姓名：

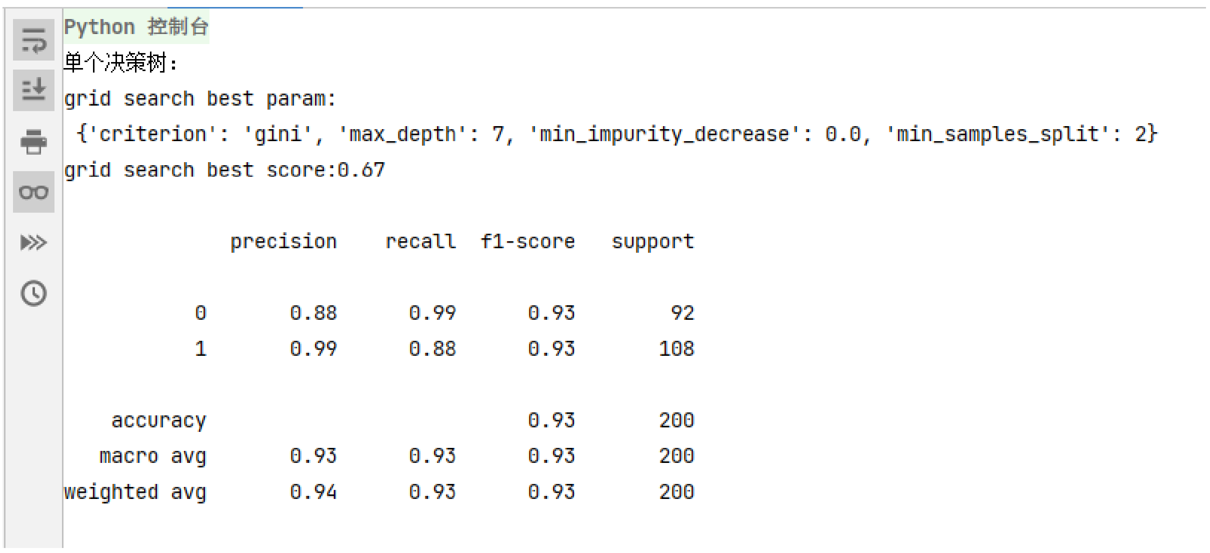
学号：

周数：

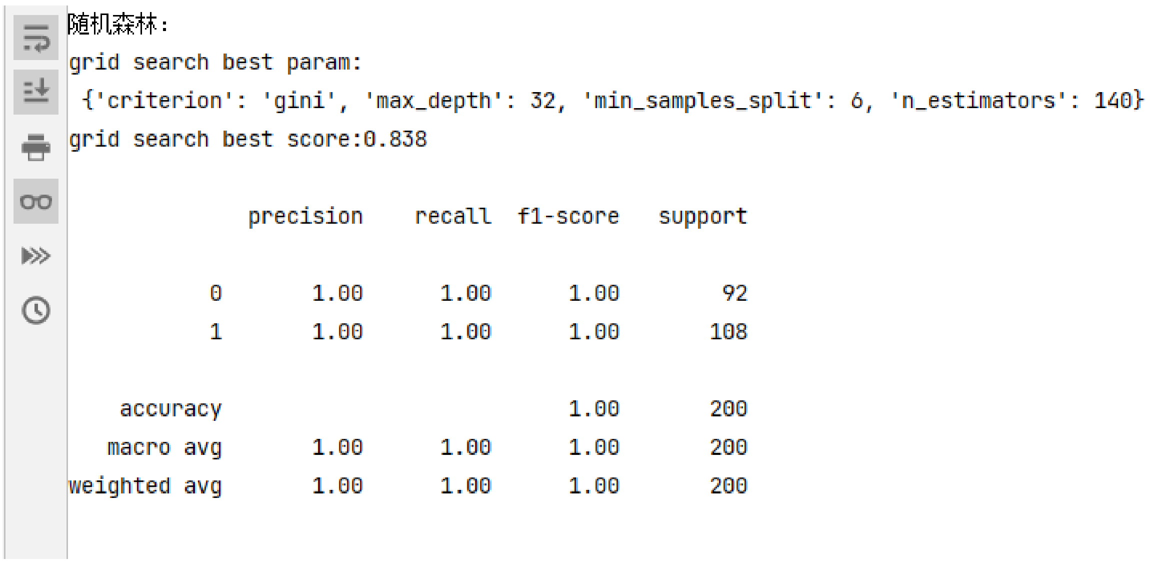
成绩：

程序：
```
from sklearn.datasets import make_classification
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier

# 生成数据集
X, y = make_classification(n_samples=1000, n_features=100,
                           n_informative=50, n_classes=2,
random_state=0)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)
# print('train datatest: {0}; test
dataset:{1}'.format(X_train.shape,X_test.shape))

# 单个决策树
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
train_score = clf.score(X_train, y_train)
test_score = clf.score(X_test, y_test)
# print('train score:{0}; test
score:{1}\n'.format(train_score,test_score))
predictions = clf.predict(X_test)
# print(classification_report(y_test,predictions))

# 网格搜索
entropy_thresholds = np.linspace(0, 1e-5, 5)
param_grid = {'criterion': ['gini'], 'min_impurity_decrease':
entropy_thresholds,
              'max_depth': range(2, 50, 5), 'min_samples_split':
range(2, 10, 2)}
clf = GridSearchCV(DecisionTreeClassifier(random_state=0), param_grid,
cv=5)
clf.fit(X, y)
best_parameters = clf.best_params_
print("单个决策树：")
print('grid search best param:\n {0}'.format(best_parameters))
print('grid search best score:{0:.2f}\n'.format(clf.best_score_))
prediction = clf.predict(X_test)
print(classification_report(y_test, prediction))
clf_best =
DecisionTreeClassifier(criterion=list(best_parameters.values())[0],
max_depth=list(best_parameters.values())[1],

min_impurity_decrease=list(best_parameters.values())[2],

min_samples_split=list(best_parameters.values())[3])
clf_best.fit(X_train, y_train)

# 随机森林模型网格搜索
param_grid = {'criterion': ['gini'], 'n_estimators':
    range(50, 151, 30),
                'max_depth': range(2, 50, 10),
                'min_samples_split': range(2, 10, 2)}
clf = GridSearchCV(RandomForestClassifier(), param_grid, cv=5)
clf.fit(X, y)
best_parameters = clf.best_params_
print("随机森林：")
print('grid search best param:\n {0}'.format(best_parameters))
print('grid search best score:{0:.3f}\n'.format(clf.best_score_))
prediction = clf.predict(X_test)
print(classification_report(y_test, prediction))

R_score = []
for i in range(1, 121):
    clf = RandomForestClassifier(n_estimators=i, max_features='sqrt',
                                 max_depth=42, min_samples_split=8,
                                 random_state=0)
    clf.fit(X, y)
    R_score.append('{0:.3f}'.format(clf.score(X_test, y_test)))

fig = plt.figure(figsize=(35, 20), dpi=200)
plt.xlabel('x')
plt.ylabel('y')
plt.xticks(fontsize=35)
plt.yticks(fontsize=35)
x = range(len(R_score))
plt.plot(x, R_score)
plt.show()
```

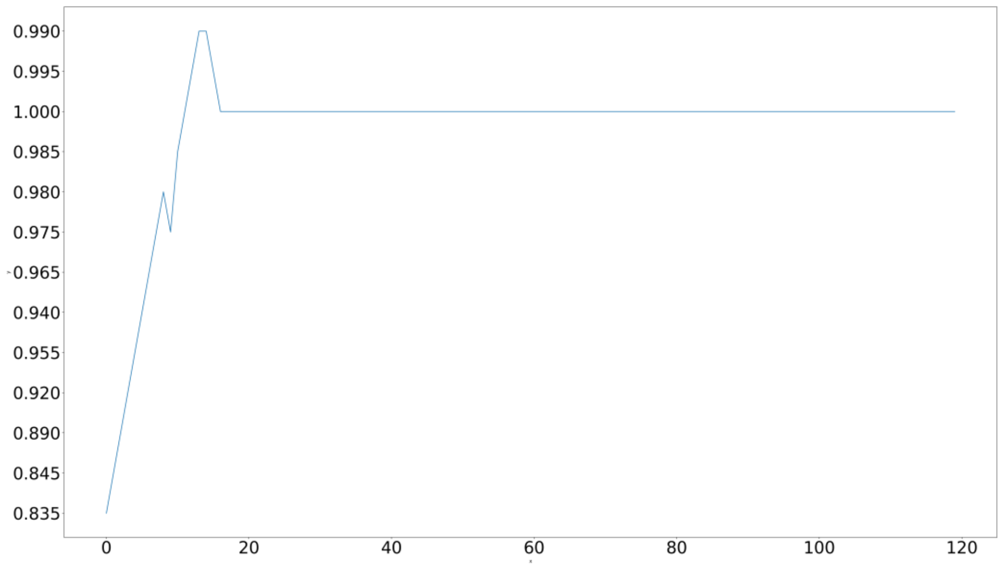
输出:



```
单个决策树:
grid search best param:
 {'criterion': 'gini', 'max_depth': 7, 'min_impurity_decrease': 0.0, 'min_samples_split': 2}
grid search best score:0.67

              precision    recall  f1-score   support

           0       0.88      0.99      0.93        92
           1       0.99      0.88      0.93       108

    accuracy                           0.93       200
   macro avg       0.93      0.93      0.93       200
weighted avg       0.94      0.93      0.93       200
```



```
随机森林:
grid search best param:
 {'criterion': 'gini', 'max_depth': 32, 'min_samples_split': 6, 'n_estimators': 140}
grid search best score:0.838

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        92
           1       1.00      1.00      1.00       108

    accuracy                           1.00       200
   macro avg       1.00      1.00      1.00       200
weighted avg       1.00      1.00      1.00       200
```


0.990
0.995
1.000
0.985
0.980
0.975
0.965
0.940
0.955
0.920
0.890
0.845
0.835
0
20
40
60
80
100
120