450 samples Xgboost-bacteria

1. Data preprocess

```
scaler_bac = StandardScaler().fit(X_bac)
X_bac_lasso = scaler_bac.transform(X_bac)

kfold = sk.model_selection.StratifiedKFold(n_splits=10, shuffle=True, random_state=0)
regr = LassoCV(cv=kfold)
regr.fit(X_bac_lasso, y_bac)
model_coef = pd.Series(regr.coef_, index = list(X_bac_all.columns[:]))
print("Variables Kept: ", str(sum(model_coef != 0)))    # Variables Kept:  19

## get the importance features
mask = top_coef != 0
X_bac_lasso = X_bac_lasso.loc[:, mask]
```

2. Xgboost classifier

2.1 Split data

```
X_bac_tain, X_bac_test, y_bac_train, y_bac_test = train_test_split(X_bac_lasso, y_bac,
                                                    shuffle=True, test_size=0.2, random_state=31, stratify=y_bac)
```

2.2 Tuning parameters

2.2.1 n_estimators

```
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=0)
param_test1 = {
 'n_estimators':range(1,100,1)
}

gsearch1 = GridSearchCV(
    estimator = XGBClassifier(
        booster = "gbtree",
        learning_rate = 0.1,
        n_estimators = 17,
        max_depth = 5,
        min_child_weight = 1,
        gamma = 0.1,
        subsample = 1.0,
        colsample_bytree = 0.8,
        colsample_bylevel = 1.0,
        reg_alpha = 0,       # L1 regularization parameter
        reg_lambda = 1.0,    # L2 regularization parameter
        objective = 'binary:logistic',
        nthread = 8,
        scale_pos_weight = 3.6,
        seed = 27,
        eval_metric = ["auc",],
        early_stopping_rounds=20,
    ),
    param_grid = param_test1,
    scoring = 'roc_auc',
    n_jobs = 8,
    cv = kfold,
    refit=True,
    return_train_score=True,
)

gsearch1.fit( X_bac_train, y_bac_train, eval_metric=["auc",] )
print(gsearch1.best_params_)    # {'n_estimators': 22}
```

2.2.2 max_depth & min_child_weight

```
param_test2 = {
 'max_depth':range(1,10,1),
 'min_child_weight':range(1,10)
}

gsearch2 = GridSearchCV(
    estimator = XGBClassifier(
        booster = "gbtree",
        learning_rate = 0.1,
        n_estimators = 22,
        max_depth = 5,
        min_child_weight = 1,
        gamma = 0.1,
        subsample = 1.0,
        colsample_bytree = 0.8,
        colsample_bylevel = 1.0,
        reg_alpha = 0,      # L1 regularization parameter
        reg_lambda = 1.0,   # L2 regularization parameter
        objective = 'binary:logistic',
        nthread = 8,
        scale_pos_weight = 3.6,
        seed = 27,
        eval_metric = ["auc",],
        early_stopping_rounds=20,
    ),
    param_grid = param_test2,
    scoring = 'roc_auc',
    n_jobs = 8,
    cv = kfold,
    refit=True,
    return_train_score=True,
)

gsearch2.fit(X_bac_tain, y_bac_tain, eval_metric="auc")
print(gsearch2.best_params_)    # {'max_depth': 3, 'min_child_weight': 6}
```

2.2.3 gamma

```
param_test3 = {
    'gamma':[i/10.0 for i in range(0,200, 1)]
}

gsearch3 = GridSearchCV(
    estimator = XGBClassifier(
        booster = "gbtree",
        learning_rate = 0.1,
        n_estimators = 22,  #gsearch1.best_params_["n_estimators"]
        max_depth = 3,      #gsearch2.best_params_["max_depth"]
        min_child_weight = 6,
#gsearch2.best_params_["min_child_weight"]
        gamma = 0.1,
        subsample = 1.0,
        colsample_bytree = 0.8,
        colsample_bylevel = 1.0,
        reg_alpha = 0,      # L1 regularization parameter
        reg_lambda = 1.0,   # L2 regularization parameter
        objective = 'binary:logistic',
        nthread = 8,
        scale_pos_weight = 3.6,
        seed = 27,
        eval_metric = ["auc",],
        early_stopping_rounds=20,
    ),
    param_grid = param_test3,
    scoring = 'roc_auc',
    n_jobs = 8,
    cv = kfold,
    refit=True,
    #verbose=2,
    return_train_score=True,
)
gsearch3.fit(X_bac_tain, y_bac_train, eval_metric="auc")
print(gsearch3.best_params_)    # {'gamma': 3.8}
```

2.2.4 Subsample & Colsample_bytree

```
## subsample and colsample_bytree
param_test4 = {
 'colsample_bylevel':[i/10.0 for i in range(2,11)],
 'colsample_bytree':[i/10.0 for i in range(2,11)]
}

gsearch4 = GridSearchCV(
    estimator = XGBClassifier(
        booster = "gbtree",
        learning_rate = 0.1,
        n_estimators = 22,      # gsearch1.best_params_["n_estimators"]
        max_depth = 3,          # gsearch2.best_params_["max_depth"]
        min_child_weight = 6,   #
gsearch2.best_params_["min_child_weight"]
        gamma = 3.8,            # gsearch3.best_params_["gamma"]
        subsample = 1.0,
        colsample_bytree = 0.8,
        colsample_bylevel = 1.0,
        reg_alpha = 0,      # L1 regularization parameter
        reg_lambda = 1.0,   # L2 regularization parameter
        objective = 'binary:logistic',
        nthread = 8,
        scale_pos_weight = 3.6,
        seed = 27,
        eval_metric = ["auc",],
        early_stopping_rounds=20,
    ),
    param_grid = param_test4,
    scoring = 'roc_auc',
    n_jobs = 8,
    cv = kfold,
    refit=True,
    #verbose=2,
    return_train_score=True,
)

gsearch4.fit(X_bac_tain, y_bac_tain, eval_metric="auc")
print(gsearch4.best_params_)  # {'colsample_bylevel': 1.0, 'colsample_bytree': 0.8}
```

2.2.5 reg_alpha & reg_lambda

```
param_test5 = {
 'reg_alpha': [0, 1e-5, 1e-2, 0.05, 0.075, 0.1, 0.15, 0.2, 0.25, 0.5, 1, 10],
 'reg_lambda': [0, 1e-2, 0.05, 0.075, 0.1, 0.25, 0.3,0.4, 0.5, 0.6, 0.7, 0.75, 0.8,0.9,1, 5, 10]
}
gsearch5 = GridSearchCV(
    estimator = XGBClassifier(
        booster = "gbtree",
        learning_rate = 0.1,
        n_estimators = 22,      # gsearch1.best_params_["n_estimators"]
        max_depth = 3,          # gsearch2.best_params_["max_depth"]
        min_child_weight = 6,   # gsearch2.best_params_["min_child_weight"]
        gamma = 3.8,            # gsearch3.best_params_["gamma"]
        subsample = 1.0,
        colsample_bytree = 0.8, #
gsearch4.best_params_["colsample_bytree"]
        colsample_bylevel = 1.0,# gsearch4.best_params_["colsample_bylevel"]
        reg_alpha = 0,          # L1 regularization parameter
        reg_lambda = 1.0,       # L2 regularization parameter
        objective = 'binary:logistic',
        nthread = 8,
        scale_pos_weight = 3.6,
        seed = 27,
        eval_metric = ["auc",],
        early_stopping_rounds=20,
    ),
    param_grid = param_test5,
    scoring = 'roc_auc',
    n_jobs = 8,
    cv = kfold,
    refit=True,
    #verbose=2,
    return_train_score=True,
)
gsearch5.fit(X_bac_tain, y_bac_tain, eval_metric="auc")
print(gsearch5.best_params_)    # {'reg_alpha': 0.075, 'reg_lambda': 0.9}
```

```
param_test6 = {
 "learning_rate": [1e-3, 1e-2, 0.05, 0.1, 0.5],
 }

gsearch6 = GridSearchCV(
    estimator = XGBClassifier(
        booster = "gbtree",
        learning_rate = 0.1,
        n_estimators = gsearch1.best_params_["n_estimators"],
        max_depth = gsearch2.best_params_["max_depth"],
        min_child_weight = gsearch2.best_params_["min_child_weight"],
        gamma = gsearch3.best_params_["gamma"],
        subsample = 1.0,
        colsample_bytree = gsearch4.best_params_["colsample_bytree"],
        colsample_bylevel = 1.0,        # gsearch4.best_params_["colsample_bylevel"],
        objective = 'binary:logistic',
        nthread = 8,
        scale_pos_weight = 3.6,
        reg_alpha = gsearch5.best_params_["reg_alpha"],
        reg_lambda = gsearch5.best_params_["reg_lambda"],
        seed=27,
        eval_metric = ["auc",],
        early_stopping_rounds=20,
    ),
    param_grid = param_test6,
    scoring = 'roc_auc',
    n_jobs = 8,
    cv = kfold,
    refit=True,
    #verbose=2,
    return_train_score=True,
)

gsearch6.fit(X_bac_tain, y_bac_tain, eval_metric="auc")
print(gsearch6.best_params_)  # {'learning_rate': 0.1}
```

```
clf_para = XGBClassifier(
    #silent = 0,  #
    nthread = 8,  # CPU threads
    learning_rate = gsearch6.best_params_["learning_rate"],
    booster = "gbtree",
    n_estimators = gsearch1.best_params_["n_estimators"],
    min_child_weight = gsearch2.best_params_["min_child_weight"],
    max_depth = gsearch2.best_params_["max_depth"],
    gamma = gsearch3.best_params_["gamma"],
    subsample = 1.0, #gsearch4.best_params_["subsample"],
    colsample_bytree = gsearch4.best_params_["colsample_bytree"],
    colsample_bylevel = 1.0, #gsearch4.best_params_["colsample_bylevel"],
    reg_lambda = gsearch5.best_params_["reg_lambda"],
    reg_alpha = gsearch5.best_params_["reg_alpha"], # L1 regularization parameter
    scale_pos_weight = 3.6,                         # L2 regularization parameter
    objective = 'binary:logistic',
    seed = 27,
    val_metric = ["auc", ],
)

clf_para.fit(X_bac_tain, y_bac_tain)
```