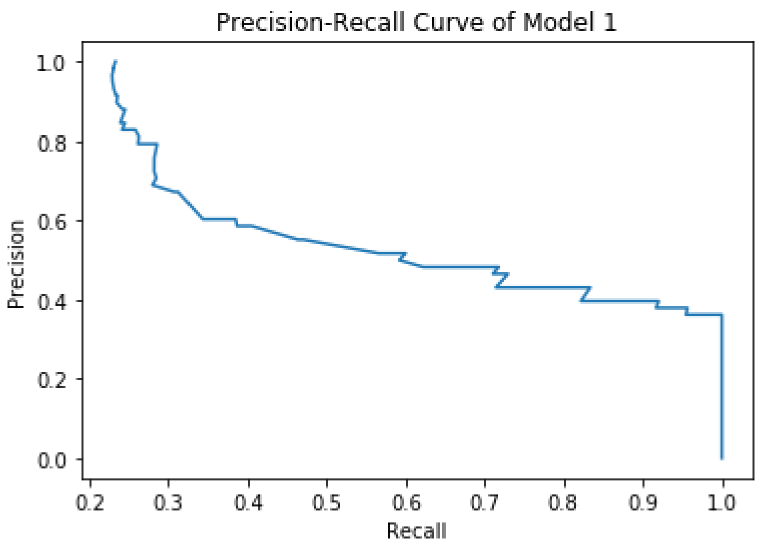
机器学习第六次作业

- 函数调用声明

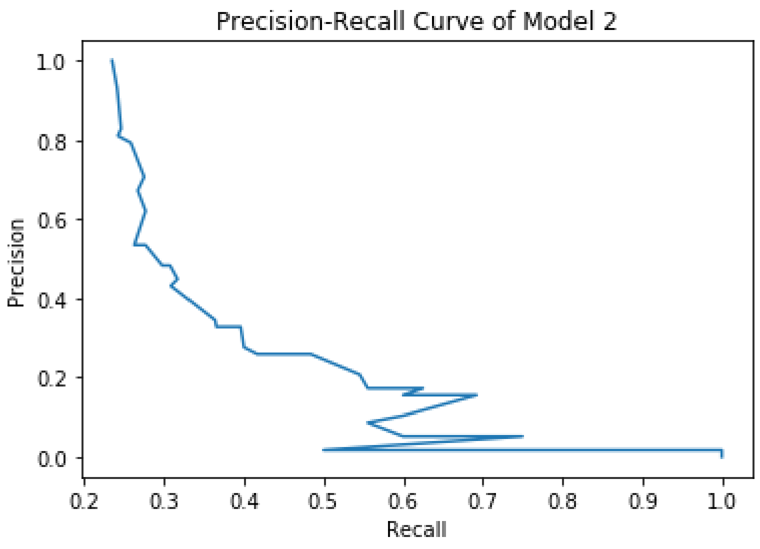
```
import pandas as pd
import numpy as np

import matplotlib
import matplotlib.pyplot as plt
from sklearn.metrics import precision_recall_curve
from sklearn.utils.fixes import signature
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import confusion_matrix
from sklearn.metrics import average_precision_score
```

- 钓鱼数据集

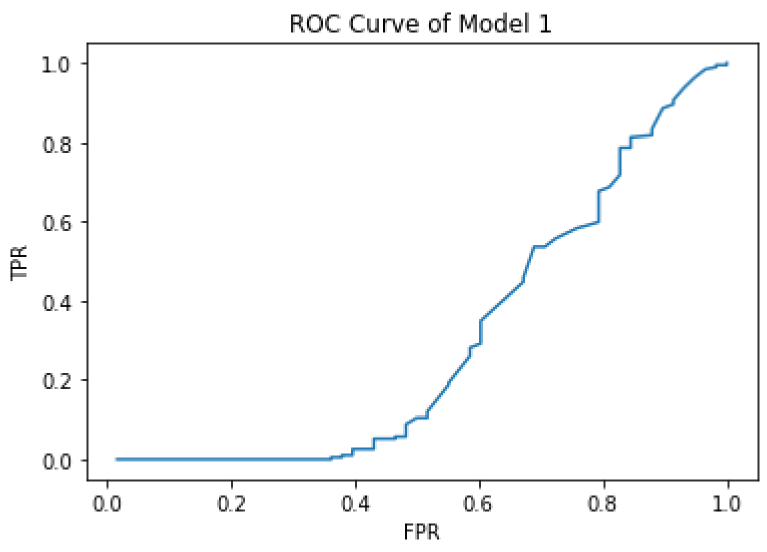
```
#读入数据
fishing_data = pd.read_excel('C:/Users/mi/Desktop/fishing.xlsx')

### 绘制第一个模型的P-R曲线
plt.figure(1)
plt.title('Precision-Recall Curve of Model 1')
plt.xlabel('Recall')
plt.ylabel('Precision')

y_label = fishing_data['是否钓到鱼']
y_pred = fishing_data['模型1预测值: 钓鱼']
precision, recall, thresholds = precision_recall_curve(y_label, y_pred)
plt.figure(1)
plt.plot(precision, recall)
plt.show()
```



```
### 绘制第二个模型的P-R曲线
plt.figure(2)
plt.title('Precision-Recall Curve of Model 2')
plt.xlabel('Recall')
plt.ylabel('Precision')

y_label = fishing_data['是否钓到鱼']
y_pred = fishing_data['模型2预测值: 钓鱼']
precision, recall, thresholds = precision_recall_curve(y_label, y_pred)
plt.figure(2)
plt.plot(precision, recall)
plt.show()
```

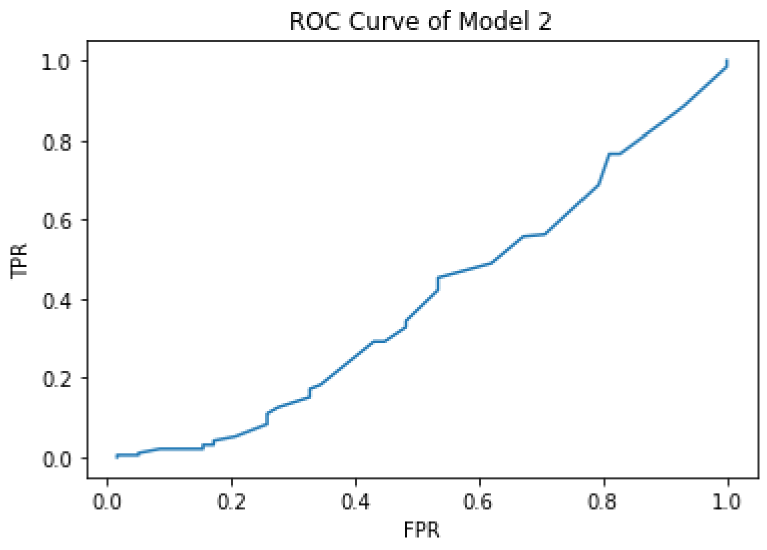


```
### 绘制第一个模型的ROC曲线并计算AUC
plt.figure(3)
plt.title('ROC Curve of Model 1')
plt.xlabel('FPR')
plt.ylabel('TPR')

y_label = fishing_data['是否钓到鱼']
y_pred = fishing_data['模型1预测值: 钓鱼']
fpr,tpr,threshold = roc_curve(y_label, y_pred)
plt.figure(3)
plt.plot(tpr, fpr)
plt.show()

### 输出AUC的值
roc_auc_1 = auc(fpr,tpr)
print('AUC of Model 1 is ' + roc_auc_1.astype(str))
```



AUC of Model 1 is 0.7007902298850576

```
### 绘制第二个模型的ROC曲线并计算AUC
plt.figure(4)
plt.title('ROC Curve of Model 2')
plt.xlabel('FPR')
plt.ylabel('TPR')

y_label = fishing_data['是否钓到鱼']
y_pred = fishing_data['模型2预测值: 钓鱼']
fpr,tpr,threshold = roc_curve(y_label, y_pred)
plt.figure(4)
plt.plot(tpr, fpr)
plt.show()

### 输出AUC的值
roc_auc_2 = auc(fpr,tpr)
print('AUC of Model 2 is ' + roc_auc_2.astype(str))
```



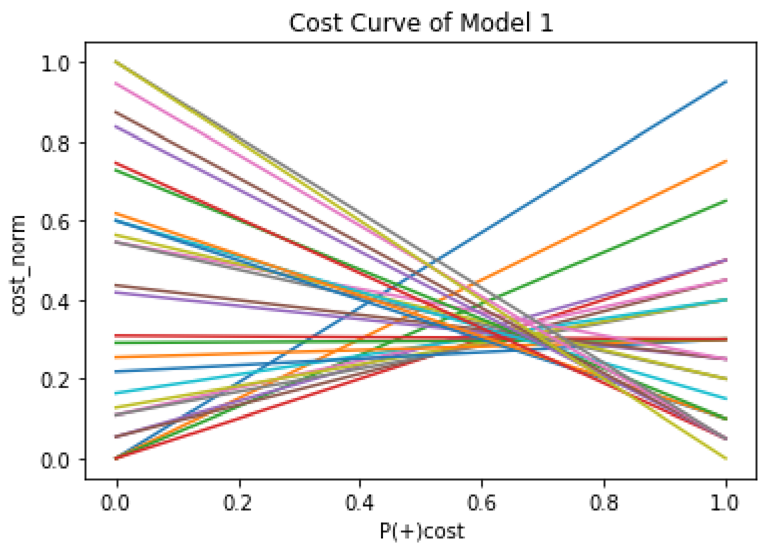
AUC of Model 2 is 0.6075790229885057

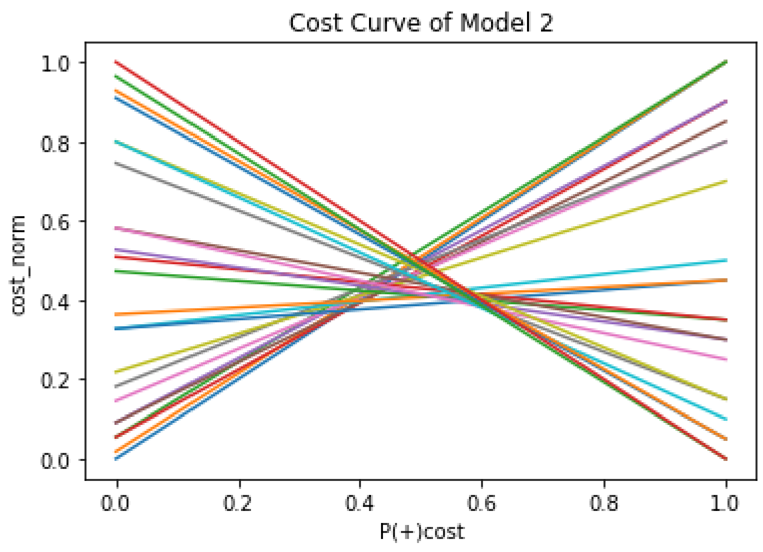
```
# 根据AUC的大小来看，模型1的预测效果要优于模型2
# 在缺少实际背景的情况下，默认阈值的选择标准是兼顾查全率与查准率并具有较低的假正率，因此考虑F1度量。
# 选择使得F1度量最大的阈值作为分类阈值
```

```
### 确定模型1的最佳阈值
y_label = fishing_data['是否钓到鱼']
y_pred = fishing_data['模型1预测值: 钓鱼']
precision, recall, thresholds = precision_recall_curve(y_label, y_pred)

precision = pd.Series(precision)
recall = pd.Series(recall)

#F1_1 = 2*precision*recall/(precision + recall)
F1_1U = precision.multiply(recall)
F1_1D = 1/(precision + recall)
F1_1 = F1_1U.multiply(F1_1D)
F1_max = max(F1_1)
F1_index = F1_1[F1_1.values == F1_max].index
threshold_1 = thresholds[F1_index]
#输出阈值
#thresholds[46] 1.2234588847708732

y_true = fishing_data['是否钓到鱼']
y_pred = (y_pred > 1.2234588847708732).astype(int)
confusion_matrix_1 = confusion_matrix(y_true, y_pred, labels = [0,1])
#输出混淆矩阵
#   0     1
#0 136   56
#1 32    26


### 确定模型2的最佳阈值
y_label = fishing_data['是否钓到鱼']
y_pred = fishing_data['模型2预测值: 钓鱼']
precision, recall, thresholds = precision_recall_curve(y_label, y_pred)

precision = pd.Series(precision)
recall = pd.Series(recall)

#F1_2 = 2*precision*recall/(precision + recall)
F1_2U = precision.multiply(recall)
F1_2D = 1/(precision + recall)
F1_2 = F1_2U.multiply(F1_2D)
F1_max = max(F1_2)
F1_index = F1_2[F1_2.values == F1_max].index
threshold_2 = thresholds[F1_index]
#输出阈值
#thresholds[6] 0.63518535

y_true = fishing_data['是否钓到鱼']
y_pred = (y_pred > 0.63518535).astype(int)
confusion_matrix_2 = confusion_matrix(y_true, y_pred, labels = [0,1])
#输出混淆矩阵
#    0   1
# 0 85  107
# 1 19  39
```

```
###绘制模型1的代价曲线
```

```
y_label = Test['是否钓到鱼']
y_pred = Test['模型1预测值: 钓鱼']
fpr,tpr,threshold = roc_curve(y_label, y_pred)
plt.figure(5)
plt.title('Cost Curve of Model 1')
plt.xlabel('P(+)cost')
plt.ylabel('cost_norm')
plt.plot([0,1],[fpr, 1 - tpr])
plt.show()

###绘制模型2的代价曲线

y_label = Test['是否钓到鱼']
y_pred = Test['模型2预测值: 钓鱼']
fpr,tpr,threshold = roc_curve(y_label, y_pred)
plt.figure(6)
plt.title('Cost Curve of Model 2')
plt.xlabel('P(+)cost')
plt.ylabel('cost_norm')
plt.plot([0,1],[fpr, 1 - tpr])
plt.show()

# 可以较直观地看到，模型2的期望总代价大于模型1的期望总代价
# 因此在非平衡代价损失的条件下，模型1优于模型2
```





- 手写体数据集

```
handwriting_data = pd.read_excel('C:/Users/mi/Desktop/handwriting.xlsx')

# 计算宏查全率和宏差准率
y_label = handwriting_data['数据标签']
y_pred_1 = handwriting_data['模型1预测']
y_pred_2 = handwriting_data['模型2预测']
y_pred_3 = handwriting_data['模型3预测']

precision_1, recall_1, thresholds_1 = precision_recall_curve(y_label,
y_pred_1)
precision_2, recall_2, thresholds_2 = precision_recall_curve(y_label,
y_pred_2)
precision_3, recall_3, thresholds_3 = precision_recall_curve(y_label,
y_pred_3)
```

```
# 模型1
## 计算宏查准率与宏查全率
macro_P_1 = np.mean(precision_1)
#0.8570437096912245

macro_R_1 = np.mean(recall_1)
#0.6577674897119341


## 计算微差准率与微查全率
TP = []
FP = []
TN = []
FN = []
thresholds_1 = list(thresholds_1)

for i in thresholds_1:
    y_pred = (y_pred_1 > i ).astype(int)
    matrix = confusion_matrix(y_label, y_pred, labels = [0,1])
    TP.append(matrix[1,1])
    FP.append(matrix[0,1])
    TN.append(matrix[0,0])
    FN.append(matrix[1,0])

TP_mean = np.mean(TP)
FP_mean = np.mean(FP)
TN_mean = np.mean(TN)
FN_mean = np.mean(FN)

micro_P_1 = TP_mean/(TP_mean + FP_mean)
#0.7824361840650566

micro_R_1 = TP_mean/(TP_mean + FN_mean)
#0.6569203795379538
```

```
# 模型2
macro_P_2 = np.mean(precision_2)
#0.5336255478748948
```

```
macro_R_2 = np.mean(recall_2)
#0.5193371607515658

TP = []
FP = []
TN = []
FN = []
thresholds_2 = list(thresholds_2)

for i in thresholds_2:
    y_pred = (y_pred_2 > i ).astype(int)
    matrix = confusion_matrix(y_label, y_pred, labels = [0,1])
    TP.append(matrix[1,1])
    FP.append(matrix[0,1])
    TN.append(matrix[0,0])
    FN.append(matrix[1,0])

TP_mean = np.mean(TP)
FP_mean = np.mean(FP)
TN_mean = np.mean(TN)
FN_mean = np.mean(FN)

micro_P_2 = TP_mean/(TP_mean + FP_mean)
#0.5215915370648141

micro_R_2 = TP_mean/(TP_mean + FN_mean)
#0.0.518331589958159
```

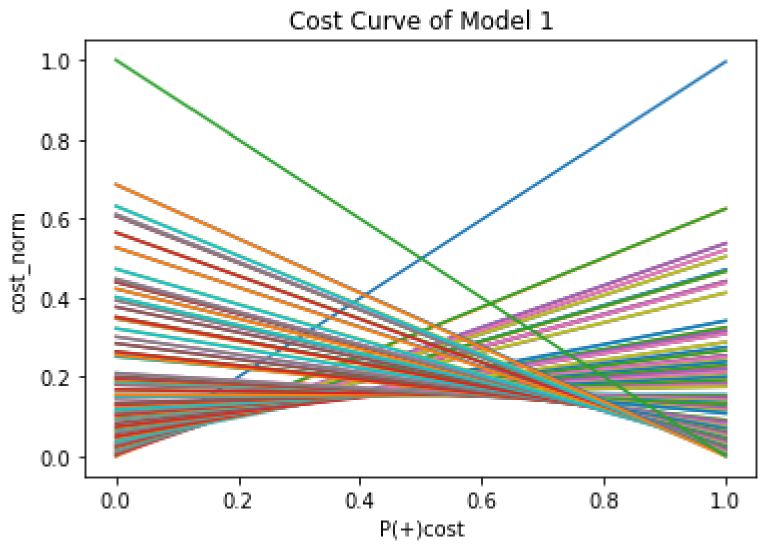
```
# 模型3
macro_P_3 = np.mean(precision_3)
#0.562709287216303

macro_R_3 = np.mean(recall_3)
#0.5292103424178896

TP = []
FP = []
TN = []
FN = []
thresholds_3 = list(thresholds_3)

for i in thresholds_3:
    y_pred = (y_pred_3 > i ).astype(int)
    matrix = confusion_matrix(y_label, y_pred, labels = [0,1])
    TP.append(matrix[1,1])
    FP.append(matrix[0,1])
    TN.append(matrix[0,0])
    FN.append(matrix[1,0])

TP_mean = np.mean(TP)
FP_mean = np.mean(FP)
TN_mean = np.mean(TN)
FN_mean = np.mean(FN)

micro_P_3 = TP_mean/(TP_mean + FP_mean)
#0.533781512605042
```

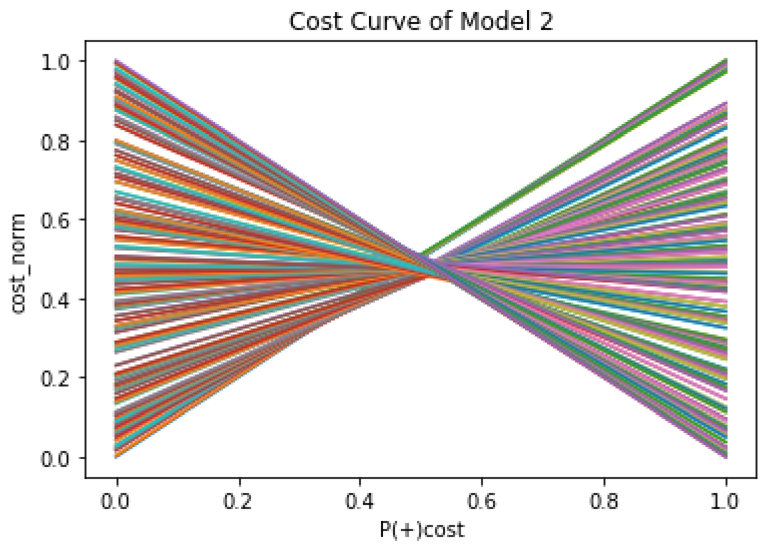
```
micro_R_3 = TP_mean/(TP_mean + FN_mean)
#0.5282212885154062
```

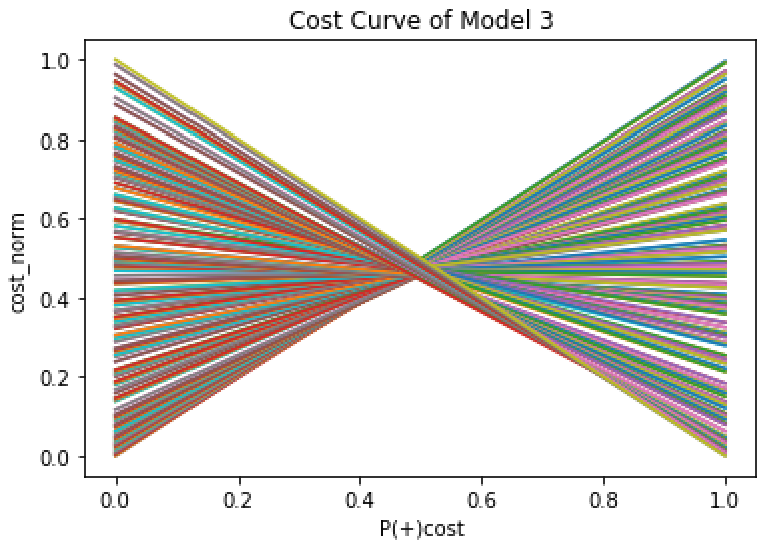
```
#采用使得F1最大的模型预测作为分类阈值

#模型1
precision_1 = pd.Series(precision_1)
recall_1 = pd.Series(recall_1)

#F1_1 = 2*precision*recall/(precision + recall)
F1_1U = precision_1.multiply(recall_1)
F1_1D = 1/(precision_1 + recall_1)
F1_1 = F1_1U.multiply(F1_1D)
F1_max = max(F1_1)
F1_index = F1_1[F1_1.values == F1_max].index
threshold_1 = thresholds_1[F1_index]
#输出阈值
#thresholds[163] 0.56287985

#模型2
precision_2 = pd.Series(precision_2)
recall_2 = pd.Series(recall_2)

#F1_1 = 2*precision*recall/(precision + recall)
F1_2U = precision_2.multiply(recall_2)
F1_2D = 1/(precision_2 + recall_2)
F1_2 = F1_2U.multiply(F1_2D)
F1_max = max(F1_2)
F1_index = F1_2[F1_2.values == F1_max].index
threshold_2 = thresholds_2[F1_index]
#输出阈值
#thresholds[6] 0.26126596

#模型3
precision_3 = pd.Series(precision_3)
recall_3 = pd.Series(recall_3)

#F1_1 = 2*precision*recall/(precision + recall)
F1_3U = precision_3.multiply(recall_3)
F1_3D = 1/(precision_3 + recall_3)
F1_3 = F1_3U.multiply(F1_3D)
F1_max = max(F1_3)
F1_index = F1_3[F1_3.values == F1_max].index
threshold_3 = thresholds_3[F1_index]
#输出阈值
#thresholds[41] 0.39014812
```

```
## 模型1的期望代价曲线
fpr,tpr,threshold = roc_curve(y_label, y_pred_1)
plt.figure(7)
plt.title('Cost Curve of Model 1')
plt.xlabel('P(+)cost')
plt.ylabel('cost_norm')
plt.plot([0,1],[fpr, 1 - tpr])
plt.show()
```

```
## 模型2的期望代价曲线

fpr,tpr,threshold = roc_curve(y_label, y_pred_2)
plt.figure(8)
plt.title('Cost Curve of Model 2')
plt.xlabel('P(+)cost')
plt.ylabel('cost_norm')
plt.plot([0,1],[fpr, 1 - tpr])
plt.show()



## 模型3的期望代价曲线

fpr,tpr,threshold = roc_curve(y_label, y_pred_3)
plt.figure(9)
plt.title('Cost Curve of Model 3')
plt.xlabel('P(+)cost')
plt.ylabel('cost_norm')
plt.plot([0,1],[fpr, 1 - tpr])
plt.show()
```







K折t检验计算结果

模型1	模型2	模型3
0.1762750879289662	0.48551979875930057	0.48503948547796194

平均错误率μ与ϵ之差均位于$\alpha = 0.1$的临界值范围内，接受原假设

伯努利误差计算结果

模型1	模型2	模型3
1.611556138135071e-07	3.49133531744815e-103	2.9921977772517855e-102

无法拒绝错误率为0.1 的假设

K折	模型1	模型2	模型3
1	1	2.5	2.5
2	1	2.5	2.5
3	1	3	2
4	1	2	3
5	1	3	2
平均序值	1	2.6	2.4

$$\tau_F \approx -0.886 < 4.459$$

在0.01的显著性下不能拒绝原假设