<https://kknews.cc/zh-tw/other/jlv5y5p.html>

https://medium.com/@jimmywu0621/%E5%8F%AF%E8%A7%A3%E9%87%8B%EF%BD%81%EF%BD%89-%E4%BB%80%E9%BA%BC%E6%98%AFshap-5ec3953e3c5b

https://towardsdatascience.com/decomposing-a-time-series-in-a-simple-and-intuitive-way-19d3213c420b

在不久的過去我們曾經介紹 integrated gradient 的方法來做機械模型解釋，今天我們要介紹一個比較廣為人知的SHAP。人工智慧學校曾經有發表一篇關於 SHAP 的 tutorial，大家可以到本貼文所附的相關鏈結去閱讀。今天，我們就來輕鬆地談什麼是 SHAP 吧！

SHAP 是 SHapley Additive exPlanations 的縮寫，主要是計算 Shapley value 當作解釋每一個特徵對分類結果的貢獻。

那麼什麼是 Shapley value 呢？

Shapley 是在賽局理論中的 cooperative game 裏定義的。主要是為了計算所有合作玩家之間的平均貢獻。而計算方法則是窮舉所有可能的排列組合後所得的貢獻值再進行平均即可。

Shapley value 已成功的運用在 linear regression，被稱為 Shapley regression values 用來解釋每一個 variable 對結果的貢獻。然而因為窮舉排列組合的數目通常是一個天文數字，所以會改已取樣的方式來完成，而這個方式計算出來的 Shapley value 被稱為 Shapley sampling values。

瞭解了這些 Classic Shapley Values 的算法，再回到我們的 Shapley Additive Explanations 來吧！在名字中的 “additive” 是指所有重新簡化的特徵貢獻加總會等於於原來模型的輸出。重新簡化的特徵會具有解讀性，但 Additivity 的性質則會保證再加入可讀性後，轉換的特徵仍然具有原模型的餞別能力。目前已存在的 model interpreting 方法，如，LIME，DeepFIFT 和 Layer-Wise Relevance Propagation 甚至 classic Shapley values 都有這個特性。

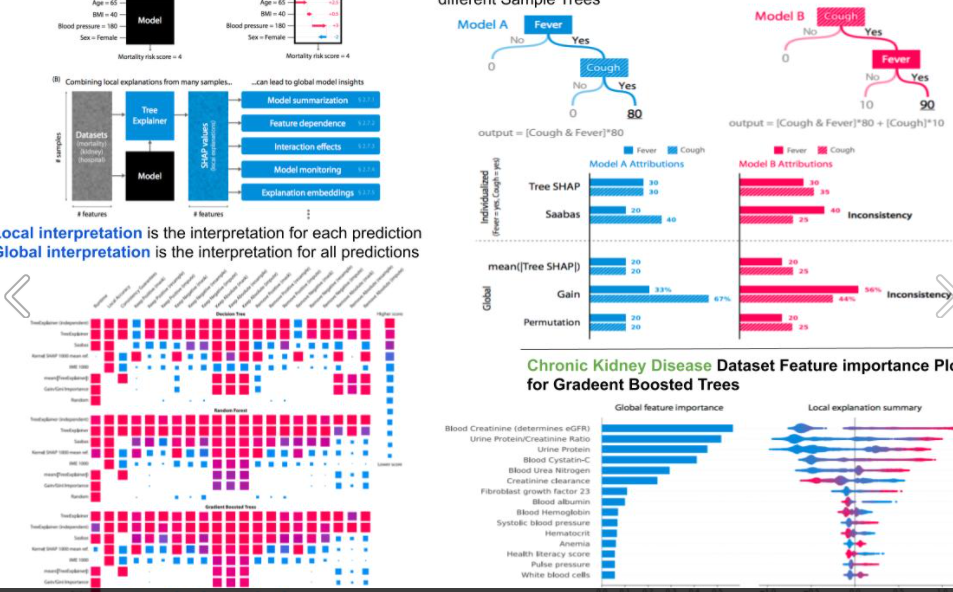
現在在發表的 Shapely package 裡有三種主要的 explainers 分別為 Kernel explainers，Deep SHAP 和 TreeExplainer。除了 Kernel explainers 是不需要針對特定模型來計算，其他兩者都針對特定模型。

關於 Kernel explainers，因為在古典的 Shapely values 計算過程中，需要 linear function，Kernel explainers 利用 kernel 的方式轉為 weighted linear function 的方法來解釋每個 feature 的貢獻。

Deep SHAP 其實是是 DeepLIFT 和 Shapely values 的結合。另外一種包括 integrated gradient 的是 Gradient Explainer。兩者都是根據類神經網路的連結模式，並用 approximate 的方法來計算 Shapely value。

最後 TreeExplainer，大家或許會覺得奇怪，樹模型本身就具有解釋力，為何在這裡還要再引入呢？這是因為雖然單一 decision tree 可以提供所指用 feature 的個別重要性，然而當使用 ensemble 方法時，太多的特徵之間的互動關係無法捕捉，再加上不同的 sample tree 的非一制性，而造成解讀困難。而 SHAP 的 framework 恰巧能夠用 approximate 的方式來計算每一個特徵對 local prediction 的貢獻（被稱為 local interpretion），也可以加總計算該特徵對所有 predictions 的 global interpretion。

在 Shapely 的網頁中也有提到他們不僅實踐了他們的方法，更合併喝其他非 Shapely value-based 的方法，如 LIME，DeepLIFT 和 Layer-wise relevance



**1 Shapley值**

SHAP值的主要思想就是Shapley值，Shapley值是一个来自合作博弈论（coalitional game theory）的方法，由Shapley在1953年创造的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收益。用shaply值去解释机器学习的预测的话，其中“总支出”就是数据集单个实例的模型预测值，“玩家”是实例的特征值，“收益”是该实例的实际预测减去所有实例的平均预测。

Shapley value起源於合作博弈論。比如說甲乙丙丁四個工人一起打工，甲和乙完成了價值100元的工件，甲、乙、丙完成了價值120元的工件，乙、丙、丁完成了價值150元的工件，甲、丁完成了價值90元的工件，那麼該如何**公平、合理**地分配這四個人的工錢呢？Shapley提出了一個合理的計算方法（有興趣地可以查看[原論文](https://link.zhihu.com/?target=https%3A//apps.dtic.mil/dtic/tr/fulltext/u2/604084.pdf)），我們稱每個參與者分配到的數額為Shapley value。

SHAP是由Shapley value啟發的可加性解釋模型。對於每個預測樣本，模型都產生一個預測值，SHAP value就是該樣本中每個特徵所分配到的數值。假設第ii個樣本為xixi，第ii個樣本的第jj個特徵為xi,jxi,j，模型對第ii個樣本的預測值為yiyi，整個模型的基線（通常是所有樣本的目標變量的均值）為ybaseybase，那麼SHAP value服從以下等式。

yi=ybase+f(xi,1)+f(xi,2)+⋯+f(xi,k)yi=ybase+f(xi,1)+f(xi,2)+⋯+f(xi, k)

其中f(xi,1)f(xi,1)為xi,jxi,j的SHAP值。直觀上看，f(xi,1)f(xi,1)就是對yiyi的貢獻值，當f(xi,1)>0f(xi,1)>0，說明該特徵提升了預測值，也正向作用；反之，說明該特徵使得預測值降低，有反作用。

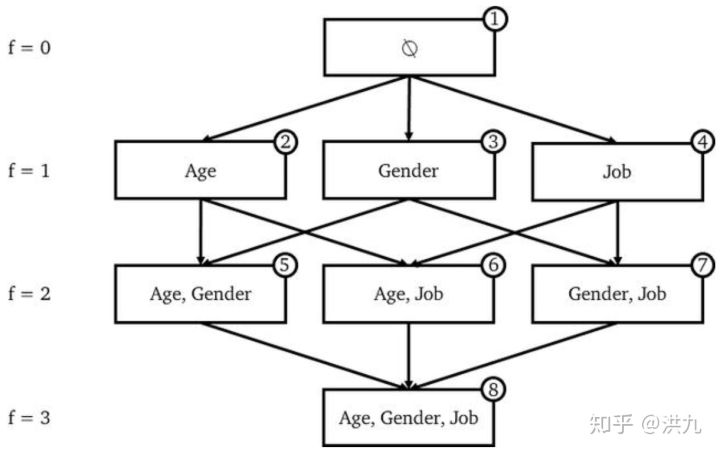
很明顯可以看出，與上一節中feature importance相比，SHAP value最大的優勢是SHAP能對於反映出**每一個樣本**中的特徵的影響力，而且還表現出影響的正負性。

## 2.2 Shape values

**Shape values**是"**SH**apley **A**dditive ex**P**lanations"的缩写。**Shap values**可以告诉我们一个特征对我们的一条预测结果产生了多大的影响。Shapley value起源于**合作博弈论，**比如说甲乙丙丁四个工人一起打工，甲和乙完成了价值100元的工件，甲、乙、丙完成了价值120元的工件，乙、丙、丁完成了价值150元的工件，甲、丁完成了价值90元的工件，那么该如何**公平、合理**地分配这四个人的工钱呢？Shapley提出了一个合理的计算方法，我们称每个参与者分配到的数额为Shapley value。

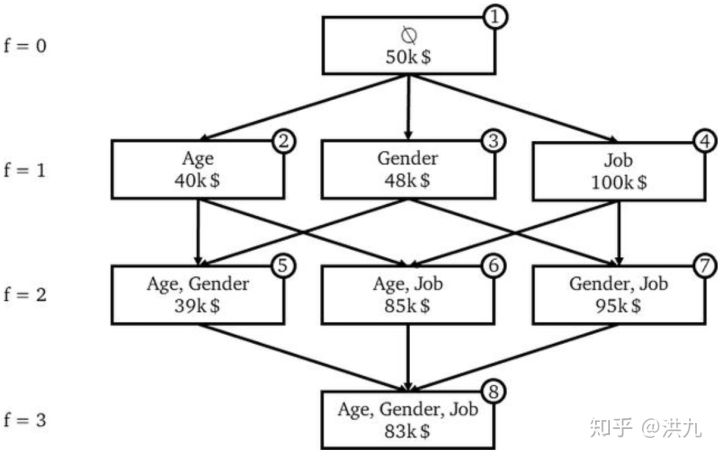
下面通过一个比较通俗易懂的例子讲解一下**Shape Values**的计算原理。假设现在有一个模型用于预测人的收入，模型考虑的因素有**年龄**、**性别**和**工作**。问题是"**年龄"**、"**性别"**和"**工作"**这三个因素对模型的贡献怎么衡量?

考虑下各种因素的组合，如下图:

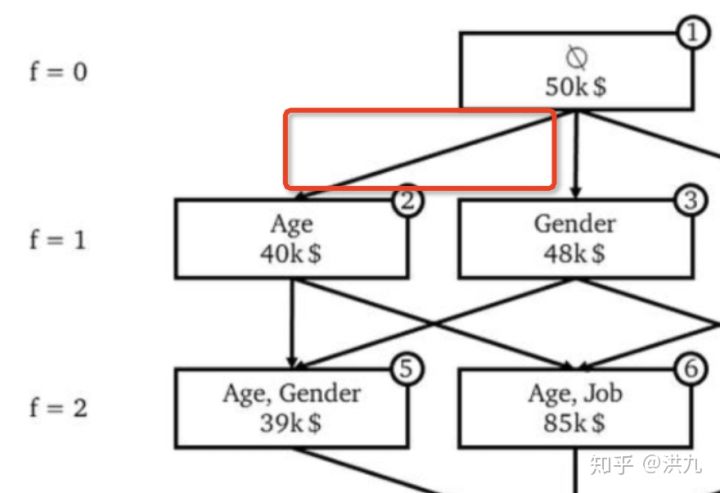


每个节点代表一个特征组合。每条边代表包含了一个在前一个组合中不存在的特征。比如③→⑤这条边，③中包含了因素"**Gender**"，而⑤包含了"**Age**"与"**Gender**"。从组合原理可知，这样的组合有  (  为因素个数)。

**接下来为每一种特征组合训练一个模型**(任意ML模型)。针对一个样本  ,每个模型对  都有一个预测值。如下图:

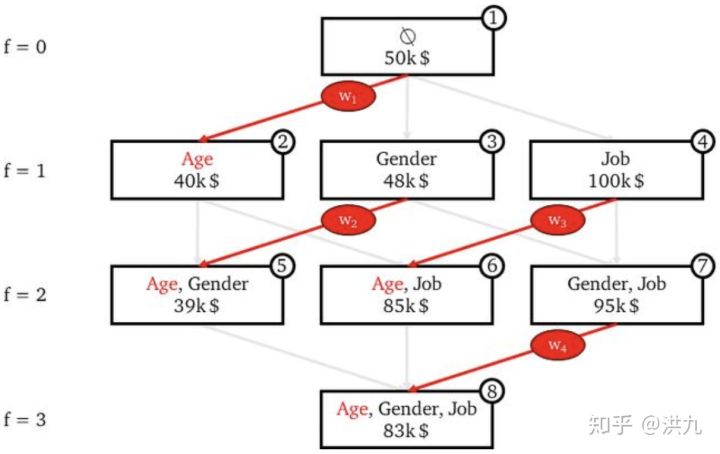


**两个连接节点的预测之间的差距**可以**归因于附加特征的影响**。这被称为特征的"**边际贡献**"。每个边都代表要素(**特征**)对模型带来的**边际贡献**。

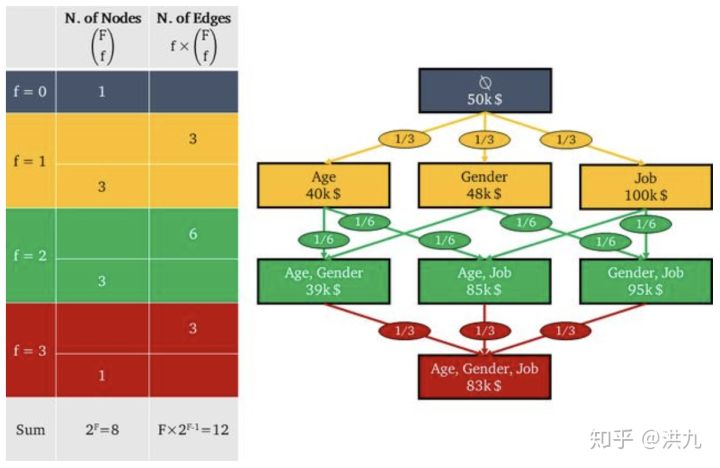


以上图中边为例，Age在该边中带来的**边际收益**为:

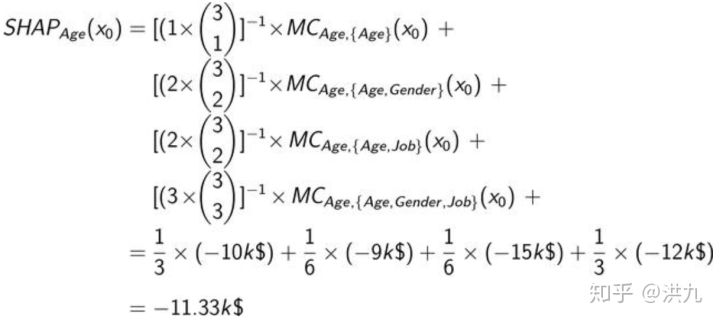
如果要考虑"Age"对整体模型的影响，则要综合所有相关边的边际收益:



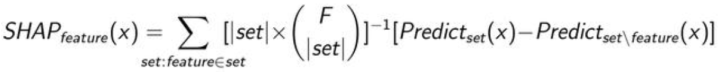
权重该如何确定呢? 其实是"**边的权值是同一'行'中边总数的倒数**"。



则最后的针对  的Age的**SHAP**值为:



所示统一的计算公式如下:



**参考文章:**[https://baijiahao.baidu.com/s?id=1654791131903418801&wfr=spider&for=pc](https://link.zhihu.com/?target=https%3A//baijiahao.baidu.com/s%3Fid%3D1654791131903418801%26wfr%3Dspider%26for%3Dpc)

上面是**Shap Values**计算的基本原理，而 Shap工具包中的具体计算方法是有差异的(目前还不清楚)，可以先理解下。很明显可以看出，与上一节中feature importance相比，SHAP value最大的优势是SHAP能对于反映出**每一个样本**中的特征的影响力，而且还表现出影响的正负性。

**Paper:A Unified Approach to Interpreting Model Predictions [**NIPS**]**

[http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf](https://link.zhihu.com/?target=http%3A//papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf)

**示例代码如下:**

**1).模型训练**

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

# 读取数据

data = pd.read\_csv('FIFA\_2018\_Statistics.csv')

# Label

y = (data['Man of the Match'] == "Yes") # Convert from string "Yes"/"No" to binary

# 特征名称

feature\_names = [i for i in data.columns if data[i].dtype in [np.int64]]

# 特征

X = data[feature\_names]

train\_X, val\_X, train\_y, val\_y = train\_test\_split(X, y, random\_state=1)

# 模型训练

my\_model = RandomForestClassifier(random\_state=0).fit(train\_X, train\_y)

**2).随机挑选样本用于计算Shap值**

# 挑选一个样本用于计算Shap values

row\_to\_show = 5

data\_for\_prediction = val\_X.iloc[row\_to\_show]

data\_for\_prediction\_array = data\_for\_prediction.values.reshape(1, -1)

# 模型预测值

my\_model.predict\_proba(data\_for\_prediction\_array)

输出结果为[0.3, 0.7]，说明对于该样本模型有0.7的概率预测为"最佳球员"。

**3).计算Shape值(这里用到了开源**shap**)**

import shap

# Create object that can calculate shap values

# 根据上面训练的模型创建shap解释器

explainer = shap.TreeExplainer(my\_model)

# Calculate Shap values

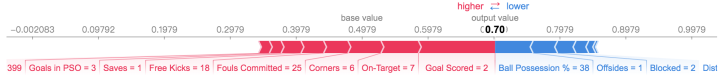
# 计算Shap values值

shap\_values = explainer.shap\_values(data\_for\_prediction)

**4).结果可视化**

shap.initjs()

shap.force\_plot(explainer.expected\_value[1], shap\_values[1], data\_for\_prediction)

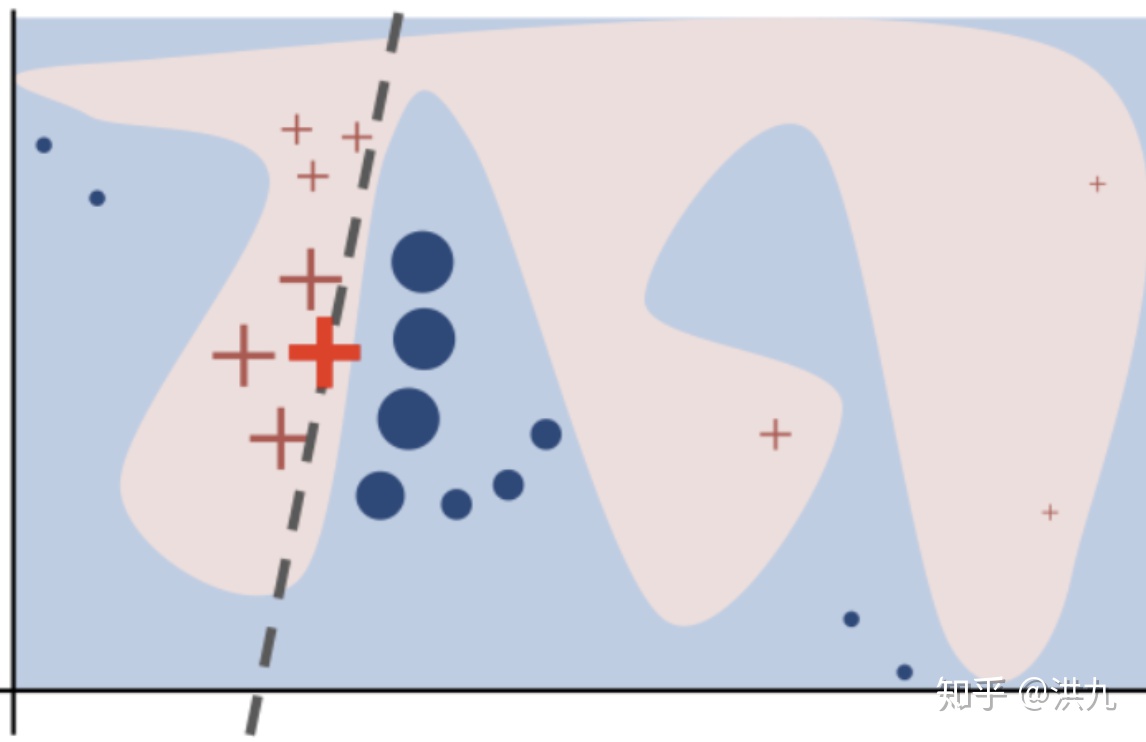


**图形解释**

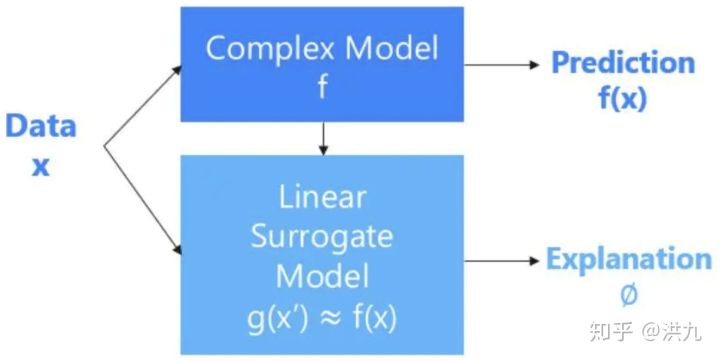
* Base value :模型在**数据集上的输出均值**：0.4979
* Output value:模型在**单个样本的输出值**：0.70
* 起正向作用的特征(**红色条**)：Goal Scored、On-Target、...
* 起负向作用的特征(**蓝色条**)：Bad Possession、Offsides、...

**2.3 LIME模型**

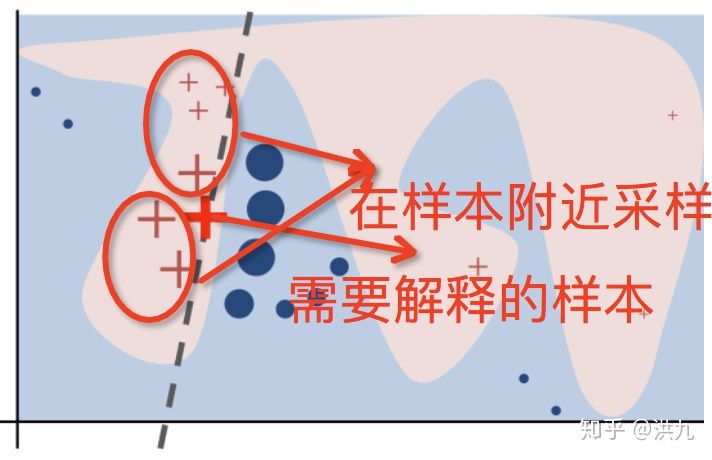
LIME(**local interpretable model-agnostic explanations**) ，局部可解释的模型无关的解释。如图所示，红色和蓝色区域表示一个**复杂的分类模型(**黑盒)，图中加粗的红色十字表示需要解释的样本。显然，我们很难从全局用一个可解释的模型(例如线性模型)去拟合它。但是，**当我们把关注点从全局放到局部时**，可以看到在**某些局部是可以用线性模型去拟合的**。虚线表示通过这些采样样本学到的局部可解释模型，在这个例子中就是一个简单的线性分类器。在此基础上，我们就可以依据这个**局部的可解释模型**对这个分类结果进行解释了。

LIME

线性模型的**权重大小反应了对应特征的重要程度**。本质上LIME**假设非线性模型的局部是近似线性的**，因此可以**用线性模型做为代理模型**去解释局部决策机制。

代理模型机制

具体做法可以选择一个"真实样本"，然后**在真实样本周围附近采样**。所谓采样就是对原始样本的特征做一些扰动，将采样出的样本用分类模型分类并得到结果（红十字和蓝色点）。用线性模型拟合采样的样本，通过线性模型的特征权重**理解非线性模型局部的决策机制**。



**还是上面的例子，代码如下:**

import lime

import lime.lime\_tabular

explainer = lime.lime\_tabular.LimeTabularExplainer(train\_X, feature\_names=feature\_names,

class\_names=['No','Yes'],

discretize\_continuous=False)

train\_sample = train\_X.sample(n=1)

pred\_p\_1 = my\_model.predict\_proba(train\_sample.values)

**可视化:**

exp = explainer.explain\_instance(train\_sample.values[0],

my\_model.predict\_proba,

num\_features=len(feature\_names),

top\_labels=1)

exp.show\_in\_notebook(show\_table=True, show\_all=False)



观察LIME的结果可以看出，"Goal Scored"特征所起的正向作用是最大的，这与"**Permuation Importance**"分析得到的结论是一致的。

**2.4 模型定制化工具(Xgboost)**

不少机器学习工具根据模型的特点提供了"**特征重要性计算工具**"，比如Xgboost。

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

# 读取数据

data = pd.read\_csv('FIFA\_2018\_Statistics.csv')

# Label

y = (data['Man of the Match'] == "Yes") # Convert from string "Yes"/"No" to binary

# 特征名称

feature\_names = [i for i in data.columns if data[i].dtype in [np.int64]]

# 特征

X = data[feature\_names]

train\_X, val\_X, train\_y, val\_y = train\_test\_split(X, y, random\_state=1)

# 模型训练

model = XGBClassifier()

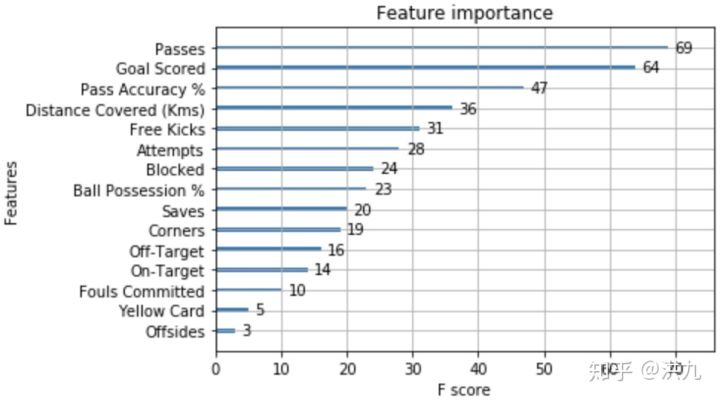
my\_model = model.fit(train\_X, train\_y)

Xgboost提供了**三种**不同计算特征重要度的方法:

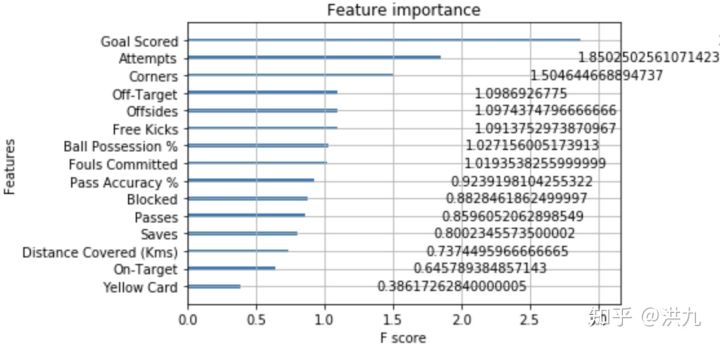
1. **importance\_type=weight**，特征重要性使用特征在所有树中**作为划分属性的次数**。
2. **importance\_type=gain**， 特征重要性使用特征在作为划分属性时**loss平均的降低**。
3. **importance\_type=cover**，特征重要性使用特征在作为划分属性时**对样本的覆盖度**。

**1).特征重要度(**importance\_type=weight**)**

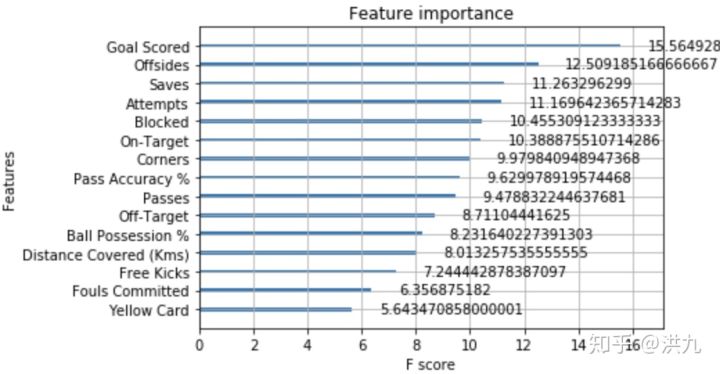
plot\_importance(my\_model,importance\_type=weight)



**2).特征重要度(**importance\_type=gain**)**



**3).特征重要度(**importance\_type=cover**)**

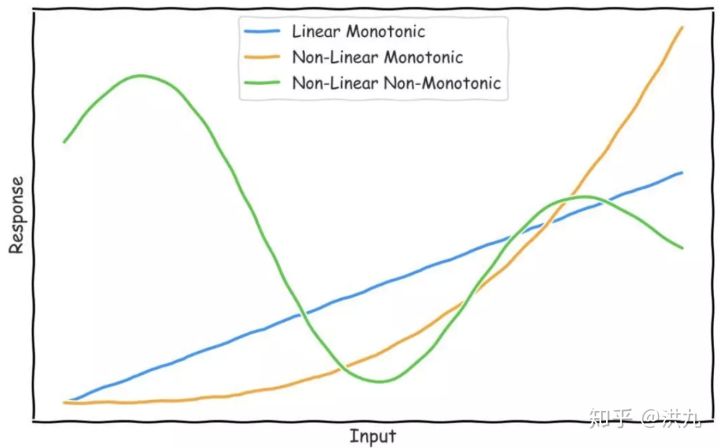


可以看到在不同的计算方法下，**特征的重要程度是一致的**。这也是此类特征重要性计算方法的最大问题所在。

## 3.单特征分析

特征重要性可以告诉我们哪些特征是重要的或不重要的，但无法回答某一特征是怎么影响预测结果的。比如:

* 在金融信用评估中，是什么因素导致顾客没有通过审核?
* 在放假预测中，经度和维度是如何影响房价的?



通常单特征与模型预测结果的关系有三种:

1. **线性，单调**：这些是最可解释的函数，特征的更改会导致目标变量在单个方向（正或负）上成比例变化。这允许我们计算相对特征重要性度量和原因代码。原因代码有助于理解为什么学习模型预测特定输入的高或低。它为我们提供了输入特征预测影响的定性度量，即它是强，弱，正还是负。
2. **非线性，单调**：机器学习算法学习的大多数响应函数都是非线性的。单调性可确保输入要素的相对变化导致目标变量在单个方向上发生变化。非线性，单调函数也是可解释的，因为它允许我们计算相对特征重要性度量和原因代码。
3. **非线性，非单调**：这些是现代机器学习算法学习的最常见的函数，并且是最难解释的。这是因为输入变量的变化会导致目标变量在任何方向上以不同的速率发生变化。我们可以为这些函数计算的唯一可解释的度量是相对特征重要性。解释这些功能是本文的重点。

**上面摘抄自:**[https://www.jianshu.com/p/c88a13ceae24](https://link.zhihu.com/?target=https%3A//www.jianshu.com/p/c88a13ceae24)

本小节要讲解的**Partial Dependence Plots(部分依赖图)**是专门用于分析特征对预测结果影响的工具。"**Partial Dependence**"顾名思义，通过只改变**部分变量**的值而固定其余变量的方式，研究**部分变量**与**预测目标**的关系。实现上，可同时改变多个样本的**部分变量为相同值**，然后求**预测结果的平均值**。开源工具包**pdpbox**提供了类似功能**。**

**实例如下:**

**1).模型训练**

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

# 读取数据

data = pd.read\_csv('FIFA\_2018\_Statistics.csv')

# Label

y = (data['Man of the Match'] == "Yes") # Convert from string "Yes"/"No" to binary

# 特征名称

feature\_names = [i for i in data.columns if data[i].dtype in [np.int64]]

# 特征

X = data[feature\_names]

train\_X, val\_X, train\_y, val\_y = train\_test\_split(X, y, random\_state=1)

# 模型训练

my\_model = RandomForestClassifier(random\_state=0).fit(train\_X, train\_y)

**2)."Goal Scored"特征的部分依赖图**

这里用到了工具包**pdpbox**

from matplotlib import pyplot as plt

from pdpbox import pdp, get\_dataset, info\_plots

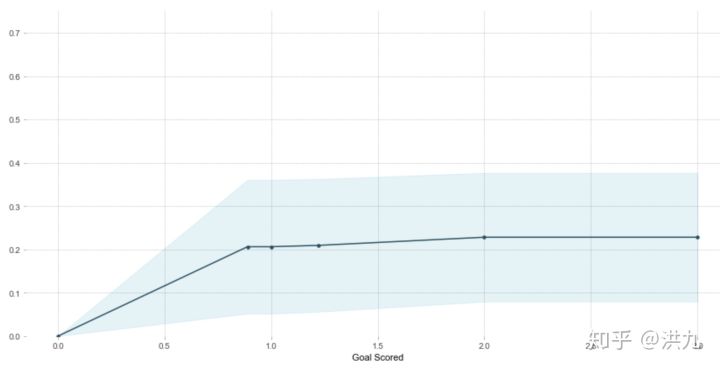
# 创建好画图所需的数据

pdp\_goals = pdp.pdp\_isolate(model=my\_model, dataset=val\_X, model\_features=feature\_names, feature='Goal Scored')

# 画出“Goal Scored”这一特征的partial dependence plot

pdp.pdp\_plot(pdp\_goals, 'Goal Scored')

plt.show()



其中，**横坐标(x轴)指的是特征"Goal Scored"的取值范围，纵坐标(y轴)代表预测结果**。可以看出随着Goal Scored的增大，该球员是最佳球员的概率也在增大。但当Goal Scored增大到一定程度后，结果趋于收敛。蓝色区域表示**置信度**。

## PS.在工程实践中，对于特征(尤其是"强特")做PDP分析是非常有必要的，判断特征值与模型预测值之间的关系是否符合预期，提前预判上线后可能会发生的风险。

## 4.多特征交互分析

很多情况下，**单独分析一个特征可能其重要性并不突出**，此时简单的认为其不重要是不严谨的。**因为当其与他特征共同出现时，由于组合效用其对模型的重要性并不能忽视**。所示，**特征间的交互分析也是很有必要的**。

还是前文的例子，利用**PDP(pdpbox)**分析下"**Goal Scored**"与"**Distance Covered (Kms)**"特征的交互对模型结果的影响是什么样的?

from matplotlib import pyplot as plt

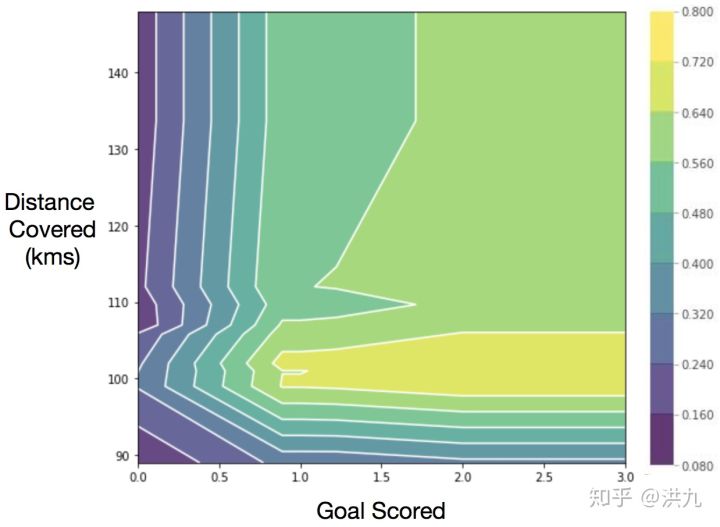
from pdpbox import pdp, get\_dataset, info\_plots

features\_to\_plot = ['Goal Scored', 'Distance Covered (Kms)']

inter1 = pdp.pdp\_interact(model=my\_model, dataset=val\_X, model\_features=feature\_names, features=features\_to\_plot)

pdp.pdp\_interact\_plot(pdp\_interact\_out=inter1, feature\_names=features\_to\_plot, plot\_type='contour')

plt.show()



上图可以看出，最高的预测值来自当球队进球且移动距离接近100KM的时候(**黄绿色部分**)。当进球数为0的时候，跑动距离多少都没有用，相当于在做"**无用功**"。

同样，Shap也提供了相关的工具，并且可以展示出**样本的分布**。

import shap

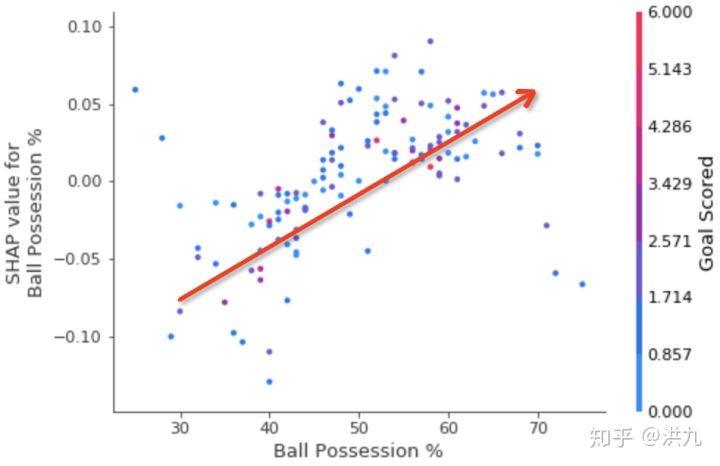
# 创建解释器

explainer = shap.TreeExplainer(my\_model)

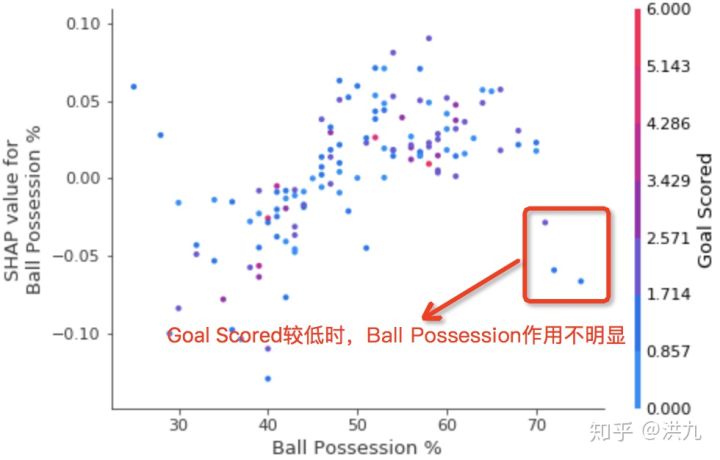
shap\_values = explainer.shap\_values(X)

# 可视化

shap.dependence\_plot('Ball Possession %', shap\_values[1], X, interaction\_index="Goal Scored")



**从整体趋势上看**，图中每一个点都代表一个样本，"Ball Possession(控球率)"越高对模型的**正向贡献**越大。



**从局部细节上看**，右下部分的几个样本，由于Goal Scored较低(**样本点颜色**)，导致**尽管Ball Possession较高但对模型的正向影响(SHAP Values)较小**。

## 通过这样的分析，可以抓到更多关于模型决策机制的信