实验名称：模型评估

实验人：刘逸杰，学号：202210310115

一、实验目的

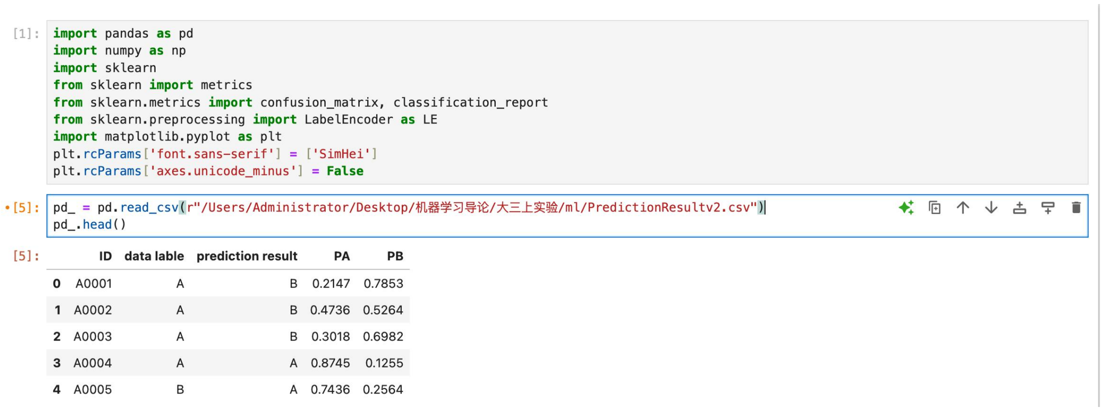
了解机器学习的模型的评估方法，学会使用评估方法区评估一个模型，并给出模型的好坏。

二、实验内容

1. 计算模型预测结果的混淆矩阵。
2. 计算模型预测结果的精度、错误率。
3. 画出 ROC 曲线并计算 AUC 值。

三、实验步骤

代码结果与过程分析：



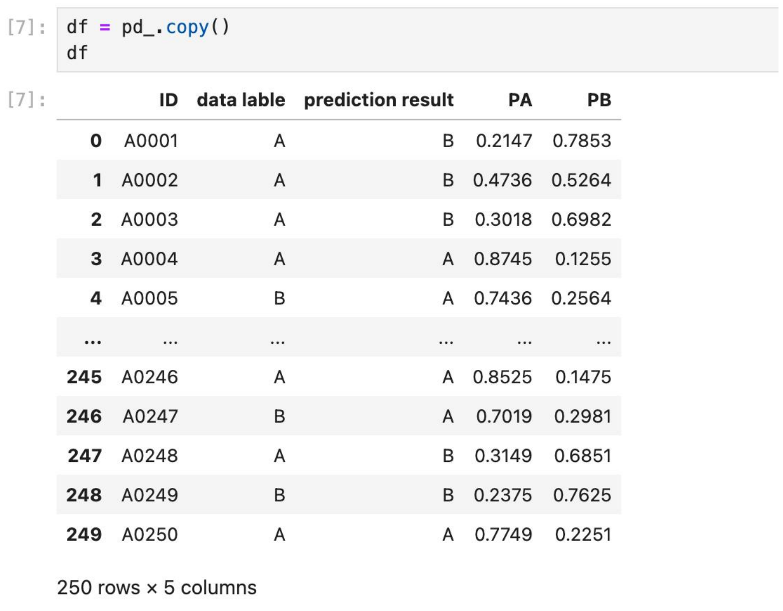
```
[1]: import pandas as pd
     import numpy as np
     import sklearn
     from sklearn import metrics
     from sklearn.metrics import confusion_matrix, classification_report
     from sklearn.preprocessing import LabelEncoder as LE
     import matplotlib.pyplot as plt
     plt.rcParams['font.sans-serif'] = ['SimHei']
     plt.rcParams['axes.unicode_minus'] = False
```

```
•[5]: pd_ = pd.read_csv(r"/Users/Administrator/Desktop/机器学习导论/大三上实验/ml/PredictionResultv2.csv")
      pd_.head()
```

[5]:

	ID	data lable	prediction result	PA	PB
0	A0001	A	B	0.2147	0.7853
1	A0002	A	B	0.4736	0.5264
2	A0003	A	B	0.3018	0.6982
3	A0004	A	A	0.8745	0.1255
4	A0005	B	A	0.7436	0.2564

导入库与工具，设置字体，然后再传入数据



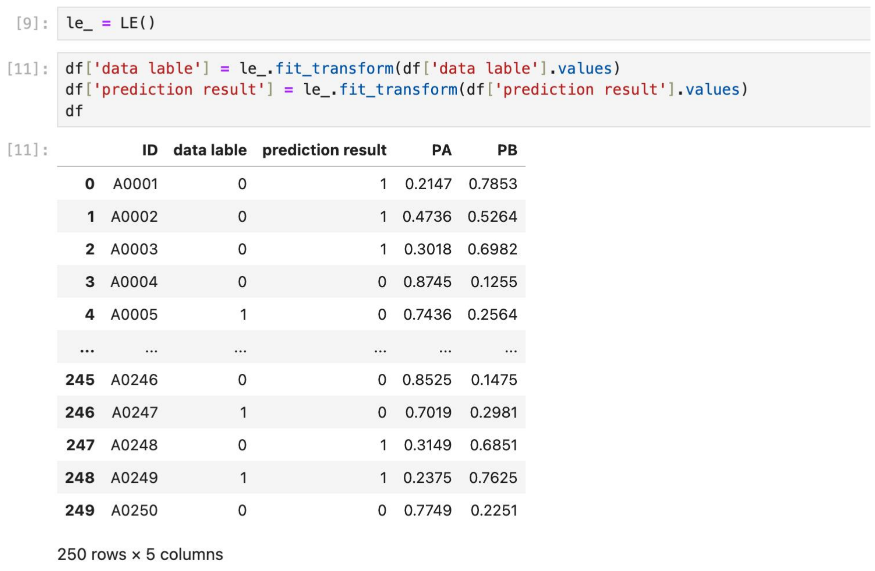
```
[7]: df = pd_.copy()
     df
```

[7]:

	ID	data lable	prediction result	PA	PB
0	A0001	A	B	0.2147	0.7853
1	A0002	A	B	0.4736	0.5264
2	A0003	A	B	0.3018	0.6982
3	A0004	A	A	0.8745	0.1255
4	A0005	B	A	0.7436	0.2564
...	...	...	...	...	...
245	A0246	A	A	0.8525	0.1475
246	A0247	B	A	0.7019	0.2981
247	A0248	A	B	0.3149	0.6851
248	A0249	B	B	0.2375	0.7625
249	A0250	A	A	0.7749	0.2251

250 rows × 5 columns

创建副本



```
[9]:  le_ = LE()
```

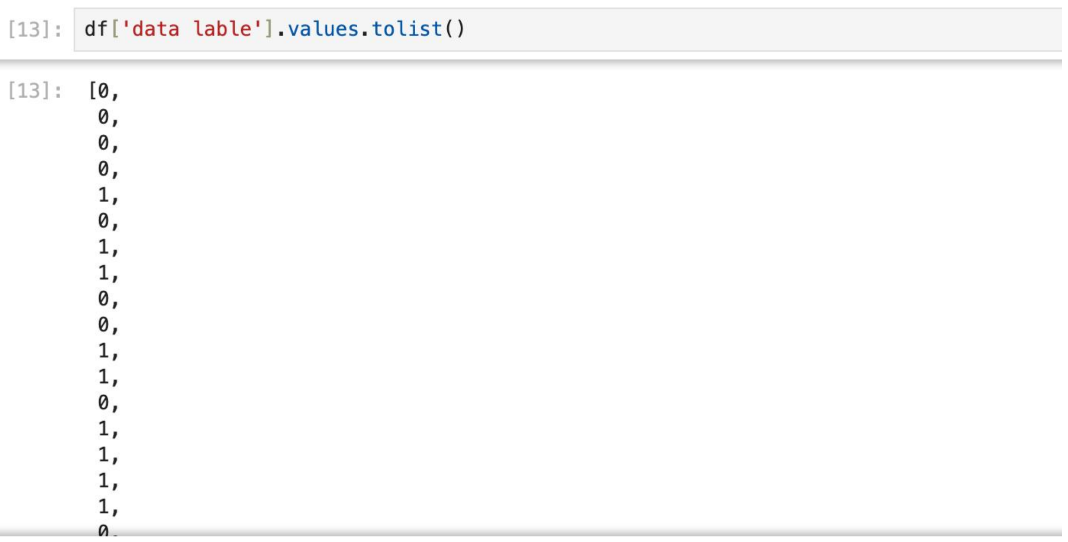
```
[11]: df['data lable'] = le_.fit_transform(df['data lable'].values)
      df['prediction result'] = le_.fit_transform(df['prediction result'].values)
      df
```

[11]:

	ID	data lable	prediction result	PA	PB
0	A0001	0	1	0.2147	0.7853
1	A0002	0	1	0.4736	0.5264
2	A0003	0	1	0.3018	0.6982
3	A0004	0	0	0.8745	0.1255
4	A0005	1	0	0.7436	0.2564
...	...	...	...	...	...
245	A0246	0	0	0.8525	0.1475
246	A0247	1	0	0.7019	0.2981
247	A0248	0	1	0.3149	0.6851
248	A0249	1	1	0.2375	0.7625
249	A0250	0	0	0.7749	0.2251

250 rows × 5 columns

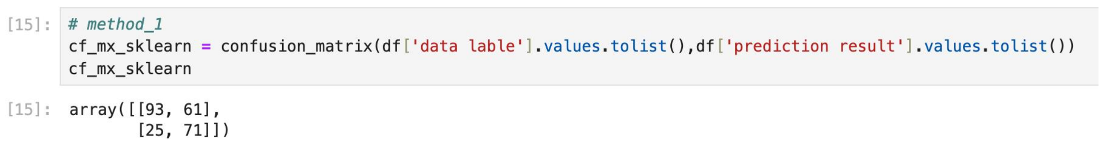
创建 LE 实例，转换数据标签，显示更新后的 DataFrame



```
[13]: df['data lable'].values.tolist()
```

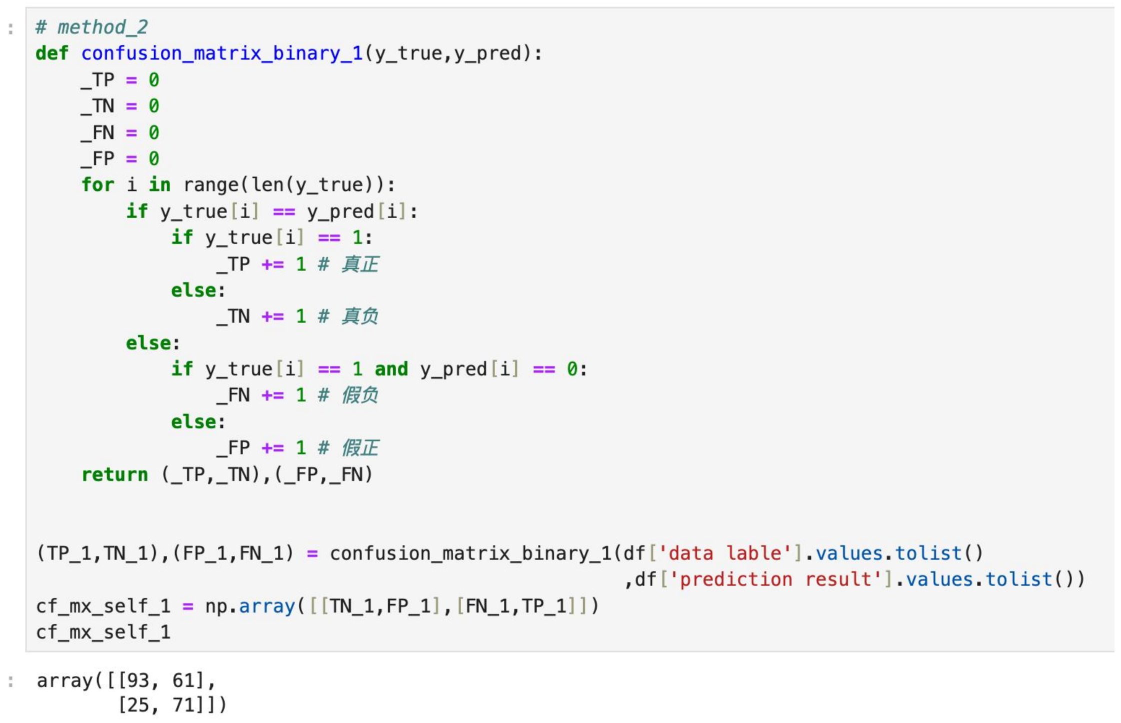
```
[13]: [0,
       0,
       0,
       0,
       1,
       0,
       1,
       1,
       0,
       0,
       1,
       1,
       0,
       1,
       1,
       1,
       1,
       0
```

提取值，转换为列表，赋值



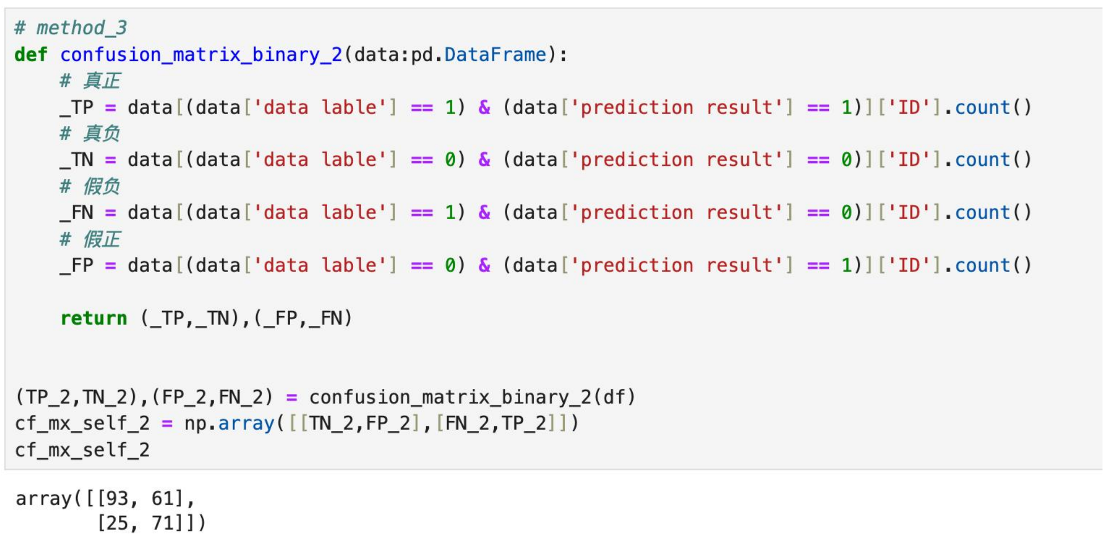
```
[15]: # method_1
      cf_mx_sklearn = confusion_matrix(df['data lable'].values.tolist(),df['prediction result'].values.tolist())
      cf_mx_sklearn
```

```
[15]: array([[93, 61],
             [25, 71]])
```



```
# method_2
def confusion_matrix_binary_1(y_true,y_pred):
    _TP = 0
    _TN = 0
    _FN = 0
    _FP = 0
    for i in range(len(y_true)):
        if y_true[i] == y_pred[i]:
            if y_true[i] == 1:
                _TP += 1 # 真正
            else:
                _TN += 1 # 真负
        else:
            if y_true[i] == 1 and y_pred[i] == 0:
                _FN += 1 # 假负
            else:
                _FP += 1 # 假正
    return (_TP,_TN),(_FP,_FN)


(TP_1,TN_1),(FP_1,FN_1) = confusion_matrix_binary_1(df['data lable'].values.tolist()
                                                    ,df['prediction result'].values.tolist())
cf_mx_self_1 = np.array([[TN_1,FP_1],[FN_1,TP_1]])
cf_mx_self_1
```

```
array([[93, 61],
       [25, 71]])
```

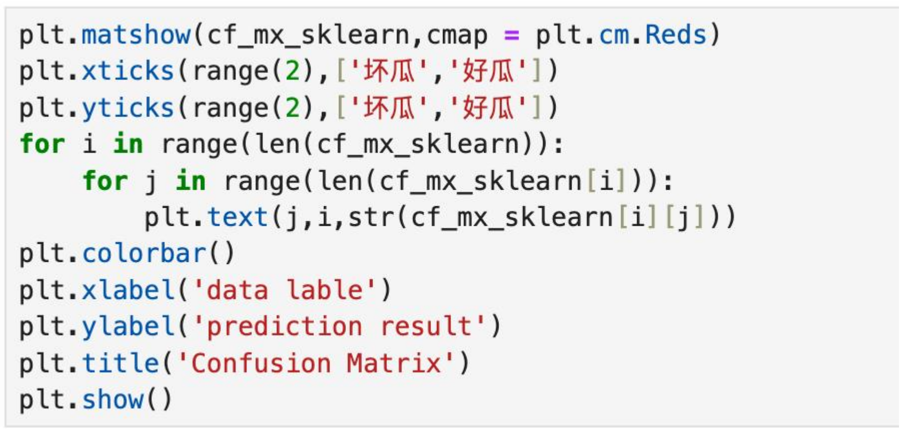


```
# method_3
def confusion_matrix_binary_2(data:pd.DataFrame):
    # 真正
    _TP = data[(data['data lable'] == 1) & (data['prediction result'] == 1)]['ID'].count()
    # 真负
    _TN = data[(data['data lable'] == 0) & (data['prediction result'] == 0)]['ID'].count()
    # 假负
    _FN = data[(data['data lable'] == 1) & (data['prediction result'] == 0)]['ID'].count()
    # 假正
    _FP = data[(data['data lable'] == 0) & (data['prediction result'] == 1)]['ID'].count()

    return (_TP,_TN),(_FP,_FN)


(TP_2,TN_2),(FP_2,FN_2) = confusion_matrix_binary_2(df)
cf_mx_self_2 = np.array([[TN_2,FP_2],[FN_2,TP_2]])
cf_mx_self_2
```

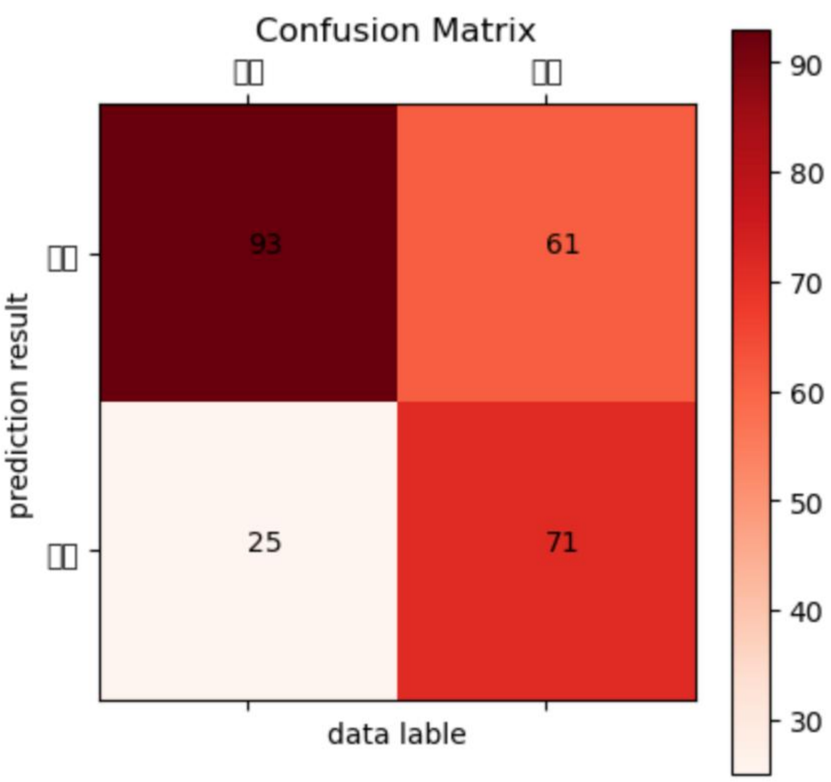
```
array([[93, 61],
       [25, 71]])
```

使用三种方法计算混淆矩阵:

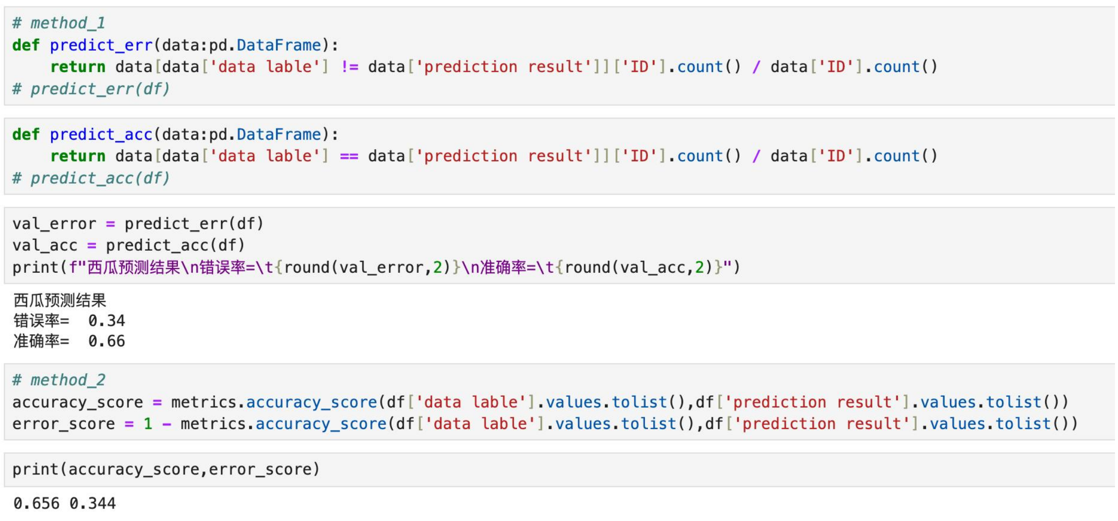
1. 直接调用 sklearn 中的函数计算
2. 遍历列表计算
3. 调用 pandas 中的函数计算



```
plt.matshow(cf_mx_sklearn,cmap = plt.cm.Reds)
plt.xticks(range(2),['坏瓜','好瓜'])
plt.yticks(range(2),['坏瓜','好瓜'])
for i in range(len(cf_mx_sklearn)):
    for j in range(len(cf_mx_sklearn[i])):
        plt.text(j,i,str(cf_mx_sklearn[i][j]))
plt.colorbar()
plt.xlabel('data lable')
plt.ylabel('prediction result')
plt.title('Confusion Matrix')
plt.show()
```



绘制混淆矩阵



```
# method_1
def predict_err(data:pd.DataFrame):
    return data[data['data lable'] != data['prediction result']]['ID'].count() / data['ID'].count()
# predict_err(df)
```

```
def predict_acc(data:pd.DataFrame):
    return data[data['data lable'] == data['prediction result']]['ID'].count() / data['ID'].count()
# predict_acc(df)
```

```
val_error = predict_err(df)
val_acc = predict_acc(df)
print(f"西瓜预测结果\n错误率=\t{round(val_error,2)}\n准确率=\t{round(val_acc,2)}")
```

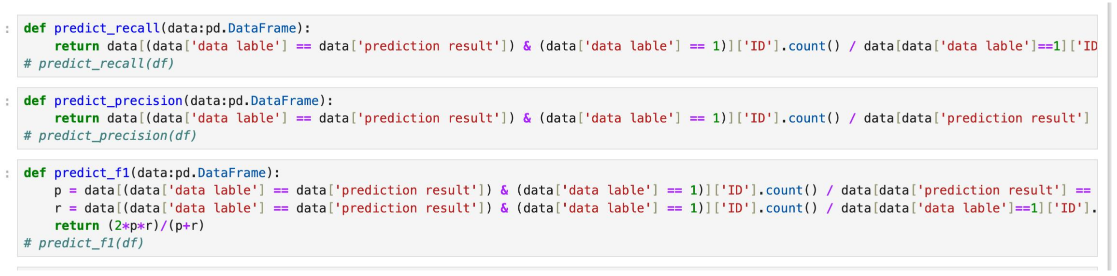
```
西瓜预测结果
错误率=	0.34
准确率=	0.66
```

```
# method_2
accuracy_score = metrics.accuracy_score(df['data lable'].values.tolist(),df['prediction result'].values.tolist())
error_score = 1 - metrics.accuracy_score(df['data lable'].values.tolist(),df['prediction result'].values.tolist())
```

```
print(accuracy_score,error_score)
```

```
0.656 0.344
```

使用两种方法定义预测准确率函数与预测错误率函数，计算验证错误率和准确率，第一种是自定义函数实现，第二种是调用 sklearn 库函数实现。



```
def predict_recall(data:pd.DataFrame):
    return data[(data['data lable'] == data['prediction result']) & (data['data lable'] == 1)]['ID'].count() / data[data['data lable']==1]['ID
# predict_recall(df)
```

```
def predict_precision(data:pd.DataFrame):
    return data[(data['data lable'] == data['prediction result']) & (data['data lable'] == 1)]['ID'].count() / data[data['prediction result']
# predict_precision(df)
```

```
def predict_f1(data:pd.DataFrame):
    p = data[(data['data lable'] == data['prediction result']) & (data['data lable'] == 1)]['ID'].count() / data[data['prediction result'] ==
    r = data[(data['data lable'] == data['prediction result']) & (data['data lable'] == 1)]['ID'].count() / data[data['data lable']==1]['ID'].
    return (2*p*r)/(p+r)
# predict_f1(df)
```

```python
# method_1
val_recall = predict_recall(df)
val_precision = predict_precision(df)
val_f1_score = predict_f1(df)
print(f"西瓜预测结果\n查全率=\t{round(val_recall,2)}\n查准率=\t{round(val_precision,2)}\nf1值=\t{round(val_f1_score,2)}")
```

```
西瓜预测结果
查全率=  0.74
查准率=  0.54
f1值=    0.62
```

```python
# method_2
def recall_precision_F1(data):
    (_TP,_TN),(_FP,_FN) = confusion_matrix_binary_2(data)
    _P = _TP / (_TP+_FP)
    _R = _TP / (_TP+_FN)
    _F1 = (2*_P*_R)/(_P+_R)
    return round(_R,2),round(_P,2),round(_F1,2)
R,P,F1 = recall_precision_F1(df)
print(f"西瓜预测结果\n查全率=\t{R}\n查准率=\t{P}\nf1值=\t{F1}")
```

```
西瓜预测结果
查全率=  0.74
查准率=  0.54
f1值=    0.62
```

```python
# method_3
precision_score = metrics.precision_score(df['data lable'].values.tolist(),df['prediction result'].values.tolist())
recall_score = metrics.recall_score(df['data lable'].values.tolist(),df['prediction result'].values.tolist())
f1_score = metrics.f1_score(df['data lable'].values.tolist(),df['prediction result'].values.tolist())
```

```python
print(precision_score,recall_score,f1_score)
```

```
0.5378787878787878 0.7395833333333334 0.6228070175438597
```

```python
# method_4
print(classification_report(df['data lable'].values.tolist(),
                            df['prediction result'].values.tolist()))
```

```
              precision    recall  f1-score   support

           0       0.79      0.60      0.68       154
           1       0.54      0.74      0.62        96

    accuracy                           0.66       250
   macro avg       0.66      0.67      0.65       250
weighted avg       0.69      0.66      0.66       250
```

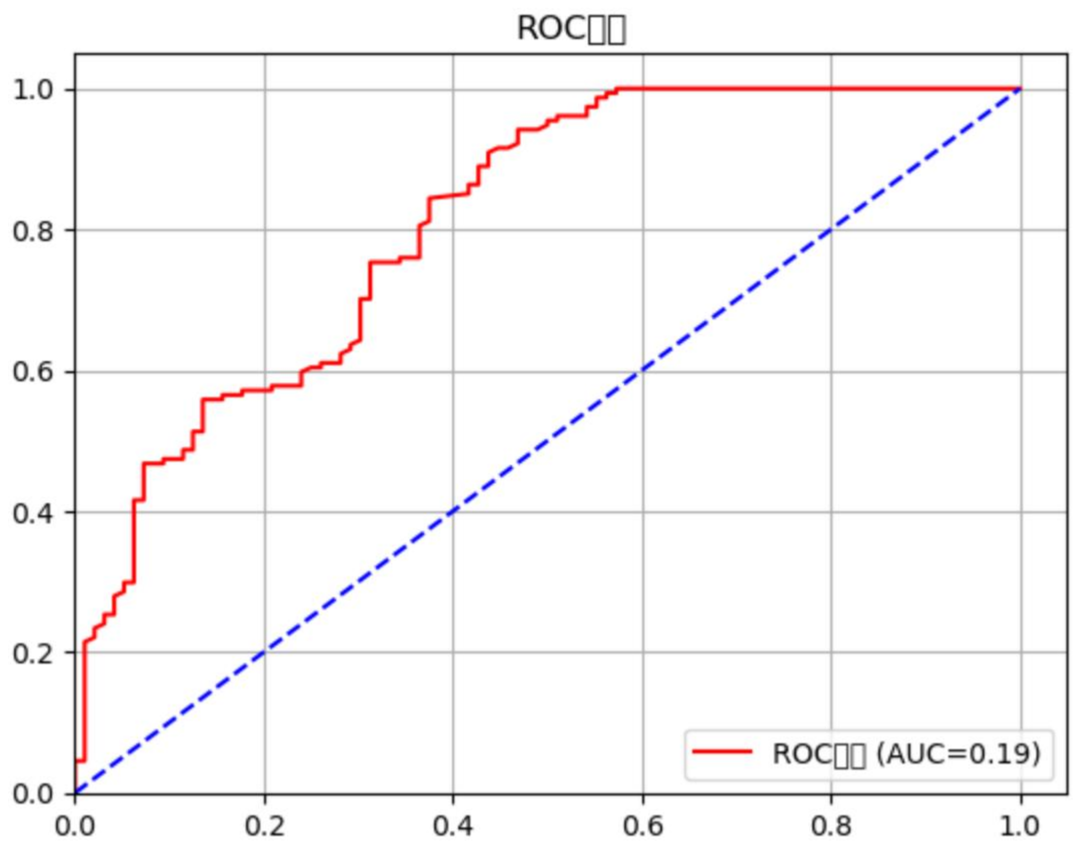
使用四种方法计算召回率、精准率、F1 分数
方法一：自定义函数 predict_recall、predict_precision 和 predict_f1
方法二：自定义函数 recall_precision_F1，混淆矩阵函数 confusion_matrix_binary_2
方法三：使用 sklearn.metrics
方法四：使用 classification_report 函数

```python
y = df['data lable'].values.tolist()
scores = df['PA'].values.tolist()
fpr, tpr, thresholds = metrics.roc_curve(y, scores, pos_label=1)
auc = metrics.auc(fpr, tpr)
auc
```

```
0.1893939393939394
```

提取标签和分数，计算 ROC 曲线，计算输出 AUC

```python
plt.figure()
plt.plot(tpr, fpr, color='red', label=f'ROC曲线 (AUC={round(auc,2)})')
plt.plot([0, 1], [0, 1], color='blue' , linestyle='--')
plt.xlim([.0, 1.05])
plt.ylim([.0, 1.05])
plt.xlabel('假正例率')
plt.ylabel('真正例率')
plt.title('ROC曲线')
plt.legend(loc="lower right")
plt.grid(True)
plt.show()
```



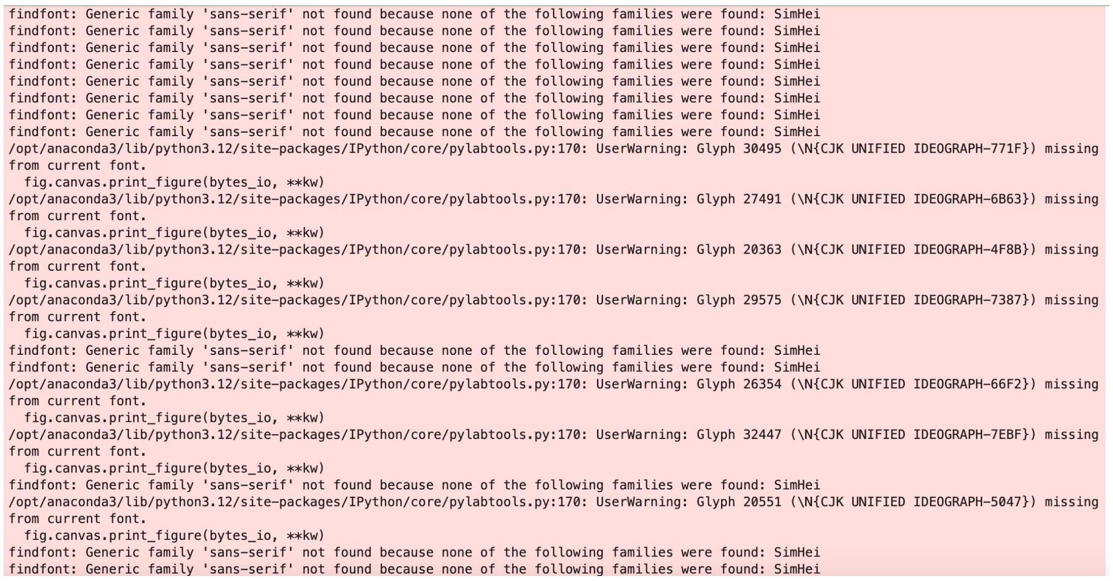
绘制 ROC 曲线

四、本次实验总结

实验中遇到的问题以及解决办法：

1.读入 excel 文件的时候出现问题，查询后选择了将其后缀改为 csv 然后使用 read_csv() 函数读入数据，询问同学后了解直接使用 windows 提供的路径需要使用’/’或者’\\’表示文件位置。

2.生成图表时出现了如下错误：



```
findfont: Generic family 'sans-serif' not found because none of the following families were found: SimHei
findfont: Generic family 'sans-serif' not found because none of the following families were found: SimHei
findfont: Generic family 'sans-serif' not found because none of the following families were found: SimHei
findfont: Generic family 'sans-serif' not found because none of the following families were found: SimHei
findfont: Generic family 'sans-serif' not found because none of the following families were found: SimHei
findfont: Generic family 'sans-serif' not found because none of the following families were found: SimHei
findfont: Generic family 'sans-serif' not found because none of the following families were found: SimHei
findfont: Generic family 'sans-serif' not found because none of the following families were found: SimHei
/opt/anaconda3/lib/python3.12/site-packages/IPython/core/pylabtools.py:170: UserWarning: Glyph 30495 (\N{CJK UNIFIED IDEOGRAPH-771F}) missing
from current font.
  fig.canvas.print_figure(bytes_io, **kw)
/opt/anaconda3/lib/python3.12/site-packages/IPython/core/pylabtools.py:170: UserWarning: Glyph 27491 (\N{CJK UNIFIED IDEOGRAPH-6B63}) missing
from current font.
  fig.canvas.print_figure(bytes_io, **kw)
/opt/anaconda3/lib/python3.12/site-packages/IPython/core/pylabtools.py:170: UserWarning: Glyph 20363 (\N{CJK UNIFIED IDEOGRAPH-4F8B}) missing
from current font.
  fig.canvas.print_figure(bytes_io, **kw)
/opt/anaconda3/lib/python3.12/site-packages/IPython/core/pylabtools.py:170: UserWarning: Glyph 29575 (\N{CJK UNIFIED IDEOGRAPH-7387}) missing
from current font.
  fig.canvas.print_figure(bytes_io, **kw)
findfont: Generic family 'sans-serif' not found because none of the following families were found: SimHei
findfont: Generic family 'sans-serif' not found because none of the following families were found: SimHei
/opt/anaconda3/lib/python3.12/site-packages/IPython/core/pylabtools.py:170: UserWarning: Glyph 26354 (\N{CJK UNIFIED IDEOGRAPH-66F2}) missing
from current font.
  fig.canvas.print_figure(bytes_io, **kw)
/opt/anaconda3/lib/python3.12/site-packages/IPython/core/pylabtools.py:170: UserWarning: Glyph 32447 (\N{CJK UNIFIED IDEOGRAPH-7EBF}) missing
from current font.
  fig.canvas.print_figure(bytes_io, **kw)
findfont: Generic family 'sans-serif' not found because none of the following families were found: SimHei
/opt/anaconda3/lib/python3.12/site-packages/IPython/core/pylabtools.py:170: UserWarning: Glyph 20551 (\N{CJK UNIFIED IDEOGRAPH-5047}) missing
from current font.
  fig.canvas.print_figure(bytes_io, **kw)
findfont: Generic family 'sans-serif' not found because none of the following families were found: SimHei
findfont: Generic family 'sans-serif' not found because none of the following families were found: SimHei
```

但是能够生成成功，根据返回的错误信息来看是我调用 simhei 中文字库失败，最后的处理结果似乎是跳过了该步骤直接生成图表并且返回的原本应该出现中文字符的地方出现

方框。