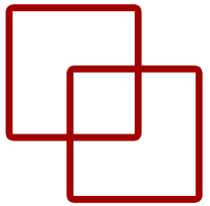
Bootstrap, Bagging y Stacking

Dr. Manuel Castillo-Cara, Luis Sarro
www.manuelcastillo.eu
Departamento de Inteligencia Artificial
Escuela Técnica Superior de Ingeniería Informática
Universidad Nacional de Educación a Distancia (UNED)

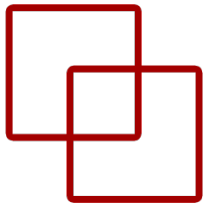
Índice



- Método Bootstrap
- Bootstrap Aggregation
  - Random Forest
  - Estimar el rendimiento
  - Tutorial Bagging
  - Otros modelos Bagging
- Boosting
  - Strong & Weak lerners
  - AdaBoost – Tutorial
- Stacking

Método Bootstrap

Definición



- Bootstrap es un poderoso método estadístico para estimar una cantidad a partir de una muestra de datos.
  - Fácil de entender si la cantidad es una estadística descriptiva como una media o una desviación estándar.

Definición

- Bootstrap es un poderoso método estadístico para estimar una cantidad a partir de una muestra de datos.
  - Fácil de entender si la cantidad es una estadística descriptiva como una media o una desviación estándar.
- Supongamos que tenemos una muestra de 100 valores *(X)* y nos gustaría obtener una estimación de la media de la muestra como

$$mean(x) = \frac{1}{100} \times \sum_{i=1}^{100} x_i$$

Definición

- Bootstrap es un poderoso método estadístico para estimar una cantidad a partir de una muestra de datos.
  - Fácil de entender si la cantidad es una estadística descriptiva como una media o una desviación estándar.
- Supongamos que tenemos una muestra de 100 valores *(X)* y nos gustaría obtener una estimación de la media de la muestra como

$$mean(x) = \frac{1}{100} \times \sum_{i=1}^{100} x_i$$

- Sabemos que nuestra muestra es pequeña y que nuestra media tiene error. Podemos mejorar la estimación de nuestra media usando el **procedimiento bootstrap**:
  1. Crear muchas (por ejemplo, 1000) submuestras aleatorias de nuestro conjunto de **datos con reemplazo** (es decir, podemos seleccionar el mismo valor varias veces).
  2. Calcular la media de cada **submuestra**.
  3. Calcular el **promedio** de todas nuestras medias recopiladas y utilícelo como nuestra **media estimada** para los datos.

Definición

- Bootstrap es un po
  - Fácil de enten
    estándar.
- Supongamos que t
  de la muestra com

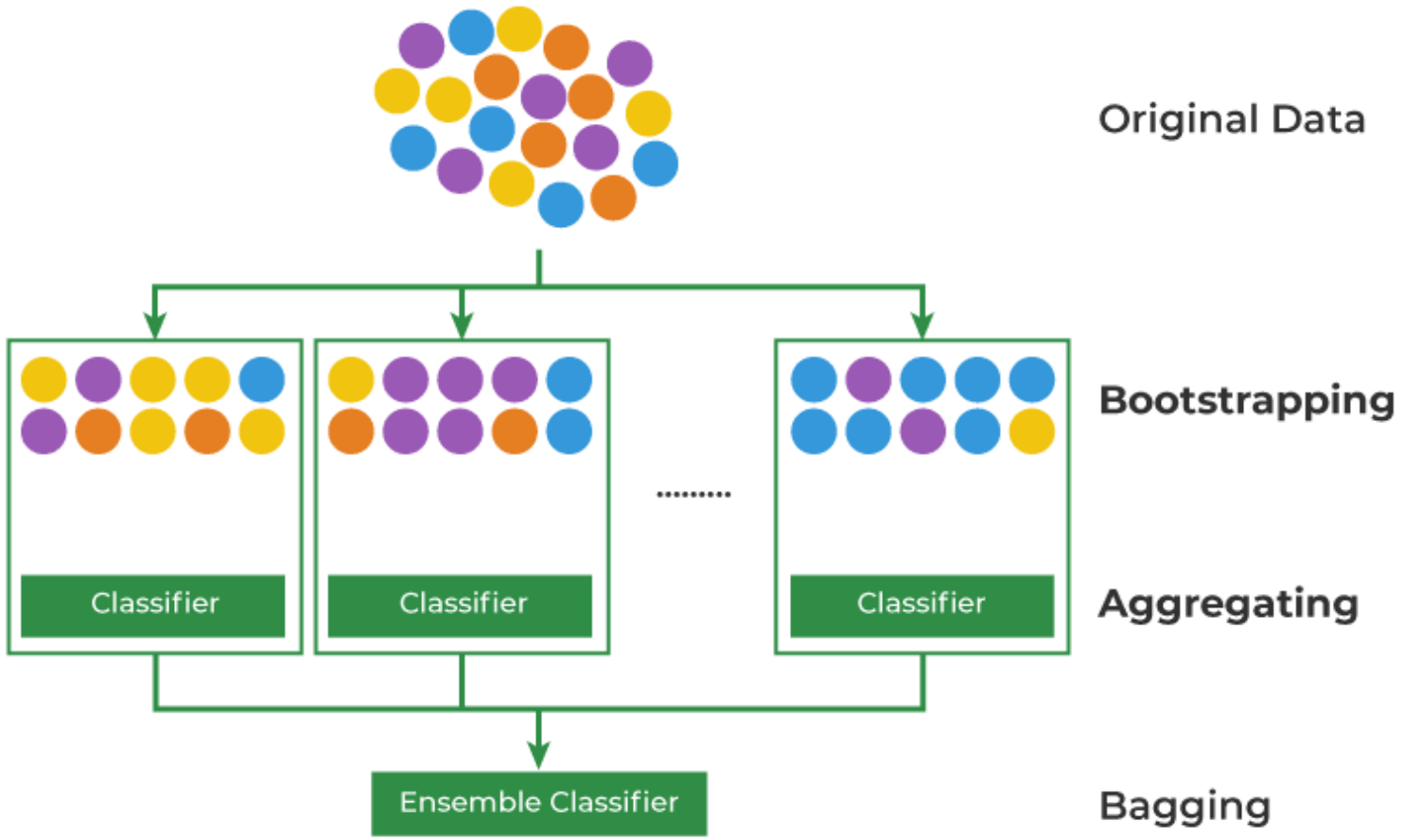
Ejemplo: Utilizamos 3 remuestreos con media: 2.3, 4.5. y 3.3.
Por tanto, el promedio de los remuestreos es: 3.367

- Sabemos que nuestra muestra es pequeña y que nuestra media tiene error. Podemos mejorar la estimación de nuestra media usando el **procedimiento bootstrap**:

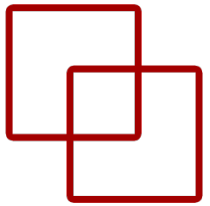
1. Crear muchas (por ejemplo, 1000) submuestras aleatorias de nuestro conjunto de **datos con reemplazo** (es decir, podemos seleccionar el mismo valor varias veces).
2. Calcular la media de cada **submuestra**.
3. Calcular el **promedio** de todas nuestras medias recopiladas y utilícelo como nuestra **media estimada** para los datos.

Bootstrap Aggregation

Bootstrap Aggregation (Bagging)

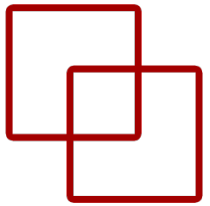


Bootstrap Aggregation (Bagging)
Definición (I)



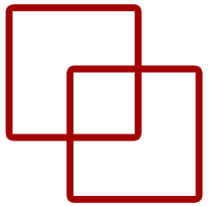
- **Combina** las predicciones de múltiples algoritmos.

Bootstrap Aggregation (Bagging)
Definición (I)



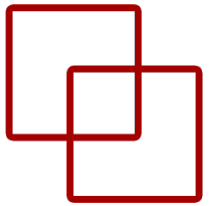
- **Combina** las predicciones de múltiples algoritmos.
- Debilidad: Si se cambian los datos de entrenamiento (e.g., se entrena un árbol con un subconjunto de datos de entrenamiento), el **árbol** de decisión resultante puede ser bastante **diferente** y, a su vez, las **predicciones** pueden ser bastante **diferentes**.

Bootstrap Aggregation (Bagging)
Definición (I)



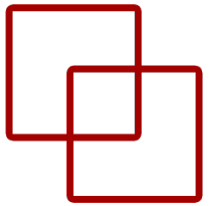
- **Combina** las predicciones de múltiples algoritmos.
- Debilidad: Si se cambian los datos de entrenamiento (e.g., se entrena un árbol con un subconjunto de datos de entrenamiento), el **árbol** de decisión resultante puede ser bastante **diferente** y, a su vez, las **predicciones** pueden ser bastante **diferentes**.
- Permite **reducir la varianza** de algoritmos con alta varianza (e.g. CART tiene una gran varianza).

Bootstrap Aggregation (Bagging)
Definición (I)



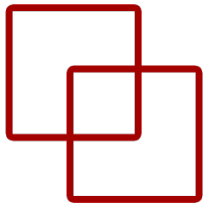
- **Combina** las predicciones de múltiples algoritmos.
- Debilidad: Si se cambian los datos de entrenamiento (e.g., se entrena un árbol con un subconjunto de datos de entrenamiento), el **árbol** de decisión resultante puede ser bastante **diferente** y, a su vez, las **predicciones** pueden ser bastante **diferentes**.
- Permite **reducir la varianza** de algoritmos con alta varianza (e.g. CART tiene una gran varianza).
- Ejemplo: dataset de 1000 instancias usando CART. Bagging funcionaría así:
  1. Crear muchas (por ejemplo, 100) **submuestras** aleatorias de nuestro conjunto de datos con reemplazo.
  2. Entrenar un modelo **CART en cada muestra**.
  3. Dado un nuevo dataset, calcular la predicción **promedio** de cada modelo.

Bootstrap Aggregation (Bagging)
Definición (II)



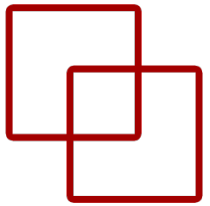
- El parámetro relevante de Bagging es la **cantidad de árboles** a crear:
  - Esto se puede elegir aumentándolo hasta que comience a **dejar de mostrar mejoras** (e.g, *k-fold cross validation*).
  - Una gran cantidad de árboles puede llevar mucho tiempo, pero no **sobreajustará** los datos de entrenamiento.

Bootstrap Aggregation (Bagging)
Definición (II)



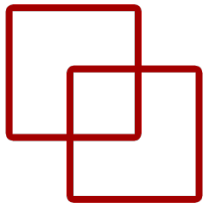
- El parámetro relevante de Bagging es la **cantidad de árboles** a crear:
  - Esto se puede elegir aumentándolo hasta que comience a **dejar de mostrar mejoras** (e.g, *k-fold cross validation*).
  - Una gran cantidad de árboles puede llevar mucho tiempo, pero no **sobreajustará** los datos de entrenamiento.
- Ejemplo: Tenemos 5 CART con las siguientes predicciones para **una misma instancia** de entrada: azul, azul, rojo, azul y rojo → Tomaríamos la clase más frecuente y predeciríamos azul.

Bootstrap Aggregation (Bagging)
Definición (II)



- El parámetro relevante de Bagging es la **cantidad de árboles** a crear:
  - Esto se puede elegir aumentándolo hasta que comience a **dejar de mostrar mejoras** (e.g, *k-fold cross validation*).
  - Una gran cantidad de árboles puede llevar mucho tiempo, pero no **sobreajustará** los datos de entrenamiento.
- Ejemplo: Tenemos 5 CART con las siguientes predicciones para **una misma instancia** de entrada: azul, azul, rojo, azul y rojo → Tomaríamos la clase más frecuente y predeciríamos azul.
- Nos preocupa poco que los árboles individuales se ajusten a *train*. Por esta razón y por motivos de eficiencia
  - Los CART individuales se expanden **profundamente** (e.g., pocas muestras de entrenamiento en cada nodo de hoja del árbol); y
  - Los árboles **no** se **podan**.

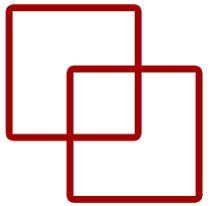
Bootstrap Aggregation (Bagging)
Definición (II)



- El parámetro relevante de Bagging es la **cantidad de árboles** a crear:
  - Esto se puede elegir aumentándolo hasta que comience a **dejar de mostrar mejoras** (e.g, *k-fold cross validation*).
  - Una gran cantidad de árboles puede llevar mucho tiempo, pero no **sobreajustará** los datos de entrenamiento.
- Ejemplo: Tenemos 5 CART con las siguientes predicciones para **una misma instancia** de entrada: azul, azul, rojo, azul y rojo → Tomaríamos la clase más frecuente y predeciríamos azul.
- Nos preocupa poco que los árboles individuales se ajusten a *train*. Por esta razón y por motivos de eficiencia
  - Los CART individuales se expanden **profundamente** (e.g., pocas muestras de entrenamiento en cada nodo de hoja del árbol); y
  - Los árboles **no** se **podan**.
- Estos árboles tendrán una **alta varianza y un sesgo bajo**.
  - Característica principal de los submodelos cuando se combinan predicciones mediante *Bagging*.

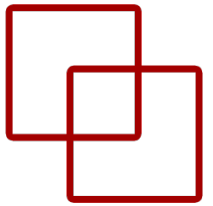
Random Forest

Random Forest
Definición (I)



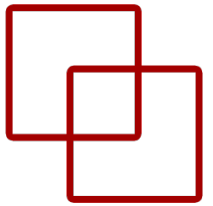
- RF son una mejora a Bagging.
- Un problema con los árboles de decisión como CART es que son **codiciosos**.

Random Forest
Definición (I)



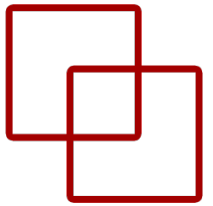
- RF son una mejora a Bagging.
- Un problema con los árboles de decisión como CART es que son **codiciosos**.
  - Eligen en qué variable dividir usando un algoritmo *Greedy* que **minimiza el error**.

Random Forest
Definición (I)



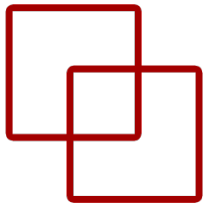
- RF son una mejora a Bagging.
- Un problema con los árboles de decisión como CART es que son **codiciosos**.
  - Eligen en qué variable dividir usando un algoritmo *Greedy* que **minimiza el error**.
  - Con Bagging, los árboles de decisión pueden tener muchas **similitudes estructurales** y, a su vez, dar lugar a una **alta correlación** en sus predicciones.

Random Forest
Definición (I)



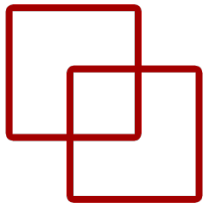
- RF son una mejora a Bagging.
- Un problema con los árboles de decisión como CART es que son **codiciosos**.
  - Eligen en qué variable dividir usando un algoritmo *Greedy* que **minimiza el error**.
  - Con Bagging, los árboles de decisión pueden tener muchas **similitudes estructurales** y, a su vez, dar lugar a una **alta correlación** en sus predicciones.
  - Combinar predicciones de múltiples modelos funciona mejor si las **predicciones de los submodelos no están correlacionadas** o, en el mejor de los casos, están **débilmente** correlacionadas.

Random Forest
Definición (II)



- RF cambia el algoritmo de la forma en que se aprenden los subárboles para que las predicciones resultantes de todos los subárboles tengan menos correlación
  - En CART → al seleccionar un punto de división, el algoritmo de aprendizaje puede examinar todas las variables y todos los valores de las variables para seleccionar el punto de división óptimo.

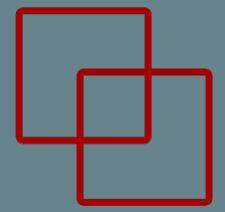
Random Forest
Definición (II)



- RF cambia el algoritmo de la forma en que se aprenden los subárboles para que las predicciones resultantes de todos los subárboles tengan menos correlación
  - En CART → al seleccionar un punto de división, el algoritmo de aprendizaje puede examinar todas las variables y todos los valores de las variables para seleccionar el punto de división óptimo.
- RF cambia este procedimiento para que el algoritmo de aprendizaje se limite a una **muestra aleatoria** de características entre las que buscar.
  - El **número de características** que se pueden buscar en cada punto de división (m) debe especificarse como parámetro del algoritmo.
  - Puede **probar diferentes valores** y ajustarlos mediante validación cruzada.

Random Forest
Definición (II)

- RF cambia el algoritmo de la forma en que se aprenden los subárboles para que las predicciones resultantes de todos los subárboles tengan menos correlación
  - En CART → al seleccionar un punto de división, el algoritmo de aprendizaje puede examinar todas las variables y todos los valores de las variables para seleccionar el punto de división óptimo.
- RF cambia este procedimiento para que el algoritmo de aprendizaje se limite a una **muestra aleatoria** de características entre las que buscar.
  - El **número de características** que se pueden buscar en cada punto de división (m) debe especificarse como parámetro del algoritmo.
  - Puede **probar diferentes valores** y ajustarlos mediante validación cruzada.
- El valor de m dependerá del problema, e.g., si se tiene $p=25$ variables de entrada:
  - **Clasificación** sería → $m = sqrt(p) = sqrt(25) = 5$
  - **Regresión** sería → $m = p \ / \ 3 = 25 \ / \ 3 = 8.333 = 8$

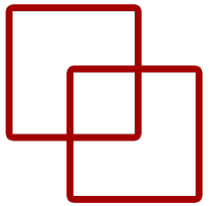


UNED

ETS de Ingeniería Informática

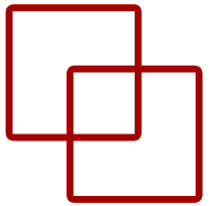
Estimar el rendimiento

Estimar el rendimiento



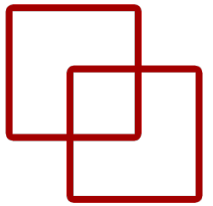
- Para cada muestra *bootstrap* tomada de los datos de entrenamiento, quedarán muestras que no se incluyeron.
  - Estas muestras se denominan muestras fuera de bolsa (**OOB – Out-of-bag**).

Estimar el rendimiento



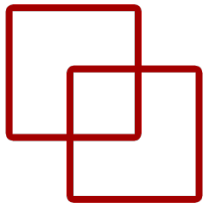
- Para cada muestra *bootstrap* tomada de los datos de entrenamiento, quedarán muestras que no se incluyeron.
  - Estas muestras se denominan muestras fuera de bolsa (**OOB – Out-of-bag**).
- El rendimiento de cada modelo en las **muestras omitidas**, cuando se promedia, puede proporcionar una métrica estimada de los modelos Bagging.

Estimar el rendimiento



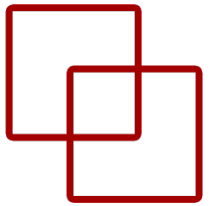
- Para cada muestra *bootstrap* tomada de los datos de entrenamiento, quedarán muestras que no se incluyeron.
  - Estas muestras se denominan muestras fuera de bolsa (**OOB – Out-of-bag**).
- El rendimiento de cada modelo en las **muestras omitidas**, cuando se promedia, puede proporcionar una métrica estimada de los modelos Bagging.
- Este rendimiento estimado a menudo se denomina **estimación OOB**.

Estimar el rendimiento



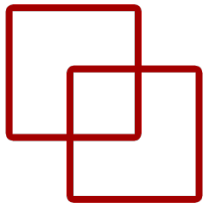
- Para cada muestra *bootstrap* tomada de los datos de entrenamiento, quedarán muestras que no se incluyeron.
  - Estas muestras se denominan muestras fuera de bolsa (**OOB – Out-of-bag**).
- El rendimiento de cada modelo en las **muestras omitidas**, cuando se promedia, puede proporcionar una métrica estimada de los modelos Bagging.
- Este rendimiento estimado a menudo se denomina **estimación OOB**.
- Estas medidas de rendimiento son una **estimación confiable del error de test** y se correlacionan bien con las estimaciones de error de validación cruzada.

Importancia de variables



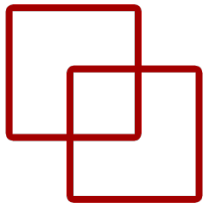
- A medida que se construyen los árboles de decisión Bagging, podemos calcular cuánto cae la función de **error para una variable en cada punto de división**.

Importancia de variables



- A medida que se construyen los árboles de decisión Bagging, podemos calcular cuánto cae la función de **error para una variable en cada punto de división**.
- En problemas de regresión, esto puede ser la caída en el error de **la suma al cuadrado** y, en clasificación, podría ser la **puntuación de Gini**. (e.g. si baja mucho la pérdida)

Importancia de variables

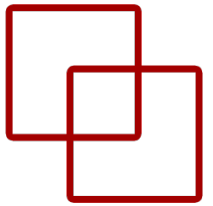


- A medida que se construyen los árboles de decisión Bagging, podemos calcular cuánto cae la función de **error para una variable en cada punto de división**.
- En problemas de regresión, esto puede ser la caída en el error de **la suma al cuadrado** y, en clasificación, podría ser la **puntuación de Gini**. (e.g. si baja mucho la pérdida)
- Estas caídas en el error se pueden promediar en todos los árboles y sus resultados para proporcionar una estimación de la **importancia de cada variable de entrada**.

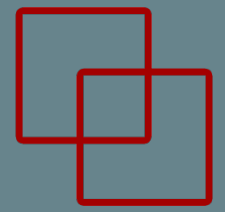
Importancia de variables

- A medida que se construyen los árboles de decisión Bagging, podemos calcular cuánto cae la función de **error para una variable en cada punto de división**.
- En problemas de regresión, esto puede ser la caída en el error de **la suma al cuadrado** y, en clasificación, podría ser la **puntuación de Gini**. (e.g. si baja mucho la pérdida)
- Estas caídas en el error se pueden promediar en todos los árboles y sus resultados para proporcionar una estimación de la **importancia de cada variable de entrada**.
- Cuanto **mayor fue la caída** (en el error) cuando se eligió la variable, **mayor fue su importancia.**

Importancia de variables



- A medida que se construyen los árboles de decisión Bagging, podemos calcular cuánto cae la función de **error para una variable en cada punto de división**.
- En problemas de regresión, esto puede ser la caída en el error de **la suma al cuadrado** y, en clasificación, podría ser la **puntuación de Gini**. (e.g. si baja mucho la pérdida)
- Estas caídas en el error se pueden promediar en todos los árboles y sus resultados para proporcionar una estimación de la **importancia de cada variable de entrada**.
- Cuanto **mayor fue la caída** (en el error) cuando se eligió la variable, **mayor fue su importancia.**
- Estos resultados pueden ayudar a identificar subconjuntos de **variables** de entrada que pueden ser **más o menos relevantes** para el problema.



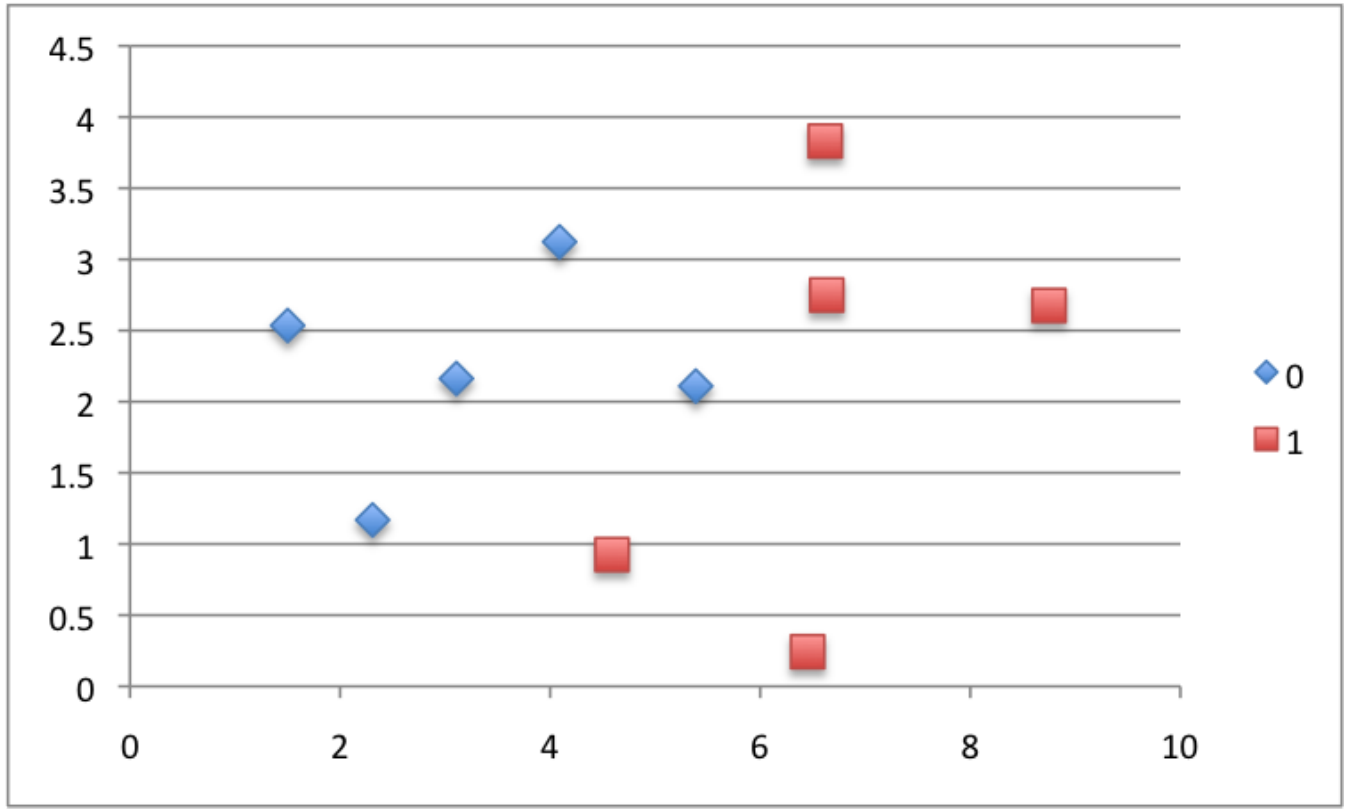
UNED

ETS de Ingeniería Informática

Tutorial Bagging

Dataset

X1	X2	Y
2.309572387	1.168959634	0
1.500958319	2.535482186	0
3.107545266	2.162569456	0
4.090032824	3.123409313	0
5.38660215	2.109488166	0
6.451823468	0.242952387	1
6.633669528	2.749508563	1
8.749958452	2.676022211	1
4.589131161	0.925340325	1
6.619322828	3.831050828	1



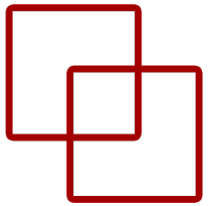
Modelos

- La parte interesante de Bagging es cómo se **combinan** para hacer predicciones de conjunto.
- Idearemos **3 árboles de decisión** a partir de los datos de entrenamiento.
- Los **puntos de división** se eligieron manualmente para demostrar exactamente cómo se pueden combinar diferentes perspectivas sobre el mismo problema para proporcionar un mayor rendimiento.
- Un árbol de decisión con un solo punto de división se llama **tocón de decisión**.
- Usaremos **3 tocones de decisión** con los siguientes puntos de división:

$$Model1 : X1 \leq 5.38660215$$
$$Model2 : X1 \leq 4.090032824$$
$$Model3 : X2 \leq 0.925340325$$

Predicciones – Modelo 1

$$Model1 : X1 \leq 5.38660215$$
$$Model2 : X1 \leq 4.090032824$$
$$Model3 : X2 \leq 0.925340325$$



- Modelo 1 --> X1 ≤ 5,38660215; separamos los datos en **2 grupos**.
- Las instancias asignadas a los 2 grupos serán clasificadas:
  - (i) LEFT: clase 0; y (ii) RIGHT: clase 1

Model 1 Predictions

X1	X2	Y	X1 Split	Group	Prediction	Error	Accuracy
2,30957239	1,168959634	0	5,38660215	LEFT	0	0	90
1,50095832	2,535482186	0		LEFT	0	0	
3,10754527	2,162569456	0		LEFT	0	0	
4,09003282	3,123409313	0		LEFT	0	0	
5,38660215	2,109488166	0		LEFT	0	0	
6,45182347	0,242952387	1		RIGHT	1	0	
6,63366953	2,749508563	1		RIGHT	1	0	
8,74995845	2,676022211	1		RIGHT	1	0	
4,58913116	0,925340325	1		LEFT	0	1	
6,61932283	3,831050828	1		RIGHT	1	0	

Predicciones – Modelo 1

$$Model1 : X1 \leq 5.38660215$$
$$Model2 : X1 \leq 4.090032824$$
$$Model3 : X2 \leq 0.925340325$$

- Modelo 1 --> X1 ≤ 5,38660215; separamos los datos en **2 grupos**.
- Las instancias asignadas a los 2 grupos serán clasificadas:
  - (i) LEFT: clase 0; y (ii) RIGHT: clase 1
- El modelo solo obtuvo 1 instancia incorrecta → Accuracy del 90%.

Model 1 Predictions

X1	X2	Y	X1 Split	Group	Prediction	Error	Accuracy
2,30957239	1,168959634	0	5,38660215	LEFT	0	0	90
1,50095832	2,535482186	0		LEFT	0	0	
3,10754527	2,162569456	0		LEFT	0	0	
4,09003282	3,123409313	0		LEFT	0	0	
5,38660215	2,109488166	0		LEFT	0	0	
6,45182347	0,242952387	1		RIGHT	1	0	
6,63366953	2,749508563	1		RIGHT	1	0	
8,74995845	2,676022211	1		RIGHT	1	0	
4,58913116	0,925340325	1		LEFT	0	1	
6,61932283	3,831050828	1		RIGHT	1	0	

Predicciones – Modelo 2

$$Model1 : X1 \leq 5.38660215$$
$$Model2 : X1 \leq 4.090032824$$
$$Model3 : X2 \leq 0.925340325$$

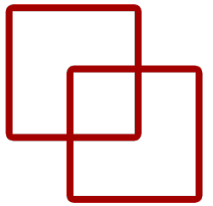
- Modelo 2: X1 ≤ 4,090032824; separamos los datos en 2 grupos.
- Las instancias asignadas a los dos grupos serán clasificados:
  - (i) LEFT: clase 0; y (ii) RIGHT: clase 1
- El modelo solo obtuvo una instancia incorrecta → Accuracy del 90%.

Model 2 Predictions

X1	X2	Y	X1 Split	Group	Prediction	Error	Accuracy
2,30957239	1,168959634	0	4,09003282	LEFT	0	0	90
1,50095832	2,535482186	0		LEFT	0	0	
3,10754527	2,162569456	0		LEFT	0	0	
4,09003282	3,123409313	0		LEFT	0	0	
5,38660215	2,109488166	0		RIGHT	1	1	
6,45182347	0,242952387	1		RIGHT	1	0	
6,63366953	2,749508563	1		RIGHT	1	0	
8,74995845	2,676022211	1		RIGHT	1	0	
4,58913116	0,925340325	1		RIGHT	1	0	
6,61932283	3,831050828	1		RIGHT	1	0	

Predicciones – Modelo 2

$$Model1 : X1 \leq 5.38660215$$
$$Model2 : X1 \leq 4.090032824$$
$$Model3 : X2 \leq 0.925340325$$



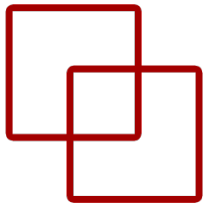
- Modelo 2: X1 ≤ 4,090032824; separamos los datos en 2 grupos.
- Las instancias asignadas a los dos grupos serán clasificados:
  - (i) LEFT: clase 0; y (ii) RIGHT: clase 1

Model 2 Predictions

X1	X2	Y	X1 Split	Group	Prediction	Error	Accuracy
2,30957239	1,168959634	0	4,09003282	LEFT	0	0	90
1,50095832	2,535482186	0		LEFT	0	0	
3,10754527	2,162569456	0		LEFT	0	0	
4,09003282	3,123409313	0		LEFT	0	0	
5,38660215	2,109488166	0		RIGHT	1	1	
6,45182347	0,242952387	1		RIGHT	1	0	
6,63366953	2,749508563	1		RIGHT	1	0	
8,74995845	2,676022211	1		RIGHT	1	0	
4,58913116	0,925340325	1		RIGHT	1	0	
6,61932283	3,831050828	1		RIGHT	1	0	

Predicciones – Modelo 3

$$Model1 : X1 \leq 5.38660215$$
$$Model2 : X1 \leq 4.090032824$$
$$Model3 : X2 \leq 0.925340325$$



- Modelo 3: X2 ≤ 0,925340325; separamos los datos en 2 grupos.
- Las instancias asignadas a los dos grupos serán clasificados:
  - (i) RIGHT: clase 0; y (ii) LEFT: clase 1

Model 3 Predictions

X1	X2	Y	X2 Split	Group	Prediction	Error	Accuracy
2,30957239	1,168959634	0	0,92534033	RIGHT	0	0	70
1,50095832	2,535482186	0		RIGHT	0	0	
3,10754527	2,162569456	0		RIGHT	0	0	
4,09003282	3,123409313	0		RIGHT	0	0	
5,38660215	2,109488166	0		RIGHT	0	0	
6,45182347	0,242952387	1		LEFT	1	0	
6,63366953	2,749508563	1		RIGHT	0	1	
8,74995845	2,676022211	1		RIGHT	0	1	
4,58913116	0,925340325	1		LEFT	1	0	
6,61932283	3,831050828	1		RIGHT	0	1	

Predicciones – Modelo 3

$$Model1 : X1 \leq 5.38660215$$
$$Model2 : X1 \leq 4.090032824$$
$$Model3 : X2 \leq 0.925340325$$

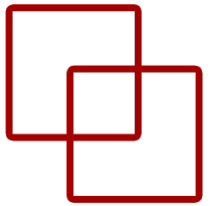
- Modelo 3: X2 ≤ 0,925340325; separamos los datos en 2 grupos.
- Las instancias asignadas a los dos grupos serán clasificados:
  - (i) RIGHT: clase 0; y (ii) LEFT: clase 1
- Peor rendimiento (pero correctamente la 5ta y penúltima) → Accuracy: 70%.

Model 3 Predictions

X1	X2	Y	X2 Split	Group	Prediction	Error	Accuracy
2,30957239	1,168959634	0	0,92534033	RIGHT	0	0	70
1,50095832	2,535482186	0		RIGHT	0	0	
3,10754527	2,162569456	0		RIGHT	0	0	
4,09003282	3,123409313	0		RIGHT	0	0	
5,38660215	2,109488166	0		RIGHT	0	0	
6,45182347	0,242952387	1		LEFT	1	0	
6,63366953	2,749508563	1		RIGHT	0	1	
8,74995845	2,676022211	1		RIGHT	0	1	
4,58913116	0,925340325	1		LEFT	1	0	
6,61932283	3,831050828	1		RIGHT	0	1	

Predicciones finales

$$Model1 : X1 \leq 5.38660215$$
$$Model2 : X1 \leq 4.090032824$$
$$Model3 : X2 \leq 0.925340325$$



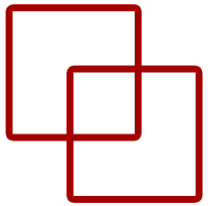
- Para las predicciones tomamos lo más común: moda estadística.
- Las 10 instancias de entrenamiento se clasificaron correctamente con una precisión del 100%.

Bagged Predictions

X1	X2	Predictior	Y	Error	Accuracy
2,30957239	1,168959634	0	0	0	100
1,50095832	2,535482186	0	0	0	
3,10754527	2,162569456	0	0	0	
4,09003282	3,123409313	0	0	0	
5,38660215	2,109488166	0	0	0	
6,45182347	0,242952387	1	1	0	
6,63366953	2,749508563	1	1	0	
8,74995845	2,676022211	1	1	0	
4,58913116	0,925340325	1	1	0	
6,61932283	3,831050828	1	1	0	

Predicciones finales

$$Model1 : X1 \leq 5.38660215$$
$$Model2 : X1 \leq 4.090032824$$
$$Model3 : X2 \leq 0.925340325$$

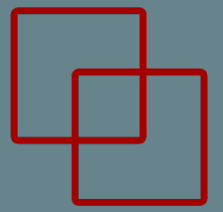


- Para las prediccio estadística.
- Las 10 instancias correctamente con una precisión del 10

Hemos especializado un arbol a cada instancia y al agregarlos en conjunto acertamos, ¡y sin overfitting!

Bagged Predictions

X1	X2	Predictior	Y	Error	Accuracy
2,30957239	1,168959634	0	0	0	100
1,50095832	2,535482186	0	0	0	
3,10754527	2,162569456	0	0	0	
4,09003282	3,123409313	0	0	0	
5,38660215	2,109488166	0	0	0	
6,45182347	0,242952387	1	1	0	
6,63366953	2,749508563	1	1	0	
8,74995845	2,676022211	1	1	0	
4,58913116	0,925340325	1	1	0	
6,61932283	3,831050828	1	1	0	

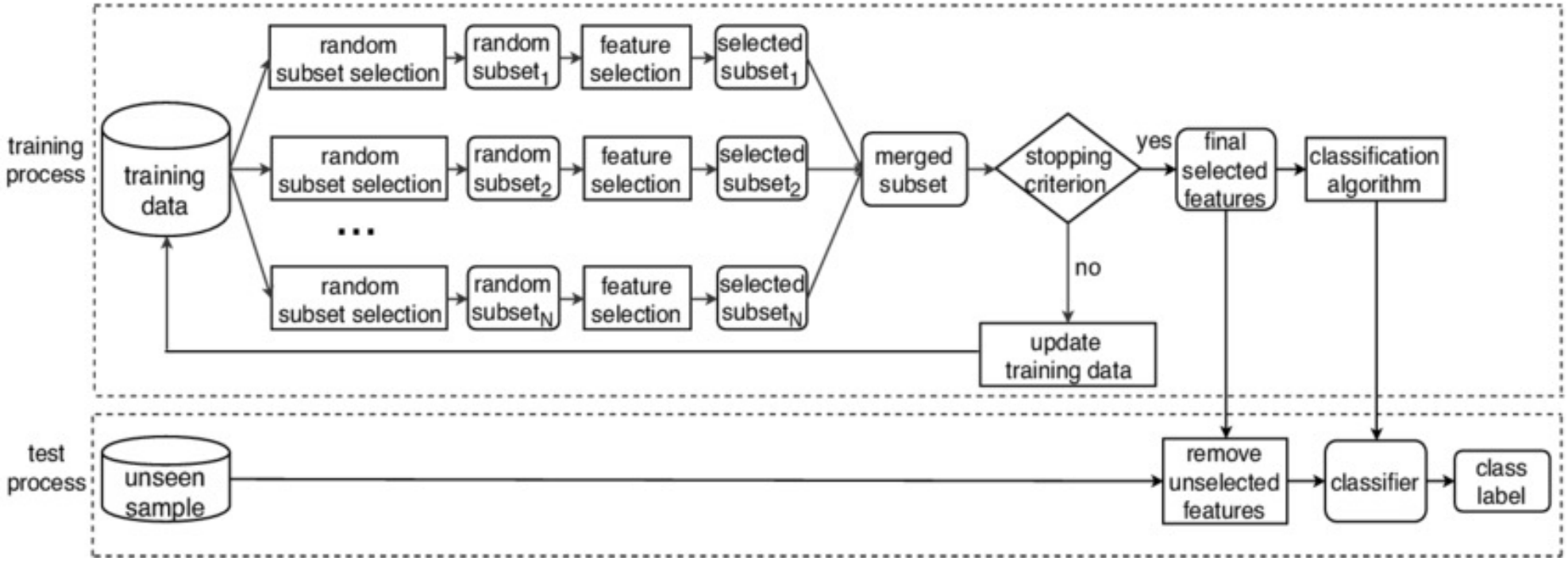


UNED

ETS de Ingeniería Informática

Otros modelos Bagging

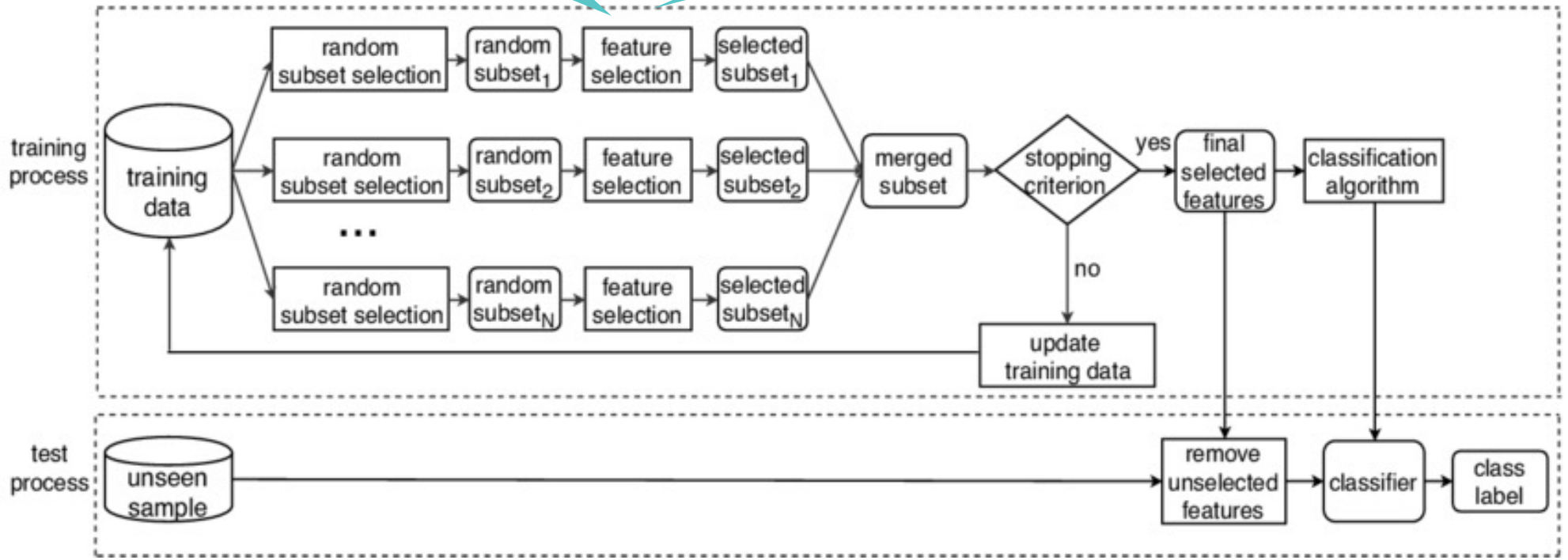
Feature Selection Subspace



Feature Selection Subspace

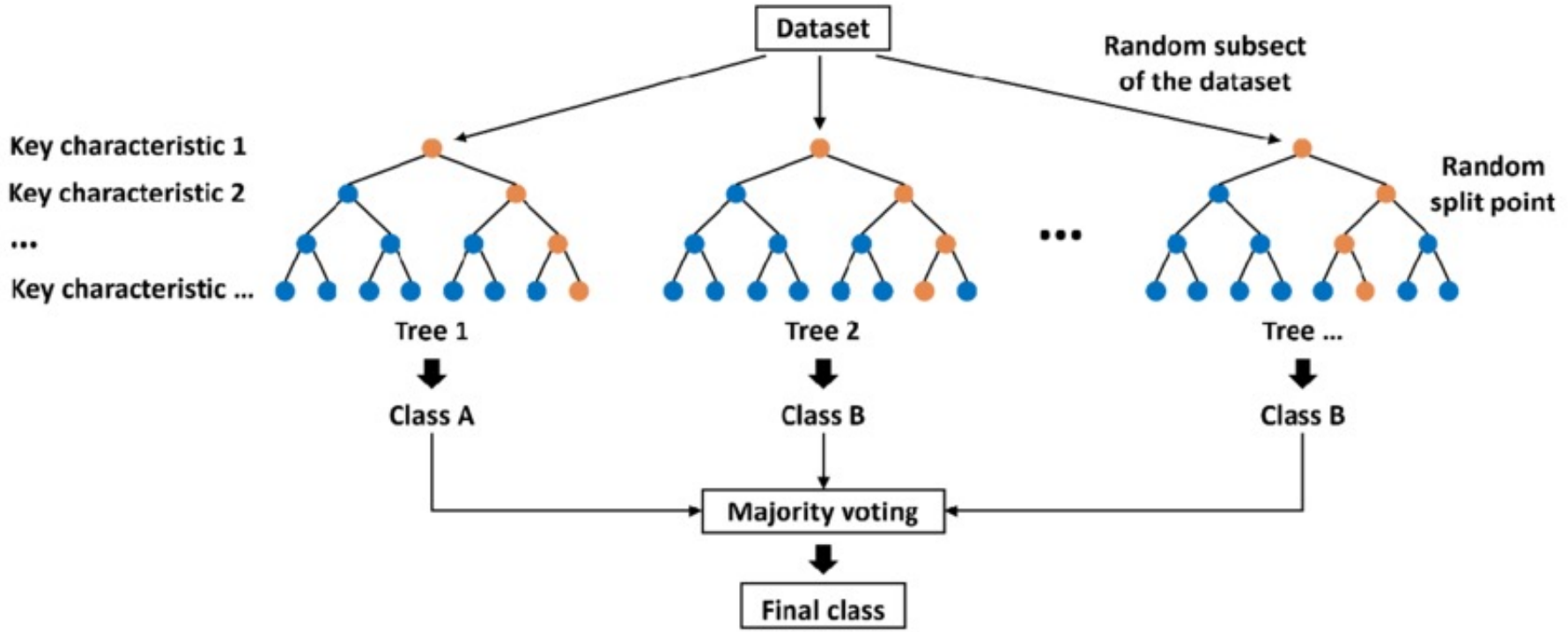
Método simple: misma técnica FS en todos los subespacios

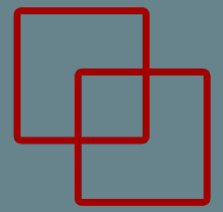
Método Múltiple: Diferentes FS en cada uno de los subespacios



Extra Trees

La diferencia principal con RF es que: RF utiliza Greedy para la división de característica mientras ET lo hace aleatoriamente



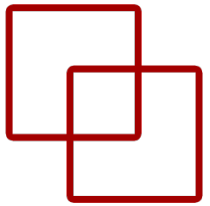


UNED

ETS de Ingeniería Informática

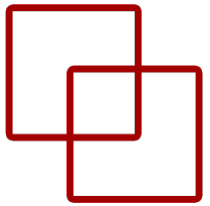
Boosting

Métodos boosting



- Son métodos de conjunto general que crea un **clasificador fuerte a partir de varios clasificadores débiles**.

Métodos boosting

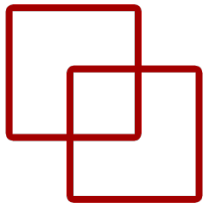


- Son métodos de conjunto general que crea un **clasificador fuerte a partir de varios clasificadores débiles**.
- Se hace construyendo un modelo a partir de *train* y luego creando un segundo modelo que intenta **corregir los errores** del primer modelo.

Métodos boosting

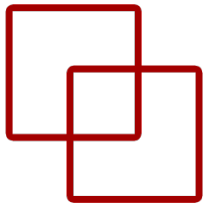
- Son métodos de conjunto general que crea un **clasificador fuerte a partir de varios clasificadores débiles**.
- Se hace construyendo un modelo a partir de *train* y luego creando un segundo modelo que intenta **corregir los errores** del primer modelo.
- Los modelos se agregan hasta que el conjunto de entrenamiento se predice perfectamente o se agrega **un número máximo de modelos**.

Métodos boosting

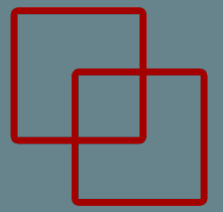


- Son métodos de conjunto general que crea un **clasificador fuerte a partir de varios clasificadores débiles**.
- Se hace construyendo un modelo a partir de *train* y luego creando un segundo modelo que intenta **corregir los errores** del primer modelo.
- Los modelos se agregan hasta que el conjunto de entrenamiento se predice perfectamente o se agrega **un número máximo de modelos**.
- **AdaBoost** fue el primer algoritmo exitoso desarrollado para la clasificación binaria.

Métodos boosting



- Son métodos de conjunto general que crea un **clasificador fuerte a partir de varios clasificadores débiles**.
- Se hace construyendo un modelo a partir de *train* y luego creando un segundo modelo que intenta **corregir los errores** del primer modelo.
- Los modelos se agregan hasta que el conjunto de entrenamiento se predice perfectamente o se agrega **un número máximo de modelos**.
- **AdaBoost** fue el primer algoritmo exitoso desarrollado para la clasificación binaria.
- Los métodos boosting modernos se basan en AdaBoost, sobre todo Grandient Boost Machine.

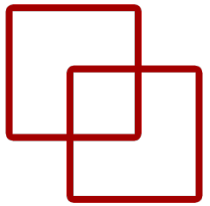


UNED

ETS de Ingeniería Informática

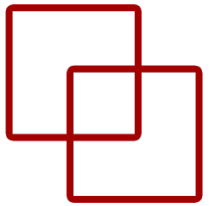
Clasificadores fuertes y débiles

Background



- Es común describir las técnicas de aprendizaje en conjunto en términos de **clasificadores débiles y fuertes**.

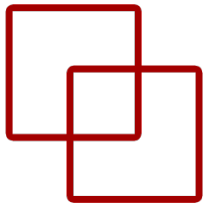
Background



- Es común describir las técnicas de aprendizaje en conjunto en términos de **clasificadores débiles y fuertes**.

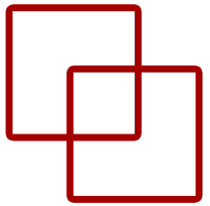
- Por ejemplo, podemos desear construir un clasificador **fuerte** a partir de las predicciones de **muchos clasificadores débiles**.

Clasificador débil



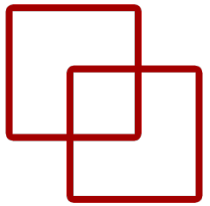
- Funciona ligeramente mejor que un **modelo ingenuo**, i.e., ligeramente superior al 50% (en clasificación binaria).

Clasificador débil



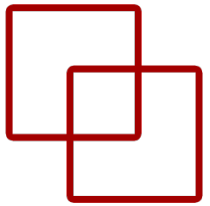
- Funciona ligeramente mejor que un **modelo ingenuo**, i.e., ligeramente superior al 50% (en clasificación binaria).
- Boosting se refiere a la taxonomía de algoritmos que pueden convertir a los **clasificadores débiles en clasificadores fuertes**.

Clasificador débil



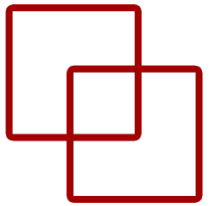
- Funciona ligeramente mejor que un **modelo ingenuo**, i.e., ligeramente superior al 50% (en clasificación binaria).
- Boosting se refiere a la taxonomía de algoritmos que pueden convertir a los **clasificadores débiles en clasificadores fuertes**.
- El tipo de modelo débil más utilizado es **CART** (la debilidad del árbol puede controlarse mediante la profundidad del árbol durante la construcción).

Clasificador débil



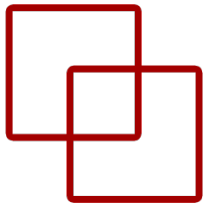
- Funciona ligeramente mejor que un **modelo ingenuo**, i.e., ligeramente superior al 50% (en clasificación binaria).
- Boosting se refiere a la taxonomía de algoritmos que pueden convertir a los **clasificadores débiles en clasificadores fuertes**.
- El tipo de modelo débil más utilizado es **CART** (la debilidad del árbol puede controlarse mediante la profundidad del árbol durante la construcción).
- Otros candidatos de aprendizaje débil:
  - **k-NN con k=1** operando en una o un subconjunto de variables de entrada.
  - **MLP** con un único nodo operando sobre una o un subconjunto de variables de entrada.
  - **Naive Bayes** operando con una única variable de entrada.

Clasificador fuerte



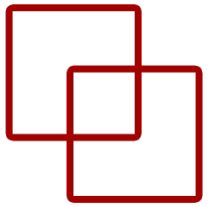
- Es un clasificador que logra una **métrica buena**, i.e., funciona muy bien en comparación con un modelo ingenuo.

Clasificador fuerte



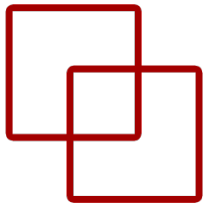
- Es un clasificador que logra una **métrica buena**, i.e., funciona muy bien en comparación con un modelo ingenuo.
- Desarrollamos un clasificador fuerte cuando **ajustamos** un modelo directamente a un conjunto de datos.

Clasificador fuerte



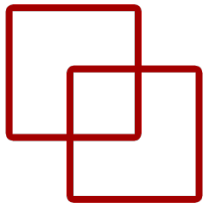
- Es un clasificador que logra una **métrica buena**, i.e., funciona muy bien en comparación con un modelo ingenuo.
- Desarrollamos un clasificador fuerte cuando **ajustamos** un modelo directamente a un conjunto de datos.
- Se puede considerar los siguientes algoritmos como técnicas para ajustar un clasificador fuerte (los **hiperparámetros** se ajustan al problema objetivo):
  - Regresión logística.
  - Máquinas de vectores soporte.
  - k-NN.

Clasificador fuerte Vs. Débil



- Se puede construir un clasificador fuerte a partir de **muchos** clasificadores débiles.
  - Clasificador débil: fácil de preparar, pero no deseable debido a su baja habilidad.
  - Clasificador fuerte: Difícil de preparar, pero deseable debido a su gran habilidad.

Clasificador fuerte Vs. Débil



- Se puede construir un clasificador fuerte a partir de **muchos** clasificadores débiles.
  - Clasificador débil: fácil de preparar, pero no deseable debido a su baja habilidad.
  - Clasificador fuerte: Difícil de preparar, pero deseable debido a su gran habilidad.
- Esto proporcionó la base para Boosting. Empieza AdaBoost y acaba con XGBoost.
- **Procedimiento**: desarrollar secuencialmente a los clasificadores débiles y agregarlos al conjunto, donde cada clasificador débil es entrenado de manera que preste más atención a las partes en las que los modelos anteriores se equivocaron.

AdaBoost

Definición

- AdaBoost (originalmente AdaBoost.M1) se empleó primeramente para mejorar el rendimiento de CART en clasificación binaria, aunque hoy en día se ha extendido a multiclase y regresión.

Definición

- AdaBoost (originalmente AdaBoost.M1) se empleó primeramente para mejorar el rendimiento de CART en clasificación binaria, aunque hoy en día se ha extendido a multiclase y regresión.
- Se utiliza mejor con **modelos débiles** predominando CART.

Definición

- AdaBoost (originalmente AdaBoost.M1) se empleó primeramente para mejorar el rendimiento de CART en clasificación binaria, aunque hoy en día se ha extendido a multiclase y regresión.
- Se utiliza mejor con **modelos débiles** predominando CART.
- Debido a que estos árboles son tan cortos y solo contienen una decisión de clasificación, a menudo se les llama tocones de decisión *(decision stumps)*.

Definición

- AdaBoost (originalmente AdaBoost.M1) se empleó primeramente para mejorar el rendimiento de CART en clasificación binaria, aunque hoy en día se ha extendido a multiclase y regresión.
- Se utiliza mejor con **modelos débiles** predominando CART.
- Debido a que estos árboles son tan cortos y solo contienen una decisión de clasificación, a menudo se les llama tocones de decisión *(decision stumps)*.
- Cada instancia del conjunto de datos de entrenamiento está ponderada. El peso inicial se establece en:

$$weight(x_i) = \frac{1}{n}$$

Donde x_i es la i-ésima instancia de entrenamiento y n es el número de instancias de entrenamiento.

Entrenamiento

- Se prepara un clasificador débil (tocón de decisión) sobre los datos de entrenamiento utilizando las muestras ponderadas.

Entrenamiento

- Se prepara un clasificador débil (tocón de decisión) sobre los datos de entrenamiento utilizando las muestras ponderadas.
- Solo se admiten problemas de clasificación binaria (de dos clases), por lo que cada tocón de decisión toma una decisión sobre una variable de entrada y **genera un valor de +1 o -1** para el valor de primera o segunda clase.

Entrenamiento

- Se prepara un clasificador débil (tocón de decisión) sobre los datos de entrenamiento utilizando las muestras ponderadas.
- Solo se admiten problemas de clasificación binaria (de dos clases), por lo que cada tocón de decisión toma una decisión sobre una variable de entrada y **genera un valor de +1 o -1** para el valor de primera o segunda clase.
- La **tasa** de clasificación errónea se calcula para el modelo entrenado:

$$error = \frac{N - correct}{N}$$

  - *error* es la tasa de clasificación errónea
  - *correct* es el número de instancias de entrenamiento predichas correctamente por el modelo
  - *N* es el número total de instancias de entrenamiento.

Entrenamiento
Ejemplo

- Si el modelo predice correctamente 78 de 100 instancias de entrenamiento, la tasa de error o clasificación errónea sería *(100-78) / 100 = 0,22*.
- Esto se modifica para utilizar la ponderación de las instancias de entrenamiento:

$$error = \frac{\sum_{i=1}^{n}(w_i \times perror_i)}{\sum_{i=1}^{n} w}$$

  - Es la suma ponderada de la tasa de clasificación errónea, donde
    - w es el peso de la instancia de entrenamiento i
    - *perror* es el error de predicción de la instancia de entrenamiento i, que es 1 si se clasifica incorrectamente y 0 si se clasifica correctamente.
- Ejemplo:
  - Tenemos 3 instancias de entrenamiento con pesos 0,01, 0,5 y 0,2. Los valores predichos fueron -1, -1 y -1, y las variables de salida reales en los casos fueron -1, 1 y -1, entonces los valores de *perror* serían 0, 1 y 0.
  - La tasa de clasificación errónea se calcularía como:

$$error = \frac{0.01 \times 0 + 0.5 \times 1 + 0.2 \times 0}{0.01 + 0.5 + 0.2}$$

$$error = 0.704$$

Entrenamiento

Valor de etapa

- Se calcula un valor de etapa (*stage*) para el modelo entrenado que proporciona una ponderación para cualquier predicción que hace. Se calcula como:

$$stage = ln(\frac{1 - error}{error})$$

  - *ln()* es el logaritmo natural
  - *error* es el error de clasificación errónea del modelo.
- Los modelos más precisos tienen más peso o contribución a la predicción final.
- Se tendrá más peso a las instancias pronosticadas incorrectamente y menos peso a las instancias pronosticadas correctamente.

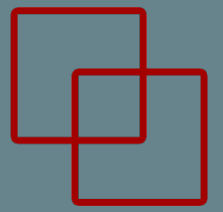
Entrenamiento
Pesos de entrenamiento

- El peso de una instancia de entrenamiento (w) se actualiza como:

$$w = w \times e^{stage \times perror}$$

  - w es el peso para una instancia de entrenamiento específica
  - e es la constante numérica Euler elevado a una potencia
  - *stage* es la tasa de clasificación errónea para el clasificador débil
  - *perror* es el error que cometió el clasificador débil al predecir la variable de salida para la instancia de entrenamiento evaluado como:
    - *perror = 0 IF y == p*
    - *perror = 1 IF y != p*
      - y es la variable de salida para la instancia de entrenamiento
      - p es la predicción del clasificador débil.

- Esto tiene el efecto de **no cambiar el peso si la instancia de entrenamiento se clasificó correctamente y hacer que el peso sea ligeramente mayor** si el clasificador débil clasificó erróneamente la instancia.

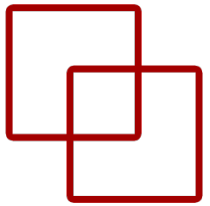


UNED

ETS de Ingeniería Informática

AdaBoost Ensemble

Resumen



- Los modelos débiles se agregan **secuencialmente** y se entrenan utilizando los datos de entrenamiento **ponderados**.

Resumen

- Los modelos débiles se agregan **secuencialmente** y se entrenan utilizando los datos de entrenamiento **ponderados**.

- El proceso continúa hasta que se crea un número preestablecido de clasificadores débiles (un **parámetro** de usuario) o no se pueden realizar más mejoras en el conjunto de datos de entrenamiento (**convergencia**).

Resumen

- Los modelos débiles se agregan **secuencialmente** y se entrenan utilizando los datos de entrenamiento **ponderados**.

- El proceso continúa hasta que se crea un número preestablecido de clasificadores débiles (un **parámetro** de usuario) o no se pueden realizar más mejoras en el conjunto de datos de entrenamiento (**convergencia**).

- Una vez completado, se queda con un grupo de clasificadores débiles, cada uno con un **valor de etapa**.

Predicciones

- Las predicciones se realizan calculando el **promedio ponderado** de los clasificadores débiles.

Predicciones

- Las predicciones se realizan calculando el **promedio ponderado** de los clasificadores débiles.
- Para una nueva instancia de entrada, cada clasificador débil calcula un valor predicho como **+1 o -1**.

Predicciones

- Las predicciones se realizan calculando el **promedio ponderado** de los clasificadores débiles.
- Para una nueva instancia de entrada, cada clasificador débil calcula un valor predicho como **+1 o -1**.
- Los valores predichos están ponderados por el **valor de cada etapa** de los clasificadores débiles.

Predicciones

- Las predicciones se realizan calculando el **promedio ponderado** de los clasificadores débiles.
- Para una nueva instancia de entrada, cada clasificador débil calcula un valor predicho como **+1 o -1**.
- Los valores predichos están ponderados por el **valor de cada etapa** de los clasificadores débiles.
- La predicción del modelo se toma como la **suma de las predicciones ponderadas**.

Predicciones

- Las predicciones se realizan calculando el **promedio ponderado** de los clasificadores débiles.
- Para una nueva instancia de entrada, cada clasificador débil calcula un valor predicho como **+1 o -1**.
- Los valores predichos están ponderados por el **valor de cada etapa** de los clasificadores débiles.
- La predicción del modelo se toma como la **suma de las predicciones ponderadas**.
- Si la suma es **positiva**, entonces se predice la **primera clase**; si es **negativa**, la **segunda clase**.

Predicciones

- Las predicciones se realizan calculando el **promedio ponderado** de los clasificadores débiles.
- Para una nueva instancia de entrada, cada clasificador débil calcula un valor predicho como **+1 o -1**.
- Los valores predichos están ponderados por el **valor de cada etapa** de los clasificadores débiles.
- La predicción del modelo se toma como la **suma de las predicciones ponderadas**.
- Si la suma es **positiva**, entonces se predice la **primera clase**; si es **negativa**, la **segunda clase**.
- Ejemplo,
  - 5 clasificadores débiles pueden predecir los valores 1, 1, -1, 1, -1.
  - A partir de una votación mayoritaria, parece que el modelo predice un valor de 1 o de primera clase.
  - Estos mismos 5 clasificadores débiles pueden tener los valores de etapa 0.2, 0.5, 0.8, 0.2 y 0.9, respectivamente.
  - Calcular la suma ponderada de estas predicciones da como resultado un resultado de -0.8, lo que sería una predicción conjunta de -1 o la segunda clase.

Predicciones

Heurísticas para mejorarlas

- **Datos de calidad**: debido a que el método de conjunto continúa intentando corregir errores de clasificación en los datos de entrenamiento, debe tener cuidado de que los datos de entrenamiento sean de alta calidad.

Predicciones
Heurísticas para mejorarlas

- **Datos de calidad**: debido a que el método de conjunto continúa intentando corregir errores de clasificación en los datos de entrenamiento, debe tener cuidado de que los datos de entrenamiento sean de alta calidad.

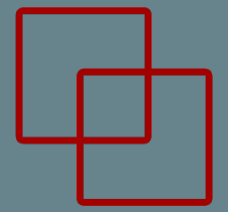
- **Valores atípicos**: Los valores atípicos obligarán al conjunto a trabajar duro para corregir casos que no son realistas.
  - Estos podrían eliminarse del conjunto de datos de entrenamiento.

Predicciones
Heurísticas para mejorarlas

- **Datos de calidad**: debido a que el método de conjunto continúa intentando corregir errores de clasificación en los datos de entrenamiento, debe tener cuidado de que los datos de entrenamiento sean de alta calidad.

- **Valores atípicos**: Los valores atípicos obligarán al conjunto a trabajar duro para corregir casos que no son realistas.
  - Estos podrían eliminarse del conjunto de datos de entrenamiento.

- **Datos ruidosos o corruptos**: Los datos ruidosos, específicamente el ruido en la variable de salida, pueden ser problemáticos.
  - Si posible, intente aislarlos y limpiarlos de su conjunto de datos de entrenamiento.



UNED

ETS de Ingeniería Informática

Tutorial AdaBoost

Objetivos

- Crear **tocón de decisión** a partir de datos de entrenamiento utilizando datos de entrenamiento **ponderados**.

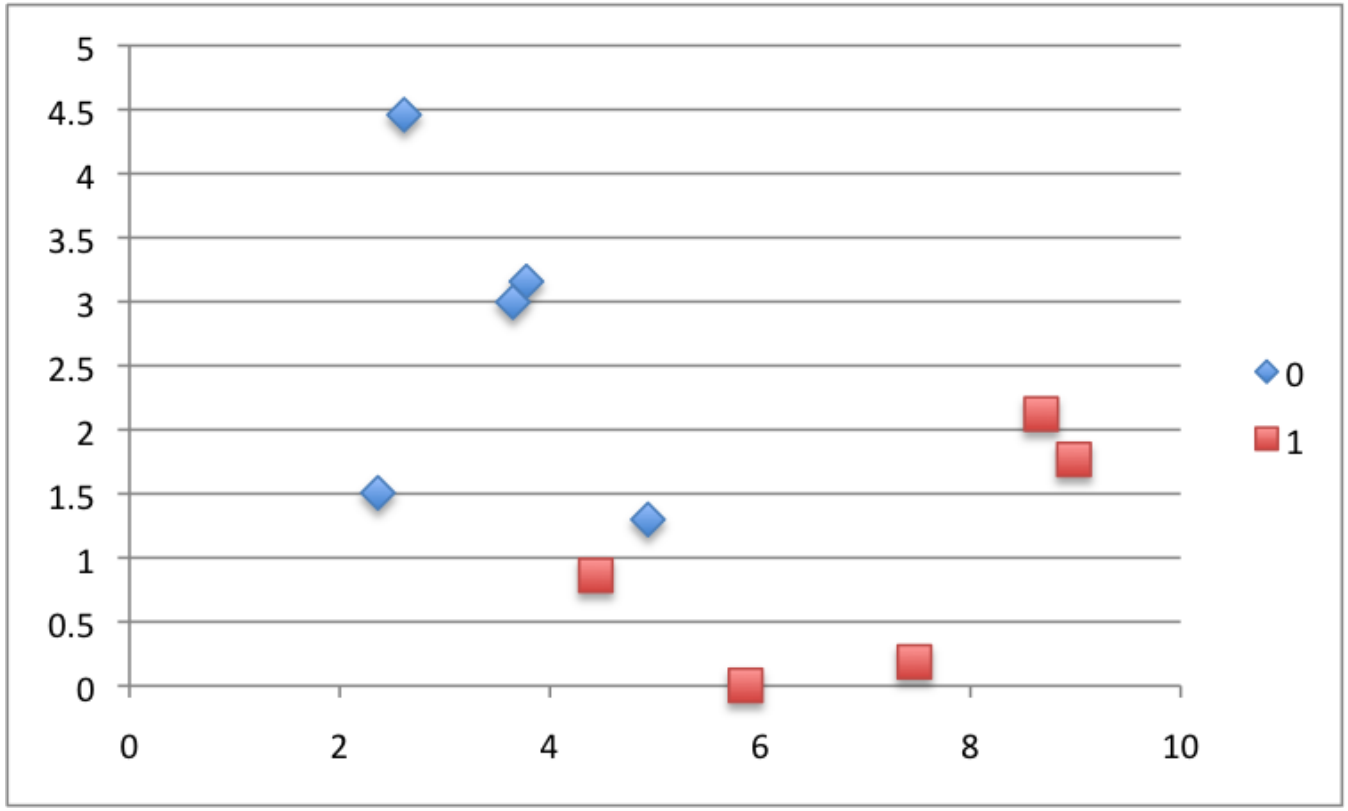
- Actualizar los **pesos** en los datos de entrenamiento para prestar más atención a las instancias difíciles de clasificar.

- Crear una **secuencia** de 3 modelos uno tras otro.

- Utilizar un modelo AdaBoost con **3 modelos débiles** para hacer predicciones.

Dataset

X1	X2	Y
3.64754035	2.996793259	0
2.612663842	4.459457779	0
2.363359679	1.506982189	0
4.932600453	1.299008795	0
3.776154753	3.157451378	0
8.673960793	2.122873405	1
5.861599451	0.003512817	1
8.984677361	1.768161009	1
7.467380954	0.187045945	1
4.436284412	0.862698005	1



Modelo 1

Modelo #1
División

- Modelo 1: tocón decisión para la variable de entrada *X1*
  - IF *X1* ≤ 4.932600453 THEN LEFT
  - IF *X1* > 4.932600453 THEN RIGHT
- Este es un punto de división deficiente para estos datos y se eligió intencionalmente para crear algunas instancias mal clasificadas.
  - Lo ideal sería el algoritmo CART para elegir un punto de división con el índice de Gini (u otros) como función de costo.
- Si aplicamos este punto de división a los datos de entrenamiento
  - Podemos ver los datos divididos en los 2 grupos.
    - Grupo *RIGHT* es predominantemente en la clase 1
    - Grupo *LEFT* es predominantemente clase 0.

Model 1: Predictions

X1	X2	Y	Weight	X1 Split	Group	Prediction
3,64754035	2,996793259	0	0,1	4,9326	LEFT	0
2,61266384	4,459457779	0	0,1		LEFT	0
2,36335968	1,506982189	0	0,1		LEFT	0
4,93260045	1,299008795	0	0,1		LEFT	0
3,77615475	3,157451378	0	0,1		LEFT	0
8,67396079	2,122873405	1	0,1		RIGHT	1
5,86159945	0,003512817	1	0,1		RIGHT	1
8,98467736	1,768161009	1	0,1		RIGHT	1
7,46738095	0,187045945	1	0,1		RIGHT	1
4,43628441	0,862698005	1	0,1		LEFT	0

Modelo #1
División

- Modelo 1: tocón decisión para
  - IF *X1* ≤ 4.932600453 THE
  - IF *X1* > 4.932600453 THE
- Este es un punto de división d
  eligió intencionalmente para c
  clasificadas.
  - Lo ideal sería el algoritmo
    división con el índice de G

Model 1: Predictions

X1	X2	Y	Weight	X1 Split	Group	Prediction
3,64754035	2,996793259	0	0,1	4,9326	LEFT	0
2,61266384	4,459457779	0	0,1		LEFT	0
2,36335968	1,506982189	0	0,1		LEFT	0
4,93260045	1,299008795	0	0,1		LEFT	0
3,77615475	3,157451378	0	0,1		LEFT	0
8,67396079	2,122873405	1	0,1		RIGHT	1
5,86159945	0,003512817	1	0,1		RIGHT	1
8,98467736	1,768161009	1	0,1		RIGHT	1
7,46738095	0,187045945	1	0,1		RIGHT	1
4,43628441	0,862698005	1	0,1		LEFT	0

- Si aplicamos este punto de división a los datos de entrenamiento
  - Podemos ver los datos divididos en los 2 grupos.
    - Grupo *RIGHT* es predominantemente en la clase 1
    - Grupo *LEFT* es predominantemente clase 0.

Modelo #1
Predicción

- Ahora que el modelo de decisión esta entrenado, podemos usarlo para hacer predicciones.
- El error en la predicción se puede calcular usando:
  - *error = 0 IF Prediction == Y*
  - *error = 1 IF Prediction != Y*
- Las predicciones y sus errores:
  - 1 error en 10 predicciones → 0.9 o 90%

Y	Group	Prediction	Error	W
0	LEFT	0	0	
0	LEFT	0	0	
0	LEFT	0	0	
0	LEFT	0	0	
0	LEFT	0	0	
1	RIGHT	1	0	
1	RIGHT	1	0	
1	RIGHT	1	0	
1	RIGHT	1	0	
1	LEFT	0	1	

Modelo #1
Predicción

- Ahora que el modelo de decisión esta entrenado, podemos usarlo para hacer predicciones.
- El error en la predicción se puede calcular usando:
  - *error = 0 IF Prediction == Y*
  - *error = 1 IF Prediction != Y*
- Las predicciones y sus errores:
  - 1 error en 10 predicciones → 0.9 o 90%

Group	Prediction	Error	Weighted Error
LEFT	0	0	0
LEFT	0	0	0
LEFT	0	0	0
LEFT	0	0	0
LEFT	0	0	0
RIGHT	1	0	0
RIGHT	1	0	0
RIGHT	1	0	0
RIGHT	1	0	0
LEFT	0	1	0,1

Modelo #1
Predicción – Error ponderado

- Calculamos la tasa de clasificación errónea para un modelo débil de tocón de decisión:

$$MisclassificationRate = \frac{\sum_{i=1}^{n}(w_i \times error_i)}{\sum_{i=1}^{n} w}$$

- Cada instancia tiene el peso inicial de *weight* = *1/N*, donde *N*, es el N.º de instancias, sería:
  - *weight = 1/10 = 0.1.*
- Usando *weight* y los errores de predicción, calculamos el error ponderado de cada predicción:

$$WeightedError = weight \times error$$

Weight	Error	Weighted Error
0.1	0	0
0.1	0	0
0.1	0	0
0.1	0	0
0.1	0	0
0.1	0	0
0.1	0	0
0.1	0	0
0.1	0	0
0.1	1	0.1

Modelo #1
Predicción – Tasa de clasificación errónea

- Ahora podemos calcular la **tasa de clasificación** errónea como:

$$MisclassificationRate = \frac{\sum(WeightedError)}{\sum(weight)}$$
$$MisclassificationRate = \frac{0.1}{1.0}$$
$$MisclassificationRate = 0.1$$

- Finalmente, utilizamos la tasa de clasificación errónea para calcular la **etapa** de este modelo débil.
  - La etapa es el peso aplicado a cualquier predicción hecha por este modelo en producción.

$$stage = ln(\frac{1 - MisclassificationRate}{MisclassificationRate})$$
$$stage = ln(\frac{1 - 0.1}{0.1})$$
$$stage = 2.197224577$$

Modelo #1
Predicción – Tasa de clasificación errónea

- Ahora podemos calcular la tasa de clasificación errónea como:

$$MisclassificationRate = \frac{\sum(WeightedError)}{\sum(weight)}$$

$$MisclassificationRate = \frac{0.1}{1.0}$$

$$MisclassificationRate = 0.1$$

Modelo que tendrá mucho peso en las prediciones → 90% accuracy

- Finalmente, utilizamos la tasa de clasificación errónea para calcular la etapa de este modelo débil.
  - La etapa es el peso aplicado a cualquier predicción hecha por este modelo en producción.

$$stage = ln(\frac{1 - MisclassificationRate}{MisclassificationRate})$$

$$stage = ln(\frac{1 - 0.1}{0.1})$$

$$stage = 2.197224577$$

Modelo #1
Resumen numérico

Model 1: Predictions

X1	X2	Y	Weight	X1 Split	Group	Prediction	Error	Weighted Error	Sum Weight	Sum Error	Misclassification	Stage	Accuracy
3,64754035	2,996793259	0	0,1	4,9326	LEFT	0	0	0	1	0,1	0,1	2,19722458	90
2,61266384	4,459457779	0	0,1		LEFT	0	0	0					
2,36335968	1,506982189	0	0,1		LEFT	0	0	0					
4,93260045	1,299008795	0	0,1		LEFT	0	0	0					
3,77615475	3,157451378	0	0,1		LEFT	0	0	0					
8,67396079	2,122873405	1	0,1		RIGHT	1	0	0					
5,86159945	0,003512817	1	0,1		RIGHT	1	0	0					
8,98467736	1,768161009	1	0,1		RIGHT	1	0	0					
7,46738095	0,187045945	1	0,1		RIGHT	1	0	0					
4,43628441	0,862698005	1	0,1		LEFT	0	1	0,1					

Modelo #1
Actualizar pesos

- Así, presta más atención a las instancias mal clasificadas
- El peso de una instancia de entrenamiento se actualiza usando:

$$w = w \times e^{stage \times perror}$$

- Conocemos el peso actual de cada instancia (*0.1*) y los errores cometidos en cada instancia de entrenamiento.

Weight
0.1
0.1
0.1
0.1
0.1
0.1
0.1
0.1
0.1
0.124573094

Modelo #1
Actualizar pesos

- Así, presta más atención a las instancias mal clasificadas
- El peso de una instancia de entrenamiento se actualiza usando:

$$w = w \times e^{stage \times perror}$$

- Conocemos el peso actual de cada instancia (*0.1*) y los errores cometidos en cada instancia de entrenamiento.
- A continuación, se muestran los pesos actualizados para cada instancia de entrenamiento.
  - Es **más grande** para que el próximo modelo que se cree tome **más atención**.

Weight
0.1
0.1
0.1
0.1
0.1
0.1
0.1
0.1
0.1
0.124573094

Modelo 2

Modelo #2
División

- Será un tocón de decisión, pero esta vez dividirá la variable *X2:*
  - *IF X2 ≤ 2.122873405 THEN LEFT*
  - *IF X2 > 2.122873405 THEN RIGHT*

- Datos divididos en los 2 grupos.
  - Grupo *LEFT* es predominantemente en la clase 1
  - Grupo *RIGHT* es predominantemente clase 0.

Y	Group
0	RIGHT
0	RIGHT
0	LEFT
0	LEFT
0	RIGHT
1	LEFT
1	LEFT
1	LEFT
1	LEFT
1	LEFT

Modelo #2
Predicción

- Calculamos las predicciones, el error y el error ponderado de esas predicciones.

- Este modelo comete **2 errores** y tendrá un Accuracy del 80%.
  - Ligeramente peor que el primer modelo.
  - Cometió diferentes errores
  - Predijo correctamente los casos que el primer modelo predijo incorrectamente.

Y	Prediction	Error	Weighted Error
0	0	0	0
0	0	0	0
0	1	1	0.1
0	1	1	0.1
0	0	0	0
1	1	0	0
1	1	0	0
1	1	0	0
1	1	0	0
1	1	0	0

Modelo #2
Predicción – Tasa de clasificación errónea

- Calculamos el valor de etapa errónea como:

$$\sum weight = 1.024573094$$
$$\sum WeightedError = 0.2$$
$$MisclassificationRate = 0.1952032521$$
$$stage = 1.416548424$$

Sum Weight	Sum Error	Misclassification	Stage	Accuracy
1,024573094	0,2	0,1952032521435	1,41654842	80

- Si nos detenemos ahí, no tendremos un modelo AdaBoost muy preciso.
- Una verificación rápida sugiere que el modelo tendrá un 60% en train.
- Puedes comprobarlo tú mismo más tarde cambiando la siguiente sección por la combinación de modelo 1 y 2.
- Necesitamos un modelo más para hacer algunas correcciones finales.

Modelo #2
Resumen numérico

Model 2: Predictions

X1	X2	Y	Weight	X2 Split	Group	Prediction	Error	Weighted Error	Sum Weight	Sum Error	Misclassification	Stage	Accuracy
3,64754035	2,996793259	0	0,1	2,122873	RIGHT	0	0	0	1,024573094	0,2	0,1952032521435	1,41654842	80
2,61266384	4,459457779	0	0,1		RIGHT	0	0	0					
2,36335968	1,506982189	0	0,1		LEFT	1	1	0,1					
4,93260045	1,299008795	0	0,1		LEFT	1	1	0,1					
3,77615475	3,157451378	0	0,1		RIGHT	0	0	0					
8,67396079	2,122873405	1	0,1		LEFT	1	0	0					
5,86159945	0,003512817	1	0,1		LEFT	1	0	0					
8,98467736	1,768161009	1	0,1		LEFT	1	0	0					
7,46738095	0,187045945	1	0,1		LEFT	1	0	0					
4,43628441	0,862698005	1	0,124573		LEFT	1	0	0					

Modelo 3

Modelo #3

Actualización de pesos

$$w = w \times e^{stage \times perror}$$

Weight
0.1
0.1
0.11521789
0.11521789
0.1
0.1
0.1
0.1
0.1
0.124573094

Modelo #3
División

- Este modelo elegirá un punto de división para X2:
  - *IF X2 ≤ 0.862698005 THEN LEFT*
  - *IF X2 > 0.862698005 THEN RIGHT*

- Podemos ver los datos divididos en los 2 grupos.
  - Grupo *LEFT* es predominantemente en la clase 1
  - Grupo *RIGHT* es predominantemente clase 0.

Y	Group
0	RIGHT
0	RIGHT
0	RIGHT
0	RIGHT
0	RIGHT
1	RIGHT
1	LEFT
1	RIGHT
1	LEFT
1	LEFT

Modelo #3
Predicción

- Ahora podemos calcular las predicciones, el error y el error ponderado de esas predicciones.

- Este modelo comete 2 errores y tendrá un accuracy del 80%.

- Calculamos el valor de etapa para este clasificador.

Y	Prediction	Error	Weighted Error
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
1	0	1	0.1
1	1	0	0
1	0	1	0.1
1	1	0	0
1	1	0	0

$$\sum(weight) = 1.055008873$$

$$\sum WeightedError = 0.2$$

$$MisclassificationRate = 0.189571865$$

$$stage = 1.452794480$$

Modelo #3
Resumen numérico

Model 3: Predictions

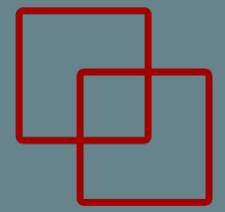
X1	X2	Y	Weight	X2 Split	Group	Prediction	Error	Weighted Error	Sum Weight	Sum Error	Misclassification	Stage	Accuracy
3,64754035	2,996793259	0	0,1	0,862698	RIGHT	0	0	0	1,055008873	0,2	0,189571865285	1,45279448	80
2,61266384	4,459457779	0	0,1		RIGHT	0	0	0					
2,36335968	1,506982189	0	0,115218		RIGHT	0	0	0					
4,93260045	1,299008795	0	0,115218		RIGHT	0	0	0					
3,77615475	3,157451378	0	0,1		RIGHT	0	0	0					
8,67396079	2,122873405	1	0,1		RIGHT	0	1	0,1					
5,86159945	0,003512817	1	0,1		LEFT	1	0	0					
8,98467736	1,768161009	1	0,1		RIGHT	0	1	0,1					
7,46738095	0,187045945	1	0,1		LEFT	1	0	0					
4,43628441	0,862698005	1	0,124573		LEFT	1	0	0					

Modelo #3
Resumen numérico

Hemos acabado con los 3 modelos.
Ahora calculamos las prediciones finales.

Model 3: Predictions

X1	X2	Y	Weight	X2 Split	Group	Prediction	Error	Weighted Error	Sum Weight	Sum Error	Misclassification	Stage	Accuracy
3,64754035	2,996793259	0	0,1	0,862698	RIGHT	0	0	0	1,055008873	0,2	0,189571865285	1,45279448	80
2,61266384	4,459457779	0	0,1		RIGHT	0	0	0					
2,36335968	1,506982189	0	0,115218		RIGHT	0	0	0					
4,93260045	1,299008795	0	0,115218		RIGHT	0	0	0					
3,77615475	3,157451378	0	0,1		RIGHT	0	0	0					
8,67396079	2,122873405	1	0,1		RIGHT	0	1	0,1					
5,86159945	0,003512817	1	0,1		LEFT	1	0	0					
8,98467736	1,768161009	1	0,1		RIGHT	0	1	0,1					
7,46738095	0,187045945	1	0,1		LEFT	1	0	0					
4,43628441	0,862698005	1	0,124573		LEFT	1	0	0					



UNED

ETS de Ingeniería Informática

Boosted predictions

Modelo boosted

Modelos contribuyentes

Model 1: Predictions

X1	X2	Y	Weight	X1 Split	Group	Prediction	Error	Weighted Error	Sum Weight	Sum Error	Misclassification	Stage	Accuracy
3,64754035	2,996793259	0	0,1	4,9326	LEFT	0	0	0	1	0,1	0,1	2,19722458	90
2,61266384	4,459457779	0	0,1		LEFT	0	0	0					
2,36335968	1,506982189	0	0,1		LEFT	0	0	0					
4,93260045	1,299008795	0	0,1		LEFT	0	0	0					
3,77615475	3,157451378	0	0,1		LEFT	0	0	0					
8,67396079	2,122873405	1	0,1		RIGHT	1	0	0					
5,86159945	0,003512817	1	0,1		RIGHT	1	0	0					
8,98467736	1,768161009	1	0,1		RIGHT	1	0	0					
7,46738095	0,187045945	1	0,1		RIGHT	1	0	0					
4,43628441	0,862698005	1	0,1		LEFT	0	1	0,1					

Model 2: Predictions

X1	X2	Y	Weight	X2 Split	Group	Prediction	Error	Weighted Error	Sum Weight	Sum Error	Misclassification	Stage	Accuracy
3,64754035	2,996793259	0	0,1	2,122873	RIGHT	0	0	0	1,024573094	0,2	0,1952032521435	1,41654842	80
2,61266384	4,459457779	0	0,1		RIGHT	0	0	0					
2,36335968	1,506982189	0	0,1		LEFT	1	1	0,1					
4,93260045	1,299008795	0	0,1		LEFT	1	1	0,1					
3,77615475	3,157451378	0	0,1		RIGHT	0	0	0					
8,67396079	2,122873405	1	0,1		LEFT	1	0	0					
5,86159945	0,003512817	1	0,1		LEFT	1	0	0					
8,98467736	1,768161009	1	0,1		LEFT	1	0	0					
7,46738095	0,187045945	1	0,1		LEFT	1	0	0					
4,43628441	0,862698005	1	0,124573		LEFT	1	0	0					

Model 3: Predictions

X1	X2	Y	Weight	X2 Split	Group	Prediction	Error	Weighted Error	Sum Weight	Sum Error	Misclassification	Stage	Accuracy
3,64754035	2,996793259	0	0,1	0,862698	RIGHT	0	0	0	1,055008873	0,2	0,189571865285	1,45279448	80
2,61266384	4,459457779	0	0,1		RIGHT	0	0	0					
2,36335968	1,506982189	0	0,115218		RIGHT	0	0	0					
4,93260045	1,299008795	0	0,115218		RIGHT	0	0	0					
3,77615475	3,157451378	0	0,1		RIGHT	0	0	0					
8,67396079	2,122873405	1	0,1		RIGHT	0	1	0,1					
5,86159945	0,003512817	1	0,1		LEFT	1	0	0					
8,98467736	1,768161009	1	0,1		RIGHT	0	1	0,1					
7,46738095	0,187045945	1	0,1		LEFT	1	0	0					
4,43628441	0,862698005	1	0,124573		LEFT	1	0	0					

Modelo boosted
Predicciones finales

- Predicciones para un solo clasificador son +1 o -1 y están ponderados por el valor de etapa del modelo. Para este problema usaremos:

$$prediction = stage \times (\text{IF } (output == 0) \text{ THEN } -1 \text{ ELSE } +1)$$

- Podemos hacer predicciones ponderadas dados los valores de entrada a partir de los datos de entrenamiento.
- Podrían tratarse fácilmente de datos no etiquetados sobre los que nos gustaría hacer predicciones.

Modelo boosted
Predicciones finales

- Predicciones para un solo clasificador son +1 o -1 y están ponderados por el valor de etapa del modelo. Para este problema usaremos:

$$prediction = stage \times (\text{IF } (output == 0) \text{ THEN } -1 \text{ ELSE } +1)$$

- Podemos hacer predicciones ponderadas dados los valores de entrada a partir de los datos de entrenamiento.
- Podrían tratarse fácilmente de datos no etiquetados sobre los que nos gustaría hacer predicciones.

Model 1: Predictions

X1	X2	Y	Weight	X1 Split	Group	Prediction	Error	Weighted Error	Sum Weight	Sum Error	Misclassification	Stage	Accuracy
3,64754035	2,996793259	0	0,1	4,9326	LEFT	0	0	0	1	0,1	0,1	2,19722458	90
2,61266384	4,459457779	0	0,1		LEFT	0	0	0					
2,36335968	1,506982189	0	0,1		LEFT	0	0	0					
4,93260045	1,299008795	0	0,1		LEFT	0	0	0					
3,77615475	3,157451378	0	0,1		LEFT	0	0	0					
8,67396079	2,122873405	1	0,1		RIGHT	1	0	0					
5,86159945	0,003512817	1	0,1		RIGHT	1	0	0					
8,98467736	1,768161009	1	0,1		RIGHT	1	0	0					
7,46738095	0,187045945	1	0,1		RIGHT	1	0	0					
4,43628441	0,862698005	1	0,1		LEFT	0	1	0,1					

Modelo boosted
Predicciones finales

- Predicciones para un solo clasificador son +1 o -1 y están ponderados por el valor de etapa del modelo. Para este problema usaremos:

$$prediction = stage \times (\text{IF } (output == 0) \text{ THEN } -1 \text{ ELSE } +1)$$

- Podemos hacer predicciones ponderadas dados los valores de entrada a partir de los datos de entrenamiento.

- Podrían tratarse fácilmente de datos no etiquetados sobre los que nos gustaría hacer predicciones.

Boosted Predictions

X1	X2	Model 1	Model 2	Model 3
3,64754035	2,996793259	-2,19722458	-1,4165484	-1,4527945
2,61266384	4,459457779	-2,19722458	-1,4165484	-1,4527945
2,36335968	1,506982189	-2,19722458	1,41654842	-1,4527945
4,93260045	1,299008795	-2,19722458	1,41654842	-1,4527945
3,77615475	3,157451378	-2,19722458	-1,4165484	-1,4527945
8,67396079	2,122873405	2,197224577	1,41654842	-1,4527945
5,86159945	0,003512817	2,197224577	1,41654842	1,45279448
8,98467736	1,768161009	2,197224577	1,41654842	-1,4527945
7,46738095	0,187045945	2,197224577	1,41654842	1,45279448
4,43628441	0,862698005	-2,19722458	1,41654842	1,45279448

Modelo boosted
Predicciones finales

- Predicciones para un solo clasificador son +1 o -1 y están ponderados por el valor de etapa del modelo. Para este problema usaremos:

$$prediction = stage \times (\text{IF } (output == 0) \text{ THEN } -1 \text{ ELSE } +1)$$

- Usando los tres modelos anteriores, podemos hacer predicciones ponderadas dados los valores de entrada a partir de los datos de entrenamiento.

- Podrían tratarse fácilmente de datos no etiquetados sobre los que nos gustaría hacer predicciones.

Boosted Predictions

X1	X2	Model 1	Model 2	Model 3
3,64754035	2,996793259	-2,19722458	-1,4165484	-1,4527945
2,61266384	4,459457779	-2,19722458	-1,4165484	-1,4527945
2,36335968	1,506982189	-2,19722458	1,41654842	-1,4527945
4,93260045	1,299008795	-2,19722458	1,41654842	-1,4527945
3,77615475	3,157451378	-2,19722458	-1,4165484	-1,4527945
8,67396079	2,122873405	2,197224577	1,41654842	-1,4527945
5,86159945	0,003512817	2,197224577	1,41654842	1,45279448
8,98467736	1,768161009	2,197224577	1,41654842	-1,4527945
7,46738095	0,187045945	2,197224577	1,41654842	1,45279448
4,43628441	0,862698005	-2,19722458	1,41654842	1,45279448

Modelo boosted
Predicciones finales

- Sumamos las predicciones de cada modelo para dar un resultado final.
  - Si un resultado es menor que 0, entonces se predice la clase 0
  - Si un resultado es mayor que 0, se predice la clase 1.
- Ahora podemos calcular las predicciones finales del modelo AdaBoost.

Sum	Prediction	Y	Error	Accuracy
-5,0665675	0	0	0	100
-5,0665675	0	0	0	
-2,2334706	0	0	0	
-2,2334706	0	0	0	
-5,0665675	0	0	0	
2,16097852	1	1	0	
5,06656748	1	1	0	
2,16097852	1	1	0	
5,06656748	1	1	0	
0,67211833	1	1	0	

Modelo boosted

Predicciones finales

- Sumamos las predicciones de cada
  - Si un resultado es menor que 0
  - Si un resultado es mayor que 0, se predic .
- Ahora podemos calcular las predicciones finales d odelo AdaBoost.

Predicciones coinciden con el valor esperado
Obtenemos un accuracy del 100%

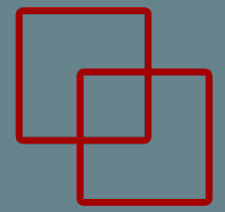
Sum	Prediction	Y	Error	Accuracy
-5,0665675	0	0	0	100
-5,0665675	0	0	0	
-2,2334706	0	0	0	
-2,2334706	0	0	0	
-5,0665675	0	0	0	
2,16097852	1	1	0	
5,06656748	1	1	0	
2,16097852	1	1	0	
5,06656748	1	1	0	
0,67211833	1	1	0	

Modelo boosted

Resumen numérico

Boosted Predictions

X1	X2	Model 1	Model 2	Model 3	Sum	Prediction	Y	Error	Accuracy
3,64754035	2,996793259	-2,19722458	-1,4165484	-1,4527945	-5,0665675	0	0	0	100
2,61266384	4,459457779	-2,19722458	-1,4165484	-1,4527945	-5,0665675	0	0	0	
2,36335968	1,506982189	-2,19722458	1,41654842	-1,4527945	-2,2334706	0	0	0	
4,93260045	1,299008795	-2,19722458	1,41654842	-1,4527945	-2,2334706	0	0	0	
3,77615475	3,157451378	-2,19722458	-1,4165484	-1,4527945	-5,0665675	0	0	0	
8,67396079	2,122873405	2,197224577	1,41654842	-1,4527945	2,16097852	1	1	0	
5,86159945	0,003512817	2,197224577	1,41654842	1,45279448	5,06656748	1	1	0	
8,98467736	1,768161009	2,197224577	1,41654842	-1,4527945	2,16097852	1	1	0	
7,46738095	0,187045945	2,197224577	1,41654842	1,45279448	5,06656748	1	1	0	
4,43628441	0,862698005	-2,19722458	1,41654842	1,45279448	0,67211833	1	1	0	

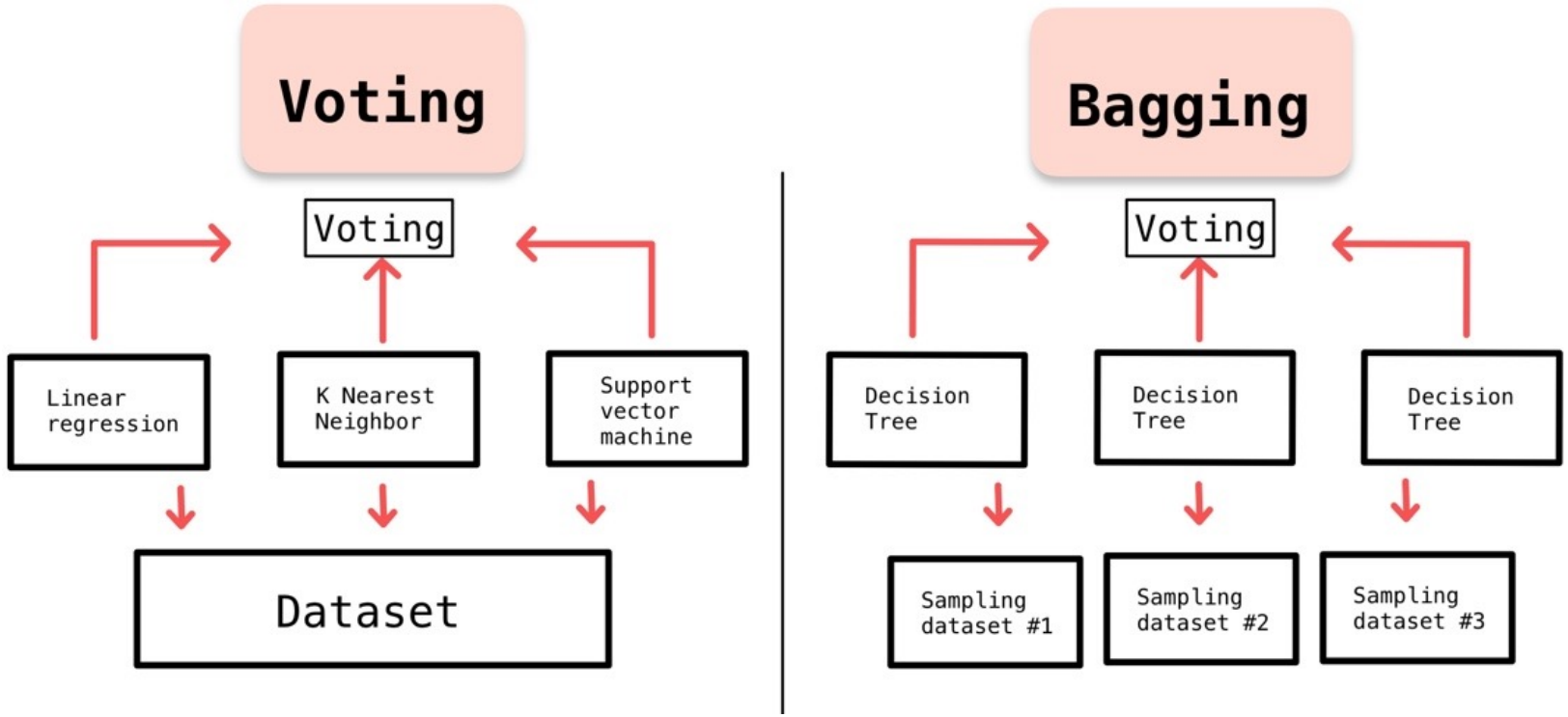


UNED

ETS de Ingeniería Informática

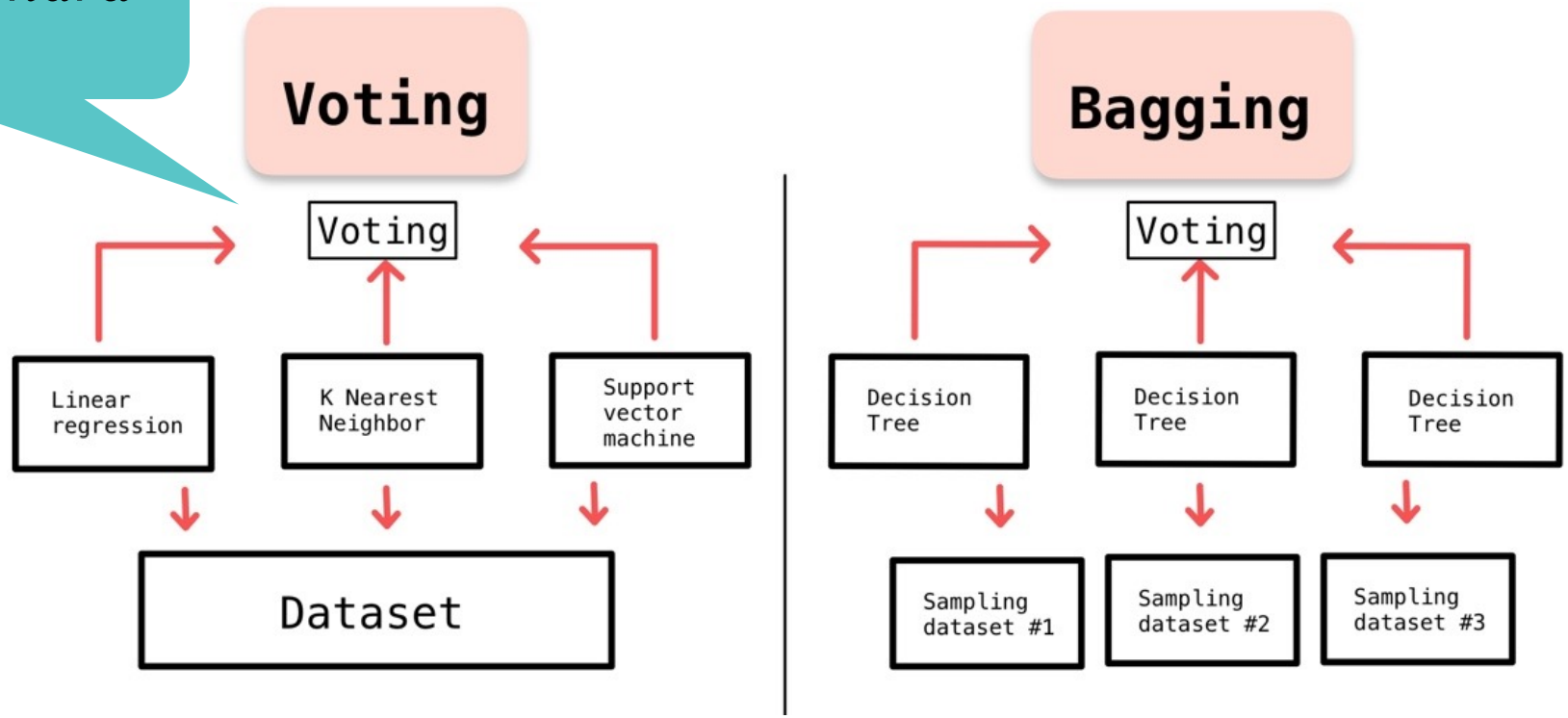
Stacking

Voting

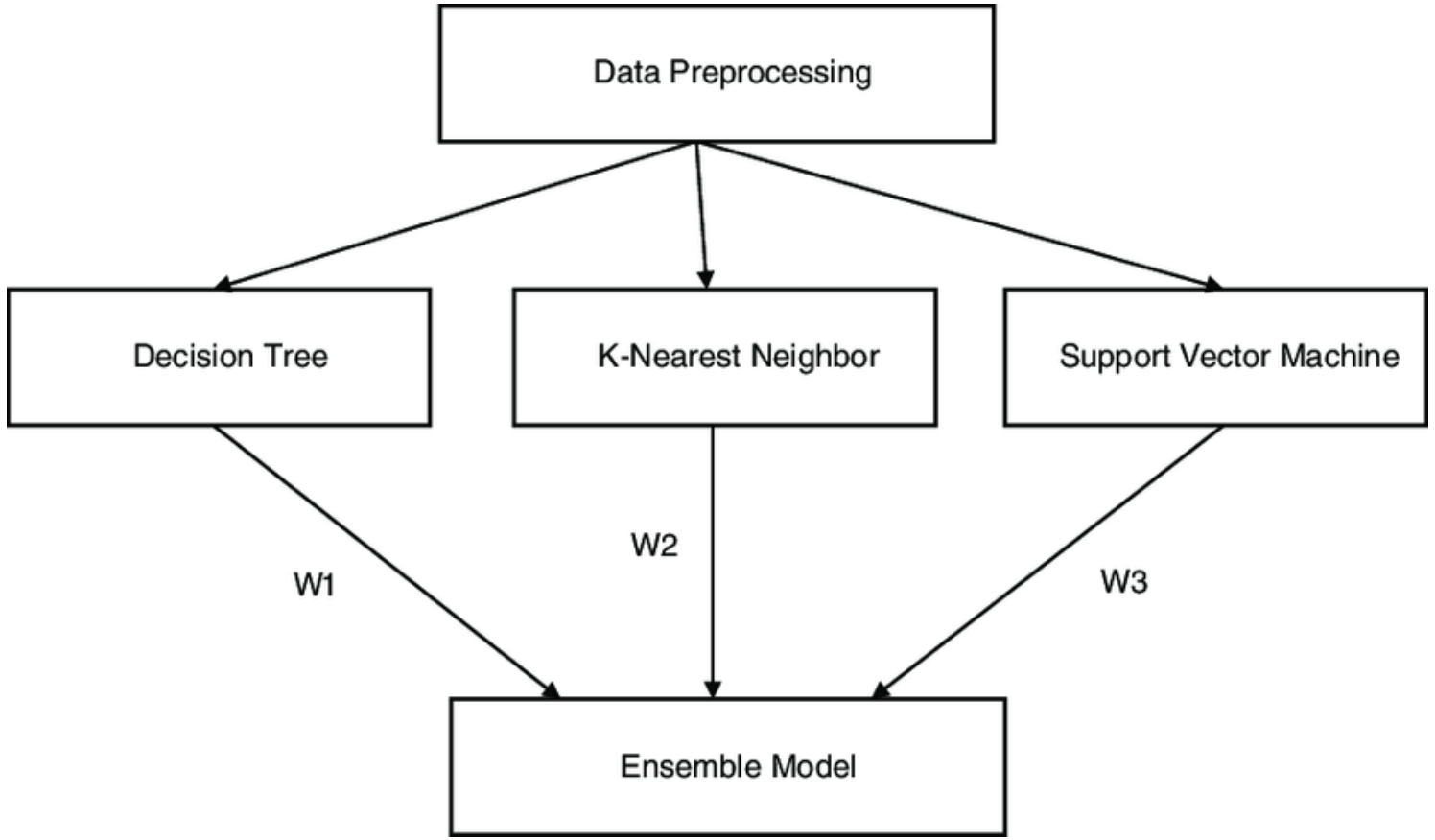

Voting
Voting
Linear regression
K Nearest Neighbor
Support vector machine
Dataset
Bagging
Voting
Decision Tree
Decision Tree
Decision Tree
Sampling dataset #1
Sampling dataset #2
Sampling dataset #3

Voting

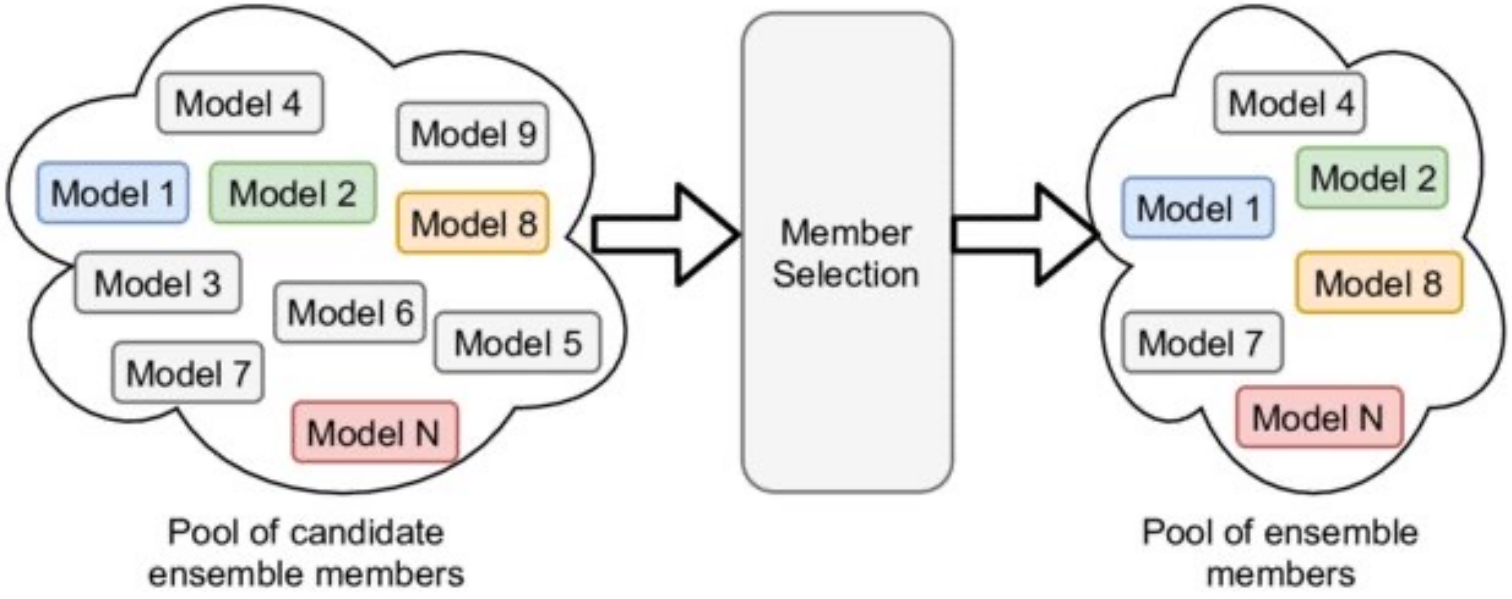
Puede ser *soft* o *hard*



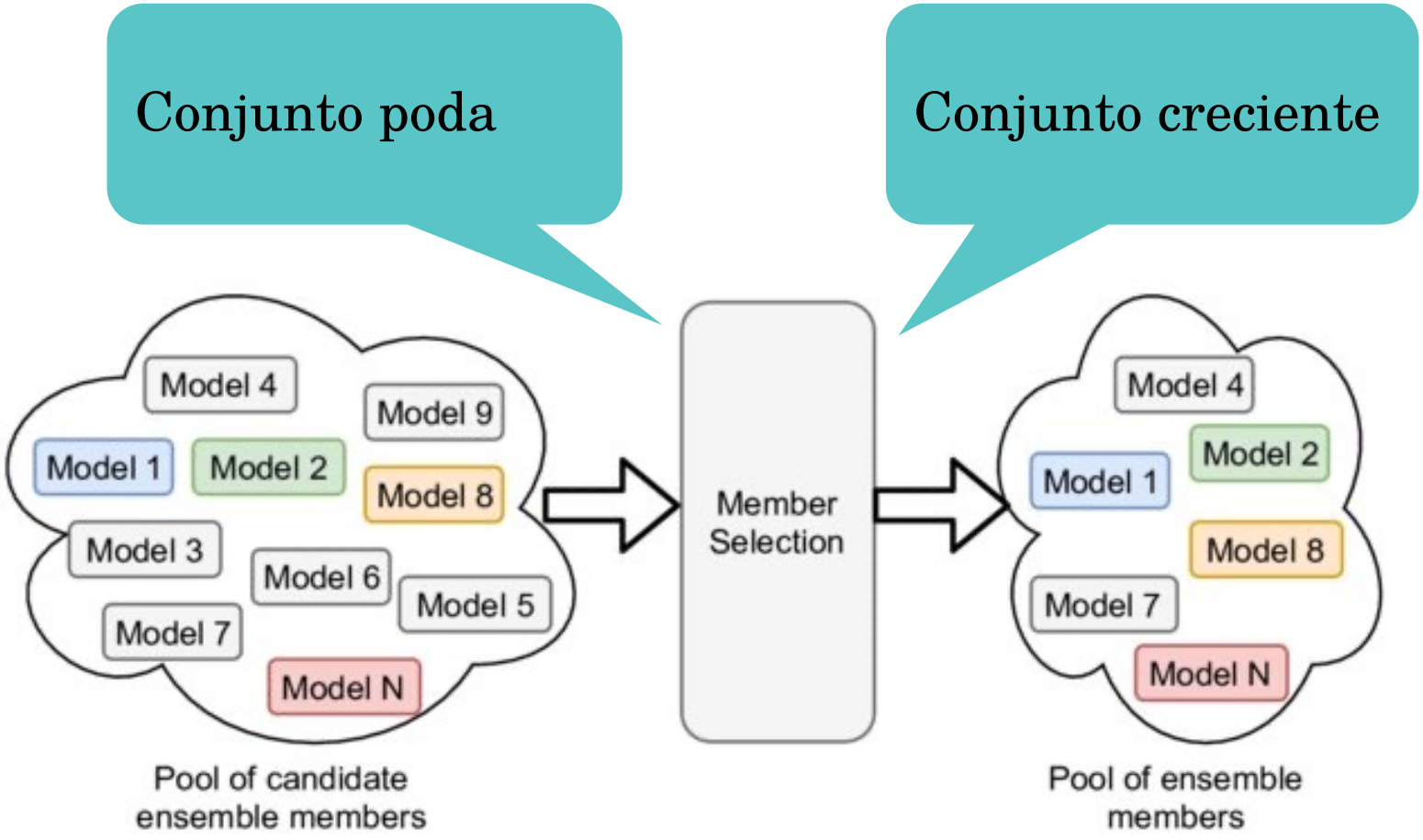
Weighted Average



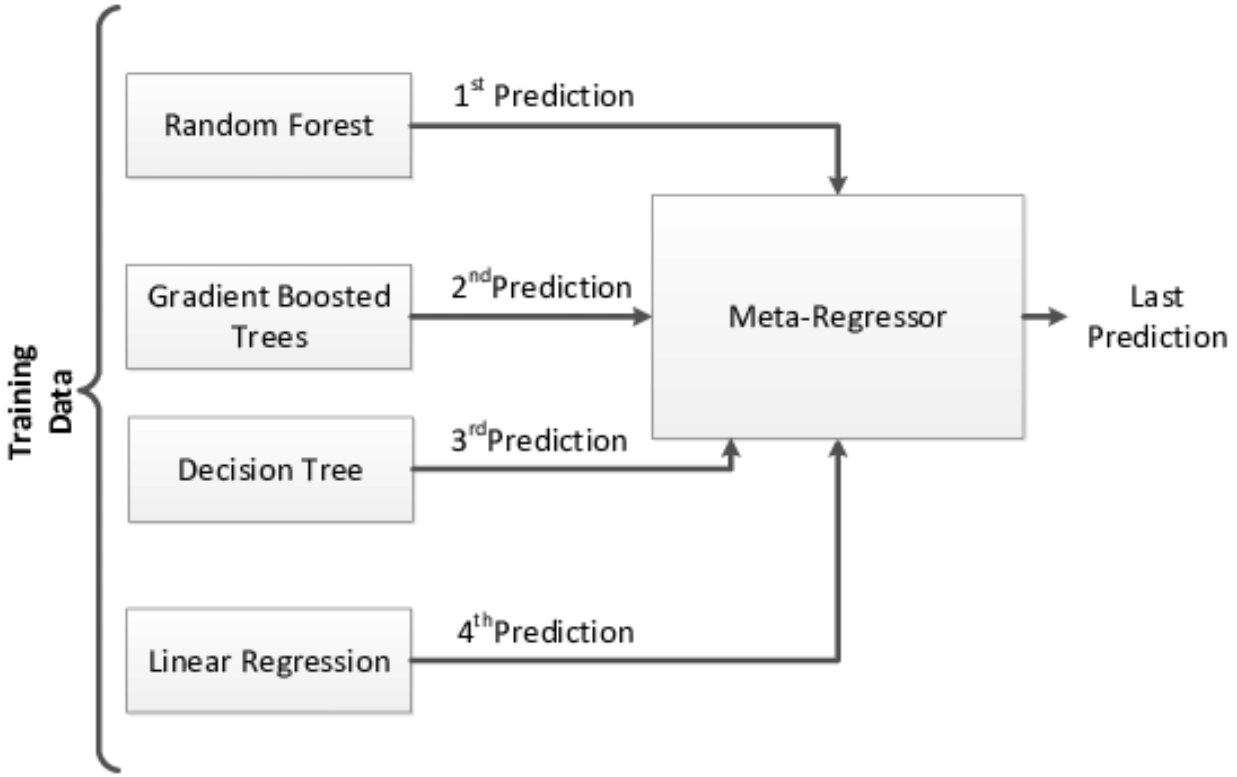
Selección de miembros del conjunto



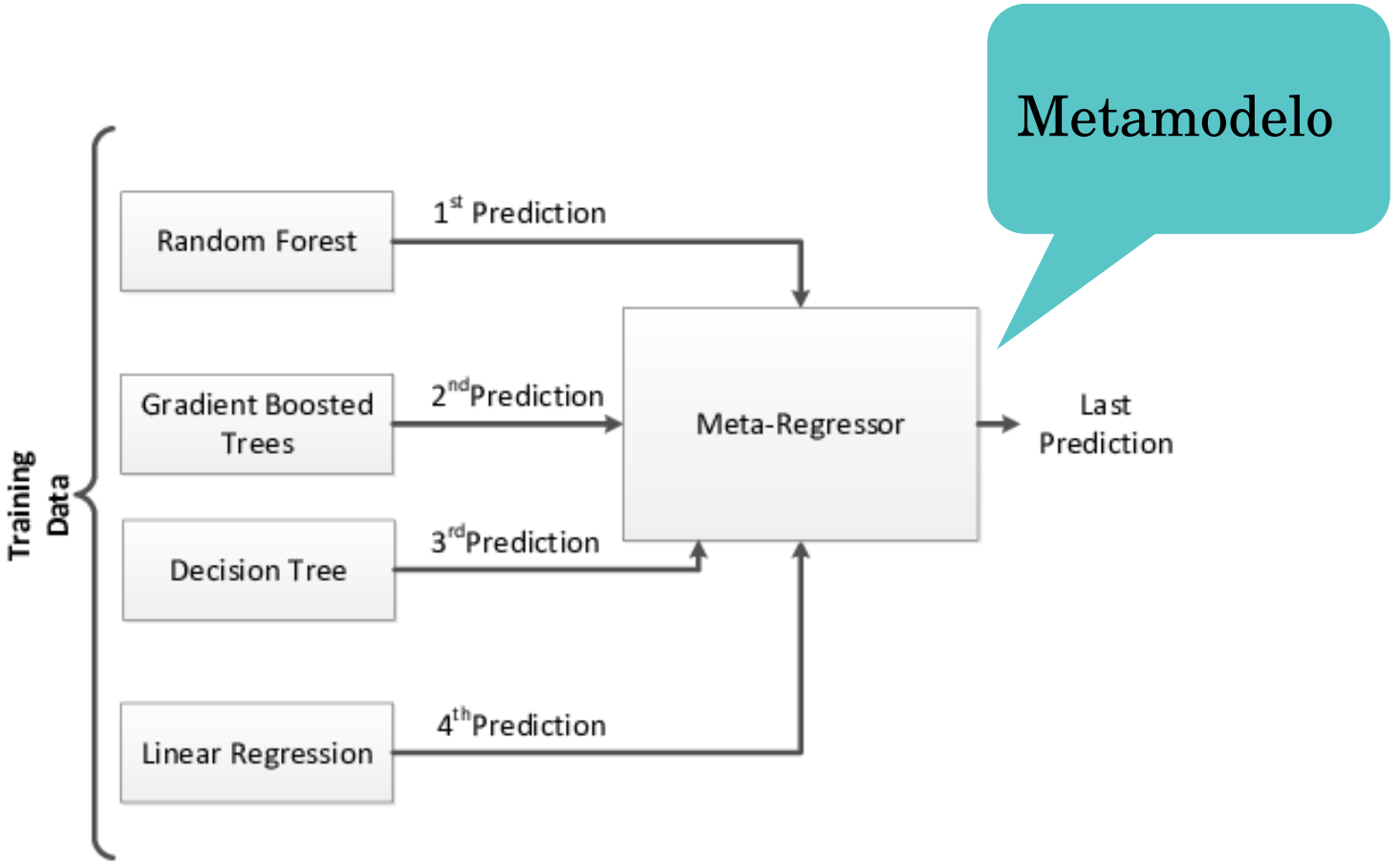
Selección de miembros del conjunto



Stacking


Training Data
Random Forest
1st Prediction
Gradient Boosted Trees
2nd Prediction
Decision Tree
3rd Prediction
Linear Regression
4th Prediction
Meta-Regressor
Last Prediction

Stacking


Metamodelo
Training Data
Random Forest
Gradient Boosted Trees
Decision Tree
Linear Regression
1st Prediction
2nd Prediction
3rd Prediction
4th Prediction
Meta-Regressor
Last Prediction

Laboratorio de programación

A trabajar

¡Gracias!

Dr. Manuel Castillo-Cara, Luis Sarro
www.manuelcastillo.eu
Departamento de Inteligencia Artificial
Escuela Técnica Superior de Ingeniería Informática
Universidad Nacional de Educación a Distancia (UNED)