SHAP值原理

笔记本：	机器学习		
创建时间：	2019/5/22 10:35	**更新时间：**	2019/5/23 14:38
作者：	lizhigu1996@163.com		

该笔记主要整理了SHAP（Shapley Additive exPlanations）框架（也就是shap库）开发者Lundberg的两篇论文A Unified Approach to Interpreting Model Predictions，Consistent Individualized Feature Attribution for Tree Ensembles，以及Christoph Molnar发布的书籍Interpretable Machine Learning的5.8部分，有兴趣的话可完整阅读该书。

1. 当前特征归因方法的不一致性

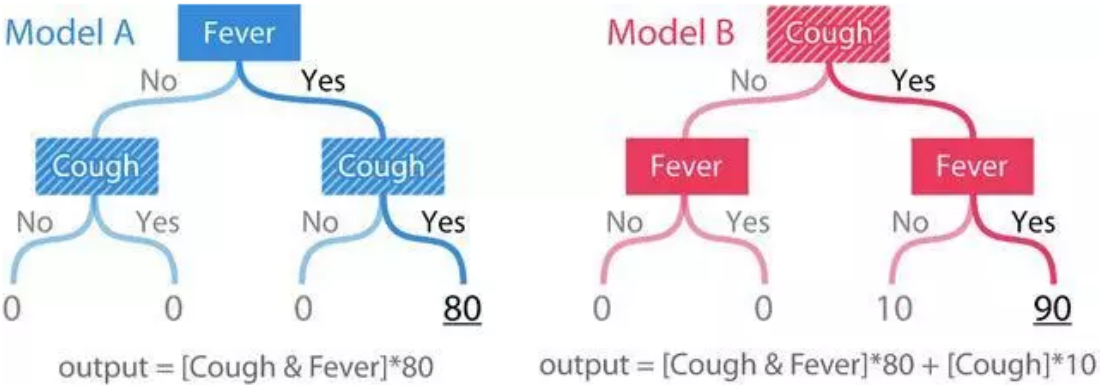
首先先定义**一致性：每当我们更改模型以使其更依赖于某个特征时，该特征的归因重要性不应该降低。**如果一致性不成立，意味着当一个模型被更改为某个特征对模型输出的影响更大时，反而会降低该特征的重要性，那么我们不能比较任意两个模型之间的归因重要性，因为具有较高分配归因的特征并不意味着模型实际上更依赖该特征。

特征归因方法可以分全局和个性化（针对个体），其中全局特征重要度是为整个数据集计算的，该类特征归因方法通常都用特征重要度表示，主要有三种方式（特征重要度的差异可查看**各模型特征重要度.ipynb**）：

1. 增益（Gain）。给定特征的所有分裂所贡献的损失或不纯度的总减少量，增益在特征选择方向上被广泛应用。
2. 分裂数（Split Count）。在所有树中一个特征被用做分裂节点的次数。
3. 置换。随机置换测试集中一个特征的值，然后观察模型误差的变化，如果一个特征的值很重要，那么遍历它会导致模型的错误大量增加。

上面三种都是基于整个数据集去计算特征重要度的，但对于树来说，计算单个预测的特征重要值的个性化方法却较少，虽然与模型无关的个性化解释方法（比如LIME）可以应用于树，但它们明显比树特定的方法慢，并且具有抽样变异性，目前我们所知的树特有的个性化解释方法只有Sabbas，该方法与经典的全局的增益方法类似，但它不是测量损失的减少，而是测量模型预期输出的变化，通过比较模型在树根处输出的期望值与子树在子节点处输出的期望值，以及当前输入的决策路径，然后将这些期望之间的差异归因于在根节点上分离的特性，通过递归地重复这个过程，最后在决策路径上的特征之间分配预期模型输出和当前输出之间的差异。

下面举两个模型的例子对归因方法的一致性进行比较，假设模型的输出是基于人的症状的风险评分，对于二元特征发烧(Fever)和咳嗽(Cough)，模型A只是一个简单的"和"函数，模型B是相同的函数，但是当为咳嗽时预测值会增加（加10分），使得模型更依赖于咳嗽，这时因咳嗽更重要，导致在模型B中咳嗽先分裂。



比较A、B模型在下面六种归因方法上的差别

1. Tree SHAP，本文提出的一种新的个性化方法。（个性化特征归因方法，为单个预测计算）

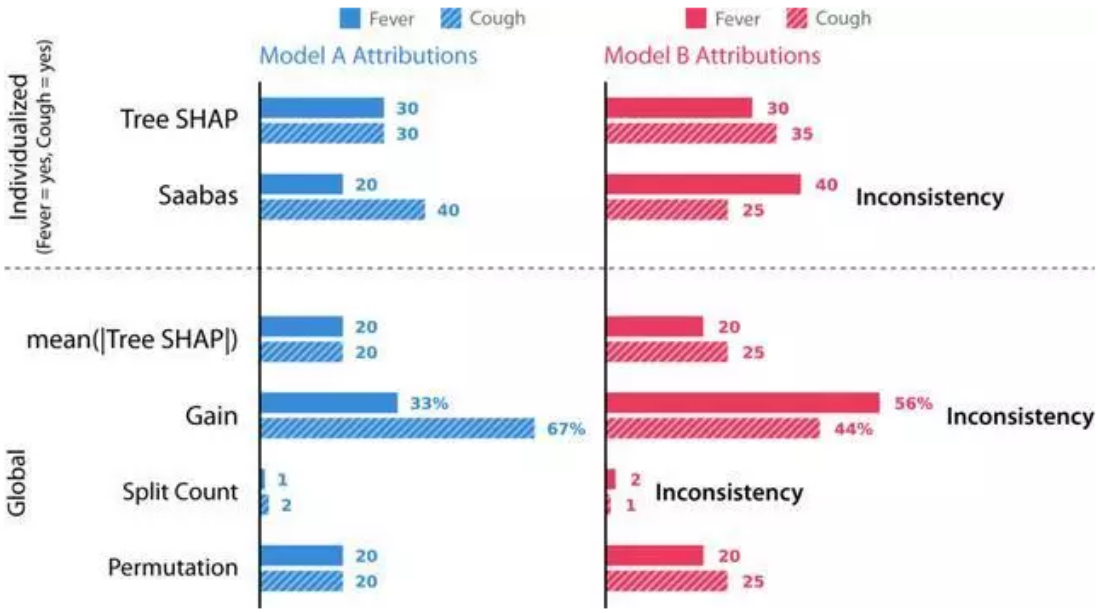
2. Saabas，个性化的启发式特征归因方法。（个性化特征归因方法，为单个预测计算）

3. mean（|Tree SHAP|），基于个性化Tree SHAP归因的平均幅度的全局归因方法（全局特征归因方法，为整个数据集计算，实际为所有样本的Tree SHAP值按照特征计算均值）

4. 增益（全局特征归因方法，为整个数据集计算）

5. 分裂数（全局特征归因方法，为整个数据集计算）

6. 置换（全局特征归因方法，为整个数据集计算）



个性化特征归因方法：Tree SHAP、Sabbas，只有SHAP值能够保证反映特征的重要性，而Saabas值可能会给出错误的结果，比如模型B中认为更大的原因是发烧，而不是咳嗽，这是不一致的表现。

全局特征归因方法：mean（|Tree SHAP|）、增益、分裂数和特征置换，只有mean（|Tree SHAP|）和置换认为模型B咳嗽比发烧更重要，这意味着在一致性上增益和分裂数不是全局特性重要性的可靠度量。

所以gain、split count和Saabas方法中的特征重要度都不一致（使B模型更加依赖咳嗽时，却认为发烧更重要），这意味着模型改变为更多地依赖于给定的特性时，分配给该特征的重要性却降低了。通常我们期望树根附近的特征比在叶子附近分裂的特征更重要（因为树是贪婪地构造的），然而增益方法偏向于更重视较低的分裂，这种偏差会导致不一致，当咳嗽变得更加重要时（因此在根部分裂），其归因重要性实际上下降。个性化的Saabas方法在我们下降树时计算预测的差异，因此它也会受

到与树中较低分割相同的偏差，随着树木越来越深，这种偏差只会增长。 相比之下，Tree SHAP方法在数学上等效于平均所有可能的特征排序的预测差异，而不仅仅是它们在树中的位置指定的排序。

所以在我们考虑的方法中，只有SHAP值和置换的方法是具有一致性的，而其中又只有SHAP值是个性化的，所以**SHAP值是唯一一致的个性化特征归因方法。**

2. Shapley值

在介绍SHAP值之前先介绍Shapley值，SHAP值的主要思想就是Shapley值，Shapley值是一种来自合作博弈论（coalitional game theory）的方法，由Shapley（1953）创造的Shapley值是一种根据玩家对总支出的贡献来为玩家分配支出的方法，玩家在联盟中合作并从这种合作中获得一定的利润，其中“总支出”就是数据集的单个实例的模型预测值，“增益”是该实例的实际预测减去所有实例的平均预测，“玩家”是实例的各特征值。

2.1 例子说明

假设以下情形：已经训练了一个机器学习模型来预测公寓价格，分别有park、size、floor、car四个特征。某个面积为50平方米（size=50）、位于二楼（floor=2nd）、附近有一个公园（park=nearby）、禁止猫咪（cat=banned）的公寓，它预测价格为300,000欧元，你需要解释这个预测，即每个特征是如何促进预测的？当所有公寓的平均预测为310,000欧元时，与平均预测相比，每个特征值对预测的贡献是多少？

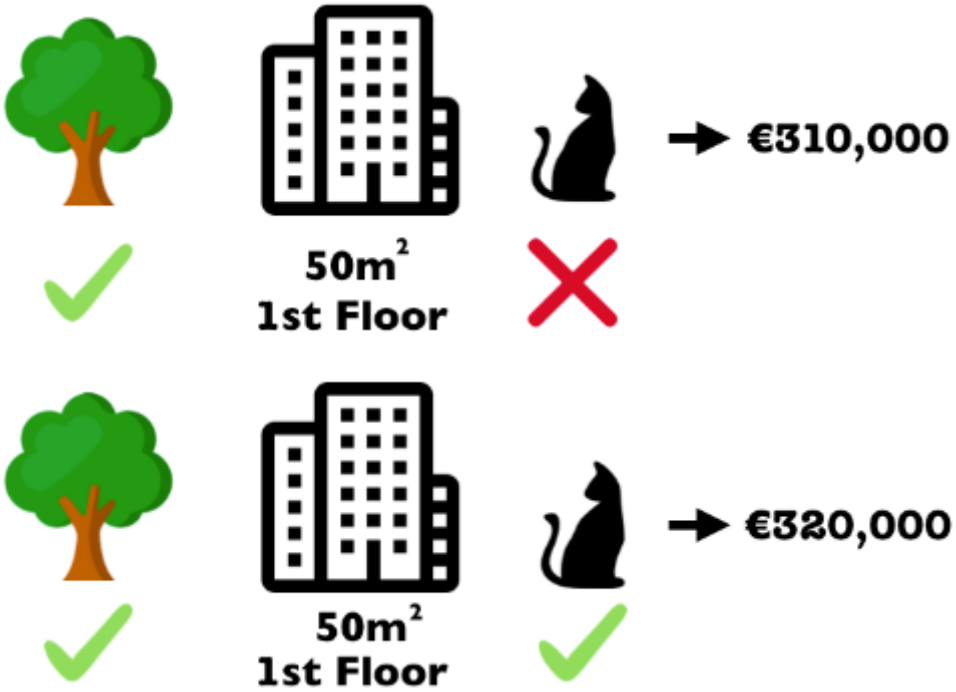
线性回归模型的答案很简单，每个特征的贡献是特征的权重乘以特征值，但这仅适用于线性模型，对于更复杂的模型我们需要不同的解决方案，例如LIME建议用局部模型来估计影响，另一种解决方案来自合作博弈论。

在我们的公寓示例中，park=nearby，cat=banned，size=50，floor=2nd的特征值共同实现了300,000欧元的预测。我们的目标是解释实际预测（300,000欧元）和平均预测（310,000欧元）之间的差异：-10,000欧元。答案可能是：park=nearby贡献了30,000欧元，size=50贡献了10,000欧元，floor=2nd贡献了0欧元，cat=banned贡献了-50,000欧元，这些贡献加起来为-10,000欧元，即最终预测减去平均预测的公寓价格。

那实际上我们应该如何计算一个特征的Shapley值？

Shapley值是所有可能联盟中特征值的平均边际贡献。在下图中，我们估计了cat=banned特征值被添加到park=nearby和size=50的联盟后的贡献。第一步，我们从数据中随机抽取另一个公寓，并使用该公寓自己的floor特征值1st，模拟出park=nearby，cat=banned和size=50的联盟，然后我们用这个组合预测公寓的价格为310,000欧元。在第二步中，我们从联盟中删除cat=banned，然后用该公寓的cat特征值（allowed或banned）替代，我们用这个组合预测公寓的价格为320,000欧元。

可以看到，基于我们随机抽取的公寓，我们预测park=nearby，cat=banned和size=50的联盟的公寓价格为310,000欧元，预测park=nearby和size=50的联盟的公寓价格为320,000欧元，那cat=banned的贡献是310,000欧元 - 320,000欧元=-10,000欧元，由于该公寓充当cat和floor特征值的“贡献者（donor）”，所以这个估计值取决于随机抽取的公寓的值，如果我们重复这个抽样步骤并取贡献的平均，我们将得到更好的估计。



上面只介绍了park=nearby和size=50联盟的贡献，而Shapley值时所有可能联盟的所有边际贡献的平均值，所以我们应该对所有可能的联盟重复这个计算。计算时间随着特征的数量和每个联盟中抽样的实例数量呈指数增长。下面是计算目标公寓的cat=banned的Shapley值的所有特征值联盟：

* 没有特征值
* park=nearby
* size=50
* floor=2nd
* park=nearby 和 size=50
* park=nearby 和 floor=2nd
* size=50 和floor=2nd
* park=nearby 和size=50 和floor=2nd.

对于这些联盟中的每一个，我们计算具有和不具有cat=banned特征值的预测公寓价格，并计算差值来获得边际贡献，Shapley值是边际贡献的（加权）平均值，我们用来自公寓数据集的随机特征值替换不在联盟中的特征的特征值，以从机器学习模型获得预测。如果我们估计所有特征值的Shapley值，我们将得到特征值中预测的完整分布（减去平均值）。

2.2 公式说明

我们感兴趣的是每个特征如何影响预测，在线性模型中很容易计算出各个特征的贡献，以下是一个数据实例的线性模型预测：

$$\hat{f}(x) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$$

其中 x 是我们想要计算贡献的实例，每个x_j（j = 1，...，p）是实例的特征值，β_j是与特征 j 对应的权重。

第 j 个特征对预测$\hat{f}(x)$的贡献ϕ_j是：

$$\phi_j(\hat{f}) = \beta_j x_j - E(\beta_j X_j) = \beta_j x_j - \beta_j E(X_j)$$

其中$E(\beta_j X_j)$是特征 j 的平均影响估计，贡献是特征影响减去平均影响之间的差异，现在我们知道每个特征对预测的贡献程度了。如果我们将一个实例的所有特征贡献相加，则结果如下：

$$\begin{aligned}\sum_{j=1}^{p}\phi_j(\hat{f}) &= \sum_{j=1}^{p}(\beta_j x_j - E(\beta_j X_j)) \\ &= (\beta_0 + \sum_{j=1}^{p}\beta_j x_j) - (\beta_0 + \sum_{j=1}^{p}E(\beta_j X_j)) \\ &= \hat{f}(x) - E(\hat{f}(X))\end{aligned}$$

也就是说数据点 x 的贡献之和等于预测值减去平均预测值。由于我们在其他非线性模型（比如集成模型）中没有类似的权重，因此我们需要一个不同的解决方案，通过合作博弈论中的Shapley机器学习模型的单个预测的特征贡献。

每个特征值的Shapley值是该特征值对支付的贡献，通过对所有可能的特征值组合进行加权和求和得到：

$$\phi_j(val) = \sum_{S\subseteq\{x_1,\ldots,x_p\}\setminus\{x_j\}} \frac{|S|!\,(p-|S|-1)!}{p!}\left(val\left(S\cup\{x_j\}\right) - val(S)\right)$$

（1）

其中 S 是模型中使用的特征的子集（联盟），x 是要解释的实例的特征值的向量，p 为特征数量 ，其中 |S|!(p-|S|-1)! 为子集 S 的权重，这个权重与顺序有关，当确定了子集 S 后，S 本身有 |S|! 种顺序组合，然后后面要跟着特征 j，那么剩余的特征则有 (p-|S|-1)! 种组合，则确定子集S后有 |S|!(p-|S|-1)! 中顺序组合。$val_x(S)$是子集 S 的预测，未包含在集合S中的特征上被边缘化：

$$val_x(S) = \int \hat{f}(x_1,\ldots,x_p)d\mathbb{P}_{x\notin S} - E_X(\hat{f}(X))$$

实际上，对每个未包含的特征执行多个集成。例如：机器学习模型使用4个特征x1，x2，x3和x4，我们估计由特征值 x1 和 x3 组成的联盟 S 的预测：

$$val_x(S) = val_x(\{x_1,x_3\}) = \int_{\mathbb{R}}\int_{\mathbb{R}} \hat{f}(x_1,X_2,x_3,X_4)d\mathbb{P}_{X_2X_4} - E_X(\hat{f}(X))$$

这样计算得到的Shapley值是唯一满足**效率性（Efficiency），对称性（Symmetry），虚拟性（Dummy）和可加性（Additivity）**四个属性的归因方法，它们可以一起被视为公平支付的定义。

效率性：特征贡献的累加等于 x 的预测和预测平均值的差异。

$$\sum_{j=1}^{p}\phi_j = \hat{f}(x) - E_X(\hat{f}(X))$$

对称性：如果两个特征值 j 和 k 的贡献对所有可能的联盟贡献相同，则它们应该相同

对于所有的$S \subseteq \{x_1, \ldots, x_p\} \setminus \{x_j, x_k\}$，如果$val(S \cup \{x_j\}) = val(S \cup \{x_k\})$，则$\phi_j = \phi_k$

虚拟性：一个不改变预测值的特征 j ，无论它添加到哪个特征值联盟中，Shapley值都应该为0。

对于所有的$S \subseteq \{x_1, \ldots, x_p\}$，如果$val(S \cup \{x_j\}) = val(S)$，则$\phi_j = 0$

可加性：对于具有组合预测$val_1 + val_2$，相应的Shapley值如下：

$\phi_j = \phi_{j1} + \phi_{j2}$

假设训练了一个随机森林，这意味着预测是许多决策树的平均值。可加性保证对于特征值可以单独计算每个树的Shapley值，对它们求平均值，并获得随机森林的特征值的Shapley值。

通过使用和不使用第 j 个特征来估计所有可能的特征值联盟（集合），以计算精确的Shapley值。对于多个特征，这个问题的确切解决方案变得有问题，因为随着更多特征的添加，可能的联盟数量呈指数增长。 Strumbelj等人（2014）提出蒙特卡罗采样的近似值：

$$\hat{\phi}_j = \frac{1}{M} \sum_{m=1}^{M} \left(\hat{f}\left(x_{+j}^m\right) - \hat{f}\left(x_{-j}^m\right) \right)$$

其中$\hat{f}\left(x_{+j}^m\right)$是 x 的预测，除了特征 j 的值，其他不在联盟内的特征值被来自随机数据点 z 的特征值替换。 x 向量x_{-j}^m几乎与x_{+j}^m相同，但值x_j^m被随机数据点 z 的特征值替换。这

单个特征值的近似Shapley估计如下：

输入：迭代次数M，感兴趣的实例 x，特征索引 j，数据矩阵 X 和机器学习模型 f

输出：第 j 个特征值的Shapley值

对于所有m = 1，...，M：

- 从数据矩阵 X 中随机抽取实例 z
- 选择特征值的随机排列，联盟的特征在前面，即1至 j-1
- 顺序实例 x：$x_o = \left(x_{(1)}, \ldots, x_{(j)}, \ldots, x_{(p)}\right)$、
- 顺序实例 z：$z_o = \left(z_{(1)}, \ldots, z_{(j)}, \ldots, z_{(p)}\right)$
- 构造两个新实例
- 有特征 j：$x_{+j} = \left(x_{(1)}, \ldots, x_{(j-1)}, x_{(j)}, z_{(j+1)}, \ldots, z_{(p)}\right)$（j+1 至 p 的特征被随机实例替换）
- 没有特征 j：$x_{-j} = \left(x_{(1)}, \ldots, x_{(j-1)}, z_{(j)}, z_{(j+1)}, \ldots, z_{(p)}\right)$（j 至 p 的特征被随机实例替换）
- 计算边际贡献：$\phi_j^m = \hat{f}\left(x_{+j}\right) - \hat{f}\left(x_{-j}\right)$

计算平均值作为Shapley值：$\phi_j(x) = \frac{1}{M} \sum_{m=1}^{M} \phi_j^m$

平均值通过 X 的概率分布隐含地对样本进行加权得到，必须为每个特征重复该过程以获得所有Shapley值。

目前SHAP值是Shapley值的替代，在Python包shap中实现，SHAP值将Shapley值方法转换为优化问题，并使用特殊的内核函数来测量数据实例的接近度。 SHAP的结果是稀疏的（包含很少特征的解释），使用Shapley值方法解释模型始终需要所有特征，而SHAP可以提供几个特征的解释，这是与经典Shapley值的最大差异。

3. SHAP值

SHAP值(Shapley Additive exPlanation)统一了博弈论（Shapley）和局部解释（LIME）的思想，通过将集成树特征归因方法与加和特征归因方法相结合，SHAP值是具有三个理想特性的唯一可能的一致个性化特征归因方法，三个理想特性分别是局部精确性、缺失性、一致性。通过局部精确性的定义，局部精确性其实对应了Shapley值的效率。Young认为Shapley值的可加性和虚拟性可以用单调性代替，而 一致性其实就是单调性，只是Lundbergs在论文中称之为一致性，并且在论文中认为对于机器学习模型，单调性意味着对称性，所以一致性对应Shapley值的可加性、虚拟性和对称性。

目前许多解释机器学习模型局部预测的方法都属于加和特征归因方法（Additive feature attribution methods），比如LIME、DeepLIFT、 分层相关传播（layer-wise relevance propagation）、Shapley回归值（Shapley regression values），Shapley抽样值（Shapley sampling values）和定量输入影响（Quantitative Input Influence）。shap库对此提出了一个统一了六种现有方法的解释预测框架SHAP（Shapley Additive exPlanations），SHAP框架的应用方向有很多，比如TreeExplainer、DeepExplainer、GradientExplainer、KernelExplainer，本文只对TreeExplainer进行说明。SHAP基于一个样本为每个特征分配特定预测的重要性值，将模型输出解释为每个输入特征的归因值之和。

定义1 加和特征归因方法有一个解释模型，它是二元变量的线性函数：

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z_i', \qquad (2)$$

其中 $z' \in \{0,1\}^M$ 表示相应特征是否缺失（1或0），M是输入特征的数目，$\phi_i \in \mathbb{R}$ 是每个特征的归因值，ϕ_0 是解释模型的常数归因值（该值其实就是前面介绍shapley值时的 $E(\hat{f}(X))$，也就是所有样本的预测均值，shap库计算SHAP值时不会显示这个常数，可以调用shap.TreeExplainer(model).expected_value显示该常数）。

SHAP值的一个重要特性是该类中只有一个唯一的解，并且具有三个理想的特性：局部准确性，缺失性和一致性。

1. **局部精确性**表示特征归因的总和等于我们要解释的模型的输出，也就是说对于每一个样本，其**各个特征的归因值 ϕ_i 与常数归因值 ϕ_0 之和等于模型的输出值。**（在shap库的实际应用中会有些许差别，SHAP值已经被内嵌到xgboost、lightgbm库中，分别通过get_booster()、booster_获取模型隐藏的提升器，用其预测可以直接得到每个特征的归因值（SHAP值），但这个归因值是经过了对数优势比（log odds ratio）转换的（实际测试GBDT也用了这个转换，但随机森林未做转换，在shap库的源代码中不同模型的处理中都有些许区别，虽然GBDT和随机森林都是sklearn.ensemble调用的，但是在shap库的源代码中，开发者将GBDT做了对数优势比转换，对随机森林却没有做），即归因值总和等于每个样本的模型输出的对数优势比 $ln\frac{y}{1-y}$，这样可以扩大归因值的数值范围，有助于按照特征分解模型解释。由于SHAP值已经被内嵌到xgboost、lightgbm库的隐藏提升器中，在实际应用时，xgboost、lightgbm可以立马计算出结果，而其他模型需要较长时间。在实际应用时可以将shap_values输出的每个样本的shap值的和加上expected_value，观察其与模型输出的关系，测试结果可看**各模型SHAP值差异探索.ipynb**文件）；

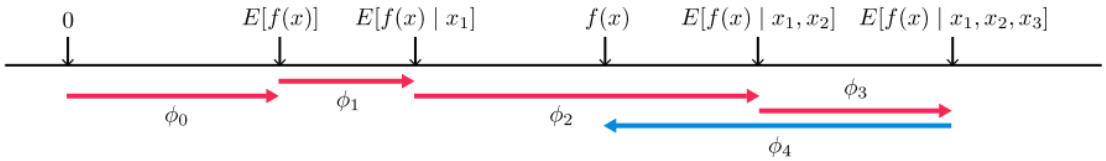
2. **缺失性**说明已经缺失的特性不重要， 即 $z_i' = 0 \implies \phi_i = 0$（这点在实际应用中较难体现，因为在训练过程一般会先将缺失值进行填补，而且很多模型无法处理缺失值）；
3. **一致性**表示，更改模型使一个特征对模型产生更大的影响时，不会减少分配给该特性的归因值。

为了计算SHAP值，我们定义$f_x(S) = E[f(x) \mid x_S]$，其中S是输入特征可能的子集合（shapley值提到的联盟），$E[f(x) \mid x_S]$是输入特征的子集S的条件期望值（shapley值提到的val函数）。 SHAP值将这些条件期望与从博弈论的经典Shapley值组合到每个特征的归因值 ϕ_i 中：

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(M - |S| - 1)!}{M!} [f_x(S \cup \{i\}) - f_x(S)] \quad (3)$$

其中N是所有输入特征的集合，M为所有输入特征的数目，$S \subseteq N \setminus \{i\}$为不包括 x_i的所有输入特征可能的集合，$f_x(S)$为特征子集 S 的预测。可以看到该公式和shapley值介绍中的公式1是相同的。

SHAP值的是唯一可能的一致且局部精确的方法，它遵循缺失性并使用条件依赖来测量缺失，这是使用SHAP值进行树集合特征归因的强烈动机。另外目前树模型唯一个性化特征归因方法Saabas使用条件依赖来满足局部准确性和缺失性，但由于只考虑单一的排序，不能满足一致性，而SHAP值计算了所有可能特征顺序的平均值，这意味着SHAP值通过消除显著的一致性问题提供了严格的理论改进。



通过上图可以看到 SHAP值将每个特征的归因值赋值为在调整该特征时预期模型预测的变化，将模型f 对于样本$\{x_1 = a_1, x_2 = a_2, x_3 = a_3, x_4 = a_4\}$的输出解释为引入条件期望的每个特征的影响 ϕ_i 的总和。上图解释了如果我们如何从$E[f(z)]$得到预测值。

此图只显示了单个排序的情况，那么上图的解释过程是：

首先S为空集时，$\phi_0 = f_x(\emptyset) = E[f(x)]$，其中$E[f(x)]$为模型预测值期望。

接下来 S 顺序加入变量x_1，此时 $\phi_1 = f_x(x_1 = a_1) - f_x(\emptyset) = E[f(x)|x_1] - E[f(x)]$，即$x_1 = a_1$时的模型预测值期望 - 模型预测值期望。

然后顺序加入 x_2 ，此时 $\phi_2 = f_x(x_1 = a_1, x_2 = a_2) - f_x(x_1 = a_1) = E[f(x)|x_1, x_2] - E[f(x)|x_1]$，即$x_1 = a_1, x_2 = a_2$时的模型预测值期望 - $x_1 = a_1$时的模型预测值期望。

以此类推，直至加入最后一个特征x_4，$\phi_4 = E[f(x)|x_1, x_2, x_3, x_4] - E[f(x)|x_1, x_2, x_3]$，即 $x_1 = a_1, x_2 = a_2, x_3 = a_3, x_4 = a_4$时的模型预测值期望 - $x_1 = a_1, x_2 = a_2, x_3 = a_3$时的模型预测值期望，此时$E[f(x)|x_1, x_2, x_3, x_4]$为四个特征单一排序下的预测值，其实就是样本 $\{x_1 = a_1, x_2 = a_2, x_3 = a_3, x_4 = a_4\}$的预测值$f(x)$。

但在实际情况下，当模型是非线性的或输入特征不是独立时，SHAP值是对所有可能的特征排序求平均值，即每一个ϕ_i计算时应考虑所有的排序情况，然后求平均，也就是根据等式3进行计算。

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(M - |S| - 1)!}{M!} [f_x(S \cup \{i\}) - f_x(S)] \quad (3)$$

首先看公式2左边的分数，分母认为在特征数为M时，M个特征在不同排序情况下有M! 种组合，比如在四个特征下有{1,2,3,4}，{1,2,4,3}，{1,3,2,4}，...{4,3,2,1}共24种组合，而分子则是在确定子集S后有多少种组合，首先子集已确定，它本身有 |S|! 种组合，然后后面要跟着x_i，那么剩余的变量则有 (M-|S|-1)! 种组合，则确定子集S后有 |S|!(M-|S|-1)! 中组合。

那么在有4个特征的情况下，我们想要计算ϕ_1时，首先$S \subseteq N \setminus \{i\}$应该有$\emptyset$，{2}，{3}，{4}，{2,3}，{2,4}，{3,4}，{2,3,4}八种特征集合。当S=$\emptyset$时， |S|!(M-|S|-1)!=0!3!=6，即第一个特征为{1}，后面的{2,3,4}有6种组合。当S={2}时，|S|!(M-|S|-1)!=1!2!=2，所以在2,1的顺序后{3,4}有2种组合。以此类推，最后当S={2,3,4}时，|S|!(M-|S|-1)!=3!0!=6，即{2,3,4}本身就有6种组合，后面只能接着{1}，这样遍历S可能的所有特征集合，然后求均值，在计算$f_x(S)$不考虑里面特征的具体顺序，所以子集 S 可能的组合情况作为权重。

至于如何计算$f_x(S)$，也就是$E[f(x) \mid x_S]$在下面介绍。

4. 计算TREE SHAP值

SHAP值理论上是最优的，本文推导了一种树集成算法**TREE SHAP**，将计算精确SHAP值的复杂度从O(TL2^M)降至O(TLD^2)，T是树的数量，L是在所有树中的叶子的最大数量，M是特征的数量，D是所有树的最大深度。这种复杂性的指数级降低，使得以前难以处理的模型(包含数千棵树和特性)的预测能够在不到一秒的时间内得到解释。

尽管SHAP值具有令人信服的理论优势，但它们的实际应用受到两个问题的阻碍:

(1)有效估计$E[f(x) \mid x_S]$的挑战。

(2)公式3的指数复杂度。

本文以树模型为研究对象，提出了针对树和集成树的快速SHAP值估计方法。我们首先定义一个缓慢但简单的算法，然后给出一个更快更复杂的TREE SHAP算法。

4.1 在O(TL2^M)时间内直接估计SHAP值

如果我们忽略计算复杂度，那么我们可以通过估计$E[f(x) \mid x_S]$计算树的SHAP值，然后使用公式3，其中$f_x(S) = E[f(x) \mid x_S]$。对于树模型$E[f(x) \mid x_S]$]可以使用算法1递归估计，其中 v 是节点值的向量，它取了内部节点的内部值，向量 a 和 b 表示每个内部节点的左右节点索引，向量 t 包含每个内部节点的阈值，d 是在内部节点中分割的特征的索引向量，向量 r 表示每个节点的覆盖(即有多少数据样本落在该节点中)，权重w表示与集合 S 匹配的训练样本落在每片叶子中的比例。

Algorithm 1 Estimating $E[f(x) \mid x_S]$

```
procedure EXPVALUE(x, S, tree = {v, a, b, t, r, d})
    procedure G(j, w)
        if v_j ≠ internal then
            return w · v_j
        else
            if d_j ∈ S then
                return G(a_j, w) if x_{d_j} ≤ t_j else G(b_j, w)
            else
                return G(a_j, wr_{a_j}/r_j) + G(b_j, wr_{b_j}/r_j)
            end if
        end if
    end procedure
    return G(1, 1)
end procedure
```

可以看到该算法和前面提到的shapley值的计算不同，shapley值对于每个集合 S 抽取 M 次样本，每次计算将集合 S 外的特征用抽取的样本替代，但这个方法是用蒙特卡罗采样的近似值去近似shapley值，这样是基于抽取样本去计算的，而TREE SHAP则是专门针对树模型去计算SHAP值的，由于树模型的特性，那么就可以用遍历节点的方法去计算集合 S 下的预测值期望。

对算法1计算$E[f(x) \mid x_S]$我的理解是累加每个根节点与集合 S 匹配的训练样本覆盖比例乘以根节点的预测值。

4.2 在O(TLD^2)时间内直接估计SHAP值

我们提出一种新的算法在多项式时间计算SHAP值，而不是指数时间。具体来说，我们提出一种O(TLD^2)运行时间和O(D^2+M)内存的算法，当数是平衡树时深度D变成D=logL，其中T为树的数量，L是所有树的叶子的最大数量，M是特征数量。多项式时间算法的流程是递归地跟踪所有可能的子集流向树的每个叶节点的比例，这类似于对公式3中的所有2^m的子集S同时运行算法1，简单地跟踪有多少子集(由算法1的覆盖分割加权)传递到树的每个分支似乎是合理的。然而，这样合并了不同大小的子集影响了这些子集的适当权重，因为公式3中的权重依赖于|S|。为了解决这个问题，我们在递归期间跟踪每个可能的子集大小。算法2中的EXTEND方法根据给定的1和0的比例增长所有的子集，而UNWIND方法则反转这个过程并与EXTEND交换，我们在树的下移过程中使用了EXTEND扩展方法，UNWIND方法用于在对同一特征进行两次拆分时撤消以前的EXTEND，并撤消叶子内路径的每个EXTEND，以计算路径中每个特性的权重。

算法2中，m是到目前为止已经分裂的特定特征的路径，它包含四个属性:

d：特征索引

z：流经这个分支的“0”路径(该特征不在集合S中)的分数

o：流经这个分支的“1”路径(该特征在集合S中)的分数

w：用于保存给定基数集合的比例。

我们用点符号来访问这些属性，对于整个向量m，d表示所有特征索引的向量。

算法2将树和集成树的精确SHAP值的计算复杂度从指数级降低到低阶多项式。

Algorithm 2 Tree SHAP

procedure TS(x, $tree = \{v, a, b, t, r, d\}$)
  ϕ = array of $len(x)$ zeros
  procedure RECURSE(j, m, p_z, p_o, p_i)
    m = EXTEND(m, p_z, p_o, p_i)
    if $v_j \neq internal$ **then**
      for $i \leftarrow 2$ to $len(m)$ **do**
        $w = sum(\text{UNWIND}(m, i).w)$
        $\phi_{m_i} = \phi_{m_i} + w(m_i.o - m_i.z)v_j$
      end for
    else
      $h, c = x_{d_j} \leq t_j \ ? \ (a_j, b_j) : (b_j, a_j)$
      $i_z = i_o = 1$
      k = FINDFIRST($m.d, d_j$)
      if $k \neq$ nothing **then**
        $i_z, i_o = (m_k.z, m_k.o)$
        m = UNWIND(m, k)
      end if
      RECURSE(h, m, $i_z r_h / r_j$, i_o, d_j)
      RECURSE(c, m, $i_z r_c / r_j$, 0, d_j)
    end if
  end procedure
  procedure EXTEND(m, p_z, p_o, p_i)
    $l = len(m)$
    $m = copy(m)$
    $m_{l+1}.(d, z, o, w) = (p_i, p_z, p_o, l = 0 \ ? \ 1 : 0)$
    for $i \leftarrow l - 1$ to 1 **do**
      $m_{i+1}.w = m_{i+1}.w + p_o m_i.w(i/l)$
      $m_i.w = p_z m_i.w[(l - i)/l]$
    end for
    return m
  end procedure

```
procedure EXTEND(m, p_z, p_o, p_i)
    l = len(m)
    m = copy(m)
    m_{l+1}.(d, z, o, w) = (p_i, p_z, p_o, l = 0 ? 1 : 0)
    for i ← l − 1 to 1 do
        m_{i+1}.w = m_{i+1}.w + p_o m_i.w(i/l)
        m_i.w = p_z m_i.w[(l − i)/l]
    end for
    return m
end procedure
procedure UNWIND(m, i)
    l = len(m)
    n = m_l.w
    m = copy(m_{1...l−1})
    for j ← l − 1 to 1 do
        if m_i.o ≠ 0 then
            t = m_j.w
            m_j.w = n · l/(j · m_i.o)
            n = t − m_j.w · m_i.z((l − j)/l)
        else
            m_j.w = (m_j.w · l)/(m_i.z(l − j))
        end if
    end for
    for j ← i to l − 1 do
        m_j.(d, z, o) = m_{j+1}.(d, z, o)
    end for
    return m
end procedure
RECURSE(1, [], 1, 1, 0)
return φ
end procedure
```

5. SHAP交互值

当前的归因方法不能直接表示相互作用，但必须在每个特征之间划分相互作用的影响。为了直接捕捉成对的相互作用效果，我们基于博弈论中Shapley交互指数扩展SHAP值，提出了SHAP交互值（SHAP INTERACTION VALUES），SHAP交互值保证了一致性的同时可以解释单个预测的交互效果。

特征归因通常在输入特征之间分配，每个特征分配一个归因值，但是我们可以通过将交互效果与主要效果分离来获得额外的信息。如果我们考虑成对的相互作用，就会得到一个属性值矩阵，表示所有的两个特征对给定模型预测的影响。由于SHAP值是基于博弈论中的经典Shapley值，所以通过更现代的Shapley交互指标可以得到对交互效果的自然扩展。

$$\Phi_{i,j} = \sum_{S \subseteq N \setminus \{i,j\}} \frac{|S|!(M - |S| - 2)!}{2(M-1)!} \nabla_{ij}(S),$$

when $i \neq j$, and

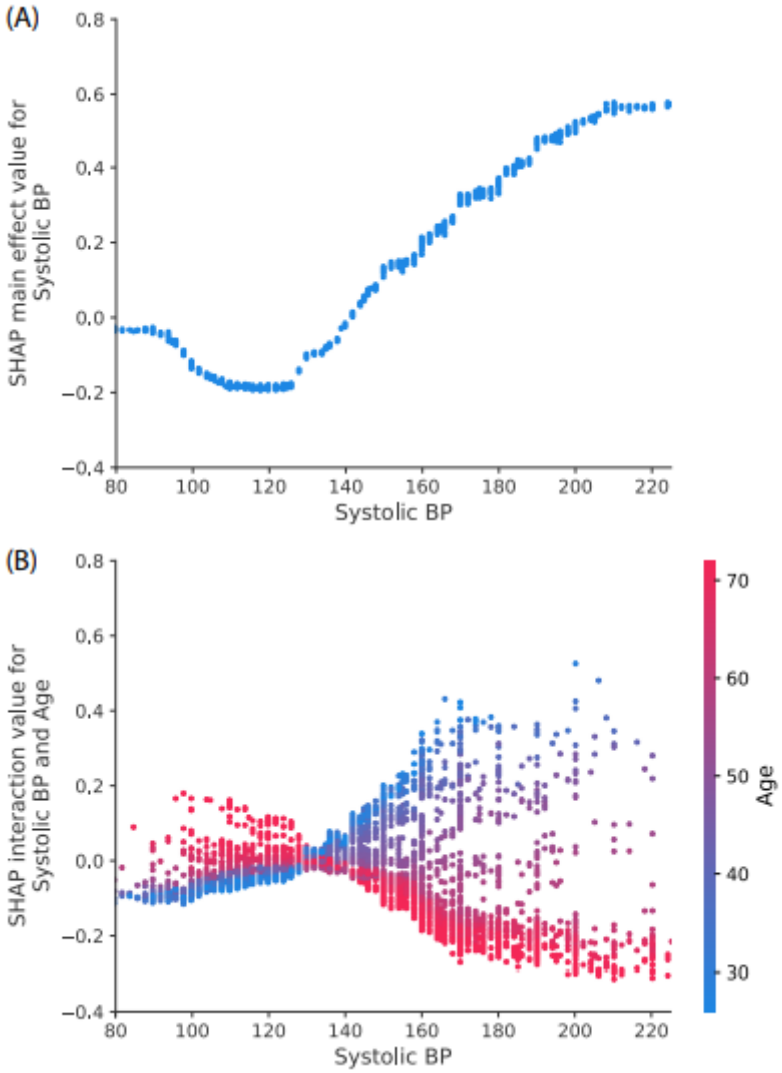
$$\begin{aligned} \nabla_{ij}(S) &= f_x(S \cup \{i,j\}) - f_x(S \cup \{i\}) - f_x(S \cup \{j\}) + f_x(S) \\ &= f_x(S \cup \{i,j\}) - f_x(S \cup \{j\}) - [f_x(S \cup \{i\}) - f_x(S)]. \end{aligned} \quad (4)$$

在方程4中SHAP交互值在特征 i 和特征 j 之间平均分割每个特性，所以 $\Phi_{i,j} = \Phi_{j,i}$ ，并且总交互作用影响为 $\Phi_{i,j} + \Phi_{j,i}$。特征对预测的主要影响可以定义为特征的SHAP值与SHAP交互值之间的差值:

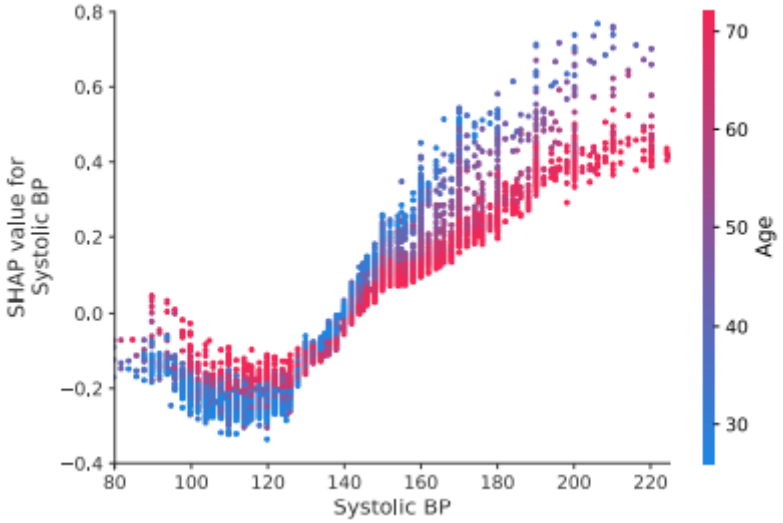
$$\Phi_{i,i} = \phi_i - \sum_{j \neq i} \Phi_{i,j} \quad (5)$$

SHAP交互值遵循与SHAP值类似的公理，允许单独考虑特征对模型预测的主要影响和交互作用影响，这种分离可以捕获集成树的重要交互。

SHAP相互作用值将Systolic BP的影响分为主要影响(图A，公式5，**自动化建模输出的报表中的单变量归因就是这个图**)与交互作用影响(图B，公式4，Systolic BP与Age有很强的交互作用）。



图A和图B图的和接近下图，下图其实就是直接绘制特征值与SHAP值的关系，每个点代表一个人，x轴表示Systolic BP，y轴表示Systolic BP引起的SHAP值，较高的SHAP值表示Systolic BP引起的死亡风险较高，根据人的Age给每个点上色表明，当你年轻的时候，你的模型更关注Systolic BP(这代表了一种互动效应)，说明直接对SHAP值绘图会受到交互作用的影响使点垂直分散，而图A的垂直分散较少，是因为所有的相互作用效应都被消除了。


SHAP value for
Systolic BP
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
80
100
120
140
160
180
200
220
Systolic BP
70
60
50
40
30
Age