4_adaboost_exercise

May 31, 2019

```
In [ ]: import numpy as np

In [ ]: # resampling

        elements = ['one', 'two', 'three']
        weights = [0.2, 0.3, 0.5]

        from numpy.random import choice
        print(choice(elements))
        print(choice(elements, size=10, replace=True, p=weights))
        # element weight  .

In [ ]: X = np.load("./tatanic_X_train.npy")
        y = np.load("./tatanic_y_train.npy")

In [ ]: from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(X,y, \
                                            test_size=0.3, random_state=101)

In [ ]: X_train[:2]

In [ ]: y_train[:10]

In [ ]: from sklearn.ensemble import AdaBoostClassifier
        from sklearn.tree import DecisionTreeClassifier

In [ ]: eclf = AdaBoostClassifier\
        (base_estimator=DecisionTreeClassifier(max_depth=2), n_estimators=500,
                                          learning_rate=0.1)

In [ ]: from sklearn.model_selection import cross_val_score
        cross_val_score(eclf, X_train, y_train, cv=5).mean()

In [ ]: from sklearn.tree import DecisionTreeClassifier
        DecisionTreeClassifier()

In [ ]: AdaBoostClassifier()
In [ ]: params = {"base_estimator__criterion" : ["gini", "entropy"],
                   "base_estimator__max_features" : [7,8,],
                   "base_estimator__max_depth" : [1,2,3,4,5],
                   "n_estimators": [23,24, 25, 26, 27],
                   "learning_rate": [0.4, 0.45, 0.5, 0.55, 0.6]
                  }

        # "base_estimator__criterion" : ["gini", "entropy"], =>
        #  "base_estimator__max_features" : [7,8,],
        #  "base_estimator__max_depth" : [1,2], => 3  .
        #  ==> base estimator(ensemble model)  parameter (decision tree parameter)
        #  "n_estimators": [23,24, 25, 26, 27],
        #  "learning_rate": [0.4, 0.45, 0.5, 0.55, 0.6]

In [ ]: # gridsearch(=>GridSearchCV), cross_validation(=>cv), ensemble(=>eclf)

        # gridsearchcv operation

In [ ]: # gridsearch , cpu   => n_jobs

In [ ]: # best soore

In [ ]: # best parameters

In [ ]: grid.best_estimator_.feature_importances_ #     feature

In [ ]: grid.score(X_train, y_train)

In [ ]: grid.score(X_test, y_test)

In [ ]:

In [ ]: # confusion matrix

In [ ]: # classification report

In [ ]:
```