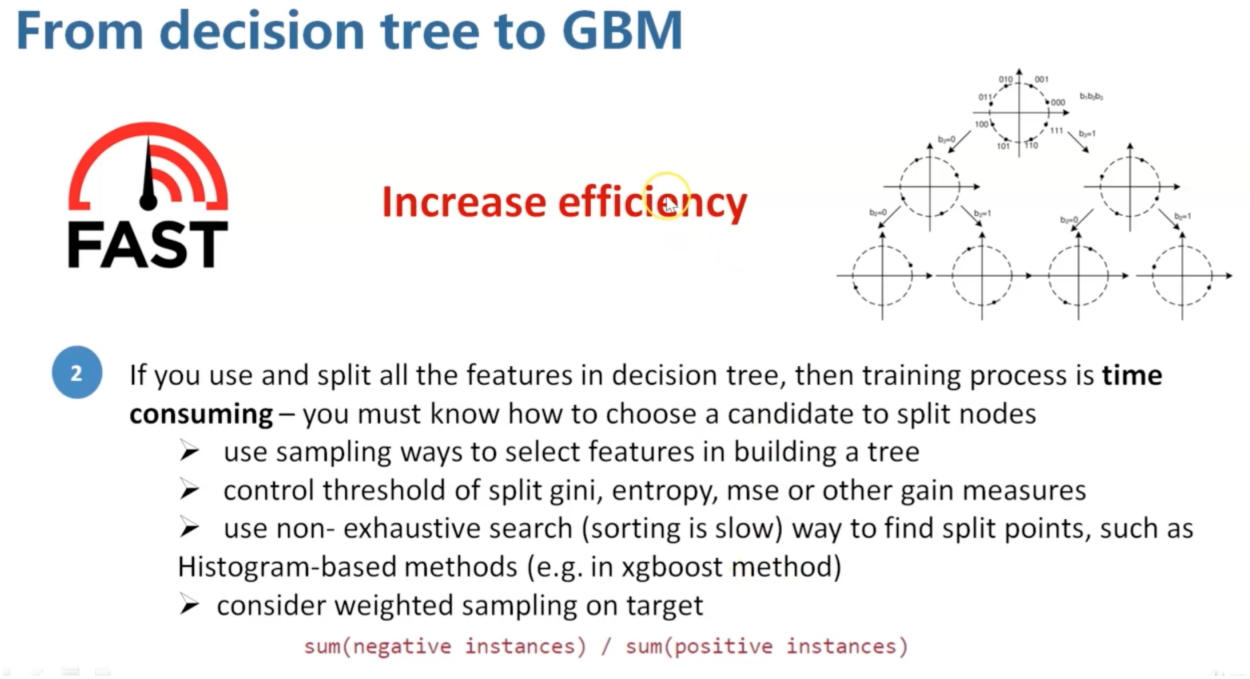
Entendiendo las debilidades de un árbol de decisión & GBM

De árboles de decisión hasta un GBM (gradient boosting model)

Si un nodo de un árbol es particionado en hojas muy granulares, entonces el modelo de árbol es muy probable que se sobreajuste (overfitting), por lo que debo tener en cuenta lo siguiente:

- El número de observaciones por hoja no debe ser muy bajo,
- La profundidad del arbol no debe ser muy grande,
- El número de hojas puede causar un problema de sobreajuste,
- Tener en cuenta la regularización (prune).

Un modelo tipo GBM ataca los problemas anteirores reduciendo la complejidad



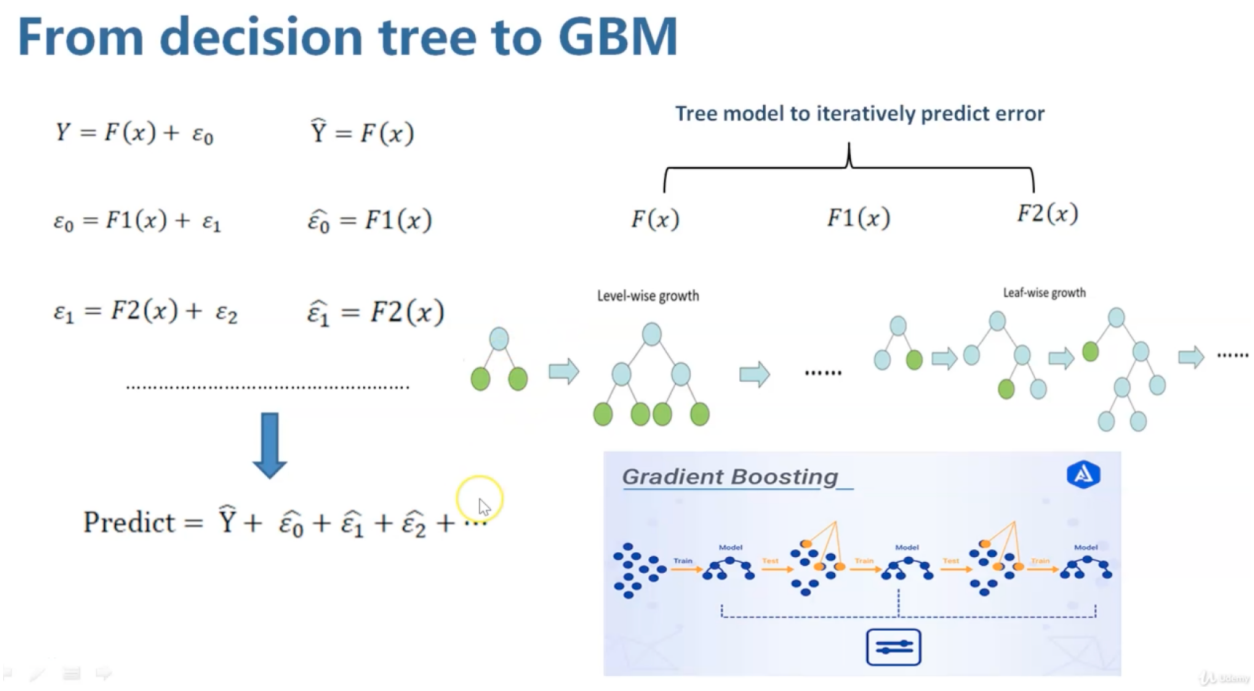
Si usamos y dividimos todas las variables de entrada al modelo (features), entonces el proceso de entrenamiento es demasiado costoso en tiempo, debo saber como escoger un candidato para dividir los nodos.

- Utilizar formas de muestreo para seleccionar características en la construcción de un árbol,
- Umbral de control del threshold del gini, entropía, mse u otras medidas de ganancia,
- Usa un método de búsqueda no exahustivo para encontrar los puntos de separación de los nodos (métodos sort son lentos), como los métodos basados en histogramas (e.g. in xgboost method).
- Considerar un muestreo ponderado en el target. $\frac{\sum negative\ instances}{\sum positive\ instances}$

Construir un GBM desde un árbol de decisión

En secciones anterior, se definió detalladamente el como es la lógica de construcción de un modelo GBM, como lo muestra la siguiente imágen ya vista anteriormente, en donde se puede resumir como:

1. Construir un primer modelo el cual prediga la target definida,
2. Construir un segundo modelo que prediga el error del j-1 modelo, siendo j el modelo actual,
3. Repetir el paso 2 iterativamente.



En pseudo-código, construímo un modelo de árbol de clasificación llamada T_0 para predecir una target, declaramos que el modelo ensamble: $GBM\ Model = T_0$, usamos el módelo clasificador para predecir (con el método $predict\_proba()$), para obtener p_target, por cada observación calculamos un error $error_0 = target - p\_target$, una vez hecho eso, hacemos lo siguiente:

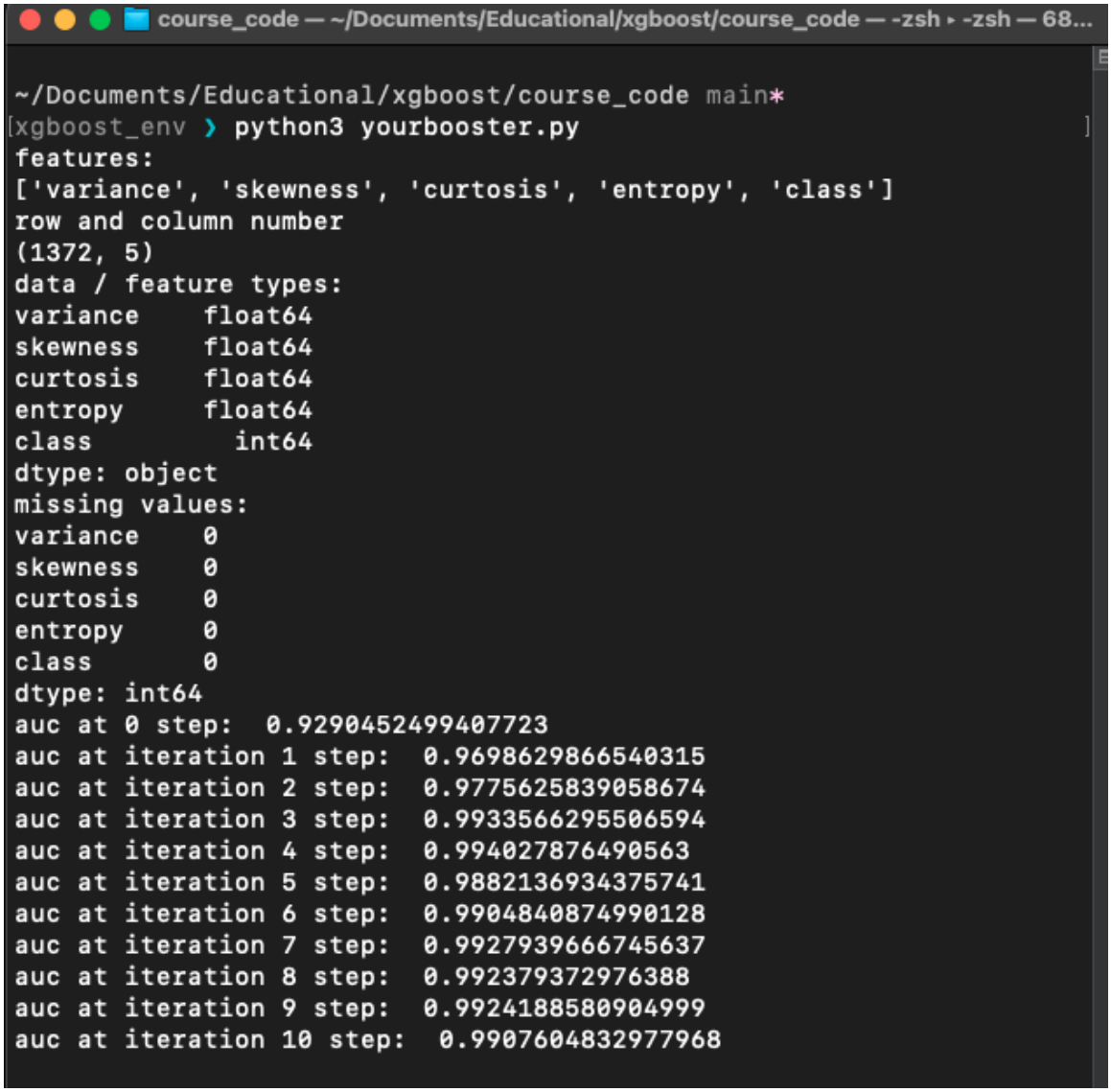
- Construímos un modelo tipo árbol de **regresión** el cual será $T\_(j+1)$ para predecir $error\_j$ y así obtener p_error_j,
- $GBM\_Model = GBM\_Model + T\_(j+1)$,

- $error\_(j+1) = error\_j - p\_error\_j,$
- $j = j + 1$

En código

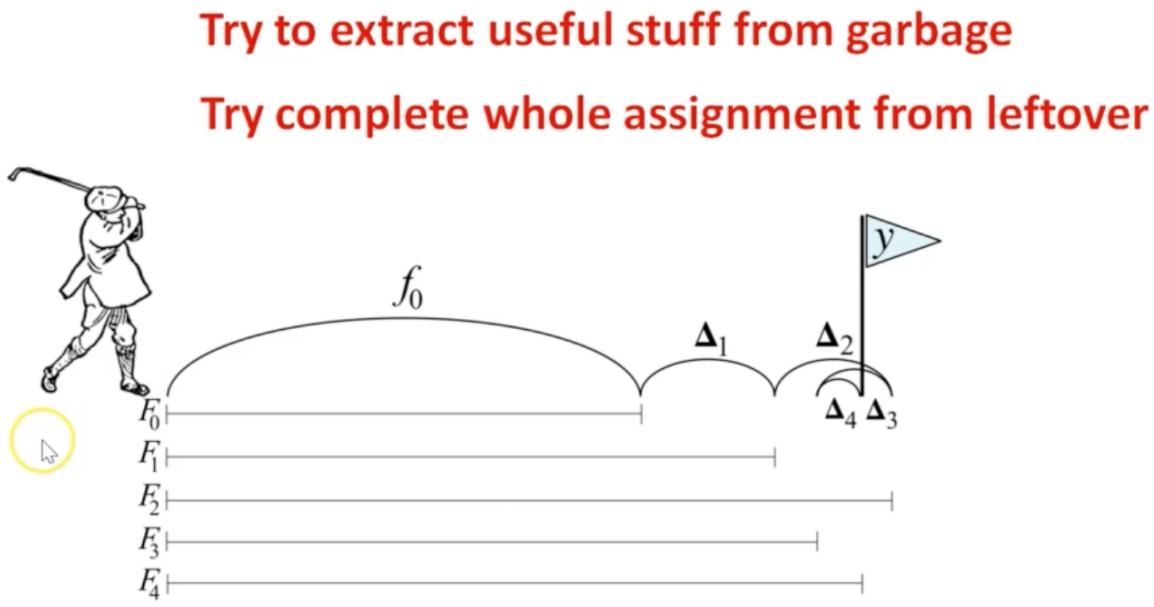
```
it_times = 10
tree_entropy = DecisionTreeClassifier(criterion = 'entropy', random_state = 12,
            max_depth = 3, min_samples_leaf = 5)

tree_entropy.fit(X_train, y_train)

prediction = tree_entropy.predict_proba(X_train)[:, 1]

prediction_test = tree_entropy.predict(X_test)
fpr, tpr, thresholds = metrics.roc_curve(y_test, prediction_test)
roc_auc = metrics.auc(fpr, tpr)
print ("auc at 0 step: ", roc_auc)

err = y_train - prediction
booster = [tree_entropy]

for j in range(it_times):
    j = j + 1
    tree_model = DecisionTreeRegressor(criterion = 'mse', random_state = 12,
            max_depth = 3, min_samples_leaf = 5)

    tree_model.fit(X_train, err)
    err_predict = tree_model.predict(X_train)
    prediction = prediction + err_predict
    err = err - err_predict
    booster.extend([tree_model])

    err_predict_test = tree_model.predict(X_test)
    prediction_test = prediction_test + 0.02*err_predict_test
    fpr, tpr, thresholds = metrics.roc_curve(y_test, prediction_test)
    roc_auc = metrics.auc(fpr, tpr)
    print ("auc at iteration " + str(j) + " step: ", roc_auc)
```

Output:

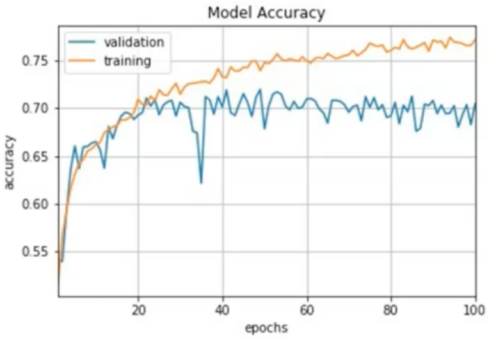


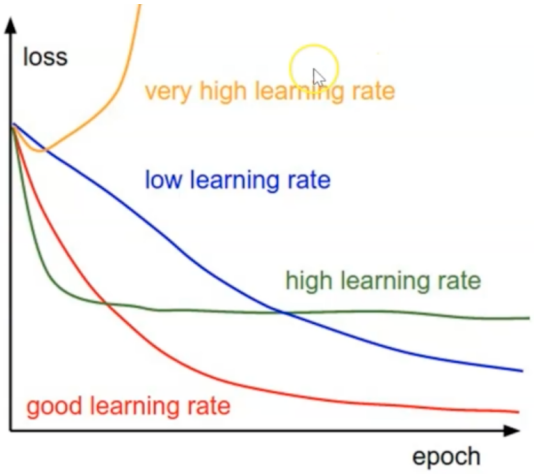
Por qué una estrategia GBM es mejor que un árbol de decisión?

1. Dado que GBM en un tipo de ensamble aditivo de módelos de árbol, en otras palabras, usa el error de la predicción original para corregir la predicción del arbol anterior, estro incrementará el accuracy del modelo.



2. Porque el GBM es un modelado iterativo y considera el 'learning rate', subsamples, etc, esto mejorará la eficiencia del algoritmo en términos de convergencia y velocidad.





3. GBM considera el número de iteraciones (el número de modelos de árbol) a ser usados, esto afecta tanto el accuracy, la estabilidad y la eficiencia (velocidad).