UNIVERSIDAD DE CÓRDOBA, ESCUELA POLITÉCNICA SUPERIOR DE CÓRDOBA

PRÁCTICA RESUMEN

Convocatoria de enero (curso académico 2016/2017)

Asignatura: Introducción a la Minería de Datos
4º Grado Ingeniería Informática (Universidad de Córdoba)

2 de enero de 2017

Rafael Ulises Baena Herruzo i32baher@uco.es
DNI: 31012668C

ÍNDICE

Práctica 1: Preprocesamiento y exploración de datos....3
Ejercicio 1.1....3
Ejercicio 1.2....6
Práctica 2: Clasificación básica....9
Ejercicio 2.1....9
Ejercicio 2.2....9
Ejercicio 2.3....10
Ejercicio 2.4....11
Ejercicio 2.5....11
Práctica 3: Clasificación avanzada....12
Ejercicio 3.1....12
Ejercicio 3.2....12
Ejercicio 3.3....12
Ejercicio 3.4....13
Ejercicio 3.5....13
Ejercicio 3.7....13
Práctica 4: Clasificación usando método multiclase....14
Ejercicio 4.1....14
Ejercicio 4.2....14
Ejercicio 4.3....14
Ejercicio 4.4....15
Ejercicio 4.5....16
Práctica 5: Reglas de asociación....17
Ejercicio 5.1....17
Ejercicio 5.2....17
Ejercicio 5.3....18
Práctica 6: Agrupación y evaluación de resultados....19
Ejercicio 6.1....19
Ejercicio 6.2....19
Ejercicio 6.4....20
Ejercicio 6.5....20

Práctica 1: Preprocesamiento y exploración de datos

Ejercicio 1.1

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 la realización de este ejercicio he elegido la base de datos *diabetes.arff*. Tiene un total de **768 instancias** y 9 atributos de tipo nominal exceptuando el atributo de clase. Aplicando el clasificador **J48**, con los parámetros por defecto, a nuestra base de datos sin aplicar ninguno de los filtros, el árbol generado quedaría como se ve en la *Ilustración 1*.

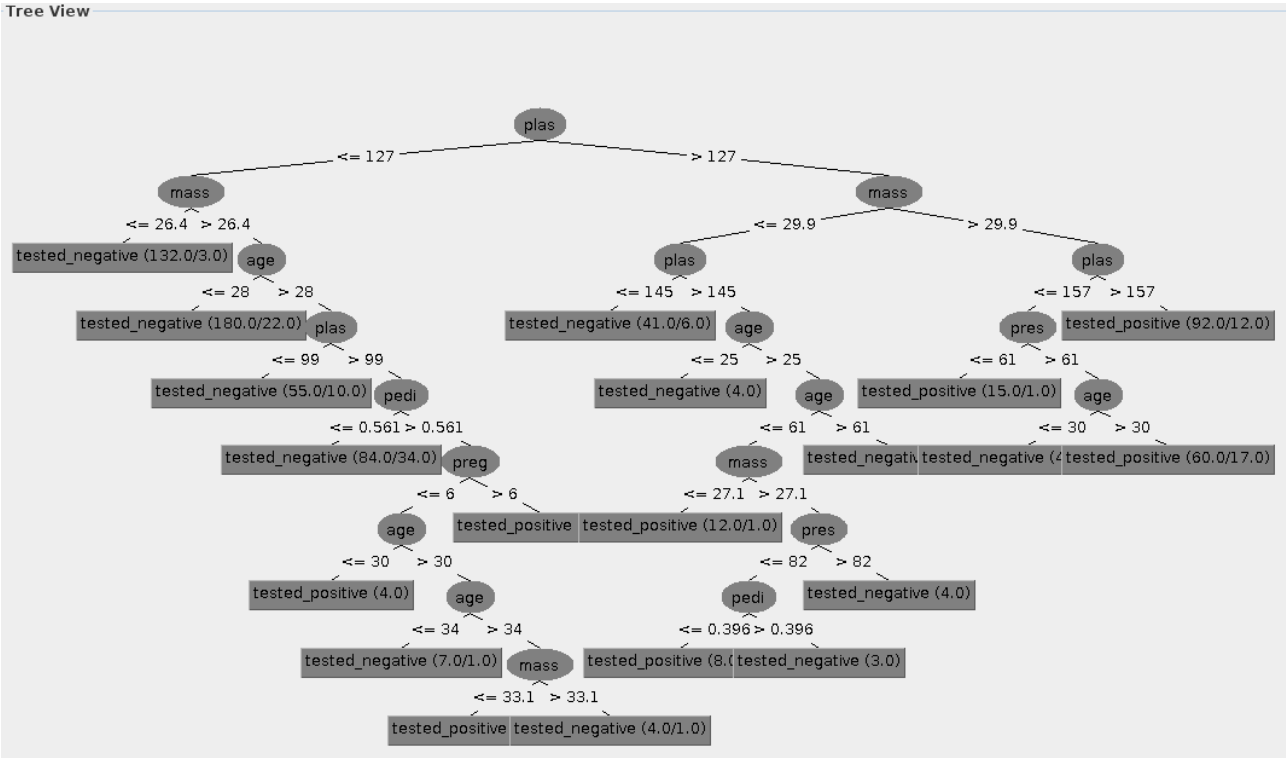

Ilustración 1: Árbol de clasificación J48

El **CCR** (porcentaje de patrones bien clasificados) obtenido es del 73.8281 % y un error de clasificación del **26.1719 %**

Aplicando filtro RandomProjection:

Es un filtro no supervisado aplicado a los atributos de la base de datos. Con este filtro podemos reducir la dimensionalidad de nuestra base de datos, esto diminuye el coste computacional. Vamos a realizar pruebas seleccionando 2, 5 y 7 atributos a obtener sin contar la clase.

- **Filtro RandomProjection con 2 atributos:**

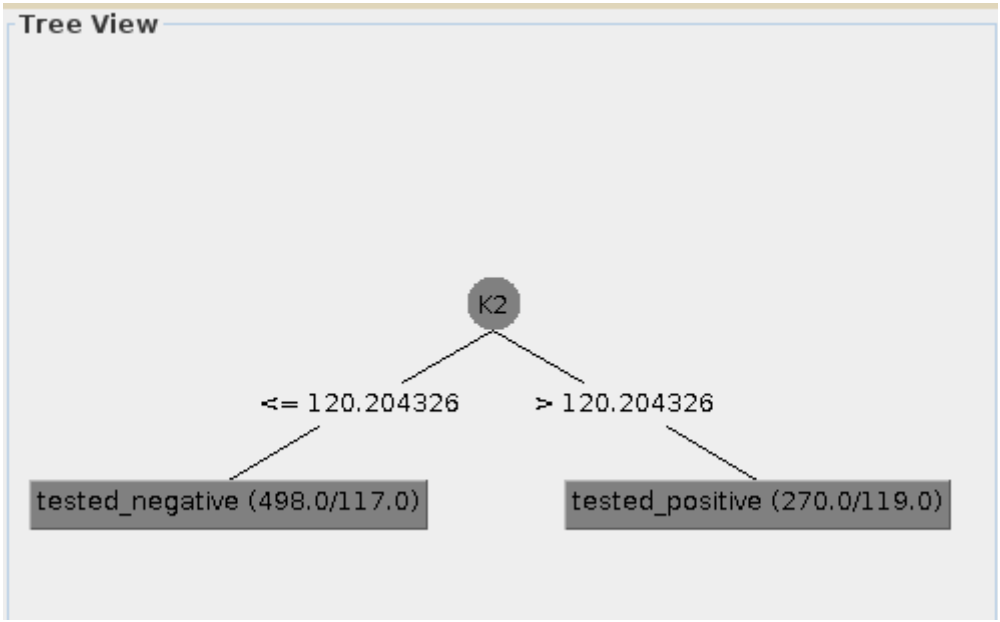

Ilustración 2: Filtro RandomProjection con 2 atributos

- **Filtro RandomProjection con 5 atributos:**

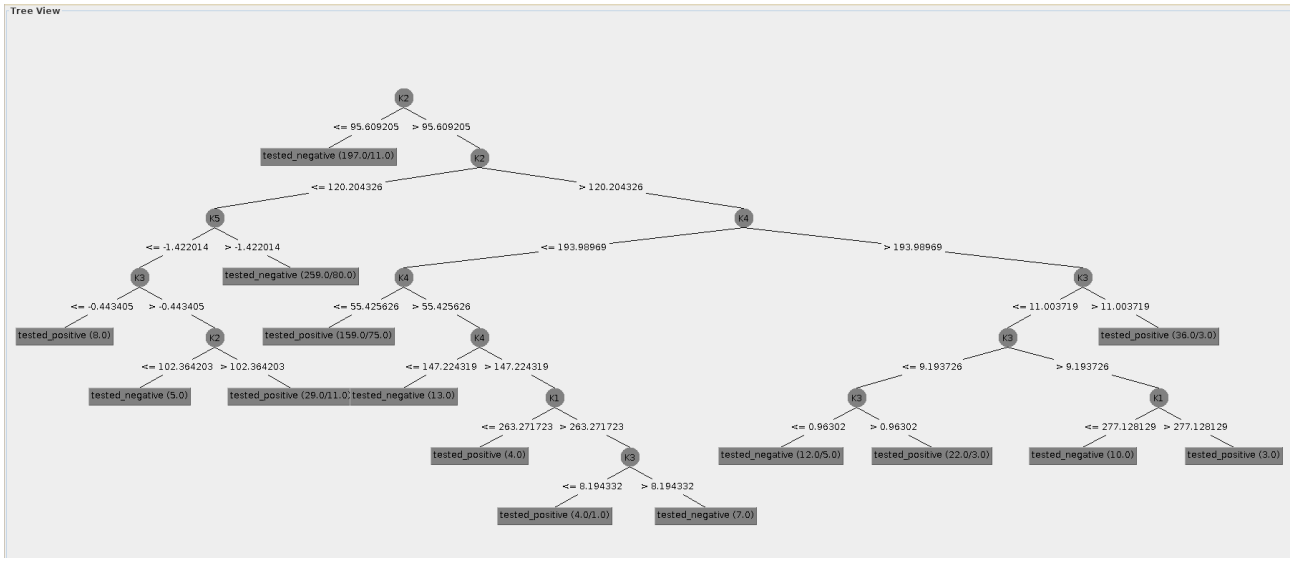

Ilustración 3: Filtro RandomProjection con 2 atributos

- **Filtro RandomProjection con 7 atributos:**

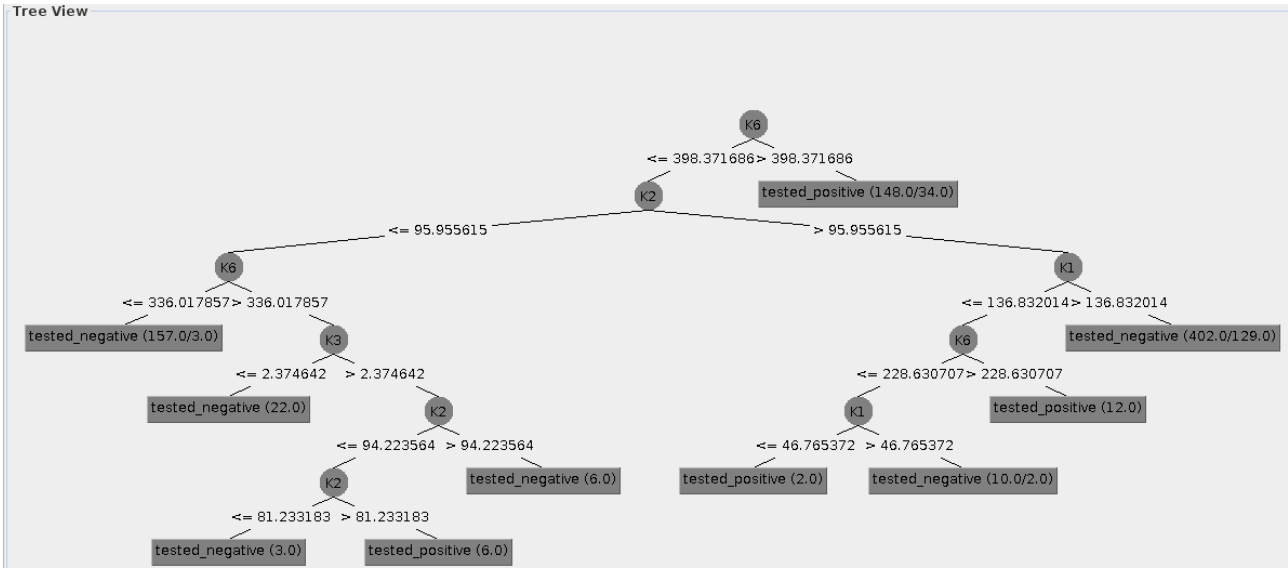

Ilustración 4: Filtro RandomProjection con 7 atributos

RandomProjection	CCR	% incorrectos	RMSE
2 atributos	67.9688	32.0313	0.4548
5 atributos	68.4896	31.5104	0.4434
7 atributos	**75.1302**	**24.8698**	**0.4139**
original	73.8281	26.1719	0.4463

Como se puede observar en la tabla superior, se van obteniendo mejores resultados conforme vamos acercándonos a el número de atributos original. Obteniendo el mejor de todos con un número de atributos igual a 7.

Con un número superior de atributos el error no mejora.

Aplicando filtro RemoveUseless:

Este es un filtro no supervisado de atributos, que elimina los atributos que superan un porcentaje máximo de varianza permitido. El problema con nuestra base de datos es que es de tipo numérico y el filtro solo se aplica a valores nominales. Para solucionarlo hemos aplicado a nuestra base de datos el filtro *NumericToNominal*, el cual transforma los datos numéricos en valores nominales.

Vamos a proceder a probar distintas varianzas y ver como clasifica el método J48 sin aplicar poda, ya que al realizarla deja un solo nodo hoja con la clase mayoritaria. Esto es debido a la transformación de valores numéricos a nominales.

RemoveUseless/ % Varianza	CCR	% incorrectos	RMSE	Atributos
99	59.8958	40.1042	0.5557	9
65	66.0156	33.9844	0.5087	8
30	64.7135	35.2865	0.5211	7
18	**68.6198**	**31.3802**	**0.4833**	**6**
15	62.5	37.5	0.5365	5
6	67.0573	32.9427	0.468	2

Como se puede observar los mejores resultados obtenidos son con una varianza de 18 y un número de atributos igual a 6. Aun así el método visto anteriormente arroja mejores resultados que el actual.

Ejercicio 1.2

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.

Hemos elegido las bases de datos *vote.arff, iris.arff* y *weather.nominal.arff*

Vamos a proceder a comentar que observamos en el histograma de atributos de las diferentes clases:

vote.arff: Contiene un total de 435 instancias distribuidas en dos clases, demócratas (267 instancias) y republicanos (168 instancias). Cada instancia está definida por un total de 17 atributos. En un principio podemos decir que la base de datos está balanceada. Si observamos los histogramas de cada característica respecto a la clase se pueden sacar diversas conclusiones a priori. Los atributos *"adoption-of-the-budget", "physician-fee-freeze" y "education-spending"* clasifican casi a la perfección a las instancias de las dos clases.

Sin aplicar ningún filtro, obtenemos un árbol con un porcentaje de patrones bien clasificados igual al 96.3218 %.

Efecto de la normalización:

Como todos los valores de nuestra base de datos, son nominales, no tendría ningún efecto la normalización.

Si aplicamos un filtro *nominalToBinary* para poder realizar la normalización y ejecutamos el algoritmo J48 obtenemos un acierto del 96.3218% el cual no varía al que había antes de aplicar la normalización.

Efecto estandarización:

Como todos los valores de nuestra base de datos, son nominales, no tendría ningún efecto la estandarización.

Si aplicamos un filtro *nominalToBinary* para poder realizar la estandarización y ejecutamos el algoritmo J48 vemos el mismo porcentaje de acierto que sin este filtro aplicado. Lo único que ha cambiado es que al estandarizar al contrario de lo que pasaba con la normalización, el valor de los atributos si que cambia por lo que el valor de corte difiere de los anteriores que eran 0 o 1.

Efecto análisis de componentes principales:

Ha reducido el número de atributos a 14 ,los datos han pasado de tener valores nominales a valores numéricos y la media de estos datos es 0.

A la hora de realizar la clasificación el **CCR es 90.8046 %** por lo que ha empeorado respecto a los anteriores.

	CCR
Original	96.3218
Normalización	96.3218
Estandarización	96.3218
Análisis de componentes principales	90.8046

Como se ve en la tabla, con esta base de datos no merece la pena el uso de ninguno de los filtros anteriores.

iris.arff: Contiene un total de 150 instancias distribuidas en tres clases, iris-setosa (50 instancias), iris-versicolor (50 instancias) e iris-virginica (50 instancias). Cada instancia está definida por un total de 5 atributos. Podemos decir que la base de datos está totalmente balanceada. Si observamos los histogramas de cada característica respecto a la clase se pueden sacar diversas conclusiones a priori. Los atributos "petallength" y "petalwidth" clasifican casi a la perfección a las instancias de las tres clases.

Sin aplicar ningún filtro, obtenemos un árbol con un porcentaje de patrones bien clasificados igual al 96 %.

Efecto de la normalización:

La normalización no ha causado ningún efecto en el histograma. A discretizado los datos en el rango 0-1. Al aplicar el clasificador obtenemos un acierto del 96 % el cual no varía al que había antes de aplicar la normalización.

Efecto estandarización:

Este filtro lo que hace es que los valores de la base de datos tengan media 0 y desviación típica 1. Al aplicar el J48 vemos el mismo porcentaje de acierto que sin este filtro aplicado.

Efecto análisis de componentes principales:

Ha reducido el número de atributos a 3 y la media de estos datos es 0. A la hora de clasificar obtiene un CCR de 84.6667% el cual a empeorado al de los anteriormente vistos.

	CCR
Original	96
Normalización	96
Estandarización	96
Análisis de componentes principales	84.6667

weather.nominal.arff: Contiene un total de 14 instancias distribuidas en dos clases, yes (9 instancias) y no (5 instancias) . Cada instancia está definida por un total de 5 atributos. Podemos decir que la base de datos está parcialmente balanceada. Si observamos los histogramas de cada característica respecto a la clase no podremos sacar ninguna conclusión a priori.

Efecto de la normalización:

Como todos los valores de nuestra base de datos, son nominales, no tendría ningún efecto la normalización. Al pasarlos a binario y aplicar la normalización tampoco varían los datos. El CCR del algoritmo J48 es de un 50%

Efecto estandarización:

Como todos los valores de nuestra base de datos, son nominales, no tendría ningún efecto la estandarización.

Si aplicamos el filtro nominalToBinary y aplicamos el filtro de estandarización, los datos pasan a tener media 0 y desviación típica 1. Aun así a la hora de clasificar obtenemos el mismo CCR que en el caso anterior.

Efecto análisis de componentes principales:

Ha aumentado el número de atributos a 6 ,los datos han pasado de tener valores nominales a valores numéricos y la media de estos datos es 0.

En este caso el algoritmo J48 obtenemos un CCR del 85.7143 % el cual ha aumentado considerablemente a los anteriormente obtenidos, por lo que en caso de tener que aplicar algún tipo de filtro aplicaríamos este.

	CCR
Original	50
Normalización	50
Estandarización	50
Análisis de componentes principales	85.7143

Práctica 2: Clasificación básica

Ejercicio 2.1

Seleccione 3 clasificadores dentro de los disponibles en Weka y 5 conjuntos de datos. No use combinaciones de modelos.

Los clasificadores seleccionados son MultilayerPerceptron, J48 y SimpleLogistic. Los conjuntos de datos seleccionados son *diabetes*, *iris*, *vote*, *zoo* y *dermatology.*

Ejercicio 2.2

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.

RRSE	MultilayerPerceptron	J48	SimpleLogistic
diabetes	88,4274 (2)	93,6293 (3)	83,1531 (1)
iris	27,3796 (1)	33,6353 (3)	32,7159 (2)
vote	42,6788 (3)	35,9085 (2)	35,168 (1)
zoo	31,7218 (1)	42,4398 (3)	35,1522 (2)
dermatology	27,7154 (2)	37,397 (3)	23,4659 (1)
Rango medio	1,8	2,8	1,4

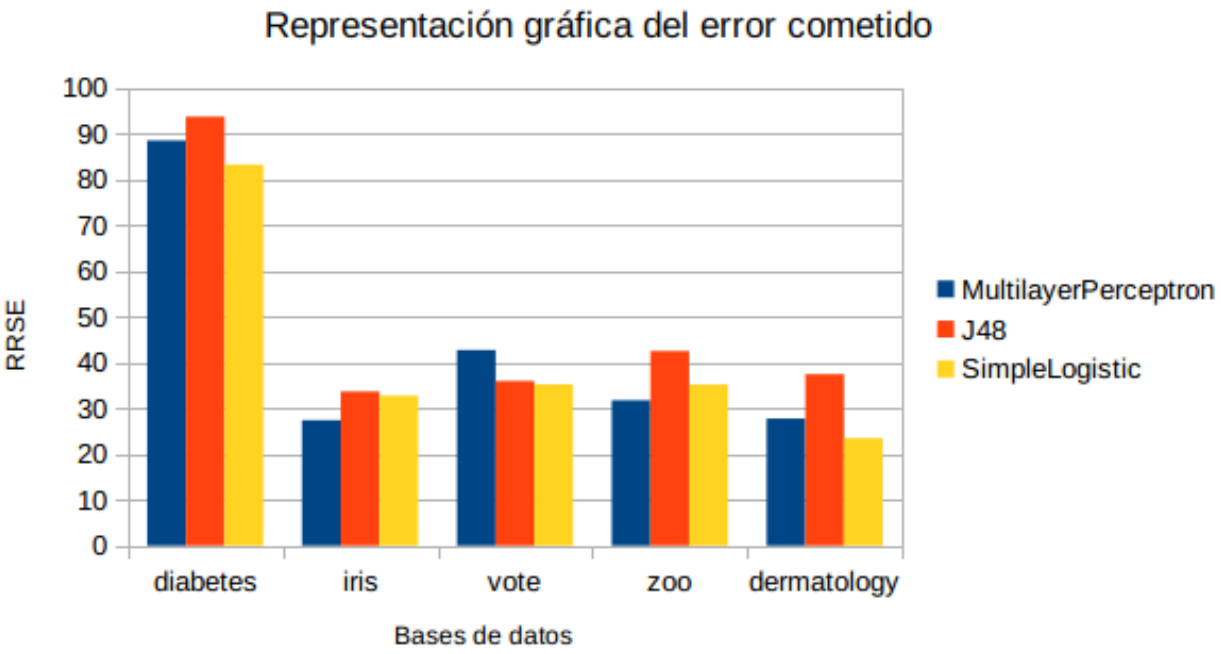

Ilustración 5: Representación gráfica del error cometido

Ejercicio 2.3

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.

Aplicando el test de Wilcoxon para comparar el algoritmo MultilayerPerceptron con el J48, obtenemos un P-value = 0,001953. Como el nivel significancia (0,05) es mayor que el P-value se rechaza la hipótesis nula, por lo que con un 95% de confianza se puede afirmar que los 2 algoritmos son distintos.

Aplicando el rango de Friedman para los 3 clasificadores anteriores se obtiene lo siguiente:

Friedman chi-squared = 5.2, df = 2, p-value = 0.07427

Obtenemos un p-valor de *0,07427*(mayor a 0.05) luego aceptamos la hipótesis nula de que los 3 algoritmos son iguales.

Aplicando el test Iman-Davenport para cada clasificador se obtiene el siguiente resultado:

Corrected Friedman's chi-squared = 4.3333, df1 = 2, df2 = 8, p-value =0.05308

Obtenemos un p-valor de *0,05308* (mayor a 0.05) luego aceptamos la hipótesis nula de que los 3 algoritmos son iguales.

Ejercicio 2.4

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

Aplicando el test de Nememyi a los 3 métodos se obtienen los siguientes resultados:

Critical difference = 1.6873, k = 3, df = 12

Como el rango medio de los 3 algoritmos no iguala o supera la diferencia crítica de 1,6873 se puede concluir que los 3 métodos no son significativamente diferentes.

Ejercicio 2.5

Enuncie las conclusiones del estudio

Tras la realización de los distintos test no paramétricos, se puede afirmar que para las bases de datos escogidas los tres clasificadores son iguales, lo cual no quiere decir que para otro tipo de bases de datos o parámetros se pueda demostrar que son distintos.

También se ha podido demostrar a través del test de Wilcoxon, que los algoritmos MultilayerPerceptron y el J48 son significativamente diferentes entre si, pero que los 3 en conjunto no lo son.

Práctica 3: Clasificación avanzada

Ejercicio 3.1

Seleccione un algoritmo de clasificación de los disponibles en Weka y al menos 10 conjuntos de datos.

Como algoritmo de clasificación se ha escogido el *J48* y como conjuntos de datos "*contact-lenses.arff, supermarket.arff, dermatology.arff, diabetes.arff, glass.arff, ionosphere.arff, iris.arff, soybean.arff, vote.arff, zoo.arff*".

Ejercicio 3.2

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

J48 (Cross-validation 10 folds).

RRSE	diabetes	iris	vote	zoo	dermatology	Contact-lenses	supermarket	Glass	Ionosphere	soybean
J48	93,6293	33,6353	35,9085	42,4398	37,397	74,3898	100	89,2727	60,4599	38,4134

Ejercicio 3.3

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

Bagging con 10 iteraciones y base learner con método J48 (Cross-validation 10 folds).

RRSE	diabetes	iris	vote	zoo	dermatology	Contact-lenses	supermarket	Glass	Ionosphere	soybean
Bagging J48	87,8986	35,4791	34,4297	33,879	28,1753	76,9675	100,001	72,4778	50,6629	36,0426

Ejercicio 3.4

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

RRSE	diabetes	iris	vote	zoo	dermatology	Contact-lenses	supermarket	Glass	Ionosphere	soybean
ADABoostM1	105,0569	45,2448	38,8575	35,2532	32,1116	87,7266	100	77,7357	53,6952	38,6178
MultiBoost	100,2428	37,6572	42,8613	38,1815	25,4424	80,7728	100	78,8071	55,2213	38,761

Ejercicio 3.5

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.

Aplicando el test Iman-Davenport para cada clasificador se obtiene el siguiente resultado:

Corrected Friedman's chi-squared = 0.090909, df1 = 1, df2 = 9, p-value= 0.7699

Obtenemos un p-valor de *0,7699* (mayor a 0.05) luego aceptamos la hipótesis nula de que no hay diferencias significativas entre los dos algoritmos de Boosting**.**

Ejercicio 3.7

Enuncie las conclusiones del estudio

Con las pruebas realizadas se puede concluir que los métodos de boosting escogidos son iguales y que al aplicar el método de combinación de clasificadores Bagging a cada uno de los conjuntos de datos, se han mejorado los errores cometidos excepto en la base de datos *supermarket.arff.*

Práctica 4: Clasificación usando método multiclase

Ejercicio 4.1

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.

Como algoritmo de clasificación se ha escogido el *J48* y como conjuntos de datos "*contact-lenses.arff, autos.arff, dermatology.arff, segment-challenge.arff, glass.arff, anneal.arff, iris.arff, lymph.arff, vowel.arff, zoo.arff*".

Ejercicio 4.2

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

RRSE	segment	iris	vowel	zoo	dermatology	Contact-lenses	autos	Glass	anneal	lymph
J48	30,2115	33,6353	59,5369	42,4398	37,397	74,3898	61,6353	89,2727	25,9118	86,5138

Ejercicio 4.3

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

En *Weka Explorer* se selecciona como método de clasificación *MultiClassClassifier* y en *method* se selecciona *1-against-1* (OVO) , *1-against-all* (OVA) o *exaustive-correction-code* (ECOC).

RRSE	OVO	OVA	ECOC
Segment	86,0888	94,9633	88,0505
iris	61,6441	65,0813	65,0813
Vowel	92,2151	61,1315	93,8906
zoo	91,9113	36,1588	94,4172
dermatology	85,4185	42,2566	88,8466
Contact-lenses	93,4754	86,5236	86,5481
autos	94,0876	61,5647	97,3097
Glass	96,0349	80,6383	99,9925
anneal	120,261	22,0014	123,1543
Lymph	94,9834	81,2018	101,1234

Ejercicio 4.4

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.

Aplicando el test Iman-Davenport para cada clasificador se obtiene el siguiente resultado:

Corrected Friedman's chi-squared = 6.5844, df1 = 2, df2 = 18, p-value =0.007144

Obtenemos un p-valor de *0,007144* (menor a 0.05) luego rechazamos la hipótesis nula de que no hay diferencias significativas entre los 3 métodos multiclase (hay diferencias).

Ahora se procede a aplicar el procedimiento de Wilcoxon para comparar cada método multiclase con el clasificador base.

- **Comparando método ovo vs clasificador base:**

  V = 210, p-value = 1.907e-06 alternative hypothesis: true location is not equal to 0

- **Comparando método ova vs clasificador base:**

  V = 210, p-value = 1.907e-06 alternative hypothesis: true location is not equal to 0

- **Comparando método ecoc vs clasificador base:**

  V = 210, p-value = 1.907e-06 alternative hypothesis: true location is not equal to 0

Obtenemos un p-valor de *1,907e-06* (menor a 0.05) luego rechazamos la hipótesis nula de que no hay diferencias significativas entre los 3 métodos multiclase y el caso base (hay diferencias).

Ahora se procede a aplicar el procedimiento de Wilcoxon para comparar los distintos métodos multiclase entre ellos.

- **Comparando método ovo vs ova:**

  V = 210, p-value = 1.907e-06 alternative hypothesis: true location is not equal to 0

- **Comparando método ovo vs ecoc:**

  V = 210, p-value = 1.907e-06 alternative hypothesis: true location is not equal to 0

- **Comparando método ova vs ococ:**

  V = 210, p-value = 9.556e-05 alternative hypothesis: true location is not equal to 0

Como se puede observar los 3 p-value obtenidos anteriormente son menores a 0.05 por lo que hay diferencias significativas entre los 3 métodos de clasificación, tal y como se había demostrado anteriormente con el test de Iman-Davenport.

Ejercicio 4.5

Enuncie las conclusiones del estudio

Tras haberse realizado las distintas pruebas con los 3 métodos de clasificación, se observa claramente que los mejores resultados se obtienen con el método *1-against-all* (OVA) y esto se ratifica al demostrar de forma estadística que existen diferencias entre ellos.

Práctica 5: Reglas de asociación

Ejercicio 5.1

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.

Se ha seleccionado el conjunto de datos *weather.nominal.arff*.

Ejercicio 5.2

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

Para controlar el umbral de soporte mínimo, en el método a priori, en la pestaña *Associate* de Weka Explorer, se selecciona el método Apriori y se modifica el valor de lowerBoundMinSupport. Obteniendo los resultados de la siguiente tabla:

lowerBoundMinSupport	N.º items L (1)	N.º items L (2)	N.º items L (3)	N.º items L (4)	N.º 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

Interesa que el soporte sea lo mayor posible, ya que este indica la representación que tienen esos items en la base de datos y mientras mayor sea, más representación tendrá. Pero también se necesita que se generen reglas, por lo que el máximo soporte será de 0,3.

Ejercicio 5.3

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 weka, el umbral mínimo de confianza se indica en el parámetro *minMetric* por lo que se van a probar distintas configuraciones de este parámetro usando un soporte de 0,2.

minMetric	N.º reglas
0,9	8
0,8	10
0,7	17
0,6	35
0,5	54
0,4	65
0,3	76
0,2	76
0,1	76

Como se puede observar en la tabla superior, conforme se va disminuyendo el umbral de confianza se van generando un mayor número de reglas, hasta dejar de generar más reglas a partir de un valor de 0,3.

Lo que interesa, es que el valor de confianza sea lo mayor posible, ya que por ejemplo un valor de *minMetric* de 0,9 significa que las 8 reglas que se han generado, son ciertas en el 90% de los casos como mínimo.

```
Apriori
=======

Minimum support: 0.2 (3 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 16

Generated sets of large itemsets:

Size of set of large itemsets L(1): 12

Size of set of large itemsets L(2): 26

Size of set of large itemsets L(3): 4

Best rules found:

 1. outlook=overcast 4 ==> play=yes 4    <conf:(1)> lift:(1.56) lev:(0.1) [1] conv:(1.43)
 2. temperature=cool 4 ==> humidity=normal 4    <conf:(1)> lift:(2) lev:(0.14) [2] conv:(2)
 3. humidity=normal windy=FALSE 4 ==> play=yes 4    <conf:(1)> lift:(1.56) lev:(0.1) [1] conv:(1.43)
 4. outlook=sunny play=no 3 ==> humidity=high 3    <conf:(1)> lift:(2) lev:(0.11) [1] conv:(1.5)
 5. outlook=sunny humidity=high 3 ==> play=no 3    <conf:(1)> lift:(2.8) lev:(0.14) [1] conv:(1.93)
 6. outlook=rainy play=yes 3 ==> windy=FALSE 3    <conf:(1)> lift:(1.75) lev:(0.09) [1] conv:(1.29)
 7. outlook=rainy windy=FALSE 3 ==> play=yes 3    <conf:(1)> lift:(1.56) lev:(0.08) [1] conv:(1.07)
 8. temperature=cool play=yes 3 ==> humidity=normal 3    <conf:(1)> lift:(2) lev:(0.11) [1] conv:(1.5)
```

Ilustración 6: Ejecución método Apriori

Práctica 6: Agrupación y evaluación de resultados

Ejercicio 6.1

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.

Se han seleccionado los siguientes conjuntos de datos: "*vehicle.arff, diabetes.arff, iris.arff, glass.arff y sonar.arff*".

Ejercicio 6.2

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

A continuación, en la *Ilustración 7* se puede observar el código de la función principal de clustering.

```
def clustering(atributos, numCentroides):
      centroides = atributos[np.random.choice(atributos.shape[0], numCentroides, replace = False)]
      kmedias = KMeans(n_clusters = numCentroides, init = centroides, max_iter = 500, n_init = 1 , random_state=0)
      dist = kmedias.fit_transform(atributos)
      centros = kmedias.cluster_centers_

      return kmedias, dist, centros
```

Ilustración 7: Código principal del algoritmo KMeans

En la *Ilustración 8* se pueden observar las librerías que se han necesitado y la función de lectura del conjunto de datos.

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from operator import itemgetter

def lectura_datos(fichero):
      datos = pd.read_csv(fichero, header = None)
      atrib = datos.values[1:,0:-1]
      clases = datos.values[1:,-1]

      return atrib, clases
```

Ilustración 8: Función de lectura de datos

Como función principal del programa se tiene el siguiente código:

```
if __name__ == '__main__':
    fichero = "Bases de datos/sonar.csv"
    nCentroides = 5
    semilla = 15

    np.random.seed(semilla)
    atrib, clases = lectura_datos(fichero)
    kmedias, distancias, centros = clustering(atrib, nCentroides)
    SSE = calcular_SSE(distancias)

    print 'Centroides: '
    print centros

    print 'SSE:'
    print SSE
```

Ilustración 9: Función main

Ejercicio 6.4

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.

El SSE se calcula sumando la distancia de cada patrón al cuadrado, por lo que conforme aumentemos el número de clusters, esta medida disminuará.

El código implementado para la obtención de esta métrica es el siguiente:

```
def calcular_SSE(distancias):
    sum = 0
    for i in range(distancias.shape[0]):
        sum = sum + pow(np.min(distancias[i]),2)
    return sum
```

Ilustración 10: Calculo del SSE

Ejercicio 6.5

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

- Ejecute el algoritmo de particionado y evalúe su rendimiento.

A continuación se va a ejecutar el algoritmo de particionado Kmeans para las 5 bases de datos anteriormente nombradas, y se evaluará su rendimiento haciendo uso de la métrica SSE.

Iris.arff

Primero probamos a ejecutar el algoritmo con un número de clusters igual al número de clases, obtienendo los siguientes resultados:

```
 # Centroides finales:
[[ 5.006       3.418       1.464       0.244     ]
 [ 6.85384615  3.07692308  5.71538462  2.05384615]
 [ 5.88360656  2.74098361  4.38852459  1.43442623]]

 # SSE:
78.945065826
```

Ilustración 11: Ejecución KMeans 3 clusters

Si se compara el SSE obtenido con diferentes números de clusters obtenemos la siguiente gráfica:

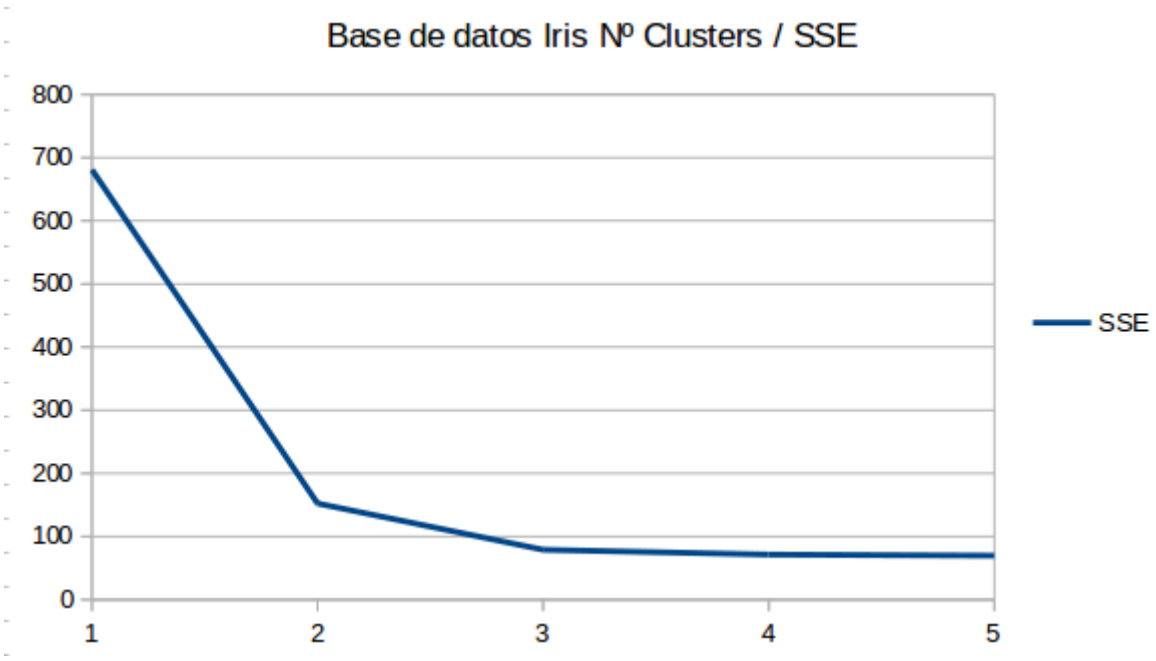

Ilustración 12: Base de datos Iris Nº Clusers / SSE

Como se puede observar, con 3 clusters se obtiene un codo, por lo que se puede suponer que ese es el valor óptimo de centroides del algoritmo.

Vehicle.arff

Primero probamos a ejecutar el algoritmo con un número de clusters igual al número de clases, obtienendo los siguientes resultados:

```
[[ 107.25         55.54166667  103.41666667  190.25         55.79166667
     5.79166667  248.70833333   26.875        27.08333333  168.08333333
   271.29166667  910.54166667  247.16666667   83.625         6.5
    14.20833333  182.66666667  183.625     ]
 [ 103.44019139   52.37320574  102.14354067  202.80861244   62.8277512
    10.13875598  211.74641148   31.39712919   23.94258373  165.10047847
   224.33492823  666.36842105  208.8277512    71.00478469    7.19138756
    15.58373206  188.95215311  197.98086124]
 [  95.           43.87586207   85.66206897  192.04827586   65.46896552
     8.73103448  174.53793103   37.68275862   20.84137931  144.63448276
   198.17931034  461.19310345  168.63448276   69.06206897    6.06206897
    13.70344828  193.91724138  200.32413793]
 [  88.21367521   41.26495726   70.93162393  145.56410256   60.32051282
     7.95726496  143.81623932   46.92094017   18.66880342  140.37393162
   165.47863248  308.05128205  157.62820513   73.59401709    6.10470085
    10.84188034  187.7008547   193.7457265 ]]

 # SSE:
3557051.67844
```

Ilustración 13: Ejecución KMeans 4 clusters

Si se compara el SSE obtenido con diferentes números de clusters obtenemos la siguiente gráfica:

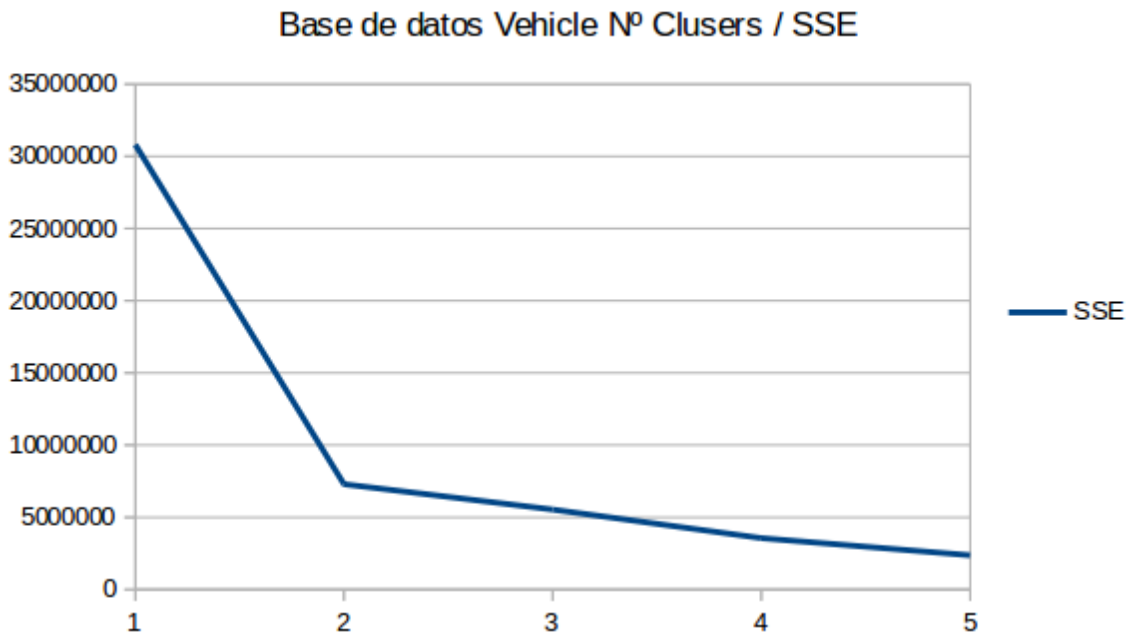

Ilustración 14: Base de datos Vehicle Nº Clusers / SSE

Como se puede observar, con 2 clusters se obtiene un codo, por lo que se puede suponer que ese es el valor óptimo de centroides del algoritmo.

Diabetes.arff

Primero probamos a ejecutar el algoritmo con un número de clusters igual al número de clases, obtienendo los siguientes resultados:

```
 # Centroides finales:
[[   3.88391376  115.26699834   68.09784411   17.6185738
32.21227197
    31.17363184    0.43757048   33.11442786]
 [   3.7030303   141.46060606   72.78787879   31.2
253.70909091
    34.98545455    0.59724848   33.7030303 ]]

 # SSE:
5142376.45598
```

Ilustración 15: Ejecución KMeans 2 clusters

Si se compara el SSE obtenido con diferentes números de clusters obtenemos la siguiente gráfica:

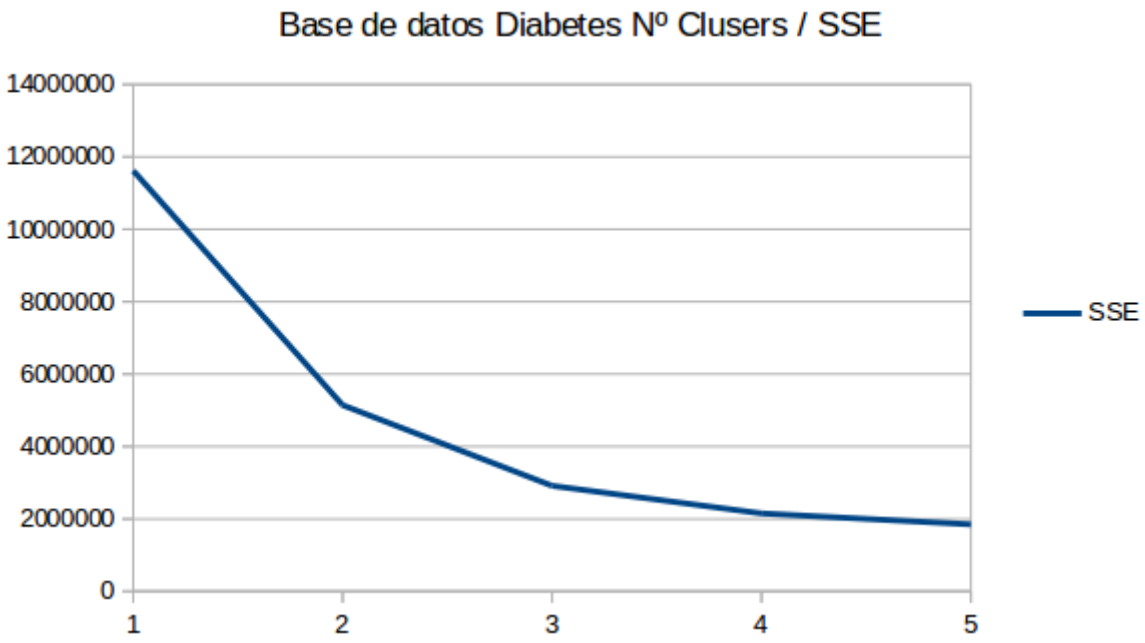

Ilustración 16: Base de datos Diabetes Nº Clusers / SSE

Como se puede observar, con 3 clusters se obtiene un codo, por lo que se puede suponer que ese es el valor óptimo de centroides del algoritmo.

Glass.arff

Primero probamos a ejecutar el algoritmo con un número de clusters igual al número de clases, obtienendo los siguientes resultados:

```
   7.16200000e+01   1.23200000e+00   6.30600000e+00   1.00400000e+00
   2.40000000e-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]
[  1.52184000e+00   1.38865517e+01   3.56586207e+00   9.31379310e-01
   7.16672414e+01   1.89655172e-01   9.54965517e+00   8.55172414e-02
   4.62068966e-02]
[  1.51883400e+00   1.34573333e+01   2.27933333e+00   1.52000000e+00
   7.26286667e+01   3.62000000e-01   9.47533333e+00   8.46666667e-02
   8.80000000e-02]]

 # SSE:
343.387654598
```

Ilustración 17: Ejecución KMeans 7 clusters

Si se compara el SSE obtenido con diferentes números de clusters obtenemos la siguiente gráfica:

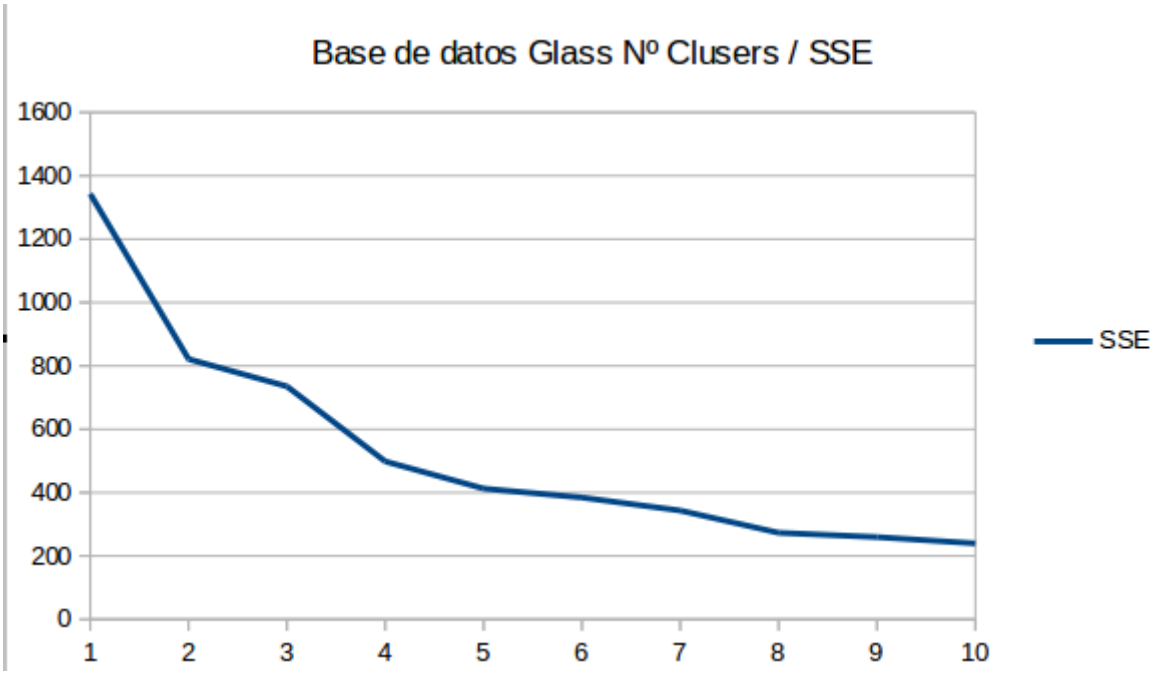

Ilustración 18: Base de datos Glass Nº Clusers / SSE

Como se puede observar, con 8 clusters se obtiene un codo, por lo que se puede suponer que ese es el valor óptimo de centroides del algoritmo.

Sonar.arff

Primero probamos a ejecutar el algoritmo con un número de clusters igual al número de clases, obtienendo los siguientes resultados:

```
  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 ]]

 # SSE:
280.867772917
```

Ilustración 19: Ejecución KMeans 2 clusters

Si se compara el SSE obtenido con diferentes números de clusters obtenemos la siguiente gráfica:

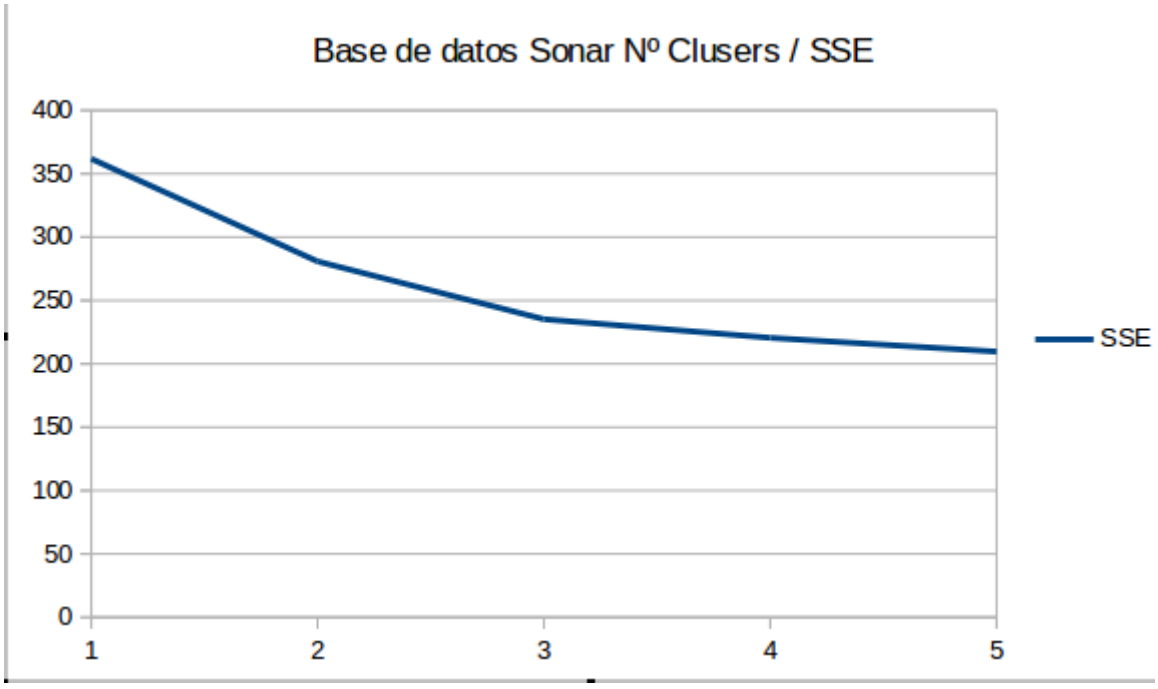

Ilustración 20: Base de datos Sonar Nº Clusers / SSE

Como se puede observar, con 3 clusters se obtiene un codo, por lo que se puede suponer que ese es el valor óptimo de centroides del algoritmo.