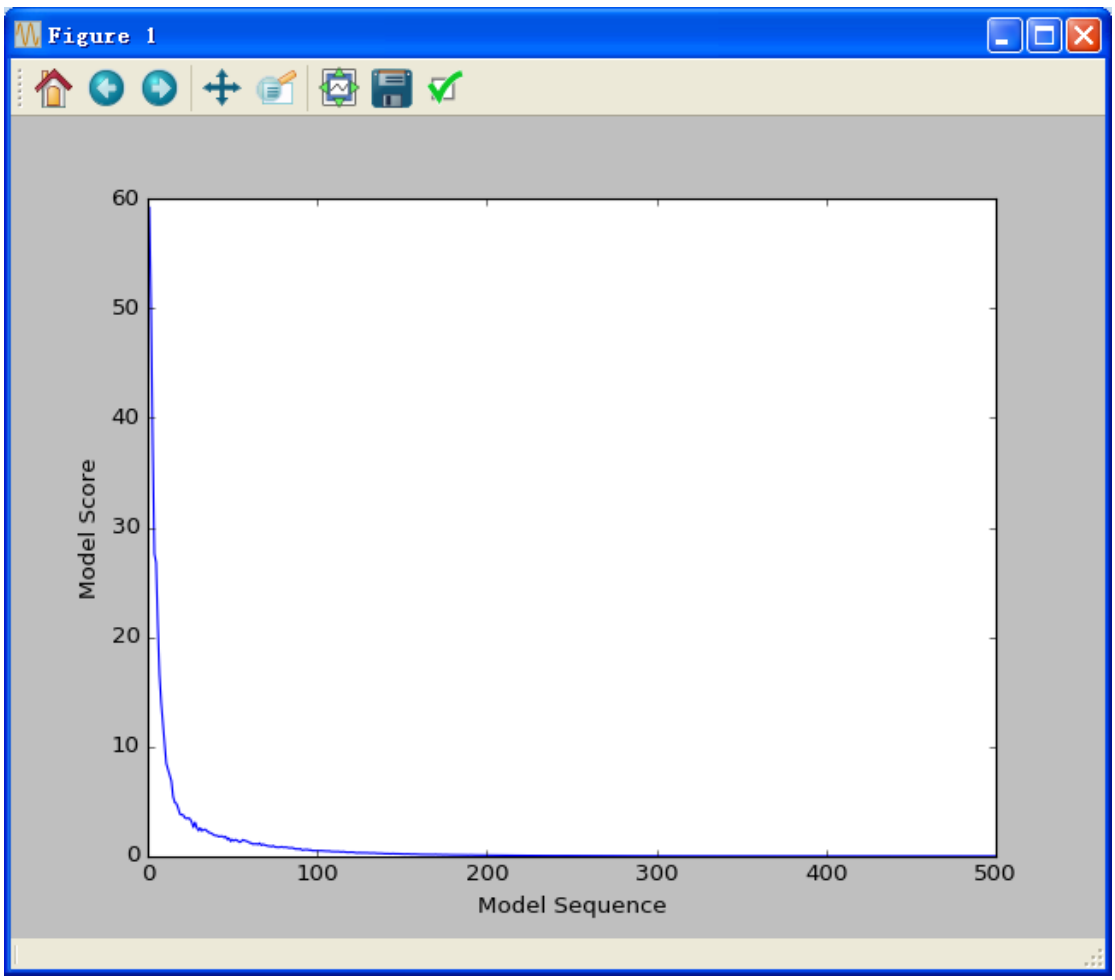
梯度提升案例

```
from sklearn.datasets import load_boston
from sklearn.cross_validation import train_test_split
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import PolynomialFeatures
import numpy as np
import matplotlib.pyplot as plt

# 数据集
boston = load_boston()
X = boston.data
y = boston.target
X_train,X_test,y_train,y_test                                    =
train_test_split(X,y,random_state=9)

# 为数据集添加多项式特征
pf = PolynomialFeatures(2,interaction_only=True)
pf.fit(X_train)
X_train_pf = pf.transform(X_train)
X_test_pf = pf.transform(X_test)

# 创建梯度提升回归器
gbr = GradientBoostingRegressor(n_estimators = 500,
                                learning_rate=0.15,
                                max_depth=3,
                                subsample=0.7,
                                random_state=77)
gbr.fit(X_train_pf,y_train)

# 在训练集上预测
y_pred = gbr.predict(X_train_pf)

print "MSE = %0.2f" % mean_squared_error(y_train,y_pred)

MSE = 0.00

# 每个模型在训练集上的损失(偏差)
for i,s in enumerate(gbr.train_score_):
    print "Estimator %d: score = %0.3f" % (i+1,s)

Estimator 1: score = 59.178
Estimator 2: score = 50.207
```

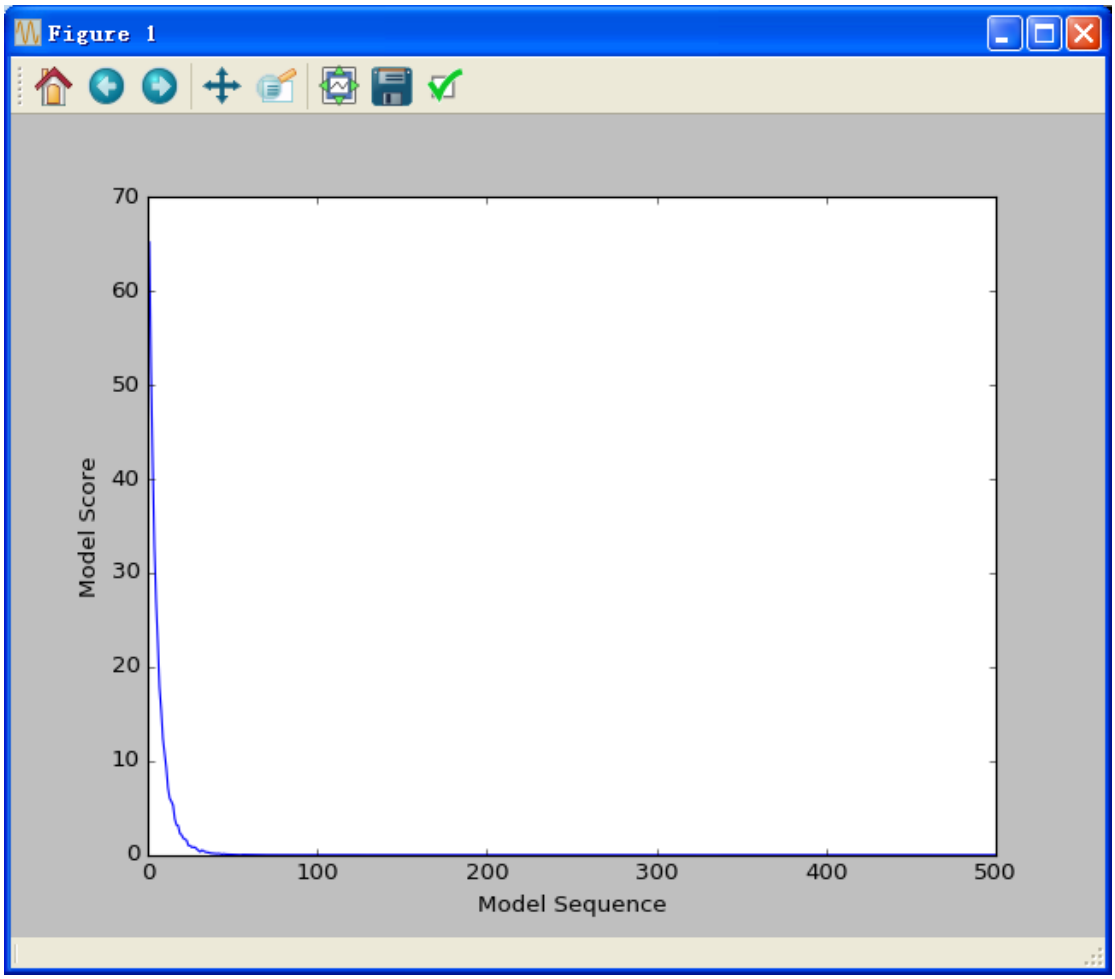
```
Estimator 3: score = 37.558
Estimator 4: score = 27.582
Estimator 5: score = 26.790
...
Estimator 498: score = 0.001
Estimator 499: score = 0.001
Estimator 500: score = 0.001

plt.figure(1)
n = gbr.estimators_.shape[0]
plt.plot(range(1,n+1,gbr.train_score_)
plt.xlabel("Model Sequence")
plt.ylabel("Model Score")
plt.show()
```




```
# 特征重要性
for i,fi in enumerate(gbr.feature_importances_):
    print "Feature %d: importance = %0.3f" % (i+1,fi)

Feature 1: importance = 0.000
```

```
Feature 2: importance = 0.003
Feature 3: importance = 0.000
Feature 4: importance = 0.002
...
Feature 92: importance = 0.026

# 在测试集上预测
y_pred = gbr.predict(X_test_pf)

print "MSE = %0.2f" % mean_squared_error(y_test,y_pred)

MSE = 10.16



# 梯度提升分类案例
from sklearn.datasets import load_iris

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.metrics import classification_report

from sklearn.metrics import accuracy_score

# 数据集
iris = load_iris()
X = iris.data
y = iris.target
X_train,X_test,y_train,y_test                                    =
train_test_split(X,y,random_state=9)

# 为数据集添加多项式特征
pf = PolynomialFeatures(2,interaction_only=True)
pf.fit(X_train)
X_train_pf = pf.transform(X_train)
X_test_pf = pf.transform(X_test)

# 创建梯度提升回归器
gbr = GradientBoostingClassifier(n_estimators = 500,
                                 learning_rate=0.15,
                                 max_depth=3,
                                 subsample=0.7,
                                 random_state=77)
gbr.fit(X_train_pf,y_train)
```

```
# 在训练集上预测
y_pred = gbr.predict(X_train_pf)

print "MSE = %0.2f" % mean_squared_error(y_train,y_pred)

MSE = 0.00

# 每个模型在训练集上的损失
for i,s in enumerate(gbr.train_score_):
    print "Estimator %d: score = %0.3f" % (i+1,s)

Estimator 1: score = 65.164
Estimator 2: score = 51.741
Estimator 3: score = 41.675
Estimator 4: score = 32.776
Estimator 5: score = 26.964
...
Estimator 498: score = 0.012
Estimator 499: score = 0.012
Estimator 500: score = 0.011

plt.figure(1)
n = gbr.estimators_.shape[0]
plt.plot(range(1,n+1,gbr.train_score_)
plt.xlabel("Model Sequence")
plt.ylabel("Model Score")
plt.show()
```




```
# 特征重要性
for i,fi in enumerate(gbr.feature_importances_):
    print "Feature %d: importance = %0.3f" % (i+1,fi)
```

```
Feature 1: importance = 0.000
Feature 2: importance = 0.003
Feature 3: importance = 0.005
Feature 4: importance = 0.014
Feature 5: importance = 0.009
Feature 6: importance = 0.006
Feature 7: importance = 0.010
Feature 8: importance = 0.014
Feature 9: importance = 0.013
Feature 10: importance = 0.011
Feature 11: importance = 0.046
```

```
# 在测试集上预测
y_pred = gbr.predict(X_test_pf)

print "MSE = %0.2f" % mean_squared_error(y_test,y_pred)
```

```
MSE = 0.03

print accuracy_score(y_test,y_pred)
0.9736842105263158

print classification_report(y_test,y_pred)
             precision    recall  f1-score   support

          0       1.00      1.00      1.00        15
          1       0.93      1.00      0.96        13
          2       1.00      0.90      0.95        10

avg / total       0.98      0.97      0.97        38
```