HENRY

Modelos de ensambles

Data Science

Agenda

- Estimación de grandes números
- Modelos de Ensambles
- Bagging
- Random Forest
- Boosting
- XG Boost, Extreme Gradient Boosting
- Bagging vs. Boosting
- Stacking
- Voting Classifier

OBJETIVOS DE LA CLASE

Al finalizar esta lecture estarás en la capacidad de...

- ➜ Reconocer el concepto de Estimación de grandes números
- ➜ Aplicar los principales Modelos de Ensambles

Al **finalizar** cada uno de los temas, tendremos un **espacio de consultas**.

Hay un **mentor** asignado para responder el **Q&A**.

¡Pregunta, pregunta, pregunta! :D

Modelos de **ensambles**

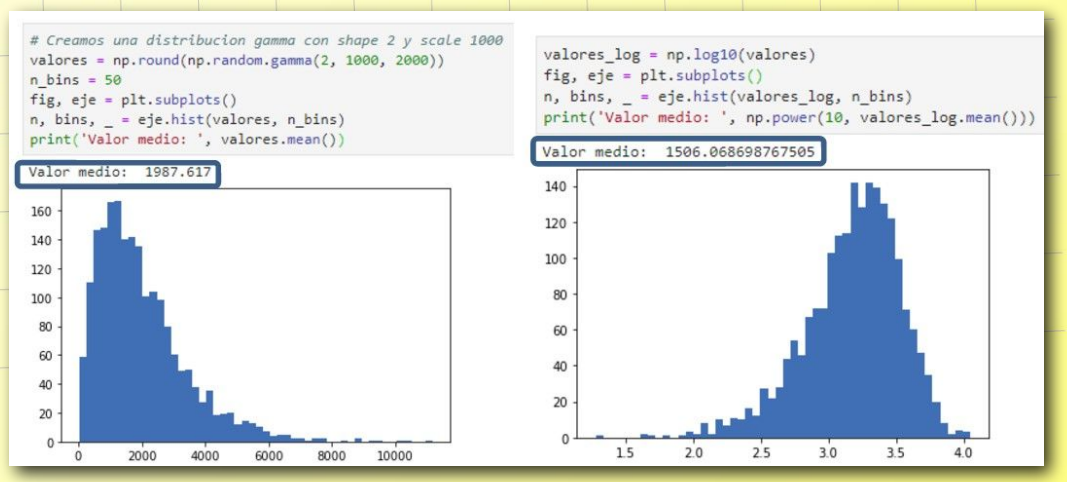
Estimación de grandes números

Con frecuencia nos encontramos con la necesidad de hacer una estimación, aún contando con poca información concreta del problema.

Si bien se puede tener un cierto conocimiento del dominio, cuando hay pocos datos disponibles, una estimación aceptable es una tarea difícil.

Estimación de grandes números

Si en lugar de usar el dato tal y como está, usamos la potencia de diez que se acerca al valor de ese dato, es decir, lo llevamos a escala logarítmica, nos encontramos con el concepto de media geométrica y en esa escala podemos tener un valor más adecuado.



Estimación de grandes números

- Se conoce como sabiduría de las masas al hecho de que juntando las estimaciones de muchas personas, en el resultado global haya una aproximación adecuada a la realidad.
- En Machine Learning este concepto se aplica en metodologías conocidas como Ensambles (Random forest, boosting, bagging), que consisten en juntar el resultado de muchos estimadores débiles, para aportar un resultado final óptimo.

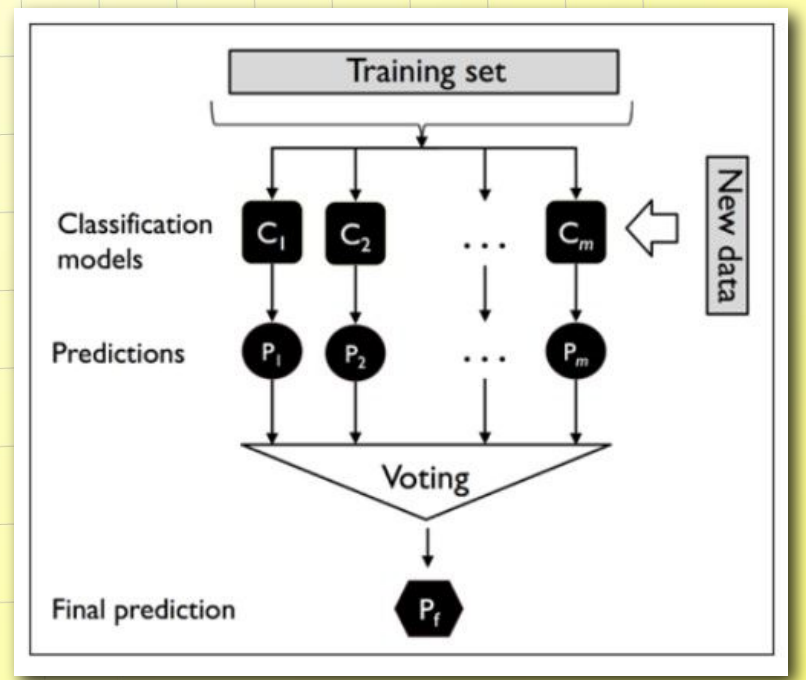
Ensambles

¿Qué es?

Consiste en entrenar muchos modelos y quedarse con el de mejor rendimiento, es decir, el que mejor clasifique.

Sin embargo, si todos los modelos son muy parecidos, no van a agregar mucha información nueva en la votación.

Se necesitan modelos diferentes entre sí, poco correlacionados.



¿Qué es?

Los modelos pueden ser diferentes entre sí por una variedad de razones:

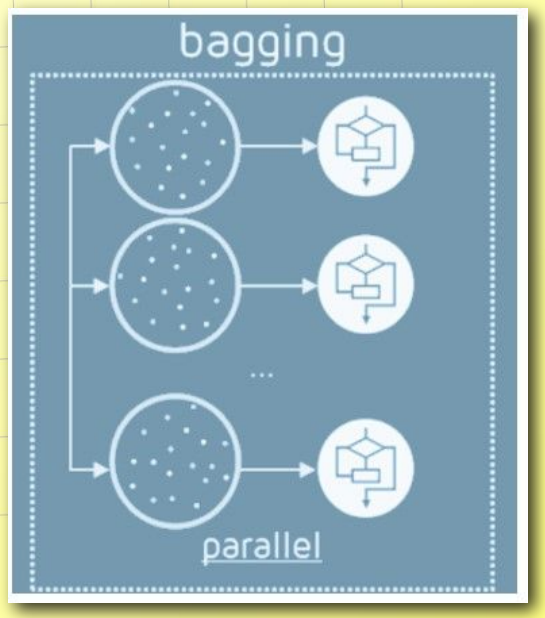
- ➜ Puede haber diferencia en la población de datos.
- ➜ Puede haber una técnica de modelado utilizada diferente.
- ➜ Puede haber una hipótesis diferente.

Veremos distintos tipos de ensambles: Bagging, Random Forest, Boosting, XG Boost, Stacking, Voting Classifier.

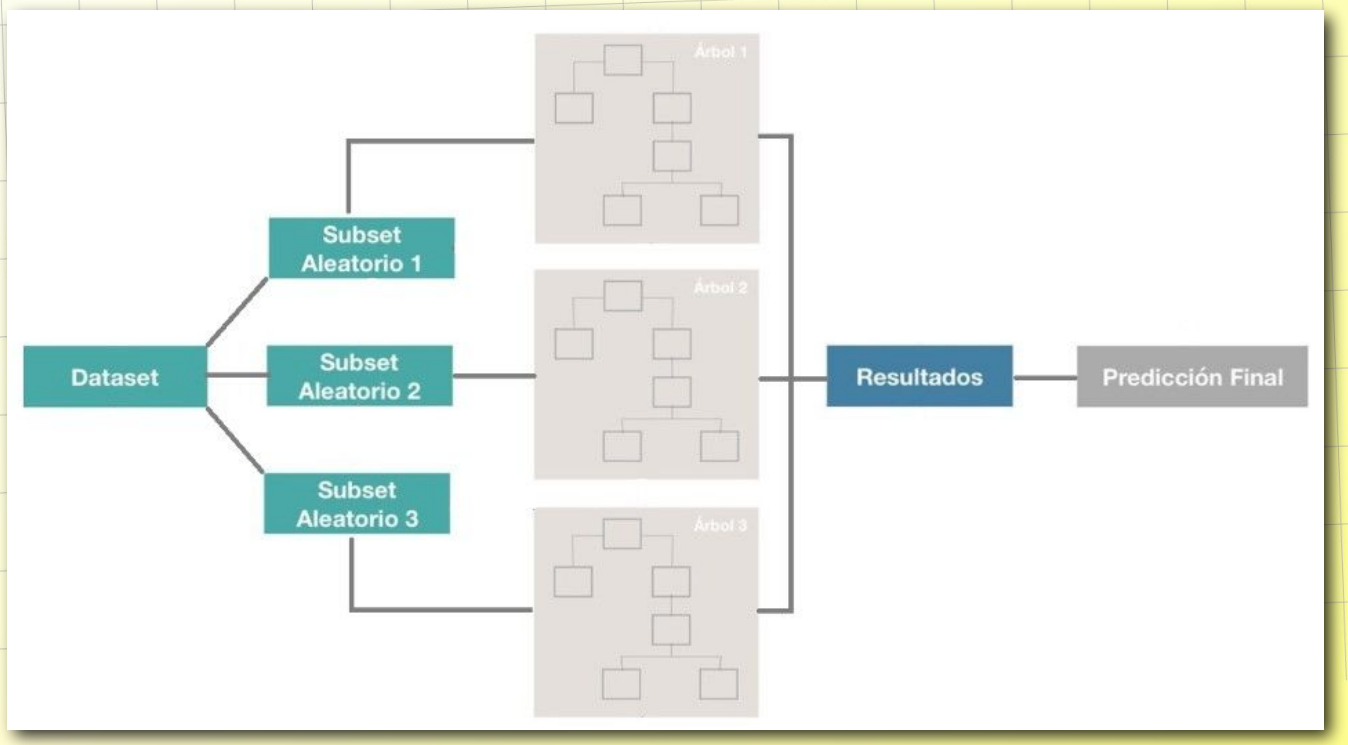
Bagging

También se conoce como *Agregación Bootstrap* (Muestreo con reemplazo de las instancias).

1. Dada una muestra de datos, se extraen varias muestras (bootstrapped) de manera aleatoria.
2. Una vez que forman las muestras bootstrapped, se entrenan los modelos de manera separada.
3. La predicción de salida final se combina en las proyecciones de todos los submodelos.



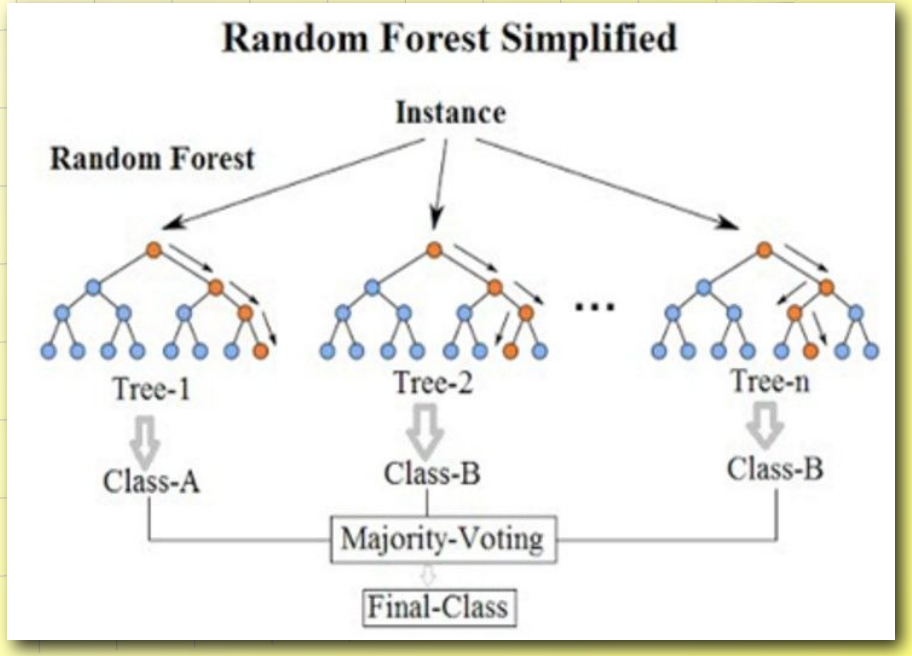
Bagging


Árbol 1
Subset Aleatorio 1
Árbol 2
Dataset
Subset Aleatorio 2
Resultados
Predicción Final
Subset Aleatorio 3
Árbol 3

Random Forest

El árbol de decisión con la profundidad suficiente hace overfitting.

Para evitar esto, se crean muchos árboles para que trabajen en conjunto, la salida de cada uno se contará como *"un voto"* y la opción más votada será la respuesta del *"Bosque Aleatorio"*.



Random Forest

FUNCIONAMIENTO

Se considera sólo un subconjunto de m atributos elegidos al azar para cada subconjunto.

1. Se seleccionan k features de las m totales (siendo k menor a m) y se crea un árbol de decisión con esas k features.
2. Se crean n árboles variando siempre la cantidad de k features
3. Se guarda el resultado de cada árbol obteniendo n salidas.
4. Se calculan los votos obtenidos para cada "clase" seleccionada y se considera a la más votada como la clasificación final del "bosque".

Random Forest

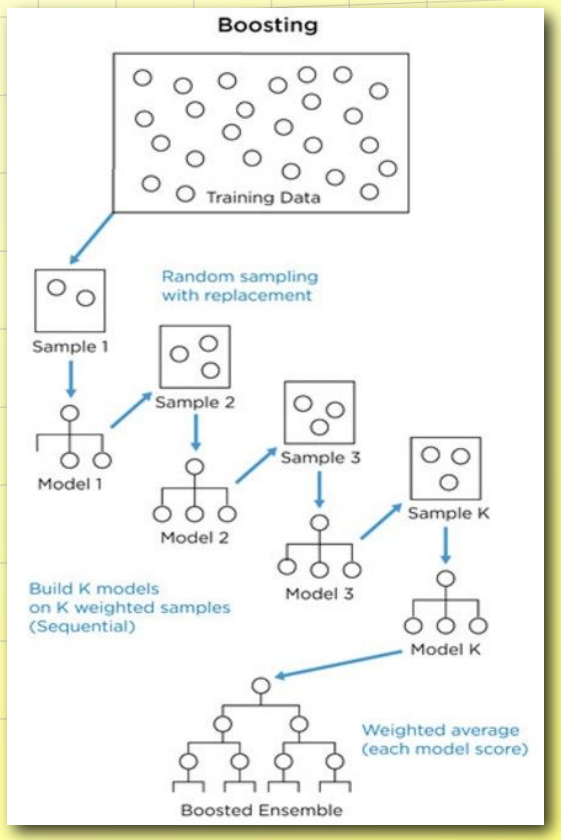
CONCLUSIONES

- Random Forest es robusto frente a outliers y ruido.
- Provee buenos estimadores de error (oob_score) e importancia de variables
- Entrenar muchos árboles puede llevar mucho tiempo, pero es fácilmente paralelizable.
- No funciona bien con conjuntos pequeños de datos.

Boosting

Se entrena una secuencia de modelos donde se da más peso a los ejemplos que fueron clasificados erróneamente por iteraciones anteriores.

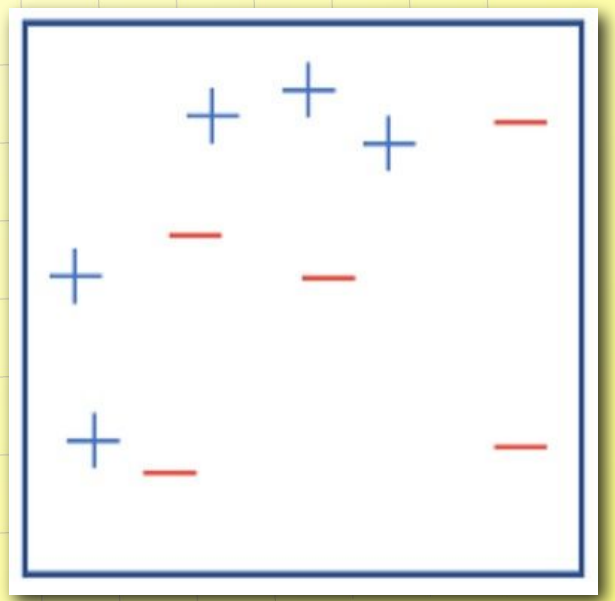
Al igual que con bagging, las tareas de clasificación se resuelven con una mayoría ponderada de votos, y las tareas de regresión se resuelven con una suma ponderada para producir la predicción final.



Boosting

EJEMPLO (Positivos y negativos)

Se plantea un problema de clasificación binaria con 10 elementos de entrenamiento, 5 positivos y 5 negativos: El algoritmo va a iterar hasta lograr una separación aceptable de las clases...



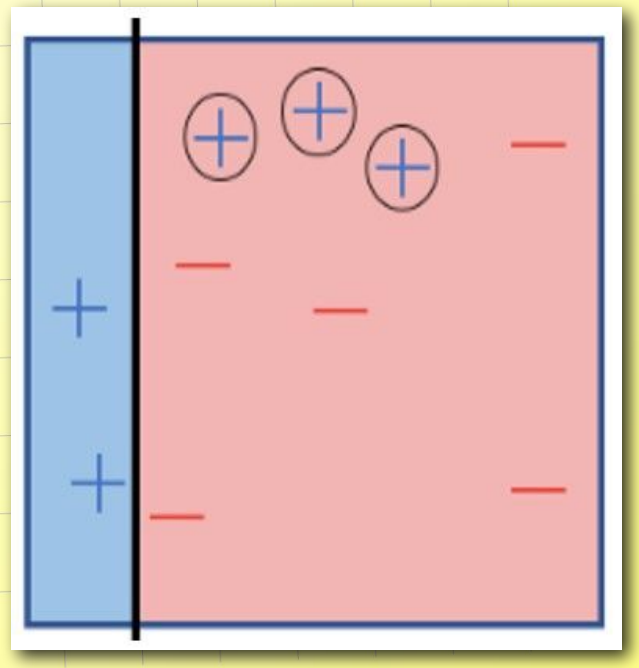
Boosting

EJEMPLO (Positivos y negativos)

El primer clasificador débil, genera una recta vertical.

A la derecha de la recta, se considera que todos los ejemplos son negativos, mientras que a la izquierda son positivos.

La recta clasifica mal a tres positivos.

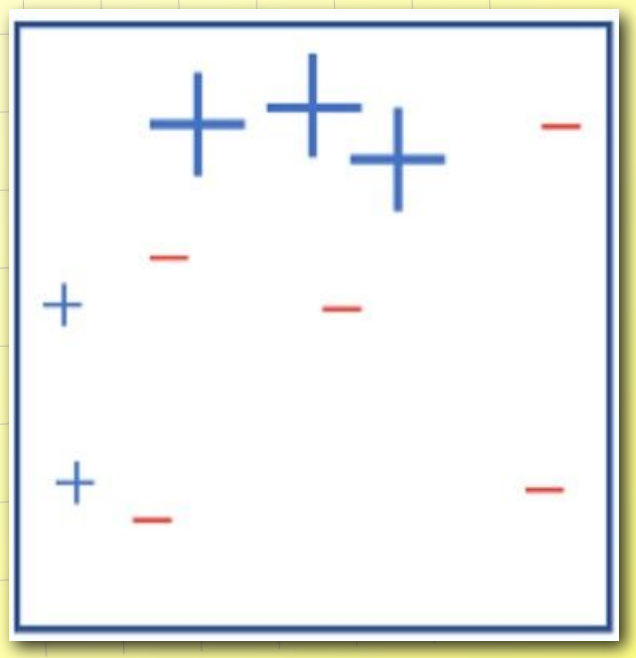


Boosting

EJEMPLO (Positivos y negativos)

Ahora los tres ejemplos mal clasificados aparecen de un mayor tamaño que el resto de los ejemplos.

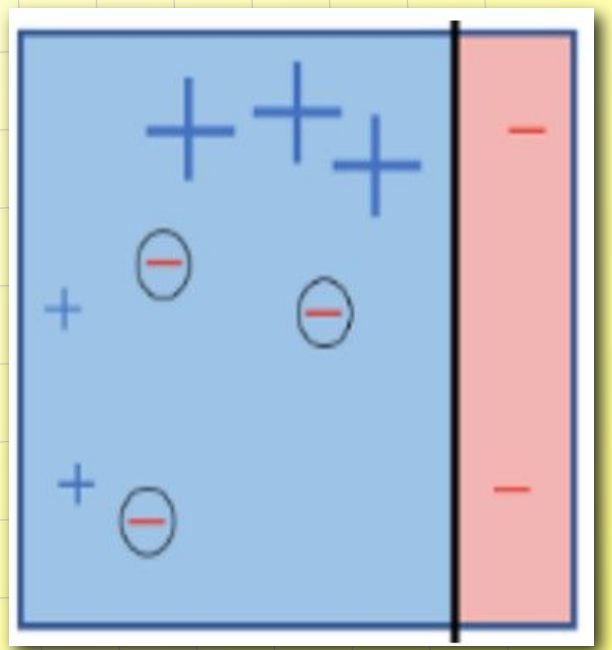
Esto simboliza que dichos ejemplos tendrán una mayor importancia al momento de seleccionar el clasificador débil de la segunda iteración.



Boosting

EJEMPLO (Positivos y negativos)

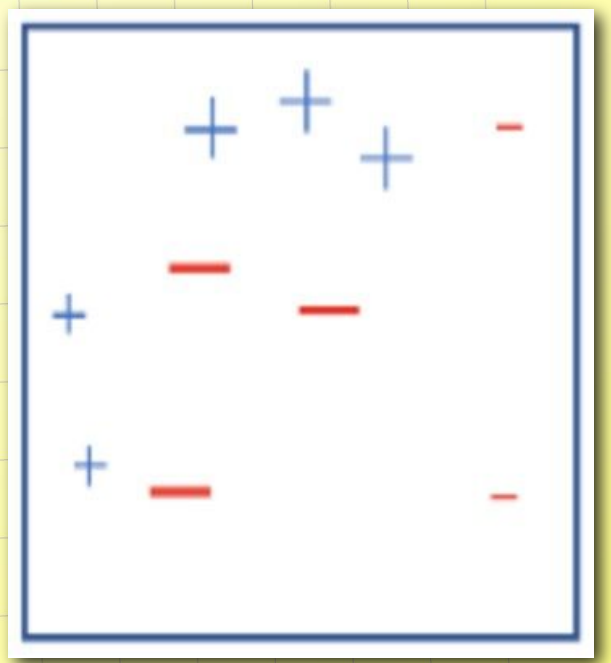
El segundo clasificador débil, es otra recta vertical colocada más hacia la derecha, se equivoca también en tres ejemplos, ya que clasifica mal ejemplos negativos.



Boosting

EJEMPLO (Positivos y negativos)

Para la tercera iteración los ejemplos negativos mal clasificados tienen ahora el mayor tamaño, es decir, tendrán mayor importancia en la siguiente iteración.

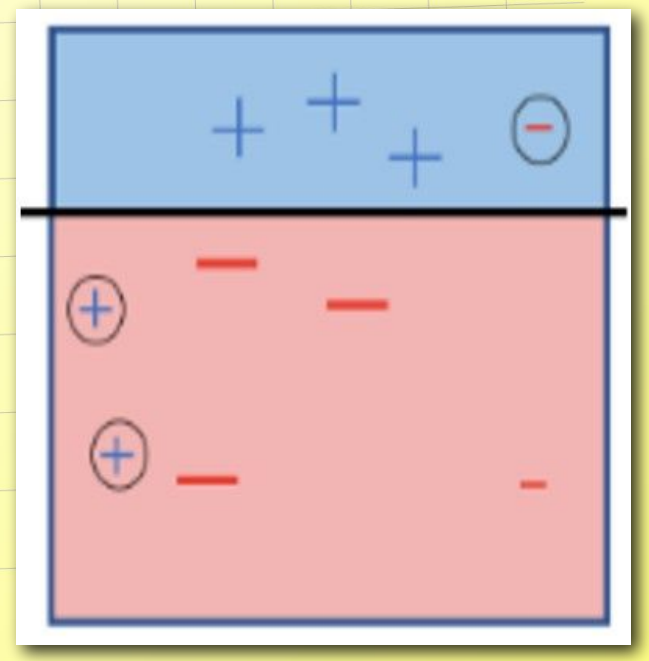


Boosting

EJEMPLO (Positivos y negativos)

En la tercera iteración el clasificador débil resultante es una recta horizontal, como se puede observar en el cuadro de la derecha.

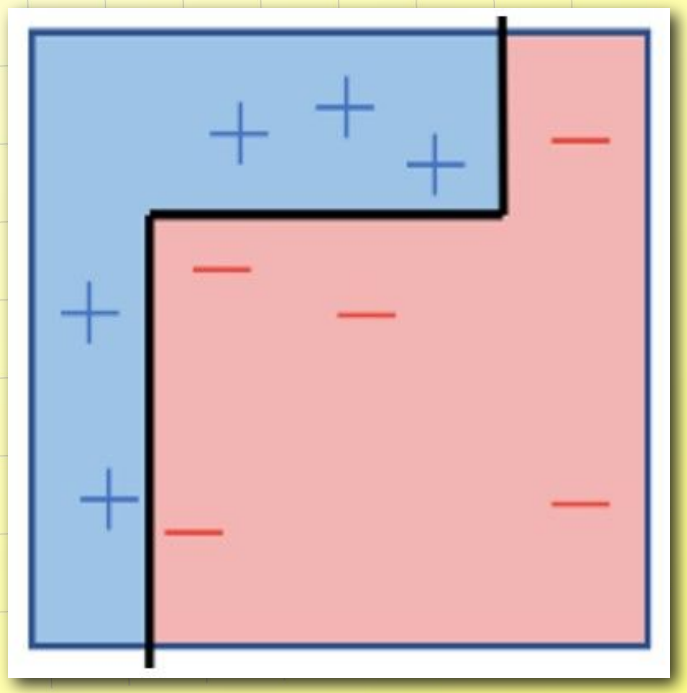
Este clasificador se equivoca en la clasificación de un ejemplo negativo y dos positivos, que de igual forma aparecen encerrados en un círculo.



Boosting

EJEMPLO (Positivos y negativos)

Finalmente, se ilustra el clasificador fuerte que resulta de crear un ensamble con tres clasificadores débiles.



XG Boost

XGBoost (*Extreme Gradient Boosting*) es un algoritmo que recientemente ha dominado el aprendizaje automático y sobre todo las competiciones de Kaggle (para datos estructurados).

Es una implementación de árboles de decisión potenciados por el algoritmo de descenso por gradiente, diseñado para aumentar la velocidad y mejorar el rendimiento.

XGBoost es una librería de software que se puede descargar e instalar y luego acceder desde una variedad de interfaces: CLI, C++, Python, R, Julia, etc.

No sólo tiene buena performance computacional, también posee un muy buen desempeño con el manejo de los datos.

XG Boost

Características principales:

- Paralelización de la construcción de árboles utilizando todos los núcleos de la CPU durante el entrenamiento.
- Computación distribuida para el entrenamiento de modelos muy grandes utilizando clusters de máquinas.
- Computación "fuera de núcleo" para conjuntos de datos muy grandes que no caben en la memoria.
- Optimización de caché de estructuras de datos y algoritmos para aprovechar al máximo el hardware.

Bagging Vs. Boosting

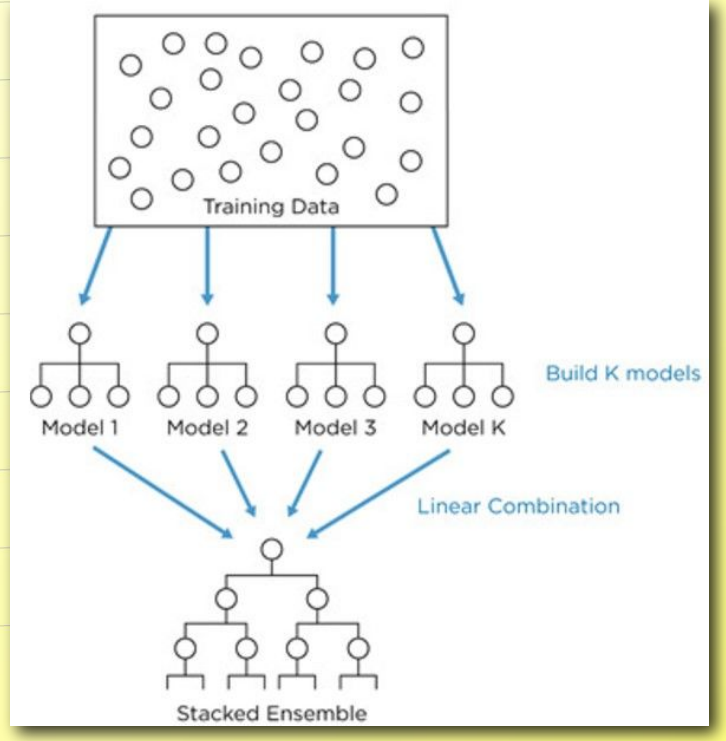
Bagging	Boosting
Modelos entrenados de manera independiente.	Bastantes modelos entrenados enfocados en mejorar las fallas de los anteriores.
Resuelve promediando los N modelos.	Promedio pesado de los N modelos (su peso depende de su performance).
Enfocado en **reducir la Varianza**. Ayuda a prevenir overfitting.	Enfocado en **reducir el Sesgo**. En casos puede causar overfitting.
Se suele usar con modelos de bajo Sesgo y alta varianza.	Se suele usar con modelos de baja varianza y alto sesgo.
Fácilmente paralelizable.	No se puede paralelizar fácilmente.

Stacking

Se crea una función de ensamble que combina los resultados de varios modelos base, en uno sólo.

Los modelos de nivel de base se entrenan con un conjunto de datos completo, y luego sus salidas se utilizan como características de entrada para entrenar una función de ensamble.

Normalmente, la función de ensamble es una simple combinación lineal de las puntuaciones del modelo base.



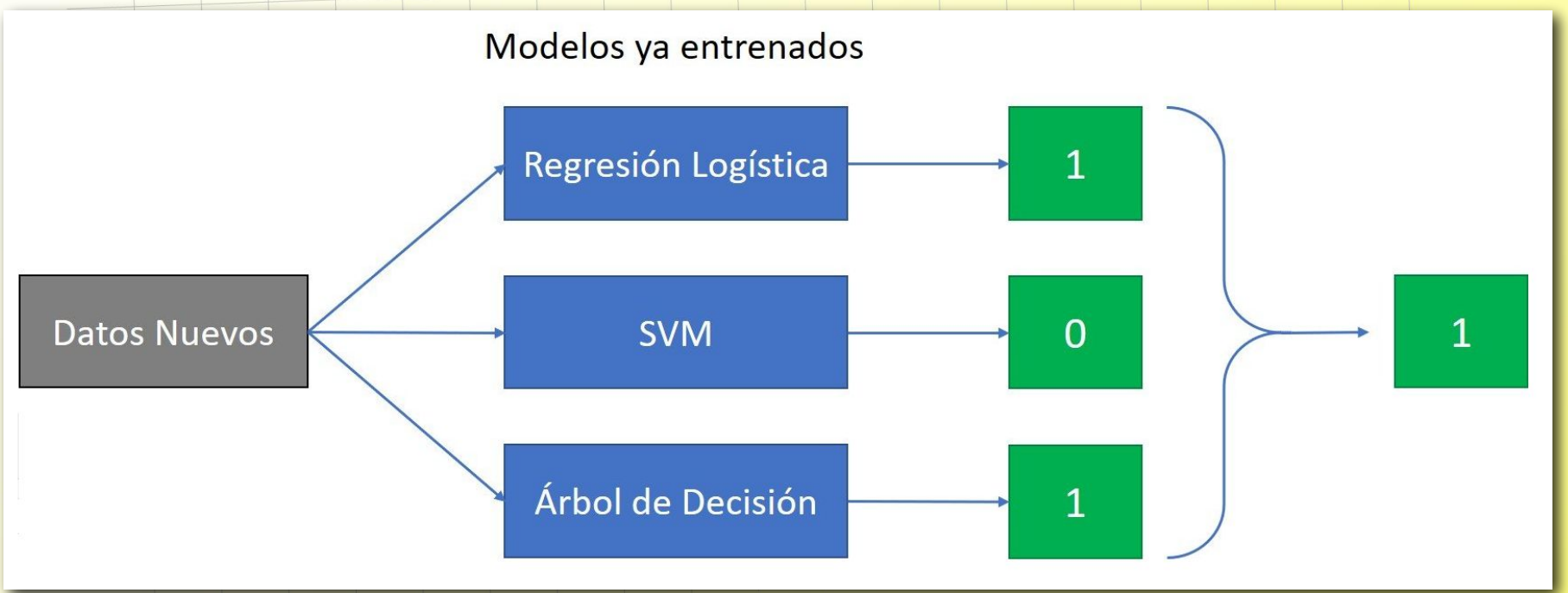
Voting Classifier

Utilizando las predicciones de múltiples clasificadores, se hacen predicciones basadas en el más frecuente.

El hiperparámetro "estimadores" crea una lista para los objetos clasificadores asignándoles nombres.

El hiperparámetro "votación" se establece en *estricto* (duro, el clasificador de votaciones emitirá juicios basados en las predicciones que aparezcan con mayor frecuencia) o *no estricto* (blando, utilizará un enfoque ponderado para tomar su decisión).

Voting Classifier



Voting Classifier

```
clf1 = clf_rnd.best_estimator_
clf2 = clf_rnd_knn.best_estimator_
clf3 = clf.best_estimator_
```

```
from sklearn.ensemble import VotingClassifier
```

```
eclf1 = VotingClassifier(estimators=[('ar1', clf1), ('ar2', clf2), ('knn', clf3)], voting='hard')
```

```
eclf1 = eclf1.fit(X_train, y_train)
```

```
y_pred_eclf1 = eclf1.predict(X_test)
```

```
score_eclf1 = accuracy_score(y_test, y_pred_eclf1)
print("Hard Voting Score % d" % score_eclf1)
```

¿PREGUNTAS?

¿Alguien dijo Homework?

HENRY

Próxima lecture

Procesamiento del Lenguaje Natural

¡Feedback!

Click on me

Dispones de un formulario en:

- Homeworks
- Guías de clase
- Slack


HENRY