ACÁMICA

# ¡Bienvenidos/as a Data Science!

# Agenda

Repaso de Bagging

Elección de Dataset a trabajar

Break

Hands on - Dataset a elección

Cierre

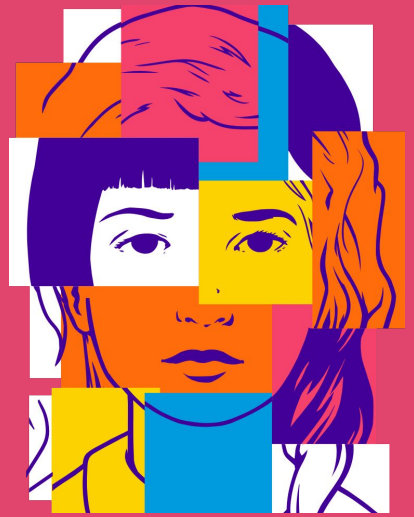
# ¿Cómo anduvieron?

# 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?

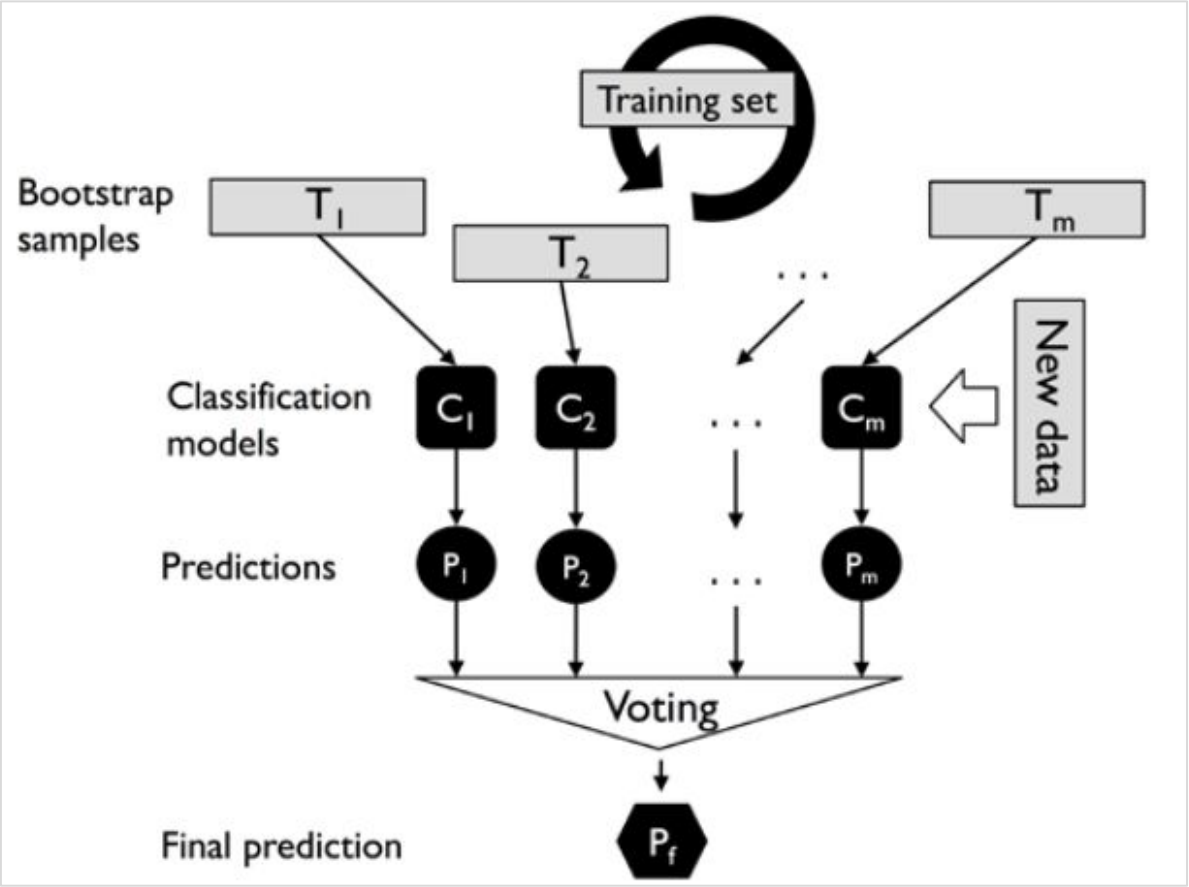
# Ensambles - Bagging



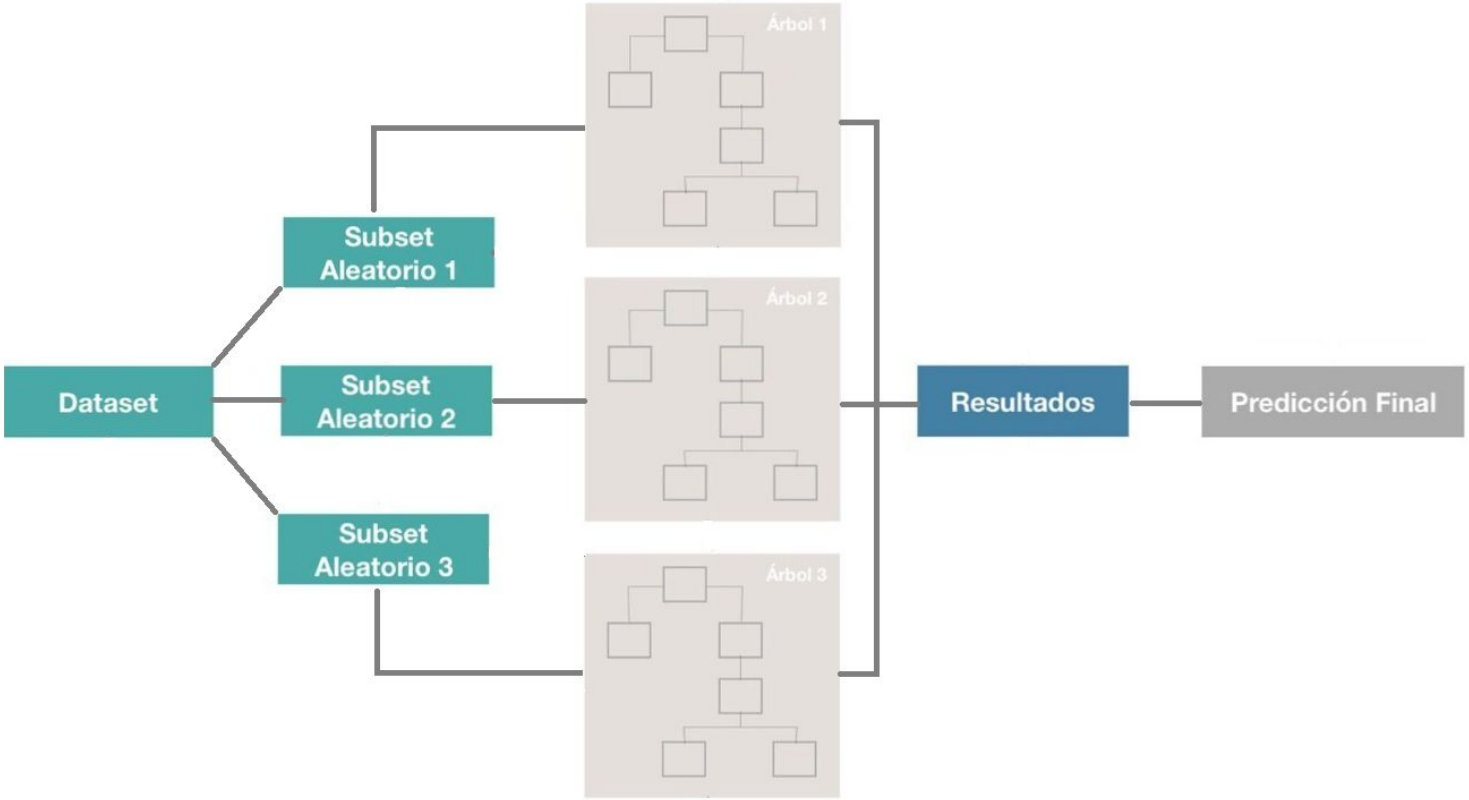
# Bagging (Bootstrap Aggregation)

Muestreo con reemplazo
de las instancias

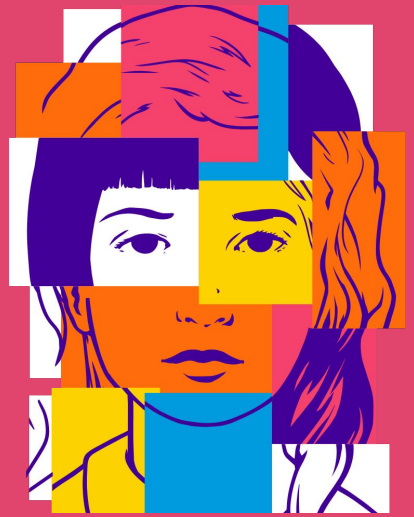
# Bagging


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

# Bagging o Bootstrap Aggregation



# Random Forest

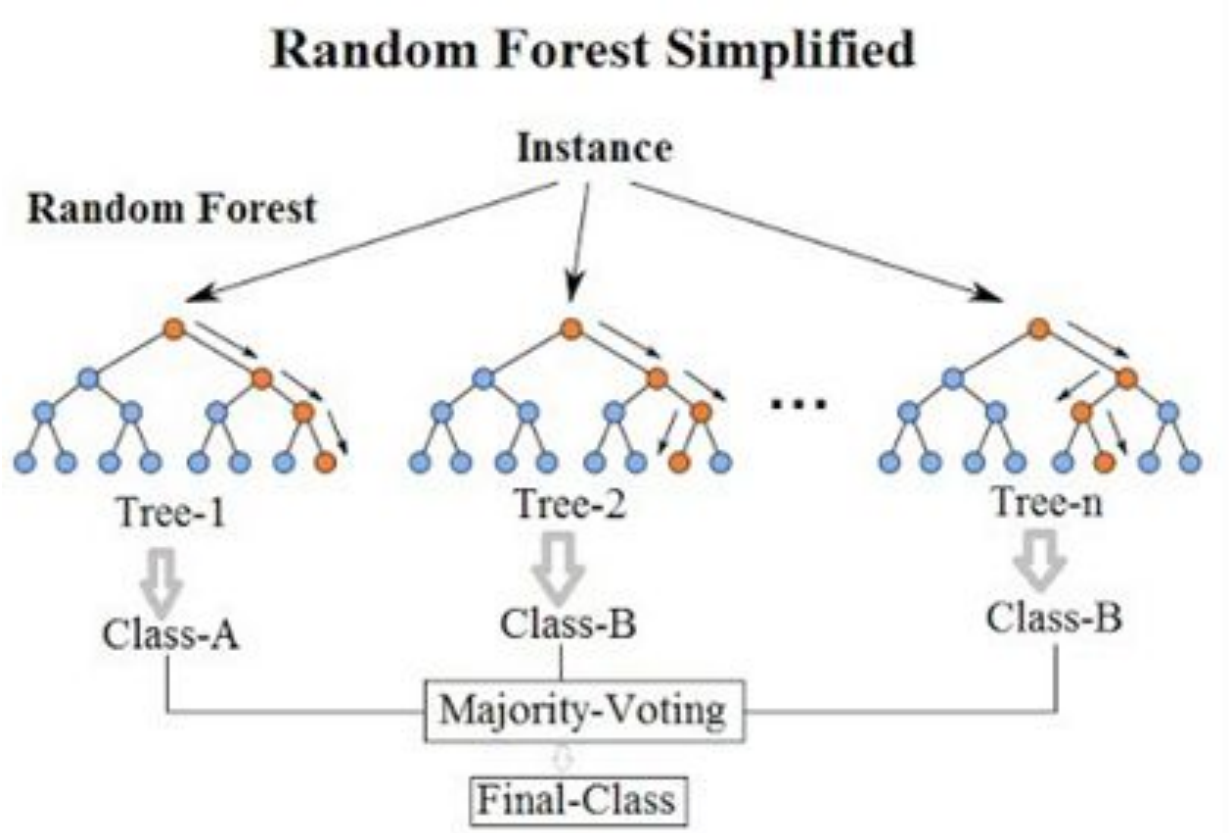


# ¿Cómo surge Random Forest?

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 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

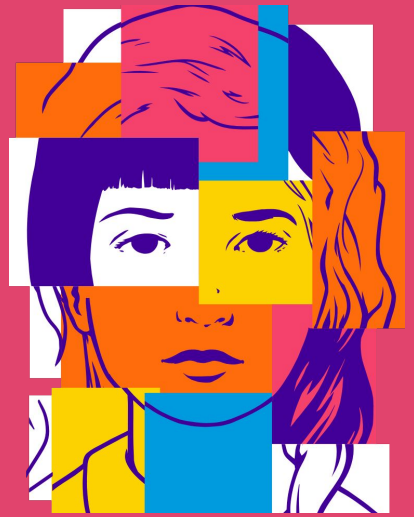
1. Bastante robusto frente a outliers y ruido
2. Provee buenos estimadores de error (oob_score) e importancia de variables
3. Si bien entrenar muchos árboles puede llevar mucho tiempo, es fácilmente paralelizable.

# ¡BREAK!

Ph. Credit: Drew Coffmann

# Manos a la obra:

**Ponemos en práctica todo lo que aprendimos hasta ahora.**



# Dataset: Opción 1

Trabajan con el dataset que les vamos a pasar nosotros sobre detección de gato o no gato en un conjunto de imágenes simples.

# Dataset: Opción 2

Trabajar con un dataset que sea de su interés. Si no tienen uno en mente, pueden buscar uno en Kaggle.

Vamos a pedirles que el problema elegido cumpla las siguientes condiciones:

- Sea un problema de clasificación
- No implique Procesamiento de lenguaje natural
- No implique trabajar con imágenes “complejas”

# Objetivo

**De mínima:** Terminar el dia con un modelo funcional del cual tengamos una idea de su desempeño medido en alguna de las métricas que estudiamos

**De máxima:** Explorar distintos modelos (KNN,SVM,RF) y/o hiperparámetros e intentar encontrar el mejor clasificador para nuestro problema.

# Hands-on training

**Hands-on training**

# DS_Encuentro_28_gatos.ipynb
# ó
# Dataset elegido

# Para la próxima

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

2. Completar los notebooks atrasados.

ACÁMICA