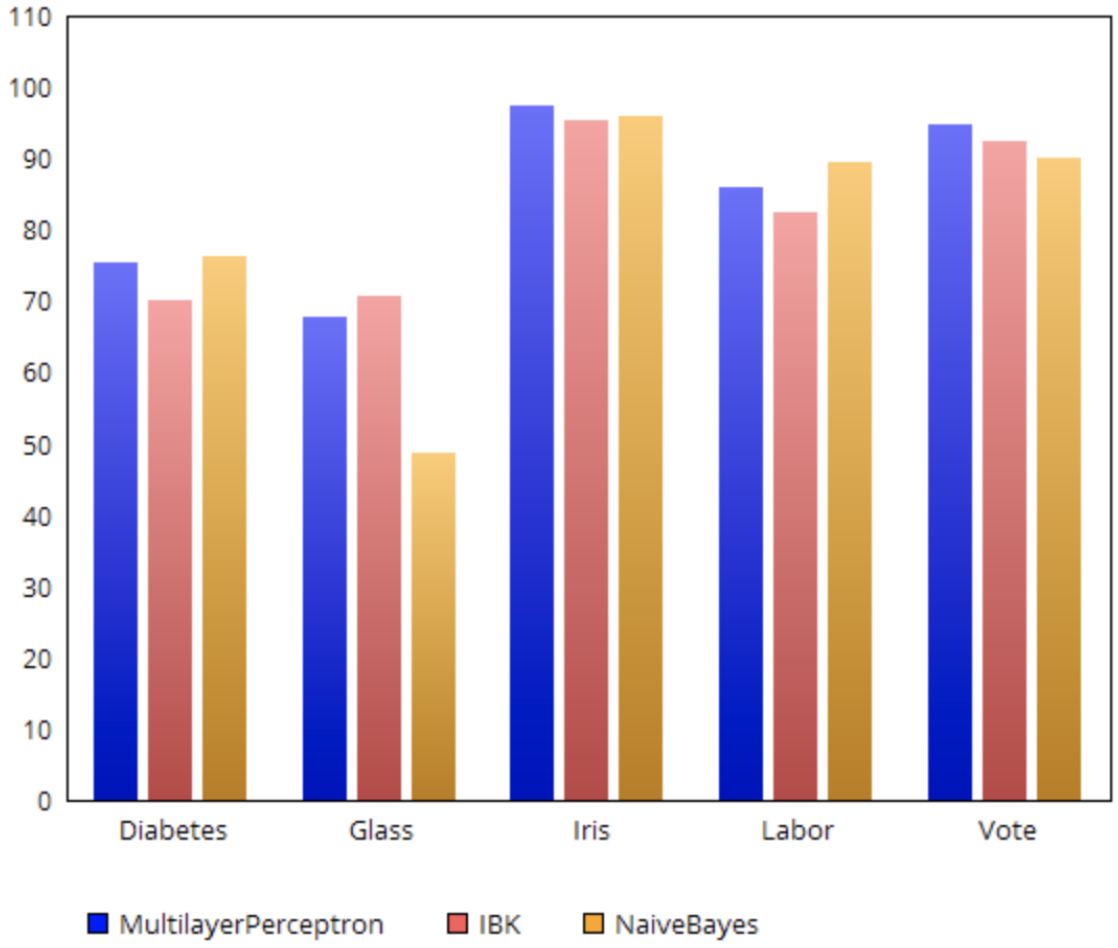
Cuadro 8: Tabla recopilatoria con los CCR y RMSE de los clasificadores aplicados al conjunto de datos labor

5. **Conjunto de datos vote**:

Dataset: vote.arff	Correctly Classified Instances	Incorrectly Classified Instances	RMSE
MultilayerPerceptron	94.7126 % (1)	5.2874 %	0.2078
IB1	92.4138 % (2)	7.5862 %	0.2420
NaiveBayes	90.1149 % (3)	9.8851 %	0.2977

Cuadro 9: Tabla recopilatoria con los CCR y RMSE de los clasificadores aplicados al conjunto de datos vote

Seguidamente, vamos a mostrar un gráfico en el que veremos mejor qué error han cometido cada uno de los clasificadores para cada conjunto de datos elegido. El resultado puede ser observado en el Gráfico 10



Cuadro 10: Gráfico del error cometido

Por último, en la Tabla 11 podremos observar los rangos medios de cada uno de los clasificadores con los conjuntos de datos usados:

	MultilayerPerceptron	IB1	NaiveBayes
Rango medio	1.6	2.4	2

Cuadro 11: Rango medio de los clasificadores

2.3. Ejercicio 3:

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.

Para este ejercicio, hemos utilizado el programa R-Commander para realizar los test pertinentes. Para el caso del test de Wilconxon, hemos escodigo los CCR resultantes de los algoritmos MultilayerPerceptron y de IBK, con K=1. En la Figura 12, podremos observar el resultado obtenido introduciendo los CCR en R-Commander y seguidamente ejecutando el test de Wilconxon.

```
    Wilcoxon rank sum test with continuity correction

data:  multilayer_perceptron and IB1
W = 13.5, p-value = 0.9166
alternative hypothesis: true location shift is not equal to 0
```

Cuadro 12: Test de Wilconxon sobre los algoritmos MLP e IB1

Como vemos, el valor de p-value es mayor que el nivel de significación (en R-Commander es 0.05), aceptamos la hipótesis nula, con lo que podemos decir que estos dos algoritmos presentan diferencias significativas entre ambos. En resultado es observable en la Figura 13.

```
    Friedman rank sum test

data:  cbind(multilayer_perceptron, IB1, Naive_bayes)
Friedman chi-squared = 2.2105, df = 2, p-value = 0.3311
```

Cuadro 13: Test de Friedman para los tres algoritmos

Como vemos, estos valores hacen que aceptemos la hipótesis nula diciendo que los tres algoritmos son parecidos.

Con respecto al test de Iman-Davenport, no vamos a utilizar R-commander. Lo que primero hay que hacer, es calcular la siguiente fórmula:

$$F_F = \frac{(N-1)X_F^2}{N(K-1) - X_F^2}$$

Siendo N el número de clases, en nuestro caso 5. Siendo K el número de algoritmos usados(3) y siendo X_F^2 el valor de test de Friedman para los tres algoritmos, siendo en nuestro caso de 1.6. Con lo que el resultado final sería:

$$F_F = \frac{(5-1)1{,}6}{5(3-1) - 1{,}6} = 0{,}7619$$

Como la probabilidad obtenida es mayor que 0.05, rechazaremos la hipótesis nula de que son diferentes y aceptaremos la hipótesis de que no existe diferencia entre ellos.

2.4. Ejercicio 4:

Use el test de Nememyi para comparar todos los 3 métodos.

2.5. Ejercicio 5:

Enuncie las conclusiones del estudio.

A la vista de los resultados, podremos decir que los tres algoritmos no presentan diferencias entre ellos, con lo que podemos decir que daría igual qué algoritmo usáramos que obtendríamos resultados parecidos.

3. Práctica 3: Clasificación avanzada

3.1. Ejercicio 1:

Seleccione un algoritmo de clasificación de los disponibles en Weka y al menos 10 conjuntos de datos.
Para la realización de este ejercicio vamos a utilizar el algoritmo de clasificación J48, ya que es el que hemos utilizado hasta ahora. Además, los conjuntos de datos que hemos elegido son:

- *diabetes.*
- *glass.*
- *iris.*
- *vote.*
- *labor.*
- *weather.*
- *soybean.*
- *contact-lenses.*
- *ionosphere.*
- *segment.*

3.2. Ejercicio 2:

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

Los resultados obtenidos pueden ser observados en la Tabla 14. Como nota, todos estos experimentos los estamos realizando con una opción de test de Cross-Validation con 10 folds.

J48	Diabetes	Glass	Iris	Vote	Labor	Weather	Soybean	Contact-lenses	Ionosphere	Segment
RRSE	93.6293	89.2727	33.6353	35.9085	97.7888	97.6586	38.4134	74.3898	60.4599	30.2115
Correctly Classified Rate	73.8281	66.8224	96	96.3218	73.6842	64.2857	91.5081	83.3333	91.453	95.7333
Incorrectly Classified Rate	26.1719	33.1776	4	3.6782	26.3158	35.7143	8.4919	16.6667	8.547	4.2667

Cuadro 14: Clasificador J48 aplicado a 10 datasets distintos

3.3. Ejercicio 3:

Aplique el método de combinación de clasificadores Bagging a cada uno de los conjuntos y anote los resultados obtenidos.

Para seleccionar el clasificador *bagging* tendremos que irnos al apartado *Classifiers/meta/bagging*. Los resultados son observables en la Tabla 15. como nota, hemos utilizado, como antes, cross-validation para el test, además de que el clasificador está con los parámetros por defecto excepto por el clasificador utilizado, J48.

Bagging	Diabetes	Glass	Iris	Vote	Labor	Weather	Soybean	Contact-lenses	Ionosphere	Segment
RRSE	87.8986	72.4792	35.4791	34.4297	73.985	122.2799	36.0426	76.9675	50.6629	25.1583
Correctly Classified Rate	74.6094	74.2991	94.6667	96.5517	85.9649	35.7143	92.6794	75	92.8775	96.6
Incorrectly Classified Rate	25.3906	25.7009	5.3333	3.4483	14.0351	64.2857	7.3206	25	7.1225	3.4

Cuadro 15: Clasificador Bagging aplicado a 10 datasets distintos

3.4. Ejercicio 4:

Seleccione al menos dos algoritmos de Boosting y aplique estos algoritmos a cada uno de los conjuntos y anote los resultados obtenidos.

Los algoritmos de Boosting elegidos son *AdaBoostM1* y *LogitBoost*. Como antes, mostraremos los resultados de cada uno de estos algoritmos con respecto a sus resultados con diez datasets diferentes. Igualmente, para el algoritmio AdaBoostM1 vamos a utilizar una clasificación basada en J48, en cambio, para el algoritmo LogitBoost vamos a utilizar el clasificador *REPTree*. En las Tablas 16 y 17 podremos observar los resultados obtenidos.

AdaBoostM1	Diabetes	Glass	Iris	Vote	Labor	Weather	Soybean	Contact-lenses	Ionosphere	Segment
RRSE	105.0569	77.7357	45.2448	38.8575	69.473	105.6997	38.6178	87.7266	53.6952	22.9428
Correctly Classified Rate	72.3958	74.2991	93.3333	95.8621	89.4737	71.4286	92.8258	70.8333	93.1624	97.5333
Incorrectly Classified Rate	27.6042	25.7009	6.6667	4.1379	10.5263	28.5714	7.1742	29.1667	6.8376	2.4667

Cuadro 16: Algoritmo AdaBoostM1 aplicado a 10 datasets distintos

LogitBoost	Diabetes	Glass	Iris	Vote	Labor	Weather	Soybean	Contact-lenses	Ionosphere	Segment
RRSE	87.2323	76.9906	36.3912	37.2033	71.3024	104.0377	32.5221	77.9139	55.1882	23.9069
Correctly Classified Rate	74.2188	72.4299	95.3333	95.8621	85.9649	64.2857	93.5578	70.8333	91.453	96.6
Incorrectly Classified Rate	25.7813	27.5701	4.6667	4.1379	14.0351	35.7143	6.4422	29.1667	8.547	3.4

Cuadro 17: Algoritmo LogitBoost aplicado a 10 datasets distintos

3.5. 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 de agrupación con el clasificador base.

Para este ejercicio, compararemos los algoritmos de Boosting y el algoritmo usado para Bagging. Como antes, lo primero que hay que hacer es calcular el valor del test de Friedman. En esta ocasión, el valor de N será de 10, K será de 3 y el valor del test de Friedman es de 0.65, con lo que la probabilidad que obtenemos es: 0.3023.

3.6. Ejercicio 7:

Enuncie las conclusiones del estudio.
Según los resultados obtenidos con el test de Iman-Davenport, los tres algoritmos anteriores no presentan diferencias entre ellos. Con lo que se podría realizar estos algoritmos para conseguir el mismo fin.

4. Práctica 4: Clasificación usando método multiclase

4.1. Ejercicio 1:

Seleccione un algoritmo clasificación de los disponibles en Weka que sea capaz de resolver problemas de más de dos clases y al menos 10 conjuntos de datos de más de 2 clases.

Para resolver este ejercicio, hemos decidido utilizar el algoritmo de clasificación J48. Para realizar los experimentos, hemos escogido los siguientes conjuntos de datos:

- *glass*
- *contact-lenses*
- *segment*
- *iris*
- *vowel*
- *dermatology*
- *vehicle*
- *zoo*
- *autos*
- *lymph*

4.2. Ejercicio 2:

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

En la Tabla 18 se pueden observar los resultados obtenidos al aplicar el algoritmo clasificador anterior.

J48	Glass	Contact-lenses	Segment	Iris	Vowel	Dermatology	Vehicle	Zoo	Autos	Lymph
RRSE	89.2727	74.3898	30.2115	33.6353	59.5369	37.397	77.4887	42.4398	61.6353	86.5138
Correctly Classified Rate	66.8224	83.3333	95.7333	96	81.5152	93.9891	72.4586	92.0792	81.9512	77.027
Incorrectly Classified Rate	33.1776	16.6667	4.2667	4	18.4848	6.0109	27.5414	7.9208	18.0488	22.973

Cuadro 18: Algoritmo J48 aplicado a datasets multiclase

4.3. Ejercicio 3:

Aplique los métodos multiclase ovo, ova y ecoc a cada uno de los conjuntos de datos y anote los resultados obtenidos.

Para aplicar los métodos multiclase OvO(One versus One), OvA(One versus All) y ECOC(Exaustive-correction-code), tenemos que seleccionar el algoritmo de clasificación *MultiClassClassifier*. Una vez elegido, en sus opciones, vamos al apartado method, ahí podremos elegir qué método utilizar. Para este método hemos elegido un clasificador J48 y para las opciones de test, cross-validation con 10 folds. Los resultados son observables en la Tabla 19. Como nota, el parámetro que tomaremos para medir los resultados será *Root Relative Squared Error*.

RRSE	Glass	Contact-lenses	Segment	Iris	Vowel	Dermatology	Vehicle	Zoo	Autos	Lymph
OvO	96.0349	93.4754	86.0888	61.6441	92.2151	85.4185	81.709	91.9113	94.0876	94.9834
OvA	80.6383	86.5236	30.2969	34.9009	61.1218	42.2566	77.4726	36.1588	61.5647	81.2018
ECOC	99.9957	86.5481	88.0507	65.0813	93.8906	88.8466	86.544	94.4172	97.3097	101.1234

Cuadro 19: Métodos multiclase OvO, OvA y ECOC aplicados a 10 conjuntos de datos

4.4. Ejercicio 4:

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.

Realizando el test de Iman-Davenport como lo hemos hecho hasta ahora, obtenemos una probabilidad mayor que 0.05, que hace, como ya sabemos, que rechazemos la hipótesis de que son diferentes y que aceptemos la hipótesis de que no presentan diferencias entre ellos.

4.5. Ejercicio 5:

Enuncie las conclusiones del estudio.

De los resultados obtenidos podemos observar que no existen diferencias visibles entre los tres algoritmos anteriores, con lo que podemos decir que podemos utilizar estos algoritmos para el mismo estudio y no encontraremos diferencias visibles.

5. Práctica 5: Reglas de asociación

5.1. Ejercicio 1:

Seleccione un conjunto de datos de los suministrados con el paquete Weka. Para que se pueda usar el método a priori es necesario que los conjuntos no contengan variables numéricas.

Para este ejercicio, al tener que tener un conjunto de datos de tipo nominal, hemos elegido el conjunto de datos *weather.nominal*.

5.2. Ejercicio 2:

Usando la herramienta Weka y el método a priori estudie el efecto del umbral de soporte mínimo en el número de itemsets seleccionados.

Para la realización de este ejercicio, primero tendremos que seleccionar el conjunto de datos en la sección *Preporcess* de Weka. Seguidamente, nos iremos a la sección *Associate*. Una vez ahí, veremos que por defecto aparece el método a priori. Para estudiar el efecto del umbral de soporte mínimo, tendremos que ir cambiando la configuración de este método, para ello, nos vamos a la configuración del método e iremos cambiando el valor de *LowerBoundMinSupport*. Los resutlados obtenidos pueden ser observados en la Tabla 20.

LowerBoundMinSupport	Nº items L1	Nº items L2	Nº items L3	Nº items L4	Nº Reglas
0.1	12	47	39	6	10
0.2	12	26	4	0	8
0.3	12	9	1	0	3
0.4	6	2	0	0	0
0.5	0	0	0	0	0

Cuadro 20: Método a priori con el conjunto de datos weather.nominal

Como ya sabemos, lo que nos interesa es que el soporte sea lo mayor posible, ya que indica la representación de todos los items sobre el conjunto de datos, es decir, a mayor soporte, mayor representación en el conjunto. Además, también nos interesaría que se generasen reglas, con lo que viendo la Tabla 20, el mayor valor para el soporte en el que se crean reglas, es de 0,3.

5.3. Ejercicio 3:

Usando el soporte mínimo que considere más apropiado según los resultados del ejercicio anterior realice un estudio del efecto del umbral mínimo de confianza en el número de reglas seleccionadas.

En este ejercicio, al igual que antes, vamos a utilizar el método a priori, pero esta vez, en vez de cambiar el valor del parámetro *LowerBoundMinSupport*, vamos

a cambiar el valor del parámtro *minMetric*, el cual corresponde con el umbral mínimo de confianza. Lo que nos interesa en esta sección es que el valor del umbral de confianza sea lo mayor posible, ya que, por ejemplo, si este método genera reglas con una confianza del 0.9, esto quiere decir que las reglas generadas son ciertas en un 90 % de los casos como mínimo. Una vez dicho esto, en la Tabla 21 podremos observar los resultados obtenidos.

Como nota, hemos decidido coger un valor de 0.3 para el parámetro *LowerBoundMinSupport*.

minMetric	Nº Reglas
0.9	3
0.8	5
0.7	6
0.6	10
0.5	10
0.4	10
0.3	10
0.2	10
0.1	10

Cuadro 21: Método a priori con el conjunto de datos weather.nominal cambiando el parámetro minMetric

Como vemos, llega un punto en el que el número de reglas se estabiliza, en nuestro caso, a partir de 0.6 de nivel de confianza ya no se generan nuevas reglas. Además, también podemos observar que a menor nivel de confianza, mayor número de reglas.

6. Práctica 6: Agrupación y evaluación de resultados

6.1. Ejercicio 1:

Seleccione al menos cinco problemas de los disponibles en el paquete de Weka o en el repositorio de la UCI como ejemplos. Use problemas que solo contengan atributos numéricos.

Para la realización de esta práctica, hemos decidido escoger los siguientes conjuntos de datos: glass, ionosphere, poker, sonar e iris. Hemos escogido estos conjuntos al tratarse de conjuntos de datos numéricos, no nominales.

6.2. Ejercicio 2:

Implemente el método de agrupación basado en particionado k-Means.

Para la realización de este ejercicio vamos a utilizar el lenguaje orientado a objetos Python, en el cual, incluiremos algunas bibliotecas que nos ayudarán a realizar la agrupación de patrones basado en el K-Means. Las bibliotecas incluidas pueden ser observadas en el Listing 1.

```
1 import numpy as np
2 import pandas as pd
3 import sklearn
4
5 from sklearn.cluster import KMeans
```

Listing 1: Bibliotecas

Seguidamente, pasamos a declarar el main para inicializar las variables pertinentes, como el número de centroides, el fichero de lectura y la semilla. Puede observarse en el Listing 2.

```
1 if __name__ == '__main__':
2     fichero = "../Datasets/iris.arff"
3     amount_centroides = 5
4     seed = 15
5
6     np.random.seed(seed)
7     atributes, classes = readData(fichero)
8     kmeans, distances, centros = clustering(atributes, amount_centroides)
9     SSE = calculateSSE(distances)
10
11    print("The centroids are: ", centros)
12    print("The SSE is: ", SSE)
```

Listing 2: Main

Como vemos, en la línea 7 del Listing 2, hay que crear una función para leer los datos desde un fichero, con lo que el Listing 3 reflejará cómo hemos solucionado este problema.

```
1 def readData(fichero):
2     data = pd.read_csv(fichero, header=None)
3     atributes = data.values[1:, 0:-1]
4     classes = data.values[1:, -1]
5 
6     return atributes, classes
```

Listing 3: readData function

Seguidamente, en la línea 8 del Listing 2, podemos observar que creamos una función para realizar el clustering de las instancias. En el Listing 4 podemos observar la solución obtenida.

```
1 def clustering(atributes, amount_centroides):
2     centroides = atributes[np.random.choice(atributes.shape[0],
      amount_centroides, replace=False)]
3     kmeans = KMeans(n_clusters = amount_centroides, init = centroides, max_iter
       = 500, n_init = 1, random_state = 0)
4     distances = kmeans.fit_transform(atributes)
5     centros = kmeans.cluster_centers_
6 
7     return kmeans, distances, centros
```

Listing 4: clustering function

Como vemos en el Listing 4, hacemos uso de una función llamada KMeans, que se incluye en la biblioteca de SKLearn, la cual ejecutará el algoritmo de k-medias con los parámetros que podemos observar. Finalmente, calculamos las distancias de las distancias y por último guardamos los centroides creados por el algoritmo.

6.3. Ejercicio 4:

Implemente la medida de evaluación de la calidad de un método de agrupación basada en SSE o en la correlación entre la matriz de incidencia y la de proximidad.

Para la realizar este ejercicio, hemos cogido el Listing 2 y hemos añadido las líneas 9 y 12 con el objetivo de crear una función para calcular la calidad de nuestra agrupación. El resultado obtenido puede ser observado en el Listing 5.

```
1 def calculateSSE(distances):
2     SSE = 0
3     for i in range(distances.shape[0])
4         SSE += pow(distances[i], 2)
5 
6     return SSE
```

Listing 5: calculateSSE function

6.4. Ejercicio 5:

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

1. **Ejecute el algoritmo de particionado y evalúe su rendimiento**

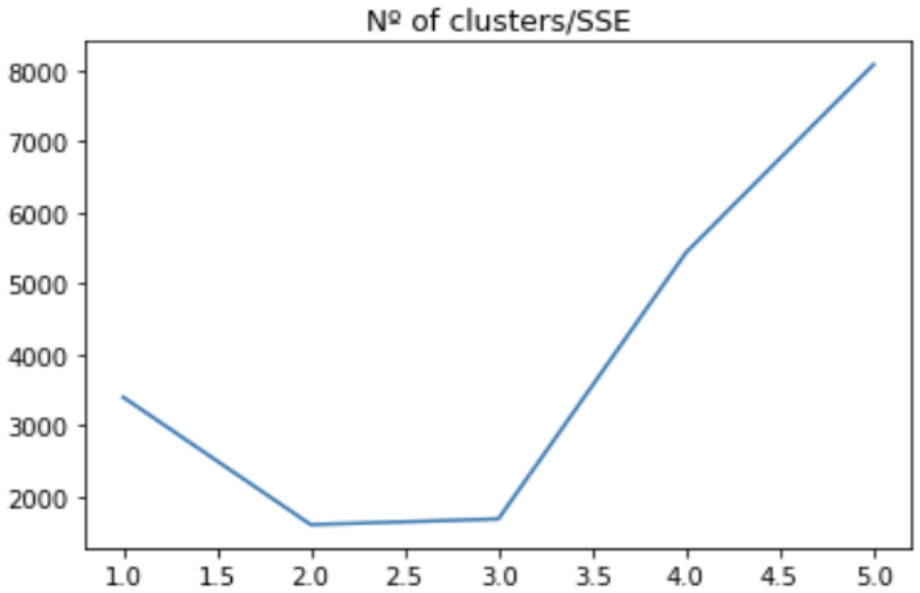
- **Conjunto de datos glass**: Lo que primero vamos a ver son los centroides y el SSE obtenido con un número de cluster igual al número de clases, que en este caso son 5. En la Figura 22 se pueden observar estos resultados.

```
The centroids are:
 [[1.51631704e+00 1.46074074e+01 1.59259259e-01 2.11814815e+00
  7.33348148e+01 1.04074074e-01 8.63925926e+00 9.50370370e-01
  1.44444444e-02]
 [1.51720634e+00 1.31336585e+01 3.49593496e+00 1.39739837e+00
  7.28041463e+01 5.93414634e-01 8.31300813e+00 4.60162602e-02
  6.43902439e-02]
 [1.52103119e+00 1.37976190e+01 3.22047619e+00 1.13738095e+00
  7.18526190e+01 2.51904762e-01 9.53238095e+00 7.11904762e-02
  6.11904762e-02]
 [1.52352895e+00 1.27352632e+01 2.59473684e-01 1.32210526e+00
  7.25357895e+01 2.56315789e-01 1.25184211e+01 1.65789474e-01
  6.94736842e-02]
 [1.51430000e+00 1.26566667e+01 0.00000000e+00 2.41666667e+00
  7.21200000e+01 5.04000000e+00 7.60666667e+00 0.00000000e+00
  0.00000000e+00]]

The SSE is: 20188.24985509597
```

Cuadro 22: Centroides y SSE obtenidos con el conjunto de datos glass

Seguidamente, en la Figura 23 podremos ver la relación existente entre el número de cluster utilizados y el SSE obtenido.



Cuadro 23: Nº de cluster/SSE conjunto de datos glass

Como podemos observar en la Figura 23, se obtiene un codo con un número de cluster igual a 2, con lo que podríamos decir que 2 clusters es el valor óptimo para este conjunto de datos.

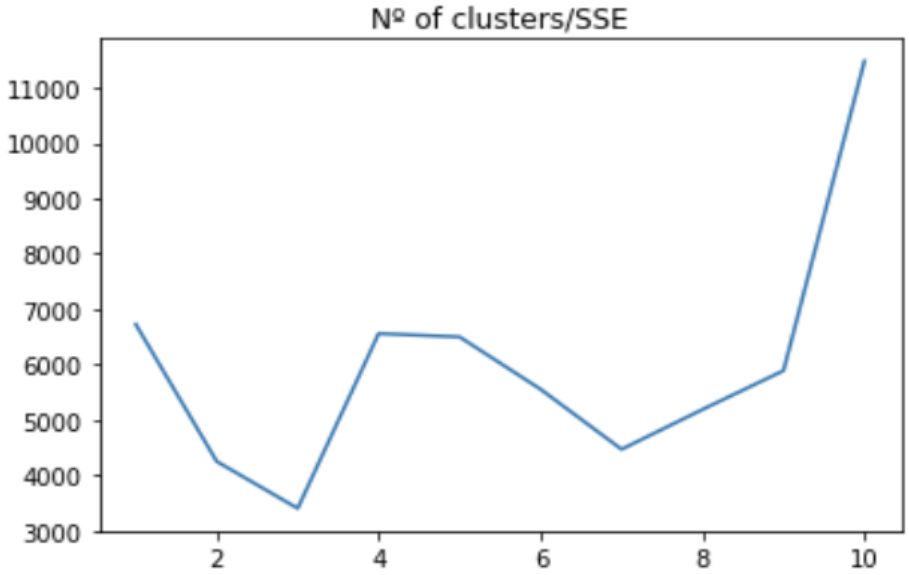
- **Conjunto de datos ionosphere**: Para este conjunto de datos vamos a hacer lo mismo que antes, vamos a mostrar los centroides y SSE obtenidos al utilizar un número de clusters igual al número de clases, que en este caso son 2. Los centroides y SSE obtenidos pueden ser observados en la Figura 24.

```
The centroids are:
 [[ 8.13664596e-01  0.00000000e+00  4.32375466e-01  1.36620745e-01
   2.97504348e-01  2.15006770e-01  2.12654472e-01  2.95393292e-01
   1.53015155e-01  3.72349814e-01  6.73456522e-02  2.86681491e-01
  -1.03470994e-01  2.00435155e-01 -2.30628012e-01  1.46891863e-01
  -9.99267081e-02  1.54167702e-02 -1.17951118e-01 -3.87036025e-02
  -1.72674658e-01 -6.12778261e-02 -8.73747205e-02 -8.45932919e-02
   3.12186957e-02 -1.27153913e-01  3.08146770e-01 -1.42737516e-01
   2.54314286e-02 -5.05184472e-02 -2.18817391e-02  2.23378882e-03
   1.49565217e-04  5.14498137e-02]
 [ 9.57671958e-01  0.00000000e+00  8.17476984e-01 -3.36641270e-02
   8.58329101e-01  3.19465079e-02  8.36042434e-01 -2.79669312e-02
   8.14937778e-01  1.93984656e-02  8.22460476e-01  4.46615344e-02
   8.29325397e-01  5.11952381e-03  8.32410635e-01  8.99518519e-03
   7.89993333e-01 -1.78104762e-02  7.64834709e-01 -9.94428571e-03
   7.69370476e-01  6.91763492e-02  7.45644286e-01 -3.20440741e-02
   7.06079101e-01 -2.11804233e-02  7.41235503e-01 -5.10730159e-03
   6.80037778e-01 -6.98952381e-03  6.71072011e-01 -6.06550265e-03
   6.47704497e-01 -1.45391534e-02]]

The SSE is: 8146.937947668108
```

Cuadro 24: Centroides y SSE obtenidos con el conjunto de datos ionosphere

Ahora, es turno de ver la relación entre SSE y el número de clusters, para ello hemos aumentado el número de clusters para nuestro algoritmo para así poder ver mejor en qué punto empieza a empeorar o mejorar. El resultado se puede observar en la Figura 25.



Cuadro 25: Nº de cluster/SSE conjunto de datos ionosphere

Como vemos en la Figura 24, obtenemos un codo para un número de clusters igual a 3, con lo que podríamos decir que esta cantidad es la adecuada para este conjunto de datos.

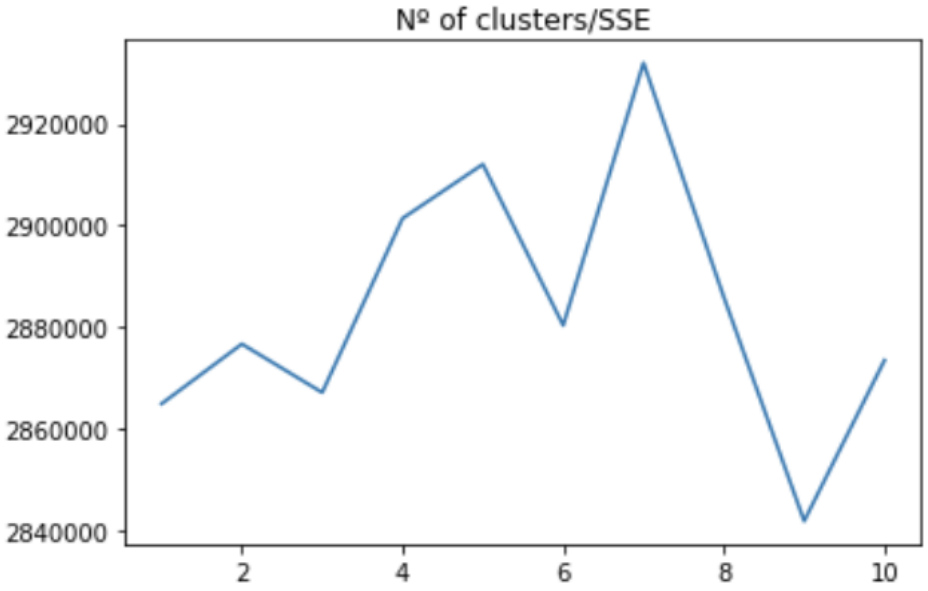
- **Conjunto de datos poker**: Como hasta ahora, lo primero que vamos a hacer es igualar el número de cluster al número de clases, en este caso 10, y vamos a ver los centroides y SSE obtenidos. El resultado es observable en la Figura 26.

```
The centroids are:
 [[ 2.4980484   5.70765027  2.46448087  9.69594067  2.51873536  9.87978142
   2.4726776  10.06986729  2.52966432  3.62685402]
 [ 2.53694581  5.42895036  2.5331565  10.01288367  2.4801061   3.6320576
   2.50208412  9.89541493  2.52785146  9.57862827]
 [ 2.53613666 10.75908892  2.50985545  9.52912834  2.51379763  9.86990802
   2.49058257  6.64038546  2.51379763 10.01664477]
 [ 2.47115772  9.07422348  2.48688987  4.31101251  2.53933038 10.39249697
   2.49415087  3.87656313  2.52722872  4.3182735 ]
 [ 2.51413983  9.2470542   2.49214454 10.3648861   2.52199529  3.97368421
   2.48428908  4.46857816  2.48468185  4.09858602]
 [ 2.51150794  3.40714286  2.47103175  9.68730159  2.49206349  9.02063492
   2.51785714  3.47619048  2.46507937  8.5547619 ]
 [ 2.51059576  4.58856457  2.50579768  3.63894442  2.55137945  9.95641743
   2.47021192  9.58136745  2.49260296  9.76609356]
 [ 2.46376812 10.44261653  2.54876616  4.24245985  2.47042695  5.07638073
   2.5048962  10.57500979  2.4947121   5.44810027]
 [ 2.50413907  3.17301325  2.50248344  4.3294702   2.50620861  4.57615894
   2.5227649   6.50082781  2.45695364  3.94660596]
 [ 2.54302789  8.36733068  2.4625498   4.18884462  2.51155378  4.16414343
   2.50079681  4.03665339  2.47968127 10.41115538]]

The SSE is: 28835463.059769507
```

Cuadro 26: Centroides y SSE obtenidos con el conjunto de datos poker

Ahora, es turno de ver la relación entre número de clusters y SSE, en la Figura 27 se puede ver el resultado.



Cuadro 27: Nº de cluster/SSE conjunto de datos poker

Como podemos observar, obtenemos un codo cuando el número de clusters es igual a 9, con lo que se puede suponer que esta será la mejor configuración para este conjunto de datos.

- **Conjunto de datos sonar**: Para este conjunto de datos haremos como hasta ahora, primero mostraremos los centroides y SSE obtenidos igualando el número de clusters al número de clases, en este caso 2. El resultado es observable en la Figura 28.

```
The centroids are:
 [[0.0281453  0.03724188 0.04110598 0.04795556 0.06292564 0.09633675
  0.11327009 0.12945983 0.17261795 0.1952265  0.21446496 0.22389402
  0.22886068 0.21831197 0.21272051 0.2422094  0.24853333 0.27406752
  0.33290855 0.41340855 0.46875556 0.48994957 0.55077692 0.6162547
  0.65021709 0.70781624 0.7361359  0.77306752 0.74974274 0.6890735
  0.60289402 0.52811795 0.51034786 0.49883761 0.4808094  0.46826154
  0.42753932 0.39873846 0.37864957 0.34855556 0.32957949 0.32111197
  0.28093077 0.24144188 0.2310812  0.18876581 0.13428291 0.10177179
  0.05514957 0.02072051 0.01651111 0.01393419 0.01054957 0.01046154
  0.00909744 0.00771282 0.00746581 0.00766154 0.00809915 0.00666752]
 [0.03047363 0.03997253 0.04733736 0.06152527 0.09098681 0.11515604
  0.13264505 0.14166374 0.18492637 0.22501538 0.26371758 0.28407033
  0.33044835 0.39718352 0.45839121 0.5537     0.63127582 0.6814978
  0.72582967 0.75543846 0.78945165 0.79697912 0.77065824 0.74516813
  0.70783187 0.68964505 0.65846484 0.5923967  0.50364286 0.44188352
  0.37793736 0.32451209 0.29748352 0.28031319 0.27912198 0.2776011
  0.28186484 0.2636956  0.25784945 0.26318681 0.2374022  0.22324066
  0.20232747 0.17888901 0.15371209 0.12445824 0.10724286 0.07811978
  0.04778791 0.02004286 0.0155     0.01275934 0.01091429 0.01155714
  0.00953846 0.00887582 0.00827582 0.00831868 0.00773846 0.0063011 ]]

The SSE is: 890.7396287881397
```

Cuadro 28: Centroides y SSE obtenidos con el conjunto de datos sonar

Ahora, en la Figura 29, podremos observar la relación existente entre el número de clusters y SSE.

Cuadro 29: Nº de cluster/SSE conjunto de datos sonar

Como podemos ver, con un número de clusters inicial de 9 se obtiene un codo donde el valor SSE es mínimo, con lo que podemos suponer que esta es la mejor configuración para este conjunto de datos.

- **Conjunto de datos iris**: Seguidamente, como hasta ahora, vamos a mostrar los centroides y SSE obtenidos igualando el número de clusters inicial al número

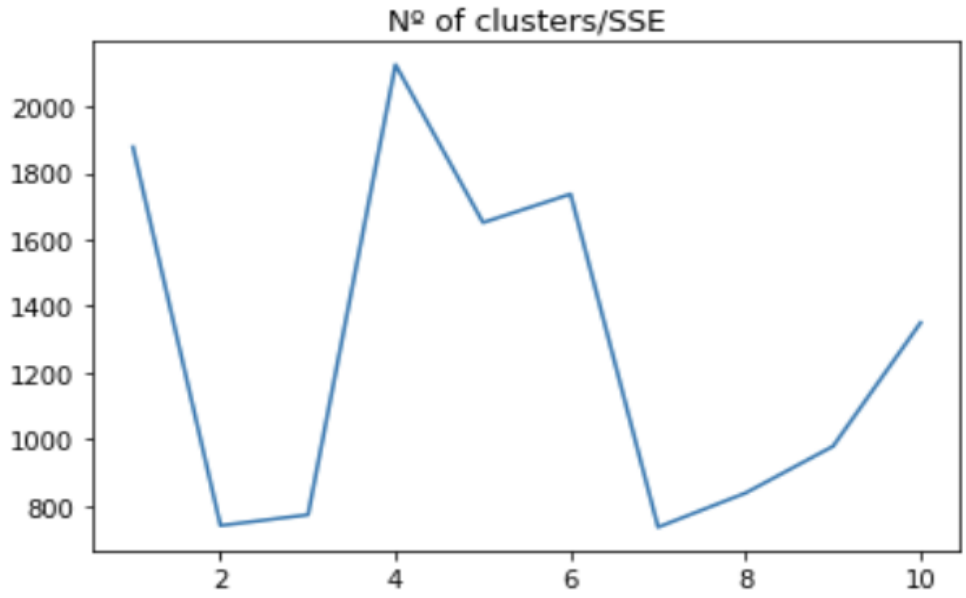
de clases del conjunto de datos, en este caso 4. El resultado es observable en la Figura 30.

```
The centroids are:
 [[5.00408163 3.41632653 1.46530612 0.24489796]
 [6.23658537 2.85853659 4.80731707 1.62195122]
 [5.52962963 2.62222222 3.94074074 1.21851852]
 [6.9125     3.1        5.846875   2.13125   ]]

The SSE is: 4948.108372079718
```

Cuadro 30: Centroides y SSE obtenidos con el conjunto de datos iris

Por último, mostraremos la relación existente entre el número de clusters y el SSE obtenido. En la Figura 31 podemos observar el resultado.



Cuadro 31: Nº de cluster/SSE conjunto de datos iris

Como vemos, obtenemos dos codos cuando el número de clusters es igual a 2 e igual a 7, con lo que podemos decir que estas dos cantidades son las mejores para este conjunto de datos.

7. Bibliografía

1. Weka datasets
2. Temario Moodle
3. Sklearn