**UNIVERSIDAD DE CÓRDOBA**
**Escuela Politécnica Superior**
Grado en Ingeniería Informática
Introducción a la Minería de Datos
Curso académico 2016-2017
Víctor Monserrat Villatoro

# 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 el conjunto de datos 'weather.arff' de la UCI MLR, usando como clasificador el método J48 sin el uso de filtros y como método para obtener el error la validación cruzada de 10 particiones, obtenemos los siguientes resultados de clasificación.

```
Correctly Classified Instances           9               64.2857 %
Incorrectly Classified Instances         5               35.7143 %

a b   <-- classified as
7 2 | a = yes
3 2 | b = no
```

- Filtro RandomProjection.

  Es un filtro de atributo, no supervisado. Reduce la cantidad de atributos del conjunto de datos proyectando sobre una dimensionalidad menor. Además, aplica un filtro binario para la transformación previa de cualquier atributo nominal del conjunto dado.

  Con el uso de este filtro obtendremos modelos más sencillos y el procesamiento será más rápido pero obtendremos peor resultado en la clasificación ya que perdemos información para el entrenamiento.

```
Correctly Classified Instances           7               50      %
Incorrectly Classified Instances         7               50      %

a b   <-- classified as
6 3 | a = yes
4 1 | b = no
```

  Podemos observar que el error de clasificación ha aumentado considerablemente tras aplicar este filtro.

- Filtro RemoveUseless.

  Es un filtro de atributo, no supervisado. Elimina atributos del conjunto de datos que no tienen varían demasiado. Esta variación máxima la podemos indicar a través de un parámetro en tanto por ciento.

  Con el uso de este filtro obtendremos modelos más sencillos y el procesamiento será más rápido sin perder demasiada precisión ya que solo eliminará los atributos que no tienen suficiente utilidad para generar el modelo.

```
Correctly Classified Instances           9               64.2857 %
Incorrectly Classified Instances         5               35.7143 %

a b   <-- classified as
7 2 | a = yes
3 2 | b = no
```

  Tras aplicar el filtro hemos observado que no se ha eliminado ninguno de los atributos para el conjunto de datos usado por lo que el resultado de la clasificación es idéntico.

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.

- Normalización.

  Es un filtro de atributo, no supervisado. Normaliza todos los datos del conjunto de datos, es decir, los reemplaza por valores entre 0 y 1 haciendo más fácil su interpretación y modelado.

```
Correctly Classified Instances           9               64.2857 %
Incorrectly Classified Instances         5               35.7143 %

a b   <-- classified as
7 2 | a = yes
3 2 | b = no
```

Los resultados de la clasificación tras usar el filtro no varían.

- Estandarización.

  Es un filtro de atributo, no supervisado. Estandariza todos los datos del conjunto de datos, es decir, los reemplaza por valores que den media igual a 0 y varianza igual a 1. Esto permite representarlos de forma normal y gaussiana, lo que facilita los cálculos.

```
Correctly Classified Instances           9               64.2857 %
Incorrectly Classified Instances         5               35.7143 %

a b   <-- classified as
7 2 | a = yes
3 2 | b = no
```

  Los resultados de la clasificación tras usar el filtro no varían.

- Análisis en componentes principales.

  Es un filtro de atributo, no supervisado. Analiza y estudia los principales componentes del conjunto de datos para representar los atributos como una regresión lineal en base a los más importantes del conjunto dado. Este número de atributos será especificado al filtro. Además, normaliza los datos.

```
Correctly Classified Instances          10               71.4286 %
Incorrectly Classified Instances         4               28.5714 %

a b   <-- classified as
7 2 | a = yes
2 3 | b = no
```

  Los resultados de la clasificación han mejorado reduciendo la cantidad de atributos en uno para el conjunto de datos, se ha clasificado bien un patrón más que anteriormente.

# 2. Práctica 2: Clasificación básica.

## 2.1. Ejercicio 1.

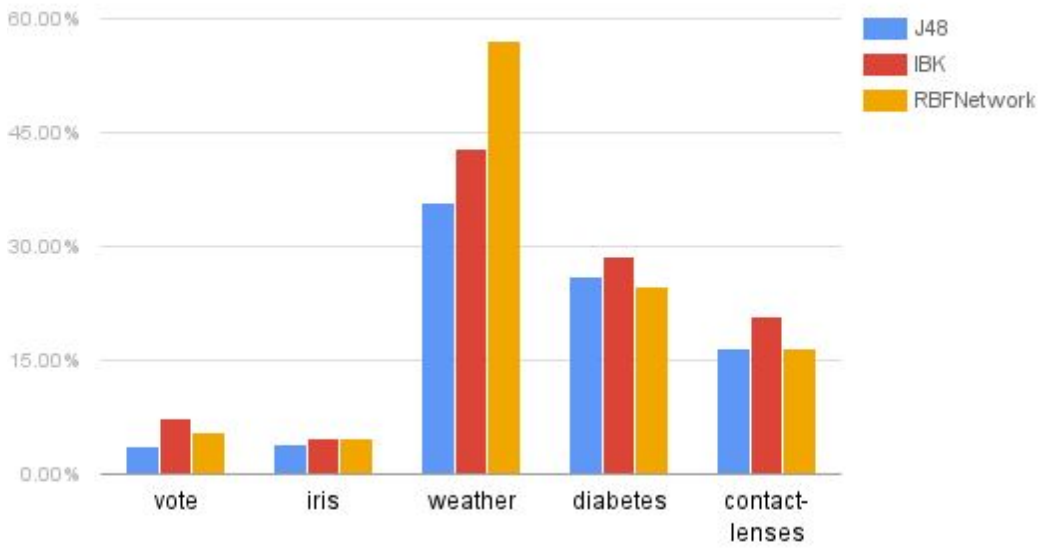
Seleccione 3 clasificadores dentro de los disponibles en Weka y 5 conjuntos de datos. No use combinaciones de modelos.

Usaremos clasificadores de distinto tipo, que serán el ya usado J48 (árbol de decisión), IBK (lazy learning) y RBFNetwork (red neuronal). En el caso de IBK, el número de clusters será igual al número de clases del conjunto de datos.

Los conjuntos de datos usados serán 'vote.arff', 'iris.arff', 'weather.arff', 'diabetes.arff' y 'contact-lenses.arff'.

## 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.




Ver de forma interactiva.

## 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.

Vamos a usar el test de Wilcoxon para comparar J48 (árbol de decisión) y RBFNetwork (red neuronal) sobre los 5 problemas anteriores porque son los dos clasificadores que normalmente obtienen mejores clasificaciones.

Para ello, usaremos una aplicación web que nos devuelve el resultado del test (http://calculator.vhex.net/calculator/statistics/wilcoxon-test), a través de la siguiente fórmula:

$$Z = \frac{\mathcal{P} - \frac{n(n+1)}{4}}{\sqrt{\frac{n(n+1)(2n+1)}{24}}}$$

El resultado obtenido ha sido -0.417786. Sin embargo, el tamaño de N (5) no es suficientemente grande para comparar estos dos algoritmos por lo que no es posible rechazar ni aceptar la hipótesis nula entre los dos clasificadores.

Ahora obtendremos el rango de Friedman para los 3 clasificadores con otra aplicación web (http://vassarstats.net/fried3.html) con k = 3 (clasificadores) y N = 5 (problemas), usando la siguiente fórmula:

$$\left[\frac{12}{nk(k+1)}\sum_{i=1}^{k} R_i^2\right] - 3n(k+1)$$

Hemos obtenido como resultados $csq_r$ = 4.9, df = 2 y P = 0.0863. Tendremos que tratar el valor de P como una aproximación imperfecta porque el valor de N es menor a 7. Como 4.605 ($\alpha$ = 0.1) < 4.9 < 5.991 ($\alpha$ = 0.05) podemos rechazar la hipótesis nula y afirmar que hay diferencias significativas con un 95% de confianza. Los rangos obtenidos han sido 1.3, 2.7 y 2 respectivamente.

Ahora aplicaremos el test de Iman-Davenport sobre los tres clasificadores aplicando la siguiente fórmula (N = 5, k = 3):

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2}$$

Se obtiene un valor de 3.84, por lo que se rechaza con un 90% de confianza la hipótesis nula. Así, que podemos decir que hay diferencias significativas entre los tres algoritmos.

2.4. Ejercicio 4.

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

Antes de nada calculamos el error estándar siguiendo la siguiente fórmula, siendo n = 5 y k = 3:

$$SE = \sqrt{\frac{k(n+1)}{12}}$$

El error estándar obtenido es de 1.225. Ahora debemos calcular la mínima distancia significativa (MSD), que es igual al producto del error estándar y q (igual a 3.31 por ser k = 3). Por lo tanto, MSD = 4.054. Como las diferencias de los rangos (1.3, 2.7 y 2) no exceden ninguna del MSD podemos decir que tienen un comportamiento similar los tres.

2.5. Ejercicio 5.
Enuncie las conclusiones del estudio.

Debido al escaso número de problemas utilizados (N bajo) el estudio no es todo lo fiable que nos gustaría y no podemos ser fieles a sus resultados.

Al realizar el test de Wilcoxon no hemos sacado nada en claro por este motivo. Con los test de Friedman y de Iman-Davenport hemos llegado a la misma conclusión, existen diferencias significativas entre los tres clasificadores. Sin embargo, obtenemos luego en el test de Nemenyi que los tres clasificadores tienen un comportamiento similar.

La conclusión final es que deberíamos de realizar el estudio con un número mayor de problemas, es decir, aumentar N y poder así realizar un estudio que nos aporte unos resultados más definitivos.

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.

Como algoritmo de clasificación usaremos el J48 y como conjuntos de datos usaremos ‘contactlenses.arff’, ‘diabetes.arff’, ‘glass.arff’, ‘ionosphere.arff’, ‘iris.arff’, ‘labor.arff’, ‘segmenttest.arff’, ‘soybean.arff’, ‘vote.arff’ y ‘weather.arff’.

3.2. Ejercicio 2.

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

| Conjunto | Correctly Classified Instances |
|---|---|
| Contact-lenses | 83.3333% |
| Diabetes | 73.8281% |
| Glass | 66.8224% |
| Ionosphere | 91.453% |
| Iris | 96% |
| Labor | 73.6842% |
| Segment-test | 93.4568% |
| Soybean | 91.5081% |
| Vote | 96.3218% |
| Weather | 64.2857% |

3.3. Ejercicio 3.

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

Para ello debemos configurar el clasificador como combinación de Bagging y de J48 como se muestra en la siguiente imagen.

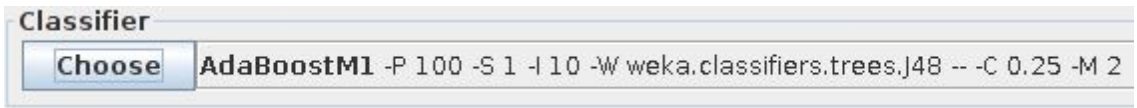

Una vez configurado, se repite el proceso anterior, ahora usando esta combinación de clasificadores y se obtienen los siguientes resultados:

| Conjunto | Correctly Classified Instances |
|---|---|
| Contact-lenses | 75% |
| Diabetes | 74.6094% |
| Glass | 75.7009% |
| Ionosphere | 92.8775% |
| Iris | 94% |
| Labor | 82.4561% |
| Segment-test | 94.6914% |
| Soybean | 92.6794% |
| Vote | 96.092% |
| Weather | 42.8571% |

### 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.

Aplicaremos los algoritmos de Boosting AdaBoostM1 y MultiBoostAB. Tanto para AdaBoostM1 como para MultiBoostAB tendremos que especificar qué algoritmo queremos realizar, en nuestro caso el J48 como ya hemos dicho antes.





Una vez configurados ambos algoritmos podemos proceder a ejecutarlos. Los resultados obtenidos son los que se muestran en la siguiente tabla:

| Conjunto | CCI (AdaBoostM1) | CCI (MultiBoostAB) |
|---|---|---|
| Contact-lenses | 70.8333% | 79.1667% |
| Diabetes | 72.3958% | 74.8698% |
| Glass | 74.2991% | 74.7664% |
| Ionosphere | 93.1624% | 92.3077% |
| Iris | 93.3333% | 95.3333% |
| Labor | 89.4737% | 80.7018% |
| Segment-test | 95.9259% | 95.9259% |
| Soybean | 92.8258% | 92.3865% |
| Vote | 95.8621% | 95.4023% |
| Weather | 71.4286% | 64.2857% |

### 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 realizar el test de Iman-Davenport primero debemos calcular el valor de $csq_r$ a tráves del test de Friedman con una aplicación web (http://vassarstats.net/fried4.html) con k = 4 (clasificadores) y N = 10 (problemas), usando la siguiente fórmula:

$$\left[\frac{12}{nk(k+1)}\sum_{i=1}^{k}R_i^2\right] - 3n(k+1)$$

Hemos obtenido como resultado $csq_r = 0.99$.

Ahora aplicaremos el test de Iman-Davenport sobre los cuatro clasificadores aplicando la siguiente fórmula (N = 10, k = 4):

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2}$$

Se obtiene un valor de 0.3071, por lo que no se puede rechazar con la hipótesis nula. Así, que no podemos decir que haya diferencias significativas entre los cuatro algoritmos. Por lo tanto, no se aplicará el test de Wilcoxon ya que no existen diferencias significativas.

### 3.6. Ejercicio 7.

Enuncie las conclusiones del estudio.

La conclusión del estudio realizado sería que no se puede confirmar que existan diferencias significativas entre J48, J48 + bagging, AdaBoostM1 y MultiBoostAB para los diez problemas que hemos utilizado.

## 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.

Como algoritmo de clasificación usaremos el J48 y como conjuntos de datos usaremos ‘contactlenses.arff’, ‘diabetes.arff’, ‘glass.arff’, ‘ionosphere.arff’, ‘iris.arff’, ‘labor.arff’, ‘segmenttest.arff’, ‘soybean.arff’, ‘vote.arff’ y ‘weather.arff’.

4.2. Ejercicio 2.

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

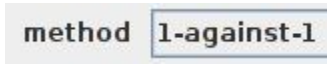
| Conjunto | Correctly Classified Instances |
|---|---|
| Autos | 81.9512% |
| Contact-lenses | 83.3333% |
| Dermatology | 93.9891% |
| Flags | 59.2784% |
| Glass | 66.8224% |
| Iris | 96% |
| Page-blocks | 96.8756 |
| Segment | 96.9264% |
| Segment-test | 93.4568% |
| Zoo | 92.0792% |

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.

Los métodos multiclase que vamos a utilizar son ovo, ova y ecoc:


- One-vs-one (OVO): entrena un clasificador para cada par de clases.



- One-vs-all (OVA): aprende a discriminar un tipo de clase del resto.


- Exhaustive correction code (ECOC): combina diversos clasificadores binarios.


| Conjunto | CCI (OVO) | CCI (OVA) | CCI (ECOC) |
|---|---|---|---|
| Autos | 70.2439% | 82.9268% | 81.4634% |
| Contact-lenses | 70.8333% | 70.8333% | 70.8333% |
| Dermatology | 96.4481% | 91.5301% | 96.9945% |
| Flags | 58.7629% | 59.7938% | 58.7629% |
| Glass | 71.9626% | 70.0935% | 75.2336% |
| Iris | 96% | 94% | 94% |
| Page-blocks | 97.1862% | 97.0583% | 97.2958% |
| Segment | 96.9264% | 95.8874% | 98.0952% |
| Segment-test | 93.5802% | 91.8519% | 95.679% |
| Zoo | 91.0891% | 92.0792% | 96.0396% |

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.

Para realizar el test de Iman-Davenport primero debemos calcular el valor de $csq_r$ a tráves del test de Friedman con una aplicación web (http://vassarstats.net/fried4.html) con k = 4 (clasificadores) y N = 10 (problemas), usando la siguiente fórmula:

$$\left[\frac{12}{nk(k+1)}\sum_{i=1}^{k}R_i^2\right]-3n(k+1)$$

Hemos obtenido como resultado $csq_r = 3.27$.

Ahora aplicaremos el test de Iman-Davenport sobre los cuatro clasificadores aplicando la siguiente fórmula (N = 10, k = 4):

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2}$$

Se obtiene un valor de 1.1010, por lo que no se puede rechazar con la hipótesis nula con un 99% de confianza. Así, que no podemos decir que haya diferencias significativas entre los cuatro algoritmos. Por lo tanto, no se aplicará el test de Wilcoxon ya que no existen diferencias significativas.

4.5. Ejercicio 5.

Enuncie las conclusiones del estudio.

La conclusión del estudio realizado sería que no se puede confirmar que existan diferencias significativas entre J48 y ningún método multiclase utilizado para los diez problemas que hemos elegido.

# 5. Práctica 5: Reglas de asociación.

## 5.1. Ejercicio 1.

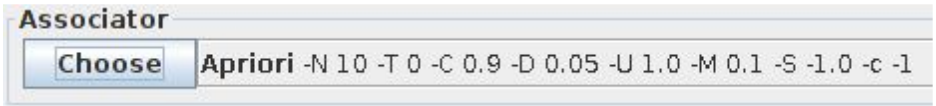
Seleccione un conjunto de datos de los suministrados con el paquete Weka. Para que se pueda usar el método a priori es necesario que los conjuntos no contengan variables numéricas.

Como conjunto de datos usaremos 'contact-lenses.arff' porque no contiene variables numéricas.

## 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 el uso del método a priori en Weka debemos de utilizar la pestaña 'Associate' y seleccionar el método a priori como método de asociación.




| Umbral de soporte mínimo | Número de itemsets |
|---|---|
| 0.1 | 4 (12, 49, 84 y 26) |
| 0.2 | 3 (11, 21 y 6) |
| 0.3 | 2 (10 y 5) |
| 0.4 | 2 (7 y 1) |
| 0.5 | 2 (7 y 1) |
| 0.6 | 0 |

Podemos observar que a mayor umbral de soporte mínimo, menores reglas se crean, ya que estamos obligando a que se cojan reglas con más ítems, es decir, de un nivel superior. Es por esto que a partir de 0.6 no existen itemsets, ya que no existe ninguna regla que tenga un 60% de los ítems del conjunto de ítems.

Para el estudio se ha variado el parámetro `lowerBoundMinSupport`.

## 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.

Usando un soporte mínimo igual a 0.1 para considerar casi todas las reglas, ya que no se trata de un conjunto de datos extenso, vamos a proceder a estudiar el efecto del umbral mínimo de confianza en el número de reglas seleccionadas.

| Umbral mínimo de confianza | Número de reglas |
|---|---|
| 0.4 | 479 |
| 0.5 | 421 |
| 0.6 | 146 |
| 0.7 | 108 |
| 0.8 | 89 |
| 0.9 | 83 |

Podemos observar que a mayor umbral mínimo de confianza, menos reglas se crean, ya que estamos obligando a que se cojan reglas con mayor confianza y por lo tanto más escasas.

Para el estudio se ha variado el parámetro 'minMetric' siendo la confianza el tipo de la métrica.

# 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.

Como conjuntos de datos usaremos ‘diabetes.arff’, ‘glass.arff’, ‘ionosphere.arff’, ‘iris.arff’ y ‘segment-test.arff’ porque no contiene variables numéricas.

## 6.2. Ejercicio 2.

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

Para la implementación del método de agrupación basado en el particionado k-Means hemos utilizado las librerías de python ‘numpy’ y ‘sklearn’. El código generado se puede ver a continuación junto a la explicación de este.

```
14  # Cargar el dataset
15  X_sparse, y = load_svmlight_file("dataset.libsvm")
16
17  # Convertirlo a formato denso
18  X = np.array(X_sparse.todense())
19
20  # Generación de centroides
21  centroids = np.empty([n_classes, X.shape[1]])
22  for c in range(n_classes):
23      centroids[c] = np.mean(X[y == c], axis=0)
```

Primero de todo, cargamos el conjunto de datos correspondiente. La transformación de los ficheros de datos se han realizado con la herramienta de Weka habilitada para ello. Es necesario convertir los datos en una matriz densa porque por defecto se trata de una matriz dispersa.

Lo siguiente es la generación de los centroides, que se han inicializado a través de la media de todos los patrones de entrenamiento por clase y en orden como se puede ver.

Esto facilitará que al ejecutar el algoritmo de clustering cada cluster esté asignado a una clase y el orden de clusters sea idéntico al de clases, así podremos identificar fácilmente la agrupación de cada clase.

```
# Validación cruzada (k=10)
kf = KFold(n_splits=10)
for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    # k-Means
    kmeans = KMeans(n_clusters=n_classes, n_init=1, init=centroids).fit(X_train)
    prediction = kmeans.predict(X_test)
```

Por último, se ejecuta el algoritmo de agrupación (k-Means), el cuál vamos a realizar con una validación cruzada con k igual a 10 y obtendremos una predicción. El número de clusters es igual al número de clases, que se especifica antes de la ejecución, para que cada cluster esté asociado a una clase concreta como se ha especificado.

### 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.

En base a la predicción obtenida en el ejercicio anterior podemos obtener el error cuadrático medio y la tasa de patrones clasificados correctamente. Con estos estadísticos podremos evaluar la calidad del método para un conjunto de datos específico.

```
    # Evaluación
    mse += mean_squared_error(y_test, prediction)
    ccr += accuracy_score(prediction, y_test)*100

# Evaluación Medidas
mse /= 10
print mse
ccr /= 10
print ccr
```

Además, calculamos la media de las variantes de la validación cruzada como se puede apreciar para obtener datos más reales y fiables.

A continuación se muestra todo el script completo para su posible análisis:

```python
# -*- coding: utf-8 -*-

import numpy as np

from sklearn.datasets import load_svmlight_file
from sklearn.model_selection import KFold
from sklearn.cluster import KMeans
from sklearn.metrics import mean_squared_error
from sklearn.metrics import accuracy_score

# Número de clases
n_classes = 2

# Cargar el dataset
X_sparse, y = load_svmlight_file("dataset.libsvm")

# Convertirlo a formato denso
X = np.array(X_sparse.todense())

# Generación de centroides
centroids = np.empty([n_classes, X.shape[1]])
for c in range(n_classes):
    centroids[c] = np.mean(X[y == c], axis=0)
```

```python

# Medidas de evaluación
mse = 0
ccr = 0

# Validación cruzada (k=10)
kf = KFold(n_splits=10)
for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    # k-Means
    kmeans = KMeans(n_clusters=n_classes, n_init=1, init=centroids).fit(X_train)
    prediction = kmeans.predict(X_test)

    # Evaluación
    mse += mean_squared_error(y_test, prediction)
    ccr += accuracy_score(prediction, y_test)*100

# Evaluación Medidas
mse /= 10
print mse
ccr /= 10
print ccr
```

### 6.4. Ejercicio 5.

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

#### 6.4.1. Ejercicio 5.1.

Ejecute el algoritmo de particionado y evalúe su rendimiento.

- Diabetes (n_clases = 2).
  - MSE: 0.343967874231.
  - CCR: 65.6032125769%.

- Glass (n_clases = 7).
  - No ha sido posible el cálculo porque desborda la memoria porque los valores de entrada son demasiado grandes y la librería no es capaz de hacer el algoritmo solicitado.

- Ionosphere (n_clases = 2).
  - MSE: 0.29373015873.
  - CCR: 70.626984127%.

- Iris (n_clases = 3).
  - MSE: 0.0933333333333.
  - CCR: 90.6666666667%.

- Segment-test (n_clases = 7).
  - MSE: 3.43950617284.
  - CCR: 56.6666666667%.