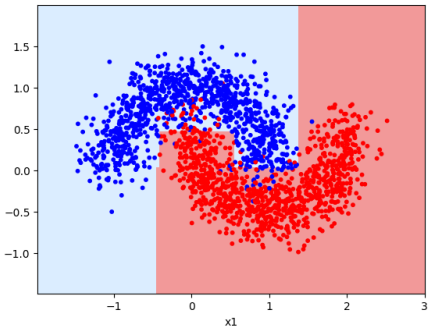
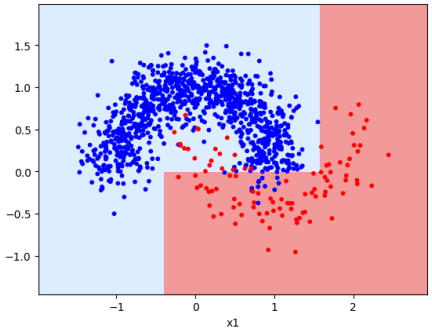
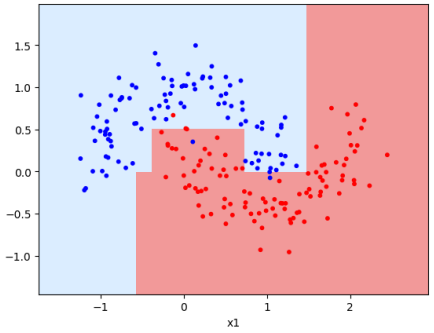
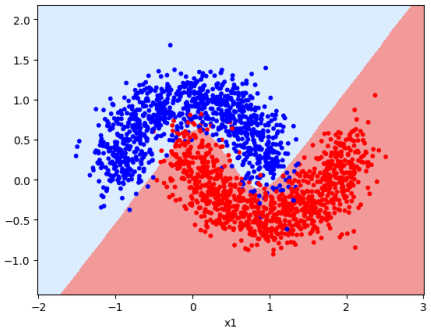
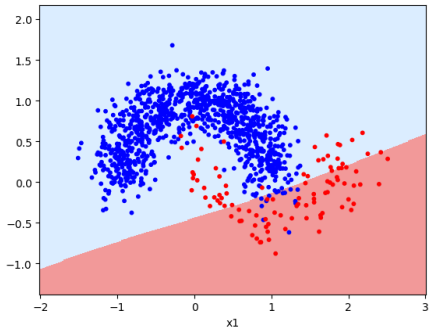
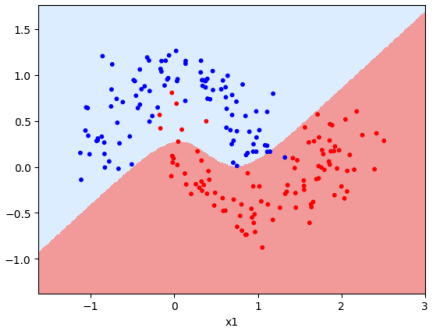
Tratamiento de Datos Desbalanceados

¿Por qué?

Superficies de decisión inducidas por diferentes algoritmos

	Referencia	Desbalance 100:1	Balanceados
Árbol de Decisión			
Perceptrón Multicapa			

¿Por qué?

- Clases pobremente representadas.
- Sesgo favorable a las clases mayoritarias.
  - Particularmente en clasificadores que minimizan el error sobre todo el conjunto de datos.
- Deterioro del desempeño de los algoritmos de minería de datos.
- Presente en muchas aplicaciones reales.

Tipos

- Between-class imbalance
- Within-class imbalance
  - Casos raros

Estrategias

- Definición del Problema
- Datos
- Algoritmos

Definición del problema

- **Redefinir el problema**
  - Enfocarse en un subdominio del problema o una partición de los datos donde el desbalance sea disminuido
- **Emplear métricas de evaluación adecuadas**
  - Algunas métricas tienden a inducir sesgos favorables a la clase mayoritaria, por ejemplo, si se evalúa el modelo de acuerdo con el porciento de instancias correctas.
  - Seleccionar métricas más adecuadas para datos desbalanceados

A nivel de los datos

- **Aprendizaje activo:** Los algoritmos de este tipo pueden determinar ejemplos sobre los que existe mayor incertidumbre en la clasificación y requerir información adicional. Puede esperarse que las instancias que representan clases minoritarias o casos raros sean los que provoquen mayor incertidumbre.

Estrategias

A nivel de los datos

- **Muestreo**
  - **Submuestreo (undersampling):** Extraer una muestra de la clase mayoritaria de tamaño proporcional a la cantidad de instancias de clase minoritaria. Algoritmo *One-side selection* para remover instancias de la clase mayoritaria que se encuentren cerca de la frontera con la clase minoritaria.
  - **Sobremuestreo (oversampling):** Añadir mediante algún tipo de muestreo con remplazo instancias de la clase minoritaria. Algoritmo SMOTE.

Estrategias

A nivel de los algoritmos

- Algoritmos que eviten el particionamiento voraz o recursivo de los datos.
- Algoritmos que empleen métricas diseñadas para trabajar con datos desbalanceados.
- Sesgo inductivo adecuado para datos desbalanceados.
- Algoritmos que implícita o explícitamente favorezcan los casos y clases raras.
- Aprender solamente las clases raras.

A nivel de los algoritmos

- **Algoritmos que eviten el particionamiento voraz o recursivo de los datos:**
  - Algoritmos como el ID3, realizan un particionamiento de los datos de forma voraz (siempre utilizando el mejor atributo local).
  - En la práctica, este enfoque tiene dificultades en casos de datos desbalanceados.

A nivel de los algoritmos

- **Algoritmos que empleen métricas diseñadas para trabajar con datos desbalanceados:**
  - Algunos algoritmos emplean métricas para guiar el aprendizaje, por ejemplo, el Perceptrón Multicapa en su definición más simple intenta minimizar el **error en la clasificación** métrica que sesga el aprendizaje en favor de la clase mayoritaria.
  - Pueden emplearse otras métricas basadas en la precisión, la cobertura o la F-Medida.

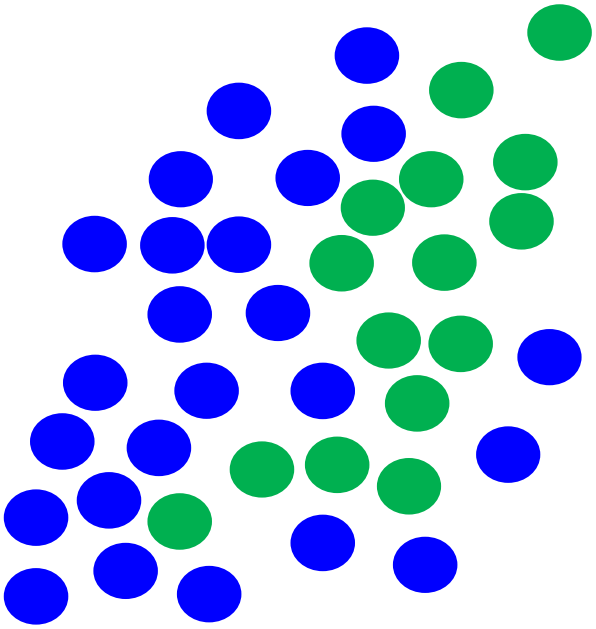
A nivel de los algoritmos

- **Sesgo inductivo adecuado para datos desbalanceados:**
  - Muchos algoritmos de aprendizaje favorecen la generalización frente a la especialización lo que es adecuado para aprender casos comunes, pero no casos raros.
  - Un enfoque contrario es intentar mantener aquellos casos específicos que representan clases minoritarias o casos raros.

A nivel de los algoritmos

- **Algoritmos que implícita o explícitamente favorezcan los casos y clases raras**
  - Por ejemplo, los algoritmos que implementan el llamado *cost-sensitive learning*.
  - Se puede ponderar los diferentes tipos de errores, dando mayor importancia a las clasificaciones incorrectas de la clase o casos raros.
- **Aprender solamente las clases raras**
  - El algoritmo puede aprender solamente las clases minoritarias o aprender primero estas y luego modelar el resto de los datos. Por ejemplo, el algoritmo Ripper.

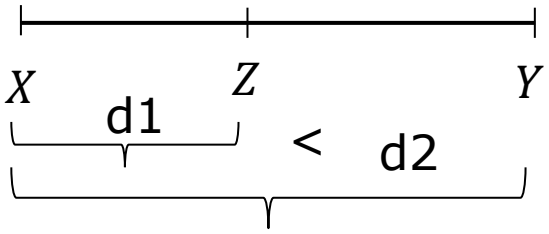
Enlace de Tomek



$X \in$ ●

$Y \in$ ●

- El par (X, Y) participa en un enlace de Tomek si no existe Z tal que:



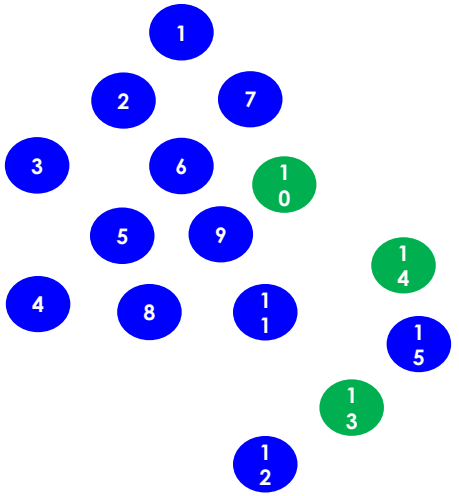
En otras palabras, los enlaces de Tomek identifican pares de instancias de clases diferentes que son vecinos más cercanos entre sí

Identifican instancias que son potencialmente ruido o instancias en la frontera

Submuestreo

One-side selection (Kubat)

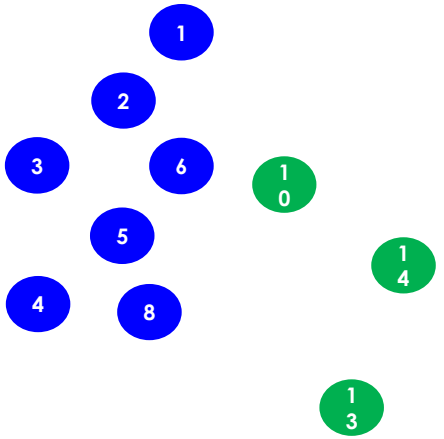
- Mantener las instancias de la clase minoritaria
- Seleccionar un subconjunto representativo de la clase mayoritaria
- Tipos de instancia
  - Ruido
  - Instancias en la frontera
  - Instancias redundantes
  - Instancias correctas



Submuestreo

One-side selection (Kubat)

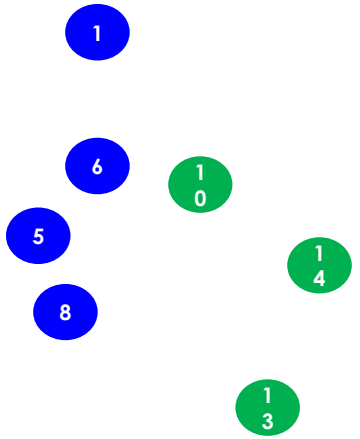
- Eliminar ruido e instancias, de la clase mayoritaria, en la frontera



Submuestreo

One-side selection (Kubat)

- Eliminar instancias redundantes de la clase mayoritaria



Submuestreo

One-side selection (Kubat)

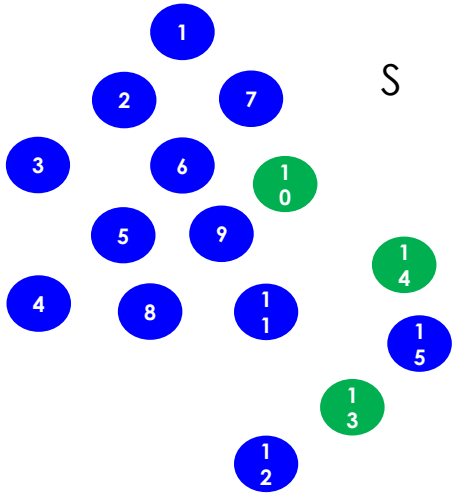
1- Sea S el conjunto original de entrenamiento.

2-Inicializar C con todas las instancias de las clases minoritarias y un ejemplo elegido aleatoriamente por cada clase mayoritaria.

3- Clasificar las instancias en S utilizando la regla 1-NN utilizando los ejemplos en C. Adicionar a C las instancias mal clasificadas.

4- Eliminar de C los ejemplos de la clase mayoritaria que participen en enlaces de Tomek.

Submuestreo



One-side selection (Kubat)

1- Sea S el conjunto original de entrenamiento.

2-Inicializar C con todas las instancias de las clases minoritarias y un ejemplo elegido aleatoriamente por cada clase mayoritaria.

3- Clasificar las instancias en S utilizando la regla 1-NN utilizando los ejemplos en C. Adicionar a C las instancias mal clasificadas.

4- Eliminar de C los ejemplos de la clase mayoritaria que participen en enlaces de Tomek.

Submuestreo

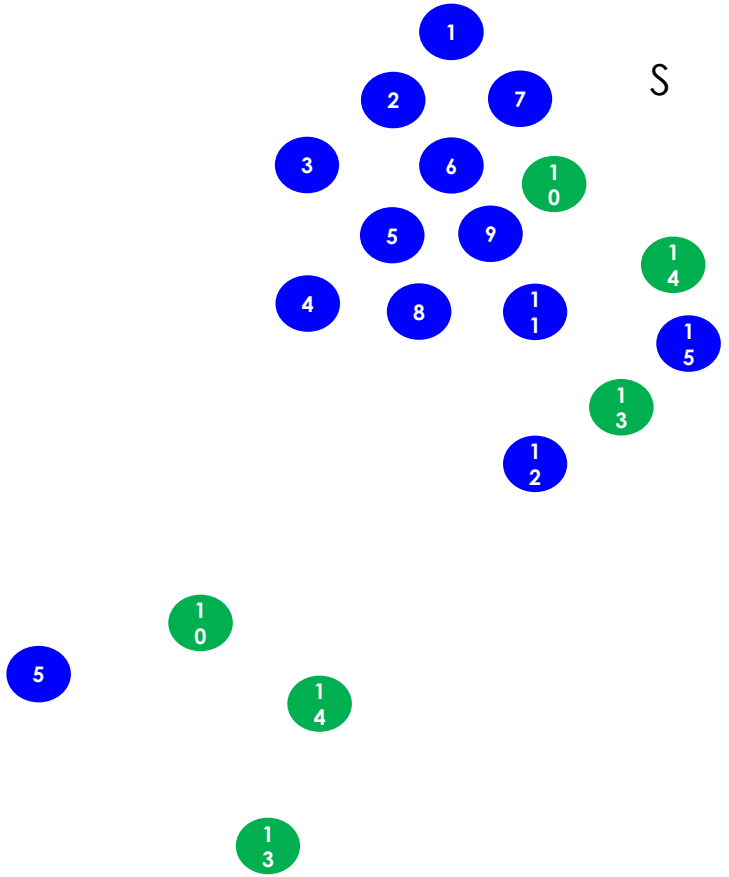
One-side selection (Kubat)

1- Sea S el conjunto original de entrenamiento.

2-Inicializar C con todas las instancias de las clases minoritarias y un ejemplo elegido aleatoriamente por cada clase mayoritaria.

3- Clasificar las instancias en S utilizando la regla 1-NN utilizando los ejemplos en C. Adicionar a C las instancias mal clasificadas.

4- Eliminar de C los ejemplos de la clase mayoritaria que participen en enlaces de Tomek.



Submuestreo

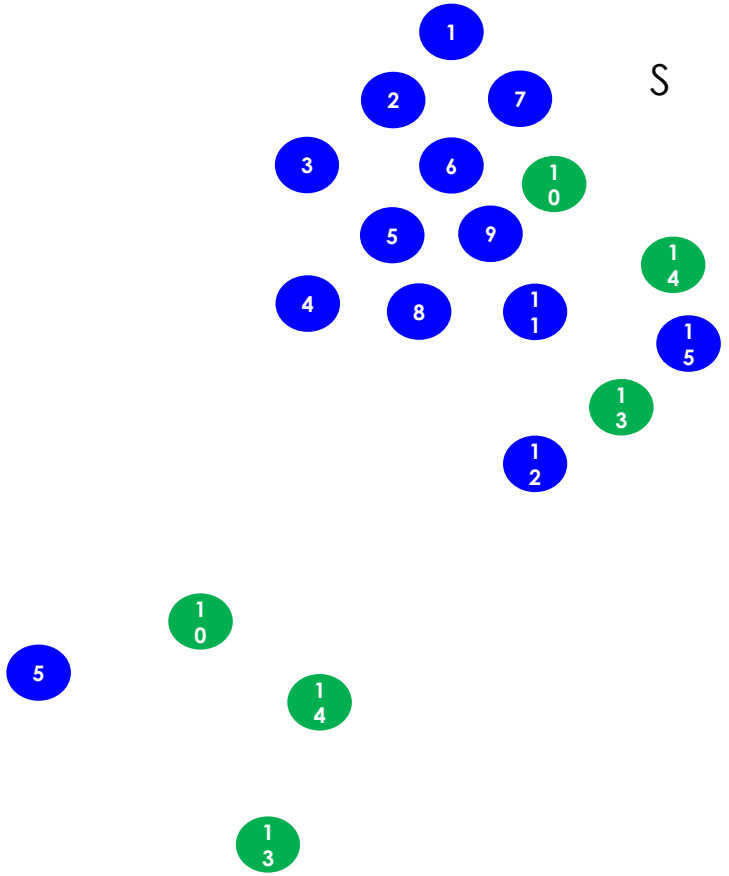
One-side selection (Kubat)

1- Sea S el conjunto original de entrenamiento.

2-Inicializar C con todas las instancias de las clases minoritarias y un ejemplo elegido aleatoriamente por cada clase mayoritaria.

3- Clasificar las instancias en S utilizando la regla 1-NN utilizando los ejemplos en C. Adicionar a C las instancias mal clasificadas.

4- Eliminar de C los ejemplos de la clase mayoritaria que participen en enlaces de Tomek.



Submuestreo

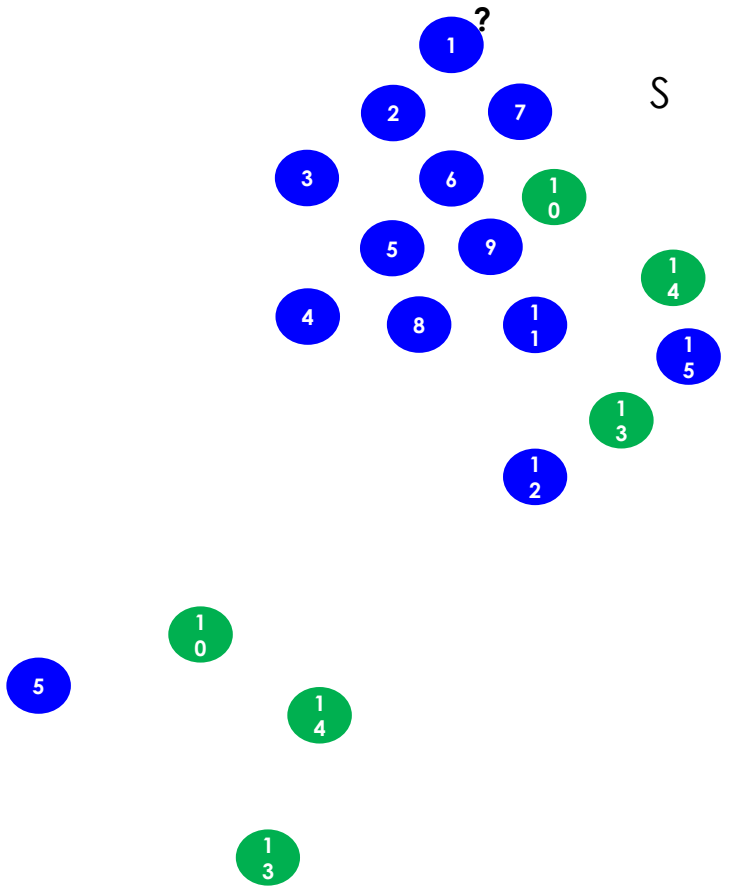
One-side selection (Kubat)

1- Sea S el conjunto original de entrenamiento.

2-Inicializar C con todas las instancias de las clases minoritarias y un ejemplo elegido aleatoriamente por cada clase mayoritaria.

3- Clasificar las instancias en S utilizando la regla 1-NN utilizando los ejemplos en C. Adicionar a C las instancias mal clasificadas.

4- Eliminar de C los ejemplos de la clase mayoritaria que participen en enlaces de Tomek.



Submuestreo

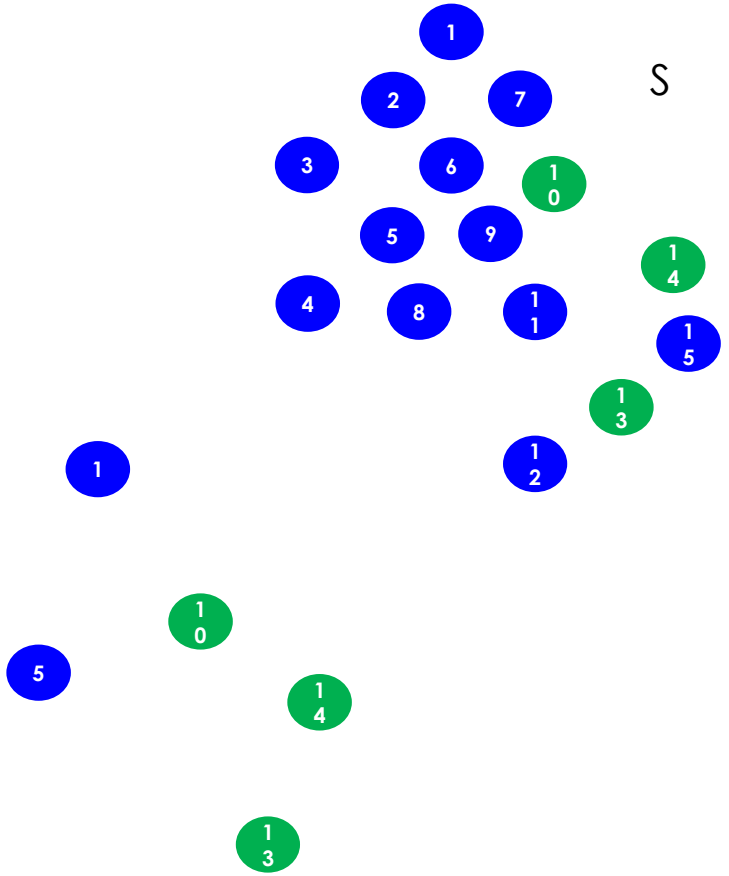
One-side selection (Kubat)

1- Sea S el conjunto original de entrenamiento.

2-Inicializar C con todas las instancias de las clases minoritarias y un ejemplo elegido aleatoriamente por cada clase mayoritaria.

3- Clasificar las instancias en S utilizando la regla 1-NN utilizando los ejemplos en C. Adicionar a C las instancias mal clasificadas.

4- Eliminar de C los ejemplos de la clase mayoritaria que participen en enlaces de Tomek.



Submuestreo

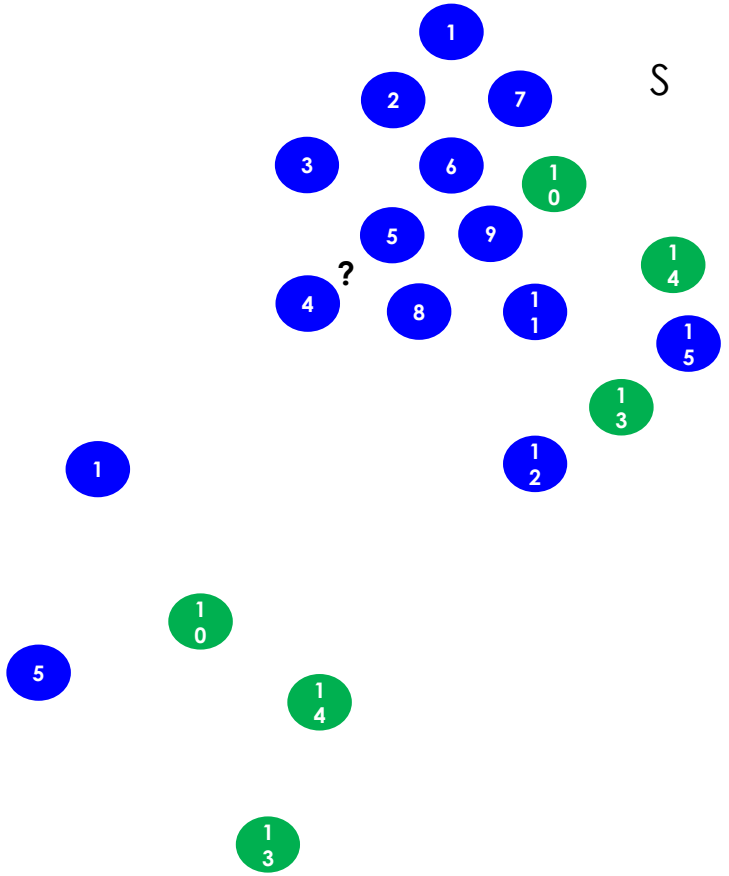
One-side selection (Kubat)

1- Sea S el conjunto original de entrenamiento.

2-Inicializar C con todas las instancias de las clases minoritarias y un ejemplo elegido aleatoriamente por cada clase mayoritaria.

3- Clasificar las instancias en S utilizando la regla 1-NN utilizando los ejemplos en C. Adicionar a C las instancias mal clasificadas.

4- Eliminar de C los ejemplos de la clase mayoritaria que participen en enlaces de Tomek.



Submuestreo

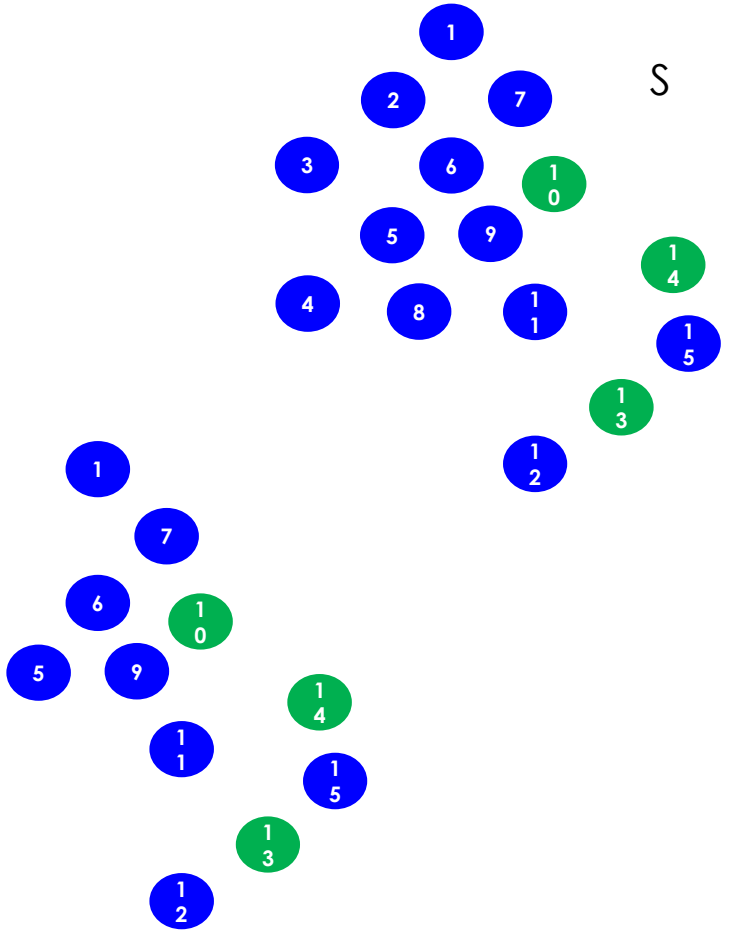
One-side selection (Kubat)

1- Sea S el conjunto original de entrenamiento.

2-Inicializar C con todas las instancias de las clases minoritarias y un ejemplo elegido aleatoriamente por cada clase mayoritaria.

3- Clasificar las instancias en S utilizando la regla 1-NN utilizando los ejemplos en C. Adicionar a C las instancias mal clasificadas.

4- Eliminar de C los ejemplos de la clase mayoritaria que participen en enlaces de Tomek.



Submuestreo

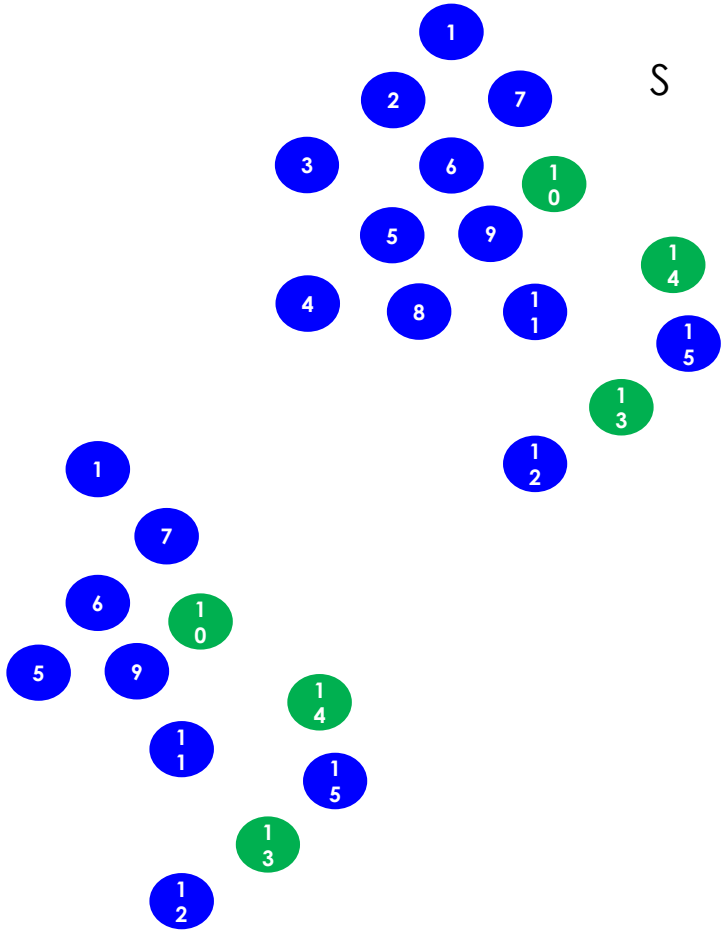
One-side selection (Kubat)

1- Sea S el conjunto original de entrenamiento.

2-Inicializar C con todas las instancias de las clases minoritarias y un ejemplo elegido aleatoriamente por cada clase mayoritaria.

3- Clasificar las instancias en S utilizando la regla 1-NN utilizando los ejemplos en C. Adicionar a C las instancias mal clasificadas.

4- Eliminar de C los ejemplos de la clase mayoritaria que participen en enlaces de Tomek.



Submuestreo

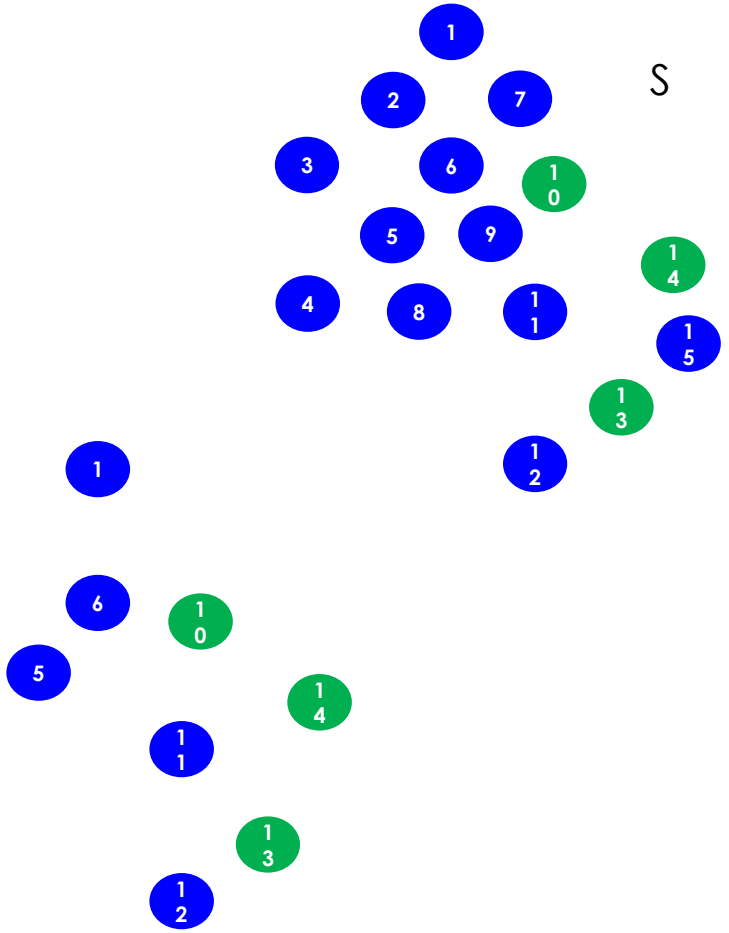
One-side selection (Kubat)

1- Sea S el conjunto original de entrenamiento.

2-Inicializar C con todas las instancias de las clases minoritarias y un ejemplo elegido aleatoriamente por cada clase mayoritaria.

3- Clasificar las instancias en S utilizando la regla 1-NN utilizando los ejemplos en C. Adicionar a C las instancias mal clasificadas.

4- Eliminar de C los ejemplos de la clase mayoritaria que participen en enlaces de Tomek.



Sobremuestreo

SMOTE (**S**ynthetic **M**inority **O**ver-sampling **TE**chnique)

1- Sean: N el % de sobremuestreo, x_i instancias de la clase minoritaria, K el # de vecinos más cercanos

2- Para cada $x_i \in Clase\ minoritaria$

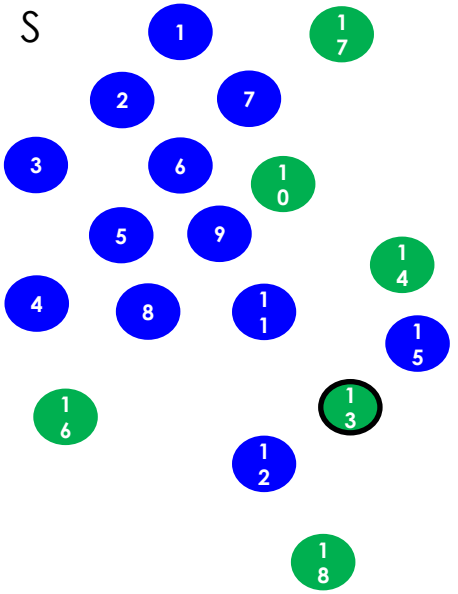
2.1 Buscar los k-vecinos de igual clase de x_i, sean estos , NN_{xi}^{k}

2.2 Seleccionar aleatoriamente $\frac{N}{100}$ vecinos de entre NN_{xi}^{k}, sea este grupo $NN_{xi}^{N/100}$

2.3 Para cada $nn_i \in NN_{xi}^{N/100}$, crear instancia artificial entre x_i y nn_i.

Fin

Sobremuestreo



SMOTE (**S**ynthetic **M**inority **O**ver-sampling **TE**chnique)

1- Sean: $\mathrm{N} = 200$ el % de sobremuestreo, x_i instancias de la clase minoritaria, $\mathrm{K} = 3$ el # de vecinos más cercanos

2- Para $x_{13} \in$ *Clase minoritaria*

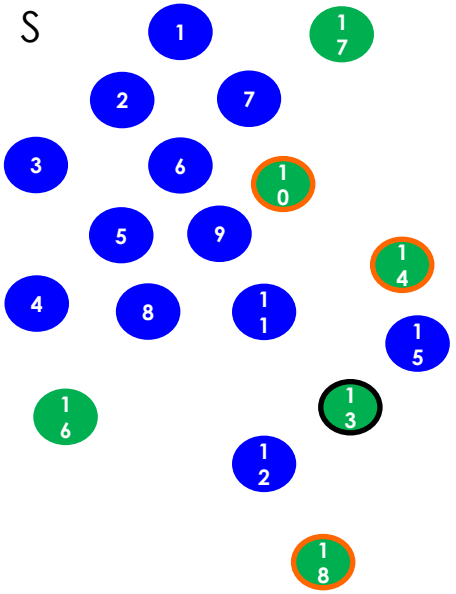
2.1 Buscar los k-vecinos de igual clase de x_{13}, sean estos , NN_{xi}^{k}

2.2 Seleccionar aleatoriamente $\frac{N}{100}$ vecinos de x_i entre NN_{xi}^{k}, sea este grupo $NN_{xi}^{N/100}$

2.3 Para cada $nn_i \in NN_{xi}^{N/100}$, crear instancia artificial entre x_i y nn_i.

Fin

Sobremuestreo



SMOTE (**S**ynthetic **M**inority **O**ver-sampling **TE**chnique)

1- Sean: $\mathrm{N} = 200$ el % de sobremuestreo, x_i instancias de la clase minoritaria, $\mathrm{K} = 3$ el # de vecinos más cercanos

2- Para cada $x_i \in Clase\ minoritaria$

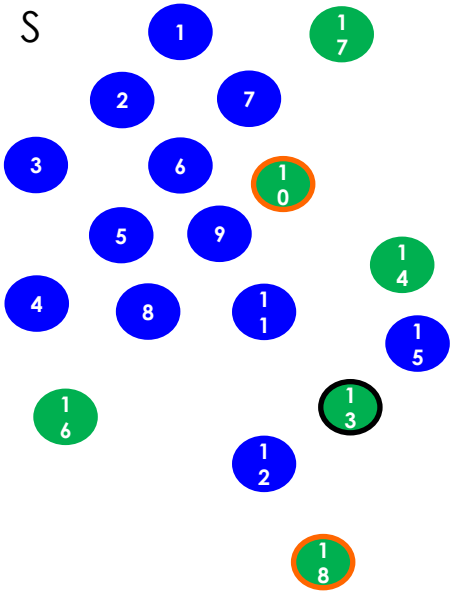
2.1 Buscar los k-vecinos de igual clase de x_{13}, sean estos , $NN_{xi}^{k} = \{x_{14}, x_{18}, x_{10}\}$

2.2 Seleccionar aleatoriamente $\frac{N}{100}$ vecinos de x_i entre NN_{xi}^{k}, sea este grupo $NN_{xi}^{N/100}$

2.3 Para cada $nn_i \in NN_{xi}^{N/100}$, crear instancia artificial entre x_i y nn_i.

Fin

Sobremuestreo



SMOTE (**S**ynthetic **M**inority **O**ver-sampling **TE**chnique)

1- Sean: $\mathrm{N} = 200$ el % de sobremuestreo, x_i instancias de la clase minoritaria, $\mathrm{K} = 3$ el # de vecinos más cercanos

2- Para cada $x_i \in Clase\ minoritaria$

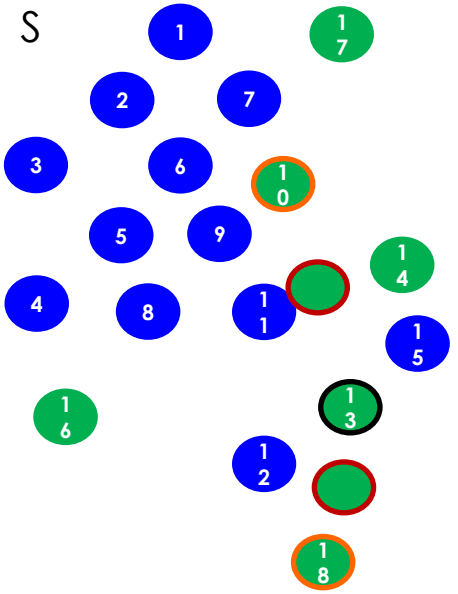
2.1 Buscar los k-vecinos de igual clase de x_{13}, sean estos , $NN_{xi}^{k} = \{x_{14}, x_{18}, x_{10}\}$

2.2 Seleccionar aleatoriamente $\frac{N}{100} = 2$ vecinos de x_{13} entre NN_{xi}^{k}, sea este grupo $NN_{xi}^{N/100} = \{x_{10}, x_{18}\}$

2.3 Para cada $nn_i \in NN_{xi}^{N/100}$, crear instancia artificial entre x_i y nn_i.

Fin

Sobremuestreo



SMOTE (**S**ynthetic **M**inority **O**ver-sampling **TE**chnique)

1- Sean: $\mathrm{N} = 200$ el % de sobremuestreo, x_i instancias de la clase minoritaria, $\mathrm{K} = 3$ el # de vecinos más cercanos

2- Para cada $x_i \in Clase\ minoritaria$

2.1 Buscar los k-vecinos de igual clase de x_{13}, sean estos , $NN_{xi}^{k} = \{x_{14}, x_{18}, x_{10}\}$

2.2 Seleccionar aleatoriamente $\frac{N}{100} = 2$ vecinos de x_{13} entre NN_{xi}^{k}, sea este grupo $NN_{xi}^{N/100} = \{x_{10}, x_{18}\}$

2.3 Para cada $nn_i \in NN_{xi}^{N/100}$, crear instancia artificial entre x_i y nn_i.

Fin

Clasificador K-NN

Sean

- S el conjunto de entrenamiento,
- $x_?$ la instancia a clasificar,
- K número de vecinos más cercanos,
- $d(x_?, x_i)$ medida de disimilitud entre $x_?$ y x_i

1- Buscar NN_{xi}^{k} , k-vecinos más cercanos a x_i

2- Asignar a x_i la clase mayoritaria en NN_{xi}^{k}

Clasificador K-NN

Sean

- S el conjunto de entrenamiento,
- $x_?$ la instancia a clasificar,
- K número de vecinos más cercanos,
- $d(x_?, x_i)$ medida de disimilitud entre $x_?$ y x_i

1- Buscar NN_{xi}^k , k-vecinos más cercanos a x_i

2- Asignar a x_i la clase mayoritaria en NN_{xi}^k

Clasificador K-NN Modificado

Sean

- S el conjunto de entrenamiento,
- $x_?$ la instancia a clasificar,
- K número de vecinos más cercanos,
- $d_m(x_?, x_i)$ medida de disimilitud entre $x_?$ y x_i

1- Buscar NN_{xi}^k , k-vecinos más cercanos a x_i

2- Asignar a x_i la clase mayoritaria en NN_{xi}^k

$$d_m(x_?, x_i) = \frac{{N_t}^m}{N} d(x_?, x_i)$$

ID3 Mejorado

$$\boldsymbol{GA(S,A) = E(S) - \sum_{v \in Valores(A)} \frac{|S_v|}{|S|} E(S_v)}$$

donde:

- $Valores(A)$ es el conjunto de valores posibles del atributo A

- $S_v = \{s \in S | A(s) = v\}$

- $A(s)$= valor del atributo A en la instancia s

$$\boldsymbol{GA(S,A) = E(S) - \sum_{v \in Valores(A)} \frac{|S_v^*|}{|S^*|} \frac{|S_v|}{|S|} E(S_v)}$$

donde:

S^* denota la clase minoritaria.

Notar que el enfoque se centra en el caso de una clase minoritaria.

Fin