XGBoost: Sklearn’s API versus Booster nativo

XGBoost es una biblioteca de aprendizaje automático de código abierto que se utiliza ampliamente para la optimización de modelos de aprendizaje automático. XGBoost proporciona dos implementaciones diferentes del algoritmo Gradient Boosting: una implementación nativa en la biblioteca XGBoost, y otra implementación que se puede utilizar a través de la interfaz de programación de aplicaciones (API) de scikit-learn.

La principal ventaja de utilizar la implementación nativa de XGBoost es que está diseñada específicamente para trabajar con la biblioteca XGBoost y se puede aprovechar al máximo el rendimiento de la biblioteca. Además, la implementación nativa de XGBoost proporciona un mayor control sobre el entrenamiento del modelo y permite acceder a una mayor cantidad de opciones y parámetros que pueden mejorar el desempeño del modelo.

Por otro lado, una de las principales ventajas de utilizar la implementación de XGBoost a través de la API de scikit-learn es que se integra perfectamente con el resto de las herramientas y bibliotecas de scikit-learn. Esto facilita el preprocesamiento de los datos y el uso de las herramientas de scikit-learn para tareas como validación cruzada y selección de modelos. Además, la API de scikit-learn es más fácil de usar para aquellos que ya están familiarizados con scikit-learn.

En general, no hay una respuesta clara a cuál de las dos implementaciones es mejor, ya que ambas tienen sus propias ventajas y desventajas. La elección depende del contexto específico del problema y de las preferencias del usuario.

Sklearn’s XGBoost (clasificador o regresor) pueden ser convertidos a un boost:

- booster = clf.get_booster()

En las diferencias más notorias, se listan:

- Número de estimadores / Rounds de entrenamiento

- Sklearn: n_estimator,
- Native booster: num_boost_round.

- Forma de ingestar datos:
  - Native booster: A través del objeto DMatrix, el cual es un tipo de set de datos especial necesario para el uso del booster nativo de XGBoost. Es un tipo de objeto optimizado para la velocidad de entrenamiento.

```
dtrain = xgb.DMatrix(data=X_train.values,
                     feature_names=X_train.columns,
                     label=y_train.values)

dvalid = xgb.DMatrix(data=X_valid.values,
                     feature_names=X_valid.columns,
                     label=y_valid.values)
```

dentro de sus parámetros, podemos especificar:

- data: debe ser un arreglo de numpy o un dataframe,
- label: (lista uni-dimensional de numpy o dataframe, opcional) etiquetas del set de entrenamiento.
- missing: (float opcional) Valor que seteará en caso de que encuentre datos nulos.

- Entrenar el modelo
  - Native booster: .train() o .cv(), este último es usado para el cross validation.
  - Sklearn: fit().

- Métricas de evaluación:

- El parámetro eval_metric para el booster nativo está dentro del diccionario de parámetros, mientras que en la API de Sklearn está dentro del método .fit().

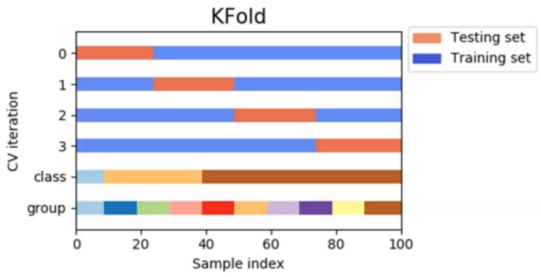
evals = [(dvalid, 'valid'), (dtrain, 'train')] will enable user to watch the process and performance of model training

```
params = {'learning_rate' : 0.03,   'max_depth' : 7,
   'objective' : 'binary:logistic',    'min_child_weight' : 5,
   'gamma' : 0.05,      'subsample': 0.8,
   'colsample_bytree': 0.8,
   'seed' : 12,
   'eval_metric' : 'auc'}
```

```
mod = xgb.train(params = params,
          dtrain=dtrain,
          num_boost_round = 1200,
          early_stopping_rounds=50,
          evals=[(dvalid,'valid'), (dtrain,'train')],
          verbose_eval=20)
```

- Tuning de parámetros en el booster nativo con el método .cv
  - El usar el método .cv entregará como resultado un historial de la evaluación del entrenamiento, permitiendo poder esocger el mejor round de acuerdo al método evaluado.

```
xgboost.cv(params, dtrain, num_boost_round=10, nfold=3, stratified=False, folds=None, metrics=(), obj=None,
 feval=None, maximize=False, early_stopping_rounds=None, fpreproc=None, as_pandas=True, verbose_eval=
None, show_stdv=True, seed=0, callbacks=None, shuffle=True)
```

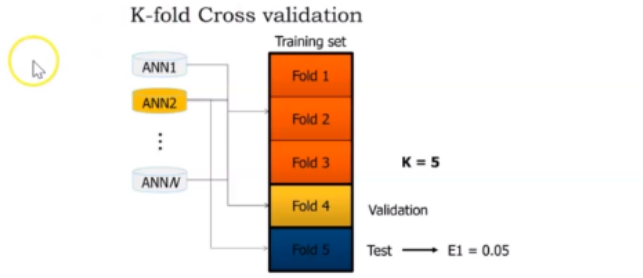


En donde los parámetros a tener en consideración son:

- num_boost_round [degfault=10]: Número de iteraciones de boosting, el número de árboles usados para el algoritmo de boosting, al igual que el n_estimators en el xgboost.
- nfold [defualt=3]: Número de golds en el cross validation.
- metrics () = El set de métricas de evaluación usadas en el cross validation, como por ejemplo ‘auc’.

- folds: Instancia de KFold o StratifiedKFold o una lista de índices de pliegues, KFolds de Sklearn u objeto estratificado. Por ejemplo:

```
from sklearn.model_selection import, StratifiedKFold, KFold, RepeatedKFold, GridSearchCV
kf = KFold(n_splits = 5, shuffle = True)
```



- verbose_eval (valor booleano): Bandera para determinar si dar el output del progreso o no.
- show_std [default= True]: muestra la desviación estándar en progreso, los resultados no son afectados por esta bandera.