**Python sklearn机器学习各种评价指标——Sklearn.metrics简介及应用示例**

**Python sklearn机器学习各种评价指标——Sklearn.metrics简介及应用示例**

**Python** [**Sklearn**](https://so.csdn.net/so/search?q=Sklearn&spm=1001.2101.3001.7020) **.metrics 简介及应用示例**

利用Python进行各种 [机器学习](https://so.csdn.net/so/search?q=%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0&spm=1001.2101.3001.7020) 算法的实现时，经常会用到sklearn（scikit-learn）这个模块/库。

无论利用机器学习算法进行回归、分类或者聚类时， **评价指标** ，即检验机器学习模型效果的定量指标，都是一个不可避免且十分重要的问题。因此，结合 [scikit-learn主页](https://scikit-learn.org/stable/) 上的介绍，以及网上大神整理的一些资料，对常用的 [评价指标](https://so.csdn.net/so/search?q=%E8%AF%84%E4%BB%B7%E6%8C%87%E6%A0%87&spm=1001.2101.3001.7020) 及其实现、应用进行简单介绍。

**一、** [**scikit-learn**](https://so.csdn.net/so/search?q=scikit-learn&spm=1001.2101.3001.7020) **安装**

网上教程很多，此处不再赘述，具体可以参照：  
<https://www.cnblogs.com/zhangqunshi/p/6646987.html>  
此外，如果安装了Anoconda，可以直接从Anoconda Navigator——Environment里面搜索添加。  
pip install -U scikit-learn

**二、 scikit-learn.metrics导入与调用**

有两种方式导入：

**方式一：**

*from sklearn.metrics import 评价指标函数名称*

例如：

|  |
| --- |
| Python from sklearn.metrics import mean\_squared\_error from sklearn.metrics import r2\_score |

调用方式为：直接使用函数名调用  
计算均方误差mean squared error

|  |
| --- |
| Python mse = mean\_squared\_error(y\_test, y\_pre) |

计算回归的决定系数R2

|  |
| --- |
| Python R2 = r2\_score(y\_test,y\_pre) |

**方式二：**

*from sklearn import metrics*

调用方式为：metrics.评价指标函数名称（parameter）

例如：  
计算均方误差mean squared error

|  |
| --- |
| Python mse = metrics.mean\_squared\_error(y\_test, y\_pre) |

计算回归的决定系数R2

|  |
| --- |
| Python R2 = metrics.r2\_score(y\_test,y\_pre) |

**三、 scikit-learn.metrics里各种指标简介**

简单介绍参见：  
<https://www.cnblogs.com/mdevelopment/p/9456486.html>  
详细介绍参见：  
<https://www.cnblogs.com/harvey888/p/6964741.html>  
官网介绍：  
<https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics>

转自第一个链接的内容，简单介绍内容如下：

**回归指标**

1. explained\_variance\_score(y\_true, y\_pred, sample\_weight=None, multioutput=‘uniform\_average’)：回归方差(反应自变量与因变量之间的相关程度)
2. mean\_absolute\_error(y\_true,y\_pred,sample\_weight=None,  
   multioutput=‘uniform\_average’)：  
   平均绝对误差
3. mean\_squared\_error(y\_true, y\_pred, sample\_weight=None, multioutput=‘uniform\_average’)：均方差
4. median\_absolute\_error(y\_true, y\_pred) 中值绝对误差
5. r2\_score(y\_true, y\_pred,sample\_weight=None,multioutput=‘uniform\_average’) ：R平方值

**分类指标**

1. accuracy\_score(y\_true,y\_pre) : 精度
2. auc(x, y, reorder=False) : ROC曲线下的面积;较大的AUC代表了较好的performance。
3. average\_precision\_score(y\_true, y\_score, average=‘macro’, sample\_weight=None):根据预测得分计算平均精度(AP)
4. brier\_score\_loss(y\_true, y\_prob, sample\_weight=None, pos\_label=None):The smaller the Brier score, the better.
5. confusion\_matrix(y\_true, y\_pred, labels=None, sample\_weight=None):通过计算混淆矩阵来评估分类的准确性 返回混淆矩阵
6. f1\_score(y\_true, y\_pred, labels=None, pos\_label=1, average=‘binary’, sample\_weight=None): F1值  
   　　F1 = 2 \* (precision \* recall) / (precision + recall) precision(查准率)=TP/(TP+FP) recall(查全率)=TP/(TP+FN)
7. log\_loss(y\_true, y\_pred, eps=1e-15, normalize=True, sample\_weight=None, labels=None)：对数损耗，又称逻辑损耗或交叉熵损耗
8. precision\_score(y\_true, y\_pred, labels=None, pos\_label=1, average=‘binary’,) ：查准率或者精度； precision(查准率)=TP/(TP+FP)
9. recall\_score(y\_true, y\_pred, labels=None, pos\_label=1, average=‘binary’, sample\_weight=None)：查全率 ；recall(查全率)=TP/(TP+FN)
10. roc\_auc\_score(y\_true, y\_score, average=‘macro’, sample\_weight=None)：计算ROC曲线下的面积就是AUC的值，the larger the better
11. roc\_curve(y\_true, y\_score, pos\_label=None, sample\_weight=None, drop\_intermediate=True)；计算ROC曲线的横纵坐标值，TPR，FPR  
    　　TPR = TP/(TP+FN) = recall(真正例率，敏感度) FPR = FP/(FP+TN)(假正例率，1-特异性)

**四、 一个应用实例**

结合官网的案例，利用自己的数据，实现的一个应用实例：

|  |
| --- |
| Python import pandas as pd import numpy as np import matplotlib.pyplot as plt from sklearn import ensemble from sklearn import metrics  ############################################################################## # Load data data = pd.read\_csv('Data for train\_0.003D.csv') y = data.iloc[:,0] X = data.iloc[:,1:] offset = int(X.shape[0] \* 0.9) X\_train, y\_train = X[:offset], y[:offset] X\_test, y\_test = X[offset:], y[offset:]  ############################################################################## # Fit regression model params = {'n\_estimators': 500, 'max\_depth': 4, 'min\_samples\_split': 2,  'learning\_rate': 0.01, 'loss': 'ls'} clf = ensemble.GradientBoostingRegressor(\*\*params)  clf.fit(X\_train, y\_train) y\_pre = clf.predict(X\_test)  # Calculate metrics mse = metrics.mean\_squared\_error(y\_test, y\_pre) print("MSE: %.4f" % mse)  mae = metrics.mean\_absolute\_error(y\_test, y\_pre) print("MAE: %.4f" % mae)  R2 = metrics.r2\_score(y\_test,y\_pre) print("R2: %.4f" % R2)  ############################################################################## # Plot training deviance  # compute test set deviance test\_score = np.zeros((params['n\_estimators'],), dtype=np.float64)  for i, y\_pred in enumerate(clf.staged\_predict(X\_test)):  test\_score[i] = clf.loss\_(y\_test, y\_pred)  plt.figure(figsize=(12, 6)) plt.subplot(1, 2, 1) plt.title('Deviance') plt.plot(np.arange(params['n\_estimators']) + 1, clf.train\_score\_, 'b-',  label='Training Set Deviance') plt.plot(np.arange(params['n\_estimators']) + 1, test\_score, 'r-',  label='Test Set Deviance') plt.legend(loc='upper right') plt.xlabel('Boosting Iterations') plt.ylabel('Deviance')  ############################################################################## # Plot feature importance feature\_importance = clf.feature\_importances\_ # make importances relative to max importance feature\_importance = 100.0 \* (feature\_importance / feature\_importance.max()) sorted\_idx = np.argsort(feature\_importance) pos = np.arange(sorted\_idx.shape[0]) + .5 plt.subplot(1, 2, 2) plt.barh(pos, feature\_importance[sorted\_idx], align='center') plt.yticks(pos, X.columns[sorted\_idx])  plt.xlabel('Relative Importance') plt.title('Variable Importance') plt.show() |

最终的运行结果为：

