PSI

Percepción y Sistemas Inteligentes

Universidad del Valle

Universidad del Valle

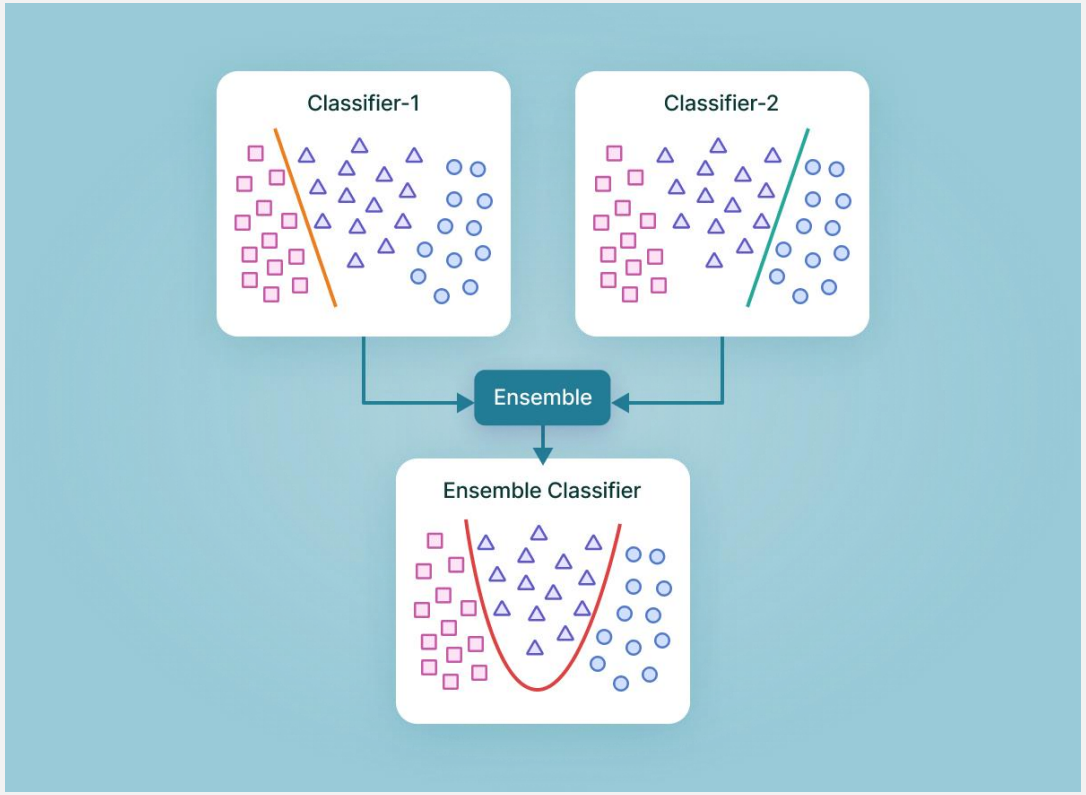
Métodos de Conjunto de Clasificadores

Ensemble Classification Methods

Humberto Loaiza Correa
Humberto.Loaiza@correounivalle.edu.co

Introducción

- Los **Métodos de Conjunto** (Ensemble) tiene como propósito construir un **grupo** de **máquinas** de aprendizaje y **combinarlos** para obtener un **desempeño mayor** al de la mejor máquina individual.



Introducción…

- Los conjuntos de clasificadores **también** se **denominan**:
  - Aprendizaje basado en comités
  - Aprendizaje con múltiples clasificadores
  - Ensamble de clasificadores (confunden la palabra *emsemble* con *assembly*)
- La **arquitectura general** contiene varias máquinas de aprendizaje **base,** como árboles de decisión, redes neuronales, SVM, clasificadores bayesianos, etc.
  - **Conjunto homogéneo**: todas las máquinas de aprendizaje son del **mismo tipo**.
  - **Conjunto heterogéneo**: todas las máquinas de aprendizaje son de **diferente tipo**.

Introducción…

- La capacidad de **generalización** de un **conjunto** suele ser **mucho mayor** que la de las máquinas **individuales**.
- Los métodos de conjunto son **atractivos** porque son capaces de potenciar (boost) **máquinas débiles** (desempeño ligeramente superior al azar), hasta convertirlos en **máquinas fuertes** que pueden hacer predicciones muy precisas.
- Los **métodos de conjuntos** se pueden **dividir** en:
  - Métodos de **combinación de clasificadores:**
    - Estudiados por la Comunidad de Reconocimiento de Patrones
  - Conjunto o **ensamble de clasificadores débiles**
    - Estudiados por la Comunidad de máquinas de aprendizaje
  - Mezcla de **expertos**
    - Estudiados por la Comunidad de Redes Neuronales

Introducción…

- **Métodos de combinación de clasificadores**
  - Utilizan máquinas fuertes con reglas (soft y hard) de combinación potentes para obtener clasificadores combinados más poderosos.
- **Conjunto o ensemble de clasificadores débiles**
  - Utilizan máquinas débiles y algoritmos potentes para potenciar el desempeño de los débiles.
  - Algunos métodos de ensamblaje son: AdaBoost, Bagging, etc.,
- **Mezcla de expertos**
  - Utilizan estrategias de divide y vencerás. Entrenan conjuntamente una mezcla de modelos paramétricos y utilizan reglas de combinación para obtener una solución global.

Introducción…

- **Pasos para construir un ensemble de clasificadores**
  - 1. Obtener las máquinas de aprendizaje base
  - 2. Establecer el procedimiento de combinación
- **Recomendación**
  - Utilizar máquinas de base lo más exactas (accurate) y diversas posible.

Introducción…

- **Observación**
  - El coste computacional de un **conjunto** no es mucho mayor que el de una **única máquina**.
    - Para una sola máquina se generan varias versiones para seleccionar el modelo y/o los parámetros; esto es comparable a la generación de las máquinas base para ensembles.
    - El coste computacional de la combinación de máquinas base suele ser pequeño, ya que la mayoría de las estrategias de combinación son sencillas.
  - Hay **trabajos empíricos**[1] que comprueban que el desempeño de un conjunto de clasificadores puede ser mayor que el mejor clasificador individual.
  - Hay **trabajos teóricos**[2] que prueban que máquinas débiles pueden convertirse en fuertes
  - [1] [Hansen and Salamon, 1990], [2] [Schapire, 1990],

Introducción…

- **Observación…**

  - Dado que las **máquinas** de aprendizaje **fuertes** son **deseables** pero **difíciles** de conseguir, y que las **débiles** son **fáciles** de obtener, los métodos de ensemble constituyen una prometedora vía para generar máquinas fuertes.

Universidad del Valle

Combinación de Clasificadores

Humberto Loaiza Correa
Humberto.Loaiza@correounivalle.edu.co

Introducción

- Se aprovechan las **ventajas individuales** de cada clasificador para alcanzar un **rendimiento global mejor** que el que podría lograrse utilizando cada uno de ellos **individualmente**.

- Esta técnica se justifica en que las distintas máquinas de aprendizaje diseñadas para un problema en particular generalmente **fallan** en la clasificación en **diferentes patrones**.

    - Es decir, incluso el "mejor" clasificador puede fallar en patrones en los que otros clasificadores tienen éxito.

- La combinación de clasificadores pretende **explotar** la **información complementaria** que residen en los distintos clasificadores.

Introducción…

- En el diseño deben considerarse los siguientes pasos.

  - **Seleccionar la regla** para combinar las salidas de las máquinas de aprendizaje individuales para obtener el mayor desempeño global.
  - **Determinar** si todos los clasificadores se **alimentan** con los **mismos vectores** de características, o si con vectores de características diferentes.

Introducción…

- Para un problema de M clases se tienen L clasificadores entrenados para generar como salida una probabilidad a posteriori cuando se evalúa el vector de características $\boldsymbol{x}$, es decir:

$$P_j(\omega_i|\boldsymbol{x}) \;;\quad i = 1,2,\dots,M;\quad j = 1,2,\dots,L$$

- Se busca obtener una probabilidad a posteriori final o global $P(\omega_i|\boldsymbol{x})$ mayor a partir de las probabilidades individuales.

- Una de las maneras de combinar $P_j(\omega_i|\boldsymbol{x})$ es utilizando la distancia de probabilidad de Kullback-Leibler (KL)

Introducción…

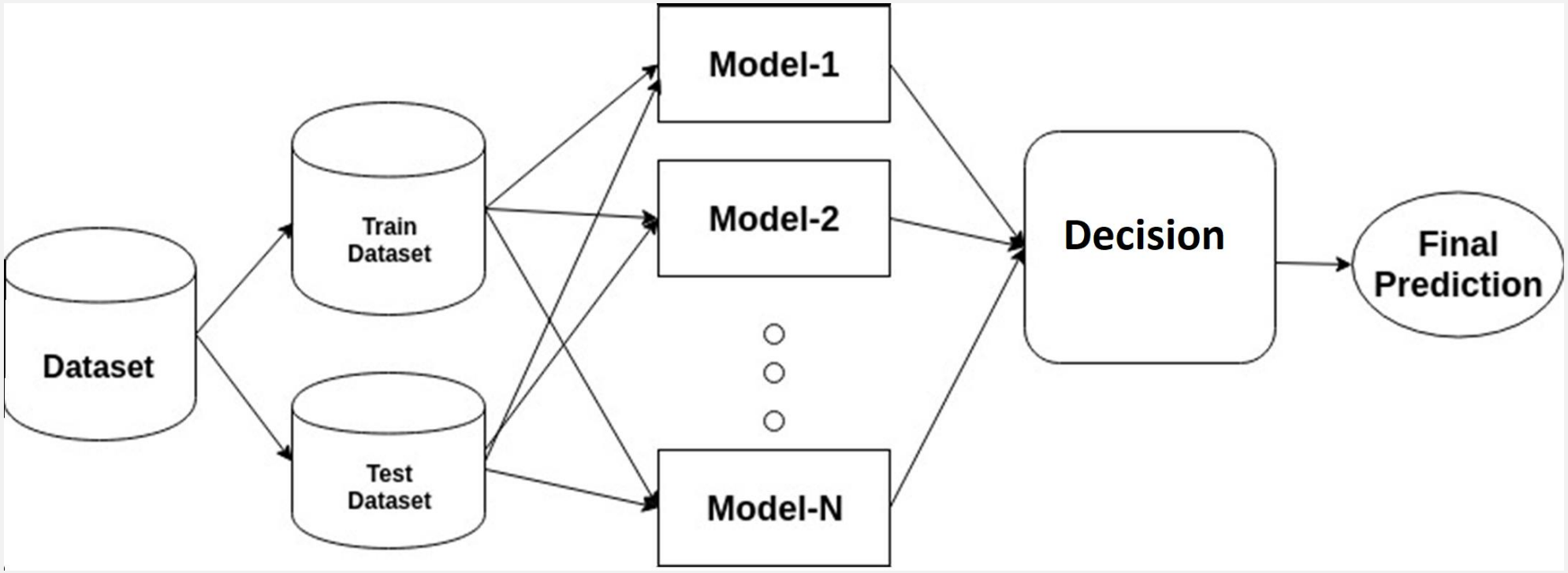
- Para un problema de M clases se tienen L clasificadores entrenados para generar como salida una probabilidad a posteriori cuando se evalúa el vector de características $\boldsymbol{x}$, es decir:

$$P_j(\omega_i|\boldsymbol{x}) \;;\quad i = 1,2,\dots,M;\quad j = 1,2,\dots,L$$

- Se busca obtener una probabilidad a posteriori final o global $P(\omega_i|\boldsymbol{x})$ mayor a partir de las probabilidades individuales.

- Una de las maneras de combinar $P_j(\omega_i|\boldsymbol{x})$ es utilizando la distancia de probabilidad de Kullback-Leibler (KL)

Introducción



Regla del Promedio Geométrico

- Busca obtener $P(\omega_i|\boldsymbol{x})$ (global) que minimice el promedio D_{av} de las distancias de probabilidad de Kullback-Leibler D_j. Donde,

$$D_{av} = \frac{1}{L}\sum_{j=1}^{L} D_j$$

$$D_j = \sum_{i=1}^{M} P(\omega_i|\boldsymbol{x}) \ \ln \frac{P(\omega_i|\boldsymbol{x})}{P_j(\omega_i|\boldsymbol{x})}$$

- Teniendo en cuenta que

$$\sum_{i=1}^{M} P_j(\omega_i|\boldsymbol{x}) = 1$$

Regla del Promedio Geométrico …

- Utilizando los multiplicadores de Lagrange para optimizar D_{av} respecto a $P(\omega_i|\boldsymbol{x})$ se obtiene:

$$P(\omega_i|\boldsymbol{x}) = \frac{1}{C}\prod_{j=1}^{L}\left(P_j(\omega_i|\boldsymbol{x})\right)^{\frac{1}{L}}$$

- Donde

$$C = \sum_{i=1}^{M}\prod_{j=1}^{L}\left(P_j(\omega_i|\boldsymbol{x})\right)^{\frac{1}{L}}$$

Regla del Promedio Geométrico …

- Al despreciar todos los términos comunes a todas las clases, la **regla de clasificación** se hace equivalente a asignar el vector desconocido a la clase que **maximice el producto** de las salidas individuales de las máquinas. Es decir:

$$P(\omega_i|\boldsymbol{x}) = \max_{\omega_i} \prod_{j=1}^{L} P_j(\omega_i|\boldsymbol{x})$$

- Esta regla también recibe el nombre de **Regla del Producto**.

Regla del Promedio Aritmético

- Debido a que la distancia de probabilidades Kullback-Leible no es una medida real de distancia (de acuerdo con la definición estrictamente matemática) por no ser simétrica, se utiliza la **distancia alternativa** KL:

$$D_j = \sum_{i=1}^{M} P_j(\omega_i|\boldsymbol{x}) \ln \frac{P_j(\omega_i|\boldsymbol{x})}{P(\omega_i|\boldsymbol{x})}$$

- La regla de combinación busca obtener $P(\omega_i|\boldsymbol{x})$ (global) que minimice el promedio D_{av} de las **distancias** de probabilidad **alternativa** de Kullback-Leibler D_j.

$$D_{av} = \frac{1}{L} \sum_{j=1}^{L} D_j$$

Regla del Promedio Aritmético …

- Utilizando los multiplicadores de Lagrange para optimizar D_{av} respecto a $P(\omega_i|\boldsymbol{x})$ y la **regla de clasificación** se hace equivalente a asignar el vector desconocido a la clase que **maximice el promedio** de las salidas individuales de las máquinas. Es decir:

$$P(\omega_i|\boldsymbol{x}) = \max_{\omega_i} \frac{1}{L} \sum_{j=1}^{L} P_j(\omega_i|\boldsymbol{x})$$

- Observación:
  - La regla del producto produce **mejores resultados** que la regla del promedio, pero puede producir resultados **menos confiables** cuando la salida $P_j(\omega_i|\boldsymbol{x})$ de algún clasificador es **cercana a cero**.
  - La regla del producto y del promedio son del tipo soft.

Regla del Voto Mayoritario

- La regla consiste en asignar el vector de características x a la clase ω_i para la cual exista un **consenso** o cuando **al menos** L_c clasificadores **coincidan**, donde

$$L_c = \begin{cases} \frac{L}{2} + 1, & para\ \ L\ par \\ \frac{L+1}{2} & para\ \ L\ impar \end{cases}$$

  - De lo contrario, **no se realiza** la asignación (opción de rechazo).

Regla del Voto Mayoritario …

- Para un sistema de L clasificadores previamente entrenados y bajo las siguientes consideraciones:
  - i. El número L es impar.
  - ii. Cada clasificador tiene la misma probabilidad p de clasificación correcta.
    - Puede obtenerse con base en el desempeño del dataset de test.
    - Variaciones de p entre los clasificadores no son críticas.
  - iii. La decisión de cada clasificador es independiente de los otros.
    - Difícil de cumplir dado que los clasificadores se alimentan de las mismas características extraídas del mismo dataset de entrenamiento.

Regla del Voto Mayoritario …

- La probabilidad de correcta clasificación $P_c(L)$después del voto mayoritario una distribución binomial

$$P_c(L) = \sum_{m=L_c}^{L} \binom{L}{m} p^m \ (1-p)^{L-m}$$

- Para $L \geq 3$ se cumple que:
  - Si $p > 0.5$, $P_c(L)$ incrementa monotónicamente con L
    - $P_c(L) \to \infty$ cuando $L \to \infty$
  - Si $p < 0.5$, $P_c(L)$ decres monotónicamente con L
    - $P_c(L) \to 0$ cuando $L \to \infty$
  - Si $p = 0.5$, $P_c(L) = 0.5$ para todo L

Regla del Voto Mayoritario …

- Lo anterior significa que la **probabilidad** de correcta clasificación $P_c(L)$ por voto mayoritario, bajo los supuestos mencionados, se **incrementa** con L siempre que se cumpla $p > 0.5$.
- La correcta clasificación está restringida al límite de Bayes.
- Como el **supuesto** de **independencia** de los L clasificadores **no es válido**, debido a que se entrenan con el mismo dataset de características extraídas, se proponen **alternativas**:
  - Generar **reglas** de combinación para **clasificadores dependientes**.
    - Aplicado en esquemas de combinación **soft** (Ej: producto, promedio,…)
  - Construir clasificadores más **independientes** (ejemplo: Entrenar clasificadores con características diferentes extraídas de datasets distintos).
    - Aplicado en esquemas de combinación **hard** (Ej: Voto Mayoritario)

Universidad del Valle

Ensemble de Clasificadores

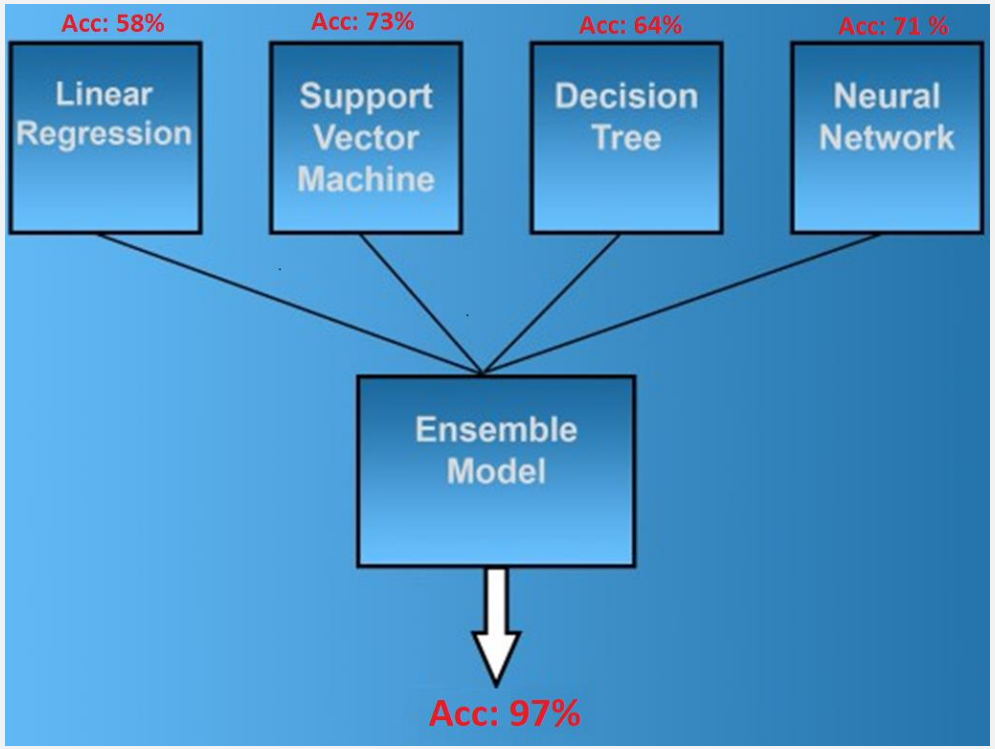
Humberto Loaiza Correa
Humberto.Loaiza@correounivalle.edu.co

Introducción

- Los métodos ensemble (conjunto) entrenan **múltiples máquinas** para resolver el **mismo problema**.

- Mientras que los métodos de aprendizaje ordinarios construyen una sola máquina a partir de los datos de entrenamiento, los métodos ensemble construyen un conjunto de máquinas para combinarlos.

- El aprendizaje por **conjuntos** también se denomina:
  - Aprendizaje basado en comités (Committee-based learning)
  - Aprendizaje de sistemas clasificadores múltiples (Learning Multiple classifier systems).

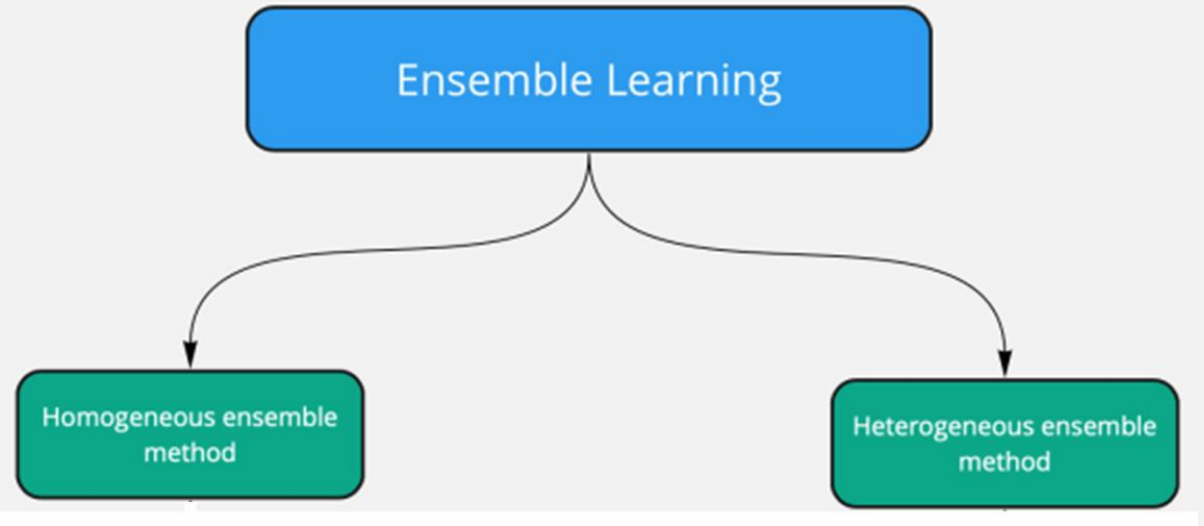


Introducción…

- Un ensemble (conjunto) contiene un grupo de máquinas de aprendizaje denominados **modelos base** (aprendices, máquinas, …).

- Los **modelos de aprendizaje base** se generan a partir de datos de entrenamiento mediante un **algoritmo de aprendizaje base** que puede ser un árbol de decisión, RNA, SVM, etc.

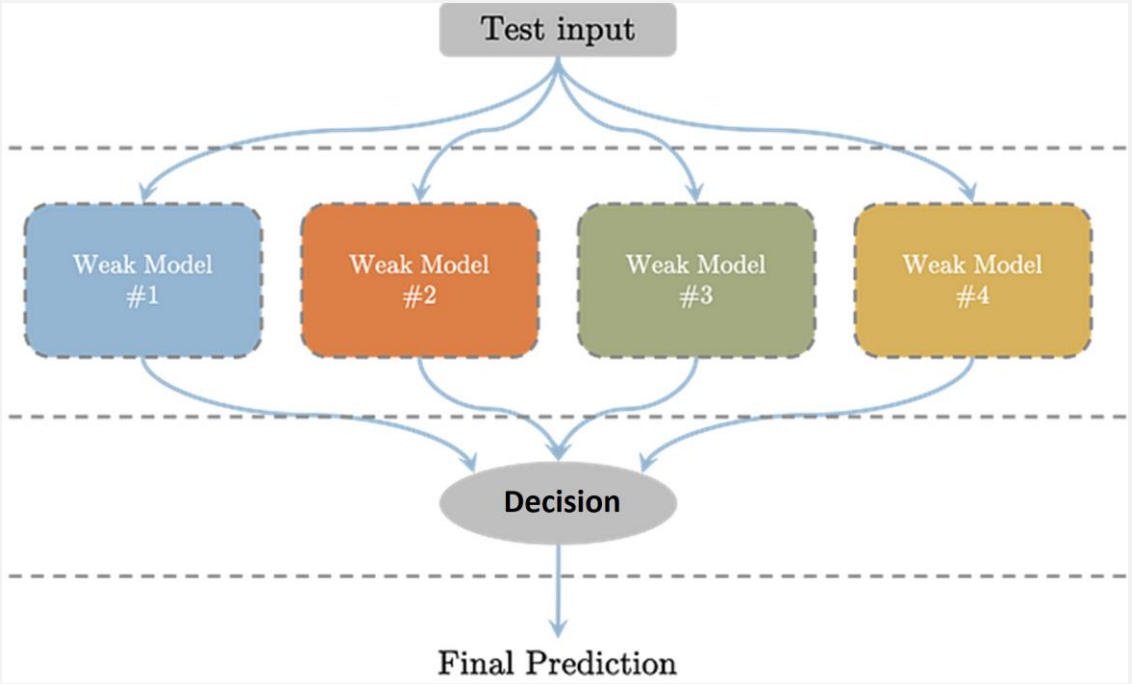
- Aprendizaje de Ensemble **Homogéneo**
  - Utilizan **un mismo** modelo base.
- Aprendizaje dee Ensemble **Heterogéneo**
  - Utilizan modelos base **diferentes**.



Introducción…

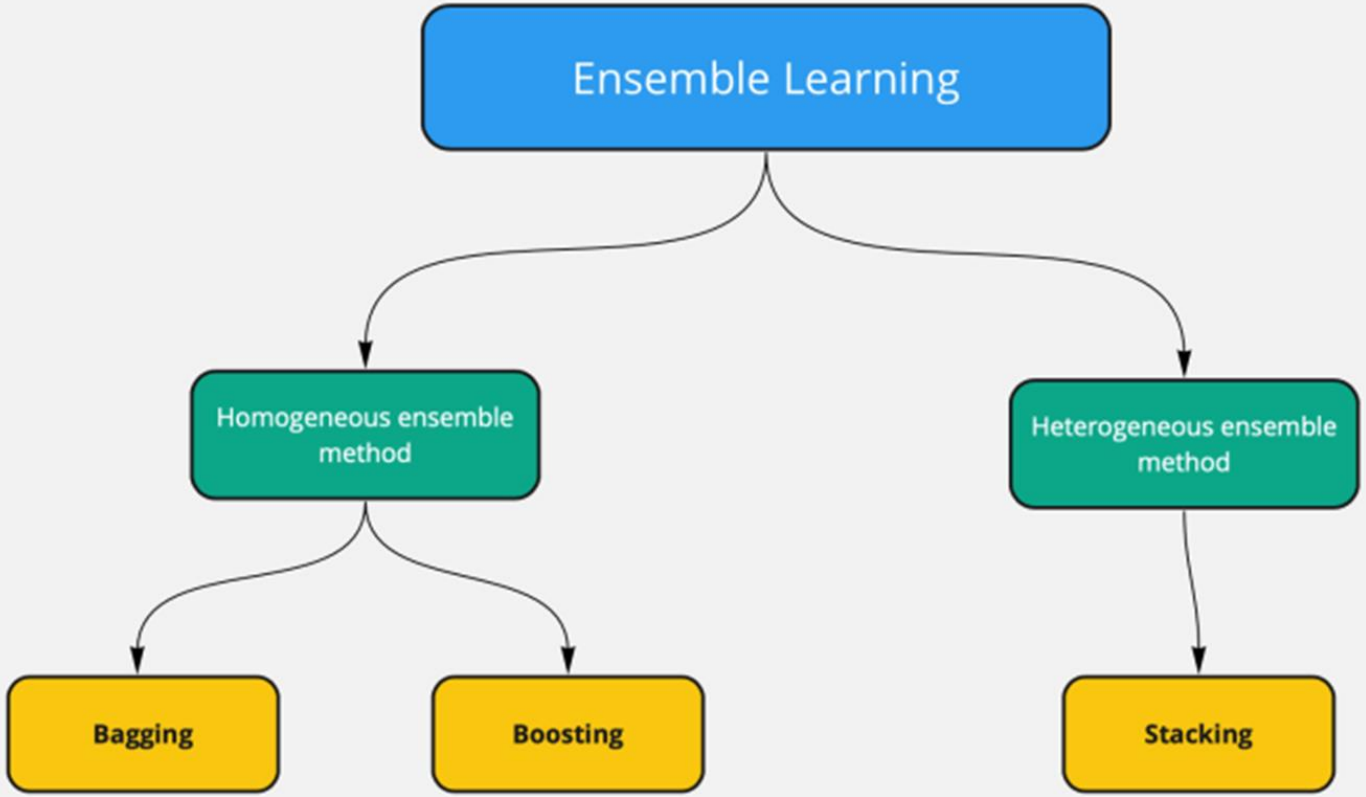
- La capacidad de **generalización** de un **ensemble** suele ser **mucho mayor** que la de los **modelos base**.

- Los métodos de **ensemble** son capaces de **potenciar** (boost) los **modelos** de aprendizaje **débiles**, (ligeramente mejor que el aleatorio), hasta convertirlos en **modelos** de aprendizaje **fuertes** que pueden hacer predicciones muy precisas.
- Por esto, los **modelos base** también se denominan **modelos débiles**.



Introducción…

- Los principales **tipos** de **ensemble** de clasificadores son:
  - **Bagging**
  - **Boosting**
  - **Stacking**

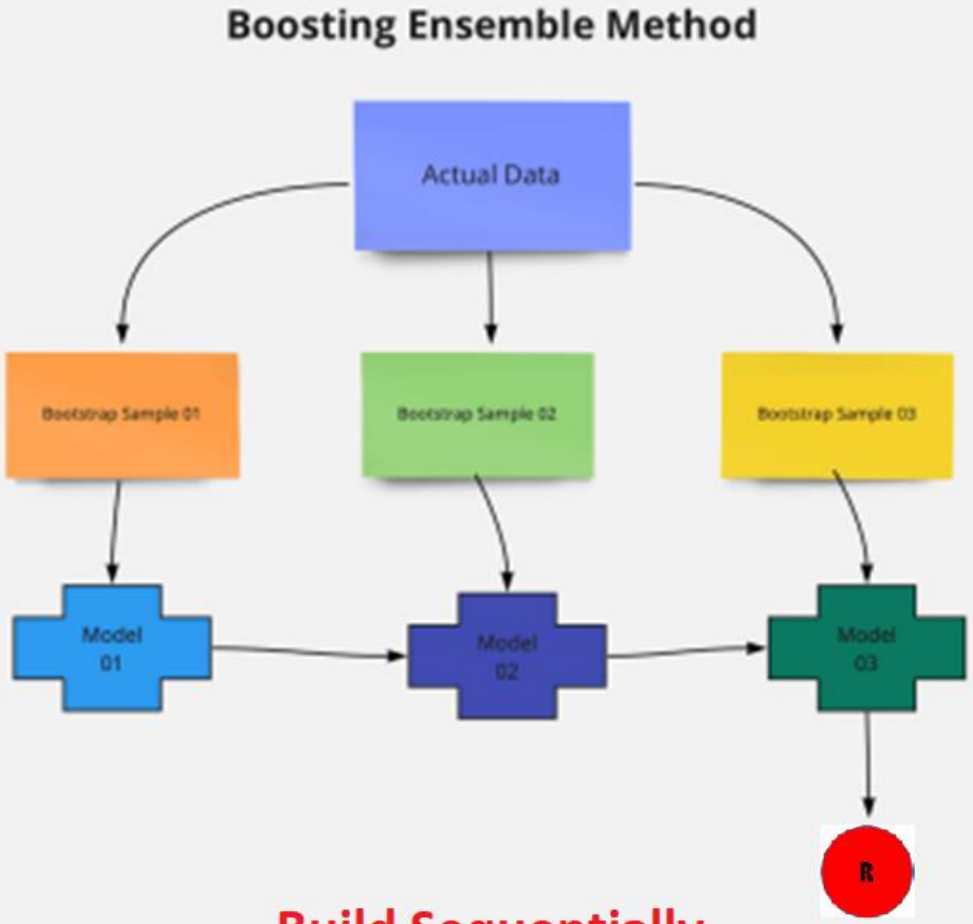


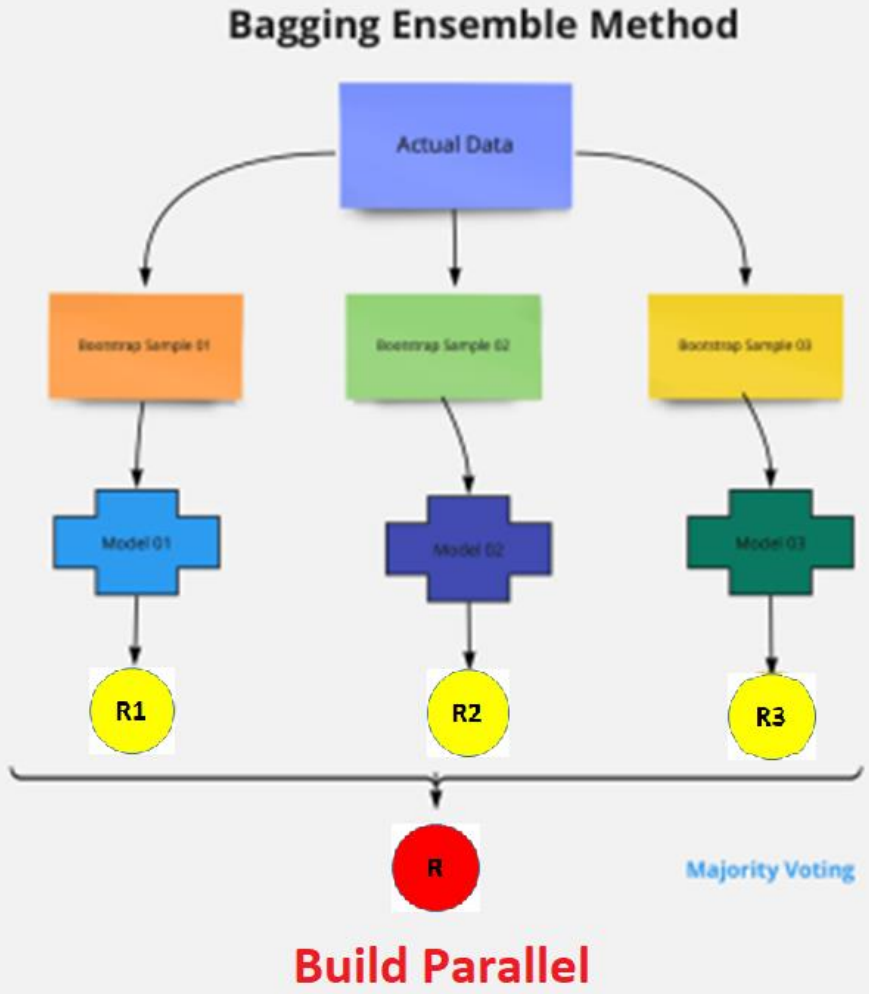
Introducción…

- **Paradigmas de construcción de ensembles**
  - Existen **dos paradigmas** generales de cómo **generar (enseñar) modelos base:**
    - Métodos de **ensemble secuenciales :** (modelos base se generan secuencialmente)
      - Aprovechan la **dependencia** entre los modelos base, ya que el **rendimiento** global se puede **aumentar** de forma residual decreciente.
      - Ejemplo: AdaBoost
    - Métodos de **ensemble paralelos (** modelos base se generan en paralelo)
      - Aprovechan la **independencia** entre los modelos base, ya que el **error** puede **reducirse** drásticamente al combinarlos.
      - Ejemplo: Bagging

Introducción…

Paradigmas de construcción de ensembles





Universidad del Valle

PSI

Percepción y Sistemas Inteligentes

Bootstrapping

Humberto Loaiza Correa
Humberto.Loaiza@correounivalle.edu.co

Bootstrapping

- El bootstrap es un **método estadístico** para **crear nuevos** datasets (sample data) a partir del dataset inicial, **conservando** las **propiedades** del dataset real.

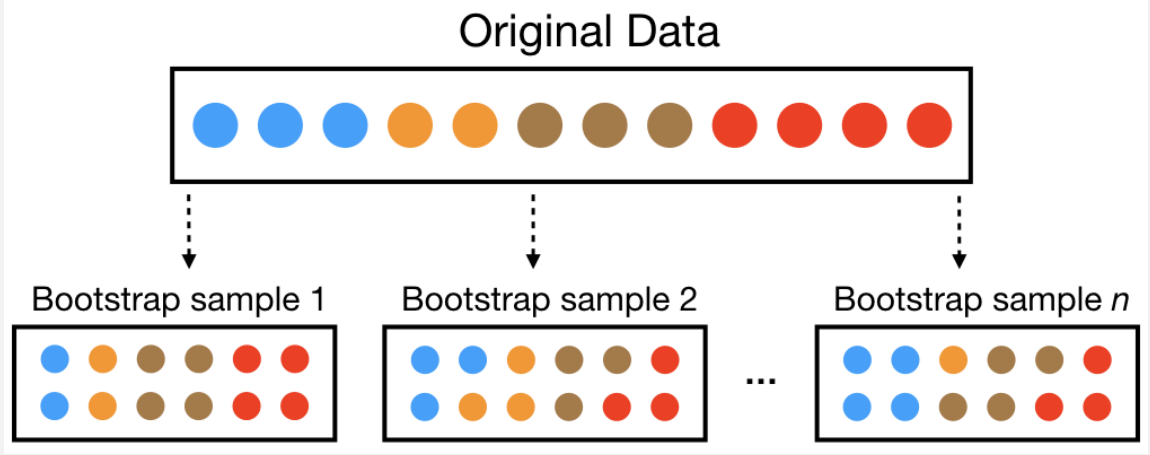
- **Cada** dataset es una **aproximación** a los datos reales y deben capturar la complejidad subyacente de los **datos reales**.

- Todos los puntos de los nuevos datasets (sample data) se toman **aleatoriamente** del original **con reemplazo** (se utilizan nuevamente para generar otros datasets).

- Este proceso permite calcular errores estándar, construir intervalos de confianza y realizar pruebas de hipótesis para numerosos tipos de estadísticas muestrales.

Bootstrapping…

- El **bootstrap** se utiliza en algoritmos de aprendizaje de ensemble (bagging), para:
  - Evitar el sobreajuste y mejorar la estabilidad de los algoritmos de aprendizaje automático.
  - Evaluar la precisión de la estimación de un parámetro o una predicción.
  - Mejorar la estimación o predicción.



Universidad del Valle

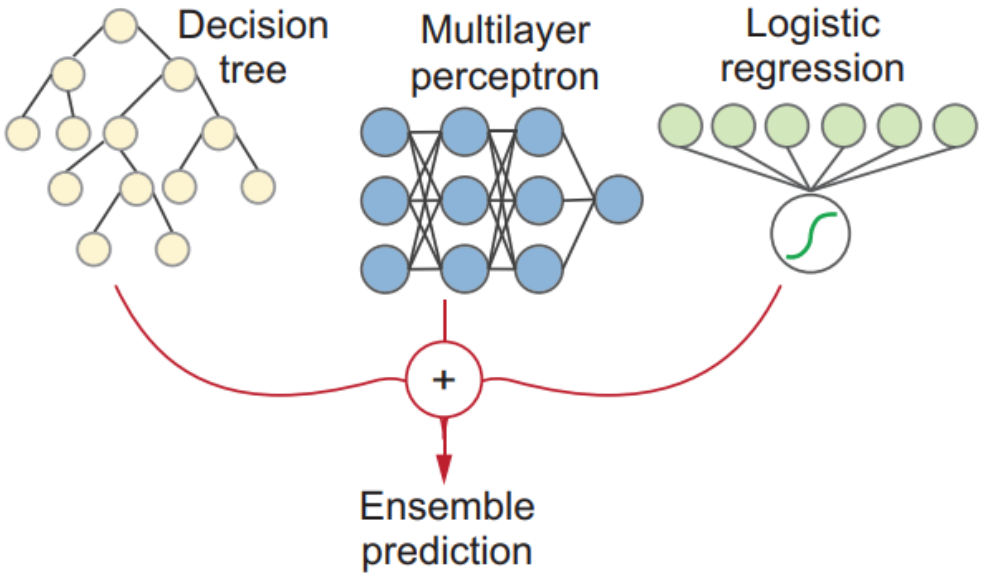
Parallel Ensembles

Humberto Loaiza Correa
Humberto.Loaiza@correounivalle.edu.co

PARALLEL HOMOGENEOUS ENSEMBLES

Use many strong learners, or complex models, trained using the same base machine-learning algorithm. Ensemble diversity is created from a single algorithm with random data or feature sampling for training each base model.

Ensembles in this family: bagging, random forests pasting, random subspaces, random patches, extremely randomized trees (Extra Trees)



Universidad del Valle

PSI

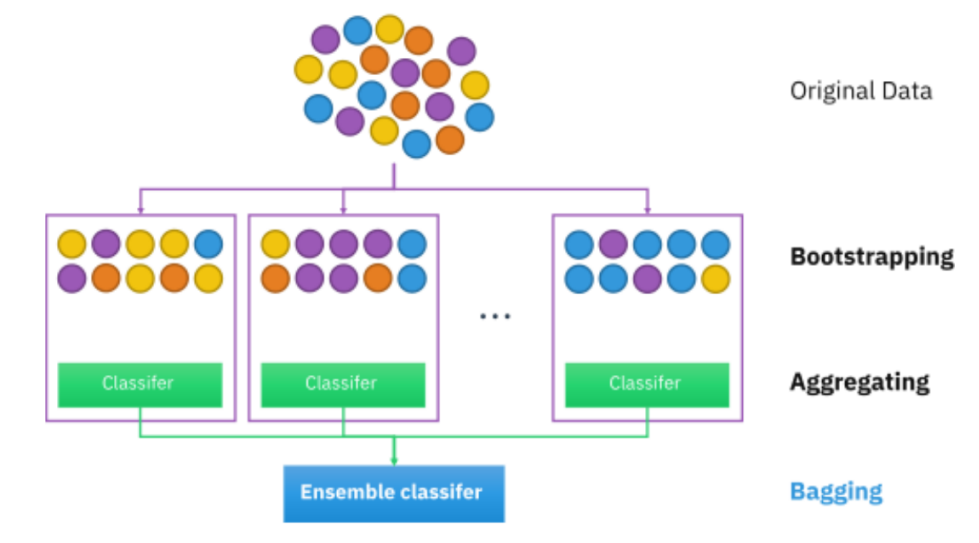
Percepción y Sistemas Inteligentes

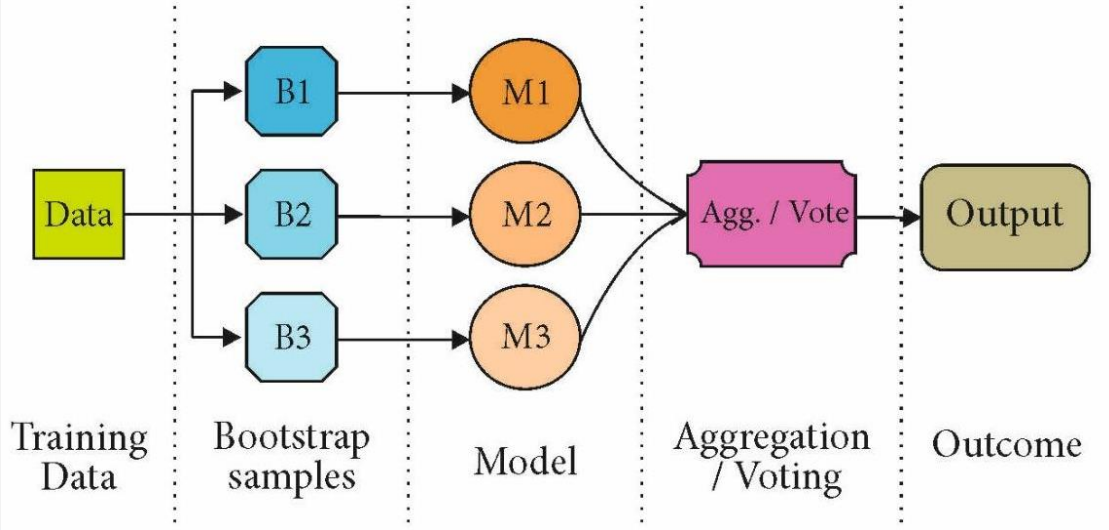
Bagging

Humberto Loaiza Correa
Humberto.Loaiza@correounivalle.edu.co

Bagging

- **Introducción**
  - Método de **ensemble paralelo**, que utiliza aprendices (modelos) base lo más independientes posible para decrementar el error.
  - **Bagging** corresponde a la abreviatura de **Bootstrap AGGregatING** [Breiman, 1996d].
    - Presenta dos etapas fundamentales: bootstrap y agregación.





Bagging

- **Introducción…**
  - Para generar los **aprendices** (modelos) **base** lo más **independiente** posible se utiliza el **Bootstrap sampling** para crear una serie de subconjuntos de datos con los que se entrena cada aprendiz base.
  - El **muestreo con reemplazo** permite que de un dataset de entrenamiento original con m ejemplos, se obtenga T nuevos datasets de m ejemplos de entrenamiento.
  - A partir de cada nuevo dataset se **entrena** un aprendiz base aplicando el **algoritmo de aprendizaje base**.
  - Algunos datos originales aparecen más de una vez, mientras que otros ejemplos originales no están presentes en los nuevos datasets.

Bagging

- **Introducción…**
  - El **Bagging** adopta las estrategias más populares para **agregar** los resultados de los aprendices de base: **votación** para la clasificación y **promedio** para la regresión.
  - Para clasificación, Bagging alimenta la instancia a sus clasificadores base y recoge todas sus salidas, luego vota las etiquetas y toma la etiqueta ganadora como predicción. Los empates se dirimen arbitrariamente.
  - Bagging puede aplicarse a clasificación binaria y multiclase.

Bagging

- **Algoritmo**

Bagging

Input: Data set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
Base learning algorithm $\mathfrak{L}$;
Number of base learners T.

Process:
1. **for** $t = 1, \ldots, T$:
2. $h_t = \mathfrak{L}(D, \mathcal{D}_{bs})$ % $\mathcal{D}_{bs}$ is the bootstrap distribution
3. **end**

Output: $H(\boldsymbol{x}) = \arg\max_{y \in \mathcal{Y}} \sum_{t=1}^{T} \mathbb{I}(h_t(\boldsymbol{x}) = y)$

Bagging

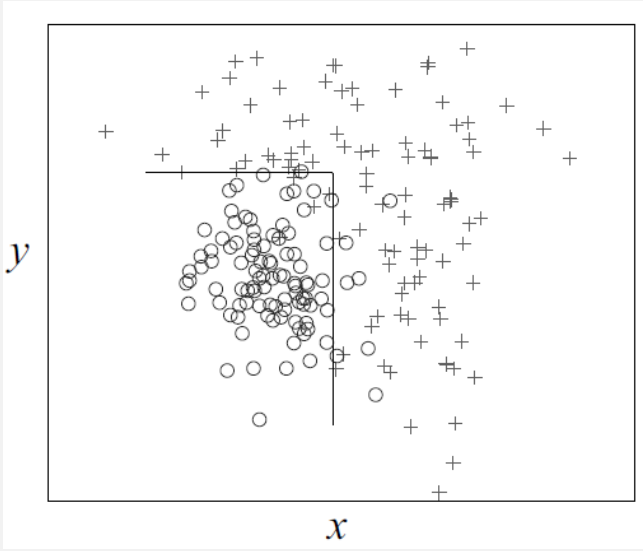
- **Algoritmo ..**
  - Para cada aprendiz base, hay aproximadamente un 36,8% de ejemplos de entrenamiento originales que **no se utilizan** en su proceso de entrenamiento.
  - Los datasets generados por bootstrap tienen un gran **solapamiento** (por ejemplo 63,2%) con el conjunto de datos original.
  - El **desempeño** del aprendiz base puede **estimarse** utilizando los **ejemplos fuera** de la bolsa y, a partir de ahí, se puede estimar el **error de generalización** del bagging.

Bagging

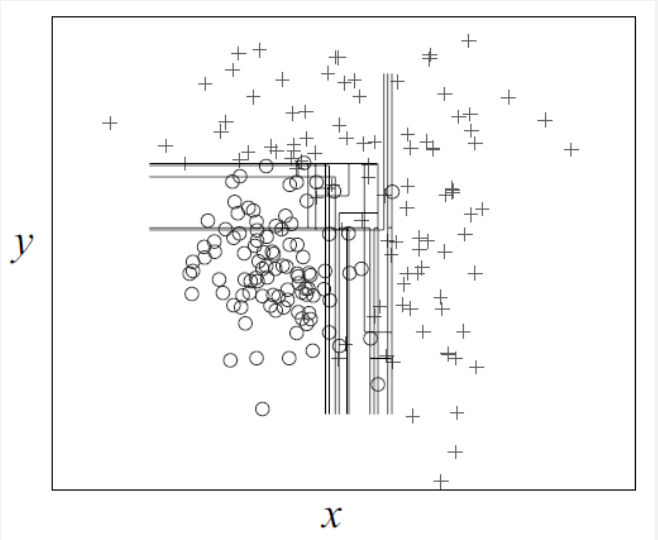
- **Algoritmo ..**
  - Si un **algoritmo de aprendizaje base** es **insensible a la perturbación** de las muestras de entrenamiento, los **aprendices base entrenados** pueden ser **bastante similares** y, por lo tanto, combinarlos **no ayudará** a mejorar la **generalización**.
  - Estos **aprendices** se denominan **estables.**
    - Ejemplo: kNN, LDA
  - Bagging genera **mejor generalización** con **aprendices inestables**
    - Ejemplo: árboles de decisión *sin poda*

Bagging

- **Ejemplo Ilustrativo**
  - Dos clases, dos dimensiones, dataset generado por pdf de tres gaussianas.
  - Fronteras de decisión de un **árbol de decisión** y el resultante de bagging con $T = 10$ árboles de decisión.

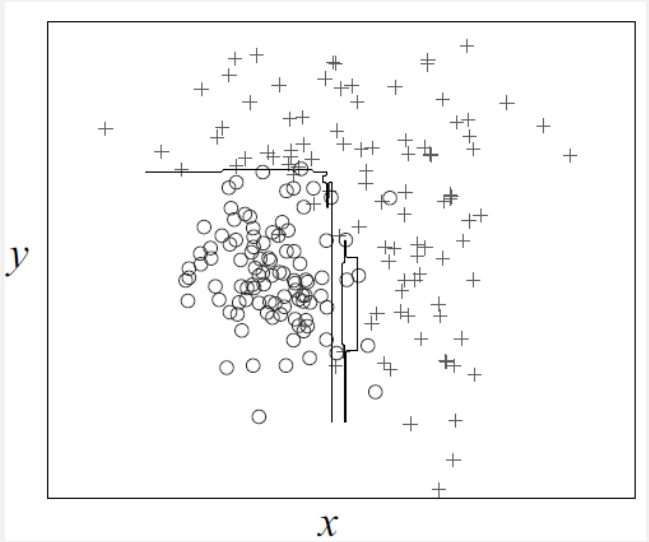


Un solo árbol de decisión



Fronteras de 10 árboles de decisión

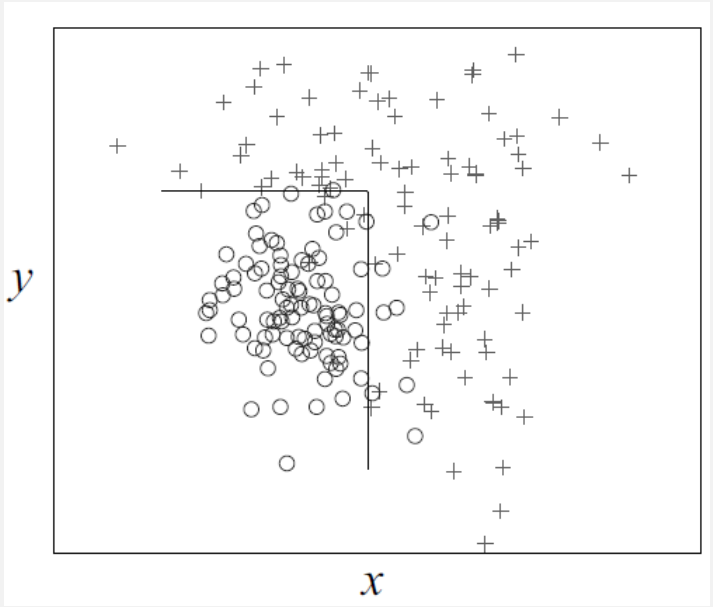
Aprendiz Inestable!



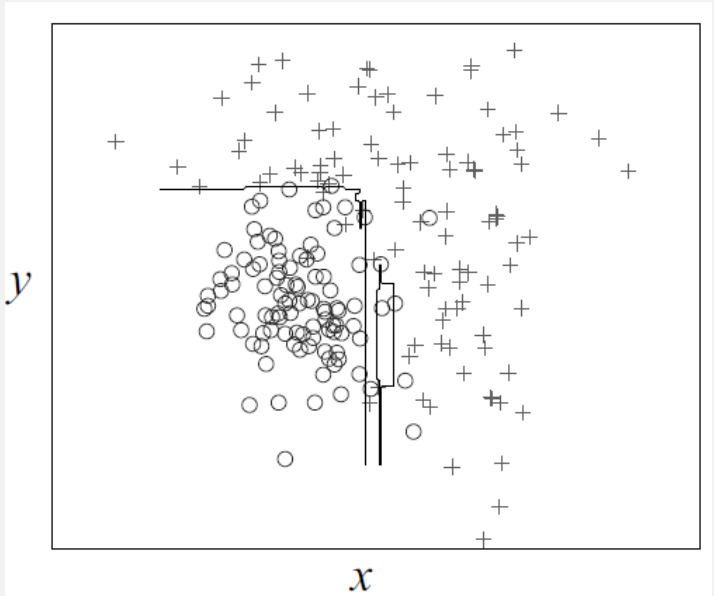
Bagging con 10 árboles de decisión

Bagging

- **Ejemplo Ilustrativo ..**
  - Bagging presenta **frontera** de decisión **más flexible** que la del árbol solo.
  - Se reduce el error de 9.4% a 8.3%



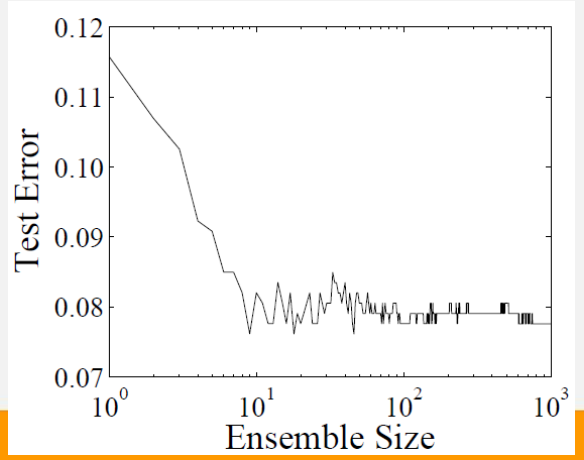
Árbol de decisión

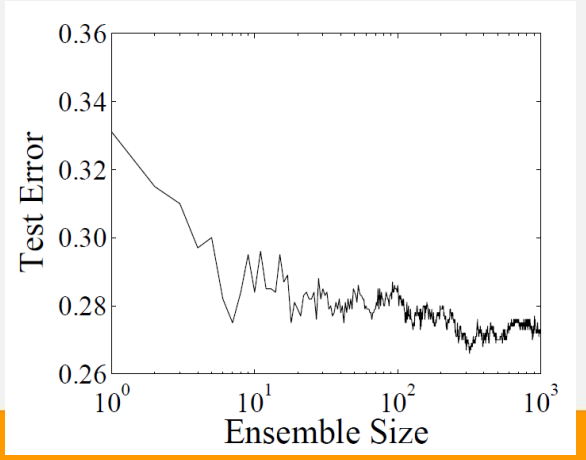


Bagging con 10 árboles de decisión

Bagging

- **Observaciones**
  - Usar aprendices base **independientes**, hace que el **error** de **generalización** se **reduzca** exponencialmente con el **tamaño** T del ensemble. Idealmente tiende a cero cuando $T \rightarrow \infty$.
  - En la **práctica**, **no** se dispone de **infinitos datos** de entrenamiento, y los aprendices base del Bagging **no son independientes**, ya que se entrenan a partir de muestras bootstrap.
  - Sin embargo, aunque el **error** no descienda a cero, el desempeño del Bagging **converge** a medida que **aumenta** el tamaño del conjunto, es decir, **el número** de **aprendices base**.





Bagging

- **Observaciones**
  - Bagging **funciona mejor** con aprendices base **altamente no-lineales**, dado que estos son **inestables** (desempeño cambia con la perturbación del dataset de entrenamiento).
  - Bagging **convergerá a una tasa de error constante** a medida que **aumenta el tamaño** del conjunto (L aprendices). Excepto en el raro caso de que el desempeño de Bagging sea igual aleatorio.
  - El clasificador Bagging ayuda a **reducir la varianza** de los estimadores individuales introduciendo **aleatoriedad** en la fase de entrenamiento de cada uno de los estimadores y haciendo un ensemble de todos los estimadores.

Bagging

- **Variantes del Algoritmo Bagging**
  - Los métodos de Bagging son muy **variados**, pero se **diferencian** principalmente por la forma en que **extraen** subconjuntos aleatorios a partir del dataset de entrenamiento
  - **Pasting:** Cuando los subconjuntos para entrenar los aprendices se extraen como subconjuntos aleatorios a partir de muestras del dataset original.
  - **Bagging**: Cuando los subconjuntos para entrenar los aprendices se obtienen de variaciones sobre la fdp del dataset original y con reemplazo (las mismas muestras pueden estar en varios cubconjuntos)
  - **Random Subspaces:** Cuando los subconjuntos para entrenar los aprendices se obtienen de un conjunto aleatorio de las características del dataset original.
  - **Random Patches:** Cuando los subconjuntos para entrenar los aprendices se obtienen de subconjuntos de muestras y de subconjuntos de características.

Bagging

- **Ejemplo**:
  - Cancer_Classification_Bagging_SVM.ipynb
  - Cancer_Classification_Bagging_KNN.ipynb
  - Cancer_Classification_Bagging_NBayes.ipynb
  - Cancer_Classification_Bagging_LogR.ipynb
  - Cancer_Classification_Bagging_Tree.ipynb
- **Nota:**
  - En scikit-learn, los métodos de Bagging están disponibles como un **metaestimador** unificado **BaggingClassifier** (resp. BaggingRegressor), tomando como entrada un **estimador determinado** y los parámetros que especifican la **estrategia** para extraer **subconjuntos aleatorios**.

Bagging

- **Ejercicio en Clase**
  - A partir de los programas de Bagging implementar el algoritmo **Random Patches.**
    - **Random Patches:** Cuando los subconjuntos para entrenar los aprendices se obtienen de subconjuntos de muestras y de subconjuntos de características.
  - Nota: cambiar los hiperparámetros
    - **max_samples** (1.0 →100%)
    - **max_features** (1.0 →100%)
  - Ver:
    - https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html

Bagging

- **Ejercicio en Clase …**
  - Obtenga un cambio de hiperparámetros que mejore el **Random Patches frente al** Bagging
- **Probar en:** :
  - Cancer_Classification_Bagging_SVM.ipynb
  - Cancer_Classification_Bagging_KNN.ipynb
  - Cancer_Classification_Bagging_NBayes.ipynb
  - Cancer_Classification_Bagging_LogR.ipynb
  - Cancer_Classification_Bagging_Tree.ipynb
- **Clasificar el dataset wine con Bagging**
  - https://scikit-learn.org/stable/datasets/toy_dataset.html

Universidad del Valle

PSI

Percepción y Sistemas Inteligentes

Random Forest

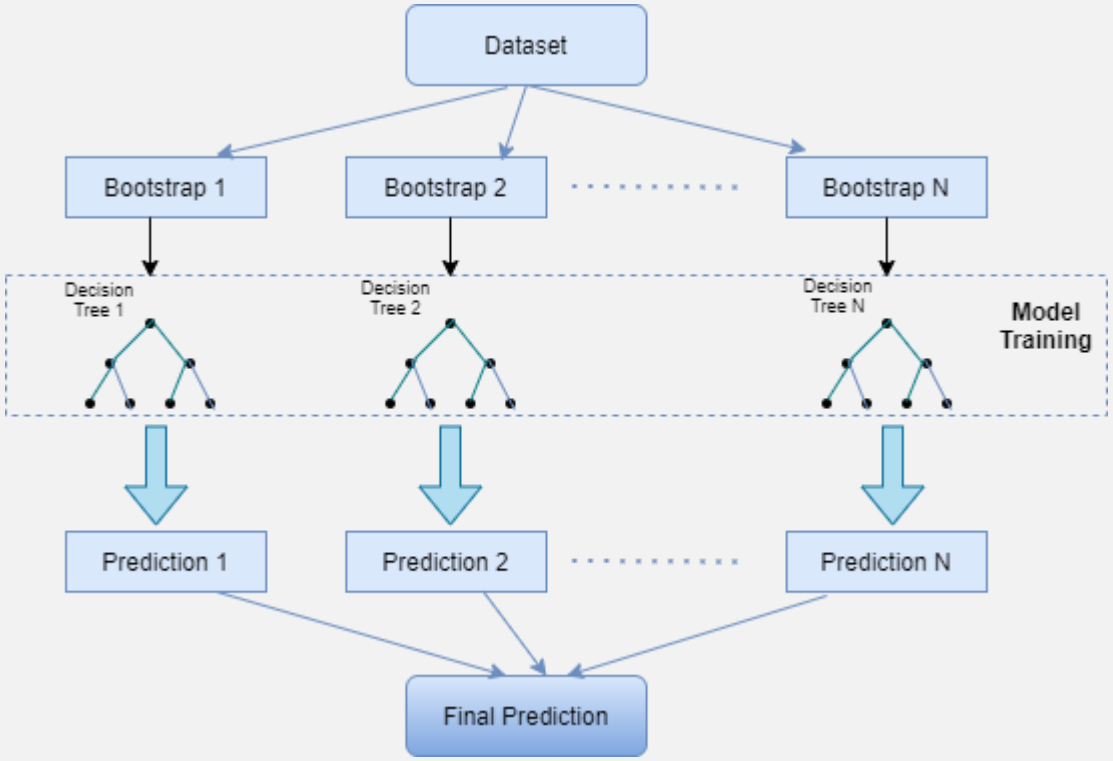
Humberto Loaiza Correa
Humberto.Loaiza@correounivalle.edu.co

Random Forest

- **Introducción**
  - Random Forests es una **extensión** de algoritmo derivado de **Bagging**.
  - Bagging es una técnica para **reducir la varianza** de una función de **predicción** estimada.
  - El Bagging **funciona bien** para procedimientos de **alta varianza** y **bajo sesgo**, como los árboles.
  - Para **regresión**, se ajusta el mismo **árbol** de regresión muchas veces con **versiones muestreadas** de los datos de entrenamiento y se **promedia** el **resultado**.
  - Para **clasificación**, un **comité de árboles** emite cada uno un **voto** para la clase **predicha**.

Random Forest

- **Introducción…**
  - La **diferencia** con el Bagging es la incorporación de la **selección aleatoria** de **características** para obtener una gran colección de árboles **descorrelacionados**.
  - Durante la **construcción** de un **árbol de decisión**, en cada paso de selección de división (split), RF primero escoge **aleatoriamente** un subconjunto de características y luego realiza el procedimiento convencional de selección de **división** con el subconjunto de características escogido.



Random Forest

- **Introducción…**
  - Los **árboles de decisión** son candidatos **ideales** para el **Bagging**, ya que pueden **capturar** estructuras de **interacción complejas** en los **datos** y, si **crecen** lo suficiente, tienen un **sesgo** relativamente bajo.

Random Forest

- **Conceptualización**
  - Dado que los **árboles** son notoriamente **ruidosos**, se **benefician** enormemente del **promediado**.
  - Además, como cada árbol generado en el Bagging está **idénticamente distribuido** (i.d.), la **esperanza** de una media de B de tales **árboles** es la **misma** que la **esperanza** de cualquiera de ellos.
  - Esto significa que el **sesgo** de los árboles en el **Bagging** es el **mismo** que el de los árboles individuales (bootstrap), y que la única posibilidad de **mejora** es mediante la **reducción** de la **varianza**.

Random Forest

- **Conceptualización…**
  - Un **promedio** de B **variables** aleatorias **i.i.d**., cada una con varianza σ^2, tiene **varianza**:

$$\frac{1}{B}\sigma^2$$

  - Si las **variables** son solo **i.d**. (idénticamente distribuidas, pero no independientes) con **correlación positiva** ρ entre **pares** de árboles, la **varianza del promedio** es:

$$\rho\,\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

    - A medida que **aumenta** B, la **varianza** tiende a

$$\rho\,\sigma^2$$

    - La cual está determinada por el valor de la **correlación** ρ de los **pares de árboles** del bagging y **limita** las **ventajas** del **promediado**.

Random Forest

- **Conceptualización**

Random Forest for Regression or Classification

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

Random Forest

- **Conceptualización…**
  - Los **bosques aleatorios** buscan mejorar la **reducción** de la **varianza** del **Bagging reduciendo** la **correlación** entre los **árboles**, **sin aumentar** demasiado la **varianza**.
  - Esto se consigue con la selección **aleatoria** de las **variables** de entrada en el proceso de **crecimiento del árbol**.
  - Esto se incluye en el algoritmo como:
    - Para un cantidad total p de características (variables)
    - *Before each split, select* $m \leq p$ *of the input variables at random as candidates for splitting.*
  - Valores típicos de m son $\sqrt{p}$ o incluso tan bajos como 1.

Random Forest

- **Conceptualización …**
  - Después del proceso de crecimiento, cada uno de los B árboles de decisión queda definido en términos de variables de división, puntos de corte en cada nodo y valores de nodo terminal.
  - La reducción a m (características) **reduce** la **correlación** entre cualquier **par de árboles** en el ensemble, y por lo tanto **reduce** la **varianza** del promedio (regression)
- **Valores recomendados**:
  - **Clasificación**: valor por defecto de $m = \sqrt{p}$ y el **tamaño** mínimo de los **nodos** es **uno**.
  - **Regresión**: valor por defecto de $m = p/3$ y el **tamaño** mínimo de los **nodos** es **cinco**.
  - En la práctica, los mejores valores para estos parámetros dependen del problema, y deben tratarse como parámetros ajustables.

Random Forest

- **Muestras fuera de bolsa (Out of Bag Samples, OOB)**
  - Una característica importante de los bosques aleatorios es el uso de muestras fuera de bolsa (oob):
    - *Para cada observación* $z_i = (x_i, y_i)$, *construya su predictor de bosque aleatorio promediando sólo los árboles correspondientes a muestras bootstrap en las que* z_i *no aparece*.
  - La estimación de **error** con **oob** es casi idéntica a la obtenida mediante crossvalidación con k-fold.
    - Una vez que el error **oob** se estabiliza, se puede terminar el entrenamiento.

Random Forest

- **Ejemplo**:
  - Cancer_Classification_RandomForest.ipynb

- **Ejercicio en Clase**
  - Cancer_Classification_RandomForest.ipynb
    - Cambiar parámetros para mejorar la clasificación
  - Clasificar el dataset wine con Random Forest
    - https://scikit-learn.org/stable/datasets/toy_dataset.html

Random Forest

- **Importancia de Características**
  - A partir de los bosques aleatorios se pueden construir **ranking** de **características**.
  - La **importancia** atribuida a la **característica** de división es la **mejora** del **criterio** de división en cada división del árbol.
  - La **mejora en el criterio** de división de **cada característica** se **acumula** por separado en cada árbol del bosque.
  - La **importancia atribuida** a la **característica** de división es el **promedio** de la mejora (normalizado) en el criterio de división obtenida en **todos los árboles** del bosque.
  - Random Forest **considera todas las variables** para hacer el ranking.

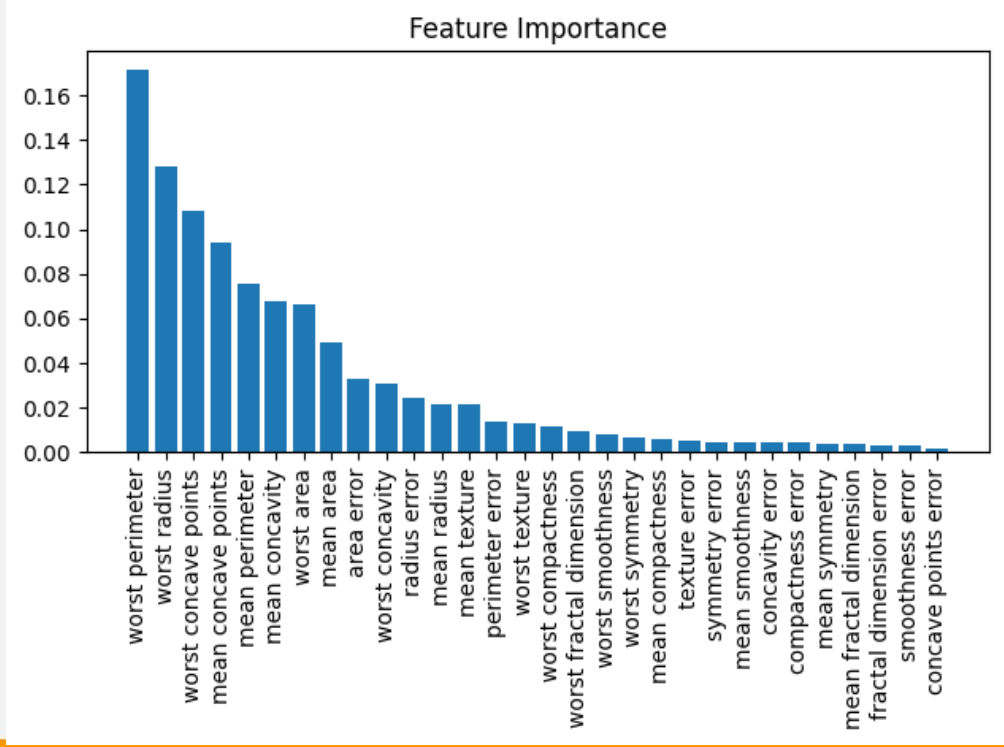
Random Forest

- **Importancia de Características…**
  - Cuando el número p de características es grande, pero la fracción de **variables relevantes** m es **pequeña**, es probable que los **bosques** aleatorios **no funcionen** bien.
    - En cada división de un árbol, se reduce la posibilidad de que se seleccionen las variables relevantes.
  - Cuando **aumenta** el número m de variables **relevantes**, el **rendimiento** de los **bosques** aleatorios es altamente **robusto** a un **aumento** del número de **variables con ruido**.
    - Esta robustez se debe en gran medida a la insensibilidad relativa del costo de clasificación errónea al sesgo y la varianza de las estimaciones de probabilidad en cada árbol.

Random Forest

- **Importancia de Características: Ejemplo**

  - Cancer_Classification_RandomForest_FeaturesSelection.ipynb




```
 1) worst perimeter                0.171088
 2) worst radius                   0.127990
 3) worst concave points           0.108371
 4) mean concave points            0.093999
 5) mean perimeter                 0.075574
 6) mean concavity                 0.067943
 7) worst area                     0.066153
 8) mean area                      0.049450
 9) area error                     0.032592
10) worst concavity                0.030939
11) radius error                   0.024119
12) mean radius                    0.021675
13) mean texture                   0.021239
14) perimeter error                0.013915
15) worst texture                  0.013366
16) worst compactness              0.011925
17) worst fractal dimension        0.009518
18) worst smoothness               0.008240
19) worst symmetry                 0.006817
20) mean compactness               0.006249
21) texture error                  0.005543
22) symmetry error                 0.004782
23) mean smoothness                0.004426
24) concavity error                0.004219
25) compactness error              0.004195
26) mean symmetry                  0.003522
27) mean fractal dimension         0.003517
28) fractal dimension error        0.003472
29) smoothness error               0.003408
30) concave points error           0.001753
```

Random Forest

- **Importancia de Características**
  - **Ejercicio en Clase**
    - Aplique técnicas de normalización de datos y remoción de outliers al dataset cáncer y compare las características seleccionadas por el algoritmo **Random Forest**.
    - Cancer_Classification_RandomForest_FeaturesSelection.ipynb

Universidad del Valle

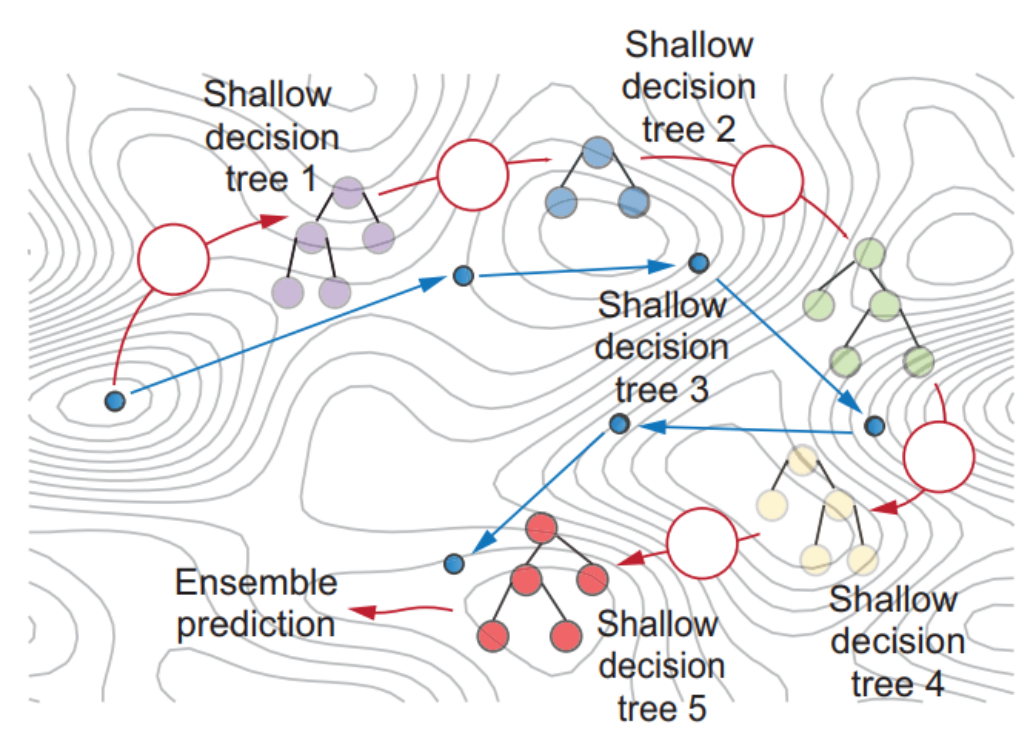
Secuencial Ensembles

Humberto Loaiza Correa
Humberto.Loaiza@correounivalle.edu.co

SEQUENTIAL ADAPTIVE BOOSTING ENSEMBLES

Use many weak learners, or simple models, trained in a stage-wise, sequential manner. Each successive model is trained to fix the mistakes made by the previously trained model, allowing the ensemble to adapt during training. The predictions of a large number of weak models are boosted into a strong model!

Ensembles in this family: AdaBoost, LogitBoost



Secuencial Ensembles

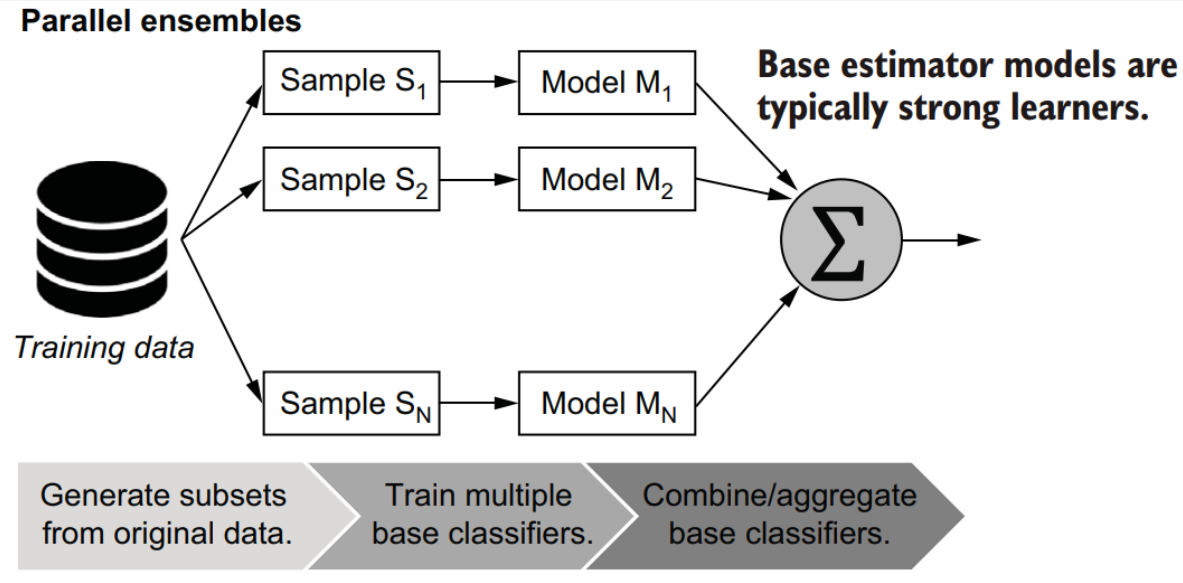
- **Introducción**
  - Los métodos de ensemble secuenciales aprovechan la **dependencia** de los **estimadores** (máquinas, aprendices, modelos,..) **base**. (A diferencia de los ensembles en paralelo)
  - Durante el aprendizaje, se **entrena** un **nuevo modelo** base de manera que éste **minimice** el **error** en que incurrió el **modelo** base **anterior**.
  - El **Boosting** es un algoritmo secuencial que tiene como finalidad **potenciar** el **desempeño** de un conjunto de **modelos débiles**.
    - A diferencia de **Bagging**, que combinan modelos de base complejos o fuertes.
  - **Boosting** comúnmente se refiere a **AdaBoost**, o **adpatative boosting** (1995).
  - **Boosting** es **simple** de **implementar**, **computacionalmente eficiente** y puede utilizarse con una amplia **variedad** de **algoritmos** de **aprendizaje** base.

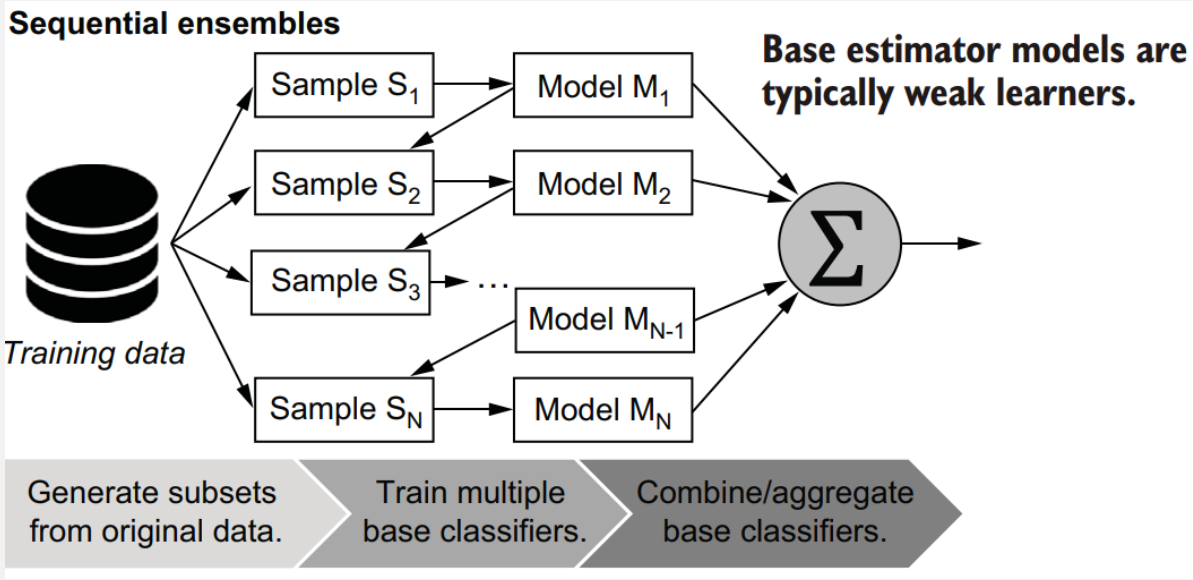
Secuencial Ensembles

- **Diferencias entre Ensembles Paralelos y Secuenciales**
  - **Entrenamiento**
    - **Paralelo:**
      - Entrenamiento de un modelo es **independiente** de los demás modelos base.
      - Típicamente emplea modelos base fuertes
    - **Secuencial:**
      - **Entrenamiento** de un modelo base **depende** del modelo base anterior.
      - Típicamente emplea **modelos base débiles** para **potenciarlos**.
        - Ejemplo: decisión de un nivel (decisión podada o truncada), árbol de decisión de un nivel.

Secuencial Ensembles

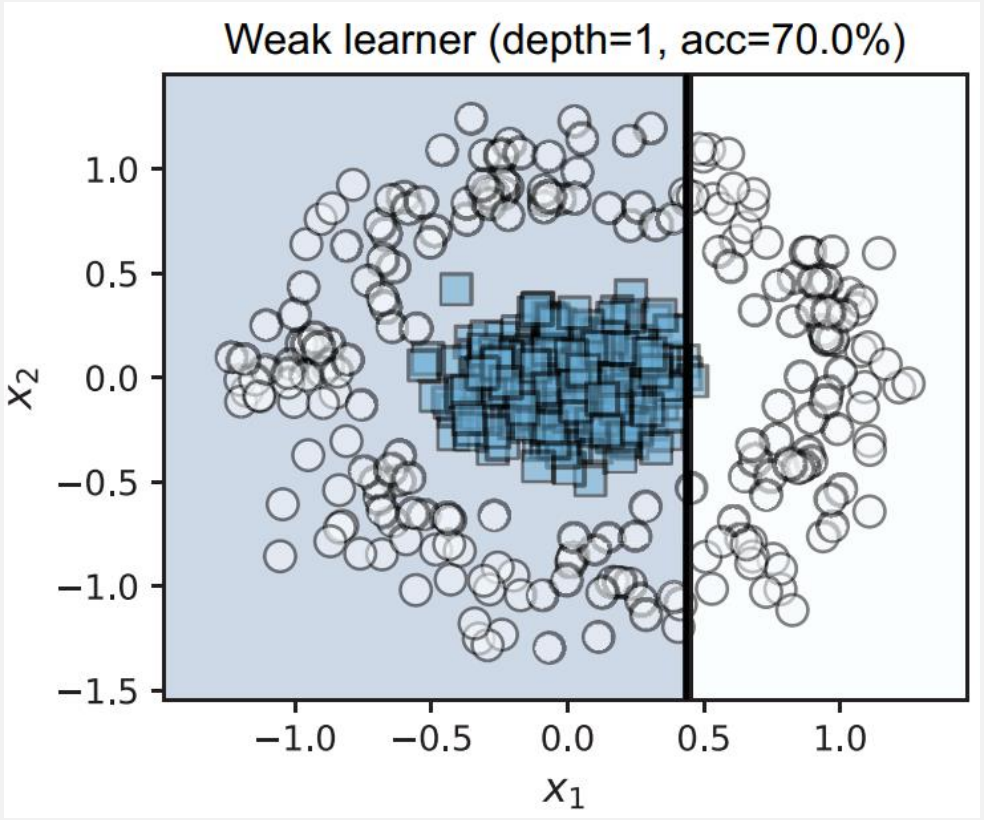
- **Diferencias entre Ensembles Paralelos y Secuenciales**

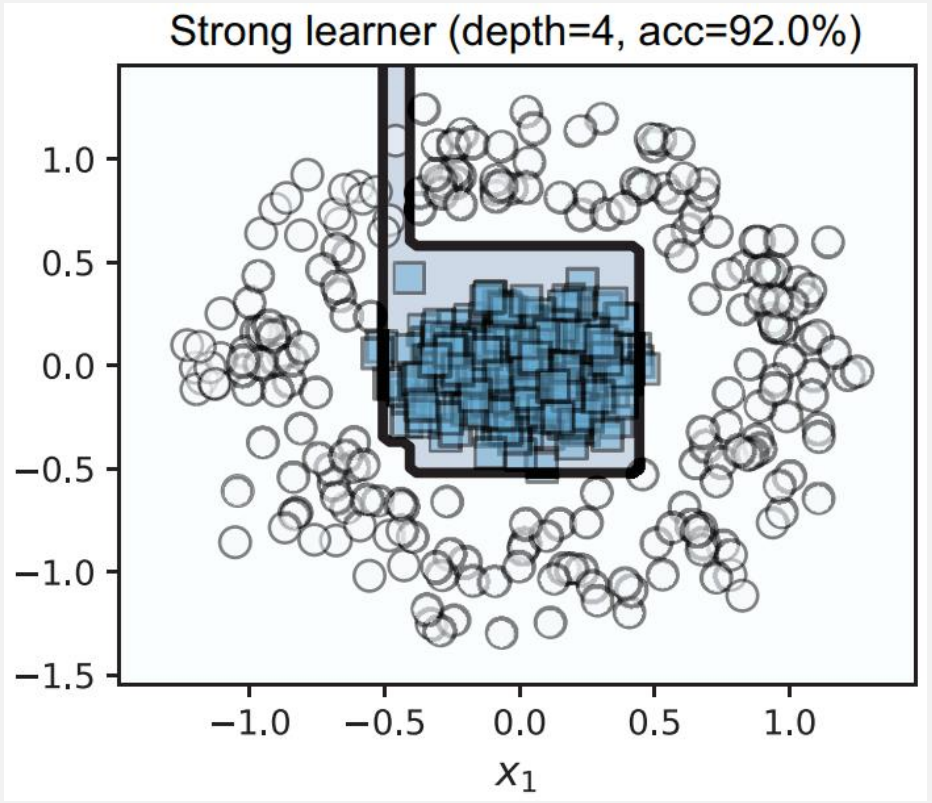




Secuencial Ensembles

- **Fronteras de decisión Modelo débil – Modelo Fuerte**





Secuencial Ensembles

- **Modelo Débil**
  - Los modelos de **decisión de un nivel** se usan **comúnmente** como aprendices **débiles** en métodos de conjuntos secuenciales como el **boosting**.
  - A medida que aumenta la **profundidad** del árbol, un modelo de **decisión de un nivel** se **convierte** en un **árbol de decisión**, transformándose en un clasificador más fuerte y su **desempeño** mejora.
  - Sin embargo, **no** es posible **aumentar arbitrariamente** el desempeño de los clasificadores, ya que comenzarán a **sobreajustarse (overfit)** durante el entrenamiento, lo que **disminuye** su **desempeño** de predicción cuando se implementan.

Universidad del Valle

AdaBoost: Adaptive Boosting

Humberto Loaiza Correa
Humberto.Loaiza@correounivalle.edu.co

AdaBoost: Adaptive Boosting

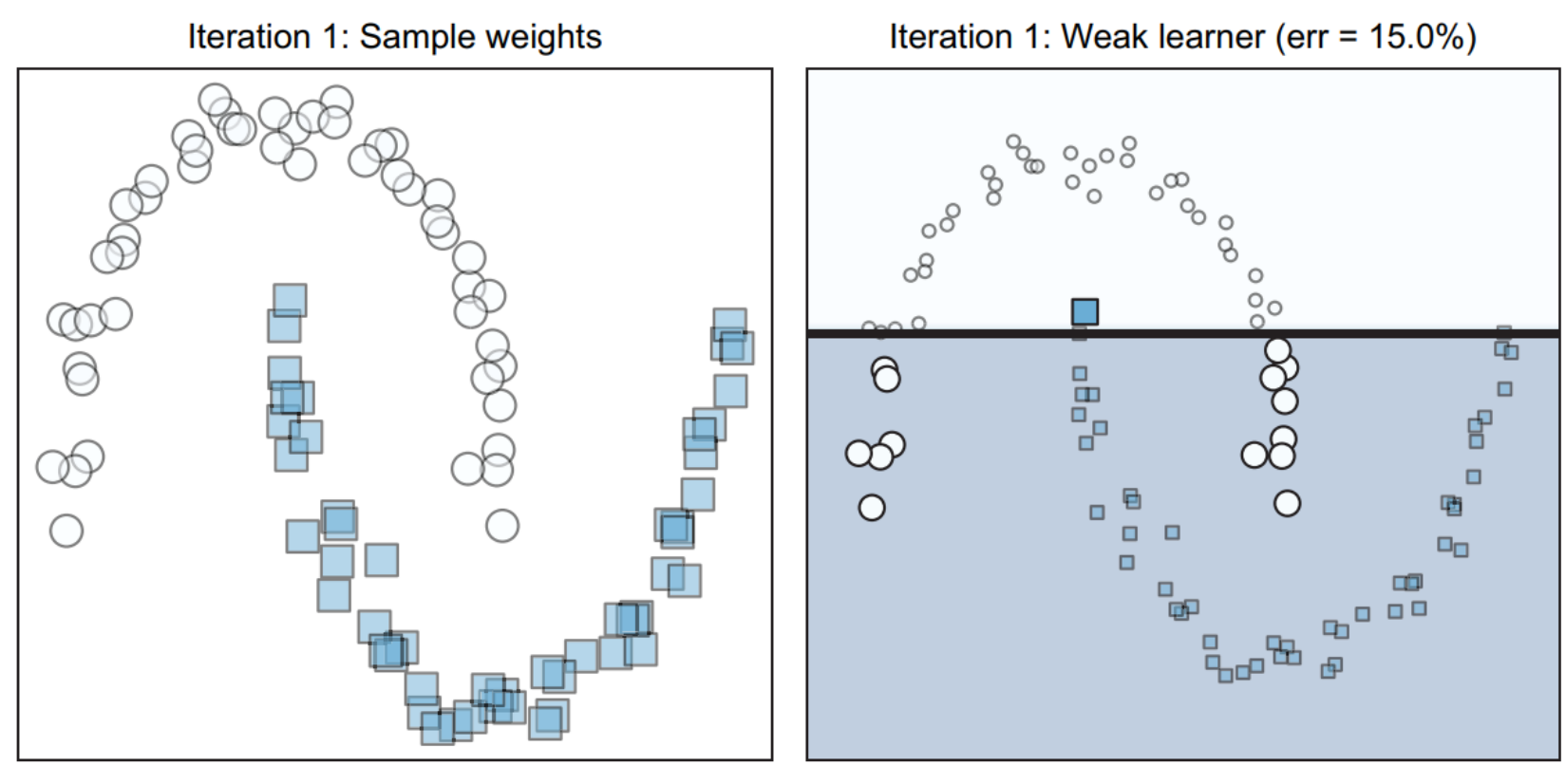
- **Introducción**
  - **AdaBoost** es un algoritmo **simpe** de **implementar** y **computacionalmente eficiente** de usar.
  - Siempre que el desempeño de cada **modelo débil** sea ligeramente **mejor** que el **azar**, el modelo final **converge** a un **modelo fuerte**.
  - AdaBoost es un algoritmo **adaptativo**: en cada **iteración**, **entrena** un nuevo **estimador base** que **corrige** los **errores** cometidos por el **estimador** base **anterior**.
    - Debe **garantizarse** que el **algoritmo de aprendizaje base priorice** los ejemplos de entrenamiento **mal clasificados**.
    - Esto lo hace **asignando pesos** a los **ejemplos** de **entrenamiento individuales**.
      - Ejemplos mal clasificados: mayor peso
      - Ejemplos bien clasificados: menor peso

AdaBoost: Adaptive Boosting

- **Introducción…**
  - Cuando se **entrena secuencialmente** el siguiente estimador base, los **pesos permiten** que el algoritmo de aprendizaje **priorice** (y posiblemente corrija) los **errores** de la iteración **anterior**.
    - Este es el **componente adaptativo** de AdaBoost, que **conduce** a un **poderoso** conjunto.
    - La **actualización** de los **pesos** de las muestras en cada iteración contribuye a la diversidad del ensemble.
  - **Cada modelo** base es **diferente** y genera una frontera de **decisión distinta** debido a que se entrena con el **mismo dataset** pero con **pesos diferentes** en cada iteración.

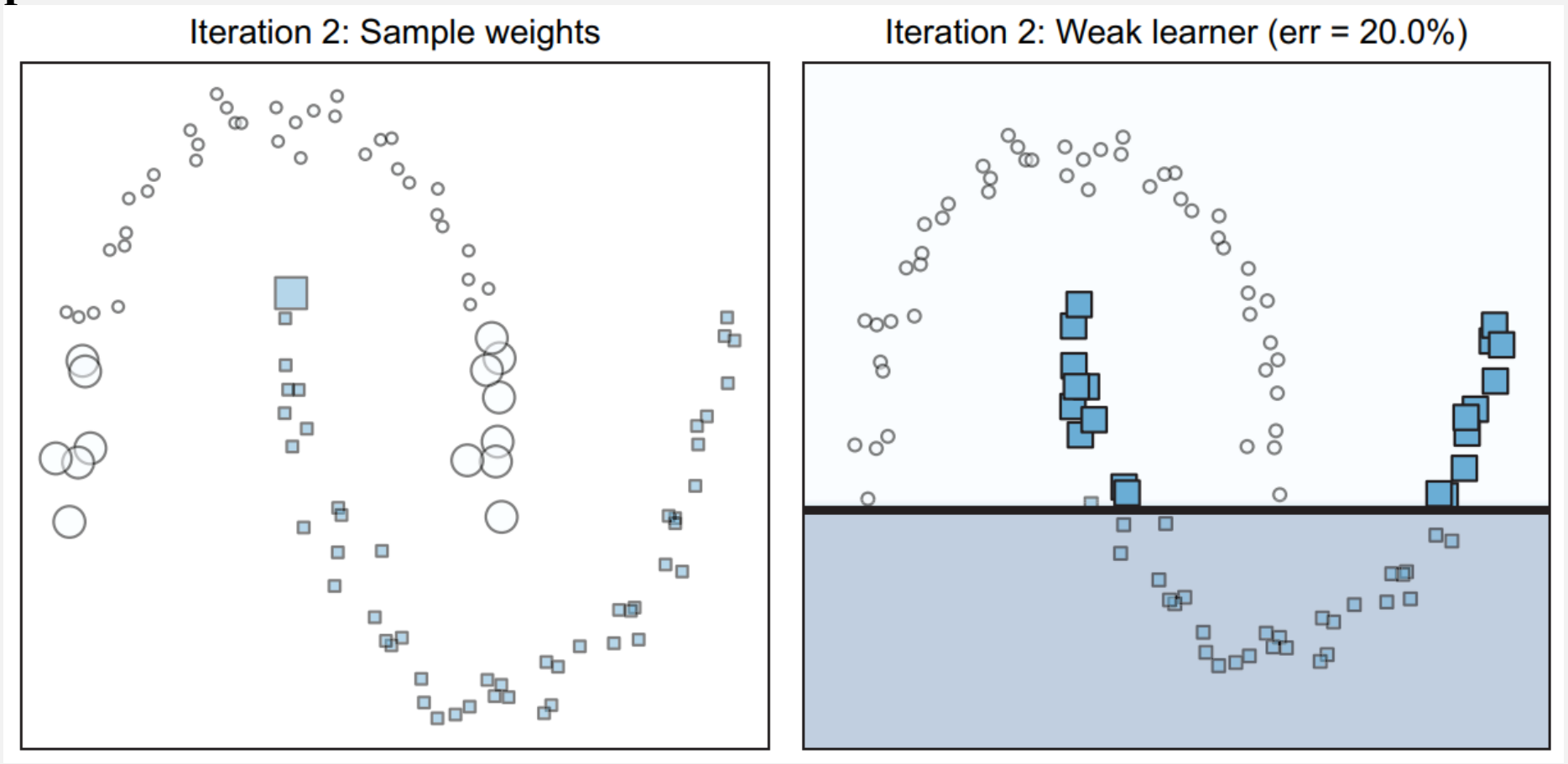
AdaBoost: Adaptive Boosting

- **Ejemplo Ilustrativo**



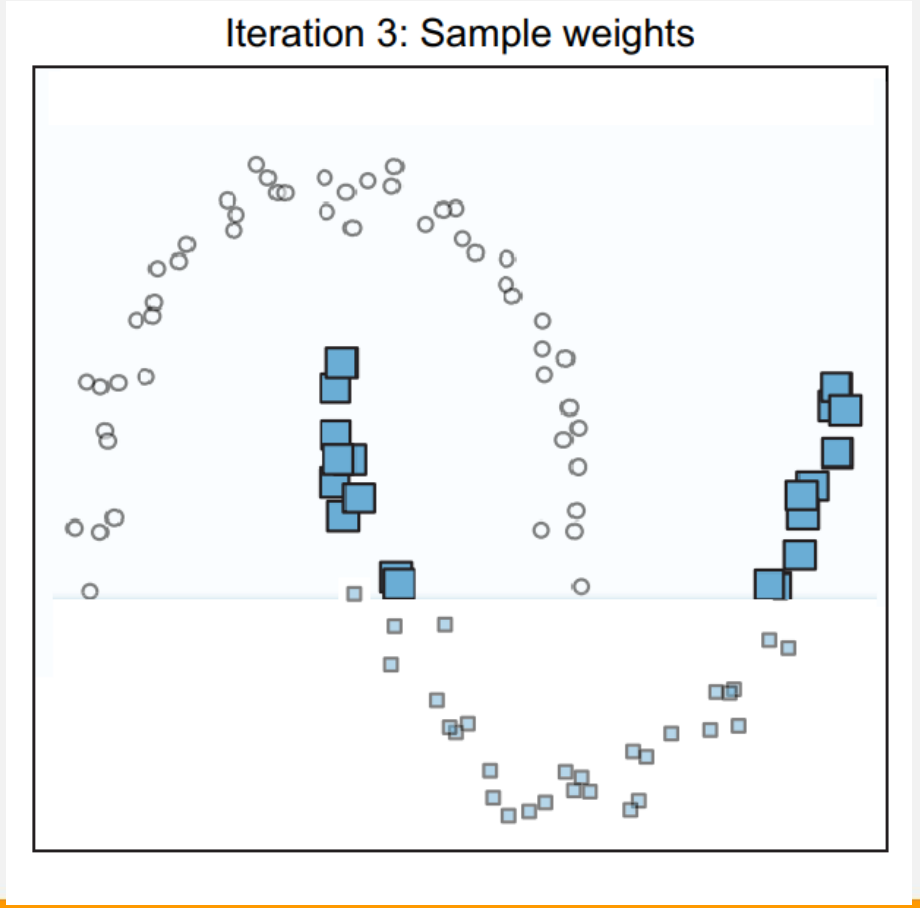
AdaBoost: Adaptive Boosting

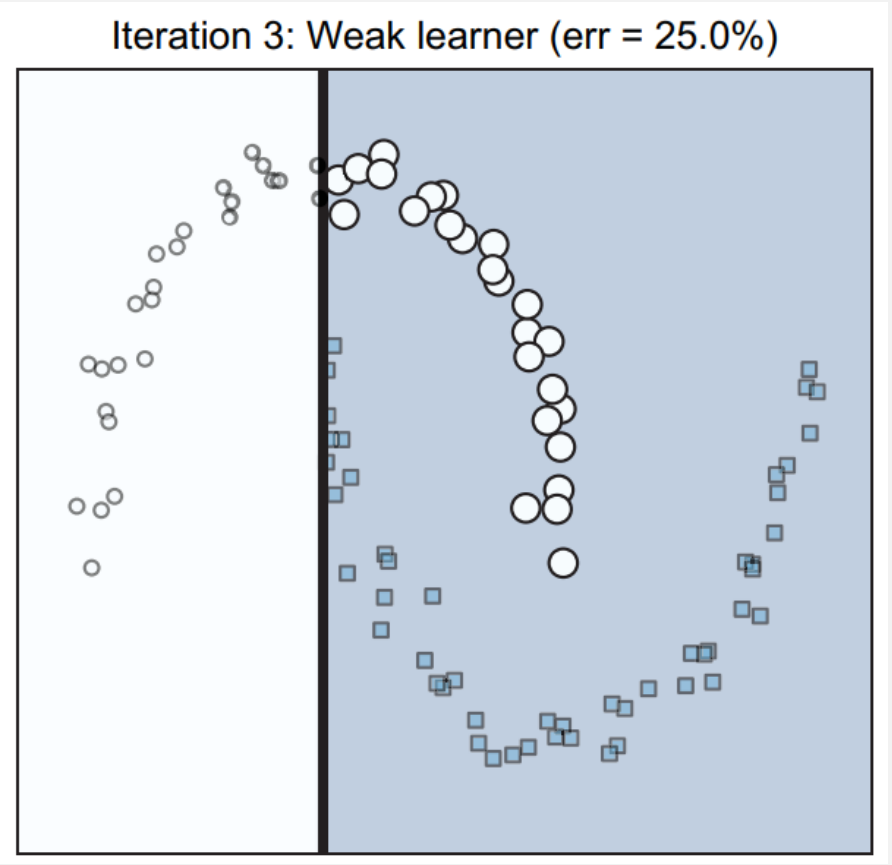
- **Ejemplo Ilustrativo**



AdaBoost: Adaptive Boosting

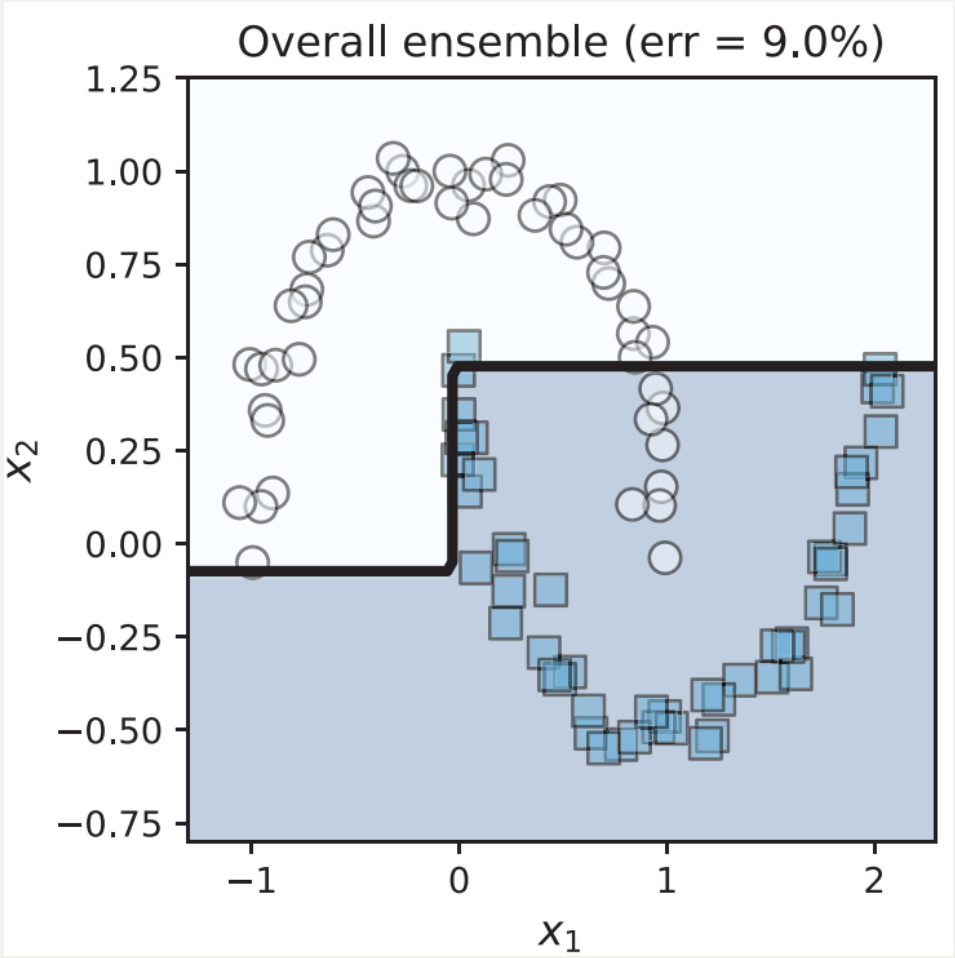
- **Ejemplo Ilustrativo**





AdaBoost: Adaptive Boosting

- **Ejemplo Ilustrativo**



AdaBoost: Adaptive Boosting

- **Algoritmo**

Algorithm The Adaboost algorithm.

1. Initialise the weights $w_i = 1/n$, $i = 1, \ldots, n$.
2. For $t = 1, \ldots, T$, (T is the number of boosting rounds)
   (a) Construct a classifier $\eta_t(\boldsymbol{x})$ from the training data with weights w_i, $i = 1, \ldots, n$.
   (b) Calculate e_t as the sum of the weights w_i corresponding to misclassified patterns.
   (c) If $e_t > 0.5$ or $e_t = 0$ then terminate the procedure, otherwise set $w_i = w_i(1 - e_t)/e_t$ for the misclassified patterns and renormalise the weights so that they sum to unity.
   (d) If $e_t < 0.5$ set $w_i = w_i(1 - e_t)/e_t$ for misclassified patterns and set
   $w_i = w_i\, e_t/(1 - e_t)$ for well-classified patterns.
   Renormalise the weigths so that they sum to unity.
3. For a two-class classifier, in which $h_t(\boldsymbol{x}) = 1$ implies $\boldsymbol{x} \in \omega_1$ and $h_t(\boldsymbol{x}) = -1$ implies $\boldsymbol{x} \in \omega_2$, form a weighted sum of the classifiers, $h_t(\boldsymbol{x})$

$$H(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t \; h_t(\boldsymbol{x}) \qquad \alpha_t = \frac{1}{2}\, log\left(\frac{1-e_t}{e_t}\right)$$

and assign $\boldsymbol{x}$ to ω_1 if $H(\boldsymbol{x}) > 0$

AdaBoost: Adaptive Boosting

- **Algoritmo**
  - Cada **modelo base** (débil) tiene un **peso** α_t que depende de su **error de entrenamiento** e_t.
    - e_t es una medida simple e inmediata del modelo base.
  - El peso α_t de cada modelo base $h_t(\boldsymbol{x})$ se calcula a partir de su error de entrenamiento e_t como:

$$\alpha_t = \frac{1}{2} \, log\left(\frac{1 - e_t}{e_t}\right)$$

  - La ecuación permite asignar **mayor peso** a los estimadores base de **mejor desempeño** (menos errores) para que su contribución a la **predicción** del **conjunto** $H(\boldsymbol{x})$ sea **mayor**.

$$H(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t \; h_t(\boldsymbol{x})$$

AdaBoost: Adaptive Boosting

- **Outliers**
  - AdaBoost es **muy susceptible** a los **valores atípicos**.
  - Los valores **atípicos** suelen ser **clasificados erróneamente** por aprendices débiles.
  - AdaBoost **aumenta** el **peso** de los ejemplos **mal clasificados**, por lo que el **peso asignado** a los valores atípicos sigue **aumentando**.
  - Cuando se entrena el siguiente aprendiz débil, éste hace una de las siguientes cosas:
    - **Continúa** clasificando **erróneamente** el valor atípico, en cuyo caso AdaBoost aumentará aún más su peso, lo que a su vez provoca que los **aprendices débiles sucesivos** clasifiquen **erróneamente**, fallen y sigan **aumentando** su **peso**.
    - **Clasifica correctamente** el valor atípico, en cuyo caso AdaBoost puede sobreajustar los datos (overfitting).

AdaBoost: Adaptive Boosting

- **Outliers…**
  - Los valores atípicos obligan a AdaBoost a dedicar un esfuerzo desproporcionado a los ejemplos de entrenamiento con ruido.
  - Los valores atípicos tienden a confundir a AdaBoost y lo hacen menos robusto.

AdaBoost: Adaptive Boosting

- **Tasa de aprendizaje (Learning rate)**

  - El primer hiperparámetro a ajustar en un AdaBoost es la **tasa de aprendizaje** η, la cual **ajusta** la **contribución** de **cada estimador** al conjunto.
    - Por ejemplo, **una tasa de aprendizaje** de 0,75 indica a AdaBoost que **disminuya** la contribución global de cada estimador base a un factor de 0,75.
  - La **influencia** negativa de los **valores atípicos** crece rápidamente con **valores altos** de la **tasa de aprendizaje**, lo que puede **anular** el **desempeño** del modelo.
    - Por lo tanto, una forma de **mitigar el efecto** de los valores atípicos **es reducir la tasa de aprendizaje**.

AdaBoost: Adaptive Boosting

- **Tasa de aprendizaje (Learning rate) …**

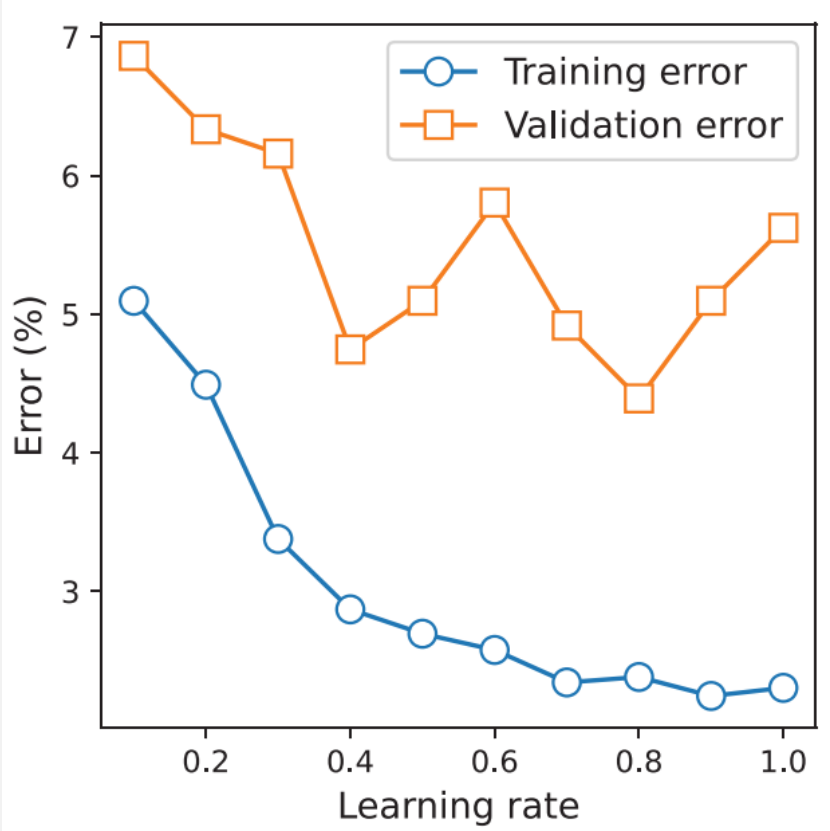
  - **Reducir** la **tasa de aprendizaje** η, **contrae** (shrinkage) al **aporte** de cada **estimador** base.
    - Constituye una forma de **regularización** del modelo para **minimizar** el **overfitting**.
    - En cada iteración t, el ensemble F_t se actualiza a F_{t+1} como:

$$F_{t+1}(\boldsymbol{x}) = F_t(\boldsymbol{x}) + \eta\, \alpha_t\, h_t(\boldsymbol{x})$$

    - Siendo α_t el peso de cada aprendiz base y $0 < \eta \leq 1$
    - Valores bajos de η implican:
      - Más iteraciones, más aprendices base para contruir un ensemble efectivo, y mejor generalización.
    - Una forma efectiva de seleccionar η es mediante validación cruzada.

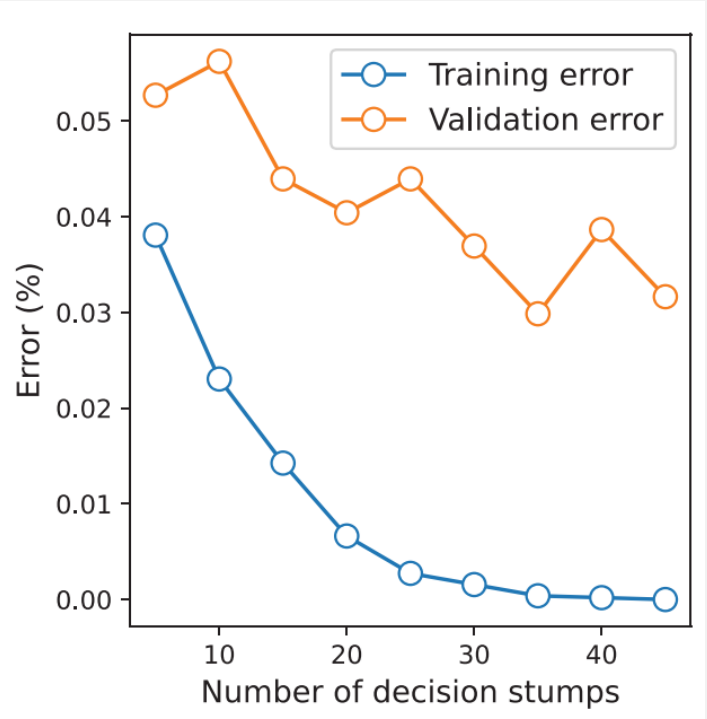
AdaBoost: Adaptive Boosting

- **Tasa de aprendizaje (Learning rate) …**



AdaBoost: Adaptive Boosting

- **Parada Temprana (Early Stopping)**
  - Identificar el **número mínimo** de **estimadores** base requeridos para construir un ensemble efectivo también se conoce como **parada temprana**.
  - La parada temprana también se puede obtener de otras maneras, como por ejemplo alcanzando un desempeño preestablecido.
  - Un número bajo de aprendices base ayuda a controlar el overfitting y reducir el tiempo de entrenamiento.



AdaBoost: Adaptive Boosting

- **Poda (Pruning)**
  - Efecto de reducir la complejidad del ensamble mediante la limitación de los elementos que lo constituyen.
  - Pre-poda: limitaciones establecidas antes del entrenamiento.
    - Ej. Limitar el máximo número de aprendices.
  - Post-poda: limitaciones establecidas después del entrenamiento.
    - Ej. Eliminar todos los aprendices con un peso menor que un umbral determinado.

AdaBoost: Adaptive Boosting

- **Ejemplo 1**:
  - Cancer_Classification_AdaBoost.ipynb
  - Evaluar el desempeño para n_estimators desde 1 hasta 60 con incrementos de 3
- **Ejercicio en Clase**
  - Cancer_Classification_AdaBoost.ipynb
    - Cambiar el parámetro: algorithm: {'SAMME', 'SAMME.R'}, *default='SAMME.R'*
    - Cambiar el parámetro: learning_rate: {0.2 0.5 0.7 1}, *default=1.0}*
    - Incluir eliminación de Outliers
  - Clasificar el dataset wine con AdaBoost
    - https://scikit-learn.org/stable/datasets/toy_dataset.html

AdaBoost: Adaptive Boosting

- **Ejemplo 2**:
  - Encontrar el número mínimo de estimadores
  - Encontrar el mejor valor de la tasa de aprendizaje por validación cruzada
  - Cancer_Classification_AdaBoost_Paramet.ipynb

- **Ejercicio en Clase**
  - Cancer_Classification_AdaBoost_Paramet.ipynb
    - Obtener un nuevo clasificador AdaBoost con los mejores parámetros encontrados
    - Cambiar el parámetro: algorithm: {'SAMME', 'SAMME.R'}, *default='SAMME.R'*
    - Incluir eliminación de Outliers