# 【基础+实践】随机森林预测心脏病患者

秋枫学习笔记 2022-03-31 10:42

以下文章来源于尤而小屋 ，作者尤而小屋

**尤而小屋**
尤而小屋，一个温馨且有爱的小屋🏠 小屋主人，一手代码谋求生存，一手掌勺享受生活，欢迎你的光临~

秋枫学习笔记
分享数据挖掘，机器学习，推荐系统等知识和学习笔记
118篇原创内容

公众号

给大家分享一个新的kaggle案例：**基于随机森林模型（RandomForest）的心脏病人预测分类**。本文涉及到的知识点主要包含：

- 数据预处理和类型转化
- 随机森林模型建立与解释
- 决策树如何可视化
- 基于混淆矩阵的分类评价指标
- 部分依赖图PDP的绘制和解释
- AutoML机器学习SHAP库的使用和解释（待提升）


导读

> Of all the applications of machine-learning, diagnosing any serious disease using a black box is always going to be a hard sell. If the output from a model is the particular course of treatment (potentially with side-effects), or surgery, or the *absence* of treatment, people are going to want to know **why**.
>
> 在机器学习的所有应用中，使用黑匣子诊断任何严重疾病总是很难的。如果模型的输出是特定的治疗过程（可能有副作用）、手术或是否有疗效，人们会想知道为什么。

> This dataset gives a number of variables along with a target condition of having or not having heart disease. Below, the data is first used in a simple random forest model, and then the model is investigated using ML explainability tools and techniques.
>
> 该数据集提供了许多变量以及患有或不患有心脏病的目标条件。下面，数据首先用于一个简单的随机森林模型，然后使用 ML 可解释性工具和技术对该模型进行研究。

感兴趣的可以参考原文学习，notebook地址为：https://www.kaggle.com/tentotheminus9/what-causes-heart-disease-explaining-the-model

## 导入库

本案例中涉及到多个不同方向的库：

- 数据预处理
- 多种可视化绘图；尤其是**shap的可视化**，模型可解释性的使用（后面会专门写这个库）
- 随机森林模型
- 模型评价等

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import export_graphviz
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
import eli5
from eli5.sklearn import PermutationImportance
import shap
from pdpbox import pdp, info_plots
np.random.seed(123)

pd.options.mode.chained_assignment = None
```

## 数据探索EDA

1、导入数据

```
In [2]: df = pd.read_excel("heart.xlsx")
        df.head()
```

Out[2]:

| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | target |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 63 | 1 | 3 | 145 | 233 | 1 | 0 | 150 | 0 | 2.3 | 0 | 0 | 1 | 1 |
| 1 | 37 | 1 | 2 | 130 | 250 | 0 | 1 | 187 | 0 | 3.5 | 0 | 0 | 2 | 1 |
| 2 | 41 | 0 | 1 | 130 | 204 | 0 | 0 | 172 | 0 | 1.4 | 2 | 0 | 2 | 1 |
| 3 | 56 | 1 | 1 | 120 | 236 | 0 | 1 | 178 | 0 | 0.8 | 2 | 0 | 2 | 1 |
| 4 | 57 | 0 | 0 | 120 | 354 | 0 | 1 | 163 | 1 | 0.6 | 2 | 0 | 2 | 1 |



2、缺失值情况

数据比较完美，没有任何缺失值！

```
In [4]: df.isnull().sum()

Out[4]: age         0
        sex         0
        cp          0
        trestbps    0
        chol        0
        fbs         0
        restecg     0
        thalach     0
        exang       0
        oldpeak     0
        slope       0
        ca          0
        thal        0
        target      0
        dtype: int64
```



字段含义

在这里重点介绍下各个字段的含义。Peter近期导出的数据集中的额字段和原notebook中的字段名字写法稍有差异（时间原因导致），还好Peter已经为大家做了一一对应的关系，下面是具体的中文含义：

1. age：年龄
2. sex 性别 1=male 0=female
3. cp 胸痛类型：4种取值情况
   - 1：典型心绞痛
   - 2：非典型心绞痛
   - 3：非心绞痛
   - 4：无症状
4. trestbps 静息血压
5. chol 血清胆固醇
6. fbs 空腹血糖 >120mg/dl ： 1=true；0=false
7. restecg 静息心电图(值0,1,2)
8. thalach 达到的最大心率
9. exang 运动诱发的心绞痛(1=yes;0=no)
10. oldpeak 相对于休息的运动引起的ST值(ST值与心电图上的位置有关)
11. slope 运动高峰ST段的坡度

- 1：upsloping向上倾斜
- 2：flat持平
- 3：downsloping向下倾斜

12. ca The number of major vessels(血管) (0-3)

13. thal A blood disorder called thalassemia，一种叫做地中海贫血的血液疾病(3 = normal；6 = fixed defect;；7 = reversable defect)

14. target 生病没有(0=no；1=yes)

原notebook中的英文含义：

- age: The person's age in years
- sex: The person's sex (1 = male, 0 = female)
- cp: The chest pain experienced (Value 1: typical angina, Value 2: atypical angina, Value 3: non-anginal pain, Value 4: asymptomatic) trestbps: The person's resting blood pressure (mm Hg on admission to the hospital)
- chol: The person's cholesterol measurement in mg/dl
- fbs: The person's fasting blood sugar (> 120 mg/dl, 1 = true; 0 = false)
- restecg: Resting electrocardiographic measurement (0 = normal, 1 = having ST-T wave abnormality, 2 = showing probable or definite left ventricular hypertrophy by Estes' criteria)
- thalach: The person's maximum heart rate achieved
- exang: Exercise induced angina (1 = yes; 0 = no)
- oldpeak: ST depression induced by exercise relative to rest ('ST' relates to positions on the ECG plot. See more here)
- slope: the slope of the peak exercise ST segment (Value 1: upsloping, Value 2: flat, Value 3: downsloping)
- ca: The number of major vessels (0-3)
- thal: A blood disorder called thalassemia (3 = normal; 6 = fixed defect; 7 = reversable defect)
- target: Heart disease (0 = no, 1 = yes)



---

下面是Peter整理的对应关系。本文中以当前的版本为标准：

| 当前 | 原文 |
| --- | --- |
| age | age |
| sex | sex |
| cp | chest_pain_type |
| trestbps | resting_blood_pressure |
| chol | cholesterol |
| fbs | fasting_blood_sugar |
| restecg | rest_ecg |
| thalach | max_heart_rate_achieved |
| exang | exercise_induced_angina |
| oldpeak | st_depression |
| slope | st_slope |
| ca | num_major_vessels |
| thal | thalassemia |
| target | target |



字段转化

**转化编码**

对部分字段进行一一的转化。以sex字段为例：将数据中的0变成female，1变成male

```
# 1、sex
```

```
df["sex"][df["sex"] == 0] = "female"
df["sex"][df["sex"] == 1] = "male"
```

In [6]:
```
# 1、sex

df["sex"][df["sex"] == 0] = "female"
df["sex"][df["sex"] == 1] = "male"
```

In [7]:
```
# 2、chest_pain_type = cp

df['cp'][df['cp'] == 1] = 'typical angina'
df['cp'][df['cp'] == 2] = 'atypical angina'
df['cp'][df['cp'] == 3] = 'non-anginal pain'
df['cp'][df['cp'] == 4] = 'asymptomatic'
```

In [8]:
```
# 3、fasting_blood_sugar = fbs

df['fbs'][df['fbs'] == 0] = 'lower than 120mg/ml'
df['fbs'][df['fbs'] == 1] = 'greater than 120mg/ml'
```



In [9]:
```
# 4、rest_ecg = restecg

df['restecg'][df['restecg'] == 0] = 'normal'
df['restecg'][df['restecg'] == 1] = 'ST-T wave abnormality'
df['restecg'][df['restecg'] == 2] = 'left ventricular hypertrophy'
```

In [10]:
```
# 5、exercise_induced_angina = exang

df['exang'][df['exang'] == 0] = 'no'
df['exang'][df['exang'] == 1] = 'yes'
```

In [11]:
```
# 6、  st_slope = slope

df['slope'][df['slope'] == 1] = 'upsloping'
df['slope'][df['slope'] == 2] = 'flat'
df['slope'][df['slope'] == 3] = 'downsloping'
```

In [12]:
```
# 7、thalassemia = thal

df['thal'][df['thal'] == 1] = 'normal'
df['thal'][df['thal'] == 2] = 'fixed defect'
df['thal'][df['thal'] == 3] = 'reversable defect'
```



**字段类型转化**

```
# 指定数据类型
df["sex"] = df["sex"].astype("object")
df["cp"] = df["cp"].astype("object")
df["fbs"] = df["fbs"].astype("object")
df["restecg"] = df["restecg"].astype("object")
df["exang"] = df["exang"].astype("object")
df["slope"] = df["slope"].astype("object")
df["thal"] = df["thal"].astype("object")
```

**生成哑变量**

```
# 生成哑变量
df = pd.get_dummies(df,drop_first=True)
df
```

Out[16]:

| | age | trestbps | chol | thalach | oldpeak | ca | target | sex_male | cp_atypical angina | cp_non-anginal pain | cp_typical angina | fbs_lower than 120mg/ml | restecg_left ventricular hypertrophy | reste |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 63 | 145 | 233 | 150 | 2.3 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | |
| 1 | 37 | 130 | 250 | 187 | 3.5 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | |
| 2 | 41 | 130 | 204 | 172 | 1.4 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | |
| 3 | 56 | 120 | 236 | 178 | 0.8 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | |
| 4 | 57 | 120 | 354 | 163 | 0.6 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 298 | 57 | 140 | 241 | 123 | 0.2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | |
| 299 | 45 | 110 | 264 | 132 | 1.2 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | |
| 300 | 68 | 144 | 193 | 141 | 3.4 | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | |
| 301 | 57 | 130 | 131 | 115 | 1.2 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | |
| 302 | 57 | 130 | 236 | 174 | 0.0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | |

303 rows × 20 columns



## 随机森林RandomForest

### 切分数据

```
# 生成特征变量数据集和因变量数据集
X = df.drop("target",1)
y = df["target"]

# 切分比例为8：2
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=10)

X_train
```

### 建模

```
rf = RandomForestClassifier(max_depth=5)
rf.fit(X_train, y_train)
```

### 3个重要属性

随机森林中3个重要的属性：

- **查看森林中树的状况**：estimators_
- 袋外估计准确率得分：oob_score_，必须是 `oob_score` 参数选择True的时候才可用
- **变量的重要性**：feature_importances_

### 决策树可视化

在这里我们选择的第二棵树的可视化过程：

```
# 查看第二棵树的状况
estimator = rf.estimators_[1]

# 全部属性
```

```
feature_names = [i for i in X_train.columns]
#print(feature_names)
```

```
# 指定数据类型
y_train_str = y_train.astype('str')
# 0-no 1-disease
y_train_str[y_train_str == '0'] = 'no disease'
y_train_str[y_train_str == '1'] = 'disease'
# 训练数据的取值
y_train_str = y_train_str.values
y_train_str[:5]
```

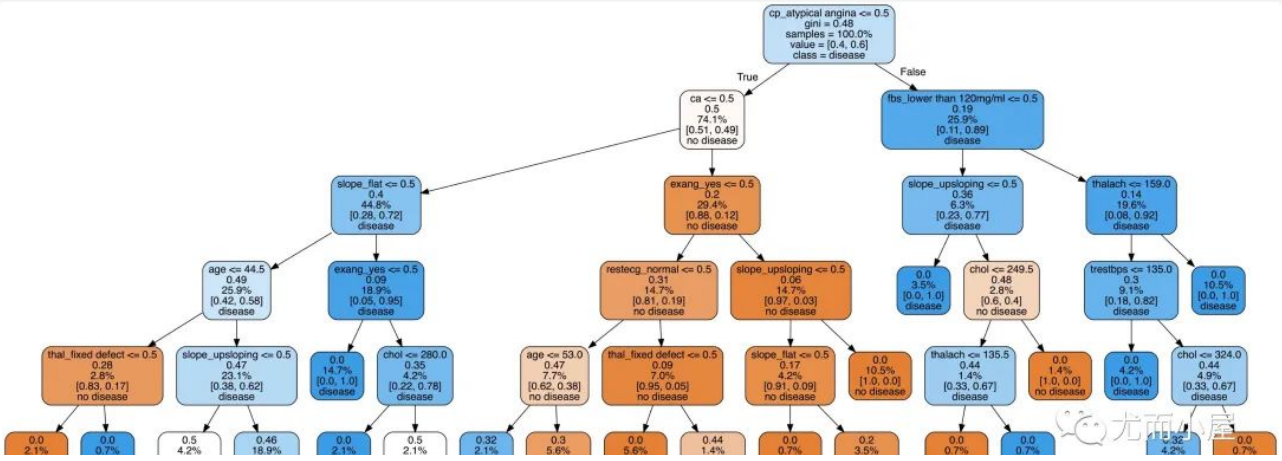
In [22]:
```
# 指定数据类型
y_train_str = y_train.astype('str')
# 0-no 1-disease
y_train_str[y_train_str == '0'] = 'no disease'
y_train_str[y_train_str == '1'] = 'disease'

y_train_str = y_train_str.values
y_train_str[:5]
```

Out[22]:
```
array(['no disease', 'disease', 'no disease', 'disease', 'disease',
       dtype=object)
```



绘图的具体代码为：

```
# 绘图显示

export_graphviz(
    estimator,   # 传入第二颗树
    out_file='tree.dot',   # 导出文件名
    feature_names = feature_names,  # 属性名
    class_names = y_train_str,  # 最终的分类数据
    rounded = True,
    proportion = True,
    label='root',
    precision = 2,
    filled = True
)

from subprocess import call
call(['dot', '-Tpng', 'tree.dot', '-o', 'tree.png', '-Gdpi=600'])

from IPython.display import Image
Image(filename = 'tree.png')
```




决策树的可视化过程能够让我们看到具体的分类过程，但是并不能解决哪些特征或者属性比较重要。后面会对部分属性的特征重要性进行探索

**模型得分验证**

关于混淆矩阵和使用特异性（specificity）以及灵敏度（sensitivity）这两个指标来描述分类器的性能：

```
# 模型预测
y_predict = rf.predict(X_test)
y_pred_quant = rf.predict_proba(X_test)[:,1]
y_pred_bin = rf.predict(X_test)

# 混淆矩阵
confusion_matrix = confusion_matrix(y_test,y_pred_bin)
confusion_matrix

# 计算sensitivity and specificity
total=sum(sum(confusion_matrix))
sensitivity = confusion_matrix[0,0]/(confusion_matrix[0,0]+confusion_matrix[1,0])
specificity = confusion_matrix[1,1]/(confusion_matrix[1,1]+confusion_matrix[0,1])
```

```
In [24]:
# 模型预测
y_predict = rf.predict(X_test)

y_pred_quant = rf.predict_proba(X_test)[:,1]

y_pred_bin = rf.predict(X_test)
```

```
In [25]:
# 混淆矩阵
confusion_matrix = confusion_matrix(y_test,y_pred_bin)
confusion_matrix
```

```
Out[25]: array([[29,  6],
                [ 4, 22]])
```

```
In [26]:
# 计算sensitivity and specificity

total=sum(sum(confusion_matrix))

sensitivity = confusion_matrix[0,0]/(confusion_matrix[0,0]+confusion_matrix[1,0])
print('Sensitivity : ', sensitivity )
```

```
Sensitivity :  0.8787878787878788
```

```
In [27]:
specificity = confusion_matrix[1,1]/(confusion_matrix[1,1]+confusion_matrix[0,1])
print('Specificity : ', specificity)
```

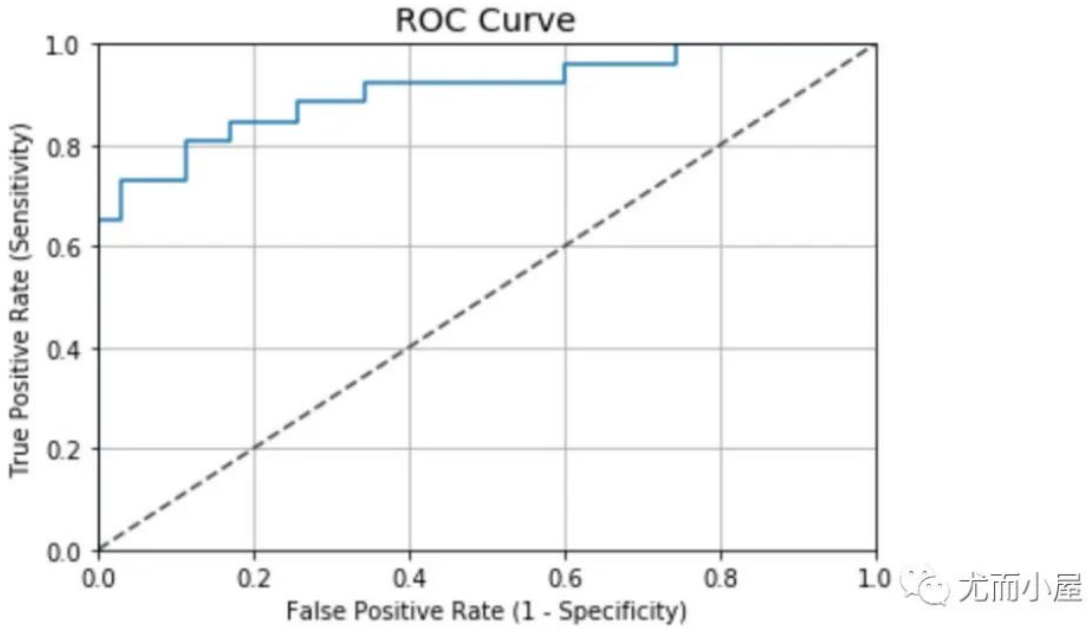


```
Specificity :  0.7857142857142857
```

**绘制ROC曲线**

```
fpr, tpr, thresholds = roc_curve(y_test, y_pred_quant)

fig, ax = plt.subplots()
ax.plot(fpr, tpr)

ax.plot([0,1],[0,1],
        transform = ax.transAxes,
        ls = "--",
        c = ".3"
       )

plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
```

```
plt.rcParams['font.size'] = 12

# 标题
plt.title('ROC Curve')
# 两个轴的名称
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
# 网格线
plt.grid(True)
```




本案例中的ROC曲线值：

```
auc(fpr, tpr)
# 结果
0.9076923076923078
```

根据一般ROC曲线的评价标准，案例的表现结果还是不错的：

- 0.90 - 1.00 = excellent
- 0.80 - 0.90 = good
- 0.70 - 0.80 = fair
- 0.60 - 0.70 = poor
- 0.50 - 0.60 = fail

**补充知识点：分类器的评价指标**

考虑一个二分类的情况，类别为1和0，我们将1和0分别作为正类（positive）和负类（negative），根据实际的结果和预测的结果，则最终的结果有4种，表格如下：

| | | 预测 | | |
|---|---|---|---|---|
| | | 1 | 0 | 合计 |
| 实际 | 1 | True Positive（TP） | False Negative（FN） | Actual Positive(TP+FN) |
| | 0 | False Positive（FP） | True Negative(TN) | Actual Negative(FP+TN) |
| 合计 | | Predicted Positive(TP+FP) | Predicted Negative(FN+TN) | TP+FP+FN+TN |

常见的评价指标：

**1、ACC**：classification accuracy，描述分类器的分类准确率

计算公式为：ACC=(TP+TN)/(TP+FP+FN+TN)

**2、BER**：balanced error rate 计算公式为：BER=1/2*(FPR+FN/(FN+TP))

**3、TPR**：true positive rate，描述识别出的所有正例占所有正例的比例 计算公式为：TPR=TP/ (TP+ FN)

**4、FPR**：false positive rate，描述将负例识别为正例的情况占所有负例的比例 计算公式为：FPR= FP / (FP + TN)

**5、TNR**：true negative rate，描述识别出的负例占所有负例的比例 计算公式为：TNR= TN / (FP + TN)

**6、PPV**：Positive predictive value计算公式为：PPV=TP / (TP + FP)

**7、NPV**：Negative predictive value计算公式：NPV=TN / (FN + TN)

其中**TPR**即为敏感度（sensitivity），**TNR**即为特异度（specificity）。

$$Sensitivity = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

$$Specificity = \frac{True\ Negatives}{True\ Negatives + False\ Positives}$$



来自维基百科的经典图形：

| | | Patients with bowel cancer (as confirmed on endoscopy) | | |
|---|---|---|---|---|
| | | Condition positive | Condition negative | |
| Fecal occult blood screen test outcome | Test outcome positive | **True positive** (TP) = 20 | **False positive** (FP) = 180 | Positive predictive value = TP / (TP + FP) = 20 / (20 + 180) = **10%** |
| | Test outcome negative | **False negative** (FN) = 10 | **True negative** (TN) = 1820 | Negative predictive value = TN / (FN + TN) = 1820 / (10 + 1820) ≈ **99.5%** |
| | | Sensitivity = TP / (TP + FN) = 20 / (20 + 10) ≈ **67%** | Specificity = TN / (FP + TN) = 1820 / (180 + 1820) = **91%** | |



可解释性

**排列重要性-Permutation Importance**

下面的内容是关于机器学习模型的结果可解释性。首先考察的是每个变量对模型的重要性。重点考量的排列重要性Permutation Importance：

```
In [30]: perm = PermutationImportance(rf,random_state=10).fit(X_test,y_test)
         perm
```

```
Out[30]: PermutationImportance(estimator=RandomForestClassifier(max_depth=5),
```

```
                                    random_state=10)
```

```
In [31]: eli5.show_weights(perm,feature_names=X_test.columns.tolist())
```

Out[31]:

| Weight | Feature |
|---|---|
| 0.0262 ± 0.0262 | thalach |
| 0.0197 ± 0.0245 | oldpeak |
| 0.0164 ± 0.0000 | cp_non-anginal pain |
| 0.0131 ± 0.0382 | thal_fixed defect |
| 0.0131 ± 0.0131 | cp_atypical angina |
| 0.0131 ± 0.0131 | chol |
| 0.0098 ± 0.0161 | sex_male |
| 0.0066 ± 0.0445 | exang_yes |
| 0.0066 ± 0.0161 | age |
| 0.0066 ± 0.0161 | slope_upsloping |
| 0.0033 ± 0.0564 | ca |
| 0.0000 ± 0.0293 | thal_reversable defect |
| 0 ± 0.0000 | thal_normal |
| 0 ± 0.0000 | fbs_lower than 120mg/ml |
| 0 ± 0.0000 | restecg_left ventricular hypertrophy |
| 0 ± 0.0000 | restecg_normal |
| 0 ± 0.0000 | trestbps |
| 0 ± 0.0000 | cp_typical angina |
| -0.0033 ± 0.0245 | slope_flat |



部分依赖图（ Partial dependence plots ， PDP）

**一维PDP**

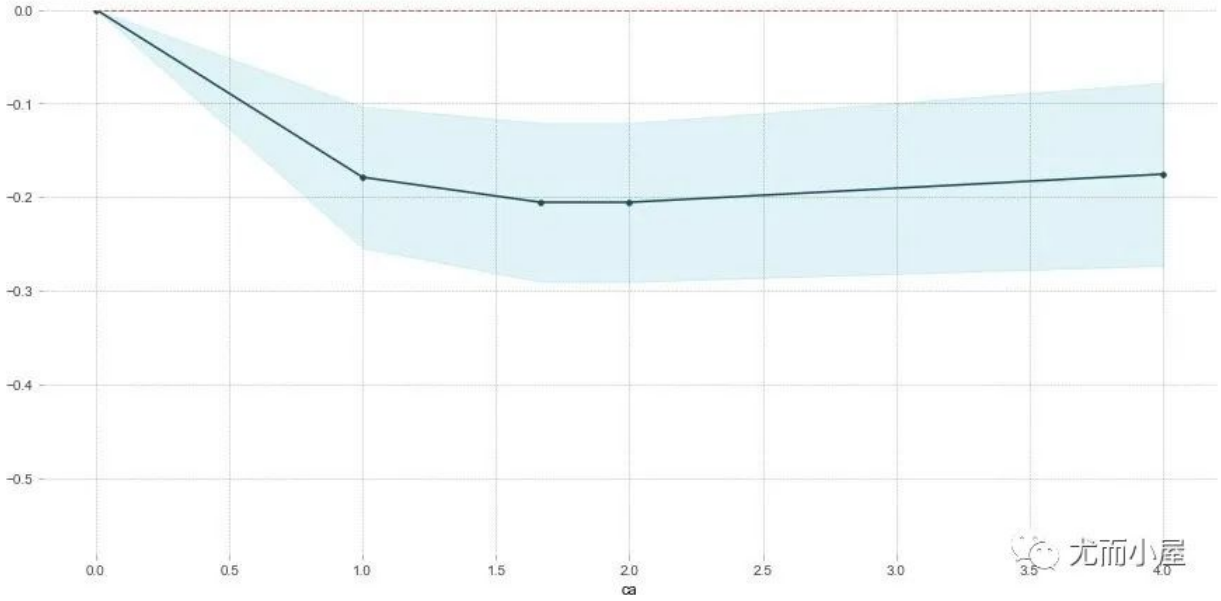
> Partial Dependence就是用来解释某个特征和目标值y的关系的，一般是通过画出Partial Dependence Plot(PDP)来体现。也就是说PDP在X1的值，就是把训练集中第一个变量换成X1之后，原模型预测出来的平均值。

**重点：查看单个特征和目标值的关系**

**字段ca**

```
base_features = df.columns.values.tolist()
base_features.remove("target")

feat_name = 'ca'  # ca-num_major_vessels 原文
pdp_dist = pdp.pdp_isolate(
    model=rf,  # 模型
    dataset=X_test,  # 测试集
    model_features=base_features,  # 特征变量：除去目标值
    feature=feat_name  # 指定单个字段
)

pdp.pdp_plot(pdp_dist, feat_name)  # 传入两个参数
plt.show()
```

通过下面的图形我们观察到：当ca字段增加的时候，患病的几率在下降。ca字段的含义是血管数量（num_major_vessels），也就是说当血管数量增加的时候，患病率随之降低



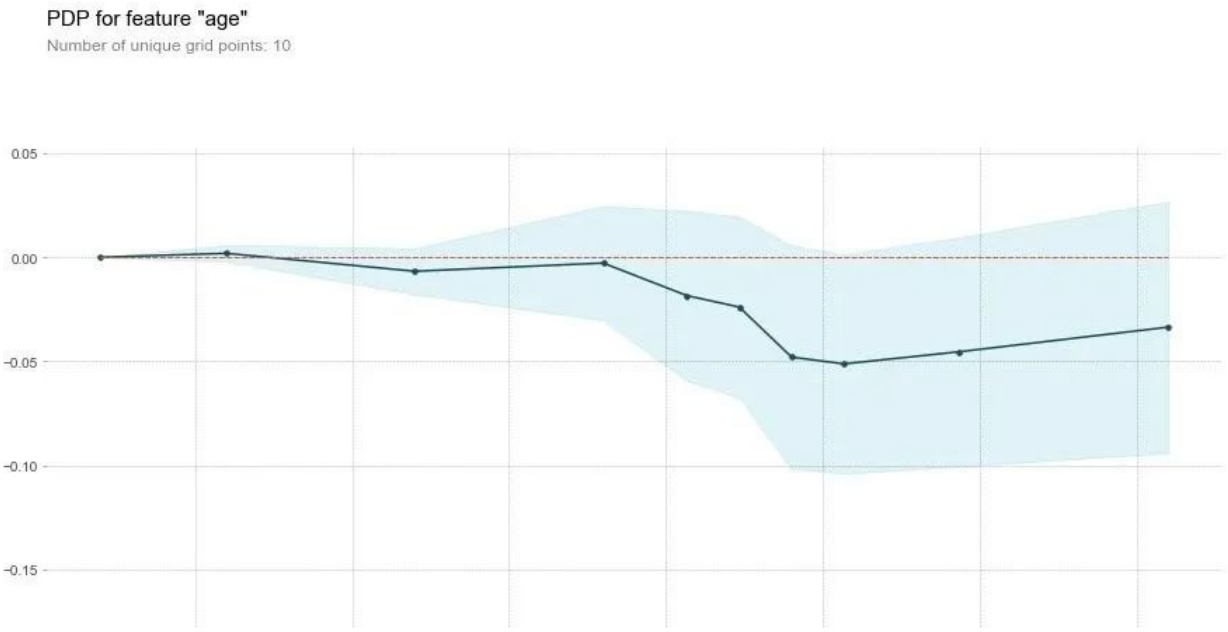

**字段age**

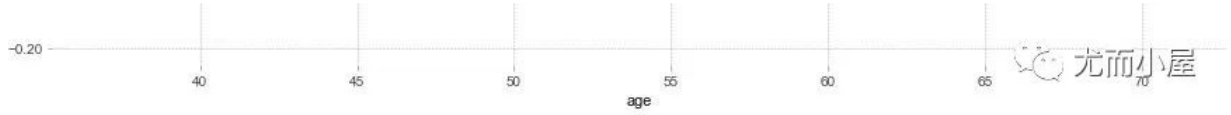
```
feat_name = 'age'

pdp_dist = pdp.pdp_isolate(
    model=rf,
    dataset=X_test,
    model_features=base_features,
    feature=feat_name)

pdp.pdp_plot(pdp_dist, feat_name)
plt.show()
```

关于年龄字段age，原文的描述：

> That's a bit odd. The higher the age, the lower the chance of heart disease? Althought the blue confidence regions show that this might not be true (the red baseline is within the blue zone).

翻译：这有点奇怪。年龄越大，患心脏病的几率越低？尽管蓝色置信区间表明这可能不是真的（红色基线在蓝色区域内）

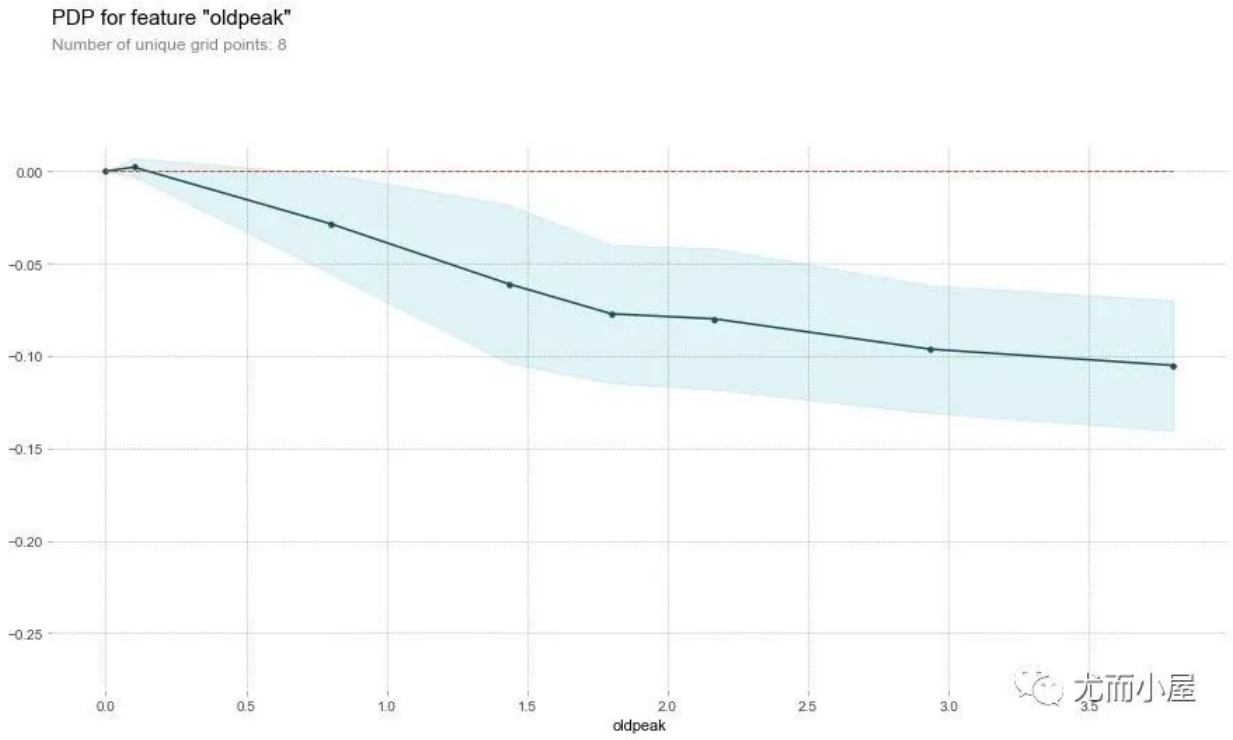





**字段oldpeak**

```
feat_name = 'oldpeak'

pdp_dist = pdp.pdp_isolate(
    model=rf,
    dataset=X_test,
    model_features=base_features,
    feature=feat_name)

pdp.pdp_plot(pdp_dist, feat_name)
plt.show()
```

oldpeak字段同样表明：取值越大，患病几率越低。




这个变量称之为“相对休息运动引起的ST压低值”。正常的状态下，**该值越高，患病几率越高**。但是上面的图像却显示了相反的结果。

作者推断：造成这个结果的原因除了the depression amount，可能还和slope type有关系。原文摘录如下，于是作者绘制了2D-PDP图形

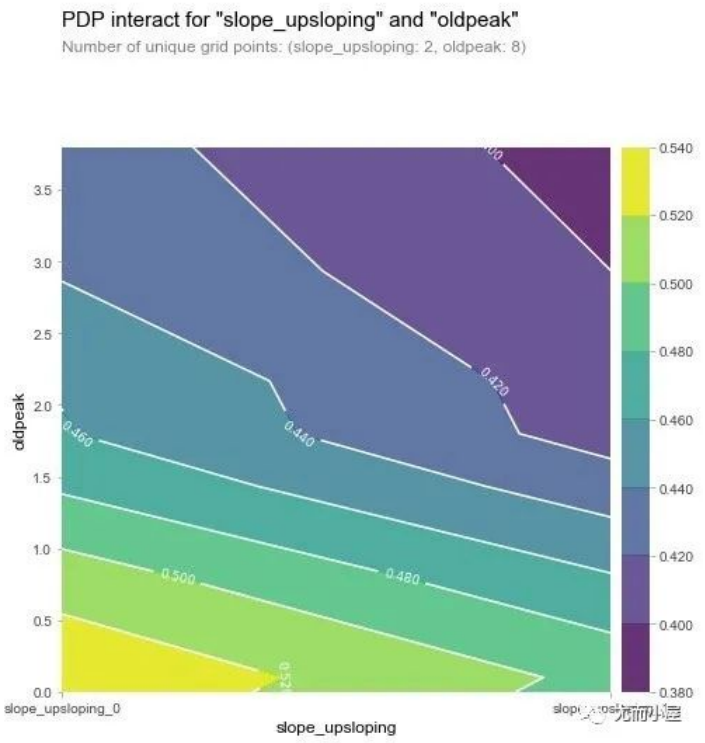
> Perhaps it's not just the depression amount that's important, but the interaction with the slope type? Let's check with a 2D PDP

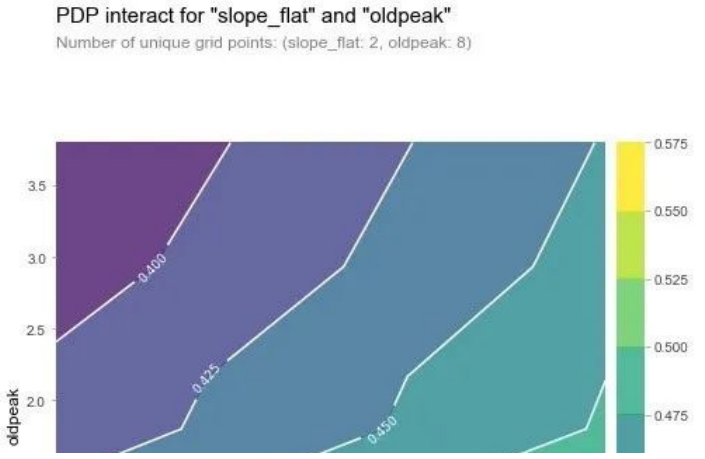
**2D-PDP图**

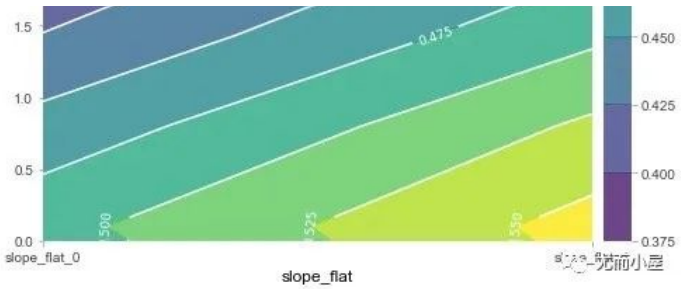
查看的是 slope_upsloping 、slope_flat和 oldpeak的关系：

```
inter1  =  pdp.pdp_interact(
    model=rf,  # 模型
    dataset=X_test,  # 特征数据集
    model_features=base_features,  # 特征
```

```
    features=['slope_upsloping', 'oldpeak'])

pdp.pdp_interact_plot(
    pdp_interact_out=inter1,
    feature_names=['slope_upsloping', 'oldpeak'],
    plot_type='contour')
plt.show()

## ------------

inter1  =  pdp.pdp_interact(
    model=rf,
    dataset=X_test,
    model_features=base_features,
    features=['slope_flat', 'oldpeak']
)

pdp.pdp_interact_plot(
    pdp_interact_out=inter1,
    feature_names=['slope_flat', 'oldpeak'],
    plot_type='contour')
plt.show()
```






从两张图形中我们可以观察到：在oldpeak取值较低的时候，患病几率都比较高（黄色），这是一个奇怪的现象。于是作者进行了如下的SHAP可视化探索：针对单个变量进行分析。

## SHAP可视化

关于SHAP的介绍可以参考文章：https://zhuanlan.zhihu.com/p/83412330 和 https://blog.csdn.net/sinat_26917383/article/details/115400327

**SHAP**是Python开发的一个"模型解释"包，可以解释任何机器学习模型的输出。下面SHAP使用的部分功能：

### Explainer

在SHAP中进行模型解释之前需要先创建一个**explainer**，SHAP支持很多类型的explainer，例如deep, gradient, kernel, linear, tree, sampling)。在这个案例我们以tree为例：

```
# 传入随机森林模型rf
explainer = shap.TreeExplainer(rf)
# 在explainer中传入特征值的数据，计算shap值
shap_values = explainer.shap_values(X_test)
shap_values
```

```
In [38]: explainer = shap.TreeExplainer(rf)
         shap_values = explainer.shap_values(X_test)

         shap_values
```

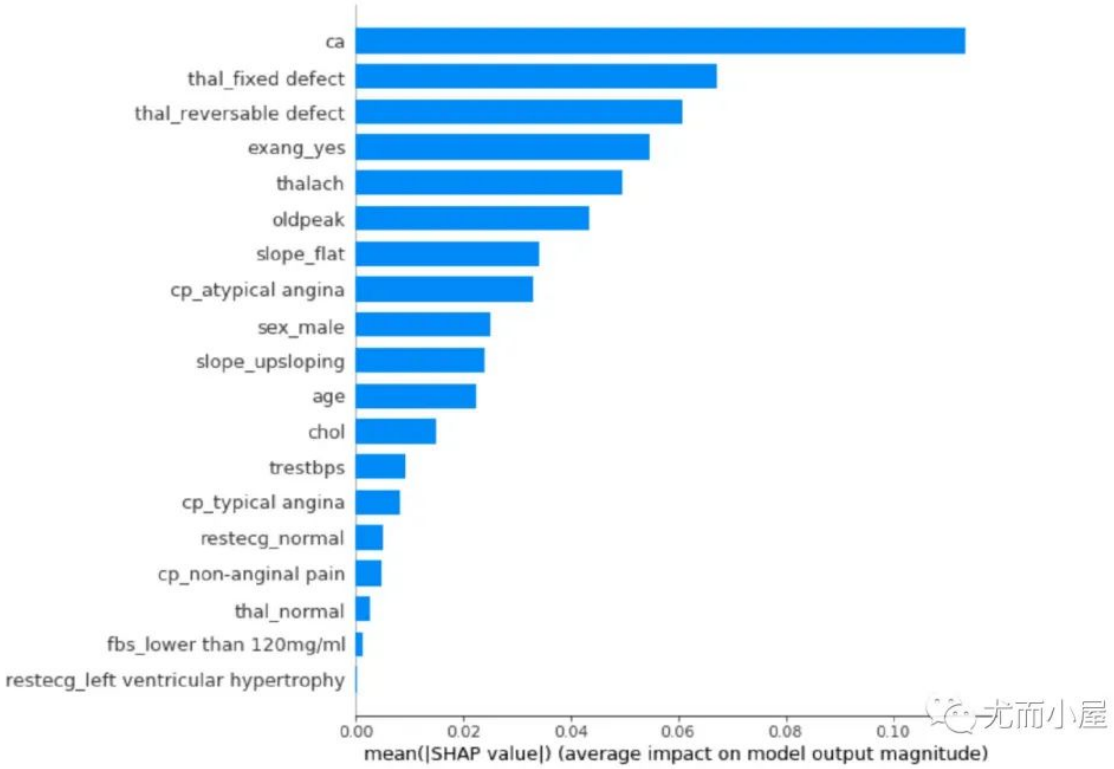
```
Out[38]: [array([[ 0.00100664,  0.00500851,  0.02376482, ...,  0.07657449,
                  -0.00142552,  0.07264502],
                 [ 0.02201797, -0.01872521,  0.00134986, ...,  0.06978621,
                  -0.0028233 ,  0.06332491],
                 [ 0.01043852,  0.00578357,  0.02127174, ...,  0.08591864,
                  -0.00367775,  0.08710552],
                 ...,
                 [-0.01366667,  0.00491541,  0.00680899, ...,  0.06954597,
                  -0.00167924,  0.06963891],
                 [ 0.00979606,  0.00706704, -0.01920246, ...,  0.07341399,
                  -0.00134914,  0.07500611],
                 [-0.02230723,  0.00753204, -0.02980397, ...,  0.06974011,
                  -0.00099324,  0.08818901]]),
          array([[-0.00100664, -0.00500851, -0.02376482, ..., -0.07657449,
                   0.00142552, -0.07264502],
                 [-0.02201797,  0.01872521, -0.00134986, ..., -0.06978621,
                   0.0028233 , -0.06332491],
                 [-0.01043852, -0.00578357, -0.02127174, ..., -0.08591864,
                   0.00367775, -0.08710552],
                 ...,
                 [ 0.01366667, -0.00491541, -0.00680899, ..., -0.06954597,
                   0.00167924, -0.06963891],
                 [-0.00979606, -0.00706704,  0.01920246, ..., -0.07341399,
                   0.00134914, -0.07500611],
                 [ 0.02230723, -0.00753204,  0.02980397, ..., -0.06974011,
                   0.00099324, -0.08818901]])]
```

image-20220131173928357

**Feature Importance**

取每个特征**SHAP值的绝对值的平均值**作为该特征的重要性，得到一个标准的条形图(multi-class则生成堆叠的条形图：

```
In [39]: shap.summary_plot(shap_values[1], X_test, plot_type="bar")
```

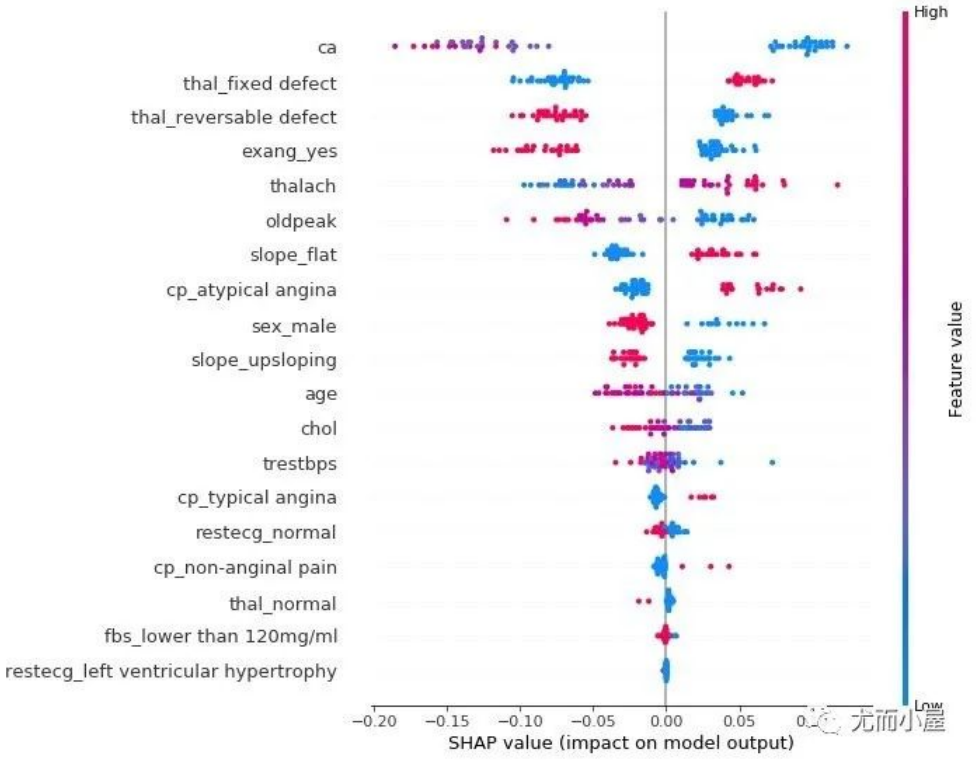



结论：能够直观地观察到ca字段的SHAP值最高

**summary_plot**

`summary plot` 为每个样本绘制其每个特征的SHAP值，这可以更好地理解整体模式，并允许发现预测异常值。

- 每一行代表一个特征，横坐标为SHAP值
- 一个点代表一个样本，颜色表示特征值的高低(红色高，蓝色低)



**个体差异**

查看单个病人的不同特征属性对其结果的影响，原文描述为：

> Next, let's pick out individual patients and see how the different variables are affecting their outcomes

```
def heart_disease_risk_factors(model, patient):

    explainer = shap.TreeExplainer(model)  # 建立explainer
    shap_values = explainer.shap_values(patient)  # 计算shape值
    shap.initjs()
    return shap.force_plot(
      explainer.expected_value[1],
      shap_values[1],
      patient)
```


从两个病人的结果中显示：

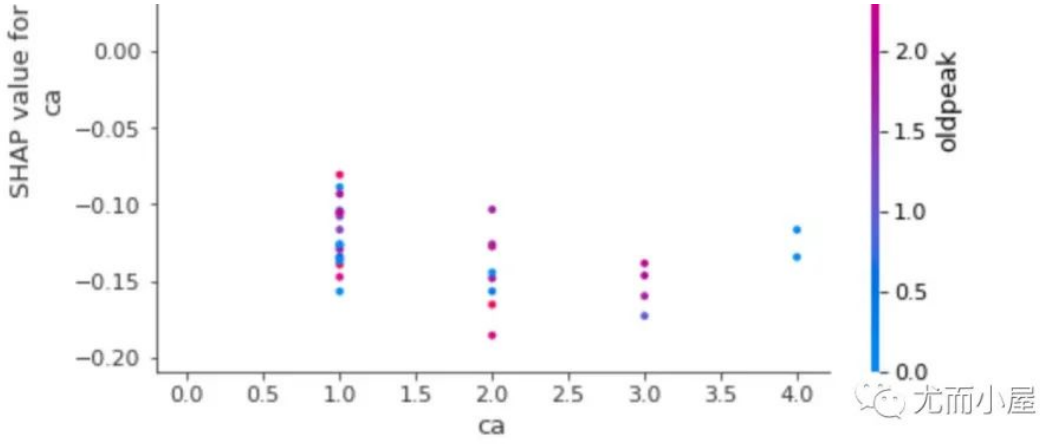
- P1：预测准确率高达29%（baseline是57%），更多的因素集中在ca、thal_fixed_defect、oldpeak等蓝色部分。
- P3：预测准确率高达82%，更多的影响因素在sel_male=0，thalach=143等

通过对比不同的患者，我们是可以观察到不同病人之间的预测率和主要影响因素。

**dependence_plot**

为了理解单个feature如何影响模型的输出，我们可以将该feature的SHAP值与数据集中所有样本的feature值进行比较：

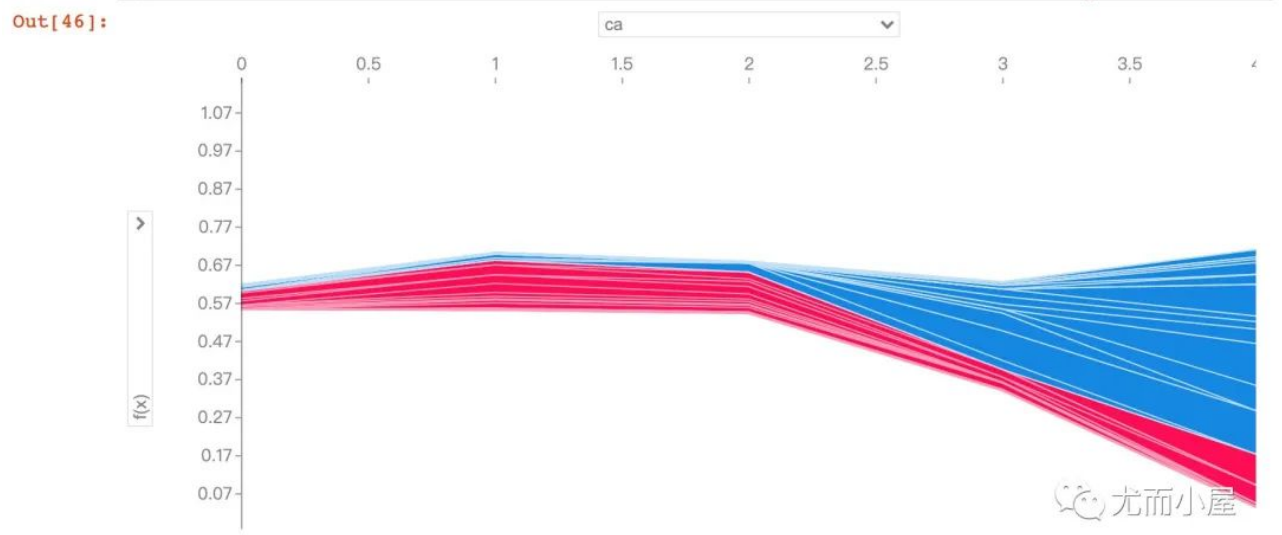
```
In [45]: ax2 = fig.add_subplot(224)

shap.dependence_plot('ca',
                     shap_values[1],
                     X_test,
                     interaction_index="oldpeak")
```




**多样本可视化探索**

下面的图形是针对多患者的预测和影响因素的探索。在jupyter notebook的交互式作用下，能够观察不同的特征属性对前50个患者的影响：

```
shap_values = explainer.shap_values(X_train.iloc[:50])
shap.force_plot(explainer.expected_value[1],
                shap_values[1],
                X_test.iloc[:50])
```

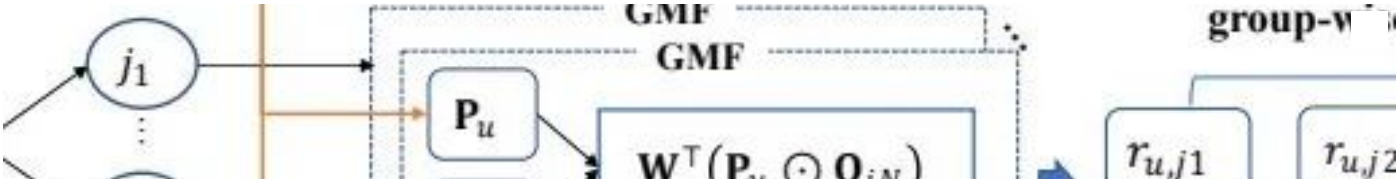





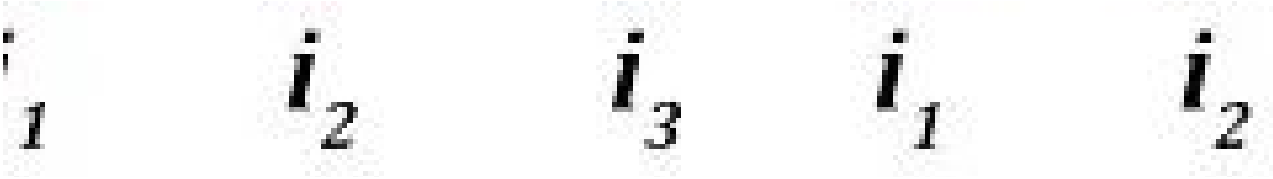
交流群：点击“联系作者”--备注“研究方向-公司或学校”

欢迎干货投稿|论文宣传|合作交流

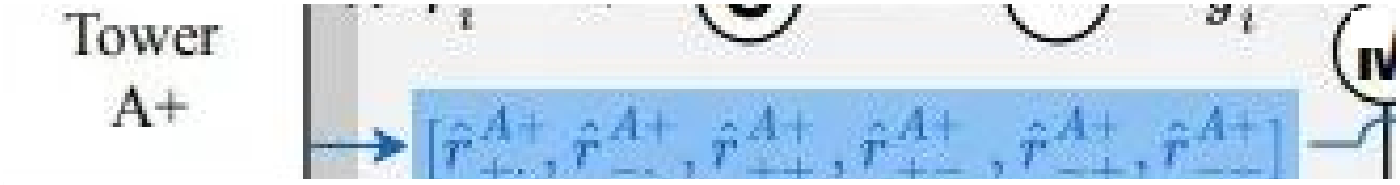
**往期推荐**




WWW'22 | GDNS：基于增益的动态负采样方法用于推荐系统




WWW'22「华为」CPR Loss：交叉成对排序损失对推荐系统纠偏




AAAI'22「腾讯」多任务推荐系统中的跨任务知识蒸馏

长按关注，更多精彩

点个 在看 你最好看

喜欢此内容的人还喜欢

GCRNN：用于化合物-蛋白质相互作用预测的图卷积递归神经网络

ScienceAI

BIB｜基于深度学习的T细胞免疫原性肽的预测和生成

生信AI

谷歌AI一次注释了10%的已知蛋白质序列，超过人类十年研究成果

ScienceAI