ACÁMICA

¡Bienvenidas/os a Data Science!

¡Gracias Juan Spinelli por la creación de los contenidos de este encuentro!

Agenda

¿Cómo anduvieron?

Repaso: Trade-off entre sesgo y varianza

Ensambles: Bagging

Break

Hands-on Training

Cierre

¿Cómo anduvieron?

Hoja de ruta

fase	ADQUISICIÓN Y EXPLORACIÓN		MODELADO				DEPLOY
entrega	Exploración de datos	Feature Engineering	Machine Learning: Clasificación y Regresión	Optimización de parámetros	Procesam. del lenguaje natural	Sistema de recomendación	Publicación de modelos
tiempo	SEM 1	SEM 5	SEM 8	SEM 12	SEM 14	SEM 18	SEM 22
	SEM 2	SEM 6	SEM 9	SEM 13	SEM 15	SEM 19	SEM 23
	SEM 3	SEM 7	SEM 10		SEM 16	SEM 20	SEM 24
	SEM 4		SEM 11		SEM 17	SEM 21	

Cronograma

Usted
Está Aquí

SEM 13	SEM 14	SEM 15	SEM 16	SEM 17
• SVM • Modelos Avanzados	• Ensambles: Bagging, Random Forest y Boosting	• Redes Neuronales • Descenso por Gradiente • Perceptrón Simple	• Perceptrón Multicapa • Repaso	• Procesamiento del lenguaje natural

Entrega 4

Entrega 5

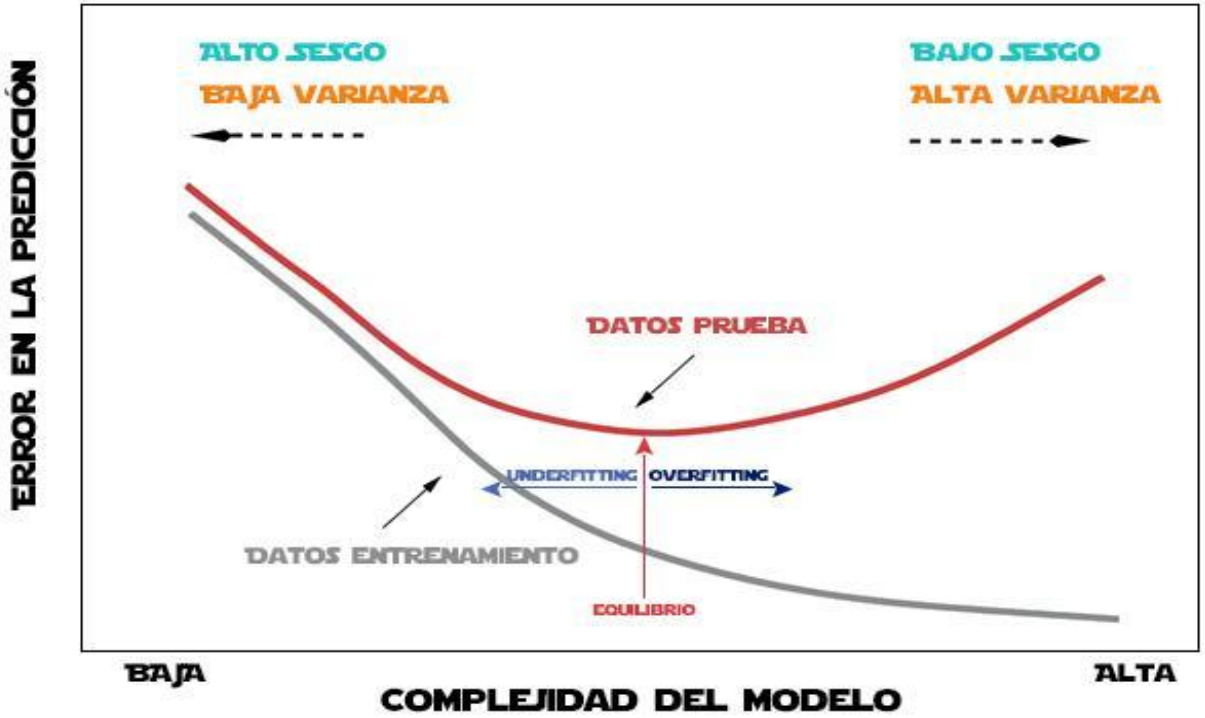
Repaso: Trade-off entre sesgo y varianza

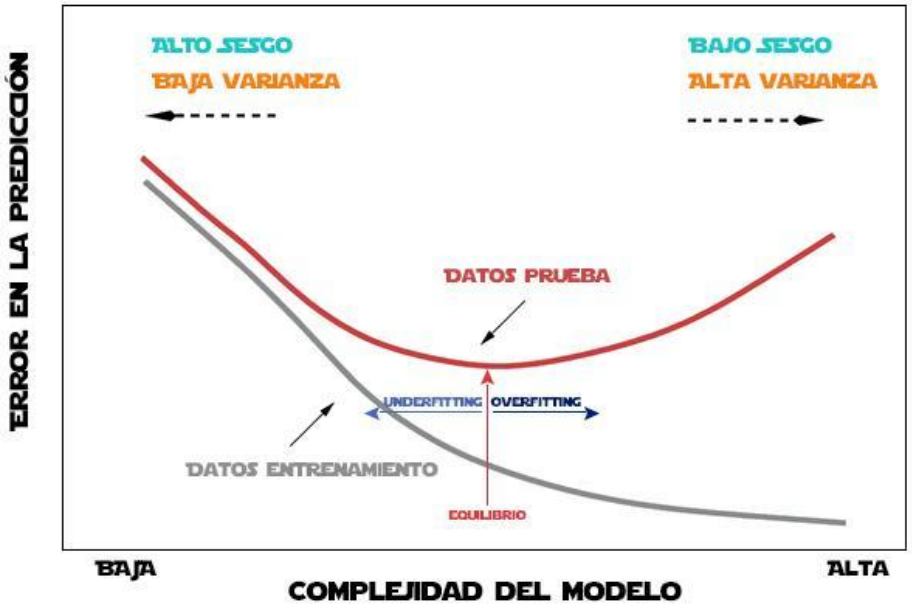
Error de predicción

El error de predicción para cualquier algoritmo de Machine Learning se puede dividir en tres partes:

Error irreducible (ruido)	Error de bias (sesgo)	Error de varianza

¿Alguien se anima a explicar lo que ven en este gráfico?





En el gráfico, si nos movemos de izquierda a derecha:

- Aumenta la complejidad de nuestro modelo
- Baja el sesgo y aumenta la varianza.
- Hasta que llega un momento en el que el error en los datos de test empieza a aumentar mientras que el de entrenamiento sigue disminuyendo. Ese punto mínimo de error en los datos de test nos indica el nivel de complejidad óptimo para nuestro modelo.

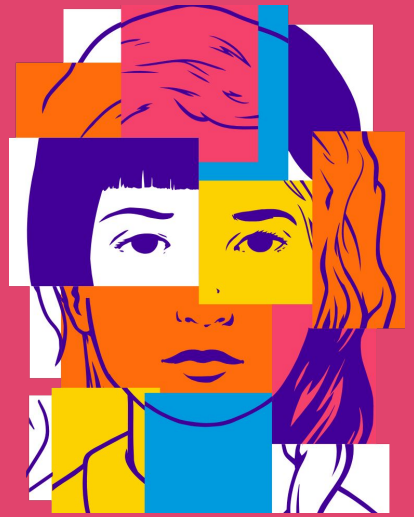
Resumen

Modelo sesgado: No logra capturar la forma de los datos. En general, tiene desempeño muy similar en el set de entrenamiento y de validación. Asociado al underfitting.

Modelo con mucha varianza: Demasiado ajustado a los datos . Tiene desempeño muy bueno en el set de entrenamiento y malo en el de validación. Asociado al overfitting.

> ## ¿Cómo diagnosticamos sesgo y varianza?
>
> **Curva de validación/complejidad:** Score en función de la complejidad. Sirve para ver regiones de baja complejidad (sesgo, underfitting) y demasiada complejidad (alta varianza, overfitting)
>
> **Curva de aprendizaje:** Score en función de la cantidad de datos. Sirve para ver, dado un modelo fijo, cómo reacciona a distintos tamaño del dataset. En particular, útil para diagnosticar alta varianza o modelo muy complejo (dado el tamaño de nuestro dataset).

Ensambles



ALTA VARIANZA - BAJO BIAS

Los algoritmos de bajo bias (alta varianza) tienden a ser más complejos, con una estructura subyacente flexible.

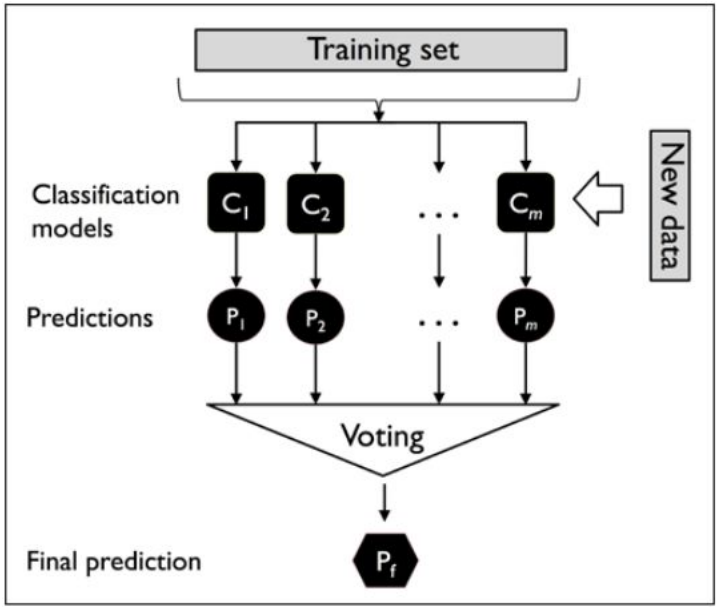
¿Podremos usar estos modelos para mejorar las predicciones?

Ensambles

Idea: entrenar muchos modelos y hacerlos votar.
La clasificación resultante es la que reciba más votos.

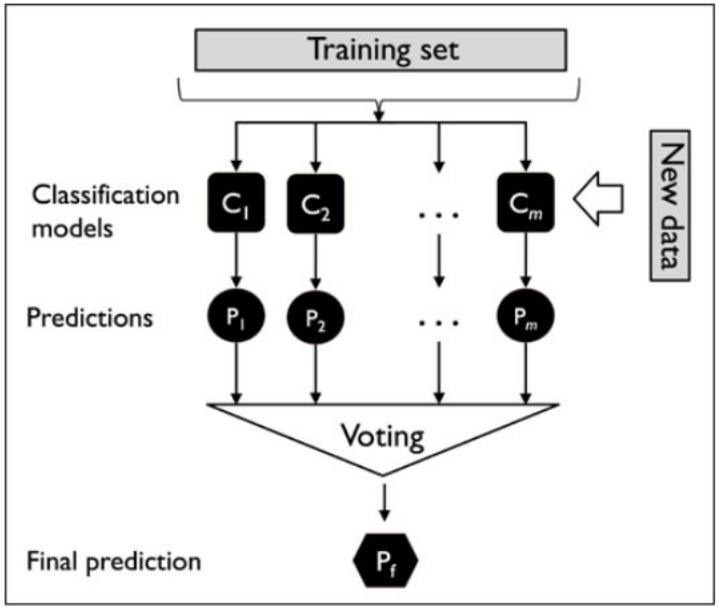
Ensambles

Idea: entrenar muchos modelos y hacerlos votar.
La clasificación resultante es la que reciba más votos.



Ensambles

Idea: entrenar muchos modelos y hacerlos votar.
La clasificación resultante es la que reciba más votos.



Aún mejor, si los modelos devuelven scores, se puede hacer una votación ponderada.

¿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.

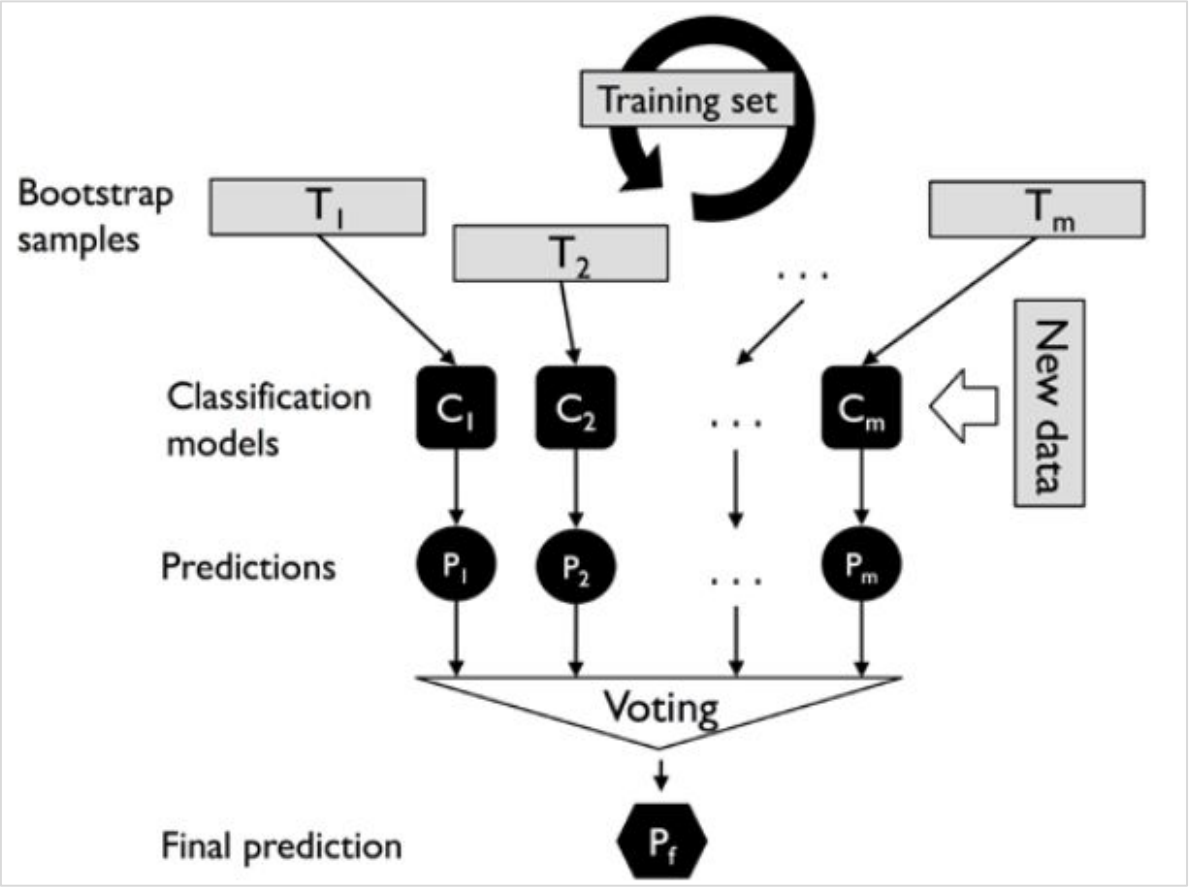
Bagging (Bootstrap Aggregation)

Muestreo con reemplazo
de las instancias

Bagging o Bootstrap Aggregation

El Bagging es una de las técnicas de construcción de conjuntos que también se conoce como Agregación Bootstrap.

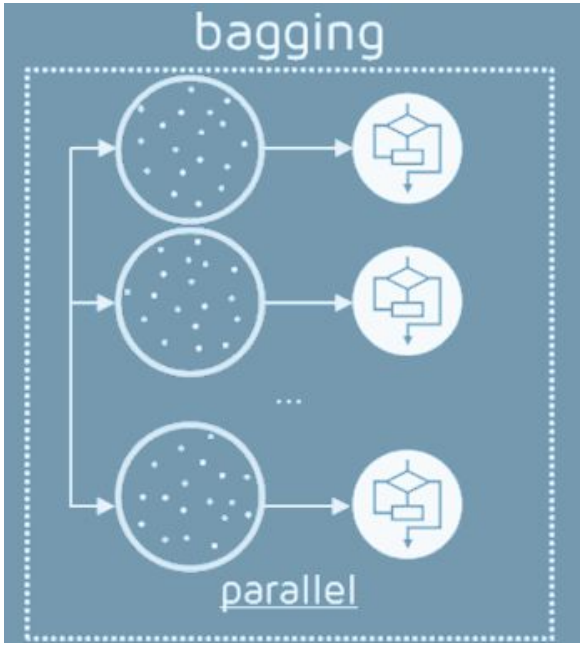
Bagging


Training set
Bootstrap samples
T_1
T_2
...
T_m
Classification models
C_1
C_2
...
C_m
New data
Predictions
P_1
P_2
...
P_m
Voting
Final prediction
P_f

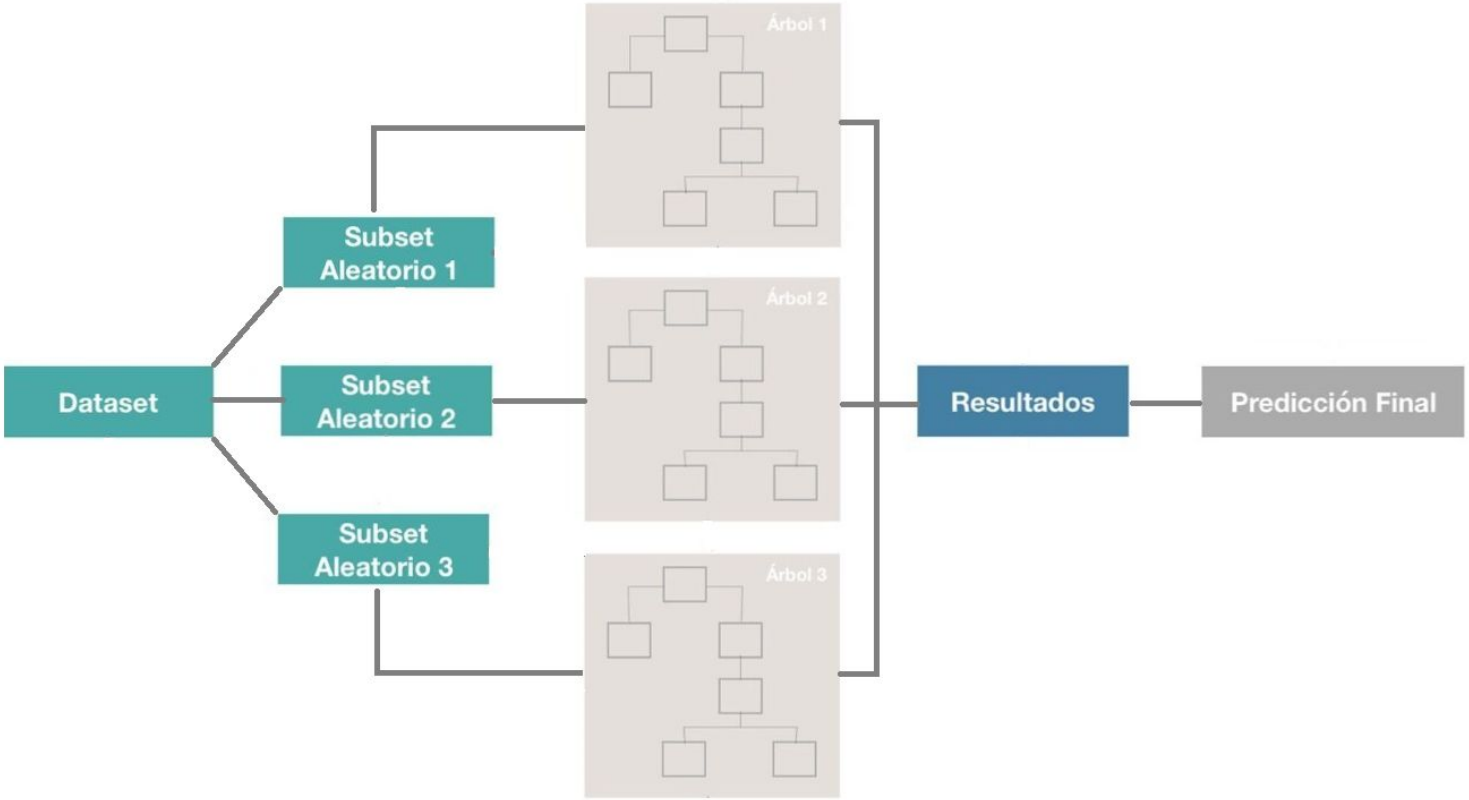
Bagging o Bootstrap Aggregation

El Bagging es una de las técnicas de construcción de conjuntos que también se conoce como Agregación Bootstrap.

- Dada una muestra de datos, se extraen varias muestras, *bootstrapped*
- Esta selección se realiza de manera aleatoria.
- Una vez que forman las muestras *bootstrapped*, se entrenan los modelos de manera separada. En general, estos modelos serán modelos con mucha varianza.
- La predicción de salida final se combina en las proyecciones de todos los submodelos.



Bagging o Bootstrap Aggregation

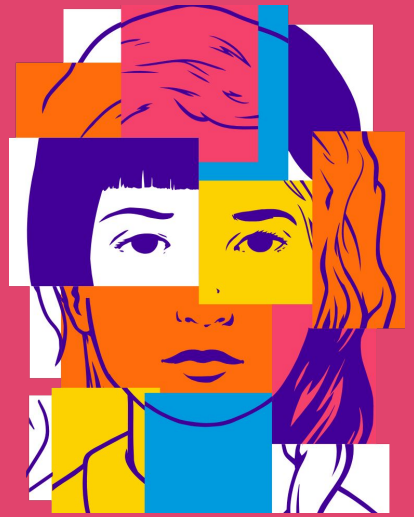


Bagging o Bootstrap Aggregation

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.

Random Forest

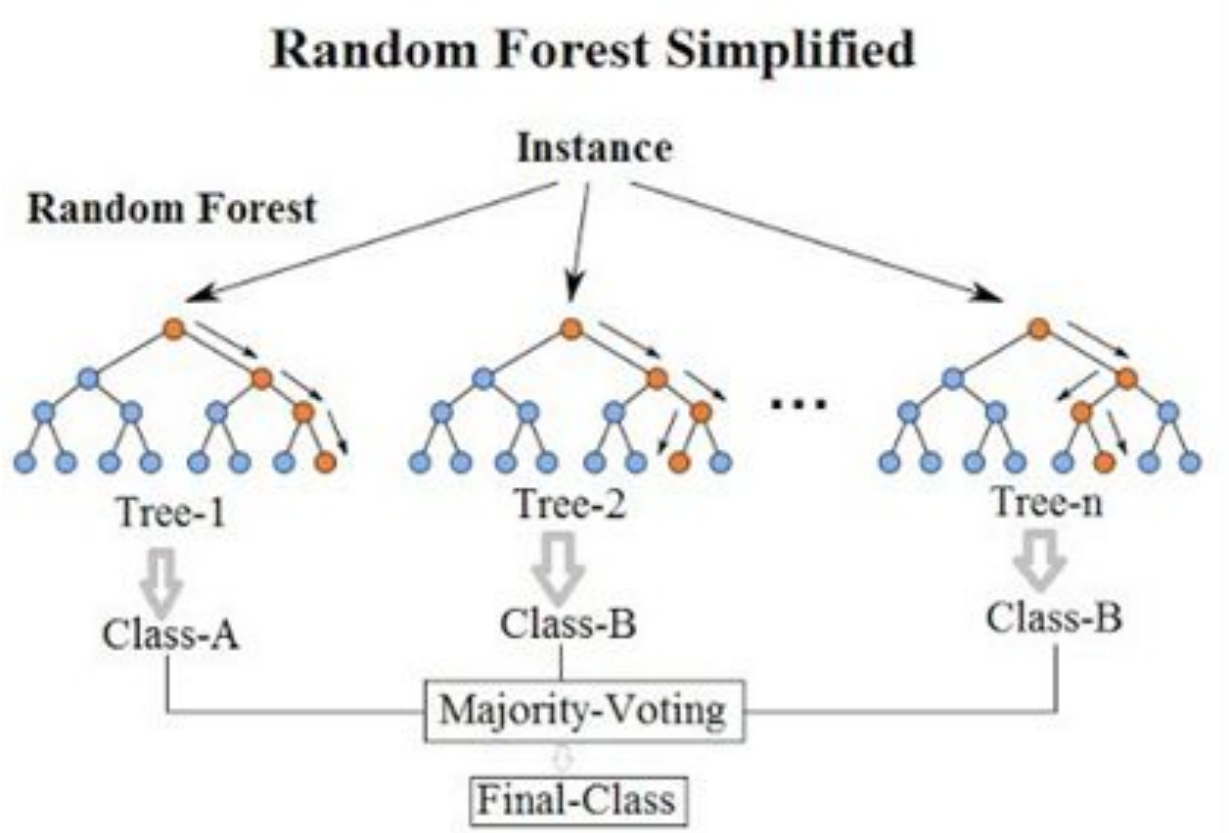


¿Cómo surge Random Forest?

Uno de los problemas que aparecía con la creación de un árbol de decisión es que si le damos la profundidad suficiente, el árbol tiende a “memorizar” las soluciones en vez de generalizar el aprendizaje (*overfitting*).

La solución para evitar esto es la de crear muchos árboles y que trabajen en conjunto.

Random Forest



Random Forest

Problema: si pocos atributos (features) son predictores fuertes, todos los árboles se van a parecer entre sí. Esos atributos terminarán cerca de la raíz para todos los conjuntos generados con bootstrap.

Random Forest

Problema: si pocos atributos (features) son predictores fuertes, todos los árboles se van a parecer entre sí. Esos atributos terminarán cerca de la raíz para todos los conjuntos generados con bootstrap.

Random Forest es igual a bagging, pero en cada nodo, hay que considerar sólo un subconjunto de *m* atributos elegidos al azar (random feature selection)

¿Cómo funciona Random Forest?

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

Random Forest • Ventajas

1. Bastante robusto frente a outliers y ruido
2. Provee buenos estimadores de error (oob_score) e importancia de variables
3. Si bien entrenar muchos árboles puede llevar mucho tiempo, es fácilmente paralelizable.

¡BREAK!

Ph. Credit: Drew Coffmann

Hands-on training

Hands-on training

DS_Encuentro_27_Bagging.ipynb

Recursos: Ensambles

Recursos

Si te quedaste con ganas de más...

- Ensemble Learning – The heart of Machine learning
- Ensemble Learning — Bagging and Boosting
- Random forest: El poder del ensamble (¡recomendado!)

Para la próxima

1. Ver los videos de la plataforma “Clasificación Avanzada: Ensambles Boosting”.

2. Completar los notebooks de hoy y atrasados.

ACÁMICA