ACÁMICA

¡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

Explicación: Boosting

Hands-on training

Break

Explicación: Stacking

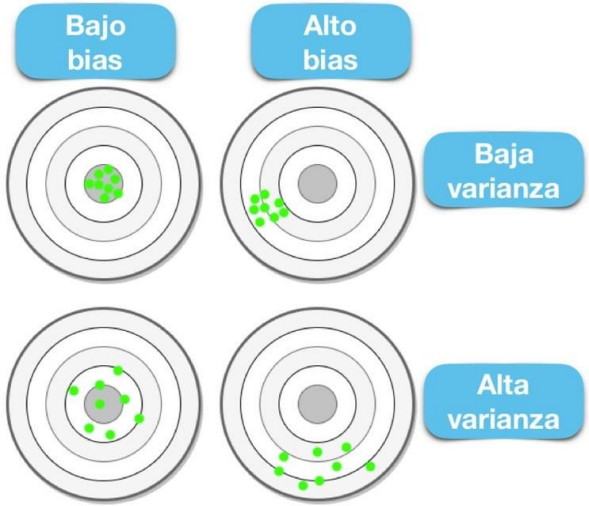
Cierre

¿Cómo anduvieron?

Repaso: Ensambles

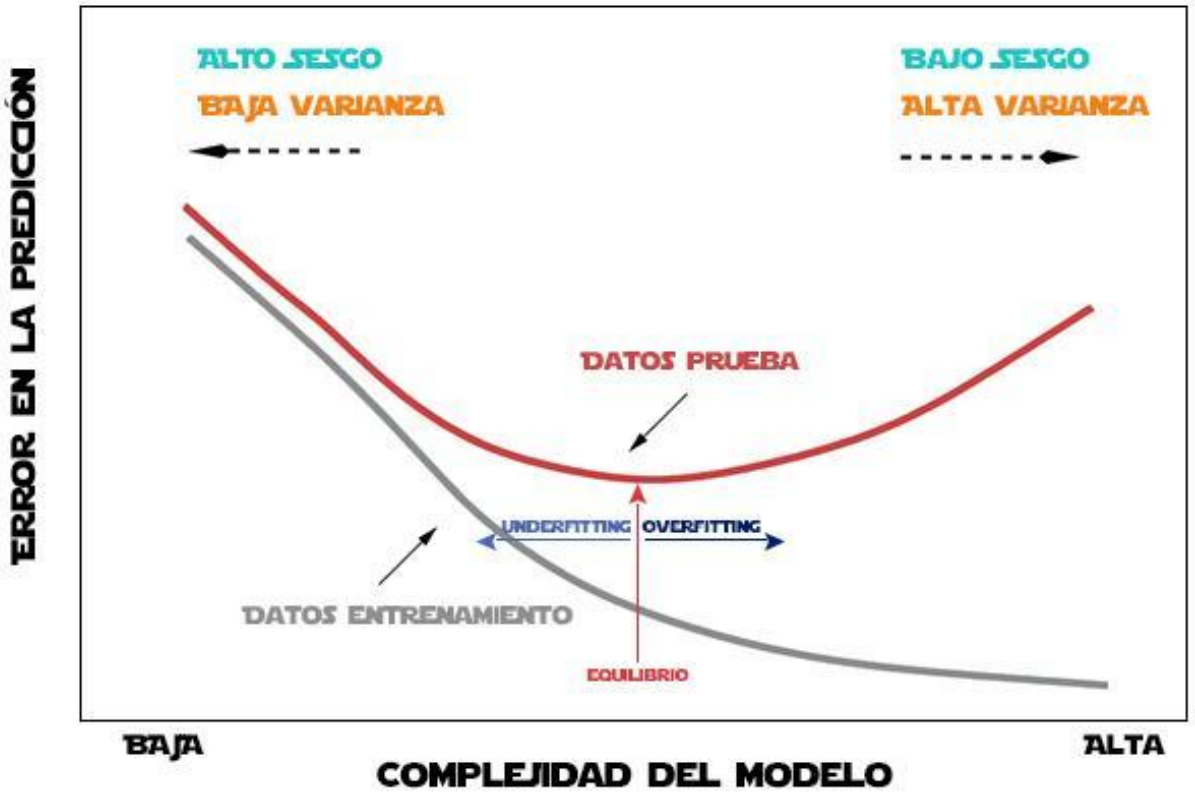
Algunos comentarios sobre errores

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).

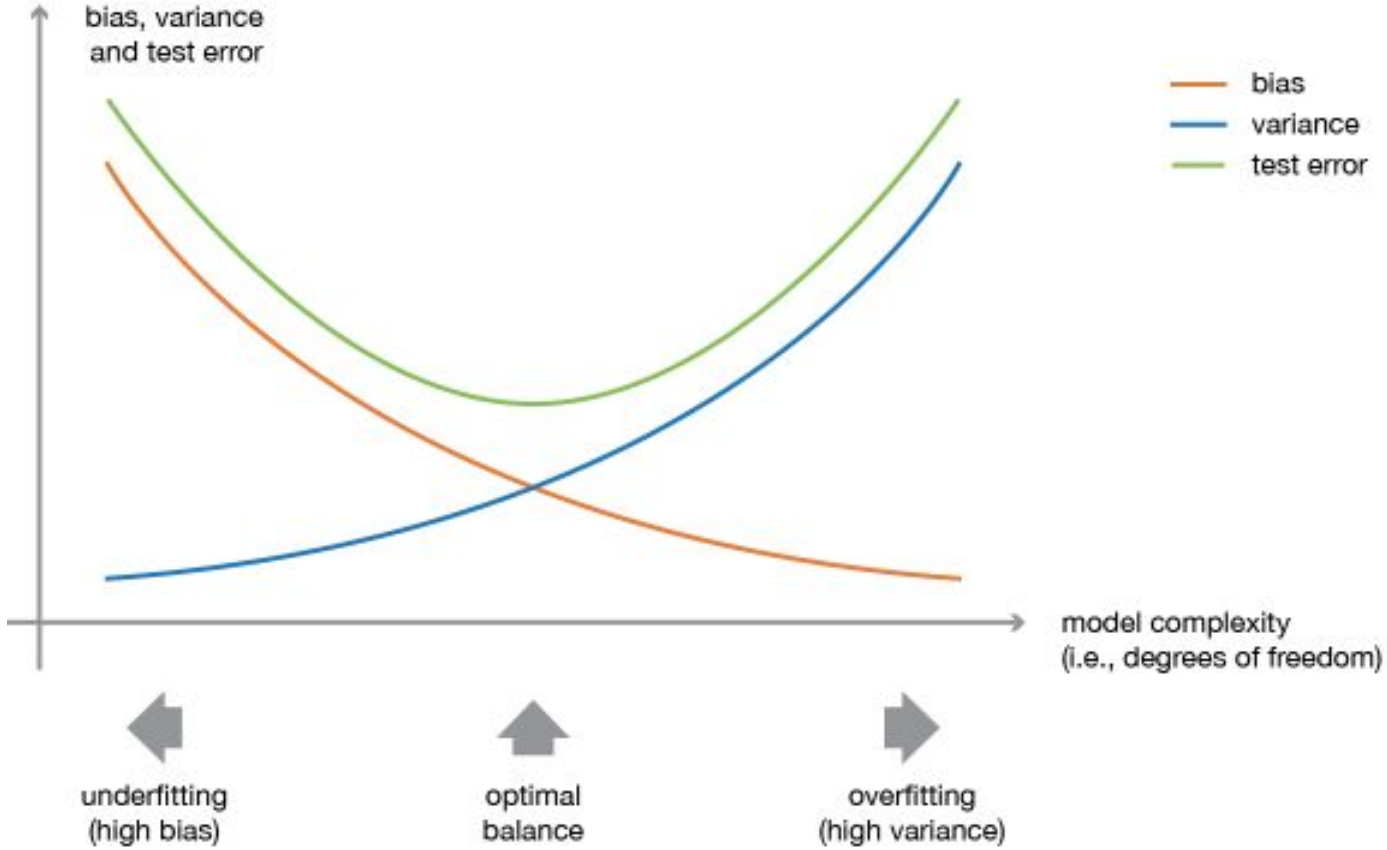


VARIANZA. La varianza es un error de sensibilidad a pequeñas fluctuaciones en el conjunto de entrenamiento. **Una alta varianza** puede hacer que un algoritmo modele el ruido aleatorio en los datos de entrenamiento, en lugar de los resultados previstos (overfitting).

Trade-off Bias-Varianza

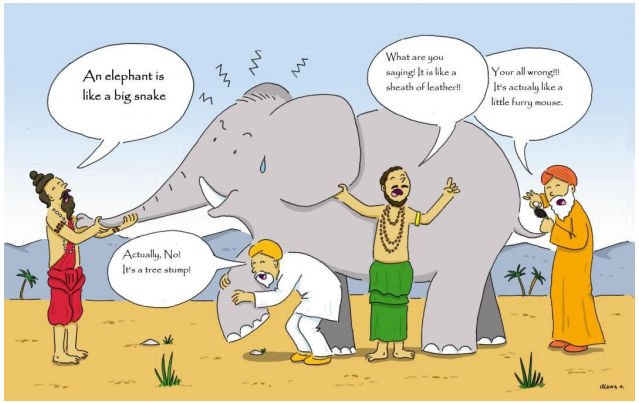


Trade-off Bias-Varianza



ENSAMBLES

“La unión hace a la fuerza”



Ensambles

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?

Ensambles

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

Técnicas de Ensamble

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)

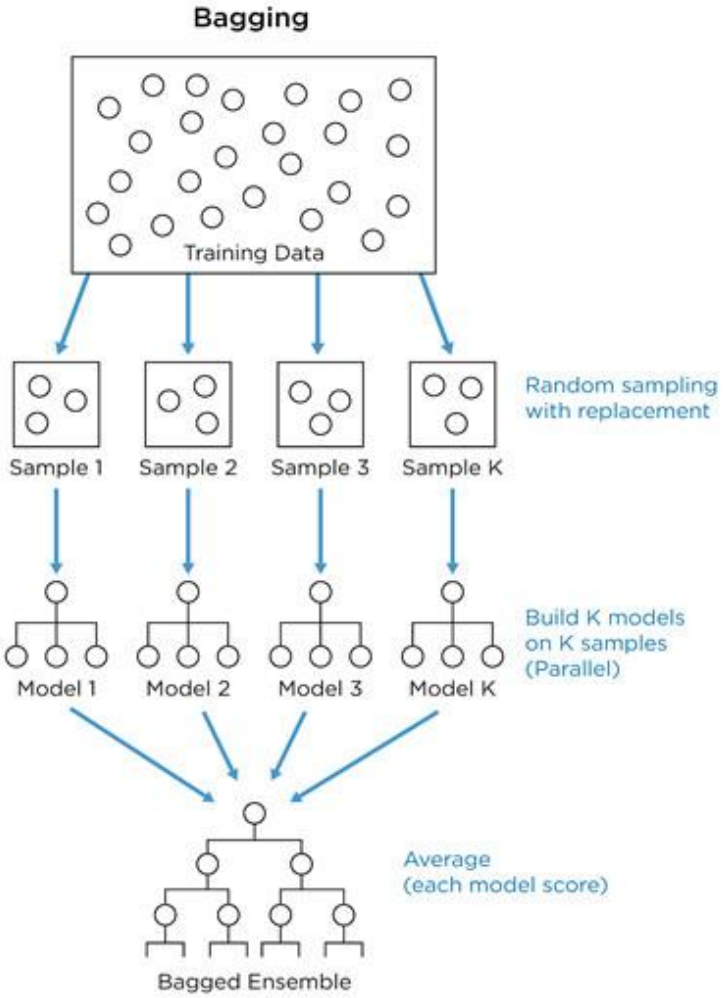
El **Bagging** implementa predictores similares en poblaciones de muestras pequeñas y luego vota entre todas las predicciones.

- 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



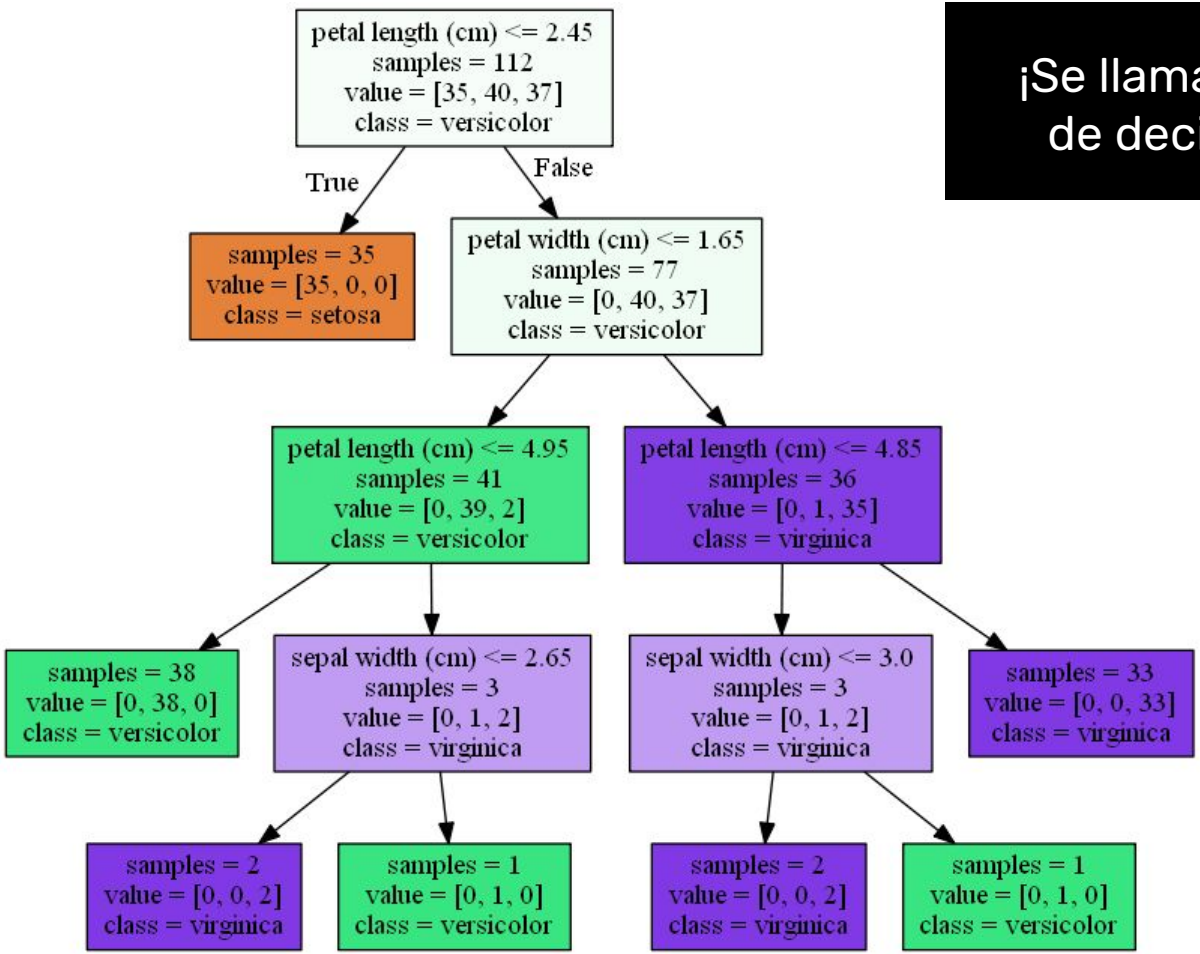
Bagging (Bootstrap Aggregation)

Ésta 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 scikit learn

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]`).

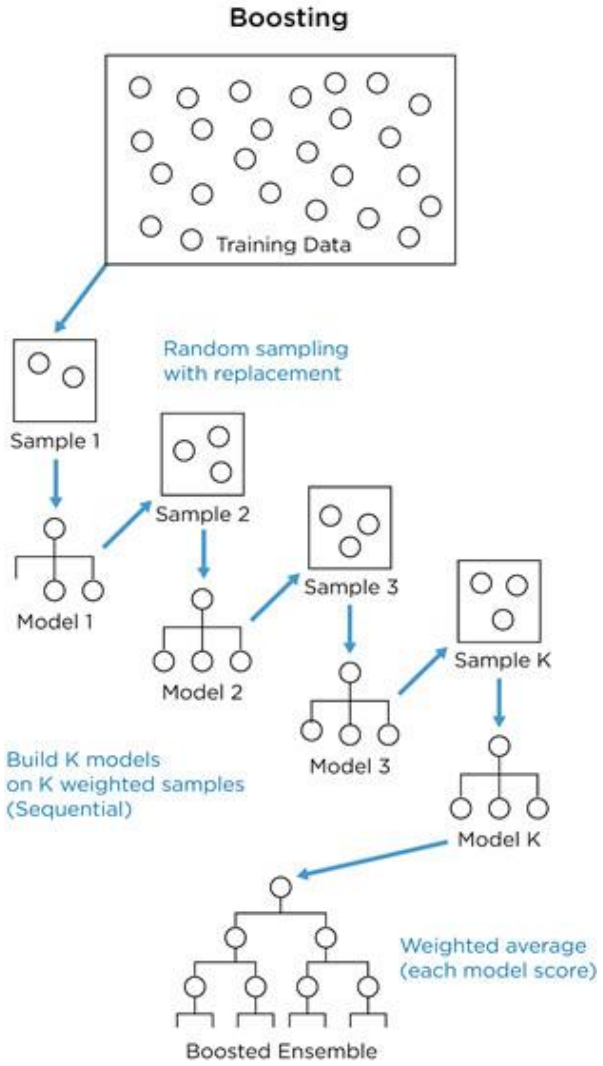
¡A leer la documentación!

Ensambles: Boosting

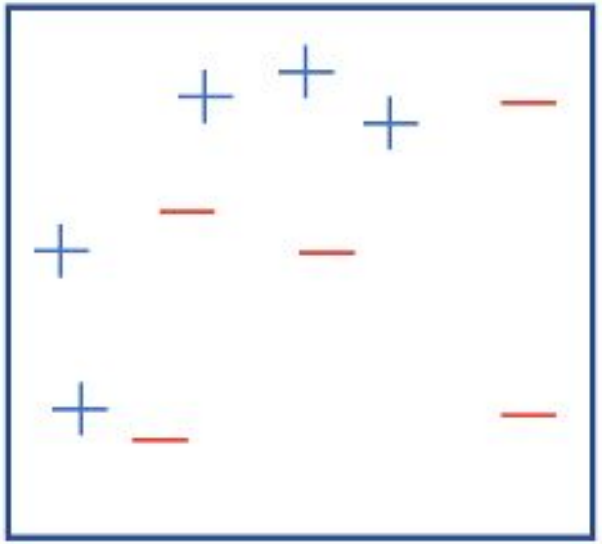
Ensambles • Boosting

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.

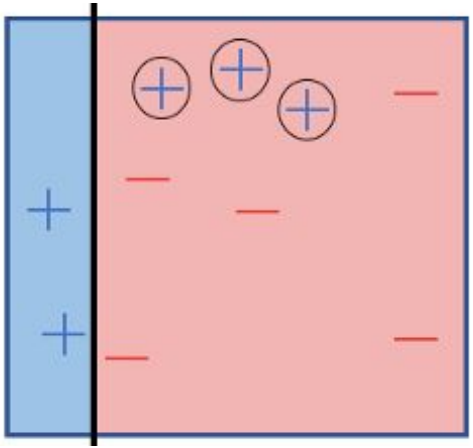


BOOSTING: AdaBoost (1 / 5)



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.**

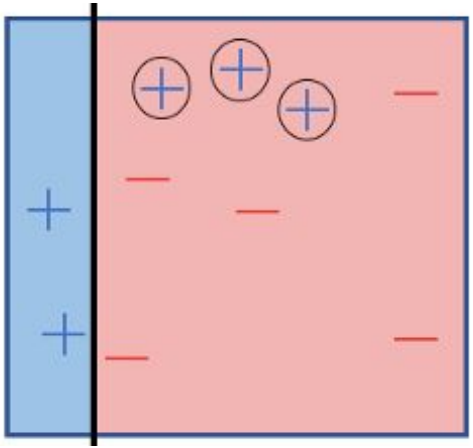
BOOSTING: AdaBoost (2 / 5)



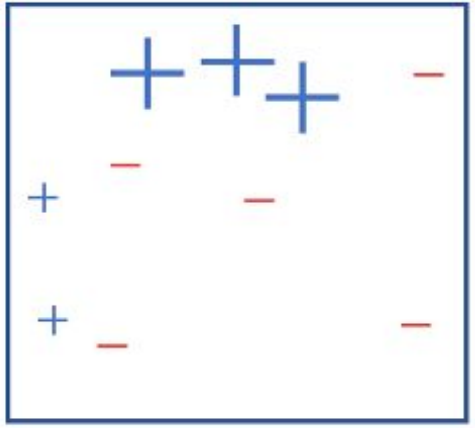
A la izquierda el primer clasificador débil, una recta vertical. A la derecha de la recta, consideramos que todos los ejemplos son negativos, mientras que a la izquierda son positivos.

La recta clasifica mal a tres positivos.

BOOSTING: AdaBoost (2 / 5)

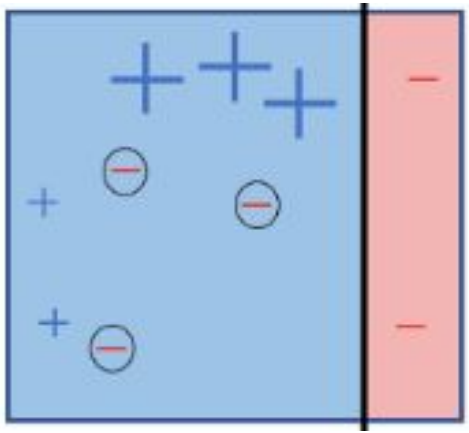


A la izquierda el primer clasificador débil, una recta vertical. A la derecha de la recta, consideramos que todos los ejemplos son negativos, mientras que a la izquierda son positivos.
La recta clasifica mal a tres positivos.

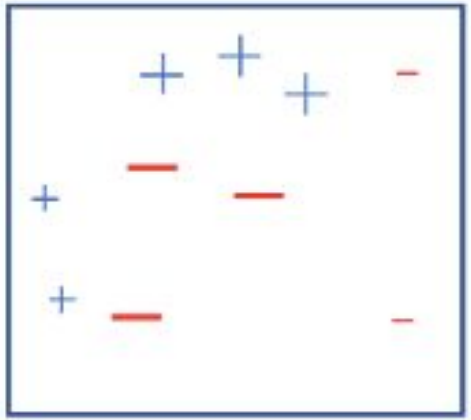


Aquí vemos como los tres ejemplos mal clasificados aparecen ahora de un mayor tamaño que el resto de los ejemplos.
Esto simboliza que dichos ejemplos tendrán una mayor importancia al momento de seleccionar el clasificador débil de la segunda iteración.

BOOSTING: AdaBoost (3 / 5)

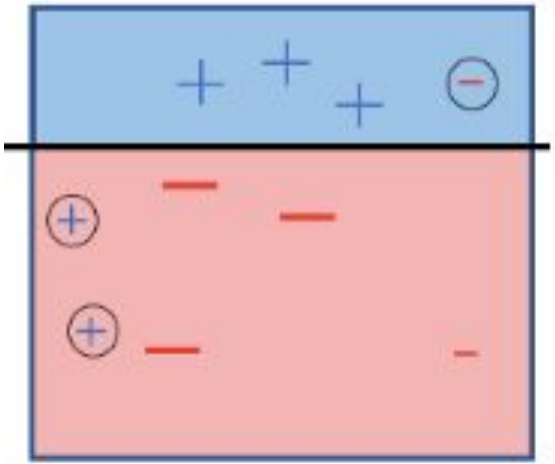


Aquí vemos al segundo clasificador débil, que es otra recta vertical colocada más hacia la derecha.
Este segundo clasificador se equivoca también en tres ejemplos,
ya que clasifica mal ejemplos negativos.



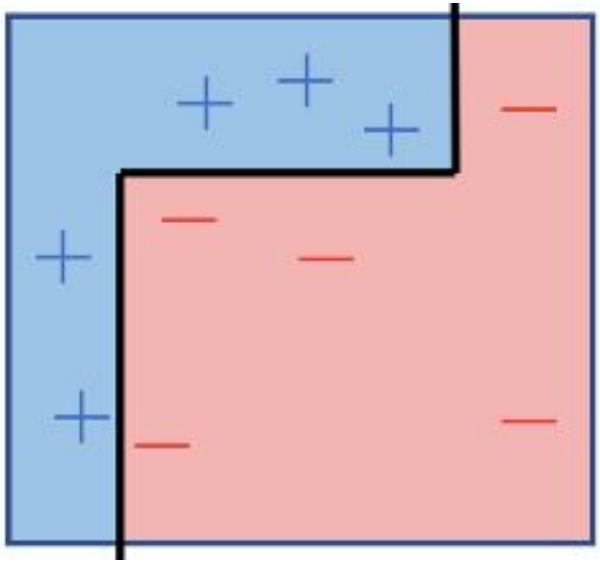
Aquí vemos que para la tercera iteración los ejemplos negativos mal clasificados tienen ahora el mayor tamaño, es decir, tendrán mayor importancia en la siguiente iteración.

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 scikit learn

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.

Disclaimer: El algoritmo de descenso por gradiente lo vamos a ver más adelante!

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.

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

Disclaimer: El algoritmo de descenso por gradiente lo vamos a ver más adelante!

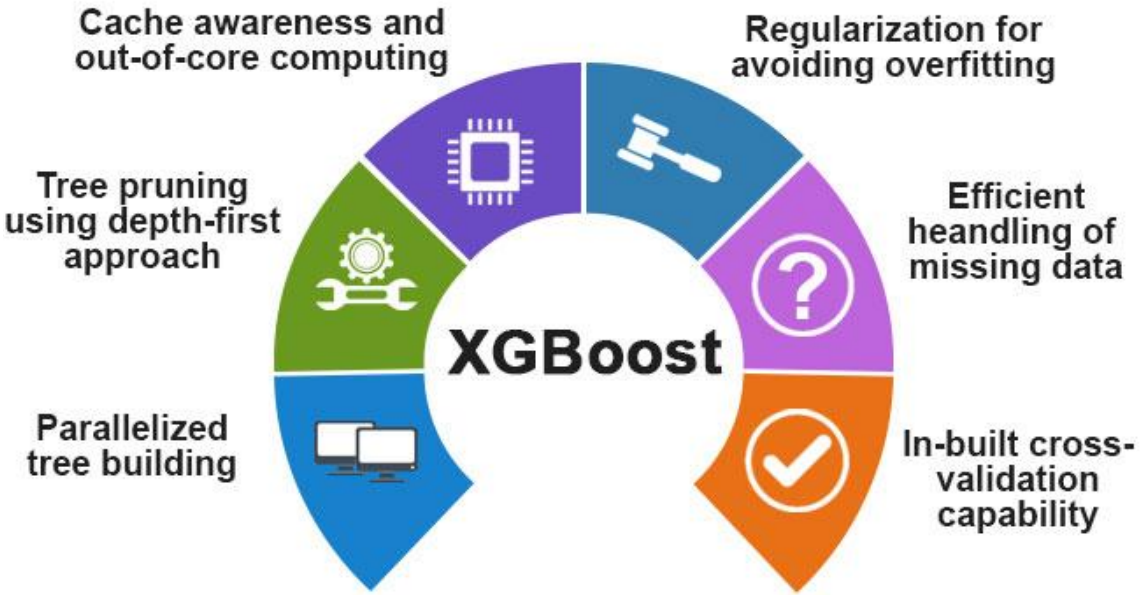
Boosting: XGBoost (Extreme Gradient Boosting)

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

¡BREAK!

Ph. Credit: Drew Coffmann

¿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.

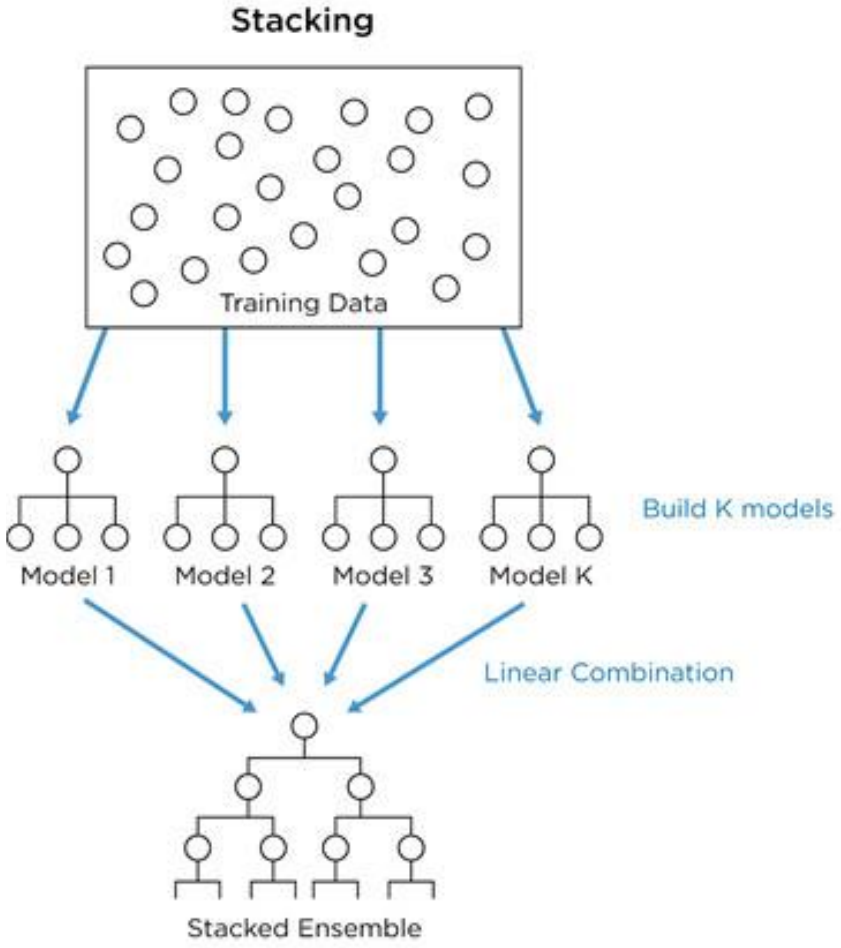
Normalmente, la función de ensamble es una simple combinación lineal de las puntuaciones del modelo base.

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

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.

ACÁMICA