upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

TEMA 3 – SELECCIÓN DE INSTANCIAS

Mikel Galar Idoate
mikel.galar@unavarra.es

Ciencia de datos con técnicas inteligentes – Pre-procesamiento
Experto Universitario en Ciencia de Datos y Big Data

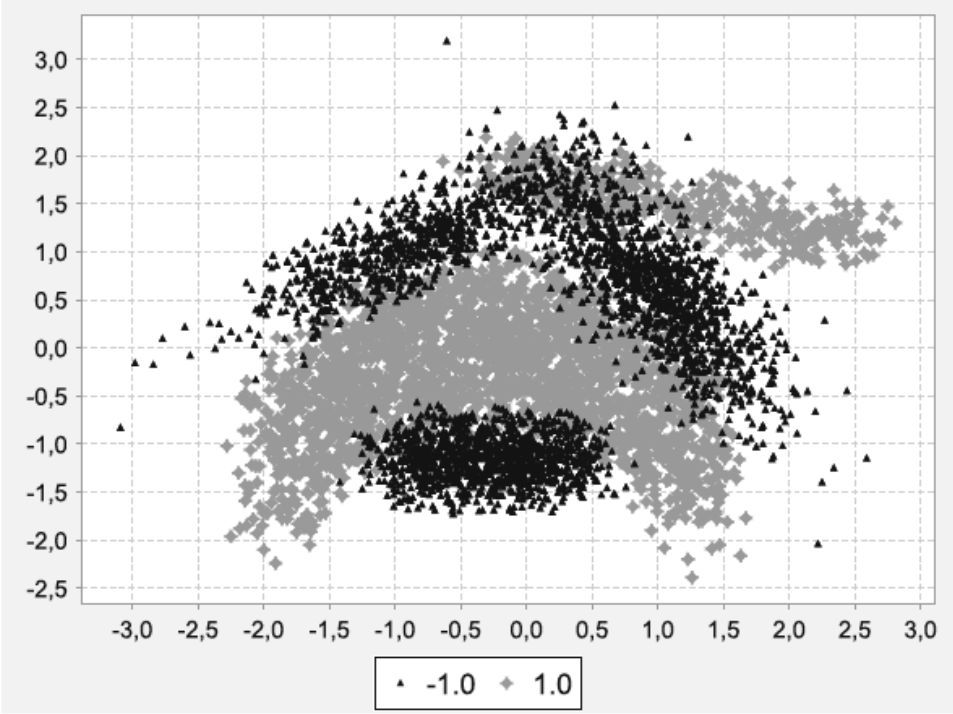
Índice

Índice

Introducción

- **Visualización del conjunto de datos Banana**

Dataset artificial
5300 ejemplos
2 clases

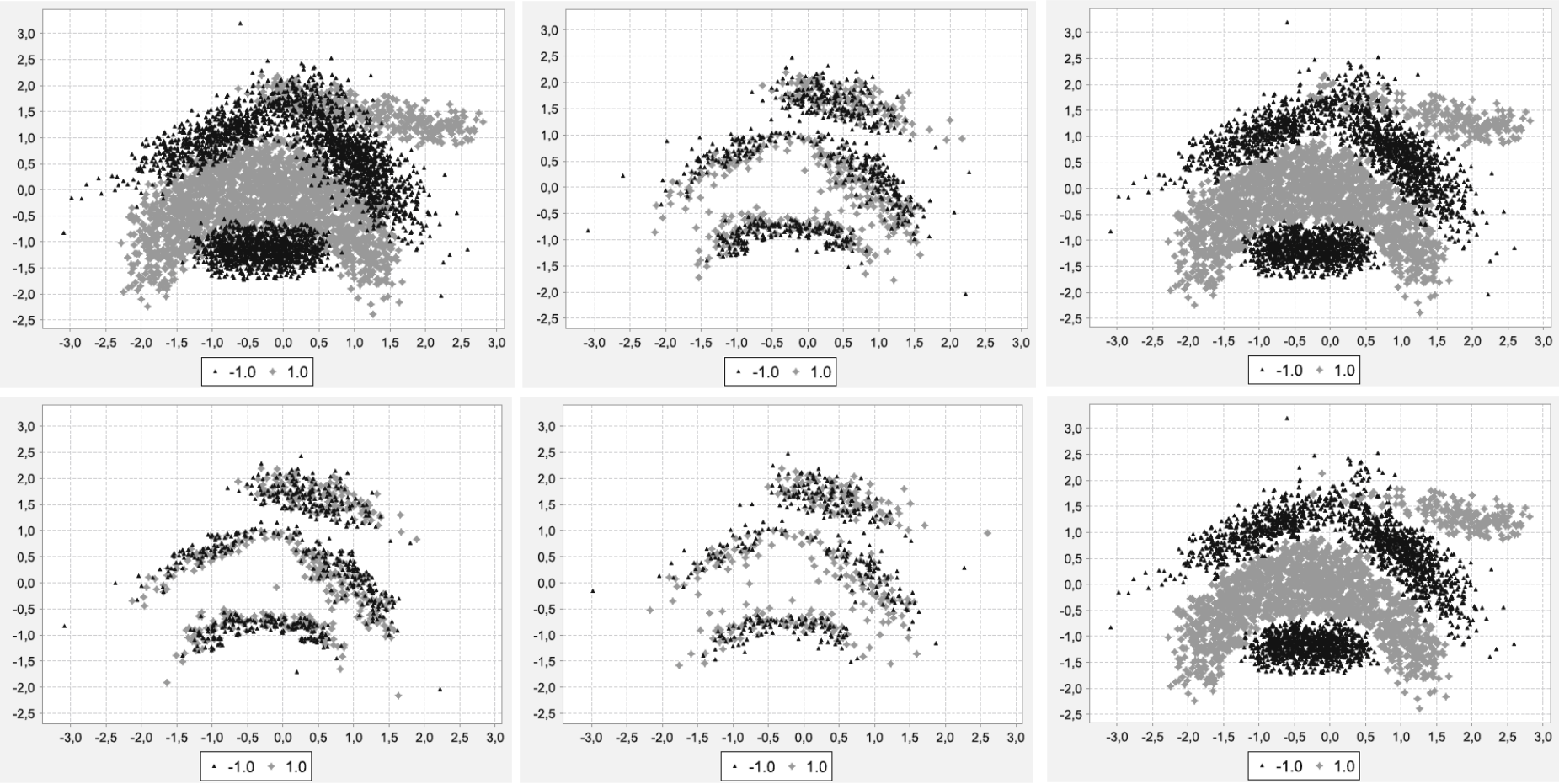


Banana Original (0.8751,0.7476)

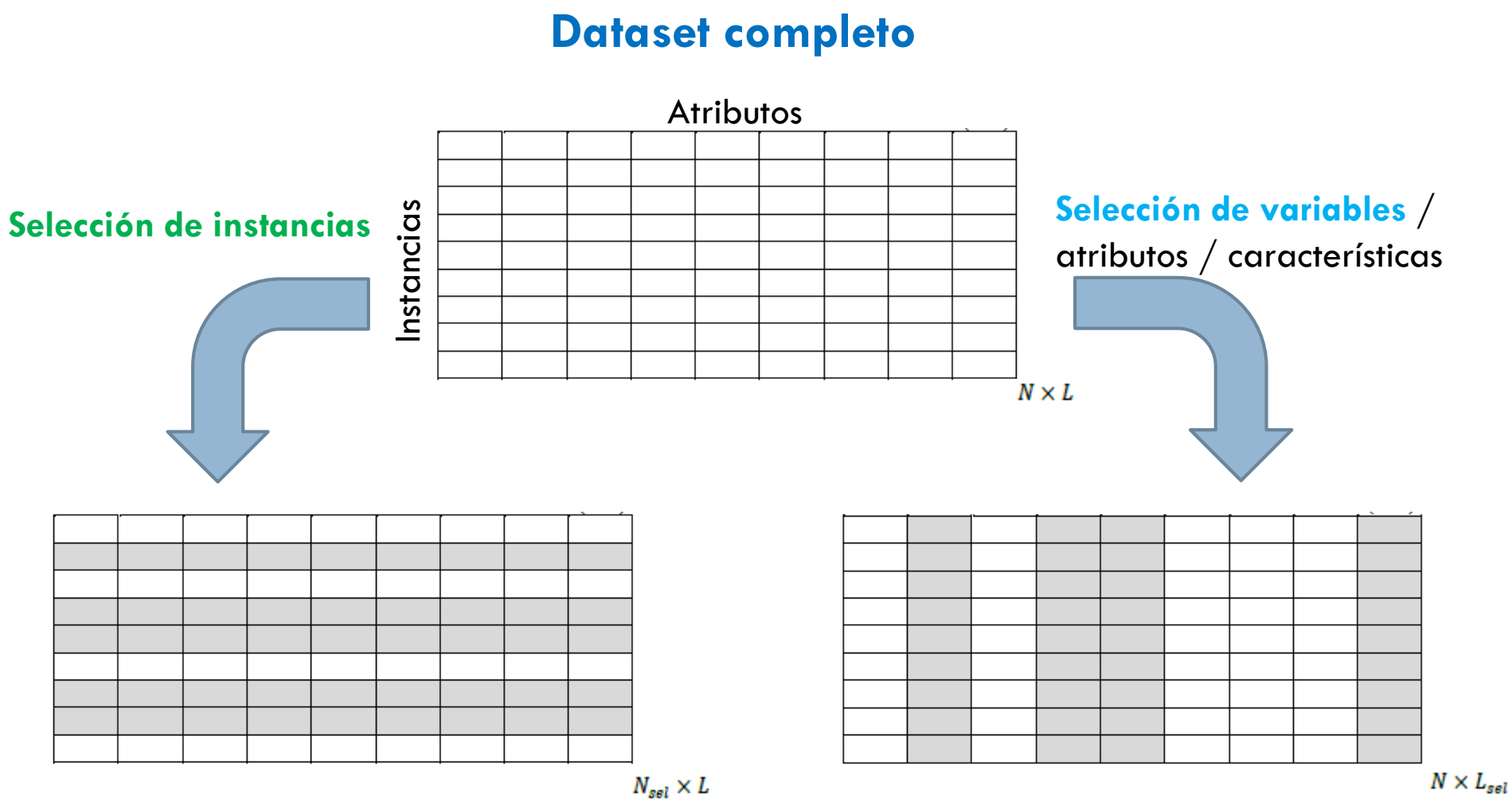
(Reduction rate, Test Accuracy, Test Kappa)

Introducción

- **Ejemplos de selección de instancias con Banana**



Introducción



Introducción

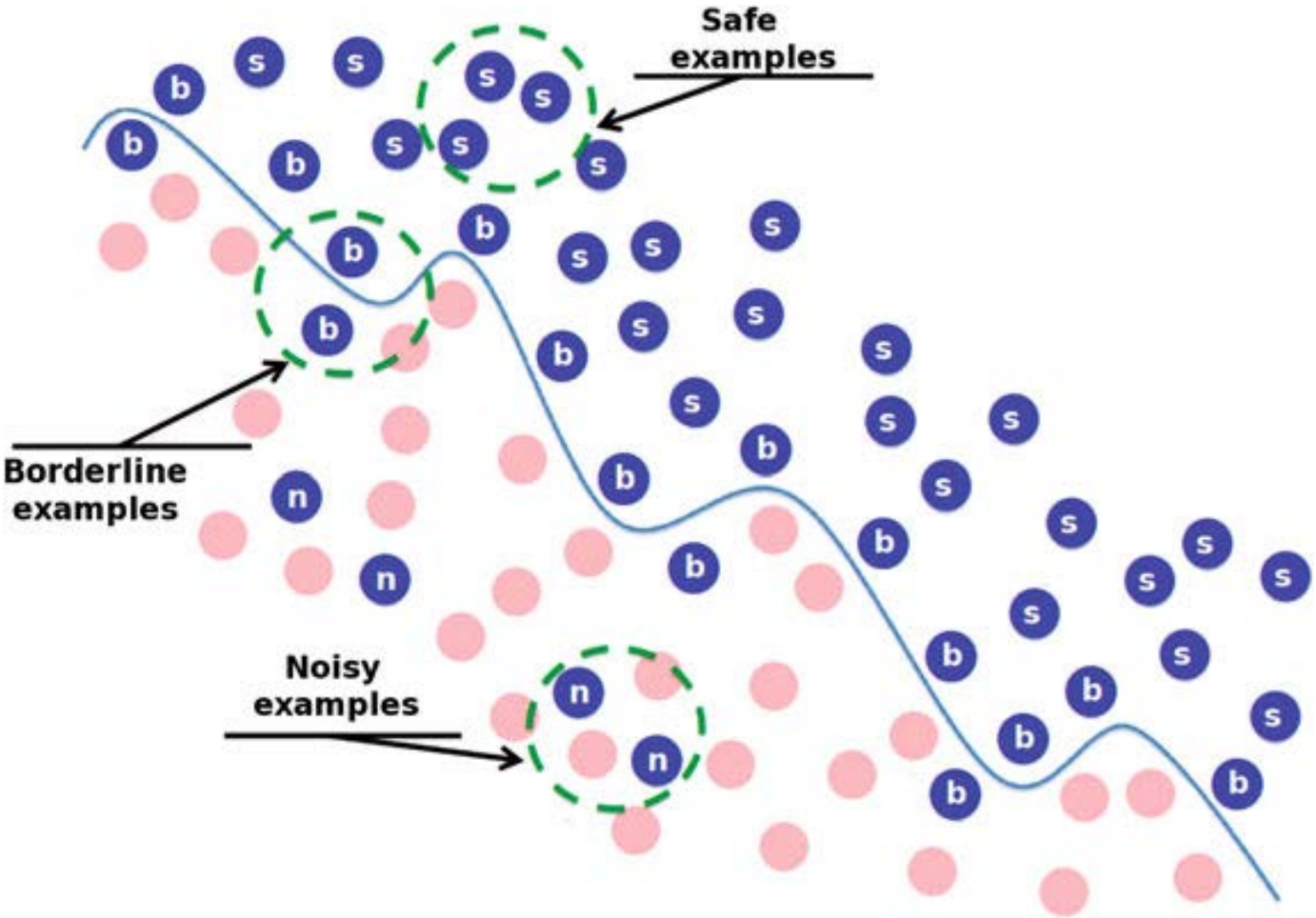
- **Selección de variables**
  - Reduce el tamaño del conjunto de datos
  - **Elimina características** redundantes o irrelevantes
- **Selección de instancias**
  - **Obtiene un subconjunto** del conjunto **de datos** disponibles (los reduce)
  - Nos debe permitir lograr el mismo objetivo en la aplicación de minería de datos que el que obtendríamos si utilizáramos el conjunto de datos original

Introducción

- **Selección de Instancias**
  - Elige **los ejemplos relevantes** para una aplicación y **lograr el máximo rendimiento**
    - **Menos datos**
      - Los algoritmos pueden aprender más rápidamente
    - **Mayor exactitud**
      - El clasificador generaliza mejor
    - Resultados **más simples**
      - Más fácil de entender

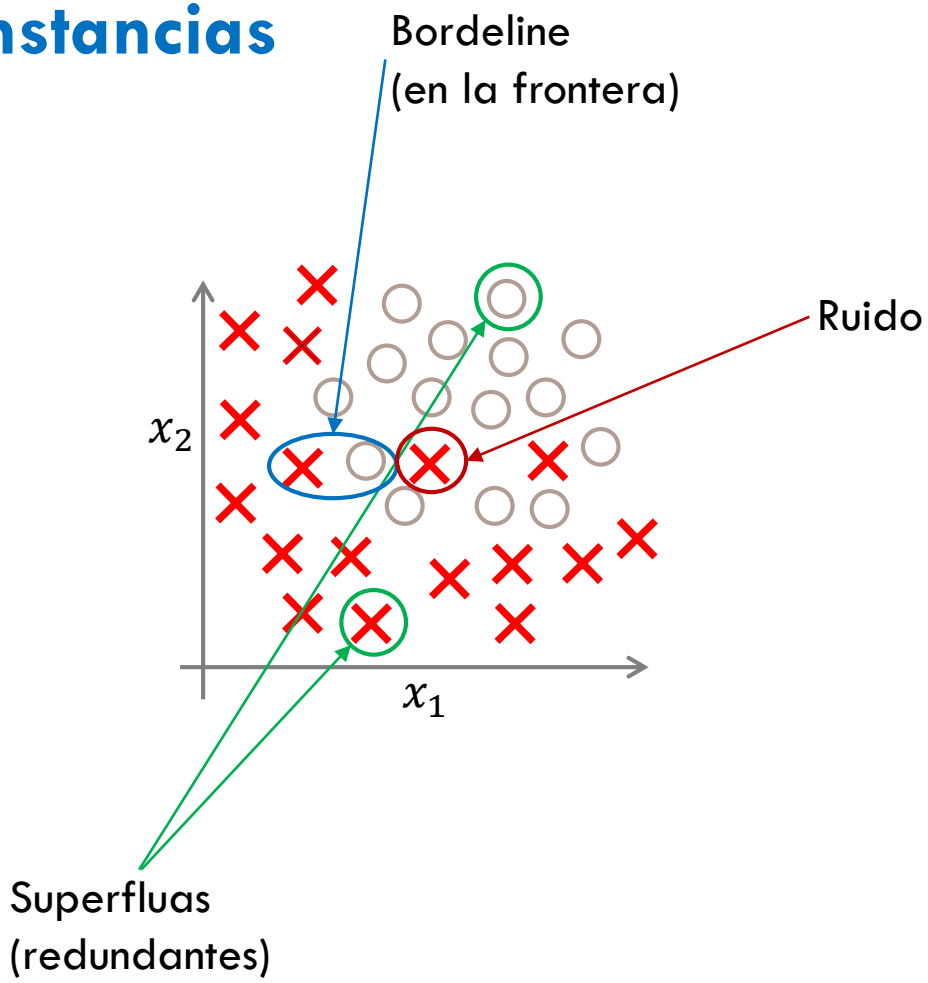
Introducción

- **Tipos de instancias**



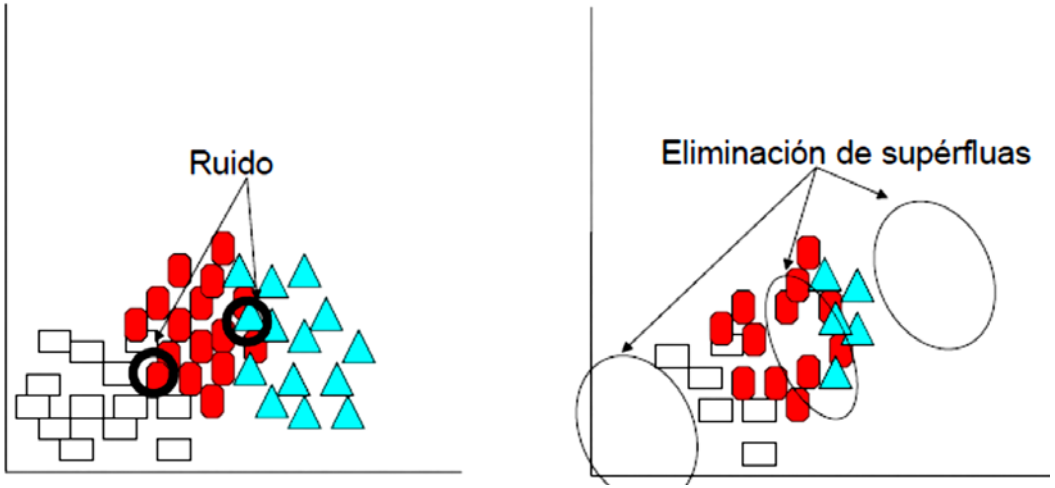
Introducción

- **Tipos de instancias**

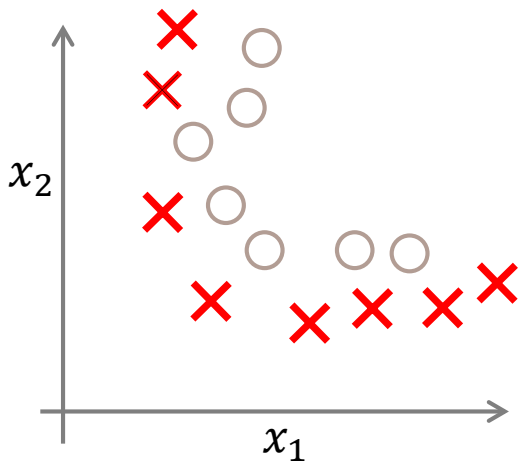


Introducción

- **Tipos de instancias**
  - **Superfluas**
    - **No son necesarias** para clasificar
    - Si las borramos, se decrementará el tiempo de clasificación
  - **Ruido** (o solapamiento)
    - **Confunden** al clasificador
    - Si las borramos, mejorará el porcentaje de aciertos esperado



Introducción

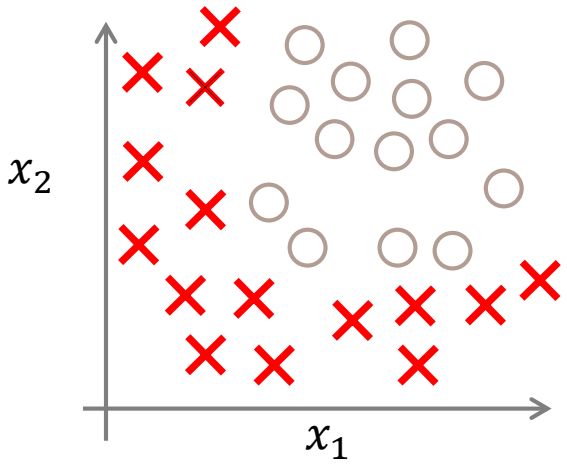


Condensación

Mantenemos instancias en las fronteras

Eliminamos instancias superfluas

Se suelen eliminar muchas instancias



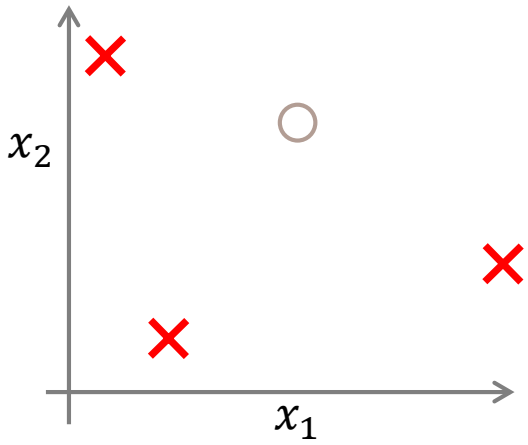
Edición

Eliminamos el ruido (instancias en las fronteras)

Más o menos extremo

Se suelen eliminar pocas instancias

Introducción



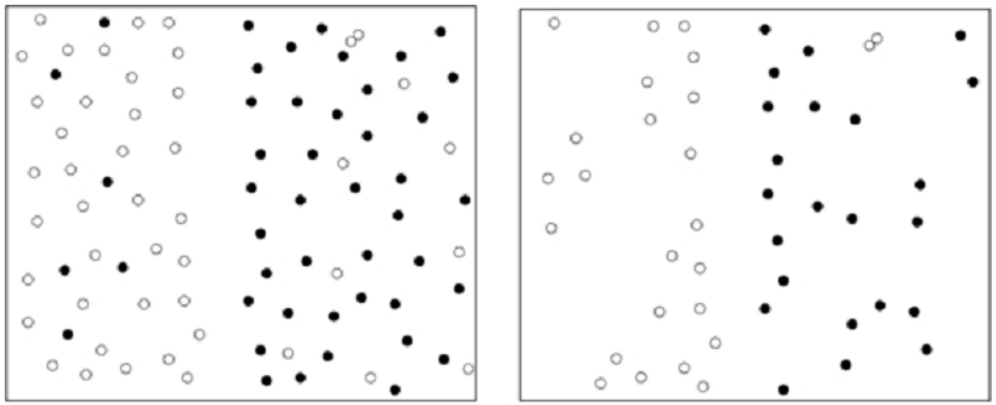
Híbridos

Buscamos el subconjunto de instancias menor con la mayor capacidad de generalización

Se elimina el ruido y las instancias superfluas

Introducción

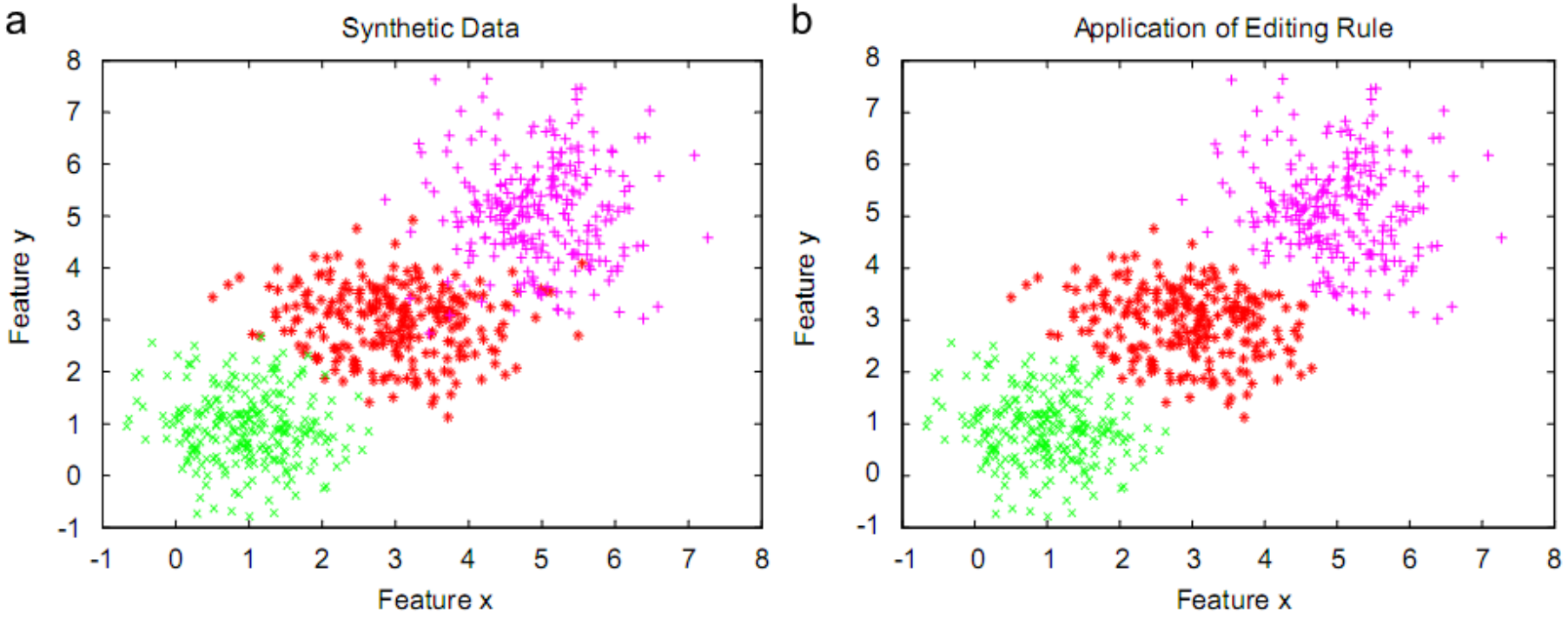
- **Ejemplo** típico de Edición
  - **ENN (Edited Nearest Neighbor) – Wilson**
    - Elimina una instancia si es clasificada incorrectamente por sus k vecinos más cercanos
      - Excepciones en el interior de una clase
      - Algunos puntos en la frontera (suaviza las fronteras)



Original ENN $(k = 1)$

Introducción

- **Ejemplo** típico de Edición
  - **ENN (Edited Nearest Neigbor) – Wilson**



Introducción

- **Selección de instancias / Instance Selection (IS)**
  - Realiza el **proceso complementario** a la **selección de características**
  - **Objetivo principal**
    - **Reducir el tamaño** de los datos seleccionando o **identificando los datos relevantes** entre todos los disponibles
    - Se trata de obtener **un subconjunto de los datos** que al ser aplicado sobre un algoritmo de minería de datos logre el **mismo resultado que se obtendría con el conjunto completo**

Introducción

- **IS vs. Remuestreo de datos**
  - **IS** realiza un **proceso inteligente** para categorizar las instancias de acuerdo al grado de irrelevancia o ruido
  - **Remuestreo: aleatorio**
- **Definición IS**
  - Un método de IS trata de obtener un subconjunto S del conjunto de entrenamiento TR tal que $S \subset TR$, S no contiene instancias superfluas y $Acc(S)$ es similar a $Acc(TR)$
    - Donde $Acc(X)$ es el porcentaje de acierto obtenido utilizando X como conjunto de entrenamiento

Introducción

- ¿**Qué aporta** la selección de instancias?
  - Permite utilizar algoritmos de minería de datos con **grandes cantidades de datos**
    - Que no podrían aplicarse sobre el conjunto completo
  - Permite **limpiar** el conjunto de datos
    - Eliminando instancias redundantes o que son ruido, **mejorando la calidad de los datos**

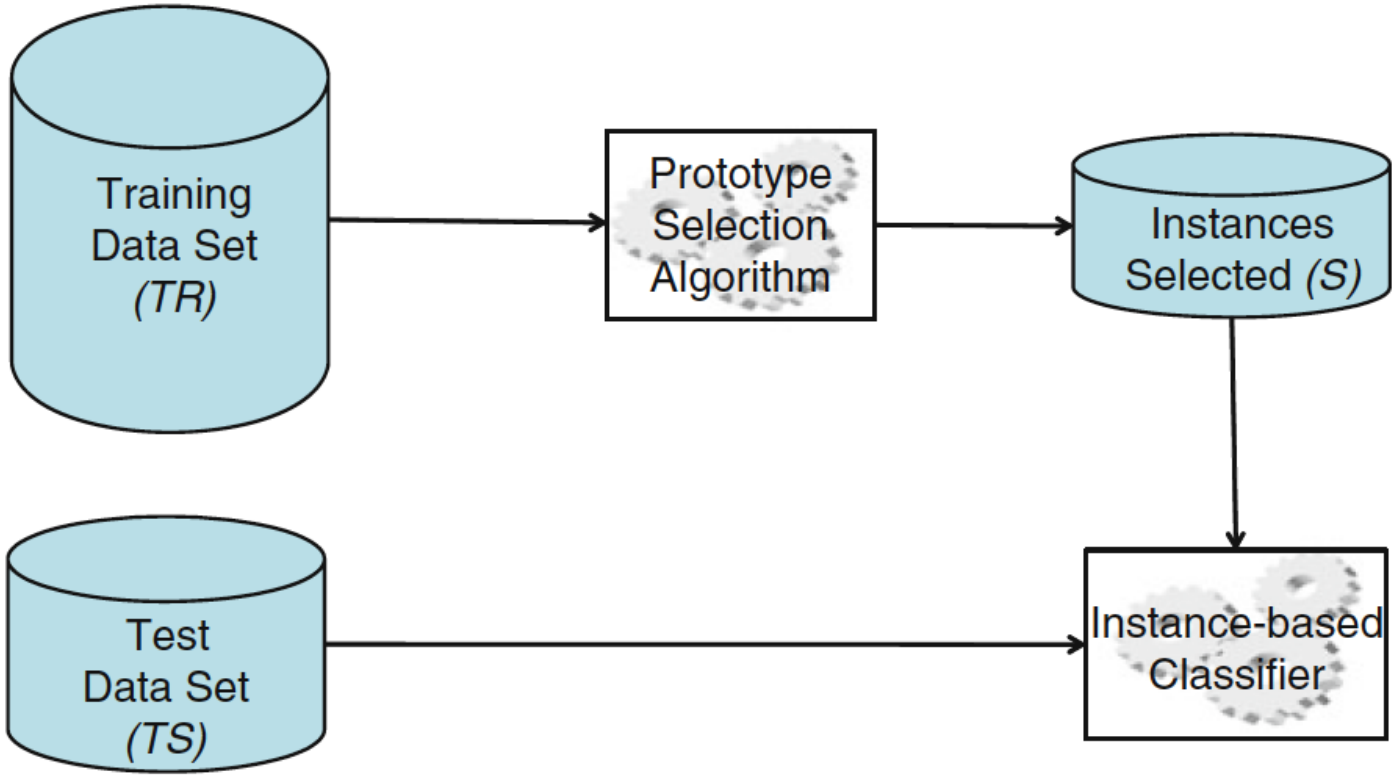
Índice

Training Set Selection vs. Prototype Selection

- Existen **diferentes términos** para hablar de **IS**
  - Obtener los datos más relevantes del conjunto de entrenamiento
  - **Instance Selection – el más general**
    - Pensado para trabajar con algoritmos como árboles de decisión, redes neuronales o SVMs
  - Vamos a distinguir **dos tipos**
    - **Prototype Selection (PS)**
      - La selección se centra en obtener un buen subconjunto para clasificadores basados en instancias (**kNN**)
    - **Training Set Selection (TSS)**
      - La selección se realiza para cualquier clasificador

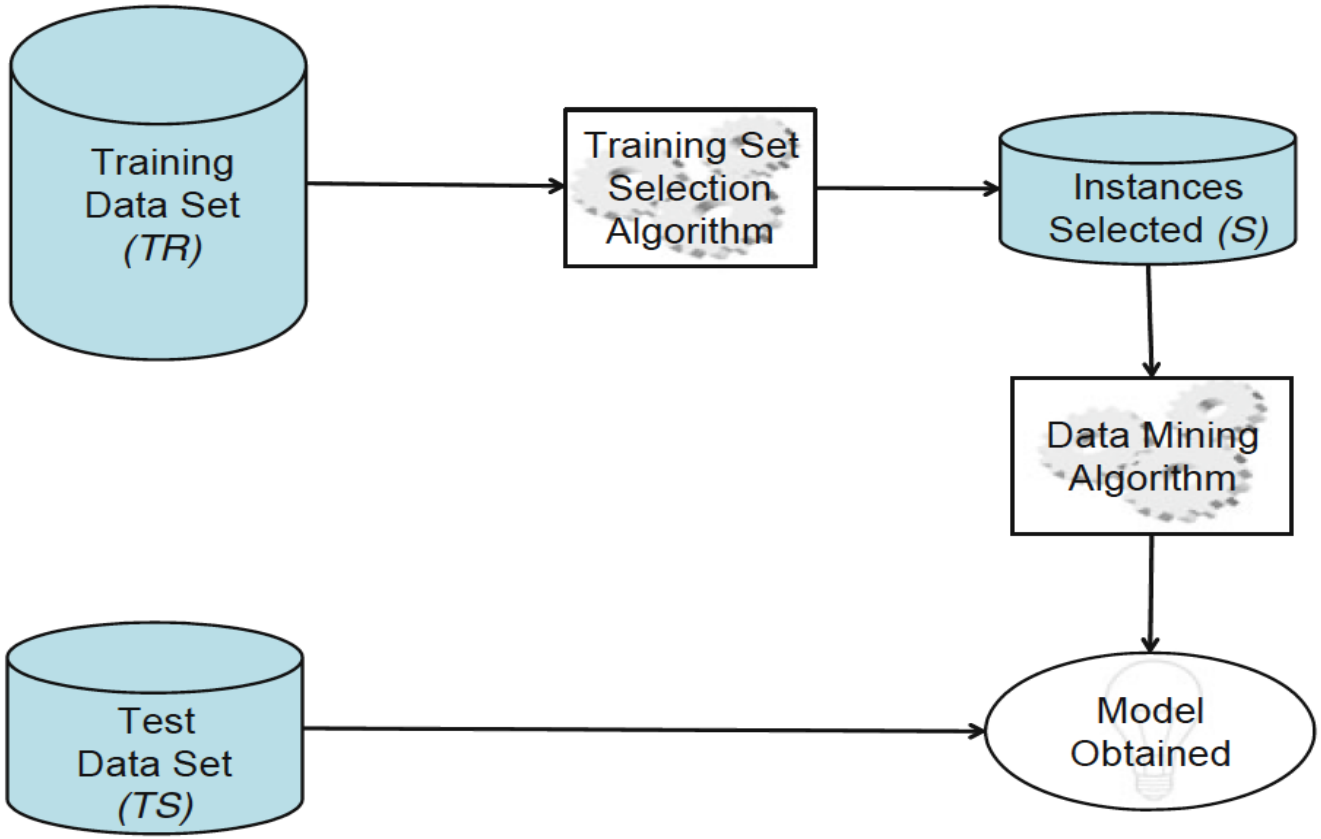
Training Set Selection vs. Prototype Selection

- **Prototype Selection**



Training Set Selection vs. Prototype Selection

- **Training Set Selection**



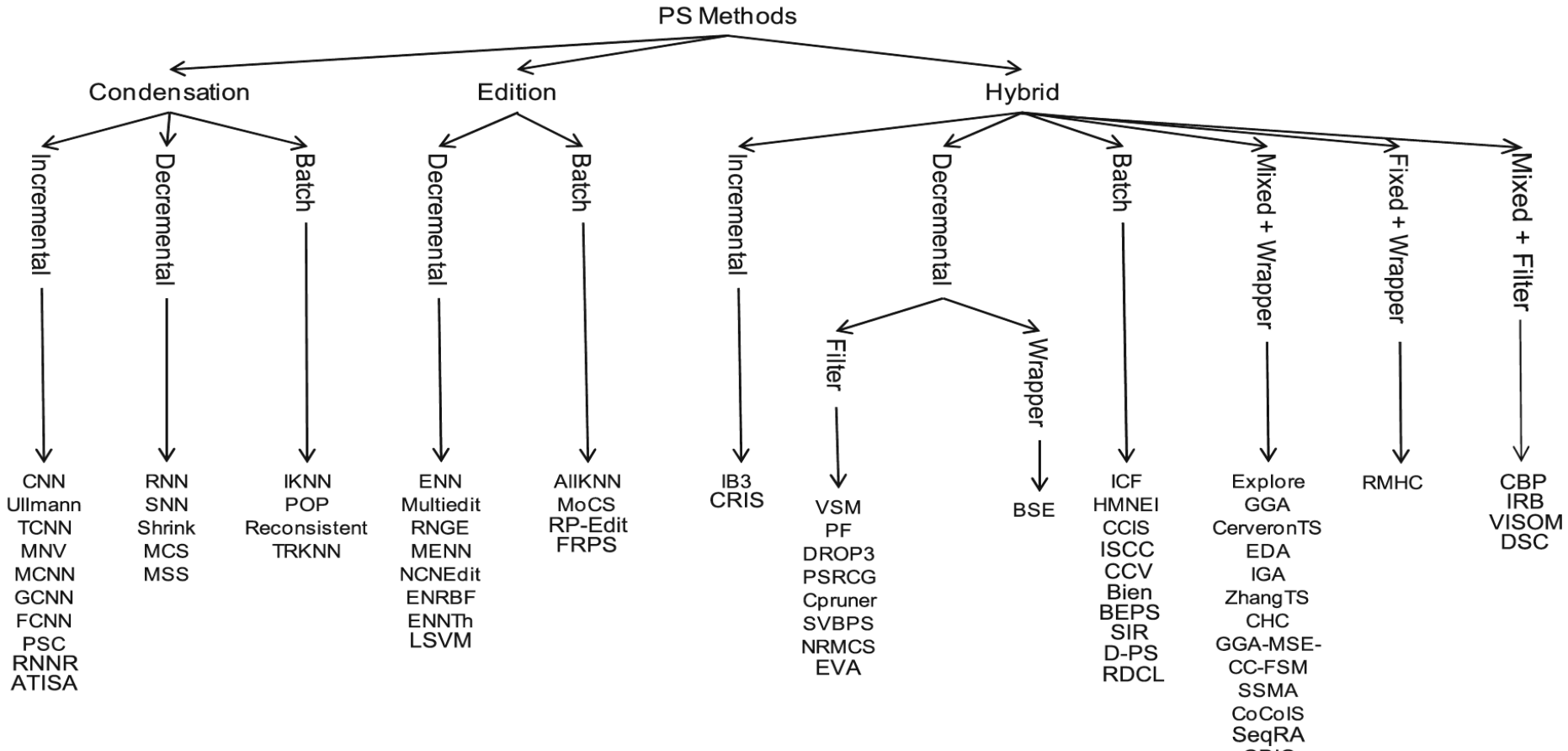
Training Set Selection vs. Prototype Selection

- Nos vamos a centrar en **Prototype Selection**
  - **90% de la literatura** existente
  - Los **métodos tipo filtro suelen funcionar bien como TSS**
    - No eliminan tantos datos
    - En TSS no se puede reducir tanto el conjunto ya que los modelos necesitan más instancias para aprender

Índice

Modelos de selección de prototipos

- **Taxonomía (tipología, dirección de búsqueda y evaluación de la búsqueda)**



Modelos de selección de prototipos

- **Tipo de selección**
  - **Condensación**
    - Mantiene las instancias cercanas a las fronteras de decisión (*border points*)
  - **Edición**
    - Busca eliminar las instancias en las fronteras de decisión (*border points*)
    - Eliminan instancias que son ruido o que no estén de acuerdo con su vecindad
  - **Híbridos**
    - Tratan de buscar el subconjunto S más pequeño que mantiene o incluso aumenta la capacidad de generalización (precisión) sobre los datos de test

Modelos de selección de prototipos

- **Dirección de la búsqueda**
  - **Incremental**
    - Comienza con un subconjunto S vacío y añade cada instancia de TR en S si satisface cierto criterio
      - El orden es importante
      - Hay modelos independientes del orden – requieren de todo el conjunto, no son 100% incrementales
      - Decisiones con poca información – errores más probables
  - **Decremental**
    - Comienza con $S = TR$ y después busca las instancias a eliminar de S
    - Importancia del orden – Pero todos los ejemplos disponibles al inicio
    - Mayor coste computacional – puede merecer la pena
  - **Batch**
    - Decremental eliminando más de una instancia
    - Se decide qué instancias eliminar y se eliminan todas a la vez

Modelos de selección de prototipos

- **Dirección de la búsqueda**
  - **Mixed**
    - Comienza con un subconjunto S pre-seleccionado e iterativamente añade o eliminan instancias que satisfacen cierta condición
      - Permite rectificaciones
  - **Fixed**
    - Subfamilia de mixed donde el número de adiciones y substracciones es el mismo
    - El número de instancias está prefijado

Modelos de selección de prototipos

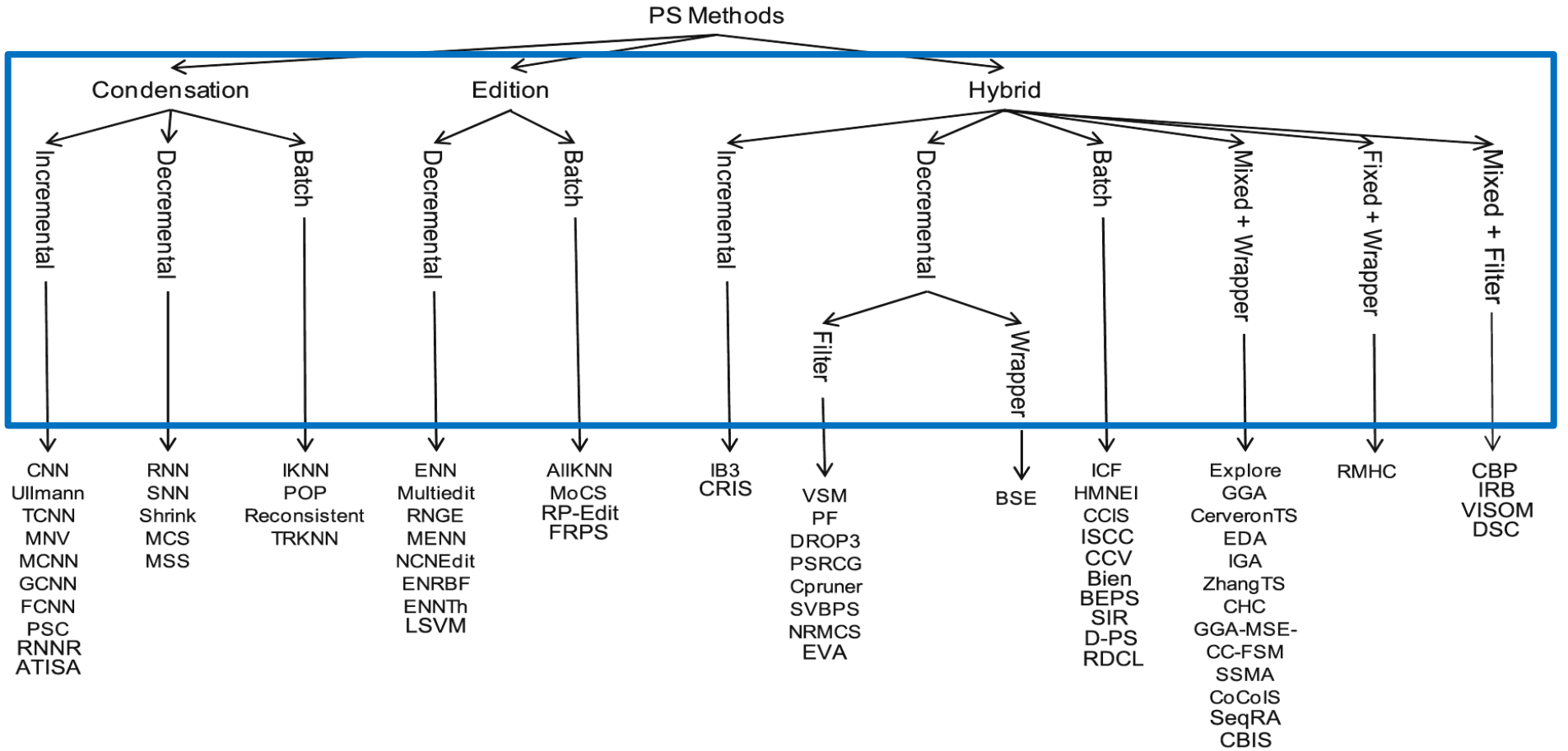
- **Evaluación de la búsqueda**
  - **Filtro**
    - Cuando kNN se usa sobre parte de los datos para determinar el criterio de añadir o eliminar instancias
    - No se utiliza el esquema de validación leave-one-out para obtener una buena estimación de la capacidad de generalización
  - **Wrapper**
    - Cuando kNN se usa para evaluar el conjunto de entrenamiento completo con leave-one-out
    - Se obtiene una buen estimación de la capacidad de generalización
      - Mejor precisión sobre los datos de test

Modelos de selección de prototipos

- **Criterios para comparar los métodos**
  - **Reducción en el almacenamiento**
    - Uno de los principales objetivos
    - Se acelera la clasificación
  - **Capacidad de generalización** (precisión en test)
    - No deberíamos perder precisión
  - **Coste computacional** (tiempo)
    - No es tan importante siempre que sea posible ejecutarlo

Modelos de selección de prototipos

- **Taxonomía (tipología, dirección de búsqueda y evaluación de la búsqueda)**



Modelos de selección de prototipos

Métodos de Selección de Prototipos (1):

Condensed nearest neighbor	CNN
Reduced nearest neighbor	RNN
Edited nearest neighbor	ENN
No name specified	Ullmann
Selective nearest neighbor	SNN
Repeated edited Nearest neighbor	RENN
All-KNN	**AllKNN**
Tomek condensed nearest neighbor	TCNN
Mutual neighborhood value	MNV
MultiEdit	MultiEdit
Shrink	Shrink
Instance based 2	IB2
Instance based 3	**IB3**
Monte carlo 1	MC1
Random mutation hill climbing	**RMHC**
Minimal consistent set	MCS

Modelos de selección de prototipos

Métodos de Selección de Prototipos (2):

Encoding length heuristic	ELH
Encoding length grow	ELGrow
Explore	**Explore**
Model class selection	MoCS
Variable similarity metric	VSM
Gabriel graph editing	GGE
Relative Neighborhood Graph Editing	**RNGE**
Polyline functions	PF
Generational genetic algorithm	GGA
Modified edited nearest neighbor	MENN
Decremental reduction optimization procedure 1	DROP1
Decremental reduction optimization procedure 2	DROP2
Decremental reduction optimization procedure 3	**DROP3**
Decremental reduction optimization procedure 4	DROP4
Decremental reduction optimization procedure 5	DROP5
Decremental encoding length	DEL
Estimation of distribution algorithm	EDA
Tabu search	CerveronTS
Iterative case filtering	ICF
Modified condensed nearest neighbor	MCNN
Intelligent genetic algorithm	IGA
Prototype selection using relative certainty gain	PSRCG
Improved KNN	IKNN

Modelos de selección de prototipos

Métodos de Selección de Prototipos (3):

Tabu search	ZhangTS
Iterative maximal nearest centroid neighbor	Iterative Ma:
Reconsistent	**Reconsisten**
C-Pruner	CPruner
Steady-state genetic algorithm	SSGA
Population based incremental learning	PBIL
CHC evolutionary algorithm	**CHC**
Patterns by ordered projections	POP
Nearest centroid neighbor edition	NCNEdit
Edited normalized radial basis function	**ENRBF**
Edited normalized radial basis function 2	ENRBF2
Edited nearest neighbor estimating class probabilistic	ENNProb
Edited nearest neighbor estimating	**ENNTh**
Class probabilistic and threshold	
Support vector based prototype selection	SVBPS
Backward sequential edition	BSE
Modified selective subset	MSS
Generalized condensed nearest neighbor	GCNN

Modelos de selección de prototipos

Métodos de Selección de Prototipos (4):

Fast condensed nearest neighbor 1	**FCNN**
Fast condensed nearest neighbor 2	FCNN2
Fast condensed nearest neighbor 3	FCNN3
Fast condensed nearest neighbor 4	FCNN4
Noise removing based on minimal consistent set	NRMCS
Genetic algorithm based on mean square error,	GA-MSE-CC-FSM
Clustered crossover and fast smart mutation	
Steady-state memetic algorithm	SSMA
Hit miss network C	HMNC
Hit miss network edition	HMNE
Hit miss network edition iterative	**HMNEI**
Template reduction for KNN	TRKNN
Prototype selection based on clustering	PSC
Class conditional instance selection	CCIS
Cooperative coevolutionary instance selection	CoCoIS

Modelos de selección de prototipos

Métodos de Selección de Prototipos (5):

Instance selection based on classification contribution	ISCC
Bayesian instance selection	EVA
Reward-Punishment editing	RP-Edit
Complete cross validation functional prototype selection	CCV
Sequential reduction algorithm	SeqRA
Local support vector machines noise reduction	LSVM
No name specified	Bien
Reverse nearest neighbor reduction	RNNR
Border-Edge pattern selection	BEPS
Class boundary preserving algorithm	CBP
Cluster-Based instance selectio	CBIS
RDCL profiling	RDCL
Multi-Selection genetic algorithm	MSGA

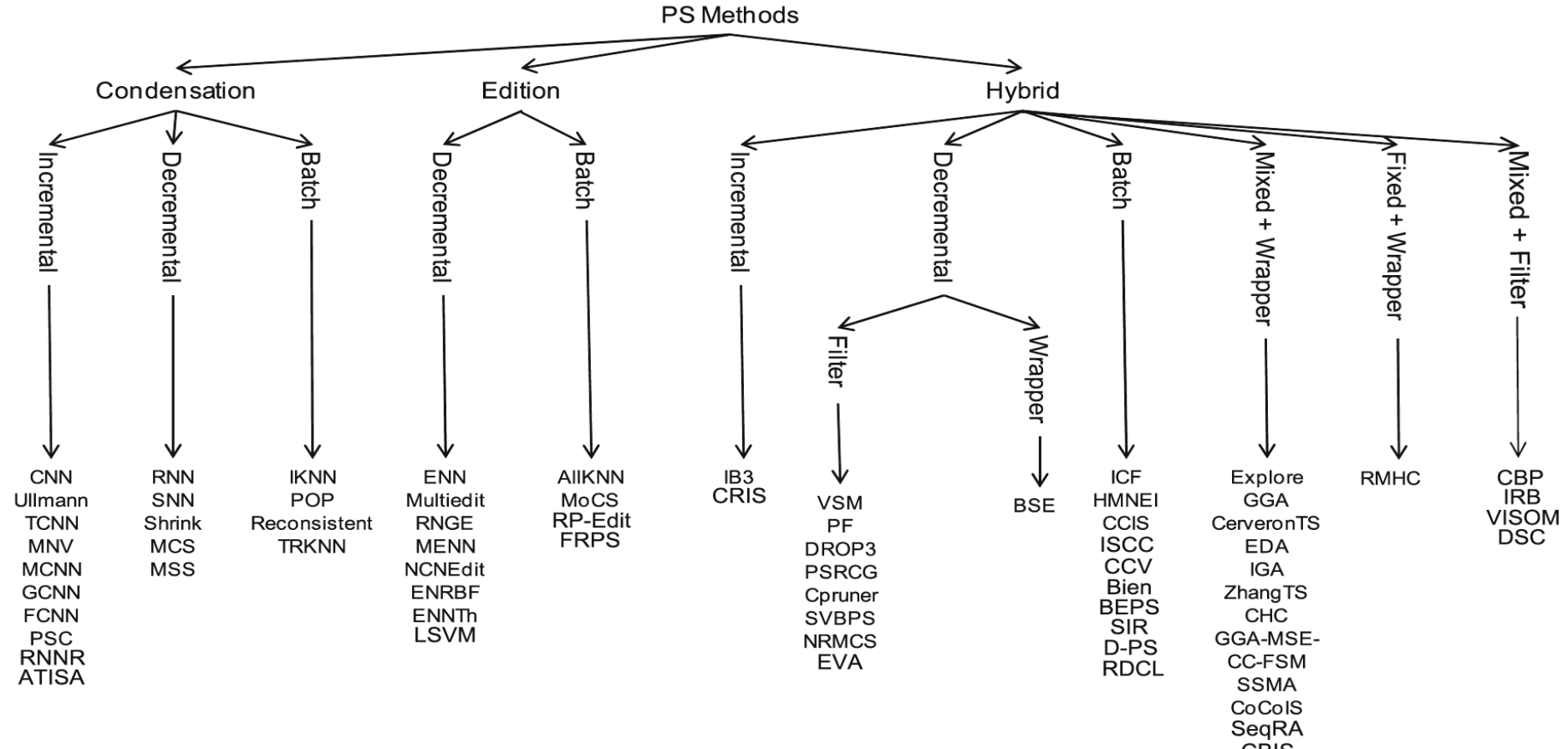
Modelos de selección de prototipos

Métodos de Selección de Prototipos (6):

Ant colony prototype reduction	Ant-PR
Spectral instance reduction	SIR
Competence enhancement by Ranking-based instance selection	CRIS
Discriminative prototype selection	D-PS
Adaptive threshold-based instance selection algorithm	ATISA
InstanceRank based on borders for instance selection	IRB
Visualization-Induced self-organizing map for prototype reduction	VISOM
Support vector oriented instance selection	SVOIS
Dominant set clustering prototype selection	DSC
Fuzzy rough prototype selection	FRPS

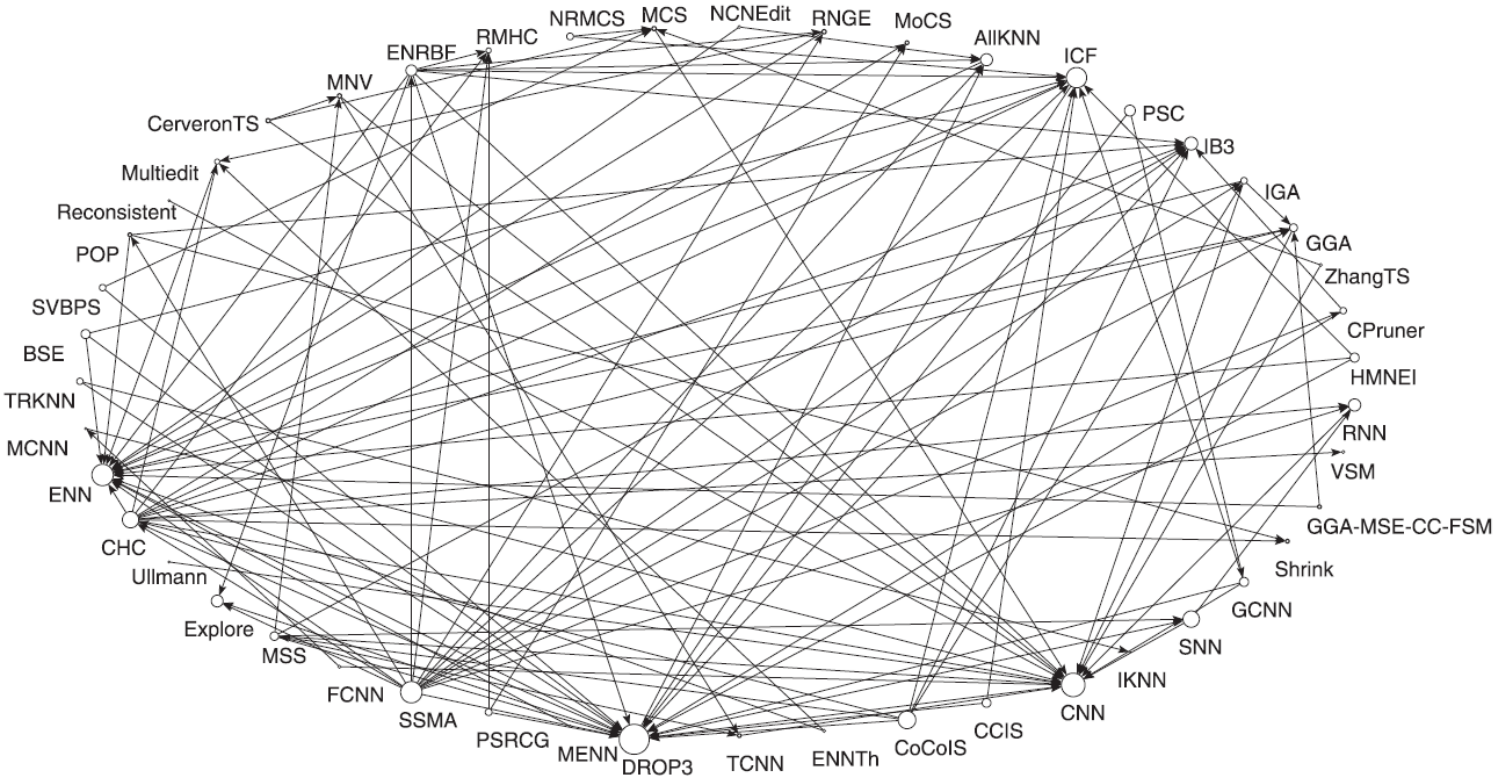
Modelos de selección de prototipos

- **Taxonomía (tipología, dirección de búsqueda y evaluación de la búsqueda)**



Modelos de selección de prototipos

- **Red de comparación entre métodos**



Índice

Descripción de los métodos

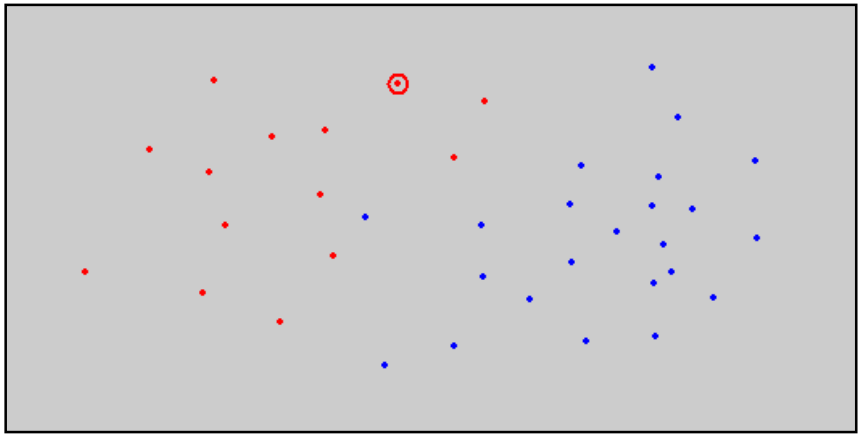
- **Condensación**
  - **Condensed Nearest Neighbor (CNN)**
    - Busca el subconjunto $S \subset TR$ tal que todos los ejemplos de TR están más cercanos a un ejemplo de S de la misma clase que a otro de otra clase
      - Es decir, todos los ejemplos en TR se aciertan usando S

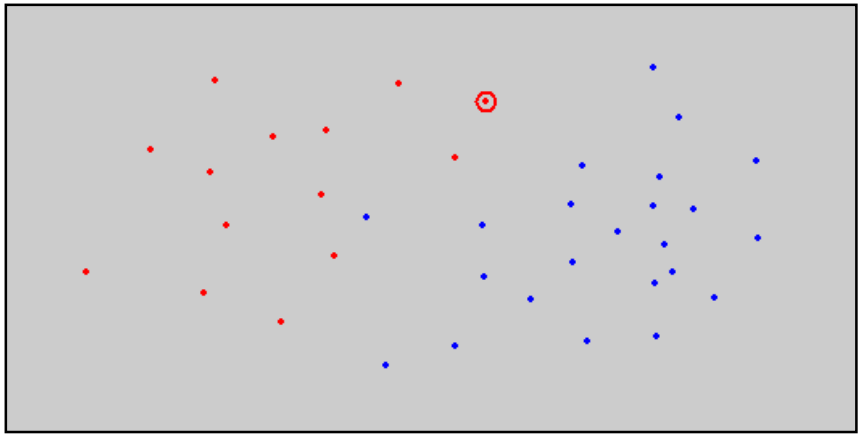
1. Inicializar S con un ejemplo aleatorio de cada clase del TR
2. Clasificar cada instancia en TR usando las instancias en S
   - Si se falla, añadir a S (se asegura que será clasificada correctamente)
3. Repetir hasta que todos los ejemplos en TR sean acertados utilizando S

Descripción de los métodos

- **Condensación**
  - **Condensed Nearest Neighbor (CNN)**

1. Inicializar S con un ejemplo aleatorio de cada clase del TR
2. Clasificar cada instancia en TR usando las instancias en S
   - Si se falla, añadir a S (se asegura que será clasificada correctamente)
3. Repetir hasta que todos los ejemplos en TR sean acertados utilizando S



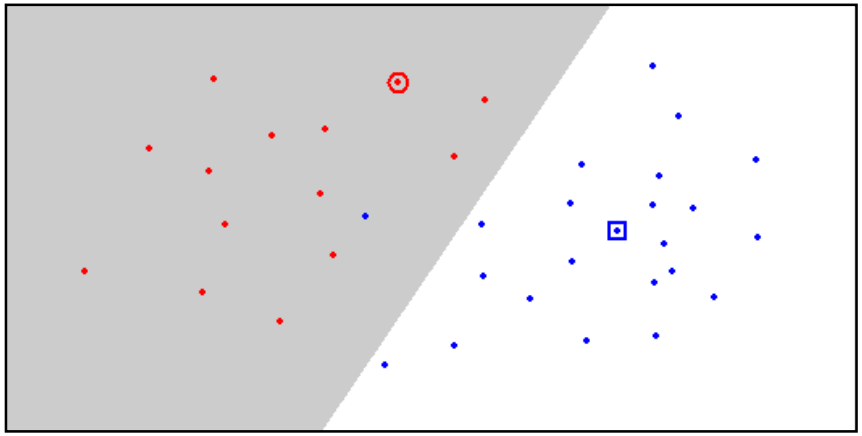


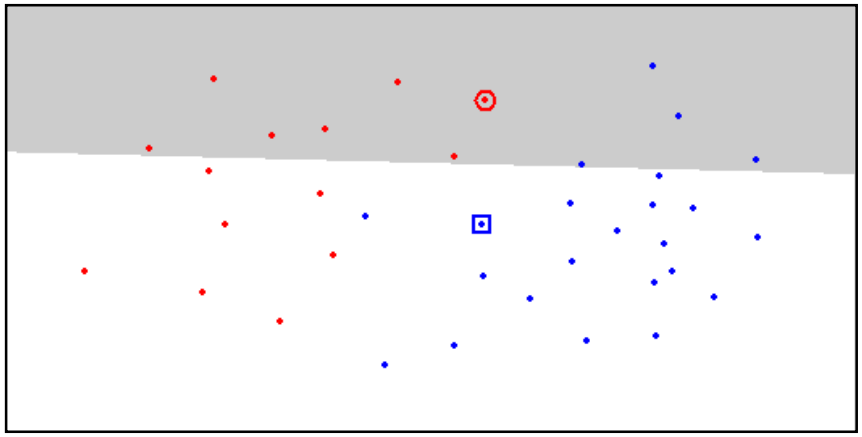
Descripción de los métodos

Condensación

Condensed Nearest Neighbor (CNN)

1. Inicializar S con un ejemplo aleatorio de cada clase del TR
2. Clasificar cada instancia en TR usando las instancias en S
   - Si se falla, añadir a S (se asegura que será clasificada correctamente)
3. Repetir hasta que todos los ejemplos en TR sean acertados utilizando S



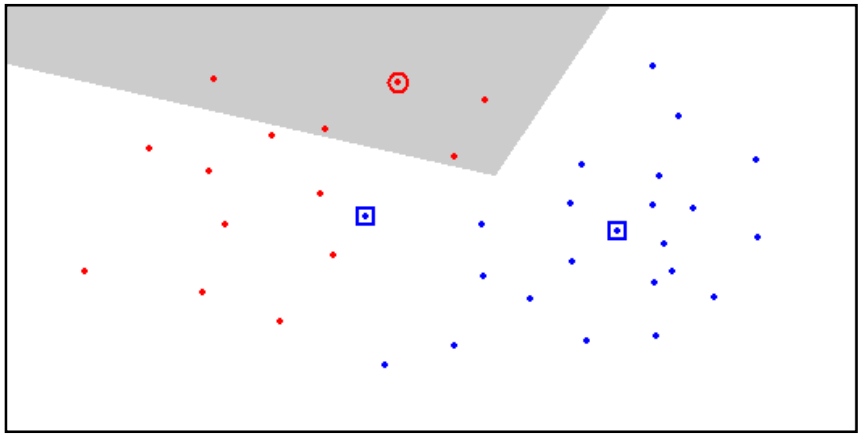


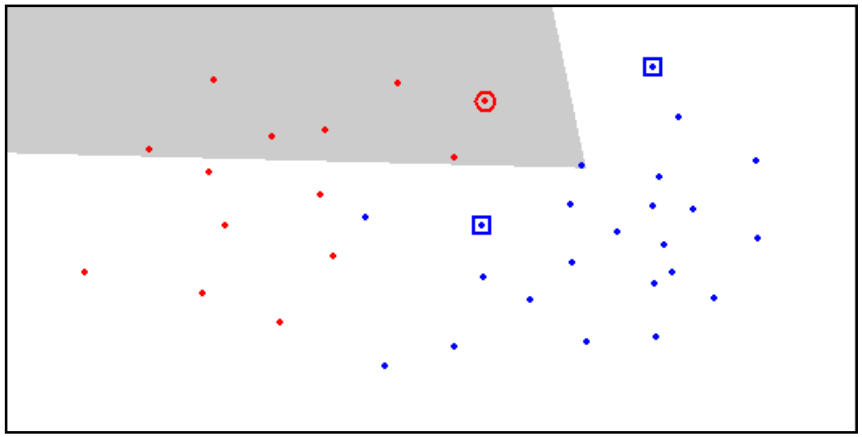
Descripción de los métodos

Condensación

Condensed Nearest Neighbor (CNN)

1. Inicializar S con un ejemplo aleatorio de cada clase del TR
2. Clasificar cada instancia en TR usando las instancias en S
   - Si se falla, añadir a S (se asegura que será clasificada correctamente)
3. Repetir hasta que todos los ejemplos en TR sean acertados utilizando S



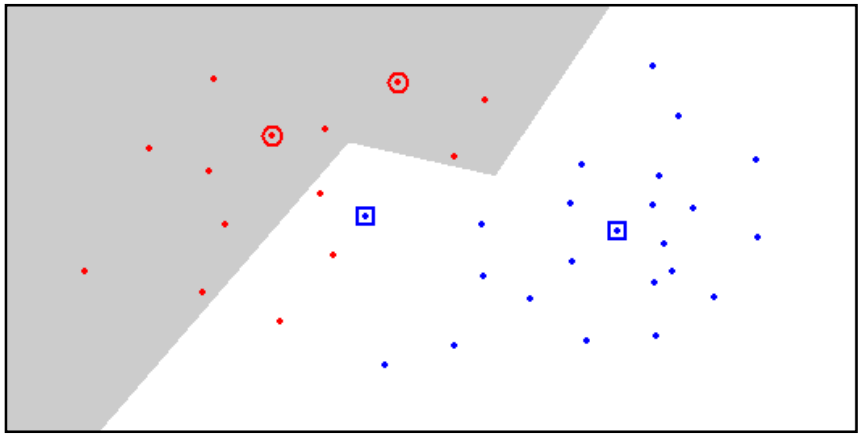


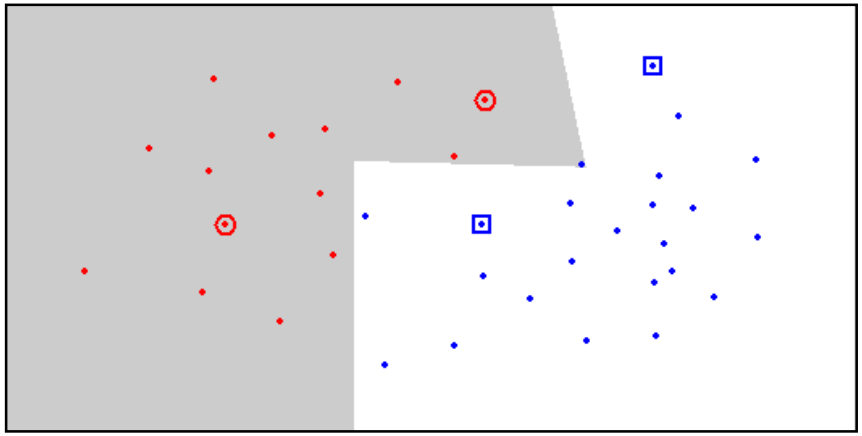
Descripción de los métodos

Condensación

Condensed Nearest Neighbor (CNN)

1. Inicializar S con un ejemplo aleatorio de cada clase del TR
2. Clasificar cada instancia en TR usando las instancias en S
   - Si se falla, añadir a S (se asegura que será clasificada correctamente)
3. Repetir hasta que todos los ejemplos en TR sean acertados utilizando S



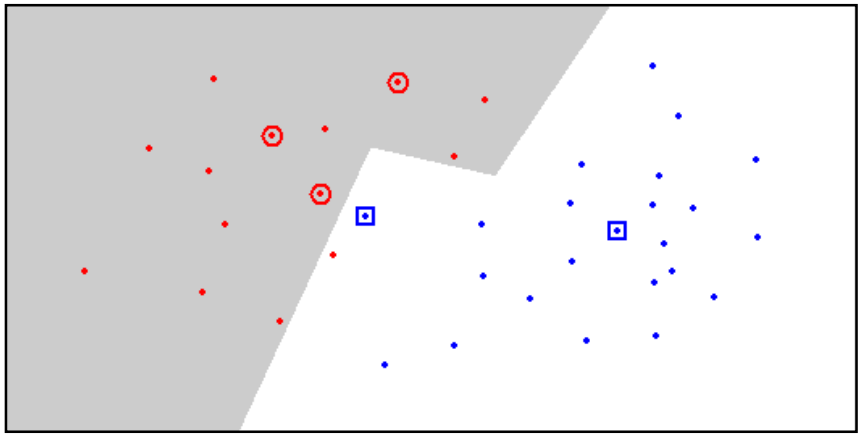


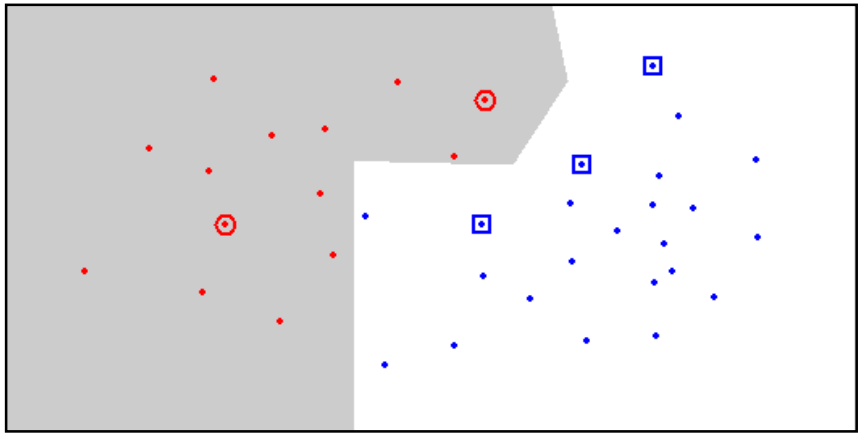
Descripción de los métodos

Condensación

Condensed Nearest Neighbor (CNN)

1. Inicializar S con un ejemplo aleatorio de cada clase del TR
2. Clasificar cada instancia en TR usando las instancias en S
   - Si se falla, añadir a S (se asegura que será clasificada correctamente)
3. Repetir hasta que todos los ejemplos en TR sean acertados utilizando S



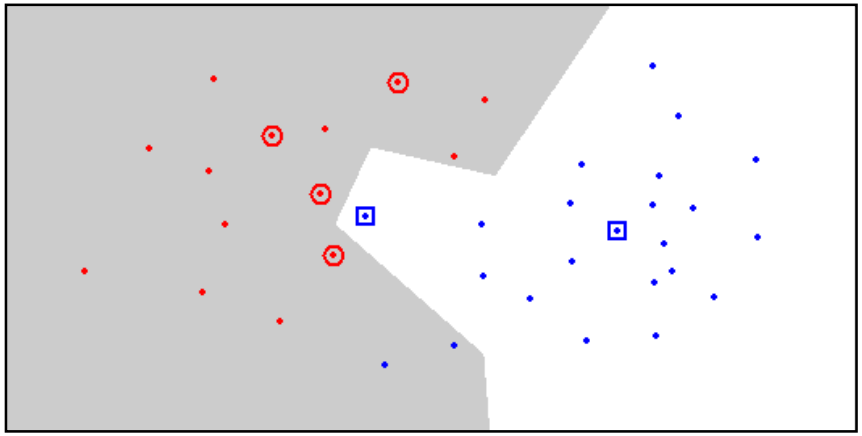


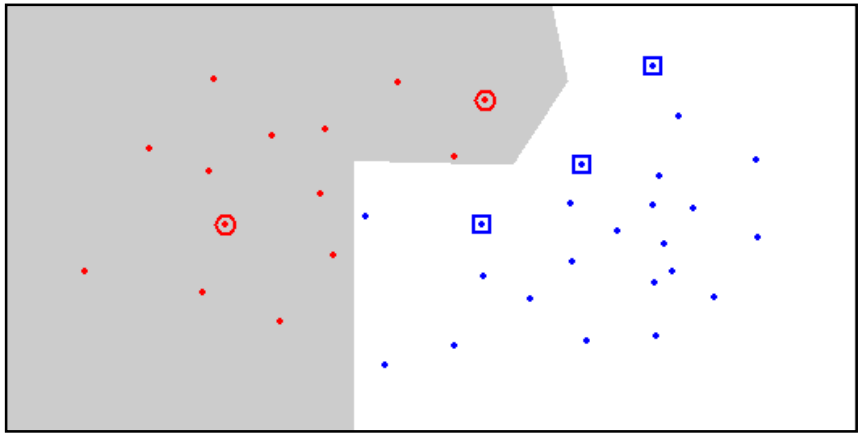
Descripción de los métodos

Condensación

Condensed Nearest Neighbor (CNN)

1. Inicializar S con un ejemplo aleatorio de cada clase del TR
2. Clasificar cada instancia en TR usando las instancias en S
   - Si se falla, añadir a S (se asegura que será clasificada correctamente)
3. Repetir hasta que todos los ejemplos en TR sean acertados utilizando S



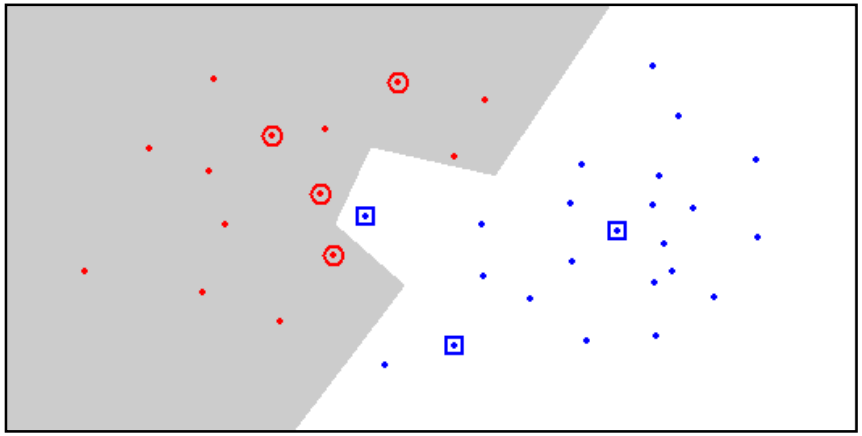


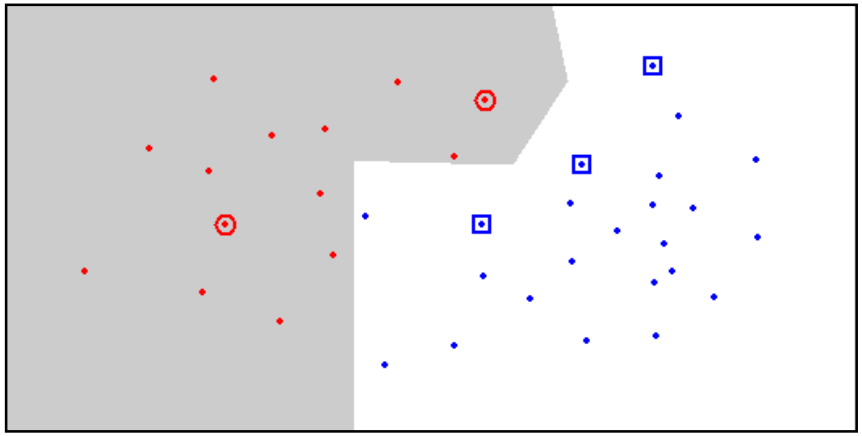
Descripción de los métodos

Condensación

Condensed Nearest Neighbor (CNN)

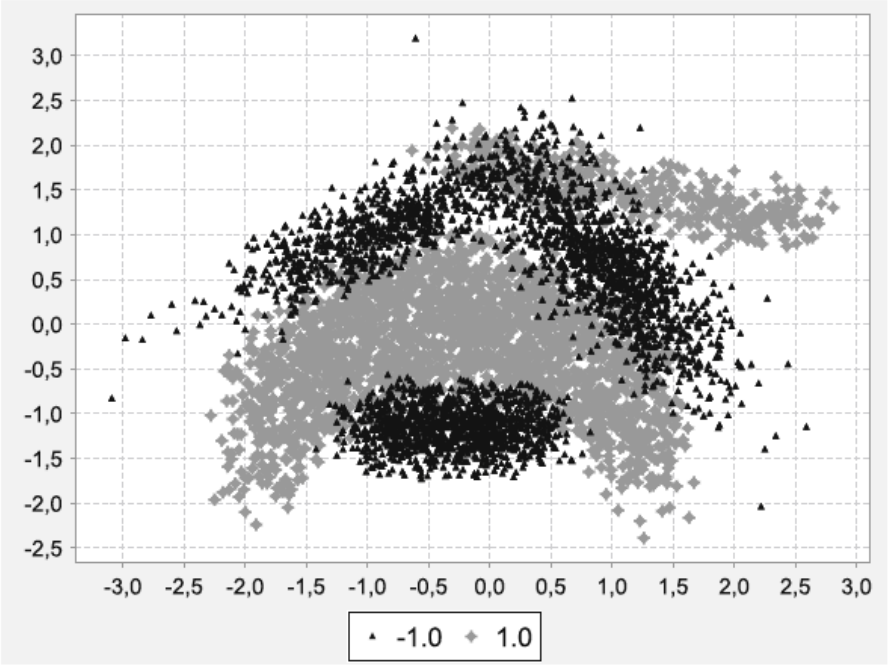
1. Inicializar S con un ejemplo aleatorio de cada clase del TR
2. Clasificar cada instancia en TR usando las instancias en S
   - Si se falla, añadir a S (se asegura que será clasificada correctamente)
3. Repetir hasta que todos los ejemplos en TR sean acertados utilizando S



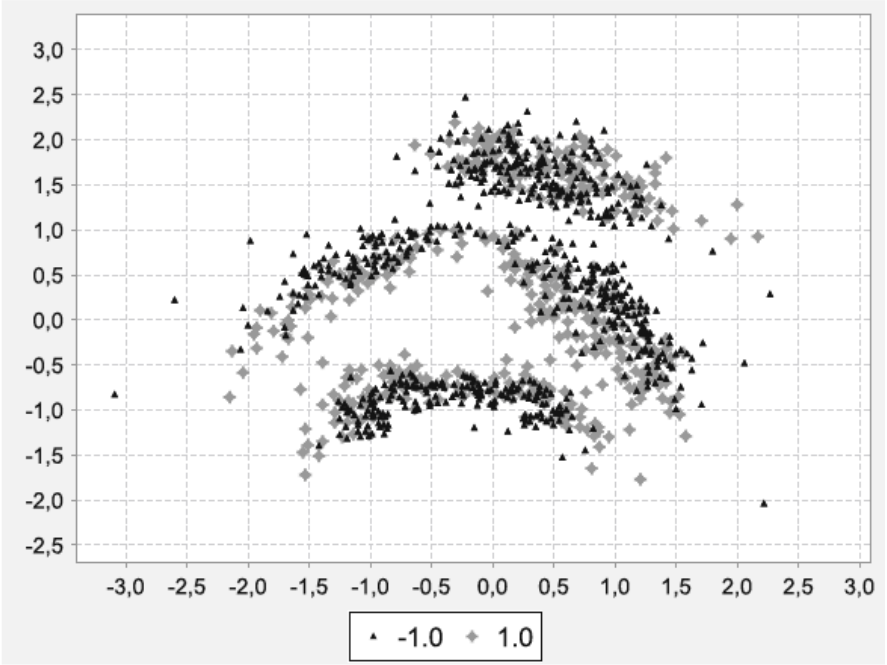


Descripción de los métodos

- **Condensación**
  - **Condensed Nearest Neighbor (CNN)**



Banana Original (0.8751,0.7476)



CNN (0.7729,0.8664,0.7304)

(Reduction rate, Test Accuracy, Test Kappa)

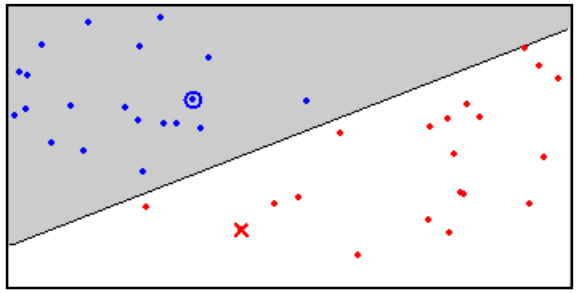
Descripción de los métodos

- **Condensación**
  - **Reduced Nearest Neighbor (RNN)**
    - Empieza con $S = S_{CNN}$ y elimina cada instancia de S si dicha eliminación no causa que ninguna otra instancia en TR sea fallada al clasificar con las instancias que quedan en S
    - Genera un subconjunto del resultado obtenido con CNN

1. Inicializa S como el subconjunto devuelto por CNN
2. Elimina un ejemplo de S si todos los ejemplos de TR siguen clasificándose correctamente sin él

Descripción de los métodos

- **Condensación**
  - **Condensed Nearest Neighbor (CNN)**
    - CNN depende mucho del orden en el que se toman las instancias
    - Tiende a conservar aquellas instancias con ruido (puesto que son mal clasificadas por las instancias en S)
  - **Reduced Nearest Neighbor (RNN)**
    - Trata de reducir el conjunto obtenido por CNN
    - Elimina instancias superfluas en el conjunto obtenido por CNN
  - No hay garantía de que CNN o RNN encuentren el conjunto consistente mínimo
    - Conjunto mínimo de ejemplos que clasifican correctamente al conjunto de entrenamiento



Conjunto Consistente Mínimo

Descripción de los métodos

- **Condensación**
  - **CNN y RNN**

CNN(Training set T): Object set S
$S = \varnothing$
Repeat
  Additions= FALSE
  For all patterns in T do
    Randomly pick O from T
    Find $s_c \in S$ such that $Distance(O,s_c) = \min_j Distance(O,s_j)$
    If class(O) $\neq$ class(s_c) then
      $S = S \cup \{O\}$
      Additions = TRUE
Until NOT(Additions)
Return S

Método *CNN*

RNN(Training set T, *CNN* set S_{CNN}) : object set S

1. $S=S_{CNN}$
2. Remove the first object from S
3. Use S to classify all objects in T:
   a) If all objects are classified correctly, go to 4
   b) If an object is classified incorrectly, return the object that was removed and go to 4
4. If every object in S has been removed once (and posibly replaced) then halt.
   Otherwise, remove the next object and go to 3.

Return S

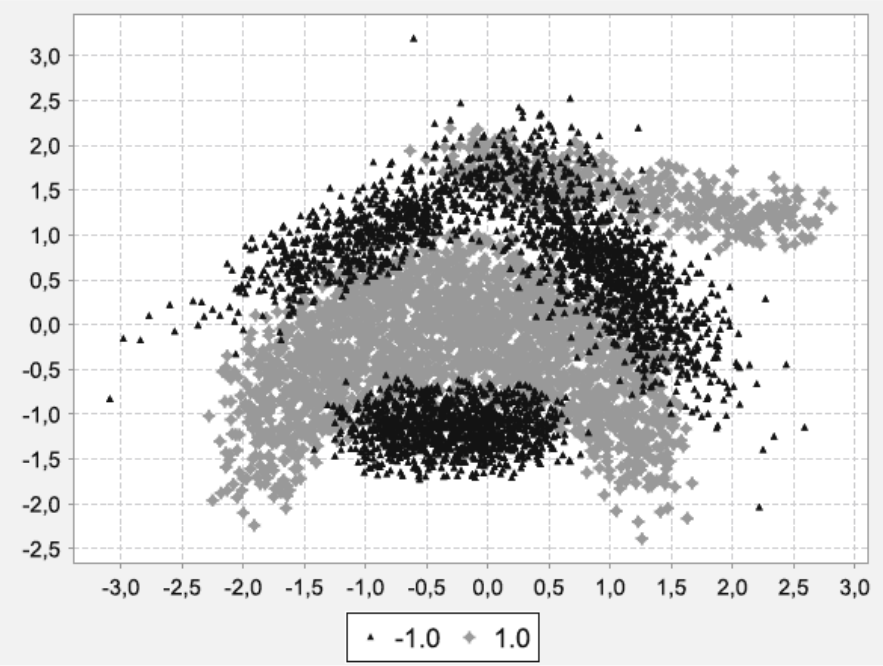
Método *RNN*

Descripción de los métodos

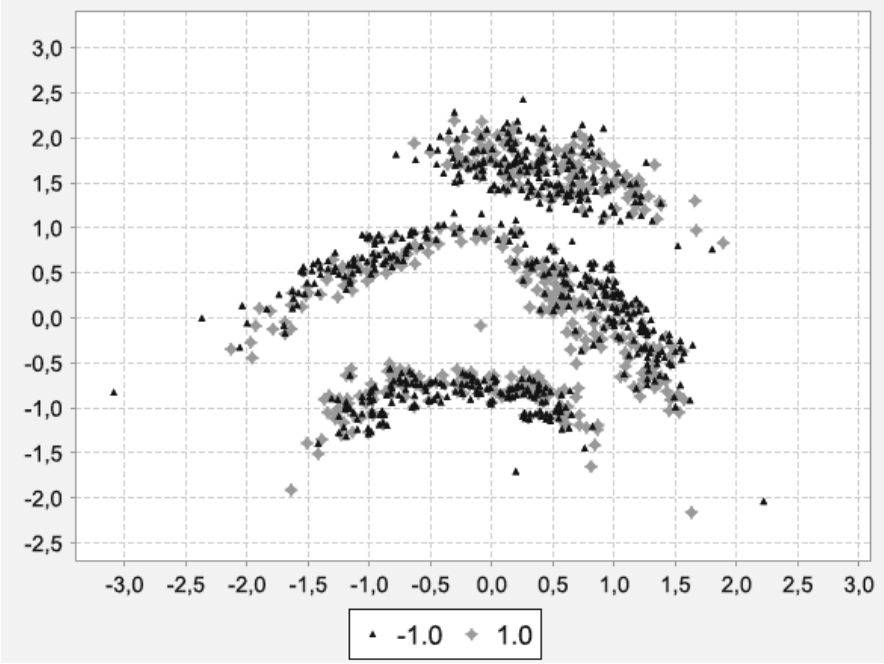
- **Condensación**
  - **Fast Condensed Nearest Neighbor family (FCNN)**
    - Familia de algoritmos basados en la idea de CNN
    - **FCNN1**
      - Comienza introduciendo en S los centroides de cada clase
      - Para cada prototipo $s \in S$ se busca su enemigo más cercano en su celda de Voronoi (ejemplos que son más cercanos a s que a ningún otro)
        - Se añade el enemigo a S (si existe)
      - Repetir el proceso hasta que no haya más enemigos en ninguna celda
    - **FCNN2** – similar a FCNN1
      - En vez de añadir el enemigo más cercano en la celda de Voronoi
        - Se añade el centroide de todos los enemigos en la celda
    - **FCNN3** – similar a FCNN1
      - En vez de añadir un prototipo por celda solo se añade un prototipo
        - Aquel que corresponde a la celda de Voronoi con más enemigos
        - S se inicializa solo con el centroide de la clase mayoritaria

Descripción de los métodos

- **Condensación**
  - **FCNN1**



Banana Original (0.8751,0.7476)

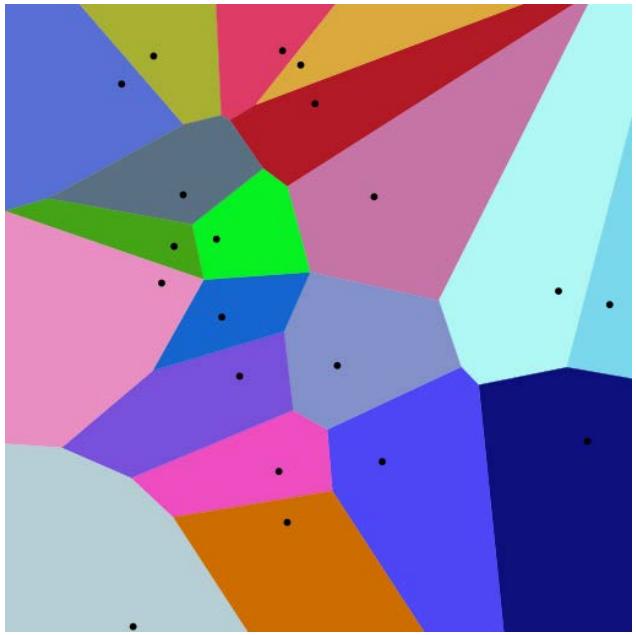


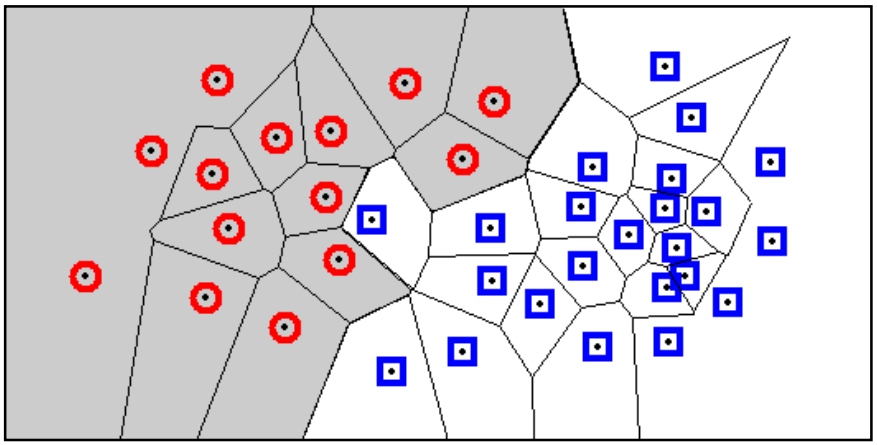
FCNN (0.8010,0.8655,0.7284)

(Reduction rate, Test Accuracy, Test Kappa)

Descripción de los métodos

- **Diagrama de Voronoi**
  - Crea regiones con todos los puntos más cercanos al punto correspondiente
    - Cada celda contiene un ejemplo y todas las localizaciones dentro de la celda están más cerca de ese punto que de cualquier otro



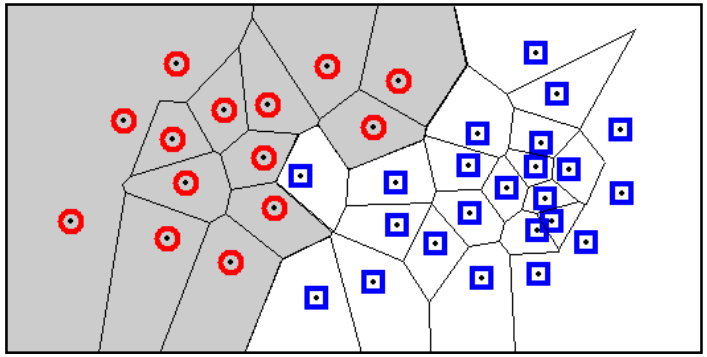


Descripción de los métodos

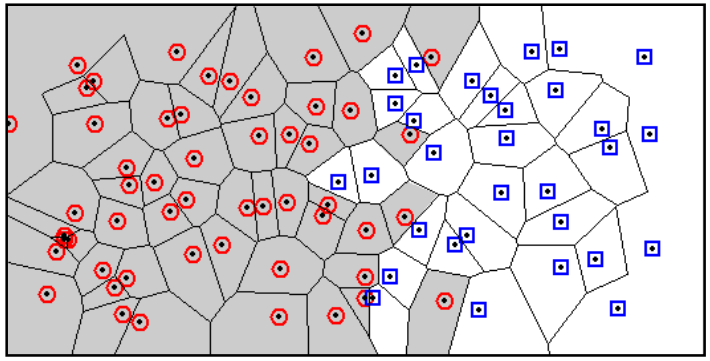
- **Edición**
  - **Edited Nearest Neighbor (ENN)** – Wilson method
    - Comienza con $S = TR$
    - Cada instancia en S se elimina si no está de acuerdo con la mayoría de sus k vecinos más cercanos

Descripción de los métodos

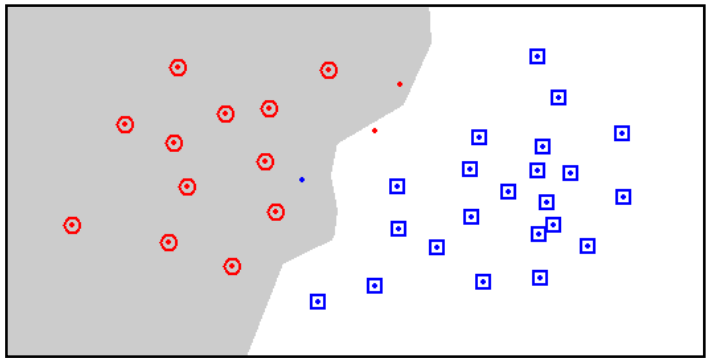
- **Edición**
  - **Edited Nearest Neighbor (ENN)** – Wilson method



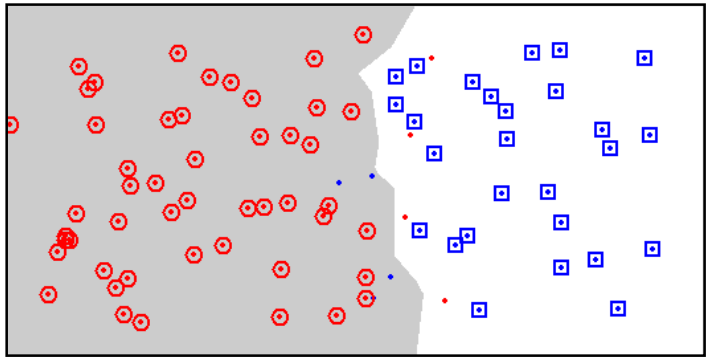
Original data



Original data



Wilson editing with k=3



Wilson editing with k=3

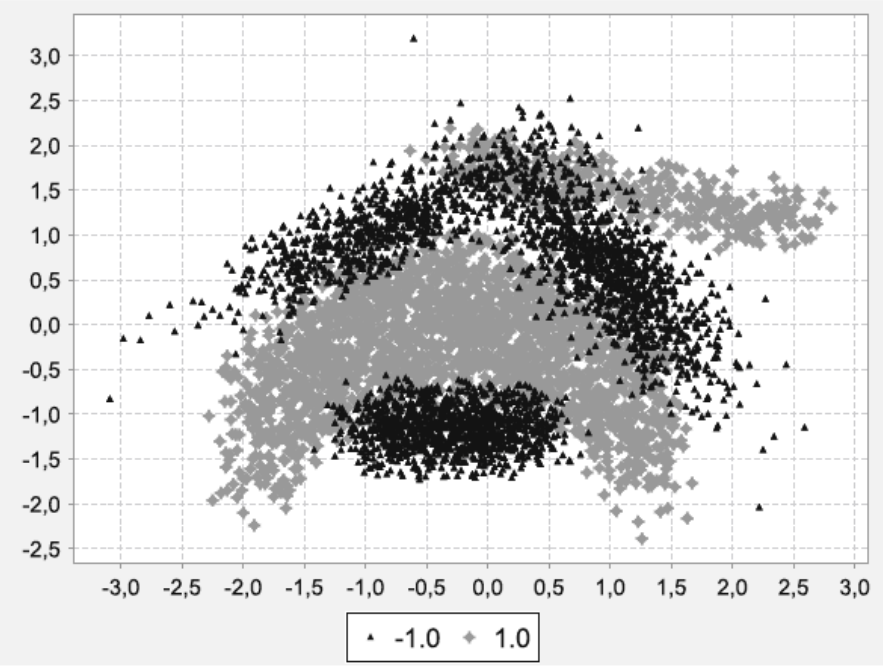
Descripción de los métodos

- **Edición**
  - **All KNN**
    - Es una extensión de ENN
    - Consiste en repetir ENN con valores de $k = 1 \ldots K$ y eliminar aquellas instancias falladas con cualquiera de dichos valores

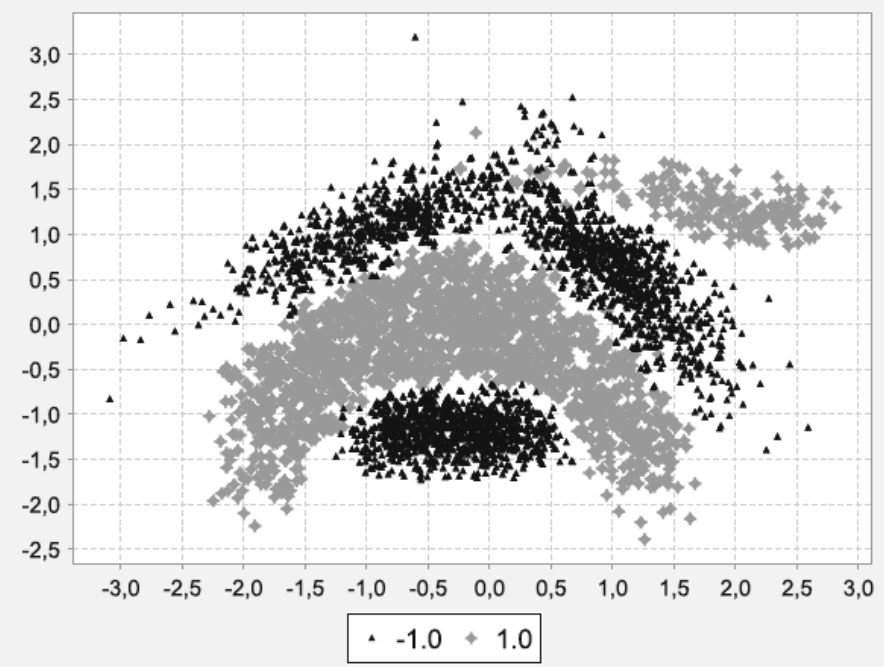
1. Repetir con $k = 1 \ldots K$
   - Marcar como instancia a eliminar las instancias incorrectamente clasificadas por sus k vecinos más cercanos
2. Eliminar instancias marcadas como instancia a eliminar

Descripción de los métodos

- **Edición**
  - **All KNN**



Banana Original (0.8751,0.7476)



AllKNN (0.1758,0.8934,0.7831)

(Reduction rate, Test Accuracy, Test Kappa)

Descripción de los métodos

- **Edición**
  - **ENN y All kNN**

ENN(Training set T): object set S

1. $S=T$
2. For each object O_i in S *do*
   a) Find the k nearest neighbors of O_i in S – $\{O_i\}$
   b) Remove O_i from S if its label disagrees with the class associated with the largest number of k nearest neighbors
3. Return S

Método *ENN*

All *k*-NN((Training set T, number of neighbors k): object set S

1. $S=T$
2. For each object O in S do
   a) i=1, flag(O)=1
   b) find i nearest neighbors of O : NN(i,O)
   c) If the majority of NN(i,O) classify O incorrectly, flag(O)=0
   d) $i = i + 1$
   e) If $i \leq k$ go to step b)
3. Eliminate from S those objects with flag(O)=0
4. Return S

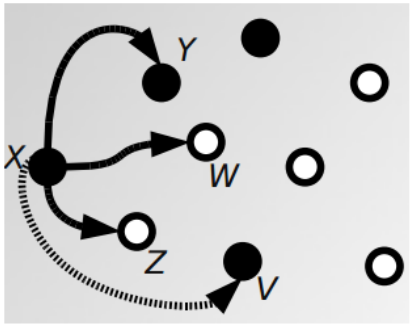
Método *All k-NN*

Descripción de los métodos

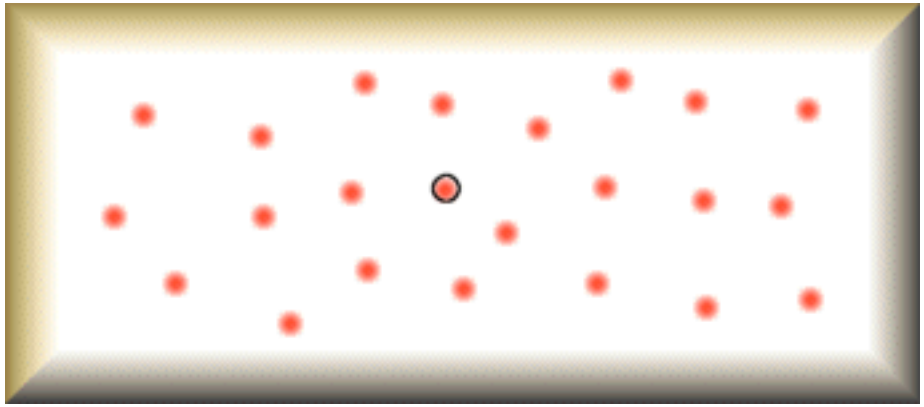
- **Híbridos**
  - **Decremental Reduction Optimization Procedure Family (DROP)**
    - Cada instancia X_i tiene k vecinos más cercanos (con k valor impar pequeño)
    - X_i tiene un enemigo más cercano
      - La instancia más cercana con clase diferente
    - Aquellas instancias que tienen a X_i como uno de sus k vecinos más cercanos se llaman asociadas de X_i

Descripción de los métodos

- **Híbridos**
  - **DROP1**
    - Comienza con $S = TR$
    - Elimina un ejemplo de S si sus asociados en S son clasificados correctamente más veces sin él que con él



Listas de Asociados				
Y	**W**	**Z**	**V**	**...**
X	X	X	X	...
...	...	...	...	...



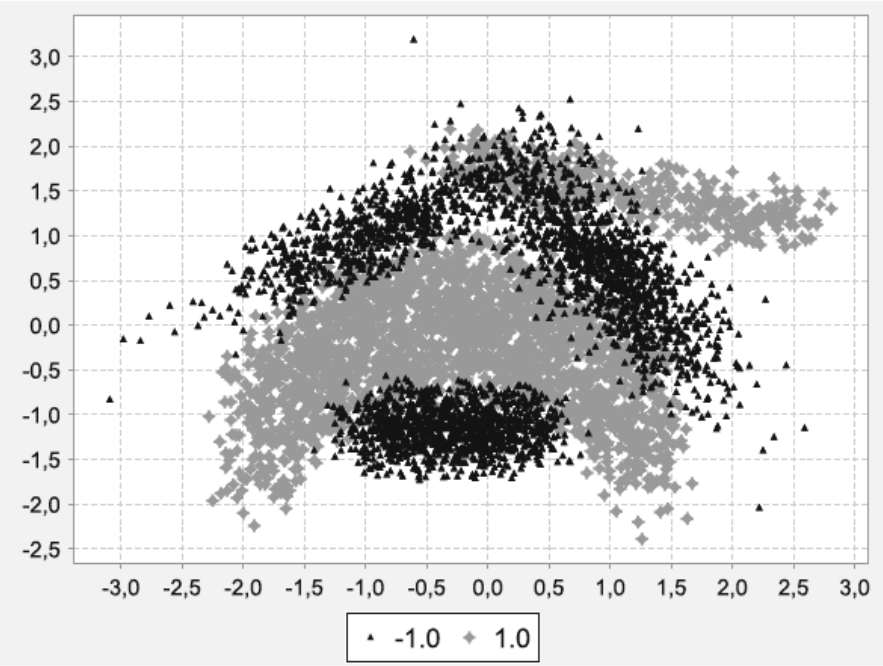
k=3
La instancia tiene 6 asociados

Descripción de los métodos

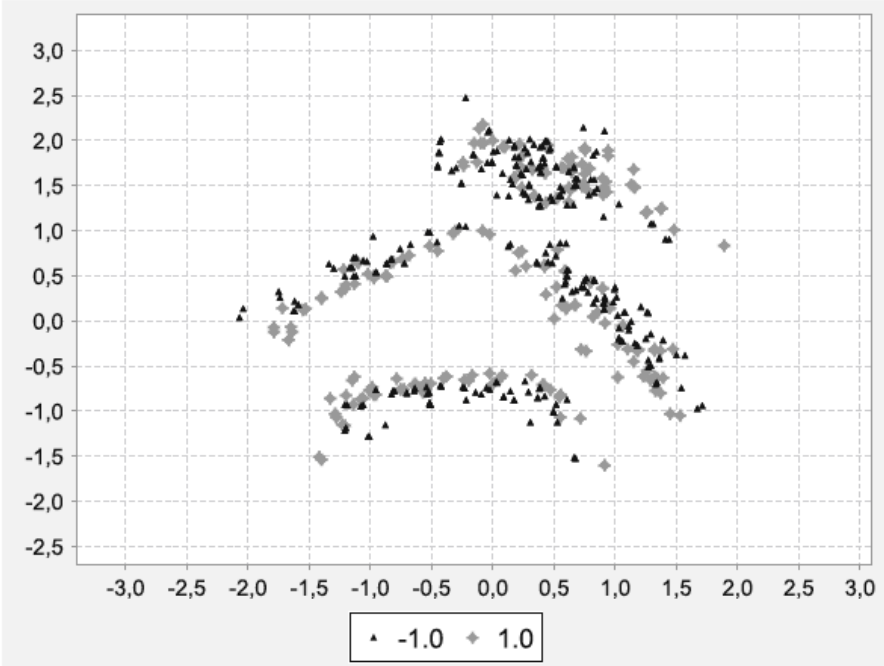
- **Híbridos**
  - **DROP2**
    - Comienza con $S = TR$
    - Elimina un ejemplo de S si sus asociados en TR son clasificados correctamente más veces sin él que con él
    - Cambia el orden de eliminación de las instancias de S
      - Reordena las instancias en S por su distancia a su enemigo más cercano (de mayor a menor)
        - Se eliminan primero instancias lejanas a la frontera de decisión
        - Hay más posibilidades de mantener los puntos en la frontera
  - **DROP3**
    - Combina DROP2 con ENN
      - Utiliza ENN primero
      - Continúa con DROP2 después

Descripción de los métodos

- **Híbridos**
  - **DROP3**



Banana Original (0.8751,0.7476)



DROP3 (0.9151,0.8696,0.7356)

(Reduction rate, Test Accuracy, Test Kappa)

Descripción de los métodos

- **Híbridos**
  - **DROP1**

1. DROP1(Training set T): Instance set S.
2. Let $S = T$.
3. For each instance P in S:
4. Find $P.N_{1..k+1}$, the $k + 1$ nearest neighbors of P in S.
5. Add P to each of its neighbors' lists of associates.
6. For each instance P in S:
7. Let *with* = # of associates of P classified correctly with P as a neighbor.
8. Let *without* = # of associates of P classified correctly without P.
9. If *without* $\geq$ *with*
10. Remove P from S.
11. For each associate A of P
12. Remove P from A's list of nearest neighbors.
13. Find a new nearest neighbor for A.
14. Add A to its new neighbor's list of associates.
15. For each neighbor N of P
16. Remove P from N's lists of associates.
17. Endif
18. Return S.

Descripción de los métodos

- **Híbridos**
  - **Iterative Case filtering (ICF)**
    - Define el conjunto local de un ejemplo X como $L(X)$
      - Contiene todos los ejemplo dentro de la híper-esfera más grande centrada en X_i tal que la híper-esfera solo contiene ejemplos de la misma clase que X_i
      - Dos propiedades
        - Cobertura (coverage)
          - Instancias en TR en las que X_i está en su híper-esfera
        - Alcance (reachability)
          - Instancias en TR que están en la híper-esfera de X_i

$$Coverage(X_i) = \{X_i' \in TR : X_i \in L(X_i')\},$$

$$Reachability(X_i) = \{X_i' \in TR : X_i' \in L(X_i)\},$$

Descripción de los métodos

- **Híbridos**
  - **Iterative Case filtering (ICF)**

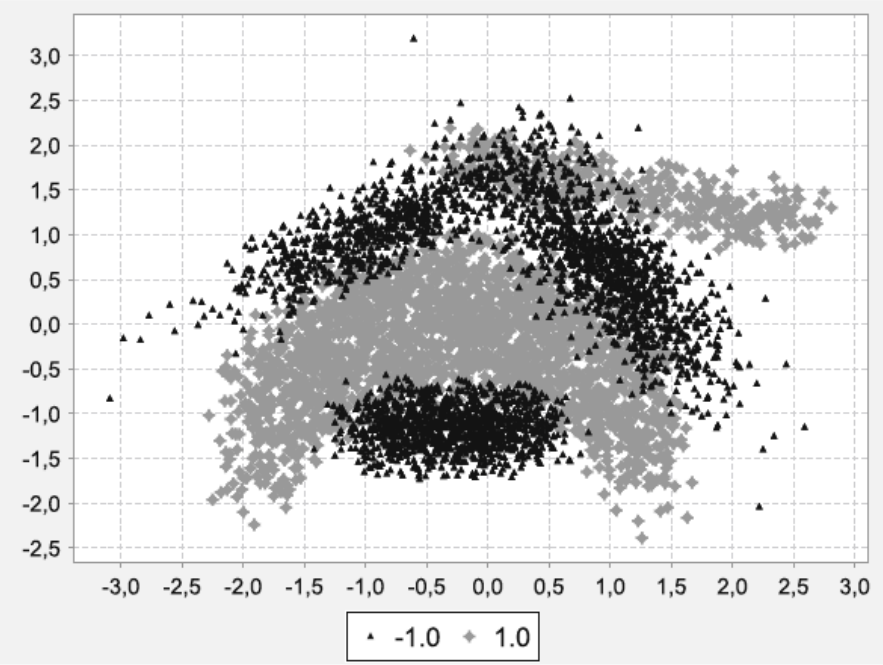
1. Se aplica ENN para eliminar ruido ($TR = S_{ENN}$)
2. Eliminar cada instancia X_i en TR si su alcance es mayor que su cobertura
3. Recalcular alcance y cobertura para las instancias restantes y repetir hasta que no haya cambios

ICF (Training set T)
// Perform Wilson Editing
For all $O \in T$ do
  If O classified incorrectly by k nearest neighbors then
    Flag O for removal
For all $O \in T$ do
  If O flagged for removal then $T = T - \{O\}$
Repeat
  For all $O \in T$ do
    Compute *recheable(O)*
    Compute *coverage(O)*
  Progress = false
  For all $O \in T$ do
    If $|recheable(O)| > |coverage(O)|$ then
      Flag O for removal
      Progress = True
  For all $O \in T$ do
    If O flagged for removal then $T = T - \{O\}$
 Until not *Progress*
Return T

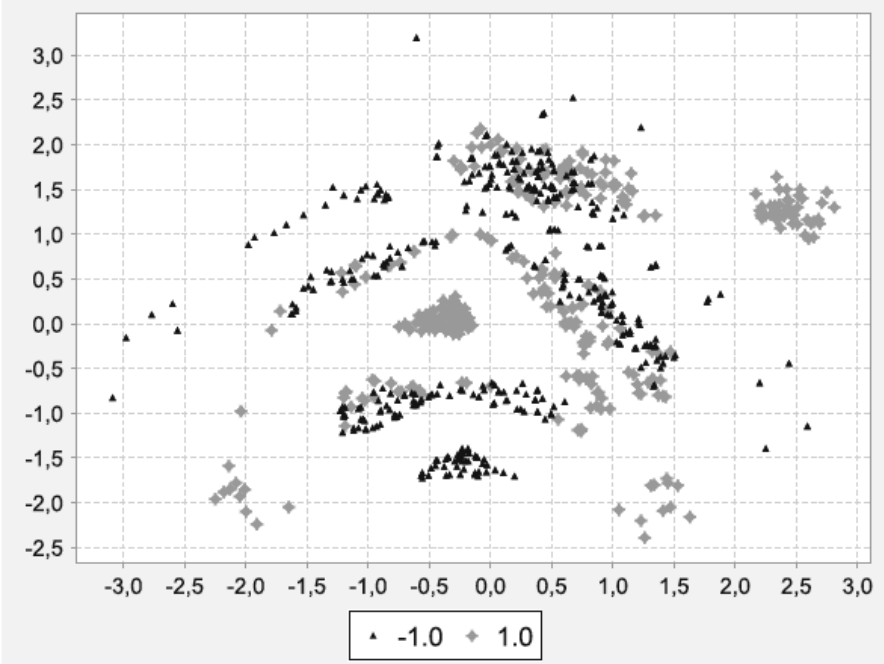
Método *ICF*

Descripción de los métodos

- **Híbridos**
  - **Iterative Case filtering (ICF)**



Banana Original (0.8751,0.7476)



ICF (0.8635,0.8081,0.6088)

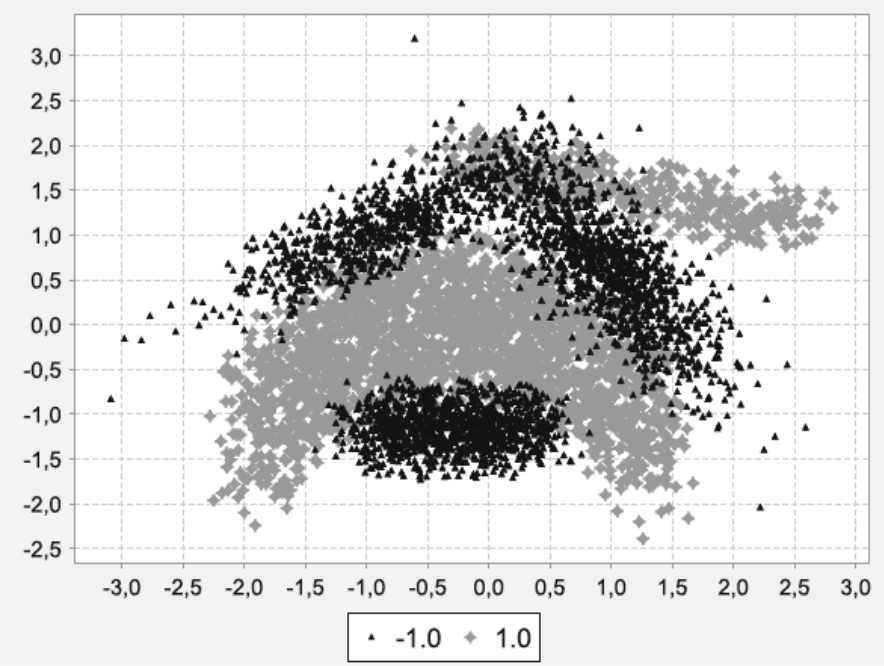
(Reduction rate, Test Accuracy, Test Kappa)

Descripción de los métodos

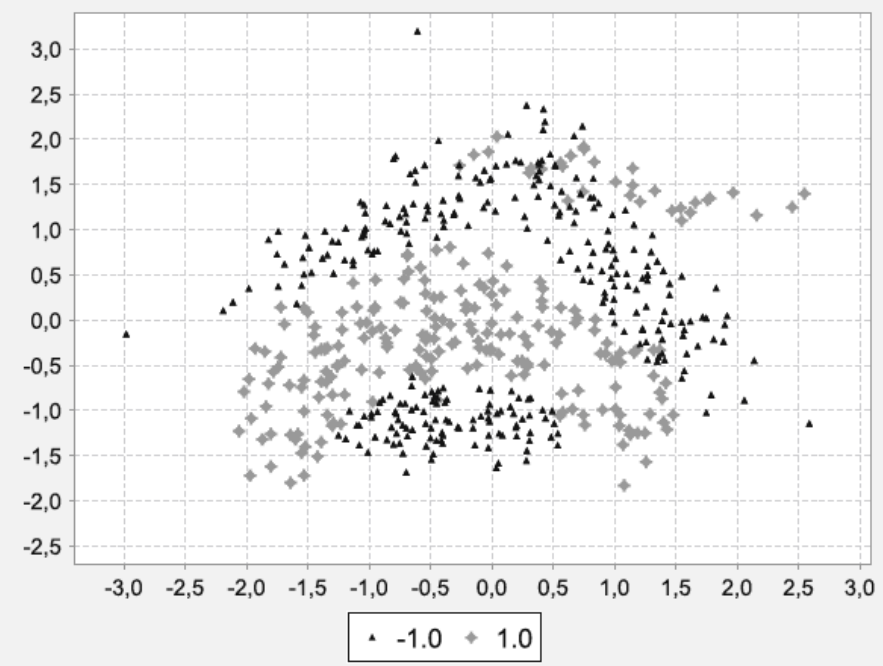
- **Híbridos**
  - **Random Mutation Hill Climbing (RMHC)**
    - Comienza con S siendo un subconjunto aleatorio de TR
      - Con un número fijo de instancias s $(s = \%|T\ R|)$
    - En cada iteración
      - Se intercambia una instancia de S con otra de $TR - S$
      - El cambio se mantiene si la precisión mejora
        - Leave-one-out para obtenerla

Descripción de los métodos

- **Híbridos**
  - **Random Mutation Hill Climbing (RMHC)**



Banana Original (0.8751,0.7476)



RMHC (0.9000,0.8972,0.7915)

(Reduction rate, Test Accuracy, Test Kappa)

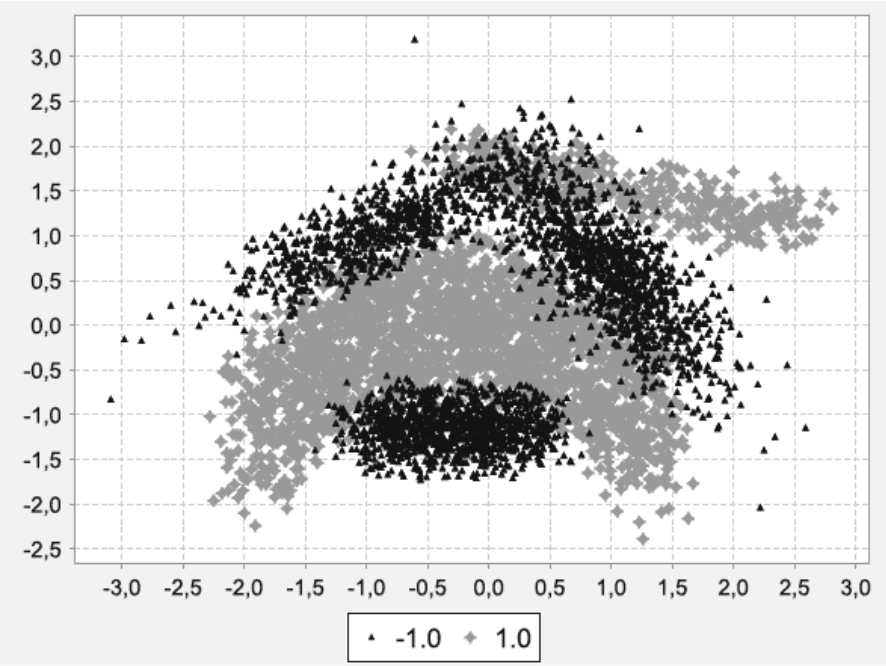
Descripción de los métodos

- **Híbridos**
  - **Steady-state memetic algorithm (SSMA) y CHC genetic algorithm**
    - Utilizan algoritmos evolutivos para obtener el conjunto óptimo de instancias en base a la precisión y el tamaño del conjunto
      - $Eval = acc(S) \cdot \alpha - |S| \cdot (1 - \alpha)$
    - Suelen ser precisos y reducir mucho

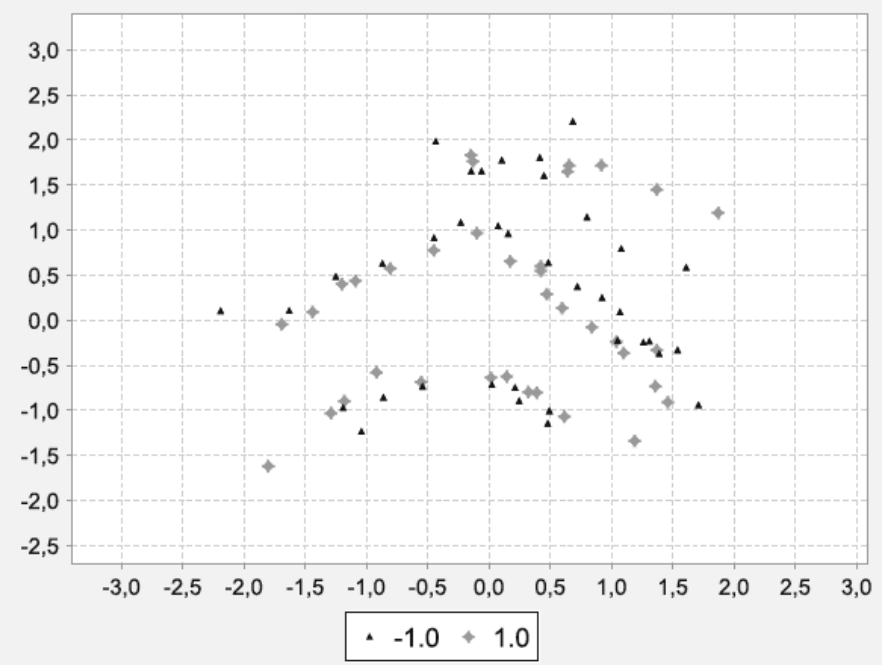
```
Initialize population
While (not termination-condition) do
   Use binary tournament to select two parents
   Apply crossover operator to create
   offspring (Off_1, Off_2)
   Apply mutation to Off_1 and Off_2
      Evaluate Off_1 and Off_2
      For each Off_i
         Invoke Adaptive-P_LS-mechanism to
         obtain P_LS_i for Off_i
         If υ(0,1) < P_LS_i then
            Perform meme optimization for Off_i
         End if
      End for
   Employ standard replacement for Off_1 and Off_2
End while
Return the best chromosome
```

Descripción de los métodos

- **Híbridos**
  - **Steady-state memetic algorithm (SSMA) y CHC genetic algorithm**



Banana Original (0.8751,0.7476)



SSMA (0.9879,0.8964,0.7900)

(Reduction rate, Test Accuracy, Test Kappa)

Otro ejemplo de aplicación

- **Útiles para simplificar los modelos**

	Execution Time(sec)	Rules Number	% Reduction	C4.5	
				%Ac Trn	%Ac Test
C4.5	265	252		99.97%	99.94%
Cnn Strat	8	83	81.61%	98.48%	96.43%
Drop1 Strat	111	3	99.97%	38.63%	34.97%
Drop2 Strat	105	82	76.66%	81.40%	76.58%
Drop3 Strat	131	49	56.74%	77.02%	75.38%
Ib2 Strat	7	48	82.01%	95.81%	95.05%
Ib3 Strat	3	74	78.92%	99.13%	96.77%
Icf Strat	242	68	23.62%	99.98%	99.53%
CHC Strat	1960	9	99.68%	98.97%	97.53%

Dataset ADULT

	No. Instan-cias - N	No. Varia-bles	No. Reglas		No. Variables/ regla		Confidencia de las Reglas N(Cond,Clas)/N	
Adult	30132	14	C4.5	IS-CHC/ C4.5	C4.5	IS-CHC/ C4.5	C4.5	IS-CHC/ C4.5
			359	5	14	3	0.003	0.167

Índice

Filtrado del ruido a nivel de datos

- Vamos a considerar **tres filtros de ruido** para detectar instancias mal etiquetadas
  - Los más **comunes y relevantes**
  - Son métodos basados en **ensembles y técnicas de voto**
  - **Idea**
    - *Recoger información de modelos diferentes permite obtener un método con mayor capacidad de detección de ejemplos mal etiquetados que usando un único modelo*

      - Ensemble Filter (EF)
      - Cross-Validated Committees Filter
      - Iterative-Partitioning Filter

Ensemble Filter

- ***Ensemble Filter* (EF)**
  - **Muy conocido** en la literatura
  - Utiliza un **conjunto de clasificadores aprendidos en diferentes subconjuntos** de los datos de entrenamiento
    - **Se utilizan como filtros** para el conjunto de entrenamiento
    - **Originalmente** se propuso el uso de
      - **Árbol de decisión C4.5**
      - **1-NN**
      - **LDA (Linear Discriminant Analysis)**

C.E. Brodley, M.A. Friedl. **Identifying Mislabeled Training Data**. *Journal of Artificial Intelligence Research 11 (1999) 131-167*.

Ensemble Filter

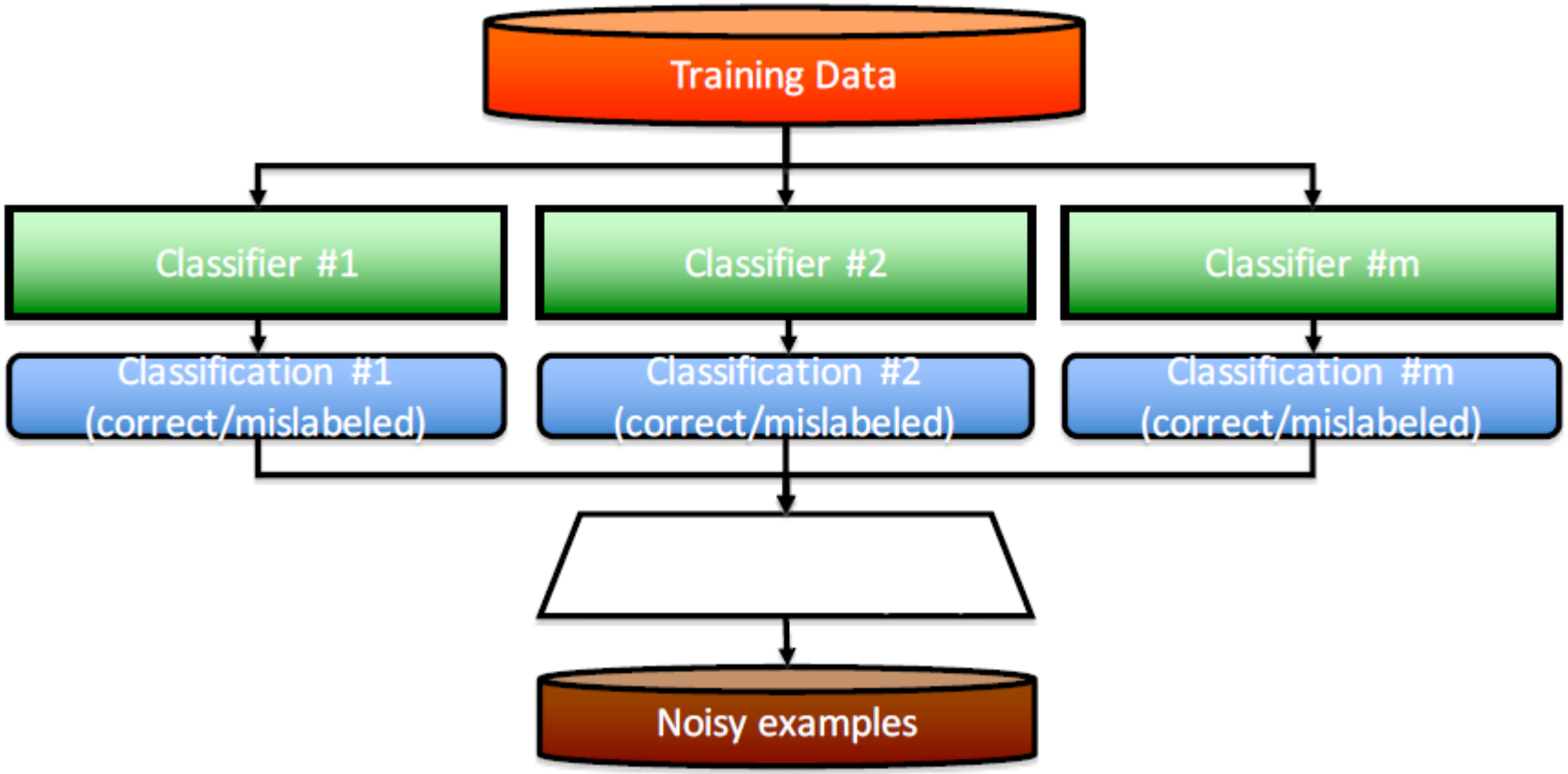
- ***Ensemble Filter* (EF)**
  - **Dos pasos**
    1. **Para cada algoritmo** de aprendizaje (C4.5, 1-NN, LDA), se utiliza una **validación cruzada de k particiones** para etiquetar cada ejemplo de entrenamiento como
       - **correcto** (predicción = etiqueta en training)
       - **mal etiquetado** (predicción ≠ etiqueta en training)
       - Es decir, entrenamos un clasificador con k-1 particiones y lo utilizamos para etiquetar las instancias en la partición restante
       - Lo repetimos k veces

Ensemble Filter

- ***Ensemble Filter* (EF)**
  - **Dos pasos**
    2. Aplicar un **esquema de voto** para identificar el conjunto final de instancias que son ruido
       - **Consenso**: elimina un ejemplo si **todos** los clasificadores lo han fallado
       - **Mayoría**: elimina un ejemplo si ha sido fallado por más de la mitad de los clasificadores

Ensemble Filter

- ***Ensemble Filter* (EF)**



Cross-Validated Committees Filter

- ***Cross-Validated Committees Filter* (CVCF)**
  - **Similar a EF**
  - Pero con **dos diferencias principales**
    1. **Utiliza el mismo algoritmo de aprendizaje (C4.5)** para crear los clasificadores en cada subconjunto de datos
       - Los autores de CVCF ponen un énfasis especial en usar **ensembles con árboles de decisión** como C4.5 porque funcionan bien para filtrar el ruido
    2. **Cada clasificador** construido con **la validación cruzada de k particiones** es usado **para etiquetar TODOS los ejemplos de entrenamiento** (no solo los de test) como
       - **correctos** (predicción = etiqueta en traning)
       - **mal etiquetados** (predicción ≠ etiqueta en traning)

S. Verbaeten, A.V. Assche. **Ensemble methods for noise elimination in classification problems**. *4th International Workshop on Multiple Classifier Systems (MCS 2003). LNCS 2709, Springer 2003, Guilford (UK, 2003) 317-325.*

Cross-Validated Committees Filter

- ***Cross-Validated Committees Filter* (CVCF)**
  - **Dos pasos**
    1. Utilizar una **validación cruzada de k particiones** para entrenar **k árboles de decision C4.5**. Etiquetar cada ejemplo de entrenamiento **con los k árboles** como
       - **correcto** (predicción = etiqueta en traning)
       - **mal etiquetado** (predicción ≠ etiqueta en traning)

       - Es decir, entrenamos un árbol C4.5 con k-1 particiones y lo utilizamos para etiquetar TODAS las instancias en las k particiones
       - Lo repetimos k veces

Cross-Validated Committees Filter

- ***Cross-Validated Committees Filter* (CVCF)**
  - **Dos pasos**
    2. Aplicar un **esquema de voto** para identificar el conjunto final de instancias que son ruido
       - **Consenso**: elimina un ejemplo si **todos** los clasificadores lo han fallado
       - **Mayoría**: elimina un ejemplo si ha sido fallado por más de la mitad de los clasificadores

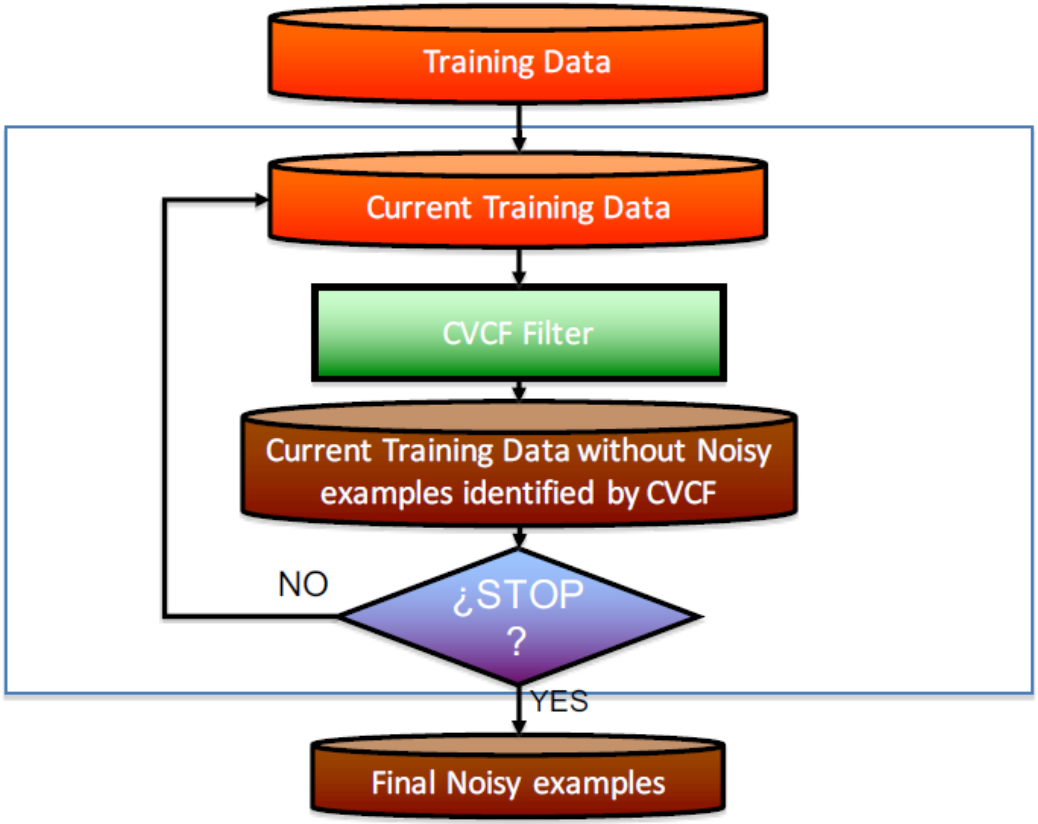
Iterative-Partitioning Filter

- ***Iterative-Partitioning Filter* (IPF)**
  - **Basado en CVCF**
  - Elimina instancias con ruido en **múltiples iteraciones**
    - Hasta que la cantidad de instancias con ruido detectadas en una iteración sea menor que un porcentaje *p* del tamaño del conjunto de entrenamiento original

T.M. Khoshgoftaar, P. Rebours. **Improving software quality prediction by noise filtering techniques**. *Journal of Computer Science and Technology 22 (2007) 387-396.*

Iterative-Partitioning Filter

- ***Iterative-Partitioning Filter* (IPF)**



Iterative-Partitioning Filter

- ***Iterative-Partitioning Filter* (IPF)**
  - **Repetir** hasta que el número de instancias eliminadas en una iteración sea menor que el umbral establecido
    1. Utilizar una **validación cruzada de k particiones** para entrenar **k árboles de decisión C4.5**. Etiquetar cada ejemplo de entrenamiento **con los k árboles** como
       - **correcto** (predicción = etiqueta en traning)
       - **mal etiquetado** (predicción ≠ etiqueta en traning)

       - Es decir, entrenamos un árbol C4.5 con k-1 particiones y lo utilizamos para etiquetar TODAS las instancias en las k particiones
       - Lo repetimos k veces

Iterative-Partitioning Filter

- ***Iterative-Partitioning Filter* (IPF)**
  - **Repetir** hasta que el número de instancias eliminadas en una iteración sea menor que el umbral establecido
    2. Aplicar un **esquema de voto** para identificar el conjunto final de instancias que son ruido
       - **Consenso**: elimina un ejemplo si **todos** los clasificadores lo han fallado
       - **Mayoría**: elimina un ejemplo si ha sido fallado por más de la mitad de los clasificadores

Filtrado de ruido

- **Resultados sobre tres clasificadores clásicos**

		Pairwise class noise					Uniform random class noise				
		0%	5%	10%	15%	20%	0%	5%	10%	15%	20%
	None	90.02	88.51	86.97	86.14	84.86	90.02	87.82	86.43	85.18	83.20
SVM	EF	90.49	89.96	89.07	88.33	87.40	90.49	89.66	88.78	87.78	86.77
	CVCF	90.56	89.86	88.94	88.28	87.76	90.48	89.56	88.72	87.92	86.54
	IPF	90.70	90.13	89.37	88.85	88.27	90.58	89.79	88.97	88.48	87.37
	None	82.46	81.15	80.35	79.39	78.49	82.46	79.81	78.55	76.98	75.68
Ripper	EF	83.36	82.87	82.72	82.43	81.53	83.46	83.03	82.87	82.30	81.66
	CVCF	83.17	82.93	82.64	82.03	81.68	83.17	82.59	82.19	81.69	80.45
	IPF	83.74	83.59	83.33	82.72	82.44	83.74	83.61	82.94	82.94	82.48
	None	83.93	83.66	82.81	82.25	81.41	83.93	82.97	82.38	81.69	80.28
C4.5	EF	84.18	84.07	83.70	83.20	82.36	84.16	83.96	83.53	83.38	82.66
	CVCF	84.15	83.92	83.24	82.54	82.13	84.15	83.61	83.00	82.84	81.61
	IPF	84.44	84.33	83.92	83.38	82.53	84.44	83.89	83.84	83.50	82.72