分类准确度

混淆矩阵

真实\预测	0	1
0	预测negtive正确TN	预测positive错误FP
1	预测negtive错误FN	预测positive正确TP

精准率与召回率

精准率：$precision = \frac{TP}{TP+FP}$

召回率：$recall = \frac{TP}{TP+FN}$

```
import numpy as np
from sklearn import datasets
digits = datasets.load_digits()
X = digits.data
y = digits.target.copy()
y[digits.target==9] = 1
y[digits.target!=9] = 0
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=666)
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)
log_reg.score(X_test, y_test)
y_log_predict = log_reg.predict(X_test)
def TN(y_true, y_predict):
    assert len(y_true) == len(y_predict)
    return np.sum((y_true == 0) & (y_predict == 0))
def FP(y_true, y_predict):
    assert len(y_true) == len(y_predict)
    return np.sum((y_true == 0) & (y_predict == 1))
def FN(y_true, y_predict):
    assert len(y_true) == len(y_predict)
    return np.sum((y_true == 1) & (y_predict == 0))
def TP(y_true, y_predict):
    assert len(y_true) == len(y_predict)
    return np.sum((y_true == 1) & (y_predict == 1))

def confusion_matrix(y_true, y_predict):
```

```
    return np.array([
        [TN(y_true, y_predict), FP(y_true, y_predict)],
        [FN(y_true, y_predict), TP(y_true, y_predict)]
    ])

def precision_score(y_true, y_predict):
    tp = TP(y_true, y_predict)
    fp = FP(y_true, y_predict)
    try:
        return tp / (tp + fp)
    except:
        return 0.0

def recall_score(y_true, y_predict):
    tp = TP(y_true, y_predict)
    fn = FN(y_true, y_predict)
    try:
        return tp / (tp + fn)
    except:
        return 0.0
```

sklearn中的实现

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
confusion_matrix(y_test, y_log_predict)
```

股票预测：注重精准率

病人诊断：注重召回率

F1 Score

$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$(调和平均值)

```
from sklearn.metrics import f1_score
print(f1_score(y_test, y_log_predict))
```

```
decision_scores = log_reg.decision_function(X_test)
y_log_predict2 = np.array(decision_scores >= 5, dtype='int')
```

调整决策边界

精确率、召回率曲线

```
precisions = []
recalls = []
thresholds = np.arange(np.min(decision_scores), np.max(decision_scores), 0.1)
for threshold in thresholds:
    y_predict = np.array(decision_scores >= threshold, dtype='int')
    precisions.append(precision_score(y_test, y_predict))
    recalls.append(recall_score(y_test, y_predict))

plt.plot(thresholds, precisions)
plt.plot(thresholds, recalls)
plt.show()
```

sklearn中的封装

```
from sklearn.metrics import precision_recall_curve
precisions, recalls, thresholds = precision_recall_curve(y_test,
decision_scores)
plt.plot(thresholds, precisions[:-1])
plt.plot(thresholds, recalls[:-1])
plt.show()
```

ROC曲线

$FPR = \frac{FP}{TN+FP}$

$TPR = \frac{TP}{TP+FN}$

```
from sklearn.metrics import roc_curve
fprs, tprs, thresholds = roc_curve(y_test, decision_scores)
plt.plot(fprs, tprs)
plt.show()
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test, decision_scores) //面积越大越好
```

对有偏数据不敏感

多分类问题

参数设定

```
from sklearn.metrics import precision_score
precision_score(y_test, y_predict, average="micro")
```

sklearn中天然支持多分类混淆矩阵

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_predict)
```

可视化错误矩阵

```
from sklearn.metrics import confusion_matrix
cfm = confusion_matrix(y_test, y_predict)
row_sums = np.sum(cfm, axis=1)
err_matrix = cfm / row_sums
np.fill_diagonal(err_matrix, 0)
plt.matshow(err_matrix, cmap=plt.cm.gray)
plt.show()
```