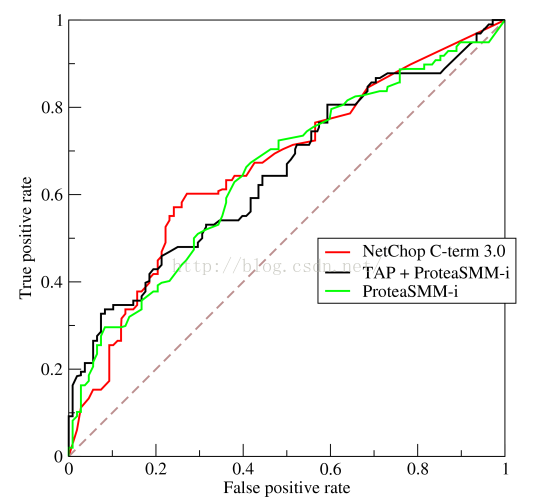
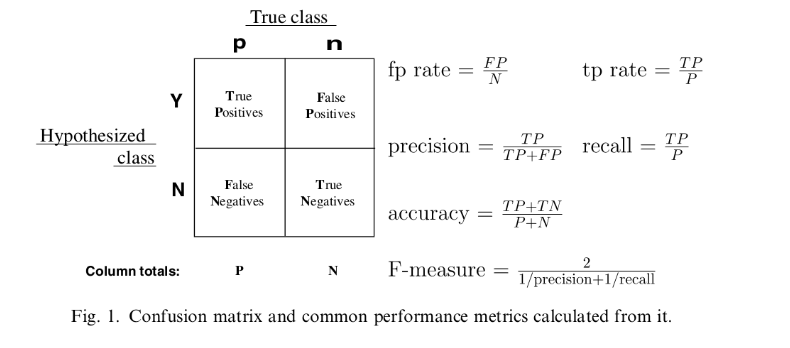
# ROC

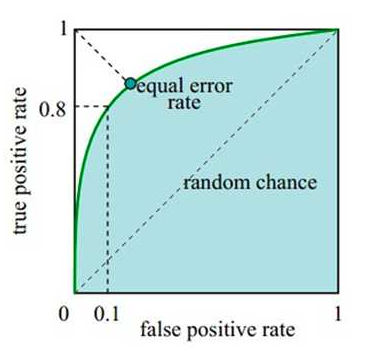
<https://kennis-research.shinyapps.io/ROC-Curves/> 曲线演示

ROC（Receiver Operating Characteristic）曲线和AUC常被用来评价一个二值分类器（binary classifier）的优劣ROC曲线越接近左上角，该分类器的性能越好。



正如我们在这个ROC曲线的示例图中看到的那样，ROC曲线的横坐标为false positive rate（FPR），纵坐标为true positive rate（TPR）。下图中详细说明了FPR和TPR是如何定义的。





# **AUC值**

## **AUC值的定义**

　　AUC值为ROC曲线所覆盖的区域面积，显然，AUC越大，分类器分类效果越好。

　　AUC = 1，是完美分类器，采用这个预测模型时，不管设定什么阈值都能得出完美预测。绝大多数预测的场合，不存在完美分类器。

　　0.5 < AUC < 1，优于随机猜测。这个分类器（模型）妥善设定阈值的话，能有预测价值。

　　AUC = 0.5，跟随机猜测一样（例：丢硬币），模型没有预测价值。

　　AUC < 0.5，比随机猜测还差；但只要总是反预测而行，就优于随机猜测。

## **AUC值的物理意义**

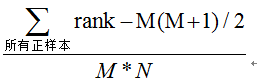
　　假设分类器的输出是样本属于正类的socre（置信度），则AUC的物理意义为，任取一对（正、负）样本，正样本的score大于负样本的score的概率。

## **AUC值的计算**

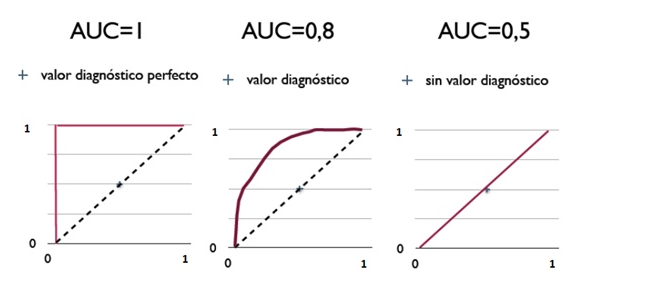
　　（1）第一种方法：AUC为ROC曲线下的面积，那我们直接计算面积可得。面积为一个个小的梯形面积之和，计算的精度与阈值的精度有关。

　　（2）第二种方法：根据AUC的物理意义，我们计算正样本score大于负样本的score的概率。取N\*M（N为正样本数，M为负样本数）个二元组，比较score，最后得到AUC。时间复杂度为O(N\*M)。

　　（3）第三种方法：与第二种方法相似，直接计算正样本score大于负样本的score的概率。我们首先把所有样本按照score排序，依次用rank表示他们，如最大score的样本，rank=n(n=N+M)，其次为n-1。那么对于正样本中rank最大的样本（rank\_max），有M-1个其他正样本比他score小，那么就有(rank\_max-1)-(M-1)个负样本比他score小。其次为(rank\_second-1)-(M-2)。最后我们得到正样本大于负样本的概率为：



时间复杂度为O(N+M)。



# **ROC与AUC的定义与使用详解**

分类模型评估：

| 指标 | 描述 | Scikit-learn函数 |
| --- | --- | --- |
| Precision | 精准度 | from sklearn.metrics import precision\_score |
| Recall | 召回率 | from sklearn.metrics import recall\_score |
| F1 | F1值 | from sklearn.metrics import f1\_score |
| Confusion Matrix | 混淆矩阵 | from sklearn.metrics import confusion\_matrix |
| ROC | ROC曲线 | from sklearn.metrics import roc |
| AUC | ROC曲线下的面积 | from sklearn.metrics import auc |

回归模型评估：

| 指标 | 描述 | Scikit-learn函数 |
| --- | --- | --- |
| Mean Square Error (MSE, RMSE) | 平均方差 | from sklearn.metrics import mean\_squared\_error |
| Absolute Error (MAE, RAE) | 绝对误差 | from sklearn.metrics import mean\_absolute\_error, median\_absolute\_error |
| R-Squared | R平方值 | from sklearn.metrics import r2\_score |

# ROC和AUC定义

ROC全称是“受试者工作特征”（Receiver Operating Characteristic）。ROC曲线的面积就是AUC（Area Under the Curve）。AUC用于衡量“二分类问题”**[机器学习](http://lib.csdn.net/base/machinelearning" \o "机器学习知识库" \t "http://blog.csdn.net/shenxiaoming77/article/details/_blank)[算法](http://lib.csdn.net/base/datastructure" \o "算法与数据结构知识库" \t "http://blog.csdn.net/shenxiaoming77/article/details/_blank)**性能（泛化能力）。

# 计算ROC需要知道的关键概念

首先，解释几个二分类问题中常用的概念：True Positive, False Positive, True Negative, False Negative。它们是根据真实类别与预测类别的组合来区分的。

假设有一批test样本，这些样本只有两种类别：正例和反例。机器学习算法预测类别如下图（左半部分预测类别为正例，右半部分预测类别为反例），而样本中真实的正例类别在上半部分，下半部分为真实的反例。

* 预测值为正例，记为P（Positive）
* 预测值为反例，记为N（Negative）
* 预测值与真实值相同，记为T（True）
* 预测值与真实值相反，记为F（False）



* TP：预测类别是P（正例），真实类别也是P
* FP：预测类别是P，真实类别是N（反例）
* TN：预测类别是N，真实类别也是N
* FN：预测类别是N，真实类别是P

样本中的真实正例类别总数即TP+FN。TPR即True Positive Rate，TPR = TP/(TP+FN)。   
 同理，样本中的真实反例类别总数为FP+TN。FPR即False Positive Rate，FPR=FP/(TN+FP)。

还有一个概念叫”截断点”。机器学习算法对test样本进行预测后，可以输出各test样本对某个类别的相似度概率。比如t1是P类别的概率为0.3，一般我们认为概率低于0.5，t1就属于类别N。这里的0.5，就是”截断点”。   
 总结一下，对于计算ROC，最重要的三个概念就是TPR, FPR, 截断点。

**截断点取不同的值，TPR和FPR的计算结果也不同。将截断点不同取值下对应的TPR和FPR结果画于二维坐标系中得到的曲线，就是ROC曲线。横轴用FPR表示。**

# sklearn计算ROC

代码：

y = np.array([1, 1, 2, 2])

scores = np.array([0.1, 0.4, 0.35, 0.8])

fpr, tpr, thresholds = metrics.roc\_curve(y, scores, pos\_label=2)

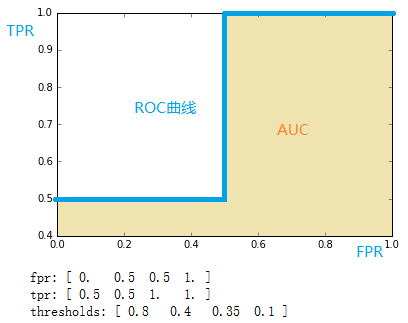
通过计算，得到的结果（TPR, FPR, 截断点）为

fpr = array([ 0. , 0.5, 0.5, 1. ])

tpr = array([ 0.5, 0.5, 1. , 1. ])

thresholds = array([ 0.8 , 0.4 , 0.35, 0.1 ])#截断点

将结果中的FPR与TPR画到二维坐标中，得到的ROC曲线如下（蓝色线条表示），ROC曲线的面积用AUC表示（淡黄色阴影部分）。



# 详细计算过程

上例给出的数据如下

y = np.array([1, 1, 2, 2])

scores = np.array([0.1, 0.4, 0.35, 0.8])

用这个数据，计算TPR，FPR的过程是怎么样的呢？

## 1. 分析数据

y是一个一维数组（样本的真实分类）。数组值表示类别（一共有两类，1和2）。我们假设y中的1表示反例，2表示正例。即将y重写为：

y\_true = [0, 0, 1, 1]

**score即各个样本属于正例的概率。**

## 2. 针对score，将数据排序

| **样本** | **预测属于P的概率(score)** | **真实类别** |
| --- | --- | --- |
| y[0] | 0.1 | N |
| y[2] | 0.35 | P |
| y[1] | 0.4 | N |
| y[3] | 0.8 | P |

## 3. 将截断点依次取为score值

将截断点依次取值为0.1,0.35,0.4,0.8时，计算TPR和FPR的结果。

### 3.1 截断点为0.1

说明只要score>=0.1，它的预测类别就是正例。   
此时，因为4个样本的score都大于等于0.1，所以，所有样本的预测类别都为P。

scores = [0.1, 0.4, 0.35, 0.8]

y\_true = [0, 0, 1, 1]

y\_pred = [1, 1, 1, 1]



TPR = TP/(TP+FN) = 1   
FPR = FP/(TN+FP) = 1

### 3.2 截断点为0.35

说明只要score>=0.35，它的预测类别就是P。   
此时，因为4个样本的score有3个大于等于0.35。所以，所有样本的预测类有3个为P（2个预测正确，1一个预测错误）；1个样本被预测为N（预测正确）。

scores = [0.1, 0.4, 0.35, 0.8]

y\_true = [0, 0, 1, 1]

y\_pred = [0, 1, 1, 1]



TPR = TP/(TP+FN) = 1   
FPR = FP/(TN+FP) = 0.5

### 3.3 截断点为0.4

说明只要score>=0.4，它的预测类别就是P。   
此时，因为4个样本的score有2个大于等于0.4。所以，所有样本的预测类有2个为P（1个预测正确，1一个预测错误）；2个样本被预测为N（1个预测正确，1一个预测错误）。

scores = [0.1, 0.4, 0.35, 0.8]

y\_true = [0, 0, 1, 1]

y\_pred = [0, 1, 0, 1]



TPR = TP/(TP+FN) = 0.5   
FPR = FP/(TN+FP) = 0.5

### 3.4 截断点为0.8

说明只要score>=0.8，它的预测类别就是P。所以，所有样本的预测类有1个为P（1个预测正确）；3个样本被预测为N（2个预测正确，1一个预测错误）。

scores = [0.1, 0.4, 0.35, 0.8]

y\_true = [0, 0, 1, 1]

y\_pred = [0, 0, 0, 1]



TPR = TP/(TP+FN) = 0.5   
FPR = FP/(TN+FP) = 0

总结

用下面描述表示TPR和FPR的计算过程，更容易记住

* TPR：真实的正例中，被预测正确的比例
* FPR：真实的反例中，被预测正确的比例

最理想的分类器，就是对样本分类完全正确，即FP=0，FN=0。所以理想分类器TPR=1，FPR=0。

# 代码实战：

## 1.Receiver Operating Characteristic (ROC) with cross validation

**import** matplotlib.pyplot **as** plt  
**from** sklearn **import** svm, datasets  
**from** sklearn.metrics **import** roc\_curve, auc,roc\_auc\_score  
**from** sklearn.model\_selection **import** StratifiedKFold  
  
*# #############################################################################  
# Data IO and generation  
  
# Import some data to play with*iris = datasets.load\_iris()  
X = iris.data  
y = iris.target  
X, y = X[y != 2], y[y != 2]  
  
*# Classification and ROC analysis  
  
# Run classifier with cross-validation and plot ROC curves*cv = StratifiedKFold(n\_splits=6)  
classifier = svm.SVC(kernel=**'linear'**, probability=**True**,  
 random\_state=11)  
tprs = []  
aucs = []  
i = 0  
**for** train, test **in** cv.split(X, y):  
 print(**"train:"**,train)  
 print(**"test:"**,test)  
 probas\_ = classifier.fit(X[train], y[train]).predict\_proba(X[test])  
 print(**"====="**,probas\_) *#0 1  
 # Compute ROC curve and area the curve* fpr, tpr, thresholds = roc\_curve(y[test], probas\_[:, 1])  
 roc\_auc\_score\_v=roc\_auc\_score(y[test], probas\_[:, 1] )  
 roc\_auc = auc(fpr, tpr)  
 aucs.append(roc\_auc)  
 plt.plot(fpr, tpr, lw=1, alpha=0.3,  
 label=**'ROC fold %d (AUC = %0.2f)'** % (i, roc\_auc))  
 i += 1  
 print(**"\*"**\*100)  
 print(roc\_auc\_score\_v)  
 *# tprs.append(fpr)  
 # aucs.append(tpr)*plt.plot([0, 1], [0, 1], linestyle=**'--'**, lw=2, color=**'r'**,  
 label=**'Chance'**, alpha=.8)  
plt.show()

## 2.精进版本

**import** numpy **as** np  
**from** scipy **import** interp  
**import** matplotlib.pyplot **as** plt  
**from** itertools **import** cycle  
  
**from** sklearn **import** svm, datasets  
**from** sklearn.metrics **import** roc\_curve, auc  
**from** sklearn.model\_selection **import** StratifiedKFold  
  
*# #########################################################################  
# Data IO and generation  
  
# Import some data to play with*iris = datasets.load\_iris()  
X = iris.data  
y = iris.target  
print(y) *# 0 1 2 为了演示两个列类别的auc，去掉第二类 #0 1*X, y = X[y != 2], y[y != 2]  
n\_samples, n\_features = X.shape  
  
*# Add noisy features*random\_state = np.random.RandomState(0)  
X = np.c\_[X, random\_state.randn(n\_samples, 200 \* n\_features)]  
  
*# #########################################################################  
# Classification and ROC analysis  
  
# Run classifier with cross-validation and plot ROC curves*cv = StratifiedKFold(n\_splits=6)  
classifier = svm.SVC(kernel=**'linear'**, probability=**True**,  
 random\_state=random\_state)  
  
tprs = []  
aucs = []  
mean\_fpr = np.linspace(0, 1, 100)  
  
i = 0  
**for** train, test **in** cv.split(X, y):  
 probas\_ = classifier.fit(X[train], y[train]).predict\_proba(X[test])  
 print(**"====="**,probas\_)  
 *# Compute ROC curve and area the curve* fpr, tpr, thresholds = roc\_curve(y[test], probas\_[:, 1])  
 tprs.append(interp(mean\_fpr, fpr, tpr))  
 tprs[-1][0] = 0.0  
 roc\_auc = auc(fpr, tpr)  
 aucs.append(roc\_auc)  
 plt.plot(fpr, tpr, lw=1, alpha=0.3,  
 label=**'ROC fold %d (AUC = %0.2f)'** % (i, roc\_auc))  
  
 i += 1  
plt.plot([0, 1], [0, 1], linestyle=**'--'**, lw=2, color=**'r'**,  
 label=**'Chance'**, alpha=.8)  
  
mean\_tpr = np.mean(tprs, axis=0)  
mean\_tpr[-1] = 1.0  
mean\_auc = auc(mean\_fpr, mean\_tpr)  
std\_auc = np.std(aucs)  
plt.plot(mean\_fpr, mean\_tpr, color=**'b'**,  
 label=**r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)'** % (mean\_auc, std\_auc),  
 lw=2, alpha=.8)  
  
std\_tpr = np.std(tprs, axis=0)  
tprs\_upper = np.minimum(mean\_tpr + std\_tpr, 1)  
tprs\_lower = np.maximum(mean\_tpr - std\_tpr, 0)  
plt.fill\_between(mean\_fpr, tprs\_lower, tprs\_upper, color=**'grey'**, alpha=.2,  
 label=**r'$\pm$ 1 std. dev.'**)  
  
plt.xlim([-0.05, 1.05])  
plt.ylim([-0.05, 1.05])  
plt.xlabel(**'False Positive Rate'**)  
plt.ylabel(**'True Positive Rate'**)  
plt.title(**'Receiver operating characteristic example'**)  
plt.legend(loc=**"lower right"**)  
plt.show()

## 3.Receiver Operating Characteristic (ROC)

*# -\*- coding: utf-8 -\*-  
# @Time : 2018/11/19 17:47  
# @Author : Z  
# @Email : S  
# @File : 8.0auctest.py*print(\_\_doc\_\_)  
*# https://scikit-learn.org/stable/auto\_examples/model\_selection/plot\_roc.html#sphx-glr-auto-examples-model-selection-plot-roc-py***import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
**from** itertools **import** cycle  
  
**from** sklearn **import** svm, datasets  
**from** sklearn.metrics **import** roc\_curve, auc  
**from** sklearn.model\_selection **import** train\_test\_split  
**from** sklearn.preprocessing **import** label\_binarize  
**from** sklearn.multiclass **import** OneVsRestClassifier  
**from** scipy **import** interp  
  
*# Import some data to play with*iris = datasets.load\_iris()  
X = iris.data  
y = iris.target  
  
*# Binarize the output*y = label\_binarize(y, classes=[0, 1, 2])  
*# """Binarize labels in a one-vs-all fashion*n\_classes = y.shape[1]  
  
*# Add noisy features to make the problem harder*random\_state = np.random.RandomState(0)  
n\_samples, n\_features = X.shape  
X = np.c\_[X, random\_state.randn(n\_samples, 200 \* n\_features)]  
  
*# shuffle and split training and test sets*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=.5,  
 random\_state=0)  
  
*# Learn to predict each class against the other*classifier = OneVsRestClassifier(svm.SVC(kernel=**'linear'**, probability=**True**,  
 random\_state=random\_state))  
y\_score = classifier.fit(X\_train, y\_train).decision\_function(X\_test)  
*# """Returns the distance of each sample from the decision boundary for  
# each class. This can only be used with estimators which implement the  
# decision\_function method.  
# Compute ROC curve and ROC area for each class*fpr = dict()  
tpr = dict()  
roc\_auc = dict()  
**for** i **in** range(n\_classes):  
 fpr[i], tpr[i], \_ = roc\_curve(y\_test[:, i], y\_score[:, i])  
 roc\_auc[i] = auc(fpr[i], tpr[i])  
  
*# Compute micro-average ROC curve and ROC area  
# ravel--- """ Return a flattened (numpy) array.  
# 将全部数据摊平处理*fpr[**"micro"**], tpr[**"micro"**], \_ = roc\_curve(y\_test.ravel(), y\_score.ravel())  
roc\_auc[**"micro"**] = auc(fpr[**"micro"**], tpr[**"micro"**])  
  
*# Plot of a ROC curve for a specific class*plt.figure()  
lw = 2*# linewdith=2*plt.plot(fpr[2], tpr[2], color=**'darkorange'**,  
 lw=lw, label=**'ROC curve (area = %0.2f)'** % roc\_auc[2])  
plt.plot([0, 1], [0, 1], color=**'navy'**, lw=lw, linestyle=**'--'**)  
plt.xlim([0.0, 1.0])  
plt.ylim([0.0, 1.05])  
plt.xlabel(**'False Positive Rate'**)  
plt.ylabel(**'True Positive Rate'**)  
plt.title(**'Receiver operating characteristic example'**)  
plt.legend(loc=**"lower right"**)  
plt.show()

# 参考：

1. [http://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc\_curve.html](http://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html" \t "http://blog.csdn.net/shenxiaoming77/article/details/_blank)
2. ROC计算公式，[http://blog.yhat.com/posts/roc-curves.html](http://blog.yhat.com/posts/roc-curves.html" \t "http://blog.csdn.net/shenxiaoming77/article/details/_blank)