XGBoost - Spark

주의!!!

> 이 예제는 일반 컴퓨터 시스템에서는 구동되지 않습니다. Spark 기반의 클라우드 분산 시스템에서 XGBoost 모델을 가동하는데 참고하기 위한 가이드라인을 제시하기 위한 코드 입니다.
>
> 클라우드 분산 시스템은 Microsoft Azure, Amazon Web Service 등을 활용하는 기업등이 대용량 데이터에 대한 학습을 수행하기 위해 구축하는 멀티 컴퓨팅 환경입니다.
>
> 실무에서 구축하는 컴퓨터 시스템에서 사용되는 코드의 최소 가이드를 제시하기 위함이니 코드의 전개를 참고만 하세요.

#01. 패키지 참조

아래의 패키지 설치가 필요하다.

```
$ pip install pyspark
```

```python
from pandas import read_csv
from pyspark.sql import SparkSession
from pyspark.ml import Pipeline
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.ml.feature import VectorAssembler
from xgboost.spark import SparkXGBClassifier
```

#02. 데이터 가져오기

```python
origin = read_csv('breast_cancer.csv')
origin.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   mean radius              569 non-null    float64
 1   mean texture             569 non-null    float64
 2   mean perimeter           569 non-null    float64
 3   mean area                569 non-null    float64
 4   mean smoothness          569 non-null    float64
 5   mean compactness         569 non-null    float64
 6   mean concavity           569 non-null    float64
 7   mean concave points      569 non-null    float64
 8   mean symmetry            569 non-null    float64
 9   mean fractal dimension   569 non-null    float64
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
 30  target                   569 non-null    int64
dtypes: float64(30), int64(1)
memory usage: 137.9 KB
```

#03. 학습 모델 적합

SparkSession 설정

> 세션이란 컴퓨터에서 하나의 작업단위를 의미합니다.

```
spark = SparkSession.builder \
    .appName("SparkXGBGridSearch") \
    .getOrCreate()
```

데이터 로드

`spark.read.csv()` 함수는 모든 데이터를 string 타입으로 가져오기 때문에 pandas dataframe을 변환하는 것이 더 좋습니다.

```
data = spark.createDataFrame(origin)
data
```

```
DataFrame[mean radius: double, mean texture: double, mean perimeter: double, mean
area: double, mean smoothness: double, mean compactness: double, mean concavity:
double, mean concave points: double, mean symmetry: double, mean fractal dimension:
double, radius error: double, texture error: double, perimeter error: double, area
error: double, smoothness error: double, compactness error: double, concavity
error: double, concave points error: double, symmetry error: double, fractal
dimension error: double, worst radius: double, worst texture: double, worst
perimeter: double, worst area: double, worst smoothness: double, worst compactness:
double, worst concavity: double, worst concave points: double, worst symmetry:
double, worst fractal dimension: double, target: bigint]
```

분류기 객체 생성

GPU를 사용할 경우 device 파라미터의 값을 "cuba"로 지정

```
xgb = SparkXGBClassifier(num_workers=2, features_col='features',
label_col='target', use_gpu=False, device="cpu")
xgb
```

```
SparkXGBClassifier_ed3d0da35c36
```

분류기 객체의 모든 property 확인

언더바(`_`)가 없는 값이 하이퍼파라미터 입니다.

```
print(dir(xgb))
```

```
['__abstractmethods__', '__annotations__', '__class__', '__class_getitem__',
'__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__',
'__getattribute__', '__gt__', '__hash__', '__init__', '__init_subclass__',
'__le__', '__lt__', '__module__', '__ne__', '__new__', '__orig_bases__',
'__parameters__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__',
'__sizeof__', '__slotnames__', '__slots__', '__str__', '__subclasshook__',
'__weakref__', '_abc_impl', '_convert_to_sklearn_model', '_copyValues',
'_copy_params', '_create_pyspark_model', '_defaultParamMap', '_dummy', '_fit',
'_gen_fit_params_dict', '_gen_predict_params_dict', '_gen_xgb_params_dict',
'_get_distributed_train_params', '_get_fit_params_default',
'_get_predict_params_default', '_get_xgb_parameters', '_get_xgb_params_default',
'_get_xgb_train_call_args', '_input_kwargs', '_is_protocol', '_paramMap',
'_params', '_prepare_input', '_prepare_input_columns_and_feature_prop',
'_pyspark_model_cls', '_query_plan_contains_valid_repartition', '_randomUID',
'_repartition_needed', '_resetUid', '_resolveParam', '_set', '_setDefault',
'_set_fit_params_default', '_set_predict_params_default',
'_set_xgb_params_default', '_shouldOwn', '_skip_stage_level_scheduling',
'_testOwnParam', '_try_stage_level_scheduling', '_validate_gpu_params',
'_validate_params', '_xgb_cls', 'arbitrary_params_dict', 'base_margin_col',
'base_score', 'booster', 'callbacks', 'clear', 'colsample_bylevel',
'colsample_bynode', 'colsample_bytree', 'copy', 'device', 'early_stopping_rounds',
'enable_sparse_data_optim', 'eval_metric', 'explainParam', 'explainParams',
'extractParamMap', 'feature_names', 'feature_types', 'feature_weights',
'featuresCol', 'features_cols', 'fit', 'fitMultiple', 'force_repartition', 'gamma',
'getFeaturesCol', 'getLabelCol', 'getOrDefault', 'getParam', 'getPredictionCol',
'getProbabilityCol', 'getRawPredictionCol', 'getValidationIndicatorCol',
'getWeightCol', 'grow_policy', 'hasDefault', 'hasParam', 'importance_type',
'interaction_constraints', 'isDefined', 'isSet', 'iteration_range', 'labelCol',
'learning_rate', 'load', 'logger', 'max_bin', 'max_cat_threshold',
'max_cat_to_onehot', 'max_delta_step', 'max_depth', 'max_leaves',
'min_child_weight', 'missing', 'monotone_constraints', 'multi_strategy',
'n_estimators', 'num_parallel_tree', 'num_workers', 'objective', 'params',
'pred_contrib_col', 'predictionCol', 'probabilityCol', 'qid_col', 'random_state',
'rawPredictionCol', 'read', 'reg_alpha', 'reg_lambda',
'repartition_random_shuffle', 'sampling_method', 'save', 'scale_pos_weight', 'set',
'setParams', 'set_device', 'subsample', 'tree_method', 'uid', 'use_gpu',
'validate_parameters', 'validationIndicatorCol', 'verbose', 'verbosity',
'weightCol', 'write', 'xgb_model']
```

하이퍼파라미터 그리드 생성

테스트할 파라미터들과 값을 Map 객체 형태로 나열합니다.

> Map은 파이썬의 딕셔너리와 비슷한 자료 구조

```
param_map = ParamGridBuilder() \
    .addGrid(xgb.max_depth, [5, 7]) \
    .addGrid(xgb.n_estimators, [50, 100]) \
    .addGrid(xgb.learning_rate, [0.1, 0.15]) \
    .build()

param_map
```

```
[{Param(parent='SparkXGBClassifier_ed3d0da35c36', name='max_depth', doc='Refer to
XGBoost doc of xgboost.sklearn.XGBClassifier for this param max_depth'): 5,
  Param(parent='SparkXGBClassifier_ed3d0da35c36', name='n_estimators', doc='Refer
to XGBoost doc of xgboost.sklearn.XGBClassifier for this param n_estimators'): 50,
  Param(parent='SparkXGBClassifier_ed3d0da35c36', name='learning_rate', doc='Refer
to XGBoost doc of xgboost.sklearn.XGBClassifier for this param learning_rate'):
0.1},
 {Param(parent='SparkXGBClassifier_ed3d0da35c36', name='max_depth', doc='Refer to
XGBoost doc of xgboost.sklearn.XGBClassifier for this param max_depth'): 5,
  Param(parent='SparkXGBClassifier_ed3d0da35c36', name='n_estimators', doc='Refer
to XGBoost doc of xgboost.sklearn.XGBClassifier for this param n_estimators'): 50,
  Param(parent='SparkXGBClassifier_ed3d0da35c36', name='learning_rate', doc='Refer
to XGBoost doc of xgboost.sklearn.XGBClassifier for this param learning_rate'):
0.15},
 {Param(parent='SparkXGBClassifier_ed3d0da35c36', name='max_depth', doc='Refer to
XGBoost doc of xgboost.sklearn.XGBClassifier for this param max_depth'): 5,
  Param(parent='SparkXGBClassifier_ed3d0da35c36', name='n_estimators', doc='Refer
to XGBoost doc of xgboost.sklearn.XGBClassifier for this param n_estimators'): 100,
  Param(parent='SparkXGBClassifier_ed3d0da35c36', name='learning_rate', doc='Refer
to XGBoost doc of xgboost.sklearn.XGBClassifier for this param learning_rate'):
0.1},
 {Param(parent='SparkXGBClassifier_ed3d0da35c36', name='max_depth', doc='Refer to
XGBoost doc of xgboost.sklearn.XGBClassifier for this param max_depth'): 5,
  Param(parent='SparkXGBClassifier_ed3d0da35c36', name='n_estimators', doc='Refer
to XGBoost doc of xgboost.sklearn.XGBClassifier for this param n_estimators'): 100,
  Param(parent='SparkXGBClassifier_ed3d0da35c36', name='learning_rate', doc='Refer
to XGBoost doc of xgboost.sklearn.XGBClassifier for this param learning_rate'):
0.15},
 {Param(parent='SparkXGBClassifier_ed3d0da35c36', name='max_depth', doc='Refer to
XGBoost doc of xgboost.sklearn.XGBClassifier for this param max_depth'): 7,
  Param(parent='SparkXGBClassifier_ed3d0da35c36', name='n_estimators', doc='Refer
to XGBoost doc of xgboost.sklearn.XGBClassifier for this param n_estimators'): 50,
  Param(parent='SparkXGBClassifier_ed3d0da35c36', name='learning_rate', doc='Refer
to XGBoost doc of xgboost.sklearn.XGBClassifier for this param learning_rate'):
0.1},
 {Param(parent='SparkXGBClassifier_ed3d0da35c36', name='max_depth', doc='Refer to
XGBoost doc of xgboost.sklearn.XGBClassifier for this param max_depth'): 7,
  Param(parent='SparkXGBClassifier_ed3d0da35c36', name='n_estimators', doc='Refer
to XGBoost doc of xgboost.sklearn.XGBClassifier for this param n_estimators'): 50,
  Param(parent='SparkXGBClassifier_ed3d0da35c36', name='learning_rate', doc='Refer
to XGBoost doc of xgboost.sklearn.XGBClassifier for this param learning_rate'):
0.15},
 {Param(parent='SparkXGBClassifier_ed3d0da35c36', name='max_depth', doc='Refer to
XGBoost doc of xgboost.sklearn.XGBClassifier for this param max_depth'): 7,
  Param(parent='SparkXGBClassifier_ed3d0da35c36', name='n_estimators', doc='Refer
to XGBoost doc of xgboost.sklearn.XGBClassifier for this param n_estimators'): 100,
  Param(parent='SparkXGBClassifier_ed3d0da35c36', name='learning_rate', doc='Refer
to XGBoost doc of xgboost.sklearn.XGBClassifier for this param learning_rate'):
0.1},
 {Param(parent='SparkXGBClassifier_ed3d0da35c36', name='max_depth', doc='Refer to
XGBoost doc of xgboost.sklearn.XGBClassifier for this param max_depth'): 7,
  Param(parent='SparkXGBClassifier_ed3d0da35c36', name='n_estimators', doc='Refer
to XGBoost doc of xgboost.sklearn.XGBClassifier for this param n_estimators'): 100,
  Param(parent='SparkXGBClassifier_ed3d0da35c36', name='learning_rate', doc='Refer
to XGBoost doc of xgboost.sklearn.XGBClassifier for this param learning_rate'):
0.15}]
```

교차검증 설정

`xgb.getLabelCol()`은 `분류기 객체 생성`단계에서 지정한 label_col의 값을 반환합니다.

```
# 교차 검증 설정
evaluator = BinaryClassificationEvaluator(labelCol=xgb.getLabelCol())
evaluator
```

```
BinaryClassificationEvaluator_671275772c4c
```

```
cross_validator = CrossValidator(estimator=xgb,
                                 estimatorParamMaps=param_map,
                                 evaluator=evaluator,
                                 numFolds=3)
cross_validator
```

```
CrossValidator_4d7c40f736e4
```

변수 어셈블리 변환

데이터프레임의 각 행을 하나의 리스트로 결합하여 하나의 변수를 갖는 데이터프레임으로 변환하는 과정

`features`는 통합된 변수의 이름

```
featuresCols = data.columns
featuresCols.remove('target')

vectorAssembler = VectorAssembler(inputCols=featuresCols, outputCol="features")
vectorAssembler
```

```
VectorAssembler_8e93de093ecb
```

파이프라인 설정

변수 변환과정과 학습모델 적합과정의 순서를 지정해 줌

```
pipeline = Pipeline(stages=[vectorAssembler, cross_validator])
pipeline
```

```
Pipeline_4caeb86489f1
```

모델 학습

단일 머신에서 가동할 경우 아래 코드에서 에러가 발생합니다.

`ServerSocket`에 대한 네트워크 에러이므로 단일 머신에서는 확인이 어려운점을 이해하시기 바랍니다.

```
# 모델 학습
cv_model = pipeline.fit(data)
cv_model
```

#04. 모델 성능 평가

```
# 최적 모델 확인
best_model = cv_model.bestModel

# 최적 하이퍼파라미터 확인
best_max_depth = best_model.getMaxDepth()
best_num_round = best_model.getNumRound()
best_eta = best_model.getEta()

print(f"Best Max Depth: {best_max_depth}")
print(f"Best Num Round: {best_num_round}")
print(f"Best Eta: {best_eta}")
```

```
# SparkSession 종료
spark.stop()
```