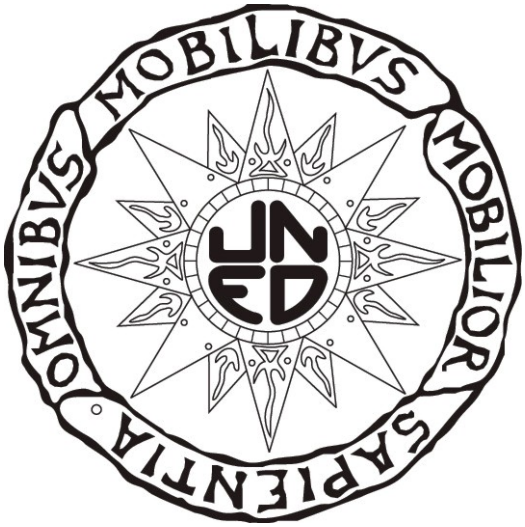


Aprendizaje Automático
ALGORITMO k-NN

k-NEAREST NEIGHBOURS

Actividad 1

ÍNDICE

1. Introducción

Las técnicas pertenecientes al denominado paradigma de aprendizaje vago, (del inglés lazy learning) son métodos de aprendizaje en los que no se infiere ningún modelo. En su lugar, el proceso de generalización a partir de los datos de entrenamiento se produce sólo en el momento en que se realiza una pregunta al sistema. De aquí, el nombre de aprendizaje vago o perezoso con el que se conoce a este tipo de algoritmos de aprendizaje.

Una de las principales ventajas del aprendizaje vago es que la respuesta a cada pregunta del sistema se obtienen mediante una aproximación local, es decir, a partir de los ejemplos de entrenamiento más parecidos, tal y como ocurre con el algoritmo k-NN. Esta propiedad de localizar se pierde en otros métodos de aprendizaje durante el proceso de generalización.

Por contra, una de las desventajas de este tipo de técnicas es que resultan normalmente lentas a la hora de responder a una pregunta del sistema, dado que, para ello, tienen que considerar todos y cada uno de los ejemplos del conjunto de entrenamiento. Sin embargo, no requieren de una fase de cada uno de los ejemplos del conjunto de entrenamiento. Sin embargo, no requieren de una fase de entrenamiento como la que sí utilizan todos aquellos paradigmas de aprendizaje basados en modelo. Otro inconveniente es que necesitan tener almacenados todos los ejemplos de entrenamiento, con el consiguiente requerimiento de espacio de almacenamiento en memoria.

2. Requisitos previos

1. Haber estudiado los capítulos 1 y 2 del texto base [Borrajo et al., 2006].
2. Haber estudiado la sección 6.3 (Capítulo 6), “Técnicas de aprendizaje vago”, del texto base [Borrajo et al., 2006]

3. Objetivos

1.- Familiarizarse con Weka en el uso de técnicas de aprendizaje vago.
2.- Experimentar con el algoritmo Ibk (versión Weka de k-NN)
3.- Comprobar la dependencia del algoritmo k-NN respecto de sus parámetros de configuración.
- Valor de k (número de vecinos más cercanos).
- Ponderación por distancia.

4.- Comprobar la dependencia del algoritmo k-NN respecto al número y poder de predicción de los atributos utilizados.

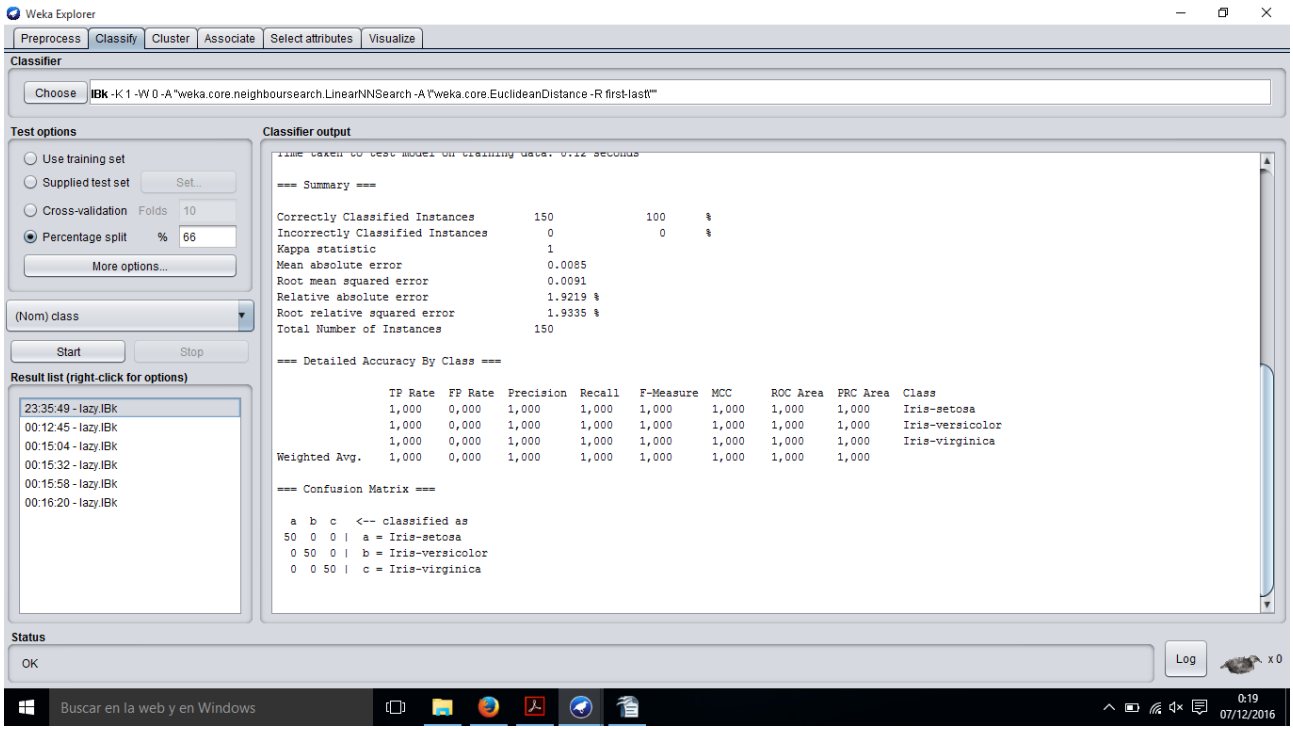
4. Material

- Programa Weka
- Archivos de datos: “iris.arff” y “iris-unknown.arff”.

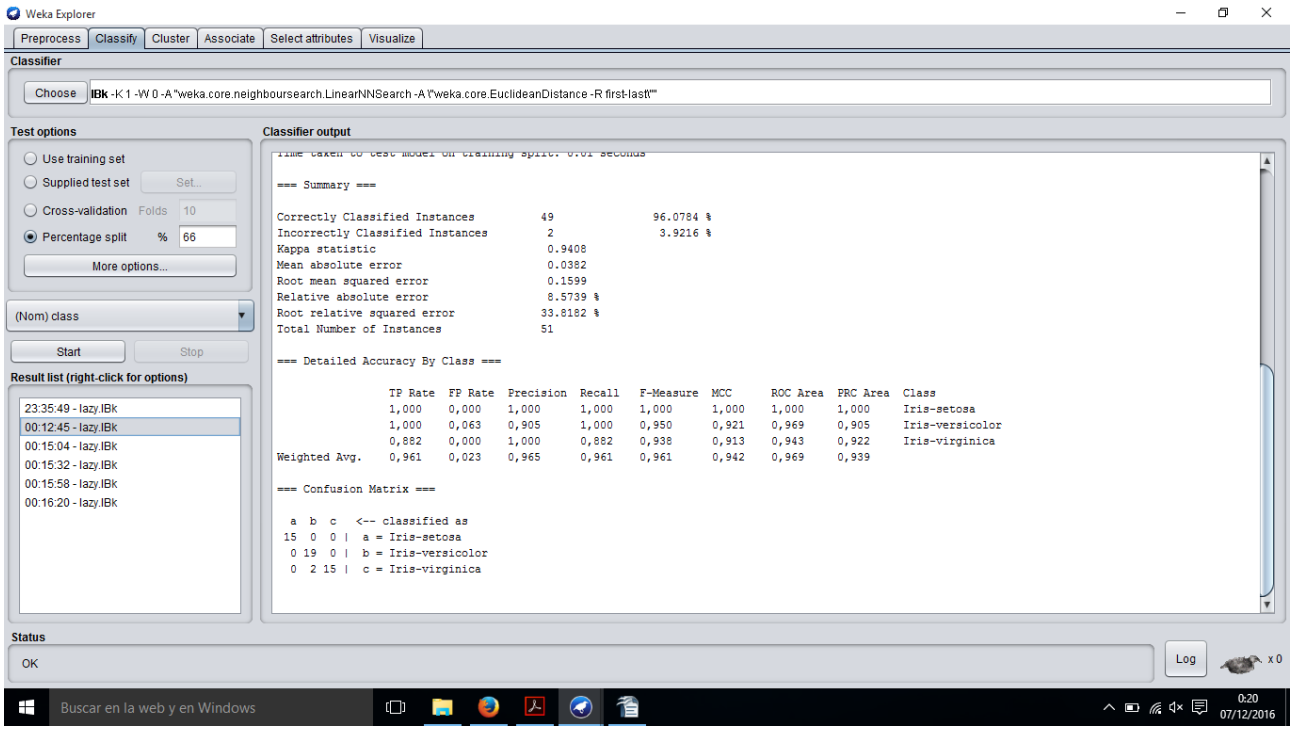
5. Actividades

Documentación a entregar apartado 5.4.-

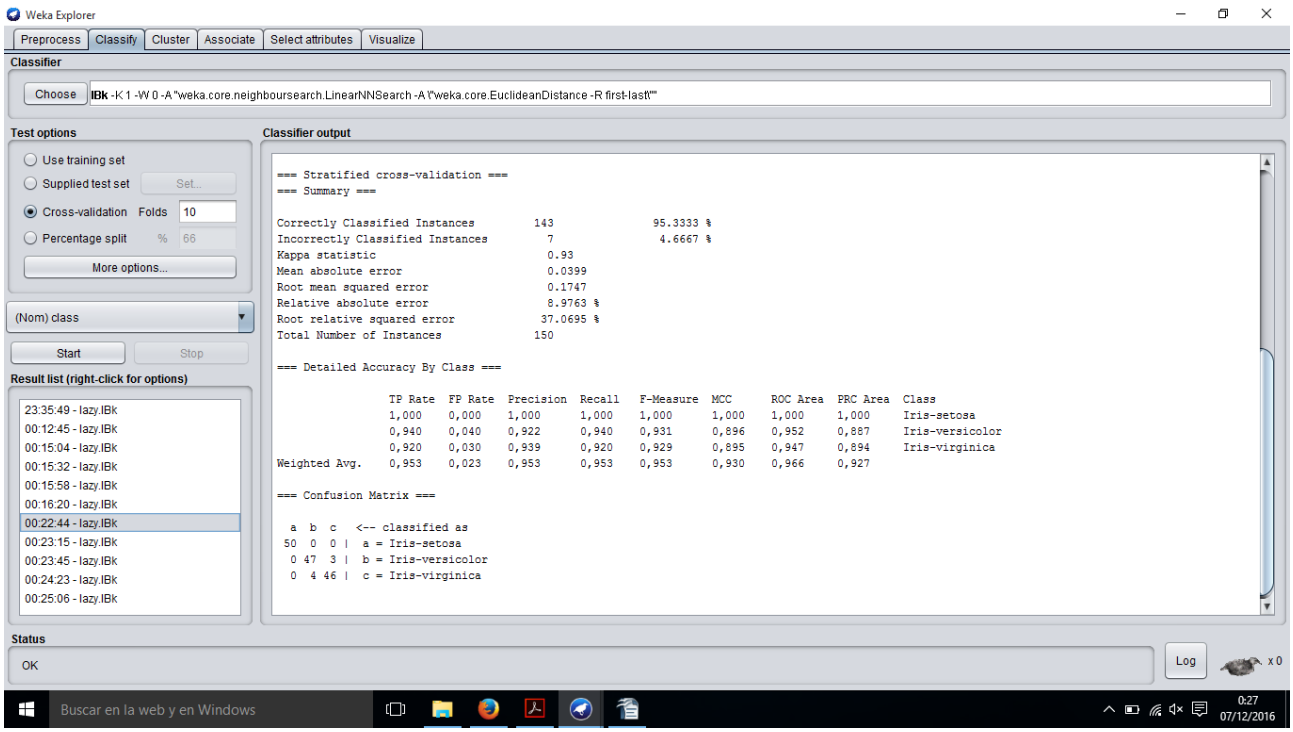
Experimento 5.4.1



Experimento 5.4.2



Experimento 5.4.3



Representar en una tabla el tanto por ciento de instancias correctamente claisficadas para cada experimento realizados y conteste a las siguientes preguntas.

TEST OPTIONS	SEMILLA	PORCENTAJE INSTANCIAS CORRECTAS CLASIFICADAS
Use training set	-	100%
Percentaje split - 66% (2/3)	1	96.0784%
Percentaje split - 66% (2/3)	2	94.1176%
Percentaje split - 66% (2/3)	3	96.0784%
Percentaje split - 66% (2/3)	4	94.1176%
Percentaje split - 66% (2/3)	5	88.2353%
Cross-validation - Folds 10	1	95.3333%
Cross-validation - Folds 10	2	95.3333%
Cross-validation - Folds 10	3	95.3333%
Cross-validation - Folds 10	4	96.00%
Cross-validation - Folds 10	5	95.3333%

1.- Por qué el mayor porcentaje se obtiene en el primer experimento.

Porque en el primer experimento se utiliza para las pruebas el mismo conjunto de entrenamiento con todas las instancias de ejemplo de entrada, por eso da un 100% de acierto, pero esta opción no es la más útil, ya que no introduce ningún grado de incertidumbre arrojando los resultados más optimistas que se pueden conseguir.

2.- Por qué fluctúa dicho porcentaje en las distintas ejecuciones del segundo y tercer experimento.

Para contestar esta pregunta, necesitamos conocer como trabaja weka, dada la configuración actual:

- En el **segundo** experimento primero se realiza una des-ordenación de las instancias (esta desordenación no sucede si activamos la casilla "preserve order for %split"), y después a partir de la semilla %split se eligen los conjuntos de entrenamiento y de prueba. En nuestro ejemplo se utilizan 51 elementos para pruebas.
- En el **tercer** experimento se van eligiendo conjuntos de 10 elementos para pruebas y el resto para entrenamiento, hasta completar todos los elementos del conjunto inicial iterando con grupos 10 elementos hasta completar todos los ejemplos de entrada. En nuestro ejemplo se utilizan 150 elementos para pruebas (iteradas en bloques de 10).

En ambos casos las fluctuaciones se deben a que al elegir los grupos elementos muy similares de distintas clases clasifican erróneamente alguna prueba, ese particionado de los grupos es el causante de la fluctuación.

3.- Por qué fluctua menos el mencionado porcentaje en el tercer experimento que en el segundo.

La razón es que el tercer experimento, utiliza mayor número de ejemplos de entrenamiento que el segundo, incluso en cada iteración y además utiliza más pruebas en todo el experimento, ya que utiliza todos los ejemplos para pruebas, aunque vaya en bloques de 10 en cada iteración, pero todos. En tal caso también se cometen más clasificaciones erróneas, pero como los porcentajes los calcula con respecto a los ejemplos de prueba, el porcentaje resultante de aciertos es mayor. Teniendo en cuenta lo anterior algún ejemplo por similitud de los grupos con el elegido entre 2 clases distintas clasificará erróneamente algunas prueba.

4.- Por qué se dice que el resultados más fiable es el obtenido en el tercer experimento, el siguiente más fiable es el obtenido en el segundo y el menos fiable corresponde al obtenido en el primero.

Como se ha dicho en el apartado 1) el primer experimento arroja los resultados más óptimos, sin introducir ningún grado de incertidumbre, esto no es lo deseable, necesitamos saber como trabaja utilizando conjunto de pruebas y entrenamiento sobre la misma entrada, para ver como trabaja el clasificador. Por tanto es el menos fiable aunque su rendimiento sea del 100%.

Por otro lado el tercer experimento con respecto al segundo utiliza todos los ejemplos de entrada para pruebas, probando con conjuntos pequeños en cada iteracion con respecto al resto de elementos que utiliza de entrenamiento. Por otro lado el segundo experimento arrojara peores resultados al no probar todo el conjunto.

5.- Dado que el algoritmo k-NN no aprende ningún modelo, a qué se asocia el porcentaje de acierto de clasificación.

Recordemos que el algoritmo k-NN es un algoritmo basado en instancias, que se basa en la similitud de los elementos del conjunto de entrada poder conocer la clase a la que pertenece, en todo caso, conocemos las clases (algoritmo supervisado) a las que pertenece cada ejemplo para entrenar al clasificador.

6.- Muestre e interprete los resultados de la matriz de confusión obtenida en el experimento 4.3 (para semilla=1).

=== Confusion Matrix ===

```
  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 47  3 |  b = Iris-versicolor
  0  4 46 |  c = Iris-virginica
```

Si sumamos todas las cantidades, se han utilizado 150 elementos de prueba, de los cuales 50 han sido clasificados Iris-setosa correctaemente. Otros 50 ejemplos de prueba Iris-versicolor de los cuales 47 han sido clasificados correctamente y 3 de ellos han sido clasificados erróneamente como Iris-virgnica. Por último las 50 pruebas restantes que eran Iris-virginica, 46 han sido clasificados correctamente y 4 erróneos clasificados como Iris-versicolor.

Documentación a entregar apartado 5.5.-

Representar en una tabla el tanto por ciento de instancias correctamente clasificadas para cada experimento reliazado y comente los resultados obtendios. De otro lado, justifique si es cierto o no la siguiente afirmación: "*El algoritmo k-NN obtiene siempre mejores resultados de clasificación a medida que aumenta el valor de k*". En el caso de que considere que dicha afirmación no es cierta, utilice un contraejemplo para refutarla, es decir, invente y muestre una distribución de, por ejemplo, no más de 10 instancias (incluyendo instancias positivas y negativas) en el plano X-Y(2D). La idea es mostrar que si, en la distribución elegida de instancias positivas y negativas, cualquiera de ellas se considerara de clasificación desconocida, entonces dicha instancia sería clasificada correctamente con k=1, pero erróneamente con k=3.

TEST OPTIONS	K	PORCENTAJE INSTANCIAS CORRECTAS CLASIFICADAS
Cross-validation - Folds 10 -sem 1	1	95.3333%
Cross-validation - Folds 10 -sem 1	2	94.6667%
Cross-validation - Folds 10 -sem 1	3	95.3333%
Cross-validation - Folds 10 -sem 1	4	95.3333%
Cross-validation - Folds 10 -sem 1	5	95.3333%
Cross-validation - Folds 10 -sem 1	6	96.6667%
Cross-validation - Folds 10 -sem 1	7	96.6667%
Cross-validation - Folds 10 -sem 1	8	96.00%
Cross-validation - Folds 10 -sem 1	9	96.00%
Cross-validation - Folds 10 -sem 1	10	96.00%

Se puede comprobar que la afirmación es falsa, ya que para k=2 se obtienen peores resultados que para k=1. Además hay 4 empates a 95.3333%, con lo cual tampoco hay mejora. Por último cabe mencionar que los últimos 3 resultados son ligeramente peores que los obtenidos para k=6 y k=7.

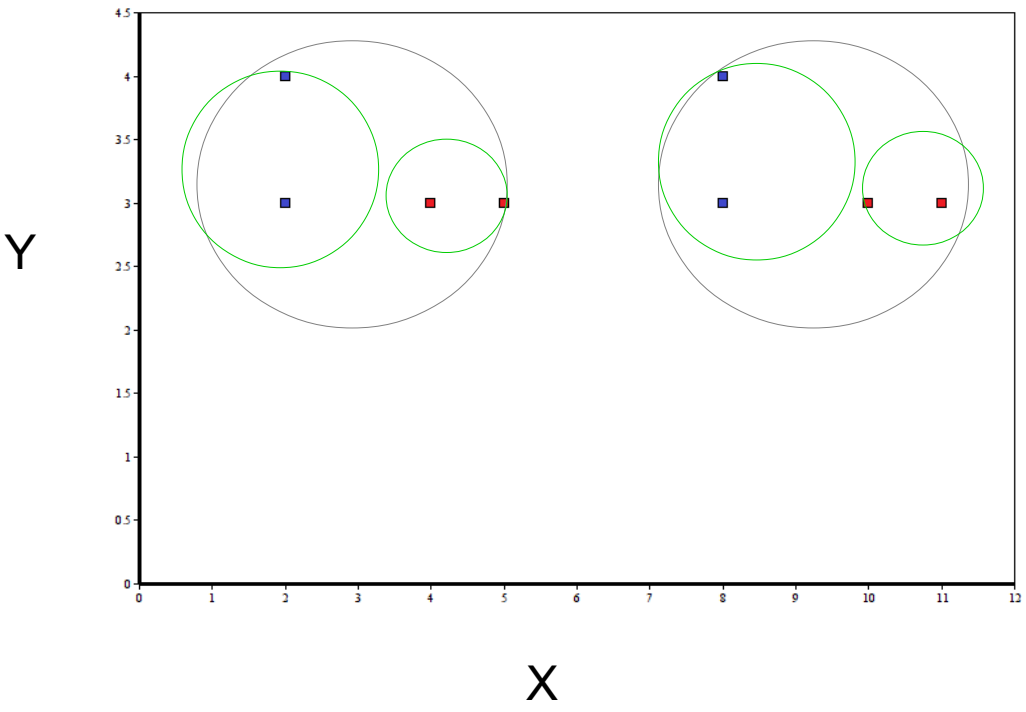
De esto podemos pensar que para cada estudio habrá un k que dará mejores resultados que otros, y que ese k es mejor si es impar, ya que será capaz de resolver los casos de empate que pudiesen darse entre vecinos, cuando en un **supuesto ficticio estudio** fuese k=6 y hubiese 3 vecinos de una clase y 3 de otra, sería necesaria utilizar una heurística para deshacer ese empate, o bien elegir un k impar por ejemplo k=5 o k=7 (si nos ofrecen buenos resultados) para que en ese **supuesto ficticio estudio** no haya igual número de vecinos de una clase que de otra. Ojo puede ocurrir que hayan más clases y volvieran a empatarse, lo cual viene de nuevo a confirmar que habrá un k mejor que otro y/o más adecuado para cada estudio.

Ahora bien siguiendo con el enunciado de la práctica, crearemos un **experimento teórico**, para intentar refutar lo comentado y ver que con k=1 clasifica correctamente y no con k=3 para cualquier instancia que elijamos:

Supongamos que nuestro conjunto de instancias llamado D esta compuesto de 8 elementos, cada uno tiene 2 atributos (X e Y, para simplificar las cosas) y los valores de clase pueden ser A y R (los cuales representaremos con los colores azul y rojo).

D={(2, 3, rojo), (2, 4, rojo), (8, 3, rojo), (8, 4, rojo), (4, 3, azul), (5, 3, azul), (10, 3, azul), (11, 3, azul)}

Si los representamos en dos dimensiones, obtendremos algo similar a esto:



Los círculos verdes simbolizan k=1 y los grises k=3.

Si elegimos cualquier ejemplo de prueba y consideramos su clasificación desconocida, vemos que al clasificarlos con k=1, tenemos un vecino de la misma clase cerca que nos clasificará correctamente el ejemplo. Sin embargo para k=3 elijamos el que elijamos se clasificará erróneamente, ya que tendremos una **ponderación de 2 elementos** de una clase diferente en contra de **1 elemento** de la clase correcta lo que dará esa clasificación errónea.

Vemos que el ejemplo ha sido preparado para que las distancias de los grupos no interfieran con el criterio de k vecinos, para dar fe lo que queremos refutar. Que a mayor k no se clasifica mejor, sino que para cada experimento existe un k adecuado.

Documentación a entregar apartado 5.6.-

Represente y muestre la gráfica correspondiente a la clase real (eje X) respecto a la clase predicha (eje Y). Identifique e indique las instancias que están clasificadas erróneamente, proporcionando para cada una de ellas su etiqueta (número de orden en el conjunto de datos), su clase real y la clase predicha. Si lo considera relevante, incluya la representación de cualquier otra gráfica que considere oportuna, y comente el porqué de dicha relevancia.

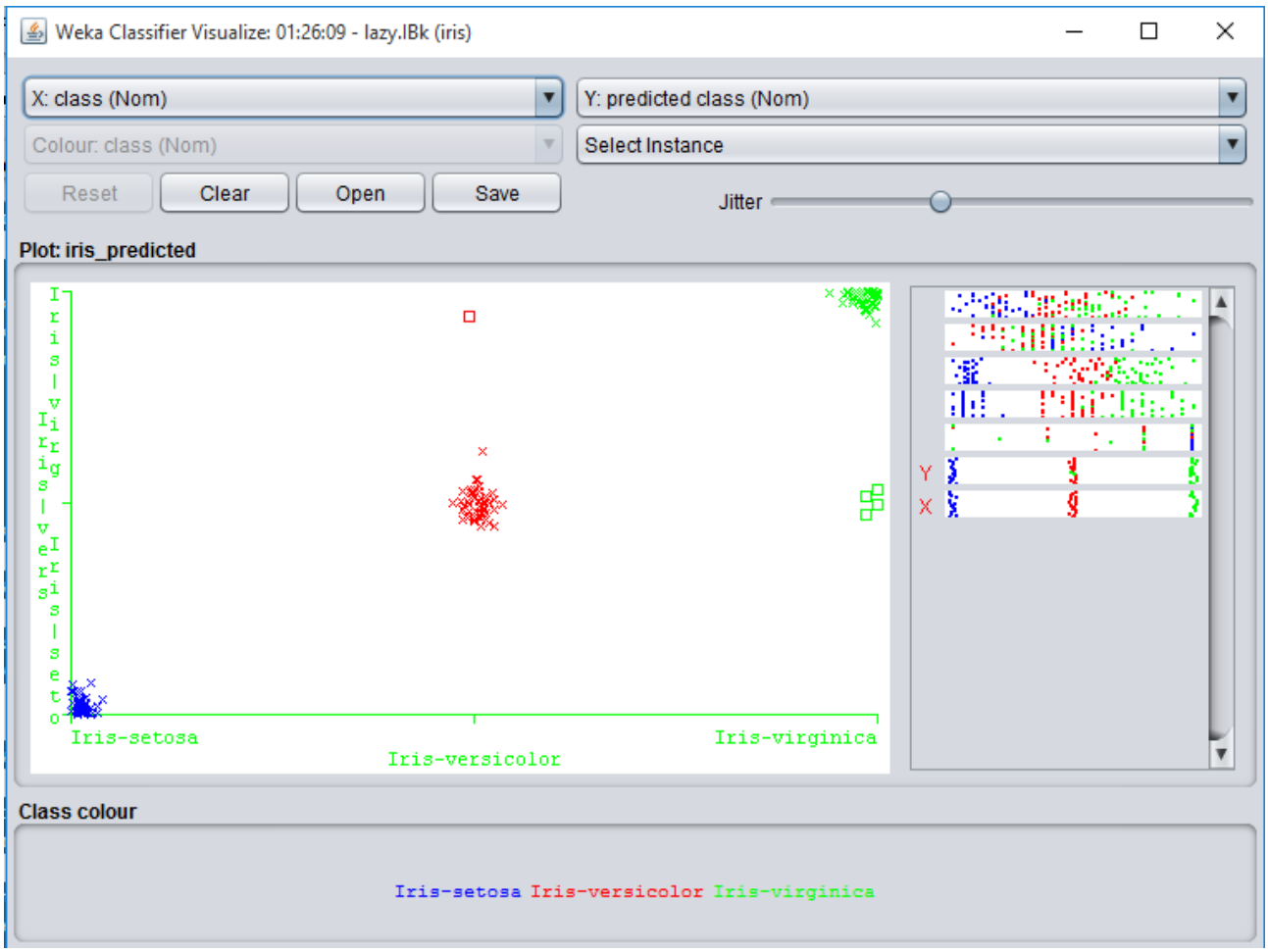
Hay dos gráficas que merecen ser comentadas, para k=6 y para k=7. Exponemos su gráfica resultados y los casos fallidos con sus etiquetas.

Para k=6 los resultados son:

```
=== Run information ===

Scheme:       weka.classifiers.lazy.IBk -K 6 -W 0 -A
"weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R
first-last\""
Relation:     iris
Instances:    150
Attributes:   5
              sepallength
              sepalwidth
              petallength
              petalwidth
              class
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

IB1 instance-based classifier
using 6 nearest neighbour(s) for classification


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         145               96.6667 %
Incorrectly Classified Instances         5                3.3333 %
Kappa statistic                          0.95
Mean absolute error                      0.0391
Root mean squared error                  0.137
Relative absolute error                  8.789  %
Root relative squared error             29.0555 %
Total Number of Instances              150



=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC
Area  PRC Area  Class
```

```
                 1,000    0,000    1,000      1,000    1,000      1,000
1,000     1,000      Iris-setosa
                 0,980    0,040    0,925      0,980    0,951      0,927
0,994     0,985      Iris-versicolor
                 0,920    0,010    0,979      0,920    0,948      0,925
0,994     0,985      Iris-virginica
Weighted Avg.    0,967    0,017    0,968      0,967    0,967      0,951
0,996     0,990

=== Confusion Matrix ===

  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 49  1 |  b = Iris-versicolor
  0  4 46 |  c = Iris-virginica
```

La gráfica para k=6 es:



Vemos 5 fallos marcados con cuadrados, 1 rojo y 4 verdes sus etiquetas son las correspondientes a las instancias 15, 109, 92, 80 y 123, todos los datos de cada instancia son los siguientes:

```
Plot : weka.classifiers.lazy.IBk (iris)
Instance: 15
      sepallength : 6.0
       sepalwidth : 2.7
      petallength : 5.1
       petalwidth : 1.6
prediction margin : -0.6642066420664207
  predicted class : Iris-virginica
            class : Iris-versicolor
```

```
Plot : weka.classifiers.lazy.IBk (iris)
Instance: 109
      sepallength : 6.0
       sepalwidth : 2.2
      petallength : 5.0
       petalwidth : 1.5
prediction margin : -0.6642066420664207
  predicted class : Iris-versicolor
            class : Iris-virginica
```

```
Plot : weka.classifiers.lazy.IBk (iris)
Instance: 92
      sepallength : 4.9
       sepalwidth : 2.5
      petallength : 4.5
       petalwidth : 1.7
prediction margin : -0.6642066420664207
  predicted class : Iris-versicolor
            class : Iris-virginica
```

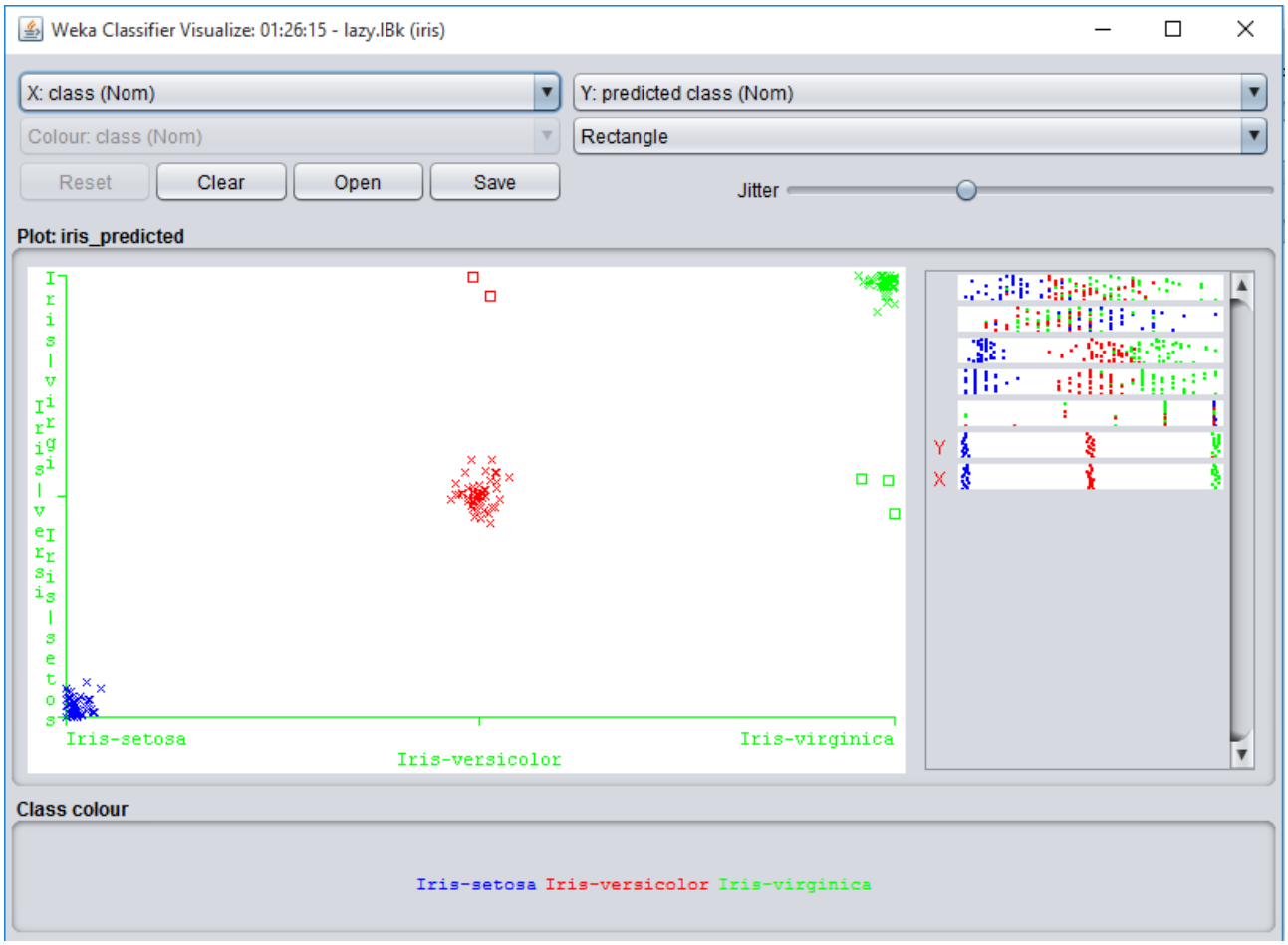
```
Plot : weka.classifiers.lazy.IBk (iris)
Instance: 80
      sepallength : 6.3
       sepalwidth : 2.8
      petallength : 5.1
       petalwidth : 1.5
prediction margin : -0.3321033210332104
  predicted class : Iris-versicolor
            class : Iris-virginica
```

```
Plot : weka.classifiers.lazy.IBk (iris)
Instance: 123
      sepallength : 6.1
       sepalwidth : 2.6
      petallength : 5.6
       petalwidth : 1.4
prediction margin : 0.0
  predicted class : Iris-versicolor
            class : Iris-virginica
```

Para k=7 los resultados son:

```
=== Run information ===

Scheme:       weka.classifiers.lazy.IBk -K 7 -W 0 -A
"weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R
first-last\""
Relation:     iris
Instances:    150
Attributes:   5
              sepallength
              sepalwidth
              petallength
              petalwidth
              class
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

IB1 instance-based classifier
using 7 nearest neighbour(s) for classification


Time taken to build model: 0.01 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         145               96.6667 %
Incorrectly Classified Instances         5                3.3333 %
Kappa statistic                          0.95
Mean absolute error                      0.0387
Root mean squared error                  0.1282
Relative absolute error                  8.7166 %
Root relative squared error             27.1942 %
Total Number of Instances              150

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC
Area  PRC Area  Class
                 1,000    0,000    1,000      1,000    1,000      1,000
1,000     1,000     Iris-setosa
                 0,960    0,030    0,941      0,960    0,950      0,925
0,996     0,990     Iris-versicolor
                 0,940    0,020    0,959      0,940    0,949      0,925
0,996     0,991     Iris-virginica
Weighted Avg.    0,967    0,017    0,967      0,967    0,967      0,950
0,997     0,994

=== Confusion Matrix ===

  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 48  2 |  b = Iris-versicolor
  0  3 47 |  c = Iris-virginica
```

La gráfica para k=7 es:



Vemos 5 fallos marcados con cuadrados, 2 rojo y 3 verdes sus etiquetas son las correspondientes a las instancias 15, 109, 92, 80 y 135, todos los datos de cada instancia son los siguientes:

```
Plot : weka.classifiers.lazy.IBk (iris)
Instance: 135
      sepallength : 6.3
       sepalwidth : 2.5
      petallength : 4.9
       petalwidth : 1.5
prediction margin : -0.14240506329113928
  predicted class : Iris-virginica
            class : Iris-versicolor
```

```
Plot : weka.classifiers.lazy.IBk (iris)
Instance: 15
      sepallength : 6.0
       sepalwidth : 2.7
      petallength : 5.1
       petalwidth : 1.6
prediction margin : -0.4272151898734177
  predicted class : Iris-virginica
            class : Iris-versicolor
```

```
Plot : weka.classifiers.lazy.IBk (iris)
Instance: 109
      sepallength : 6.0
       sepalwidth : 2.2
      petallength : 5.0
       petalwidth : 1.5
prediction margin : -0.4272151898734177
  predicted class : Iris-versicolor
            class : Iris-virginica
```

```
Plot : weka.classifiers.lazy.IBk (iris)
Instance: 80
      sepallength : 6.3
       sepalwidth : 2.8
      petallength : 5.1
       petalwidth : 1.5
prediction margin : -0.4272151898734177
  predicted class : Iris-versicolor
            class : Iris-virginica
```

```
Plot : weka.classifiers.lazy.IBk (iris)
Instance: 92
      sepallength : 4.9
       sepalwidth : 2.5
      petallength : 4.5
       petalwidth : 1.7
prediction margin : -0.4272151898734177
  predicted class : Iris-versicolor
            class : Iris-virginica
```

Podríamos atribuir esta circunstancia a eso mismo a la ponderación de los vecinos que hay en cada k, ojo no ponderación de distancias, sino de acumulación de vecinos de cada clase en relación al número k.

Documentación a entregar apartado 5.7.-

Indicar el valor de k para el que se produce el mayor porcentaje de acierto de clasificación.

IB1 instance-based classifier using 6 nearest neighbour(s) for classification

Es decir, para k=6.

¿Coincide ese valor de k con el del estudio realizado en el paso 5?

Si coincide con el k obtenido, aunque también el 5 obtuvimos que k=7 ofrecía un resultado similar.

¿El porcentaje de acierto de clasificación obtenido con el k seleccionado automaticámente coincide con el porcentaje obtenido en el paso 5 cuando allí se utilizó el mismo valor de k? De no ser asi, justifique el porqué de la discrepancia.

Según la ayuda:

crossValidate -- Whether hold-one-out cross-validation will be used to select the best k value between 1 and the value specified as the KNN parameter.

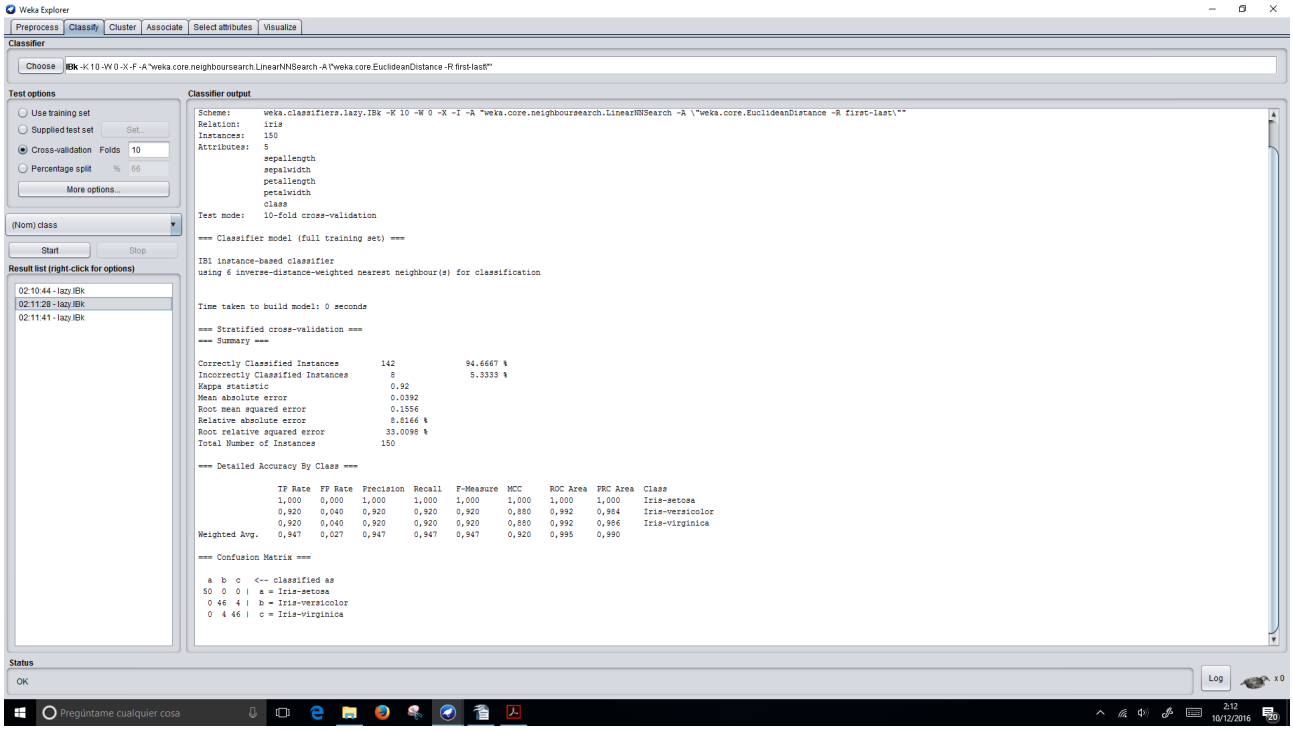
Es decir, utiliza una validación cruzada dejando uno fuera, que tal como se vio en las tutorias y en los pdf facilitados en ellas, se usa todo el el conjunto para el entrenamiento menos uno para la prueba en cada una de las iteraciones. Los resultados obtenido se ofrecen a continuación con un 95,3333% en este experimento para k=6 y en el ejercicio 5 obtuvimos un 96,6667%.

```
=== Run information ===

Scheme:       weka.classifiers.lazy.IBk -K 10 -W 0 -X -A
"weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R
first-last\""
Relation:     iris
Instances:    150
Attributes:   5
              sepallength
              sepalwidth
              petallength
              petalwidth
              class
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

IB1 instance-based classifier
using 6 nearest neighbour(s) for classification


Time taken to build model: 0 seconds
```

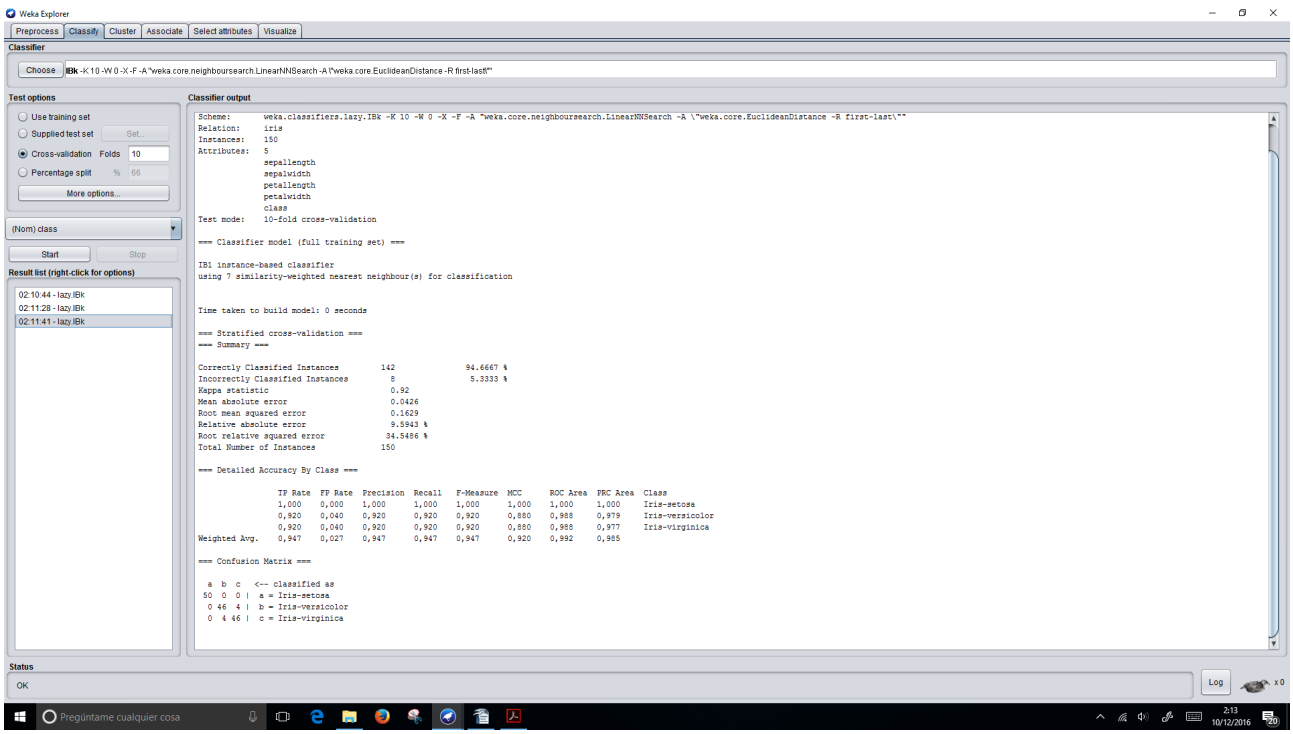
```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         143               95.3333 %
Incorrectly Classified Instances         7                4.6667 %
Kappa statistic                          0.93
Mean absolute error                      0.0446
Root mean squared error                  0.1597
Relative absolute error                 10.0239 %
Root relative squared error             33.8811 %
Total Number of Instances              150

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC
Area  PRC Area  Class
                 1,000    0,000    1,000      1,000    1,000      1,000
1,000     1,000     Iris-setosa
                 0,940    0,040    0,922      0,940    0,931      0,896
0,990     0,978     Iris-versicolor
                 0,920    0,030    0,939      0,920    0,929      0,895
0,990     0,976     Iris-virginica
Weighted Avg.    0,953    0,023    0,953      0,953    0,953      0,930
0,993     0,985

=== Confusion Matrix ===

  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 47  3 |  b = Iris-versicolor
  0  4 46 |  c = Iris-virginica
```

Documentación a entregar apartado 5.8.-

Experimento 5.8.1



Experimento 5.8.2



Indique el valor de k elegido en cada experimento y los porcentajes de acierto de clasificación respectivos. A la luz de los resultados obtenido indique si, en este caso, se justifica la opción de ponderar por distancia (compare los resultados con los obtenidos en el paso7).

Mostramos en la tabla esos valores:

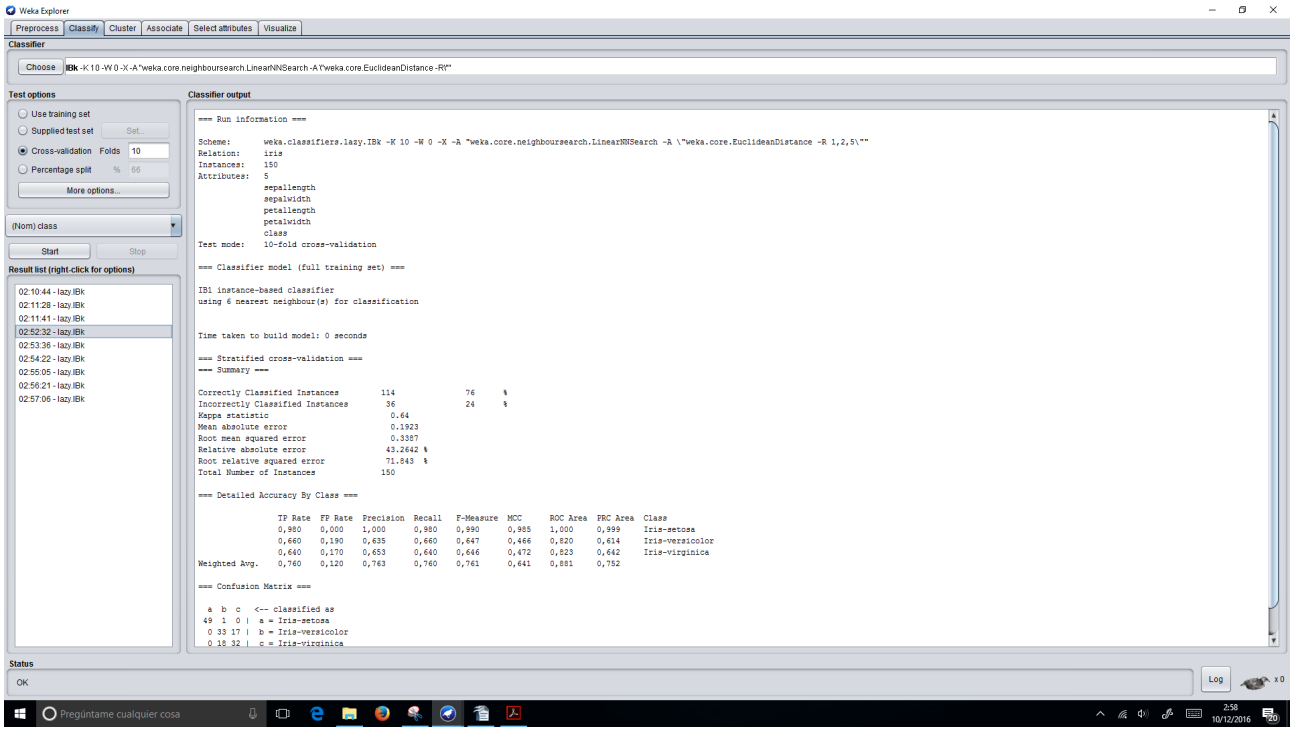
Distance Weighting	K	%ACERTO
Weight by 1/distance	6	94.6667 %
Weight by 1-distance	7	94.6667 %

Al comparar con el paso7 (95.3333 % de aciertos) vemos que obtenemos un porcentaje peor, correspondiendo con una instancia mal clasificada en este experimento. Ahora bien, la ponderación utilizada nos ayuda a deshacer empates entre el numero de instancias a la hora de clasificarlas en sus respectivas clases, con lo cual no es un parámetro (Distance Weighting) que debamos descartar por arrojar ese resultado ligeramente peor, ya que esta heurística siempre nos va a permitir clasificar con los k-vecinos con un criterio conocido.

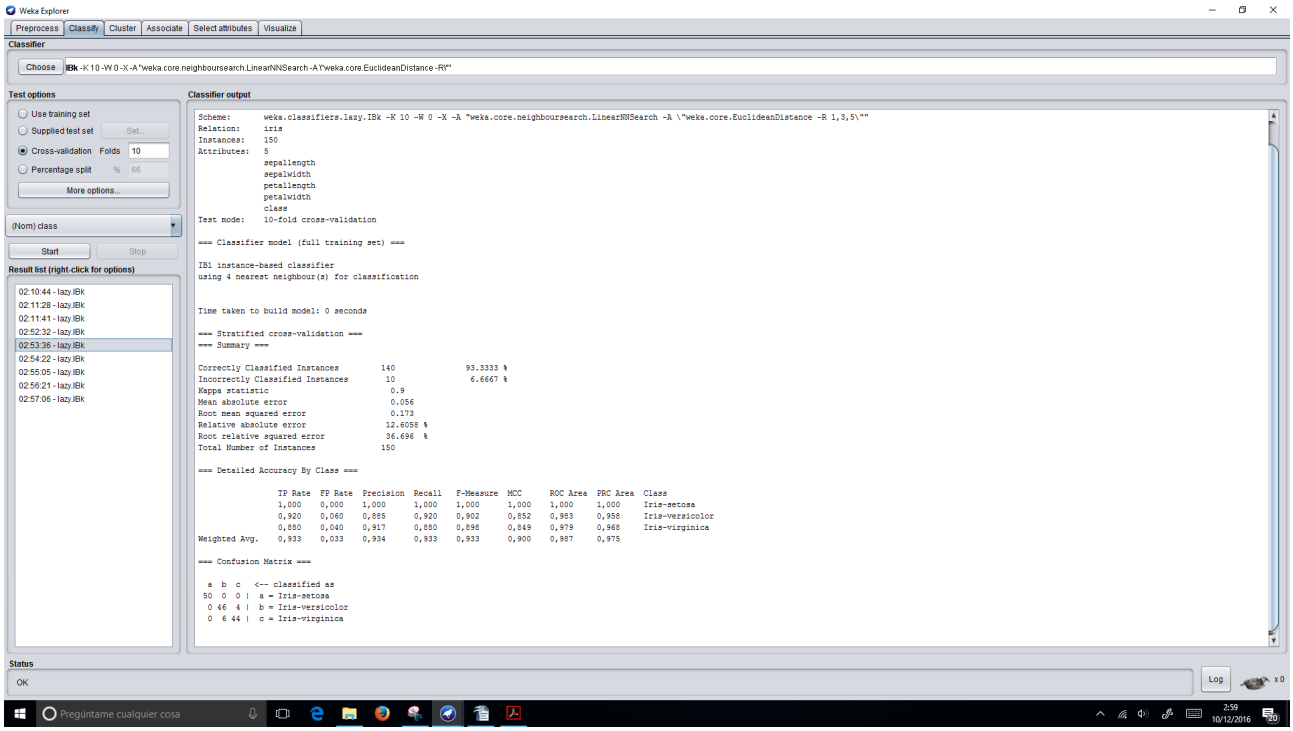
Por otro lado si intentamos buscar una explicación a lo ocurrido, es decir, que el primer experimento, llamemos A, utilice k=6 y el segundo experimento, llamemos B, utilice k=7, se podría deber a que en el A al usar esa heurística pudo clasificar ofreciendo el mejor porcentaje de acierto y sin embargo en el experimento B con esa heurística necesito un vecino más para clasificar y poder el mejor resultado.

Documentación a entregar apartado 5.9.-

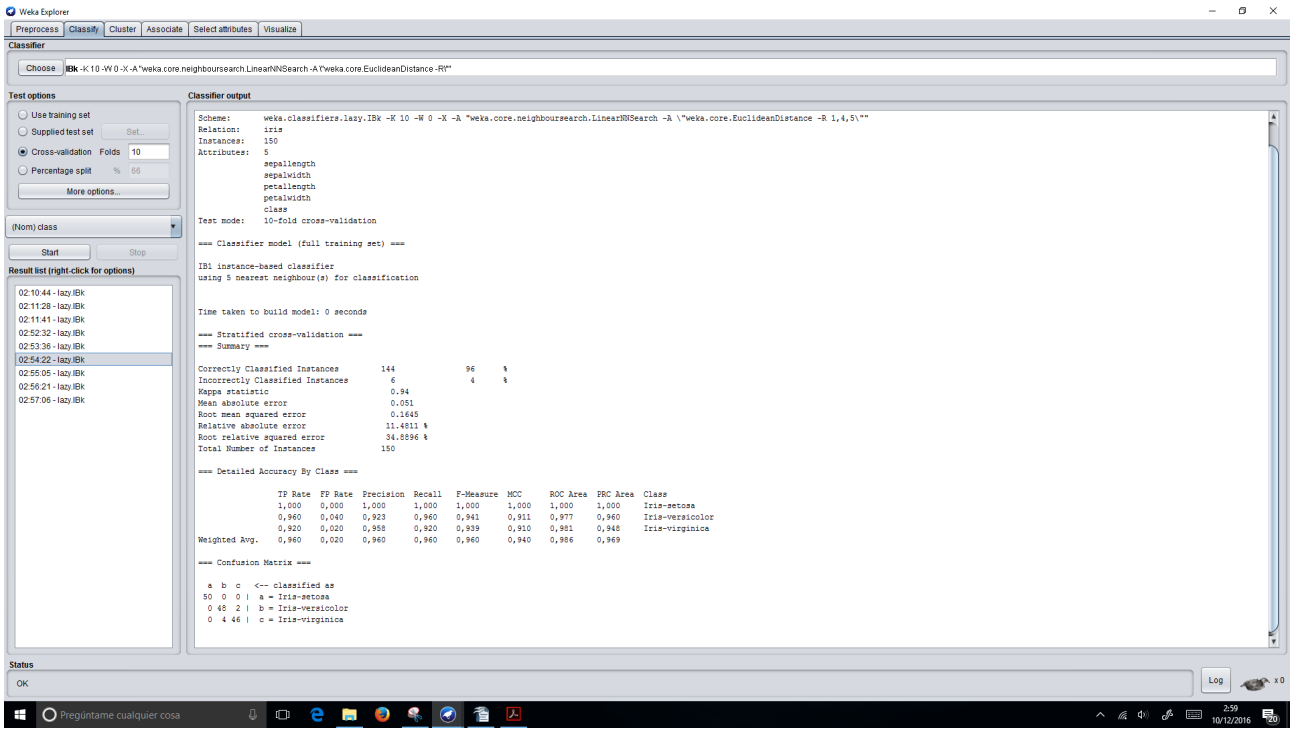
Experimento 5.9.1



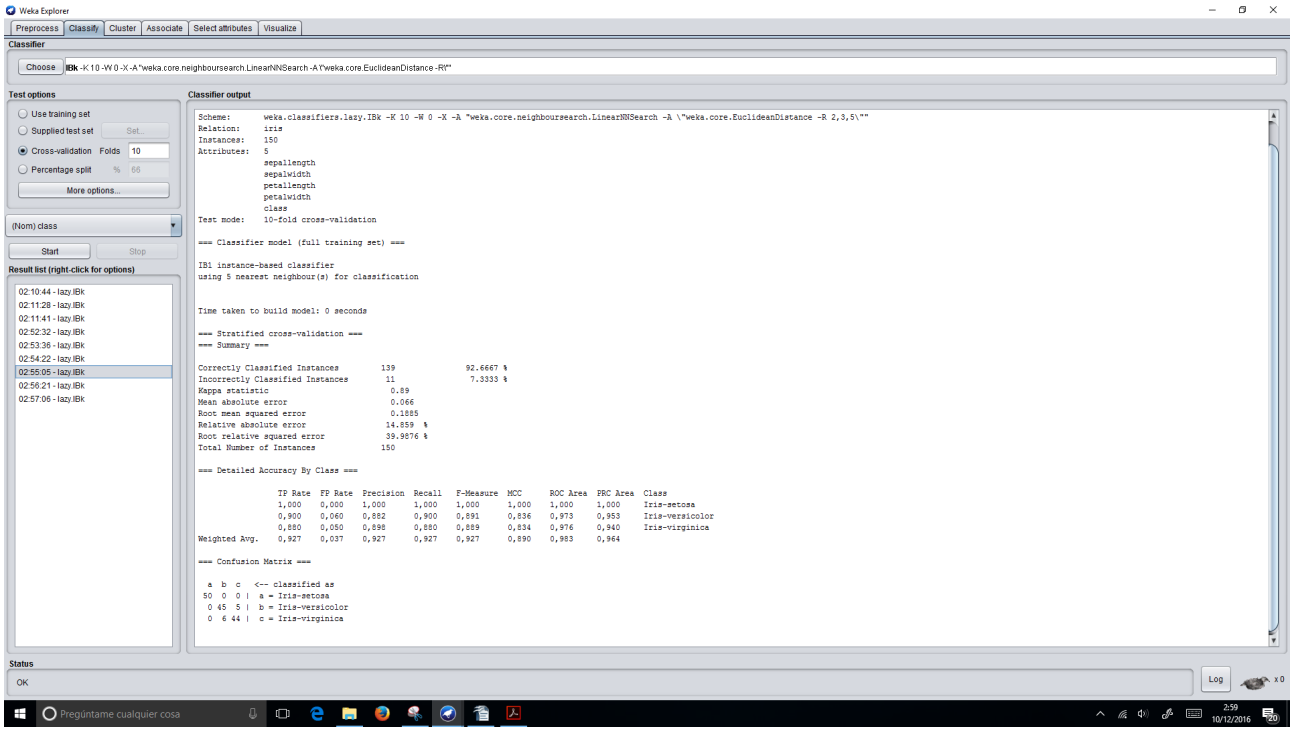
Experimento 5.9.2



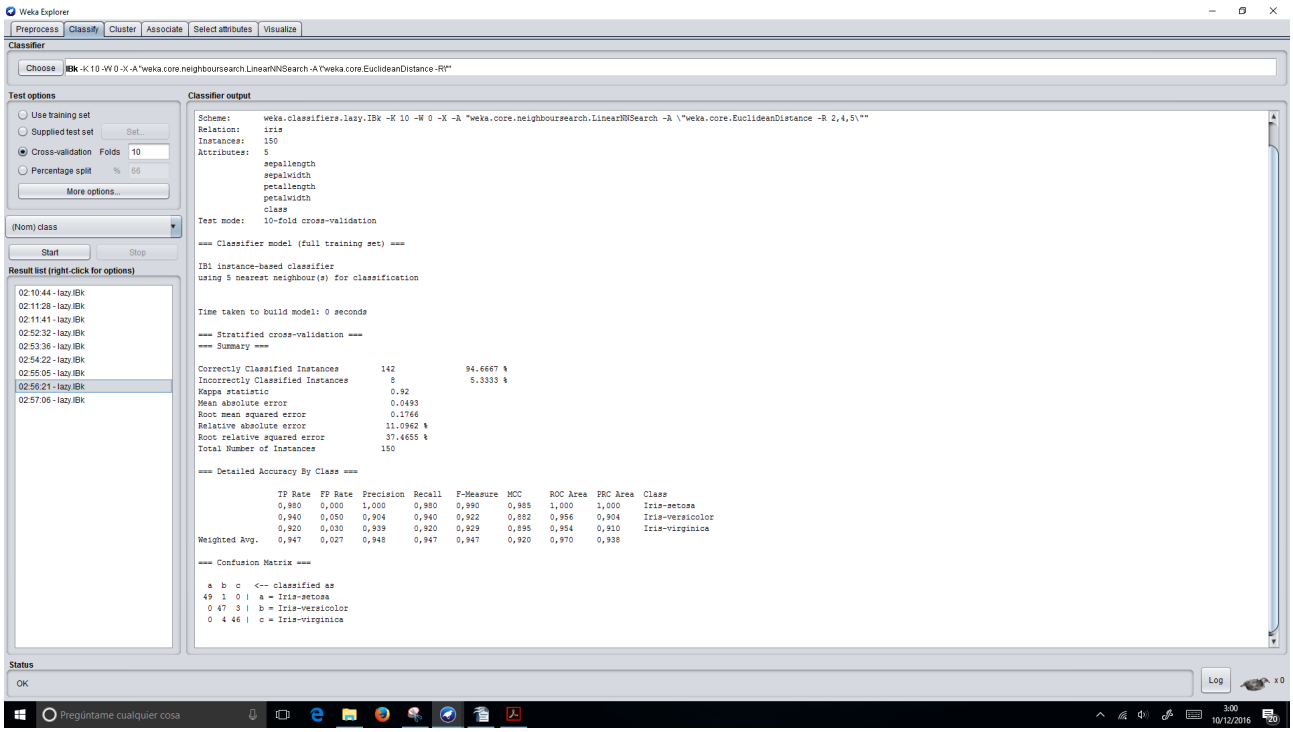
Experimento 5.9.3



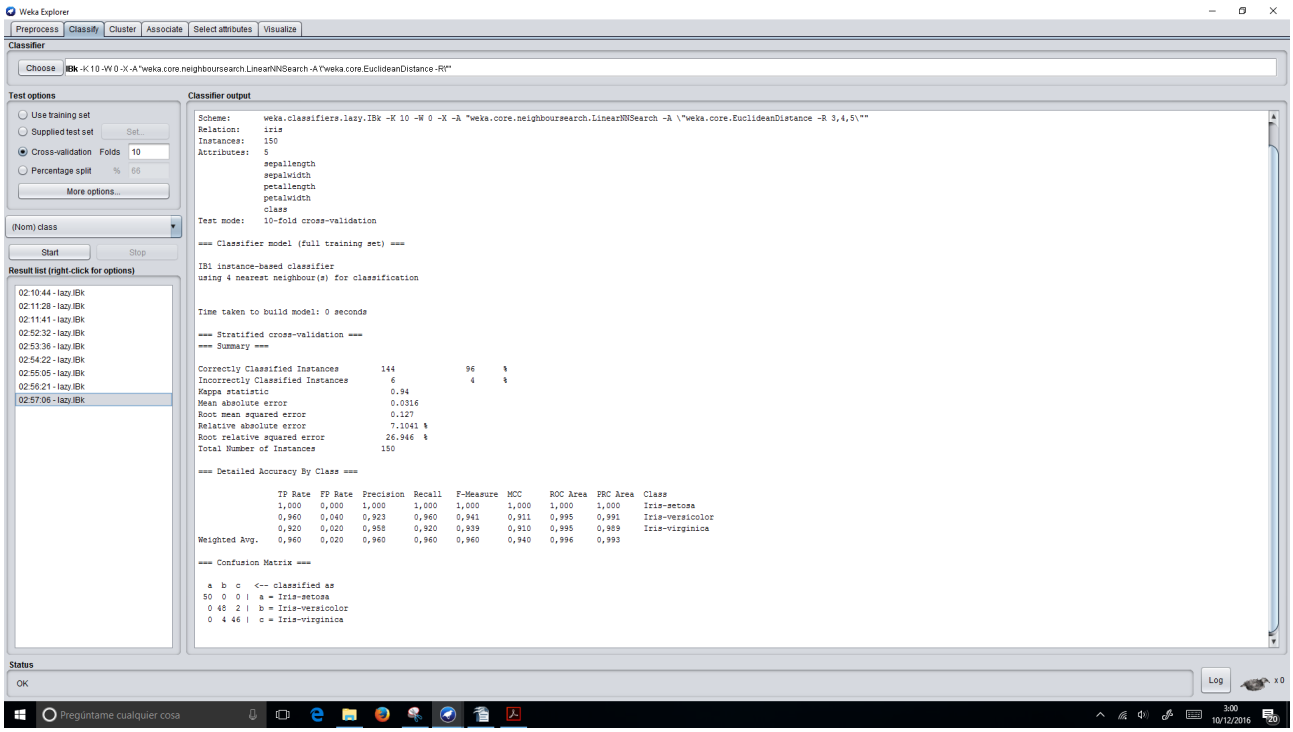
Experimento 5.9.4



Experimento 5.9.5



Experimento 5.9.6



Indicar el valor de k elegido en cada experimento y los porcentajes de acierto de clasificación respectivos. A la luz de los resultado obtenidos, indique si el algoritmo k-NN depende o no del subconjunto de atributos (elegido a partir del conjunto de atributos original). Si la respuesta es afirmativa, ¿existe alguna pareja de atributos que permita obtener un porcentaje correcto de clasificación tan o más competitivo como el que se obtuvo trabajando con todos los atributos del conjunto de datos? Compare los resultados con los obtenido en el paso 7.

Atributos elegidos	k	%ACIERTOS
1,2,5	6	76 %
1,3,5	4	93,3333%
1,4,5	5	96 %
2,3,5	5	92,6667 %
2,4,5	5	94,6667 %
3,4,5	4	96 %

Recordemos que en el paso 7 obtuvimos el porcentaje 95,33333% de aciertos, con k=6 y k=7. Vemos que efectivamente el porcentaje de acierto varia según los atributos elegidos, es decir, unos atributos clasifican mejor que otros, incluso, se puede elegir un k distinto, aunque podemos ver que con un k determinado, como el 5, obtenemos también porcentajes mejores y/o peores. Podemos ver que utilizando la pareja (4,5) se obtienen los mayores porcentajes de acierto. También podemos decir que un 4% utilizando (1,4,5) y (3,4,5) nos reparan obtener una clasificación cercana al 100% de aciertos utilizando todo el conjunto de datos para entrenamiento, como hicimos en el primer experimento de la actividad.

Documentación a entregar apartado 5.10.-

Muestre el contenido del archivo generado.

Vamos a utilizar k=6 con fold=10 sin ponderación, ya que los resultados del 5.5. con un 96,6667% ha sido el mejor valor obtenido en toda la actividad.

```
@relation iris_predicted

@attribute sepallength numeric
@attribute sepalwidth numeric
@attribute petallength numeric
@attribute petalwidth numeric
@attribute 'prediction margin' numeric
@attribute 'predicted class' {Iris-setosa,Iris-versicolor,Iris-virginica}
@attribute class {Iris-setosa,Iris-versicolor,Iris-virginica}

@data
5.2,3.5,1.6,0.2,0.996678,Iris-setosa,?
4.9,3,1.4,0.3,0.997506,Iris-setosa,?
4.7,3.5,1.3,0.2,0.996678,Iris-setosa,?
4.6,3.1,1.3,0.1,0.996678,Iris-setosa,?
5,3,1.4,0.2,0.996678,Iris-setosa,?
7.1,3.2,4.7,1.4,-0.996678,Iris-versicolor,?
6.4,2.9,4.5,1.5,-0.996678,Iris-versicolor,?
6.9,3.1,4.3,1.5,-0.996678,Iris-versicolor,?
5.5,2.3,4,1.1,-0.996678,Iris-versicolor,?
6.3,2.8,4.6,1.5,-0.830565,Iris-versicolor,?
6.5,3.3,6,2.5,-0.996678,Iris-virginica,?
5.8,2.8,5.1,1.9,-0.996678,Iris-virginica,?
7.1,3,6.1,2.1,-0.996678,Iris-virginica,?
6.3,2.9,5.6,1.9,-0.996678,Iris-virginica,?
6.5,3,5.2,2.2,-0.996678,Iris-virginica,?
```

6. Conclusiones

El aprendizaje vago se describe como una serie de técnicas las cuales no nos ofrecen un modelo de aprendizaje. Este se consigue en base a la experiencia conocida, es decir, el conocimiento de la clases mediante aprendizaje supervisado o etiquetado. La clasificación de las instancias desconocidas se realiza por ejemplo con similitud y una serie de heurísticas que no permiten refinar la clasificación. Estos algoritmos suelen utilizar fuerza bruta para procesar todas las instancias, es decir, requieren mayor espacio y cálculo que otros paradigmas de aprendizaje.

En el primer experimento pudimos poner ver como se comporta el método k-NN, el cual es una de esas técnicas perteneciente al aprendizaje vago. En ese experimento pudimos elegir tres distintas formas de entrenamiento.

La primera forma ha sido utilizando todas las instancias como entrenamiento y su poder de clasificar al 100% esas mismas instancias satisfactoriamente, pero poco útil ya que solo clasificaría correctamente ante ejemplos de clase desconocidos que tuviesen mismos valores de los atributos que hubiese en las instancias de entrenamiento. La segunda forma fue utilizar un determinado porcentaje como entrenamiento (Percentaje split) y ver como se comporta con el resto usándolo como prueba, los grupos primero se desordenan y aleatoriamente con un argumento de semilla (%split) se elegiría el grupo para entrenamiento o prueba. La tercera forma de entrenamiento (Cross-validation) ha sido eligiendo grupos aleatorios de un número de terminado de instancias (en el campo folds) para pruebas y el resto de instancias para entrenamiento, este proceso se itera para poder utilizar todos los grupos como prueba. Los mejores resultados serían los obtenidos con el Cross-validation.

El segundo experimento presentó el uso del criterio del número de vecinos, un criterio del aprendizaje vago basado en la localidad de estos, en este caso variamos el argumento k para poder conseguir con que número de vecinos nos ofrecía mejores resultados. Pudimos comprobar que no nos sirve un número k ni mayor ni menor, sino que cada estudio propuesto tiene un k que ofrecerá mejores resultados que otro.

7. Bibliografía