

ACÀMICA

¡Bienvenidas/os a Data Science!

¡Gracias Martín Gonella por la creación de los contenidos de este encuentro!



Agenda

¿Cómo anduvieron?

Repaso: Bagging y Random Forest

Metodologías Ágiles. Aplicación en la competencia.

Break

Explicación: Boosting

Hands-on training

Explicación: Stacking

Cierre



¿Cómo anduvieron?



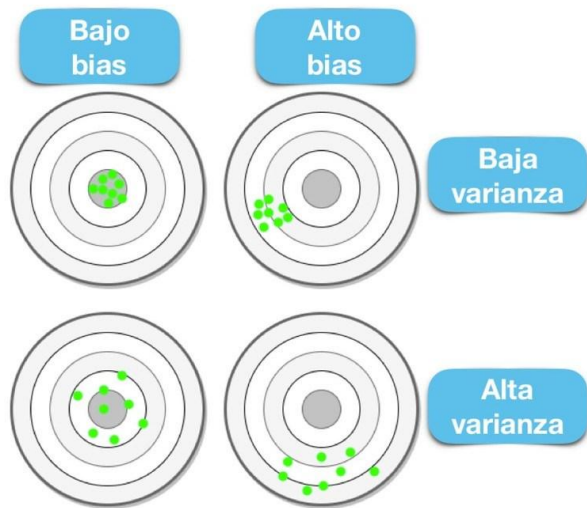
Repaso: Ensembles



Algunos comentarios sobre errores

BIAS. El error de bias o sesgo es un error de suposiciones erróneas en el algoritmo de aprendizaje.

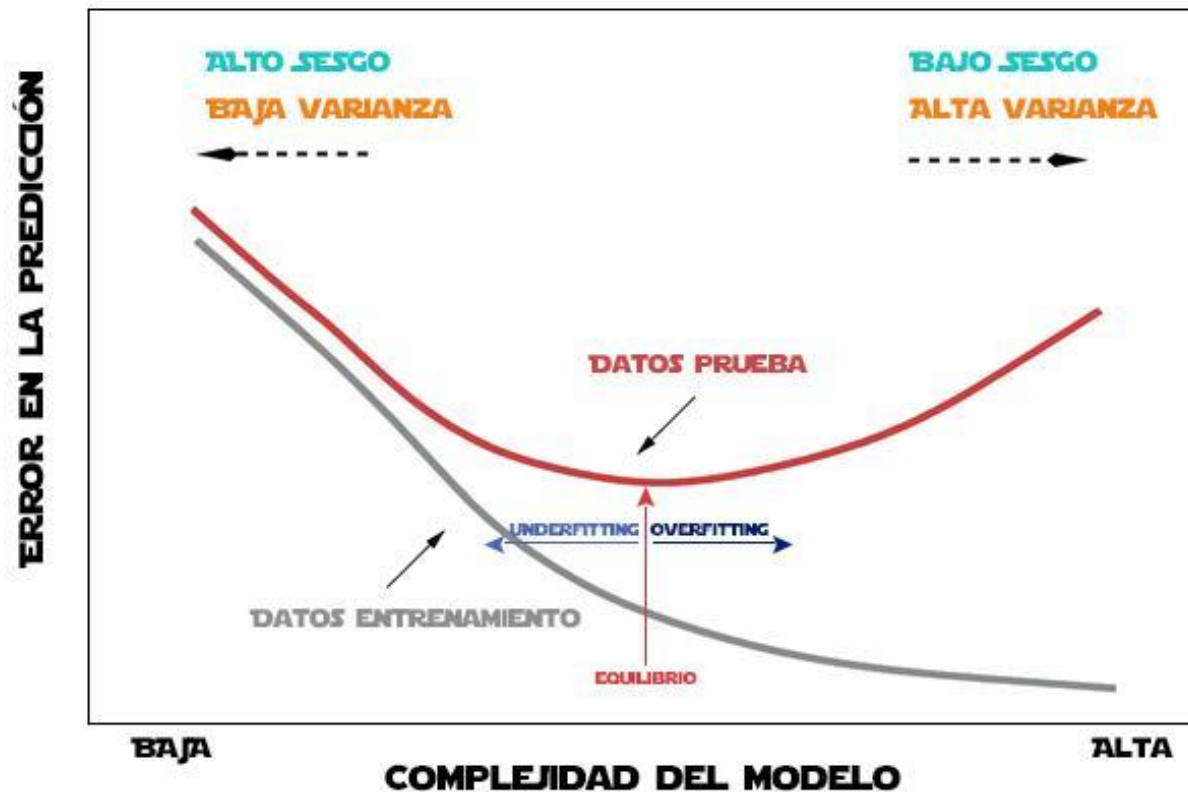
Un sesgo elevado puede hacer que un algoritmo no tenga en cuenta las relaciones relevantes entre las características y las salidas de destino (underfitting).



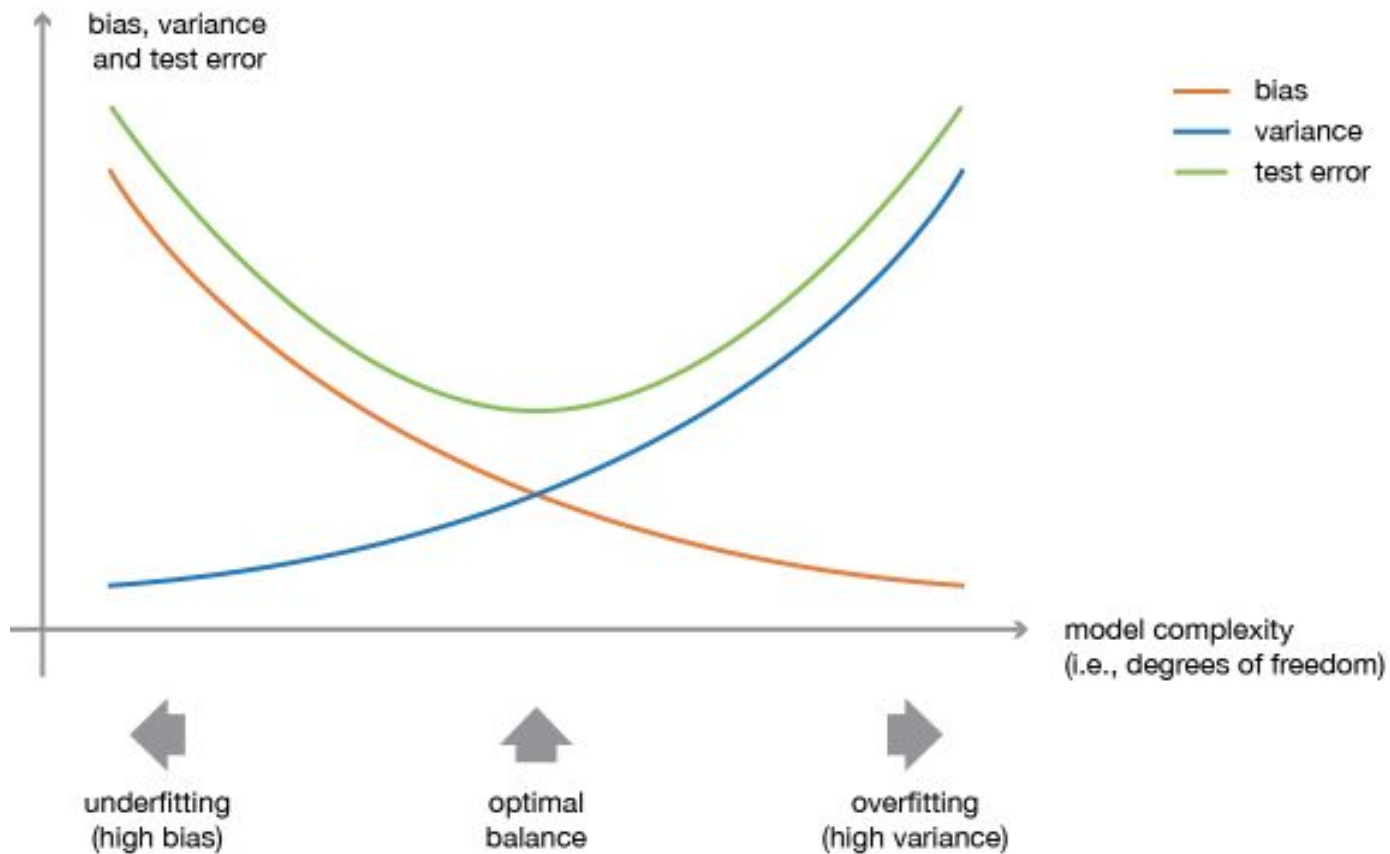
VARIANZA. La varianza es un error de sensibilidad a pequeñas fluctuaciones en el conjunto de entrenamiento.

Una alta varianza puede hacer que un algoritmo modele el ruido aleatorio en los datos de entrenamiento, en lugar de los resultados previstos (overfitting).

Trade-off Bias-Varianza

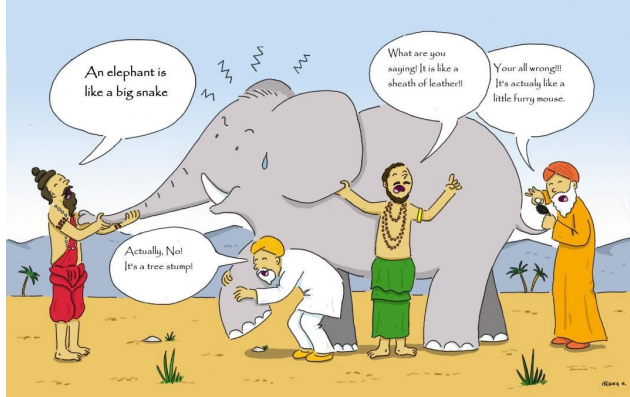


Trade-off Bias-Varianza



ENSAMBLES

"La unión hace a la fuerza"



Ensamblés

Idea: la idea aquí es **entrenar múltiples modelos**, cada uno con el objetivo de predecir o clasificar un conjunto de resultados. De esta manera **convertimos a un grupo de modelos débiles en un sólo modelo fuerte**.

¿Qué necesitamos para
que esta idea funcione?



Ensamblables

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

Necesitamos modelos diferentes entre sí, poco correlacionados.

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

Existen varias técnicas para generar modelos de ensambles.
Las más conocidas son:

BAGGING.
BOOSTING.
STACKING.





Bagging (**Bootstrap** Aggregation)

El **Bagging** implementa predictores similares en poblaciones de muestras pequeñas y luego vota entre todas las predicciones.

- Combina **Bootstrapping** (muestreo de datos con reemplazo) y **Aggregation** (votación) para formar un modelo de ensamble.



Bagging (**Bootstrap** Aggregation)

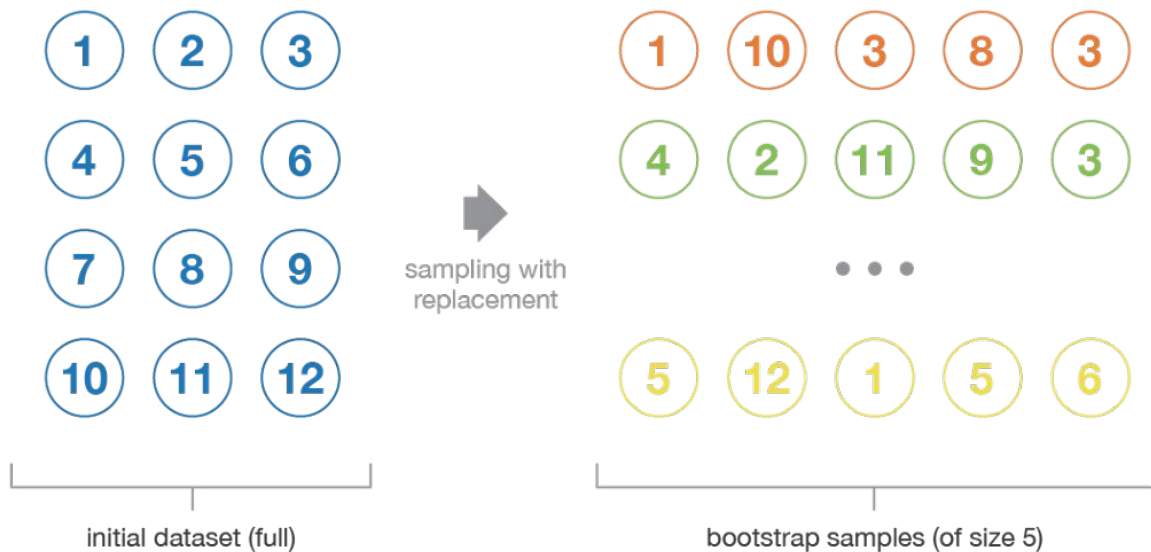
El **Bagging** implementa predictores similares en poblaciones de muestras pequeñas y luego vota entre todas las predicciones.

- Combina **Bootstrapping** (muestreo de datos con reemplazo) y **Aggregation** (votación) para formar un modelo de ensamble.

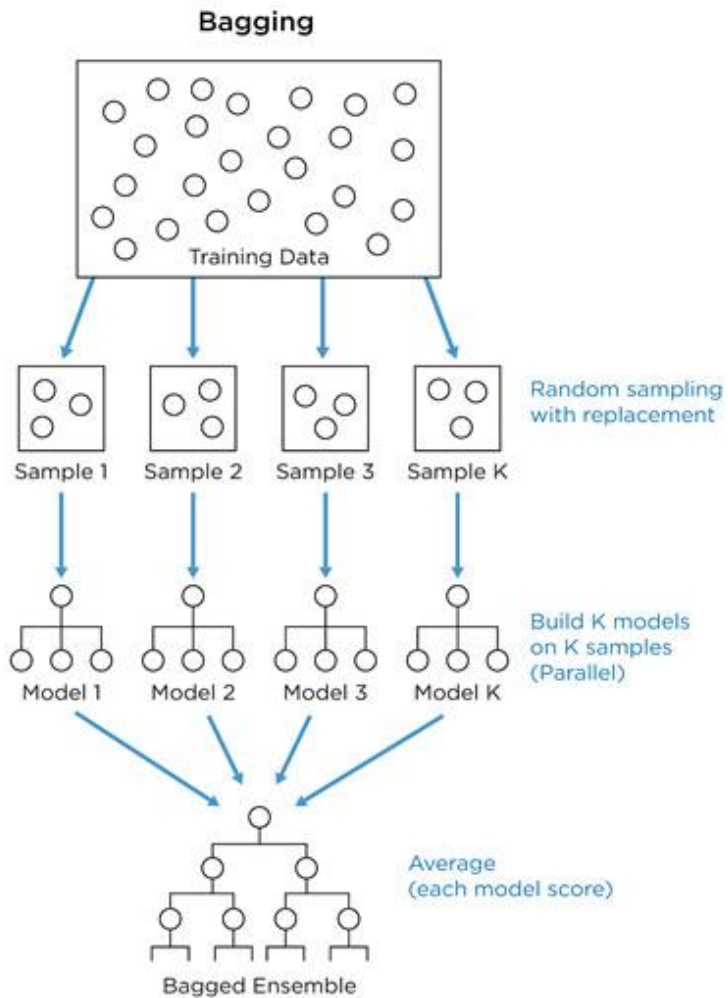
Reduce el error de varianza y ayuda a evitar el sobreajuste.



Bagging (**Bootstrap** Aggregation)



Técnicas de Ensamble



Bagging (**Bootstrap** Aggregation)

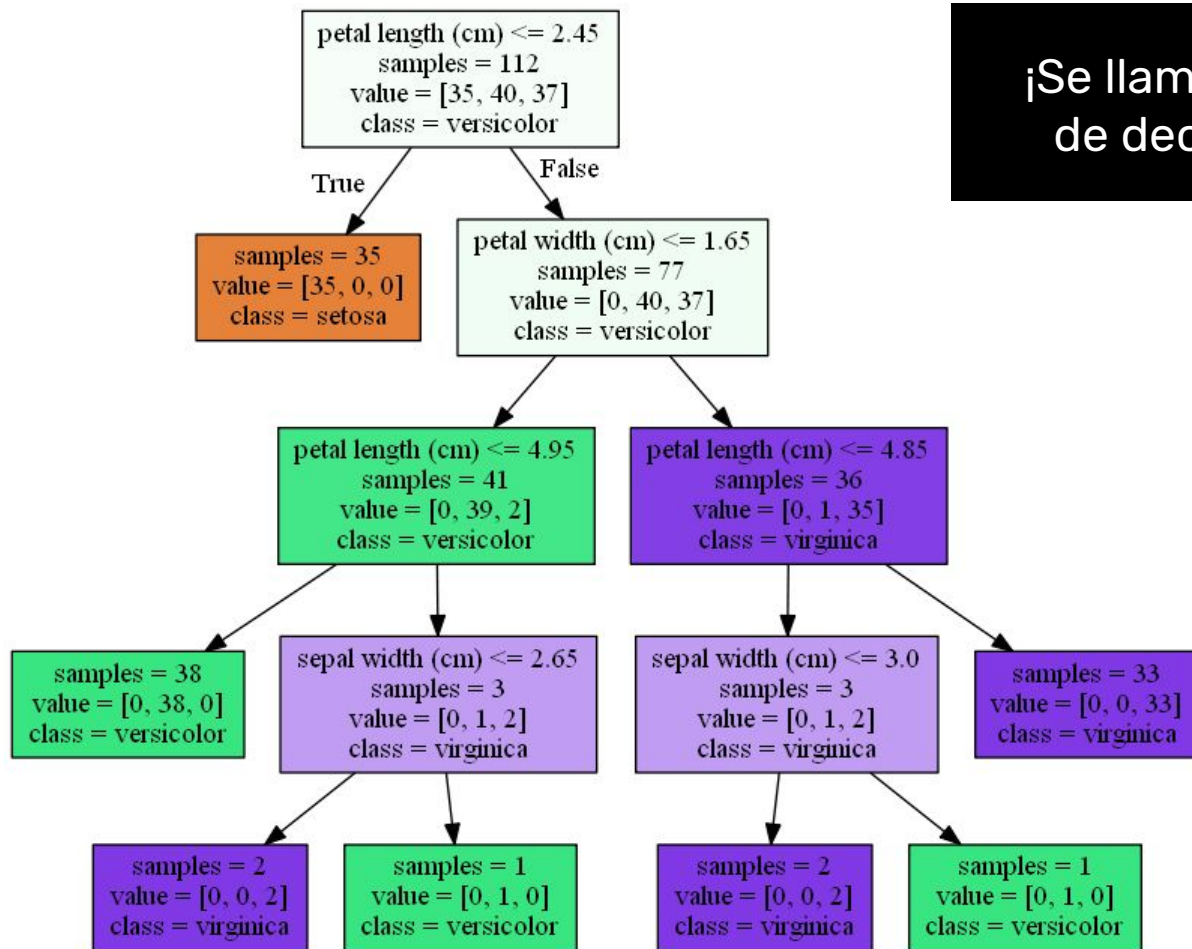
Esta técnica se puede usar con cualquier tipo de modelo: Árboles, KNN, SVM, etc.

Pero **lo más común** es que se aplique en árboles, para crear bosques.



Bagging Random Forest

¡Se llama árbol
de decisión!



Random Forest con



1.11.2. Forests of randomized trees

The `sklearn.ensemble` module includes two averaging algorithms based on randomized [decision trees](#): the RandomForest algorithm and the Extra-Trees method. Both algorithms are perturb-and-combine techniques [\[B1998\]](#) specifically designed for trees. This means a diverse set of classifiers is created by introducing randomness in the classifier construction. The prediction of the ensemble is given as the averaged prediction of the individual classifiers.

As other classifiers, forest classifiers have to be fitted with two arrays: a sparse or dense array X of size `[n_samples, n_features]` holding the training samples, and an array Y of size `[n_samples]` holding the target values (class labels) for the training samples:

```
>>> from sklearn.ensemble import RandomForestClassifier
>>> X = [[0, 0], [1, 1]]
>>> Y = [0, 1]
>>> clf = RandomForestClassifier(n_estimators=10)
>>> clf = clf.fit(X, Y)
```

Like [decision trees](#), forests of trees also extend to [multi-output problems](#) (if Y is an array of size `[n_samples, n_outputs]`).

¡A leer la documentación!

Metodologías Ágiles



Las metodologías ágiles **son aquellas que permiten adaptar la forma de trabajo a las condiciones del proyecto**, consiguiendo **flexibilidad e inmediatez en la respuesta** para **amoldar el proyecto y su desarrollo a las circunstancias específicas del entorno.**

KANBAN

Kanban es una palabra japonesa que significa **“tarjetas visuales”**.

Es una **técnica de gestión visual de las tareas**.

Implica la generación de un tablero de tareas que permitirá ver el estado de los proyectos, así como también pautar el desarrollo del trabajo de manera efectiva.

La posibilidad de poder **realizar entregas en cualquier momento, cambiar prioridades al vuelo y la visualización perfecta del flujo**, son algunos de los puntos diferenciales y de valor.

KANBAN: Pasos

1# DEFINIR EL FLUJO DE TRABAJO DE LOS PROYECTOS: Crear el tablero de forma visible y accesible para el equipo. Cada columna corresponderá a un estado concreto del flujo de tareas, que nos servirá para saber en qué situación se encuentra cada proyecto. El tablero debe tener tantas columnas como estados por los que pasa una tarea, desde que se inicia hasta que finaliza

2# VISUALIZAR LAS FASES DEL CICLO DE PRODUCCIÓN: Las tareas se dividen en pasos.. Cada una de esas partes se escribe en un post-it y se pega en el tablero, en la fase que corresponda. Dichos post-its contienen la información básica: descripción+ estimación de horas. Se puede agregar fotos, dibujos, etc.

3# STOP STARTING, START FINISHING: Se prioriza el trabajo que está en curso. El trabajo en curso debe estar limitado y existe un número máximo de tareas a realizar en cada fase. A esto, se le añade otra idea: no se puede abrir una nueva tarea sin finalizar otra.

4# CONTROL DEL FLUJO: Se trata de mantener a los trabajadores con un flujo de trabajo constante, las tareas más importantes en cola para ser desarrolladas y un seguimiento pasivo para no tener que interrumpir al trabajador en cada momento.

Pendiente	En progreso	Completado
		<div>Item #1</div> <div>t.1.6 t.1.1 t.1.3</div> <div>t.1.5 t.1.2</div>
		<div>Item #2</div> <div>t.2.3 t.2.7 t.2.5 t.2.1</div> <div>t.2.6 t.2.2 t.2.4</div>
<div>Item #3</div>	<div>t.3.5</div>	<div>t.3.2 t.3.1</div> <div>t.3.3 t.3.4</div>
<div>Item #4</div> <div>t.4.5</div>	<div>t.4.3</div> <div>t.4.4</div>	<div>t.4.1</div> <div>t.4.2</div>
<div>Item #5</div> <div>t.5.4 t.5.3 t.5.1</div> <div>t.5.5 t.5.2</div>		

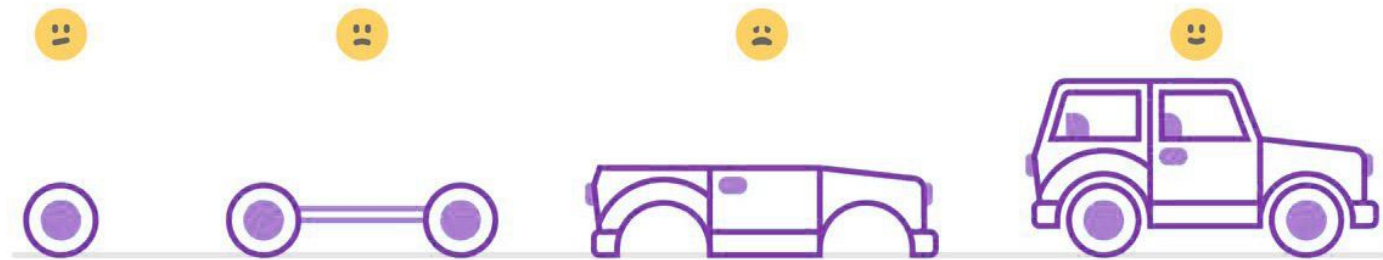
LEAN START-UP

En un marco ágil, el término lean hace referencia a que es liviano, que no le sobra nada. Se basa en la idea de

Producto Mínimo Viable (MVP, Minimum Viable Product)

para no tener que esperar a tener un producto totalmente terminado y diseñado, sino más bien tener versiones iniciales que nos permitan testear hipótesis que tenemos acerca de su valor para el cliente.

Esta metodología se centra en el usuario y las personas, iterar, trabajar en equipo, compartir información, hacer pequeñas pruebas antes de las más grandes y **desarrollar el producto final de manera incremental.**

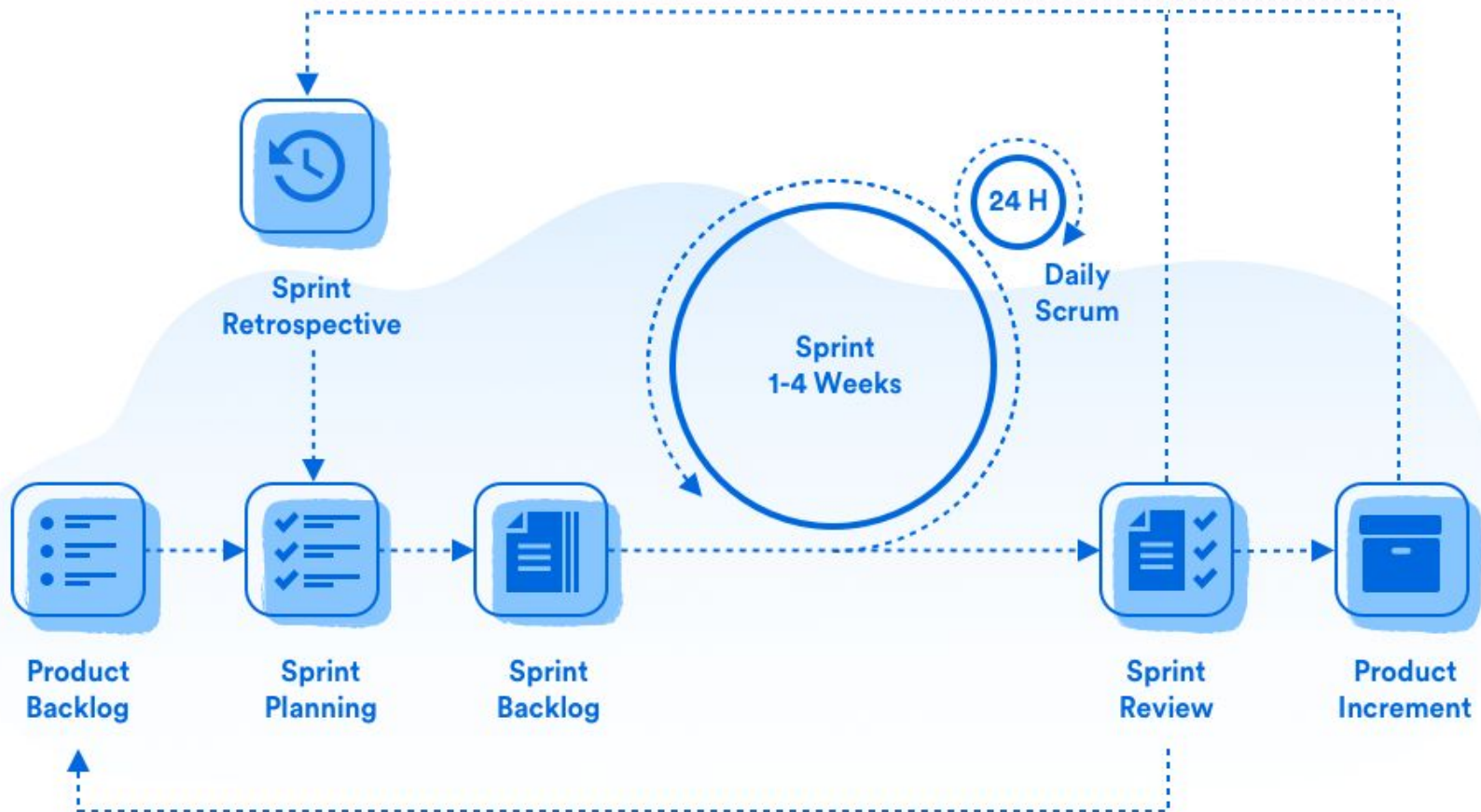


SCRUM

Scrum brinda un marco de trabajo y unas pautas para proyectos de desarrollo de software con los siguientes elementos:

Equipos de trabajo **pequeños y multidisciplinarios**, que establecen bloques de trabajo llamados **sprints**, donde sí hay una **planificación** y un **compromiso** para lograr unos objetivos en un **período de tiempo**.

El proyecto no se define ni detalla completamente a su inicio, sino que se establecen unos acuerdos iniciales y **en cada sprint** se toman las decisiones necesarias y se evalúa lo hecho en el sprint anterior (retrospectiva).



SCRUM

Ceremonias:

- Planning
- Daily
- Review
- Retrospectiva

Enfoque:

- Backlog del Producto
- Backlog del Sprint
- Entregable



Actividad

- Nos separamos por equipo en breakout rooms.
- Cada equipo deberá generar su tablero de trabajo con la herramienta Trello.
- Identificar las tareas necesarias a realizar por el equipo para alcanzar la predicción que se pide en la competencia y con ello los estados que las mismas. Plasmarlas en el tablero



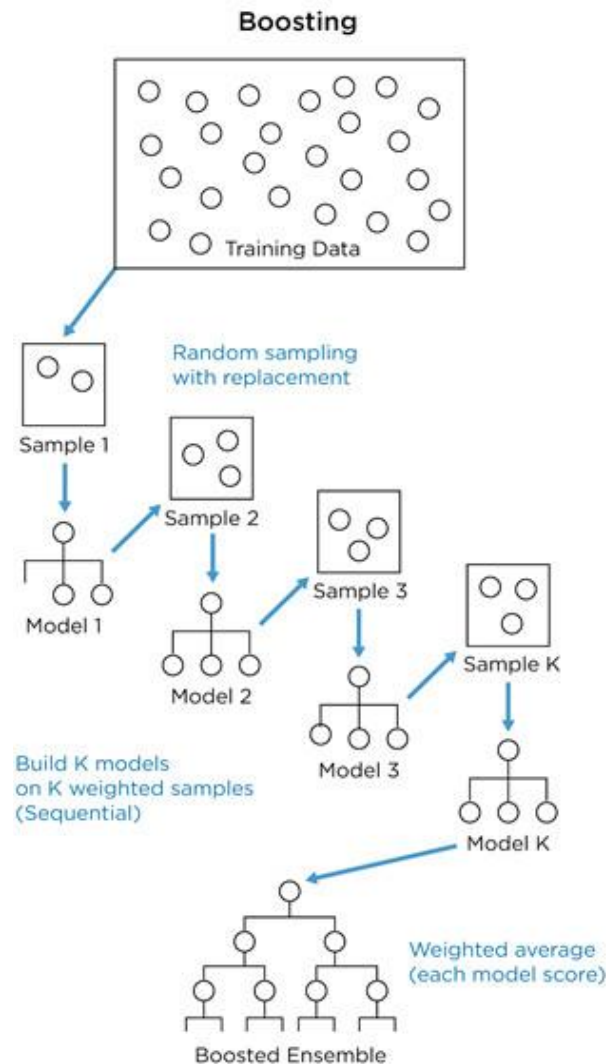
Ensembles: Boosting



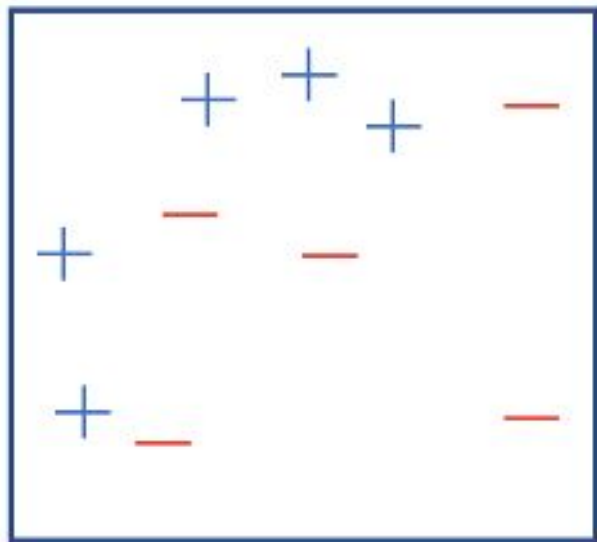
Ensambls • Boosting

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.

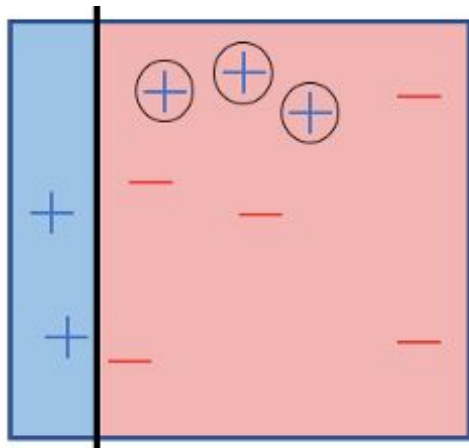


BOOSTING: AdaBoost (1 / 5)



Para explicar un poco el funcionamiento de **AdaBoost** consideremos el siguiente problema de clasificación binaria con 10 ejemplos de entrenamiento, **5 positivos y 5 negativos.**

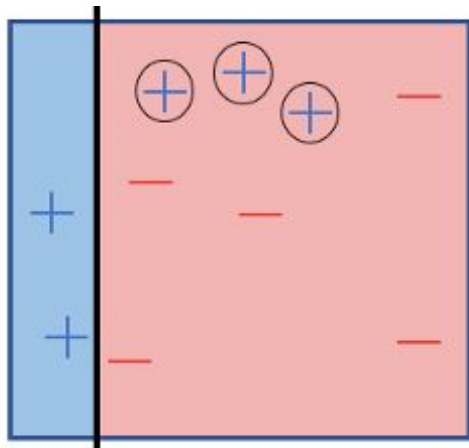
BOOSTING: AdaBoost (2 / 5)



A la izquierda el primer clasificador débil, una recta vertical. A la derecha de la recta, consideramos que todos los ejemplos son negativos, mientras que a la izquierda son positivos.

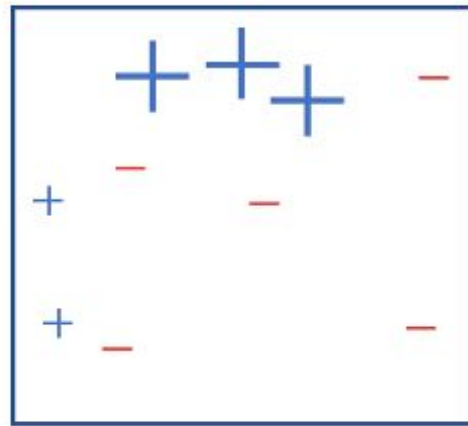
La recta clasifica mal a tres positivos.

BOOSTING: AdaBoost (2 / 5)



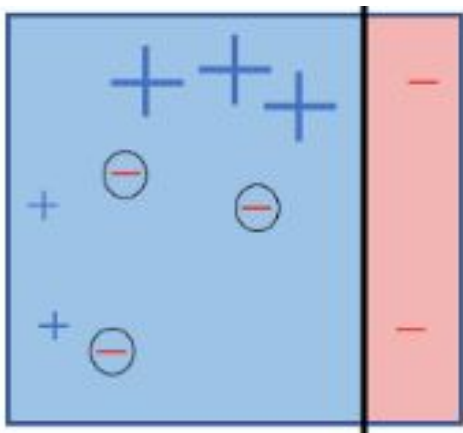
A la izquierda el primer clasificador débil, una recta vertical. A la derecha de la recta, consideramos que todos los ejemplos son negativos, mientras que a la izquierda son positivos.

La recta clasifica mal a tres positivos.



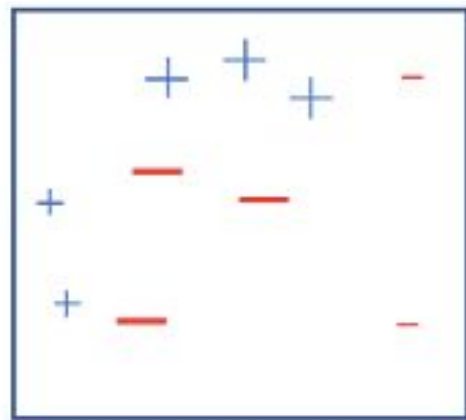
Aquí vemos como los tres ejemplos mal clasificados aparecen ahora de un mayor tamaño que el resto de los ejemplos. Esto simboliza que dichos ejemplos tendrán una mayor importancia al momento de seleccionar el clasificador débil de la segunda iteración.

BOOSTING: AdaBoost (3 / 5)



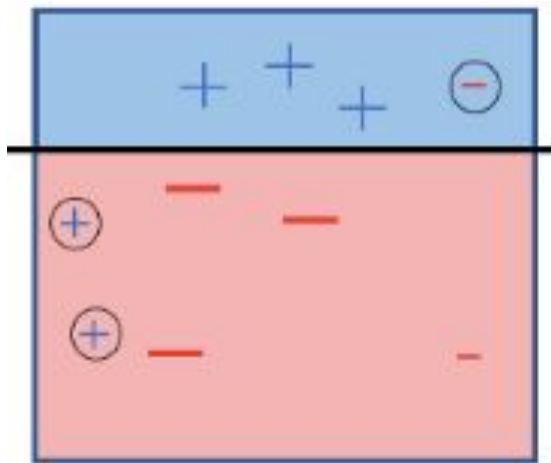
Aquí vemos al segundo clasificador débil, que es otra recta vertical colocada más hacia la derecha.

Este segundo clasificador se equivoca también en tres ejemplos, **ya que clasifica mal ejemplos negativos.**



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

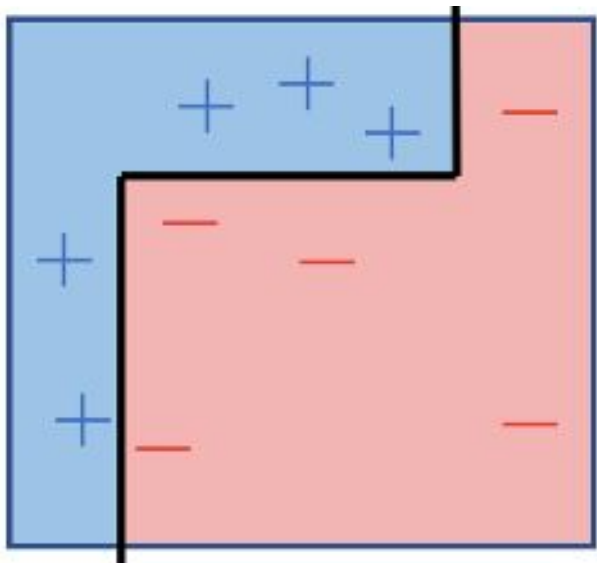
BOOSTING: AdaBoost (4 / 5)



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: AdaBoost (5 / 5)



Finalmente, se ilustra el clasificador fuerte que resulta de crear un ensamble con tres clasificadores débiles. La forma en que utilizamos estos tres clasificadores débiles es mediante una decisión por mayoría.

Cuando deseamos clasificar un nuevo ejemplo, le preguntamos a cada uno de nuestros tres clasificadores débiles su opinión. Si la mayoría opina que el nuevo ejemplo es positivo, pues entonces la decisión del clasificador fuerte será que es un ejemplo positivo.

Adaboost con



The following example shows how to fit an AdaBoost classifier with 100 weak learners:

```
>>> from sklearn.model_selection import cross_val_score
>>> from sklearn.datasets import load_iris
>>> from sklearn.ensemble import AdaBoostClassifier

>>> iris = load_iris()
>>> clf = AdaBoostClassifier(n_estimators=100)
>>> scores = cross_val_score(clf, iris.data, iris.target, cv=5)
>>> scores.mean()
0.9...
```

The number of weak learners is controlled by the parameter `n_estimators`. The `learning_rate` parameter controls the contribution of the weak learners in the final combination. By default, weak learners are decision stumps. Different weak learners can be specified through the `base_estimator` parameter. The main parameters to tune to obtain good results are `n_estimators` and the complexity of the base estimators (e.g., its depth `max_depth` or minimum required number of samples to consider a split `min_samples_split`).

¡A leer la documentación!

Boosting XG Boost

(Extreme Gradient Boosting)

Boosting: XGBoost (Extreme Gradient Boosting)



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

Boosting: XGBoost (Extreme Gradient Boosting)



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

Se puede usar con Python, **como si fuera un modelo de Scikit-Learn.**



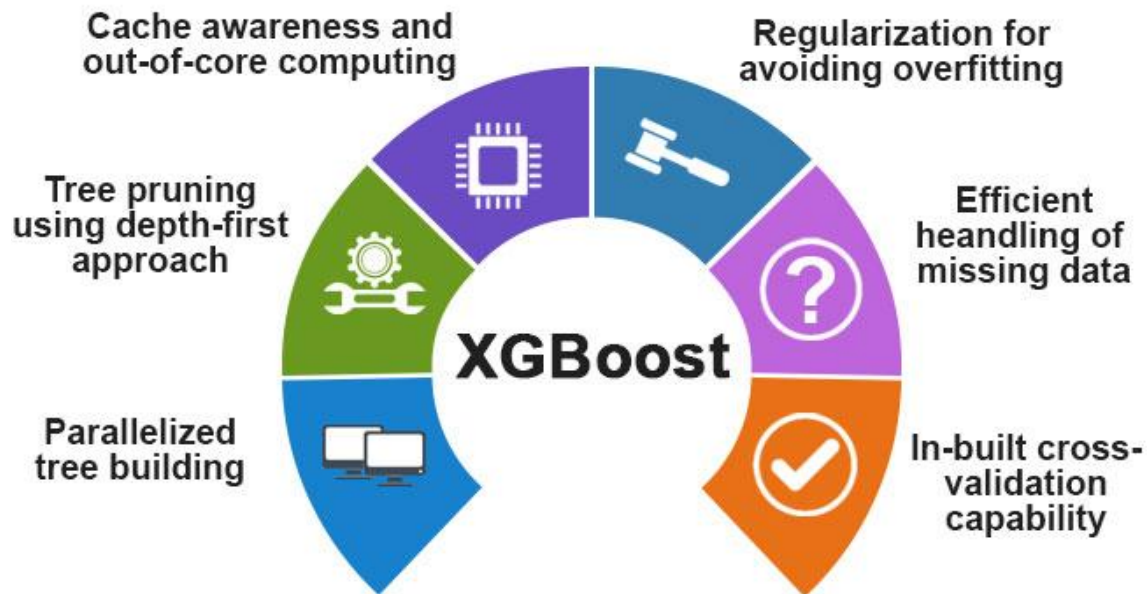
Características principales

- **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.

Boosting:
XGBoost
(Extreme
Gradient
Boosting)



Características principales



XG Boost en Python

Instalación: `sudo pip install xgboost`

Ejemplos: <https://github.com/tqchen/xgboost/tree/master/demo/guide-python>

```
diabetes = load_diabetes()

X = diabetes.data
y = diabetes.target

xgb_model = xgb.XGBRegressor(objective="reg:linear", random_state=42)

xgb_model.fit(X, y)

y_pred = xgb_model.predict(X)
```


Hands-on training



Hands-on training



DS_Encuentro_28_Boosting.ipynb

A close-up photograph of a white ceramic cup filled with a latte. The surface of the milk is decorated with intricate latte art, featuring a central heart shape surrounded by concentric, wavy lines. The cup is placed on a matching white saucer. In the background, a white napkin and a silver fork are visible, though they are out of focus. The overall lighting is soft and warm, creating a cozy atmosphere.

¡BREAK!



¿Qué es mejor, Bagging o Boosting?



Depende.



Bagging

vs.

Boosting

- Modelos entrenados de manera independiente.
- Resuelve promediando los N modelos.
- Enfocado en reducir la varianza. Ayuda a prevenir overfitting.
- Se suele usar con modelos de bajo Sesgo y alta varianza.
- Fácilmente paralelizable.

- Bastante Modelos entrenados enfocados en mejorar las fallas de los anteriores.
- Promedio pesado de los N modelos (su peso depende de su performance).
- Enfocado en reducir el Sesgo. En casos puede causar overfitting.
- Se suele usar con modelos de baja varianza y alto sesgo.
- No se puede paralelizar fácilmente.

Un último ensamble: **STACKING.**



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.

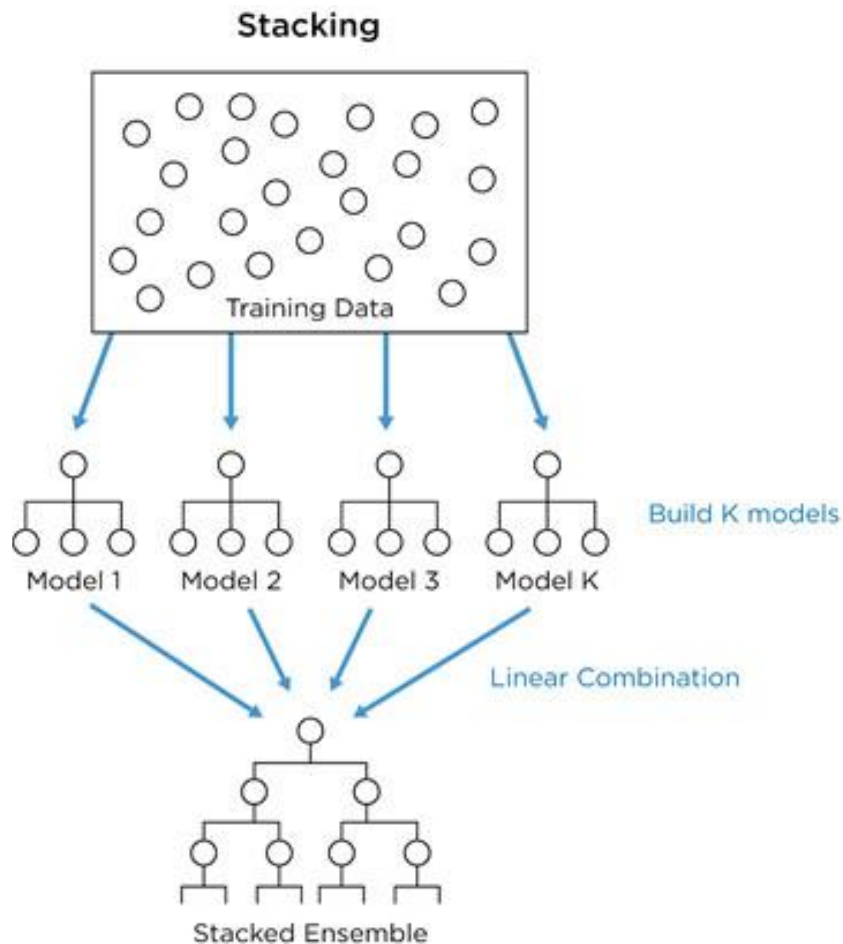
Normalmente, la función de ensamble es una simple combinación lineal de las puntuaciones del modelo base.

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.



Recursos





Si te quedaste con ganas de más...

- [Ensemble methods: bagging, boosting and stacking:](#) Excelente artículo (nivel intermedio/avanzado).
- [Understanding Random Forest:](#) Explicación (nivel básico) y ejemplo.
- [Ensamble de clasificadores usando AdaBoost:](#) Explicación (nivel básico).
- [A Gentle Introduction to XGBoost for Applied Machine Learning:](#) Introducción con referencias a material extra (nivel básico/intermedio).

Para la próxima

1. Ver los videos de la plataforma sobre “Redes Neuronales”.
2. Ponerse al día con los notebooks atrasados.
3. Terminar en la Entrega 4, si aún no lo hicieron.

ACÀMICA