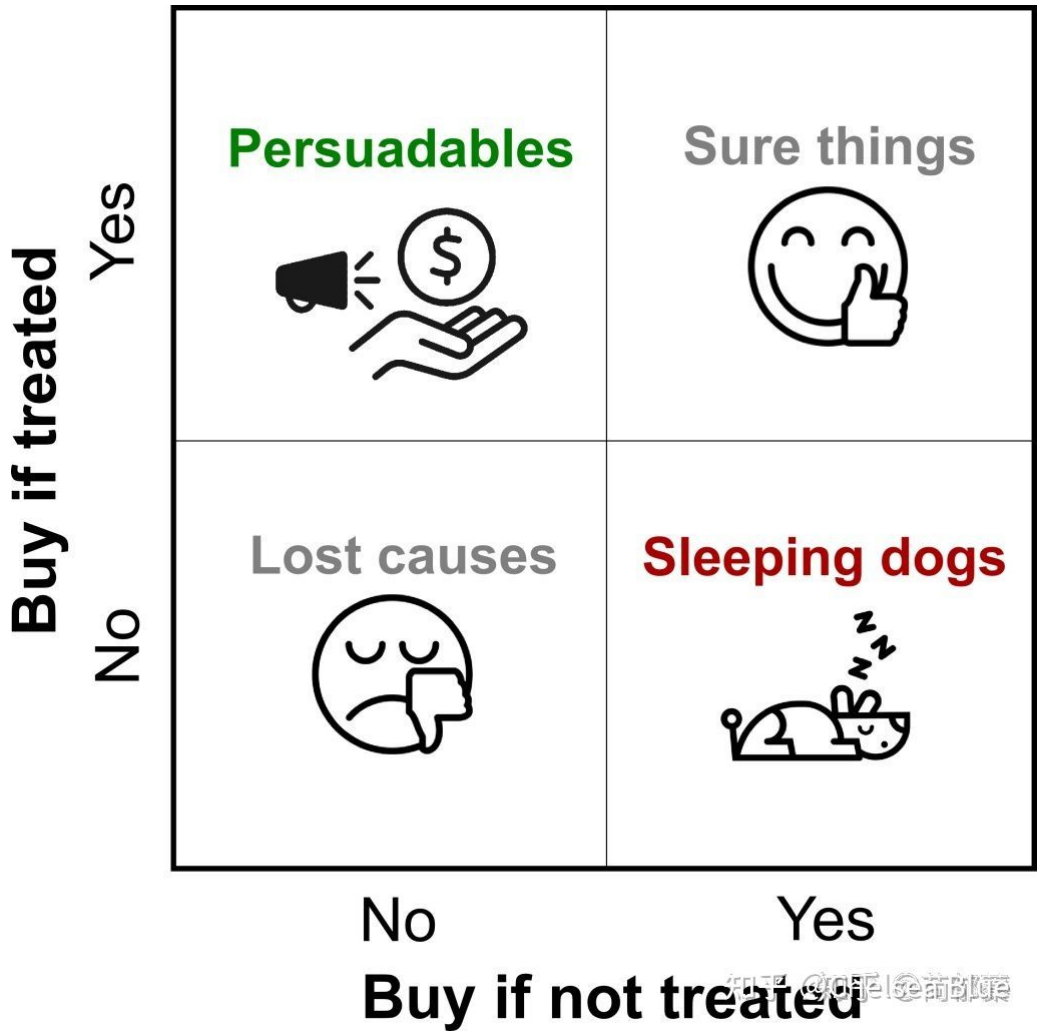
Uplift 模型总结

uplift模型的motivation



这里我们可以把treat当作给他优惠券，我们一共有四种人，左上角是给券买，不给就不买；右上角是给不给券都买；左下角是给不给券都不买；右下角是给券不买，不给券就买（论文中称之为sleeping dog）。我们希望找到图里左上角那那部分人（即给券就买，不给就不买），这里treatment就是给券，我们自然避免把钱花在其他三种人上。因此我们就需要去评价一个人对与给券比不给券带来的额外购买意愿。一般的机器学习模型是预测这个人给券后的购买概率，这不是我们要的，我们要的是这个人给券比不给券购买概率的增量。

heterogeneous treatment effects

ATE和CATE的概念。我们T=1作为给券(treated)，T=0作为不给券(untreated)，Y作为最后的购买概率(outcome)。X是这个人的一些特征(feature)

ITE：individual treatment effect

$$ITE = Y_i(T=1) - Y_i(T=0)$$

ATE：average treatment effect

$$ATE = E[Y(T=1) - Y(T=0)]$$

CATE：conditional average treatment effect

$$CATE = E[Y(T=1) - Y(T=0)|X=x]$$

uplift模型想要的是某个个体给券不给券的区别，而不是总体给券不给券的区别，所以这里就不是研究ATE，而是研究ITE。而由于我们是使用observational data，ITE是CATE的一个特殊情况，我们其实研究的是CATE：an average treatment effect specific to a subgroup of subjects, where the subgroup is defined by subjects' feature。举个例子：如果我们的特征包括性别，年龄，职业，app活跃度。那我们其实想知道的是：一个**app**活跃度高的**28**岁男性程序员给券和不给券对购买概率的差异。

而这里CATE模糊来讲，就是heterogeneous treatment effects。因为他认为总的population是heterogeneous的，所以我们要通过X来区隔出一个个subpopulation。于是我们成功从uplift模型过渡到了一个研究heterogeneous treatment effects的问题 (为了后续行文方便，heterogeneous treatment effects我们有时候会写成treatment effects或者CATE或者ITE，都理解为一个东西就好)。

problem setting

我们的对象有一系列特征 X 和treatment T（有的文章用 W），$T=1$ 时为treated，$T=0$ 时为untreated。Y^0 是 $T=0$ 时的outcome，Y^1 是 $T=1$ 时的outcome我们的目标heterogeneous treatment effects（CATE）标记为 $\tau(x) = E[Y^1 - Y^0|x]$，即given特征x，$T=1$时的outcome和 $T=0$ 时的outcome的差的期望值。

Meta-learner Algorithms

A meta-algorithm (or meta-learner) is a framework to estimate the Conditional Average Treatment Effect (CATE) using any machine learning estimators (called base learners) [2]。也就是这类算法直接用machine learning model作为基学习器来学习，比较简单。

S-learner：

这个模型最简单，直接把treatment作为特征放进模型来预测。
首先我们把 T 作为特征一起放进机器学习模型的特征， Y 是目标，然后训练一个有监督的模型 $\mu(x) = E[Y|X = x, T = t]$ 。然后我们改变 T 的值，就可以得到两个不同的结果，再一相减就好了：

$$\hat{\tau}(x) = \hat{\mu}(x, T = 1) - \hat{\mu}(x, T = 0)$$

T-learner：

这个模型不同于上一个，他先对于 $T = 0$ 的control组和 $T = 1$ 的treatment组分别学习一个有监督的模型。control组模型只用control组的数据，treatment组模型只用treatment组的数据。

$$\mu_0(x) = E[Y^0|X = x]$$
$$\mu_1(x) = E[Y^1|X = x]$$

然后要预测的样本分别进入这两个模型预测，结果再一相减就完事儿了

$$\hat{\tau}(x) = \hat{\mu}_1(x) - \hat{\mu}_0(x)$$

X-learner[6]：

类似上一个，先对于 $T = 0$ 的control组和 $T = 1$ 的treatment组分别学习一个有监督的模型。

$$\mu_0(x) = E[Y^0|X = x]$$
$$\mu_1(x) = E[Y^1|X = x]$$

然后对于 $T = 1$ 的样本和 $T = 0$ 的样本，分别使用 $\mu_0(x)$ 和 $\mu_1(x)$ 预测一个 $\hat{\mu}_0(X^1)$ 和 $\hat{\mu}_1(X^0)$，这步就是就是获得一个反事实的结果（比如对于某一个 $T = 1$ 的样本 X_i^1 ，事实结果是 Y_i^1 ，反事实结果是 $\hat{\mu}_0(X_i^1)$ 。于是我们就可以对于control组和treatment组分别计算difference D_i^0 和 D_i^1 ：

$$D_i^1 = Y_i^1 - \hat{\mu}_0(X_i^1)$$
$$D_i^0 = \hat{\mu}_1(X_i^0) - Y_i^0$$

下一步我们再用一个机器学习模型来拟合这两个difference：

$$\tau_1(x) = E[D^1|X = x]$$
$$\tau_0(x) = E[D^0|X = x]$$

最终对于一个新样本的CATE就是这两个的加权平均，权重是propensity score $g(x) = P(T = 1|X = x)$，这个可以通过一个LR模型或者任何二分类模型得到得到。最后新样本的的CATE如下：

$$\hat{\tau}(x) = g(x)\hat{\tau}_0(x) + (1 - g(x))\hat{\tau}_1(x)$$

总结

以上便是常用的3种meta-learner的方法，这类方法比较简单，也很好理解，有很强的解释性。除此之外，还有**R-Learner[3]**

Tree-based algorithms

uplift tree[4]

变量定义

首先我们定义一些变量，Y是我们的目标变量，A是树模型里的树分裂的test (或者说splitting criteria)。然后我们这里的数据集是分成两组，一个叫treatment group T，一个叫control group C。

树分裂规则

回想一般的决策树，我们的splitting criteria的选择是希望分裂前后的information gain最大，这里同理。不过不同的是，我们希望的是分裂前后，treatment group的Y和control group的Y的分布差距的增益最大。即如果我们定义 $D(P:Q)$ 为一种度量 P 和 Q 的分布差异的方式（divergence），我们希望能maximize下面的公式:

$$D_{gain}(A) = D(P^T(Y) : P^C(Y)|A) - D(P^T(Y) : P^C(Y))$$

也就是我们希望build the tree such that the distributions in the treatment and control groups differ as much as possible . 同理，和决策树 (C4.5)一样，这个gain也需要除以一个标准值来做一定的修正。

这里我们定义一下 $D(P:Q)$ 这个divergence，可以是如下三种：

$$KL(P:Q) = \sum_i p_i log \frac{p_i}{q_i}$$

$$E(P:Q) = \sum_i (p_i - q_i)^2$$

$$\chi^2(P:Q) = \sum_i \frac{(p_i - q_i)^2}{q_i}$$

也就是KL散度，欧式距离，和卡方散度。非常简单

最后我们定义一下 $D(P:Q|A)$ 这个conditional divergence。这里我们补充一下， a 是 A 的一种outcome， N 是样本数， $N(a)$ 是按照 A 分裂的结果是 a 剩下的样本数，我们有 $N = N^T + N^C$ 和 $N(a) = N^T(a) + N^C(a)$ 。

$$D(P^T(Y):P^C(Y)|A) = \sum_a \frac{N(a)}{N} D(P^T(Y|a):P^C(Y|a))$$

如何计算结果

现在我们关注当这个树建好后，我们该如何对于一个新的样本去计算这个treatment effect呢？首先现在的叶子节点都是subgroups of objects for which the treatment class distribution differs from control class distribution. 简单来讲，如果 Y 取0和1，1是我们希望的结果，那么对于一个新样本，如果落到了叶子节点 l 上，treatment effect就是 $P^T(Y=1|l) - P^C(Y=1|l)$ 。如果 Y 是一个连续变量，treatment effect就是 $E^T(Y|l) - E^C(Y|l)$ ，也就是这个叶子节点上的treatment组Y的均值减去这个叶子节点上的control组Y的均值。

如果control组是 $T=0$ ，treatment组是 $T=1$ ，最后的CATE用公式表达就是：

$$\hat{\tau}(x) = \frac{1}{|i:T_i=1, X_i \in l|} \sum_{\{i:T_i=1, X_i \in l\}} Y_i - \frac{1}{|i:T_i=0, X_i \in l|} \sum_{\{i:T_i=0, X_i \in l\}} Y_i$$

论文中还提到一个有意思的观点，如果说我们执行这个treatment的成本是 c ，每种 $Y=y$ 的利润是 v_y 。那么我们得到一个expected gain为

$$-c + \sum_y v_y (P^T(y|l) - P^C(y|l))$$

我们只需要这个值大于0，对于新样本，这个treatment就是可以执行的。

Causal Forest[5]

这个方法与其说是一个算法，不如说是一类算法，他们的核心都是把一个个建立好的causal tree (uplift tree)做一个ensemble，然后把每棵causal tree (uplift tree)计算出来的treatment effect取一个平均。treatment effect的计算方法见上面uplift tree介绍的。

当然如果要这么做，其实我们需要满足一个assumption，就是unconfoundedness，即：$T \perp Y|X$。就是在叶子结点上控制住所有的confounder X后，treatment和outcome要独立。

所以方法很简单，难点在于怎么建立causal tree才能满足。我们要建一棵honest tree。简单来说，就是我们对于任意样本 i，他的 Y_i 要不然只能用来做 $\hat{\tau}(x)$ 的预测，要不然只能用来做树的分裂依据，只能二选一。这个其实非常有道理，大家可以去看看这篇文章。

下面说两种建立honest tree的方法。

Double-Sample Tree[5]

首先我们把数据分成两个set，分别是$L-sample$ 和 $J-sample$，每个sample都有T，X和Y三种变量。我们只用 $J-sample$ 的全部数据和 $L-sample$ 里的T，X来做叶子节点的分裂，最后用 $L-sample$ 里的Y去计算 $\hat{\tau}(x)$。那么这里就有两个问题：1. 分裂依据是什么？2. 这样不是样本很浪费吗？

第一个问题，上面uplift tree的分裂规则其实不适用了，因为我们有一部分的样本($L-sample$)是不能用到Y的。怎么办呢？回忆对于普通决策树的回归树，比如CART回归树。 我们分裂依据是minimize MSE（即如下公式)

$$\sum_i (\hat{\mu}(X_i) - Y_i)^2 = \sum_i Y_i^2 - \sum_i \hat{\mu}(X_i)^2$$

推导得这等价于我们要maximize $\sum_i \hat{\mu}(X_i)^2$。因为我们有 $\hat{\mu}(X_i) = \frac{1}{n}\sum_i Y_i$，所以我们简单的数理统计公式推导就可以得到这个又等价于maximize $\hat{\mu}(X_i)$ 的variance（把他variance的公式写出来自己推推看）。所以同理，对于因果的honest tree，我们就是要maximize $\hat{\tau}(x_i)$ 的variance。这里就不需要用到 $L-sample$ 的那部分Y了吧。

第二个问题，如果只是这一棵树，确实很浪费，但是巧就巧在，如果用random forest的思想，那我们有很多树，而每棵树的 $L-sample$ 和 $J-sample$ 都是有放回抽样得到的，这样其实每个样本最后都即会被用来分裂，又会被用来预测。

Propensity Tree[5]

这个方法其实更简单，既然我们不能用Y来做为分裂依据，我们就只用T来分裂就会，Y全部留给估计 $\hat{\tau}(x)$ 就好了。

方法如下：首先我们只用数据中的X和T，T是训练目标，X是训练特征，训练模型是CART，就类似用CART训练一个propensity score。而这颗CART就是最后的causal tree。最后每个样本的预测就在这棵CART的叶子节点上用Y去做相同的 $\hat{\tau}(x)$ 估计。这个思想其实也很简单，就是每个叶子结点上都是propensity score最接近的，其实也是把他们matching了起来。所以这个方法在observational study中用的也很多。

其他树模型

理论上其他树模型也可以和S-learner一个逻辑，把t和X一起放到模型里面去训练Y，预测时候用t=1的情况减去t=0的情况，比如很多人就用BART[9]模型来这么做，当然理论上什么随机森林这种bagging的方法也是一样的，所以这种其实不能叫tree-based方法，应该叫meta-leaner的方法，因为他这里也是把一颗颗树当作base learner。