

# 450 samples Xgboost-bacteria

## 1. Data preprocess

```
1. scaler_bac = StandardScaler().fit(X_bac)
2. X_bac_lasso = scaler_bac.transform(X_bac)
3.
4. kfold = sk.model_selection.StratifiedKFold(n_splits=10, shuffle=True,
    random_state=0)
5. regr = LassoCV(cv=kfold)
6. regr.fit(X_bac_lasso, y_bac)
7. model_coef = pd.Series(regr.coef_, index = list(X_bac_all.columns[:]))
8. print("Variables Kept: ", str(sum(model_coef != 0)))      # Variables Kept: 19
9.
10. ## get the importance features
11. mask = top_coef != 0
12. X_bac_lasso = X_bac_lasso.loc[:, mask]
```

## 2. Xgboost classifier

### 2.1 Split data

```
1. X_bac_train, X_bac_test, y_bac_train, y_bac_test =
    train_test_split(X_bac_lasso, y_bac,
2.                                     shuffle=True, test_size=0.2,
    random_state=31, stratify=y_bac)
```

### 2.2 Tuning parameters

#### 2.2.1 n\_estimators

```
1. kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=0)
2. param_test1 = {
```

```

3.     'n_estimators':range(1,100,1)
4. }
5.
6. gsearch1 = GridSearchCV(
7.     estimator = XGBClassifier(
8.         booster = "gbtree",
9.         learning_rate = 0.1,
10.        n_estimators = 17,
11.        max_depth = 5,
12.        min_child_weight = 1,
13.        gamma = 0.1,
14.        subsample = 1.0,
15.        colsample_bytree = 0.8,
16.        colsample_bylevel = 1.0,
17.        reg_alpha = 0,          # L1 regularization parameter
18.        reg_lambda = 1.0,      # L2 regularization parameter
19.        objective = 'binary:logistic',
20.        nthread = 8,
21.        scale_pos_weight = 3.6,
22.        seed = 27,
23.        eval_metric = ["auc",],
24.        early_stopping_rounds=20,
25.    ),
26.    param_grid = param_test1,
27.    scoring = 'roc_auc',
28.    n_jobs = 8,
29.    cv = kfold,
30.    refit=True,
31.    return_train_score=True,
32. )
33.
34. gsearch1.fit( X_bac_train, y_bac_train, eval_metric=["auc",] )
35. print(gsearch1.best_params_)      # {'n_estimators': 22}

```

## 2.2.2 max\_depth & min\_child\_weight

```

1. param_test2 = {
2.     'max_depth':range(1,10,1),
3.     'min_child_weight':range(1,10)
4. }
5.
6. gsearch2 = GridSearchCV(
7.     estimator = XGBClassifier(

```

```

8.         booster = "gbtree",
9.         learning_rate = 0.1,
10.        n_estimators = 22,
11.        max_depth = 5,
12.        min_child_weight = 1,
13.        gamma = 0.1,
14.        subsample = 1.0,
15.        colsample_bytree = 0.8,
16.        colsample_bylevel = 1.0,
17.        reg_alpha = 0,      # L1 regularization parameter
18.        reg_lambda = 1.0,   # L2 regularization parameter
19.        objective = 'binary:logistic',
20.        nthread = 8,
21.        scale_pos_weight = 3.6,
22.        seed = 27,
23.        eval_metric = ["auc",],
24.        early_stopping_rounds=20,
25.    ),
26.    param_grid = param_test2,
27.    scoring = 'roc_auc',
28.    n_jobs = 8,
29.    cv = kfold,
30.    refit=True,
31.    return_train_score=True,
32. )
33.
34. gsearch2.fit(X_bac_tain, y_bac_tain, eval_metric="auc")
35. print(gsearch2.best_params_)      # {'max_depth': 3, 'min_child_weight':
    6}

```

## 2.2.3 gamma

```

1. param_test3 = {
2.     'gamma':[i/10.0 for i in range(0,200, 1)]
3. }
4.
5. gsearch3 = GridSearchCV(
6.     estimator = XGBClassifier(
7.         booster = "gbtree",
8.         learning_rate = 0.1,
9.         n_estimators = 22, #gsearch1.best_params_["n_estimators"]
10.        max_depth = 3,     #gsearch2.best_params_["max_depth"]
11.        min_child_weight = 6,

```

```

12.         #gsearch2.best_params_["min_child_weight"]
13.         gamma = 0.1,
14.         subsample = 1.0,
15.         colsample_bytree = 0.8,
16.         colsample_bylevel = 1.0,
17.         reg_alpha = 0,          # L1 regularization parameter
18.         reg_lambda = 1.0,      # L2 regularization parameter
19.         objective = 'binary:logistic',
20.         nthread = 8,
21.         scale_pos_weight = 3.6,
22.         seed = 27,
23.         eval_metric = ["auc",],
24.         early_stopping_rounds=20,
25.     ),
26.     param_grid = param_test3,
27.     scoring = 'roc_auc',
28.     n_jobs = 8,
29.     cv = kfold,
30.     refit=True,
31.     #verbose=2,
32.     return_train_score=True,
33. )
34. gsearch3.fit(X_bac_train, y_bac_train, eval_metric="auc")
35. print(gsearch3.best_params_)      # {'gamma': 3.8}

```

## 2.2.4 Subsample & Colsample\_bytree

```

1.     ## subsample and colsample_bytree
2.     param_test4 = {
3.         'colsample_bylevel':[i/10.0 for i in range(2,11)],
4.         'colsample_bytree':[i/10.0 for i in range(2,11)]
5.     }
6.
7.     gsearch4 = GridSearchCV(
8.         estimator = XGBClassifier(
9.             booster = "gbtree",
10.            learning_rate = 0.1,
11.            n_estimators = 22,          # gsearch1.best_params_["n_estimators"]
12.            max_depth = 3,             # gsearch2.best_params_["max_depth"]
13.            min_child_weight = 6,      #
14.            #gsearch2.best_params_["min_child_weight"]
15.            gamma = 3.8,               # gsearch3.best_params_["gamma"]
16.            subsample = 1.0,

```

```

16.         colsample_bytree = 0.8,
17.         colsample_bylevel = 1.0,
18.         reg_alpha = 0,          # L1 regularization parameter
19.         reg_lambda = 1.0,      # L2 regularization parameter
20.         objective = 'binary:logistic',
21.         nthread = 8,
22.         scale_pos_weight = 3.6,
23.         seed = 27,
24.         eval_metric = ["auc",],
25.         early_stopping_rounds=20,
26.     ),
27.     param_grid = param_test4,
28.     scoring = 'roc_auc',
29.     n_jobs = 8,
30.     cv = kfold,
31.     refit=True,
32.     #verbose=2,
33.     return_train_score=True,
34. )
35.
36. gsearch4.fit(X_bac_tain, y_bac_tain, eval_metric="auc")
37. print(gsearch4.best_params_)  # {'colsample_bylevel': 1.0,
    'colsample_bytree': 0.8}

```

## 2.2.5 reg\_alpha & reg\_lambda

```

1. param_test5 = {
2.     'reg_alpha': [0, 1e-5, 1e-2, 0.05, 0.075, 0.1, 0.15, 0.2, 0.25, 0.5, 1
    , 10],
3.     '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]
4. }
5. gsearch5 = GridSearchCV(
6.     estimator = XGBClassifier(
7.         booster = "gbtree",
8.         learning_rate = 0.1,
9.         n_estimators = 22,          # gsearch1.best_params_["n_estimators"]
10.        max_depth = 3,              # gsearch2.best_params_["max_depth"]
11.        min_child_weight = 6,      #
    gsearch2.best_params_["min_child_weight"]
12.        gamma = 3.8,                # gsearch3.best_params_["gamma"]
13.        subsample = 1.0,
14.        colsample_bytree = 0.8, #

```

```

gsearch4.best_params_["colsample_bytree"]
15.     colsample_bylevel = 1.0, # gsearch4.best_params_["colsample_bylevel"]
16.     reg_alpha = 0,           # L1 regularization parameter
17.     reg_lambda = 1.0,        # L2 regularization parameter
18.     objective = 'binary:logistic',
19.     nthread = 8,
20.     scale_pos_weight = 3.6,
21.     seed = 27,
22.     eval_metric = ["auc",],
23.     early_stopping_rounds=20,
24. ),
25.     param_grid = param_test5,
26.     scoring = 'roc_auc',
27.     n_jobs = 8,
28.     cv = kfold,
29.     refit=True,
30.     #verbose=2,
31.     return_train_score=True,
32. )
33. gsearch5.fit(X_bac_tain, y_bac_tain, eval_metric="auc")
34. print(gsearch5.best_params_)      # {'reg_alpha': 0.075, 'reg_lambda': 0.9}

```

```

1. param_test6 = {
2.     "learning_rate": [1e-3, 1e-2, 0.05, 0.1, 0.5],
3. }
4.
5. gsearch6 = GridSearchCV(
6.     estimator = XGBClassifier(
7.         booster = "gbtree",
8.         learning_rate = 0.1,
9.         n_estimators = gsearch1.best_params_["n_estimators"],
10.        max_depth = gsearch2.best_params_["max_depth"],
11.        min_child_weight = gsearch2.best_params_["min_child_weight"],
12.        gamma = gsearch3.best_params_["gamma"],
13.        subsample = 1.0,
14.        colsample_bytree = gsearch4.best_params_["colsample_bytree"],
15.        colsample_bylevel = 1.0, #
gsearch4.best_params_["colsample_bylevel"],
16.        objective = 'binary:logistic',
17.        nthread = 8,
18.        scale_pos_weight = 3.6,
19.        reg_alpha = gsearch5.best_params_["reg_alpha"],

```

```

20.         reg_lambda = gsearch5.best_params_["reg_lambda"],
21.         seed=27,
22.         eval_metric = ["auc",],
23.         early_stopping_rounds=20,
24.     ),
25.     param_grid = param_test6,
26.     scoring = 'roc_auc',
27.     n_jobs = 8,
28.     cv = kfold,
29.     refit=True,
30.     #verbose=2,
31.     return_train_score=True,
32. )
33.
34. gsearch6.fit(X_bac_tain, y_bac_tain, eval_metric="auc")
35. print(gsearch6.best_params_) # {'learning_rate': 0.1}

```

```

1.  clf_para = XGBClassifier(
2.      #silent = 0, #
3.      nthread = 8, # CPU threads
4.      learning_rate = gsearch6.best_params_["learning_rate"],
5.      booster = "gbtree",
6.      n_estimators = gsearch1.best_params_["n_estimators"],
7.      min_child_weight = gsearch2.best_params_["min_child_weight"],
8.      max_depth = gsearch2.best_params_["max_depth"],
9.      gamma = gsearch3.best_params_["gamma"],
10.     subsample = 1.0, #gsearch4.best_params_["subsample"],
11.     colsample_bytree = gsearch4.best_params_["colsample_bytree"],
12.     colsample_bylevel = 1.0,
13.     #gsearch4.best_params_["colsample_bylevel"],
14.     reg_lambda = gsearch5.best_params_["reg_lambda"],
15.     reg_alpha = gsearch5.best_params_["reg_alpha"], # L1
16.     #regularization parameter
17.     scale_pos_weight = 3.6, # L2 regularization
18.     #parameter
19.     objective = 'binary:logistic',
20.     seed = 27,
21.     val_metric = ["auc", ],
22. )
23.
24. clf_para.fit(X_bac_tain, y_bac_tain)

```

