Bagging: Técnica de muestreo que se utiliza para construir un random forest.

Bagging: Le doy una cantidad de muestreo a cada árbol aleatorio. Establezco un mínimo y máximo tanto de filas como de columnas y le doy una muestra a cada árbol. Son arboles pequeños especializados cada uno en una parte del dataset. Las muestras se asignan con reemplazo.

Es paralelizadle. No necesitas terminar un árbol para hacer el siguiente. Para ser mas eficiente, se hacen varios árboles a la vez.

Boosting: Hace varios modelos de arboles de decisiones (Varios cortes). Cada árbol aprende del anterior pero al final te quedas con todos los árboles. En el primer modelo das una muestra aleatoria, las que haga mal se le dan al siguiente. No se puede paralelizar, necesitas un árbol para hacer el siguiente. No necesariamente los últimos son mejores, si un árbol da buenos resultados, tiene pocos malos resultados con los que hacer el siguiente, por lo que el siguiente no es tan bueno, al dar malos resultados, el siguiente es bueno porque tiene mucho de lo que aprender.

Hard voting: Se vota una u otra sumando los votos (0 o 1).

Soft voting: Se suman las probabilidades. Por lo que damos importancia a que tan seguros esdtan los arboles.

Por lo que si tenemos por ejemplo:

0.51 0.51 0.51 0 0

En hard voting es 1 ya que la mayoría es 1.

En soft voting es 1.5/3 = 0.31 -> Equivale a 0.

¿Cuántos arboles?

Si el dataset es pequeño, 50, si es grande, 100, 200, 1000. Raro menos de 50 y mas de 1000.

Lo mas difícil es tener un conjunto de arboles distintos. Lo que queremos es diversidad de ideas. El objetivo es diversidad.

La tupla celestial: Random Forest o XGBoost. Solian ganar competiciones.

**NOTA**: Nos saltamos 2 modelos KNN y SVM (SVC/SVR) porque están en desuso.

**ARGUMENTOS A PONER EN RANDOM FOREST.**

Class\_weight: Penaliza mas los errores de las clases minoritarias. Es buena costumbre. Por ejemplo, tengo 90 datos de perros y 10 de gatos, penalizo mas el error de gatos. Si no dices nada, todo es un error. Pero si le dices balanceado, penalizas mas los errores de la clase minoritaria, un error de gato equivale a 10 errores, mientras que de perro equivale a 1.

N\_jobs: -1 -> Usar todos los CPU’s.

En SVC incluir argumento de probability=True

GRDS: Se usa cross validation. Sacas los mejores hiperparametros.

NOTA SOBRE RANDOM FOREST: El modelo que aplica es el TreeClassifier. Para lo UNICO QUE NOS PODRIA SERVIR HACERLO A MANI CON Bagging Classifier es que podemos en lugar de hacer varias treeclassifier, hacer varias logistic regression.

Texto

Descripción generada automáticamente

Fetch\_openml = Es un repositorio

Ir a Openml.org: Veo datasets que me interesen. Tomamos el nombre y lo importamos. Como vemos en este ejemplo:

Texto

Descripción generada automáticamente

¿Cuáles son las variables mas importantes?

N\_estimators es la MAS IMPORTANTE.

Texto

Descripción generada automáticamente

Max\_Features en auto es igual al sqrt.

XGBOOST: Se diferencia de otros en que no se importa de sklearn. Es independiente pero de allí todo es igual.

Argumento de modelos class weight: Podemos pedirle que haga un balance. Busca la proporción entre ambos y penaliza mas los errores de la clase minoritaria de acuerdo a esa proporción. Es recomendable hacerlo aunque los modelos de arboles se comportan muy bien con datasets desbalanceados.

Argumentos para optimizar Random Forest con Gridsearch

N\_estimators [50,100,1000]

Max\_depth [3,5,7,9]

Max\_features [0.2,0.4,0.6]

Max\_samples [0.2,0.6]

Min\_sample\_leaf [5,10,15,20]