調參數

Learning rate

```
forest = XGBClassifier()
# Try different numbers of n_estimators - this will take a
estimators = [10,1,0.1,0.01,0.001,0.0001,0.00001]
scores = []
for n in estimators:
    forest.set_params(learning_rate=n)
    forest.fit(X_train, y_train)
    test_y_predicted = forest.predict(X_test)
    scores.append(accuracy_score(y_test, test_y_predicted))
    #scores.append(model.score(x_test, y_test))
    print(accuracy_score(y_test, test_y_predicted))
plt.title("Effect of lr")
plt.xlabel("lr")
plt.ylabel("score")
plt.plot(estimators, scores)
```

```
0.22907099602829656
0.600128194996883
0.5516189740302693
0.5432921528857045
0.5426014218522713
0.542607275505097
0.542607275505097
```

Max depth

```
from xgboost import XGBClassifier
forest = XGBClassifier()
# Try different numbers of n_estimators - this will take a
estimators = np.arange(12, 53, 4)
scores = []
for n in estimators:
    forest.set_params(learning_rate=1,max_depth=n)
    forest.fit(X_train, y_train)
    test_y_predicted = forest.predict(X_test)
    scores.append(accuracy_score(y_test, test_y_predicted))
    #scores.append(model.score(x_test, y_test))
    print(accuracy_score(y_test, test_y_predicted))
plt.title("Effect of max_depth")
plt.xlabel("max_depth")
plt.ylabel("score")
plt.plot(estimators, scores)
```

```
0.7002461461013209
0.736325135292551
0.738584645283273
0.7370217199788098
```

N_estimators

```
warnings.filterwarnings('ignore')

from xgboost import XGBClassifier
forest = XGBClassifier()
# Try different numbers of n_estimators - this will take a minute
estimators = np.arange(20, 201, 20)
scores = []
for n in estimators:
    forest.set_params(learning_rate=1,max_depth=20,n_estimators=n)
    forest.fit(X_train, y_train)
    test_y_predicted = forest.predict(X_test)
    scores.append(accuracy_score(y_test, test_y_predicted))
    #scores.append(model.score(x_test, y_test))
    print(accuracy_score(y_test, test_y_predicted))
plt.title("Effect of n_estimators")
plt.xlabel("n_estimator")
plt.ylabel("score")
plt.plot(estimators, scores)
```

```
0.7278519728273436
0.7356812334817234
0.7373758659747649
0.7383768406079604
0.738584645283273
0.7389914741546594
0.7389856205018337
```

最後選擇: learning_rate=1,max_depth=20,n_estimators=120

```
Mean Square Error - test: 0.296
Mean Square Error - train: 0.042
Mean Absolute Error - test: 0.273
Mean Absolute Error - train: 0.040
Accuracy - test: 0.739
Accuracy - train: 0.961
```

```
Classification Report:
              precision    recall  f1-score   support

          -1       0.71      0.66      0.68     78281
           0       0.75      0.80      0.78    185120
           1       0.73      0.67      0.70     78266

   micro avg       0.74      0.74      0.74    341667
   macro avg       0.73      0.71      0.72    341667
weighted avg       0.74      0.74      0.74    341667

Accuracy: 0.7389914741546594
Precision: 0.7307161841183701
Recall: 0.7101680269345952
F-1: 0.7194464825948663
AUC: 0.7776641860748198
(peak=-1)
Confusion Matrix:
[[242031  26962]
 [ 21355  51319]]
True positives: 242031
False positives: 21355
True negatives: 51319
False negatives: 26962
AUC: 0.8029602079383822


(peak=0)
Confusion Matrix:
[[107943  36591]
 [ 48604 148529]]
True positives: 107943
False positives: 48604
True negatives: 148529
False negatives: 36591
AUC: 0.7501401490591803
```


```
(peak=1)
Confusion Matrix:
[[107943  36591]
 [ 48604 148529]]
True positives: 107943
False positives: 48604
True negatives: 148529
False negatives: 36591
AUC: 0.7501401490591803
```