

450 samples Xgboost_Virus

1. Data preprocess

```
1. kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=1)
2. scaler_virus = StandardScaler().fit(X_virus)
3. X_virus_lasso = scaler_virus.transform(X_virus)
4.
5. regr = LassoCV(cv=kfold)
6. regr.fit(X_virus_lasso, y_virus)
7.
8. model_coef = pd.Series(regr.coef_, index = list(X_virus.columns[:]))
9. print("Variables Kept: ", str(sum(model_coef != 0)))      # Variables Kept: 6
10.
11. mask = top_coef != 0
12. X_virus_lasso = X_virus_lasso[:, mask]
```

2.2. Xgboost classifier

2.2.1 Split data

```
1. X_virus_train, X_virus_test, y_virus_train, y_virus_test =
   train_test_split(X_virus_lasso, y_virus,
2.                                     shuffle=True, test_size=0.3,
   random_state=1, stratify=y_virus)
```

2.2 Tuning parameters

2.2.1 n_estimators

```
1. kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=0)
2. ## n_estimators
3. param_test1 = {
```

```

4.     'n_estimators':range(1,100,1)
5. }
6.
7. gsearch1 = GridSearchCV(
8.     estimator = XGBClassifier(
9.         booster = "gbtree",
10.        learning_rate = 0.1,
11.        n_estimators = 17,
12.        max_depth = 5,
13.        min_child_weight = 1,
14.        gamma = 0.1,
15.        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 = 6,
22.        scale_pos_weight = 1.0,
23.        seed = 27,
24.        eval_metric = ["auc",],
25.        early_stopping_rounds=20,
26.    ),
27.    param_grid = param_test1,
28.    scoring = 'roc_auc',
29.    n_jobs = 4,
30.    cv = kfold,
31.    refit=True,
32.    #verbose=2,
33.    return_train_score=True,
34. )
35.
36. gsearch1.fit( X_virus_train, y_virus_train, eval_metric=["auc",] )
37. print(gsearch1.best_params_)    #{'n_estimators': 13}

```

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 = gsearch1.best_params_["n_estimators"],
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 = 1,
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.    #verbose=2,
32.    return_train_score=True,
33. )
34.
35. gsearch2.fit(X_virus_train, y_virus_train, eval_metric="auc")
36. print(gsearch2.best_params_)      #{'max_depth': 2, 'min_child_weight':
37.                                     3}

```

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 = 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 = 0.1,
13.         subsample = 1.0,
14.         colsample_bytree = 0.8,
15.         colsample_bylevel = 1.0,
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 = 1.0,
21.         seed = 27,
22.         eval_metric = ["auc",],
23.         early_stopping_rounds=20,
24.     ),
25.     param_grid = param_test3,
26.     scoring = 'roc_auc',
27.     n_jobs = 8,
28.     cv = kfold,
29.     refit=True,
30.     #verbose=2,
31.     return_train_score=True,
32. )
33. gsearch3.fit(X_virus_train, y_virus_train, eval_metric="auc")
34. print(gsearch3.best_params_)      # {'gamma': 0.0}

```

2.2.4 Subsample & Colsample_bytree

```

1. param_test4 = {
2.     'colsample_bylevel':[i/10.0 for i in range(2,11)],
3.     'colsample_bytree':[i/10.0 for i in range(2,11)]
4. }
5.
6. gsearch4 = GridSearchCV(
7.     estimator = XGBClassifier(
8.         booster = "gbtree",
9.         learning_rate = 0.1,
10.         n_estimators = gsearch1.best_params_["n_estimators"],
11.         max_depth = gsearch2.best_params_["max_depth"],
12.         min_child_weight = gsearch2.best_params_["min_child_weight"],
13.         gamma = gsearch3.best_params_["gamma"],
14.         subsample = 1.0,
15.         colsample_bytree = 0.8,

```

```

16.         colsample_bylevel = 1.0,
17.         reg_alpha = 0,
18.         reg_lambda = 1.0,
19.         objective = 'binary:logistic',
20.         nthread = 8,
21.         scale_pos_weight = 1,
22.         seed = 27,
23.         eval_metric = ["auc",],
24.         early_stopping_rounds=20,
25.     ),
26.     param_grid = param_test4,
27.     scoring = 'roc_auc',
28.     n_jobs = 8,
29.     cv = kfold,
30.     refit=True,
31.     #verbose=2,
32.     return_train_score=True,
33. )
34.
35. gsearch4.fit(X_virus_train, y_virus_train, eval_metric="auc")
36. print(gsearch4.best_params_) # {'colsample_bylevel': 0.2,
    'colsample_bytree': 0.2}

```

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, 0.4, 0.
3.     5, 0.6, 0.7, 0.8, 1, 10],
4.     'reg_lambda': [0, 1e-2, 0.05, 0.075, 0.1, 0.25, 0.3, 0.4, 0.5, 0.6, 0.7
5.     , 0.75, 0.8, 0.85, 0.9, 1, 5]
6. }
7.
8. gsearch5 = GridSearchCV(
9.     estimator = XGBClassifier(
10.         booster = "gbtree",
11.         learning_rate = 0.1,
12.         n_estimators = gsearch1.best_params_["n_estimators"],
13.         max_depth = gsearch2.best_params_["max_depth"],
14.         min_child_weight = gsearch2.best_params_["min_child_weight"],
15.         gamma = 0, # gsearch3.best_params_["gamma"],
16.         subsample = 1.0, # gsearch4.best_params_["subsample"]
17.         colsample_bytree = gsearch4.best_params_["colsample_bytree"],
18.         colsample_bylevel = gsearch4.best_params_["colsample_bylevel"],

```

```

17.         reg_alpha = 0,
18.         reg_lambda = 1.0,
19.         objective = 'binary:logistic',
20.         nthread = 8,
21.         scale_pos_weight = 1,
22.         seed = 27,
23.         eval_metric = ["auc",],
24.         early_stopping_rounds=20,
25.     ),
26.     param_grid = param_test5,
27.     scoring = 'roc_auc',
28.     n_jobs = 8,
29.     cv = kfold,
30.     refit=True,
31.     #verbose=2,
32.     return_train_score=True,
33. )
34. gsearch5.fit(X_virus_train, y_virus_train, eval_metric="auc")
35. print(gsearch5.best_params_)      # {'reg_alpha': 0.01, 'reg_lambda': 0.
7}

```

2.2.6 learning_rate

```

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 = 0, #gsearch3.best_params_["gamma"],
13.         subsample = 1.0,
14.         colsample_bytree = gsearch4.best_params_["colsample_bytree"],
15.         colsample_bylevel = gsearch4.best_params_["colsample_bylevel"],
16.         objective = 'binary:logistic',
17.         nthread = 8,
18.         scale_pos_weight = 1,
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_virus_train, y_virus_train, eval_metric="auc")
35. print(gsearch6.best_params_)          # {'learning_rate': 0.1}

```

```

1.  clf_para = XGBClassifier(
2.      nthread = 8,
3.      learning_rate = gsearch6.best_params_["learning_rate"],
4.      booster = "gbtree",
5.      n_estimators = gsearch1.best_params_["n_estimators"],
6.      min_child_weight = gsearch2.best_params_["min_child_weight"],
7.      max_depth = gsearch2.best_params_["max_depth"],
8.      gamma = gsearch3.best_params_["gamma"],
9.      subsample = 1.0, #gsearch4.best_params_["subsample"],
10.     colsample_bytree = gsearch4.best_params_["colsample_bytree"],
11.     colsample_bylevel = gsearch4.best_params_["colsample_bylevel"],
12.     reg_lambda = gsearch5.best_params_["reg_lambda"],
13.     reg_alpha = gsearch5.best_params_["reg_alpha"],
14.     scale_pos_weight = 1.0,
15.     objective = 'binary:logistic',
16.     seed = 27,
17.     val_metric = ["auc", ],
18. )
19.
20. clf_para.fit(X_virus_train, y_virus_train)

```