doGridSearch : 격자탐색

- 주어진 범위 내의 모든 값을 입력해 하이퍼파라미터를 최적화하는 방법.

```
def optimizeHyperparameter(self, name, code, start_date, end_date, lags_count=5) :
        a_predictor = self.predictor.get(code, name)
        df_dataset=self.predictor.makeLaggedDataset(code, start_date, end_date,
        self.config.get('input_column'), self.config.get('output_column'), lags_count)
```

start_date : 모델 선정에 사용할 데이터의 시작일, end_date : 모델선정에 사용할 데이터의 마감일,

input_column : 입력변수로, prices 테이블의 칼럼 명을 사용한다.

output_column : 머신러닝에 사용할 출력 변수 명칭을 지정하는 것으로 원하는 이름으로 지정한다.

makeLaggedDataset() 함수는 "lags_count=5"를 통해 5일 전 데이터를 이용해 오늘의 주가 방향을 예측할 수 있는 데이터 셋을 만든다.

splitDataset() 함수 : split_ratio 인자에 전달된 값에 따라 전체 데이터 셋을 학습용 데이터 셋과 테스트용 데이터 셋으로 나눈다.

```
        X_train, X_test, Y_train, Y_test =  self.predictor.splitDataset(df_dataset, 'price_date',
        [self.config.get('input_column')], self.config.get('output_column'), split_ratio = 0.8)
```

"split_ratio = 0.8" : 전체의 80%를 학습용 데이터 셋으로, 20%를 테스트용 데이터 셋으로 나눈다.

```
        param_grid = {"max_depth":[3, None], "min_samples_split": [1, 3, 10], "min_samples_leaf": [1,3,10],
        "bootstrap": [True, False], "criterion" : ["gini", "entropy"]}

        a_predictor.doGridSearch(X_train.values, Y_train.values, param_grid)

def doGridSearch(self, x_train, y_train, param_grid):
        grid_search = GridSearchCV(self.classifier, param_grid = param_grid)
        grid_search.fit(x_train, y_train)

        for params, mean_score, scores in grid_search.grid_scores_:
                print("%0.3f (+/-%0.03f) for %r" %(mean_score, socres.std()*2, params))
```

출력 : 0.500 (+/-0.070) for {'min_samples_split' : 3, 'bootstrap' : True, 'criterion': 'entropy', 'max_depth' : 3, 'min_samples_leaf' : 10}

0.500 : 점수, (+/- 0.070) : 표준 편차 값, 점수와 std를 보고 최적의 하이퍼 파라미터가 무엇인지를 찾을 수 있다.

"print("%0.3f (+/-%0.03f) for %r" %(mean_score, socres.std()*2, params))"

또한 하이퍼 파라미터의 값이 변화함에 따라 점수와 표준편차가 어떻게 변하는 지도 알아 볼 수 있다.

bootstrap = true : 중복허용 여부, 페이스팅 사용할 경우 False로 설정

min_samples_split : 노드를 분할하기 위한 최소한의 샘플 데이터 수, 과 적합을 제어하는 데 사용.

min_samples_leaf : 리프노드가 되기 위해 필요한 최소한의 샘플 데이터 수, 과 적합 제어용도.

- 자식 노드가 없는 노드를 잎 노드(leaf node 리프 노드[*])라고 한다. 잎 노드가 아닌 노드를 내부 노드(internal node)라고 한다.

max_depth : 트리의 최대 깊이.

(default = None : 완벽하게 클래스 값이 결정될 때까지 분할, 깊이가 깊어지면 과 적합 될 수 있으므로 적절히 제어 필요.)

max_leaf_nodes : 리프노드의 최대 개수

min_samples_leaf : 리프노드가 되기 위해 필요한 최소한의 샘플 데이터 수, 과 적합 제어용도