

中山大学 MA7259 《机器学习》期中作业  
美国高校招生填报率统计数据的分析和预测

何瑞杰 25110801

摘要

本研究基于 1995 年美国高校数据集 (college)，旨在预测学校申请人数并识别关键影响因素；探索性分析显示数据集以私立学校为主，申请数分布极右偏，且与录取数、入学数强相关，而公私立学校在费用和毕业率上差异显著。通过线性回归 ( $R^2 \approx 0.75$ ) 和决策树 ( $R^2 = 0.85$ ) 建模，发现录取率、师生比、生均开支等特征影响显著，但决策树优于线性模型，同时揭示数据中存在因果混淆问题。最终结论强调学校资源、生源质量和学生关注点（如毕业率）共同驱动申请行为，然数据陈旧限制泛化性，需后续因果分析增强可靠性。

## 目录

1. 研究背景与目的 . . . . .	3
2. 探索性数据分析 . . . . .	3
2.1. 数据集介绍 . . . . .	3
2.2. 重要统计量分析 . . . . .	3
2.3. 相关矩阵 . . . . .	5
2.4. 公立学校和私立学校的部分特征数据分析 . . . . .	6
2.5. 降维可视化 . . . . .	7
2.6. 正态性检验 . . . . .	8
3. 方法和实验 . . . . .	9
3.1. 数据预处理 . . . . .	9
3.2. 线性回归及其变体 . . . . .	9
3.3. 决策树 . . . . .	11
4. 结论和讨论 . . . . .	12
5. 附录 . . . . .	13

## 1. 研究背景与目的

高等学校的填报是国内高中毕业生在结束高考后面临的第一重难题，在大洋彼岸也是如此。在择校时，我们常常会事先了解学校所处的地域、办学能力、师资、生源、学生毕业去向、校园生活开销等等信息后再做决定。我们想知道这些因素是如何影响填报的选择，或者说是学校收到的申请数量？

和中国大陆不同的是，美国的学生可以同时投递多所高校，但其投递的偏好依旧反映了学校的特征。本课题研究采用开放的 1995 年美国高校数据集 college 尝试通过是否为私立学校、实际入学人数、本科生人数、食宿费用等等对其报名人数进行预测，并观察在预测过程中起重要作用的因素。

## 2. 探索性数据分析

### 2.1. 数据集介绍

college 数据集一共包含 18 列，共 777 条高校数据，记录了 1995 年 US News and World Report 中的多数美国高校数据。其每一列的标识符、意义和数据类型列表如下

表 1 college 数据集的各数据域名称及含义

数据域名称	含义	数据类型
Private	私立或公立学校	Literal["Yes", "No"]
Apps	收到的申请数量	int
Accept	录取的申请数量	int
Enroll	入学新生数量	int
Top10perc	高中排名前 10% 的新生百分比	int
Top25perc	高中排名前 25% 的新生百分比	int
F.Undergrad	全日制本科学生人数	int
P.Undergrad	非全日制本科学生人数	int
Outstate	外州学生学费	int
Room.Board	食宿费用	int
Books	预估书本费用	int
Personal	预估个人开销	int
PhD	拥有博士学位的教师比例	int
Terminal	拥有最高学位的教师比例	int
S.F.Ratio	师生比例	float
perc.alumni	捐赠校友比例	int
Expend	生均教学支出	int
Grad.Rate	毕业率	int

### 2.2. 重要统计量分析

首先将 Private 一列映射至 Bool 变量，然后对所有列计算均值、标准差、极值和四分位数，结果如下

表 2 college 数据集的各数据域名称及含义

数据域	均值	标准差	最小值	25%	中位数	75%	最大值
				分位数		分位数	
Private	0.72	0.44	0.0	0.0	1.0	1.0	1.0
Apps	3001.63	3870.20	81.0	776.0	1558.0	3624.0	48094.0
Accept	2018.80	2451.11	72.0	604.0	1110.0	2424.0	26330.0
Enroll	779.97	929.17	35.0	242.0	434.0	902.0	6392.0
Top10perc	27.55	17.64	1.0	15.0	23.0	35.0	96.0
Top25perc	55.79	19.80	9.0	41.0	54.0	69.0	100.0
F.Undergrad	3699.90	4850.42	139.0	992.0	1707.0	4005.0	31643.0
P.Undergrad	855.29	1522.43	1.0	95.0	353.0	967.0	21836.0
Outstate	10440.66	4023.01	2340.0	7320.0	9990.0	12925.0	21700.0
Room.Board	4357.52	1096.69	1780.0	3597.0	4200.0	5050.0	8124.0
Books	549.38	165.10	96.0	470.0	500.0	600.0	2340.0
Personal	1340.64	677.07	250.0	850.0	1200.0	1700.0	6800.0
PhD	72.66	16.32	8.0	62.0	75.0	85.0	103.0
Terminal	79.70	14.72	24.0	71.0	82.0	92.0	100.0
S.F.Ratio	14.08	3.95	2.5	11.5	13.6	16.5	39.8
perc.alumni	22.74	12.39	0.0	13.0	21.0	31.0	64.0
Expend	9660.17	5221.76	3186.0	6751.0	8377.0	10830.0	56233.0
Grad.Rate	65.44	17.11	10.0	53.0	65.0	78.0	100.0

下面对上表中的各特征做出简要分析。数据中“Private”变量为分类变量，其值可取是 (Yes) 或否 (No)。其中私立学校占绝大多数 (约为 72.7%)。在招生规模方面，平均每所大学收到 3001 份申请 (Apps)，录取 2018 人 (Accept)，最终有 779 人入学 (Enroll)。统计分析显示，不同大学的招生规模差异巨大，三个变量的标准差均接近甚至超过其平均值，例如申请数的最大值 (48,094) 与最小值 (81) 相差悬殊。通过计算可得，粗略的平均录取率约为 67.3%，平均入学率 (报到率) 约为 38.6%。此外，申请数的分布呈极右偏形态，其中位数 (1558) 远低于均值 (3001)，说明有少量大学拥有极其庞大的申请量，拉高了整体平均水平，而大部分大学的申请数集中在较低水平。

在生源背景上，平均有 27.6% 的新生来自高中排名前 10% (Top10perc)，55.8% 来自前 25% (Top25perc)。Top25perc 的分布相对对称，其中位数与均值接近。相比之下，Top10perc 的分布呈现轻微右偏，其中位数 (23%) 低于均值 (27.6%)，表明存在顶尖生源高度集中的大学。学生规模方面，平均全日制本科生 (F.Undergrad) 为 3699 人，非全日制本科生 (P.Undergrad) 为 855 人。这两类学生数量的分布同样差异极大且呈极右偏，例如全日制学生数量的中位数 (1707) 远低于其均值，说明少数大型大学主导了数据。师生比 (S.F.Ratio) 平均为 14.09，其分布相对集中，中位数 (13.6) 与均值接近，大部分学校的师生比集中在 11.5 至 16.5 之间。

各项费用中，外州学费 (Outstate) 平均为 10,441 美元，食宿费 (Room.Board) 为 4,358 美元，书本费 (Books) 为 549 美元，个人开销 (Personal) 为 1,341 美元，而生均教学支出 (Expend) 为 9,660 美元。外州学费和生均支出的差异非常显著，其标准差巨大，反映出大学间的资源投入和收费标准悬殊。相比之下，食宿费和书本费等基础生活成本的分布则相对稳定。Outstate 和 Expend 的分布明显右偏，意味着存在一部分高学费、高支出的精英大学。

师资队伍中，平均有 72.7% 的教师拥有博士学位 (PhD)，79.7% 拥有终极学位 (Terminal)。师资力量的分布在不同大学间相对均衡，大部分学校的教师博士学位比例介于 62% 到 85% 之间，

且拥有终极学位的教师比例普遍高于博士比例，这与常识相符。校友捐赠比例 (perc.alumni) 平均为 22.7%，但其分布较为分散，表明不同大学的校友捐赠文化和忠诚度差异很大。毕业率 (Grad.Rate) 平均为 65.5%，但数据中存在一个异常值 (118%)，该值超过 100%，需要后续处理。若剔除异常值影响，毕业率的分布大致对称，中位数与均值基本一致，但不同大学间的毕业率差异仍然不小（标准差为 17.2%）。

总而言之，数据通过上述几个统计量可见：

1. 数据集以私立学校为主 (72.7%)
2. 大学在规模和资源上存在明显的头部效应
3. 从申请到录取再到入学，数量大幅减少，平均入学率仅为 38.6%
5. 不同学校的师生比、师资博士比例、基础生活成本等指标差异不大

### 2.3. 相关矩阵

接下来对数据之间的相关性进行分析，计算两两特征之间的 Pearson 相关系数，得到如图 1 所示的相关矩阵。

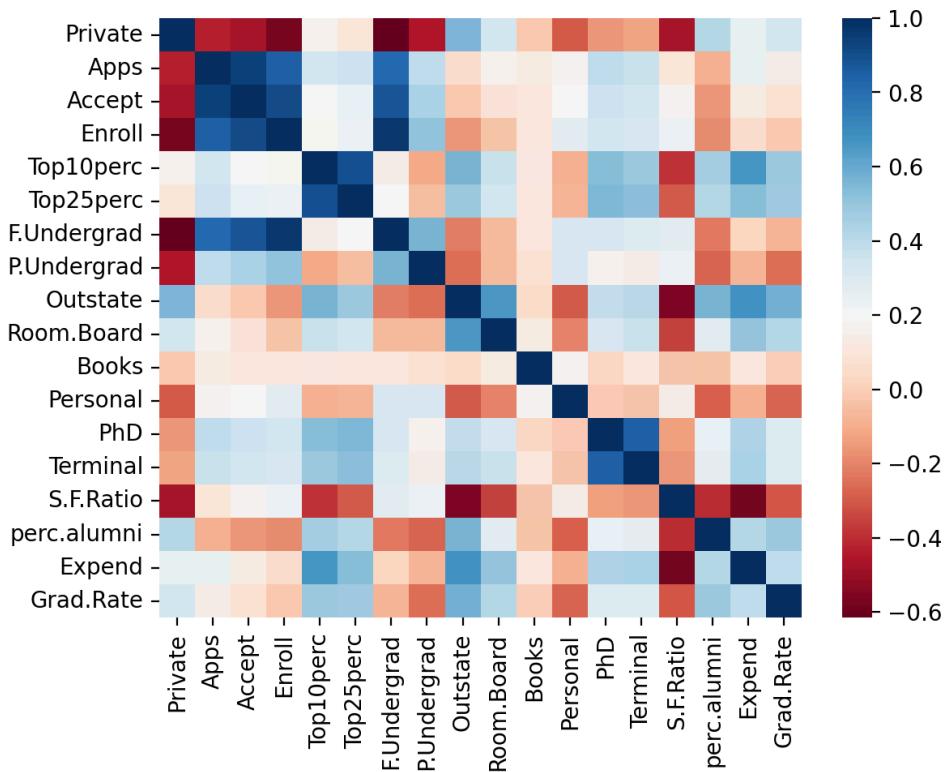


图 1 特征相关矩阵热图

其中第  $i$  行第  $j$  列的方格颜色表示第  $i$  个特征和第  $j$  个特征之间的相关系数。从整体来看，申请数 (Apps)、录取数 (Accept) 与入学数 (Enroll) 呈现显著的强正相关，颜色深度接近 1.0，这显示出“招生漏斗”的现象：申请数量多的学校通常录取和入学人数也较多。教师中博士比例 (PhD) 与终极学位比例 (Terminal) 的相关系数较高，这是因为“终极学位”通常包含博士学位。外州学费 (Outstate) 与生均教学支出 (Expend) 的正相关则体现了资源投入与收费水平的正向关联，印证了高学费、高支出的精英大学的存在。

在负相关关系中，师生比 (S.F.Ratio) 与教师博士比例 (PhD) 呈现约一定的负相关性，可能因为师生比较低的学校更注重小班教学和师资质量，从而博士比例更高。私立学校 (Private) 与师生比的负相关则表明私立学校倾向于维持更低的师生比。值得注意的是，书本费 (Books) 和个

人开销 (Personal) 与其他特征的相关性较弱，这与前面通过六值得到的“基础生活成本的分布相对稳定”这一结论呼应。

## 2.4. 公立学校和私立学校的部分特征数据分析

在初步的分析中，我们注意到仅存的一个二值特征为学校是否为私立 (Private)，因此可以将私立和公立学校分开，分别观察它们的各指标的分布差异。其结果如下图所示。

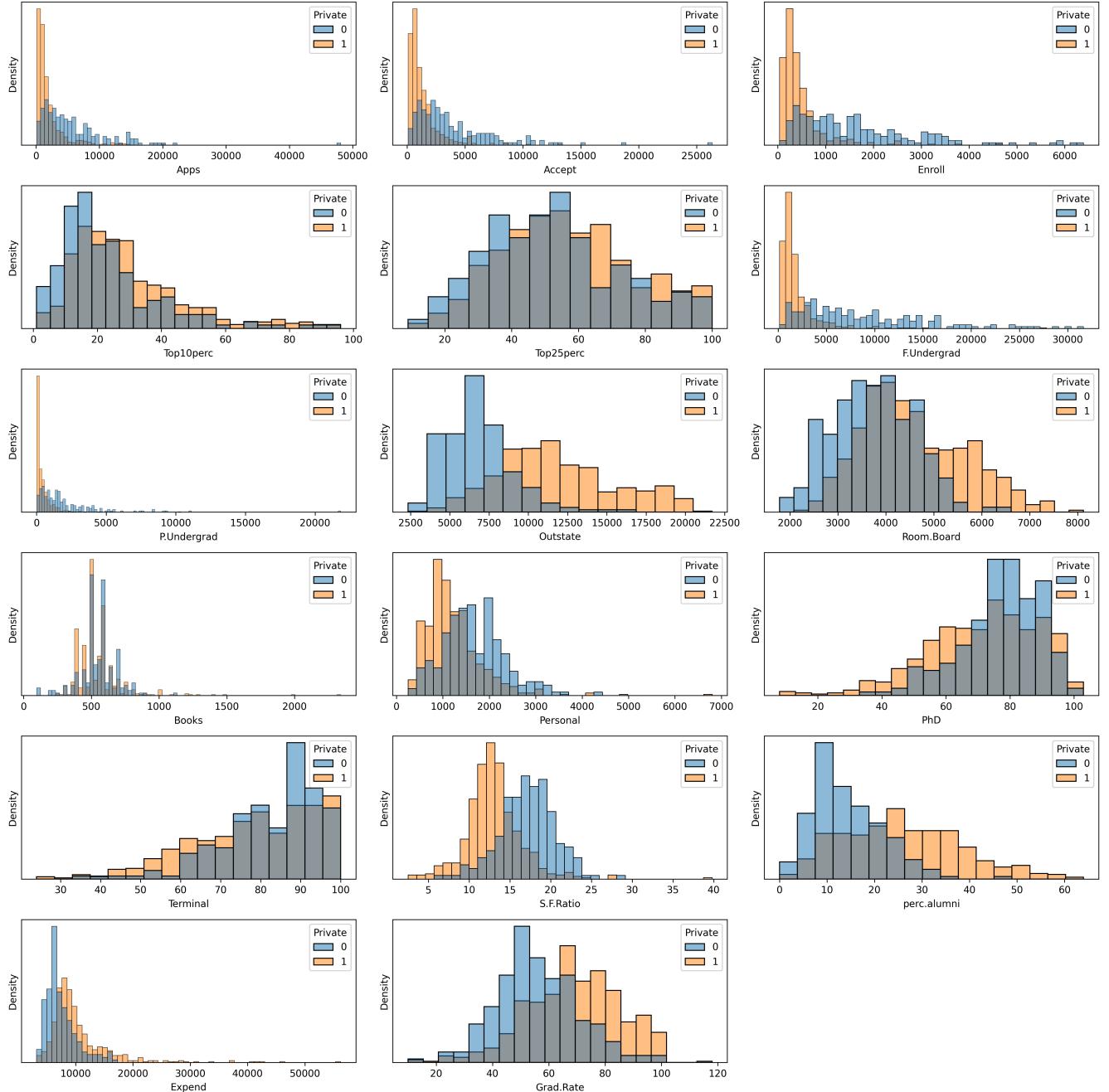


图 2 公立学校和私立学校的在各方面特征分布直方图的对比，蓝色表示公立学校，橙色表示私立学校

可以看到，公立学校和私立学校中来自中学中前 10% 或是前 25% 学生的占比分布、教师中终极学位比例、学校的生均开销和学生的书本开销差距不大，但申请相关指标（包括申请、录取、入学人数）、全日制和非全日制本科生人数中，私立学校都比公立学校对应的特征值少且分布集中。另一方面，私立学校学生的学费和食宿开支 (Outstate 的约一万至两万美元和 Room.Board 的约

三千美元至约七千美元)相比公立学校更高, 毕业率(70%左右)和捐款校友比例(约为15%~45%)也更高; 而学生的预估个人开销则更低(约一千美元)。

最后, 我们选取上图中公立学校和私立学校分布差异较大的几个特征: 来自中学中前10%的学生占比、校友捐赠比例、外州学生学费、学生的食宿开支和申请人数, 绘制了二维分布散点图和核密度估计的等高线图, 如图3所示, 为对角线上直方图的可视化效果, 我对一些特征进行了缩放。

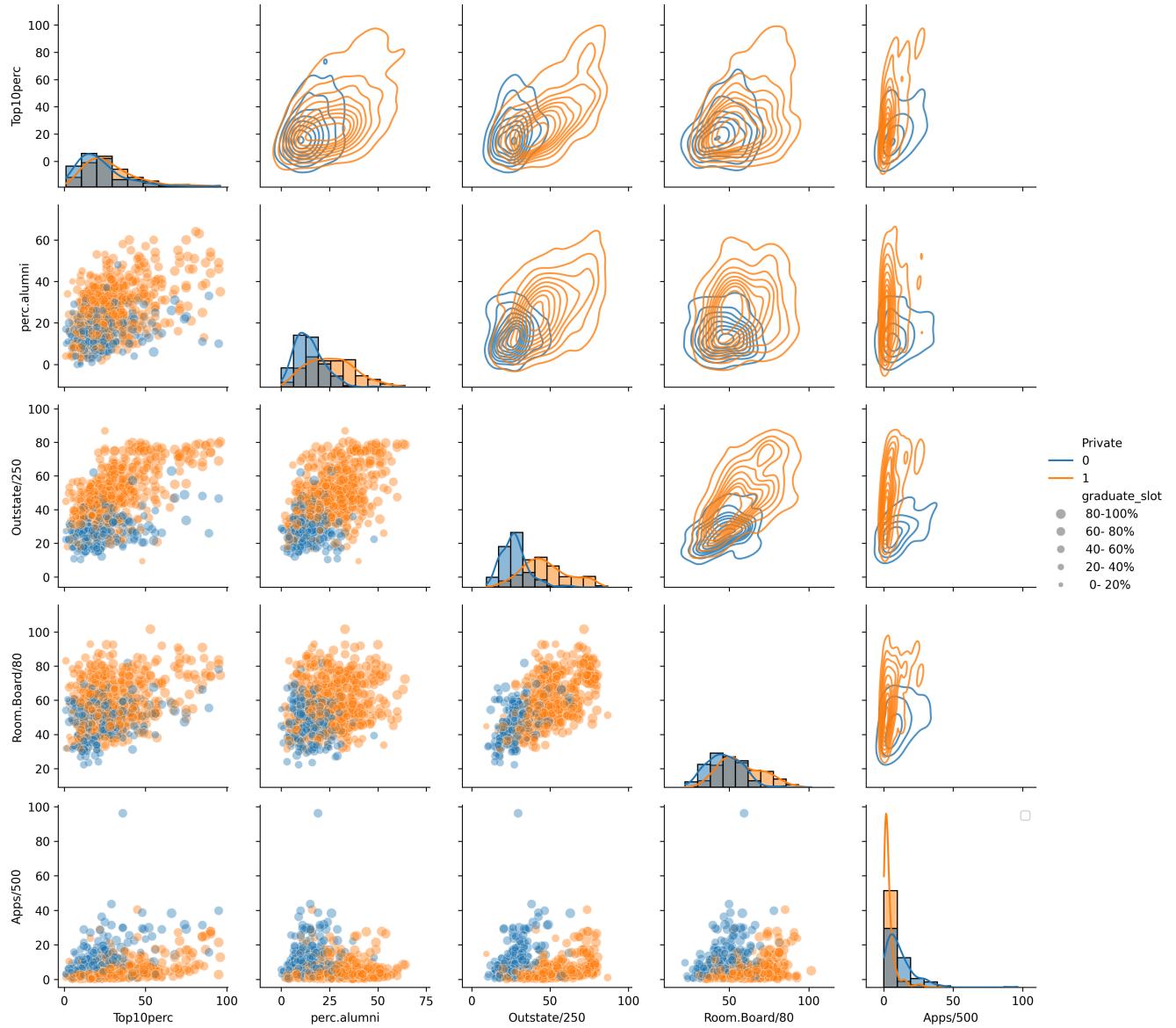


图3 公立学校和私立学校的部分特征的二维分布散点图, 蓝色表示公立学校, 橙色表示私立学校

第二行第一列、第三行第一列和第二列和第四行的第一列和第三列的散点图可以看出线性趋势: 录取中学前十之一的学生的学校校友捐赠比例更高、外州学生学费更高、学生们的生活开销也更高。对于公立学校的申请人数和前十之一学生比例、校友捐赠比例、外州学生学费和生活开销正相关, 而私立学校的申请人数则与这些指标的关系不大。

## 2.5. 降维可视化

最后我们尝试对这十余个指标通过PCA和t-SNE算法降维至二维, 并将这两个算法得到的降维结果绘制在二维平面, 如图4所示, 其中横纵轴分别表示降维后在抽象空间中的坐标, 难以解释

实际含义：并用不同颜色标注对应特征点（对应着某所学校）的申请人数的对数值 (logApps)：颜色越接近红色，说明申请人数越低；颜色越接近蓝色，说明申请人数越高。

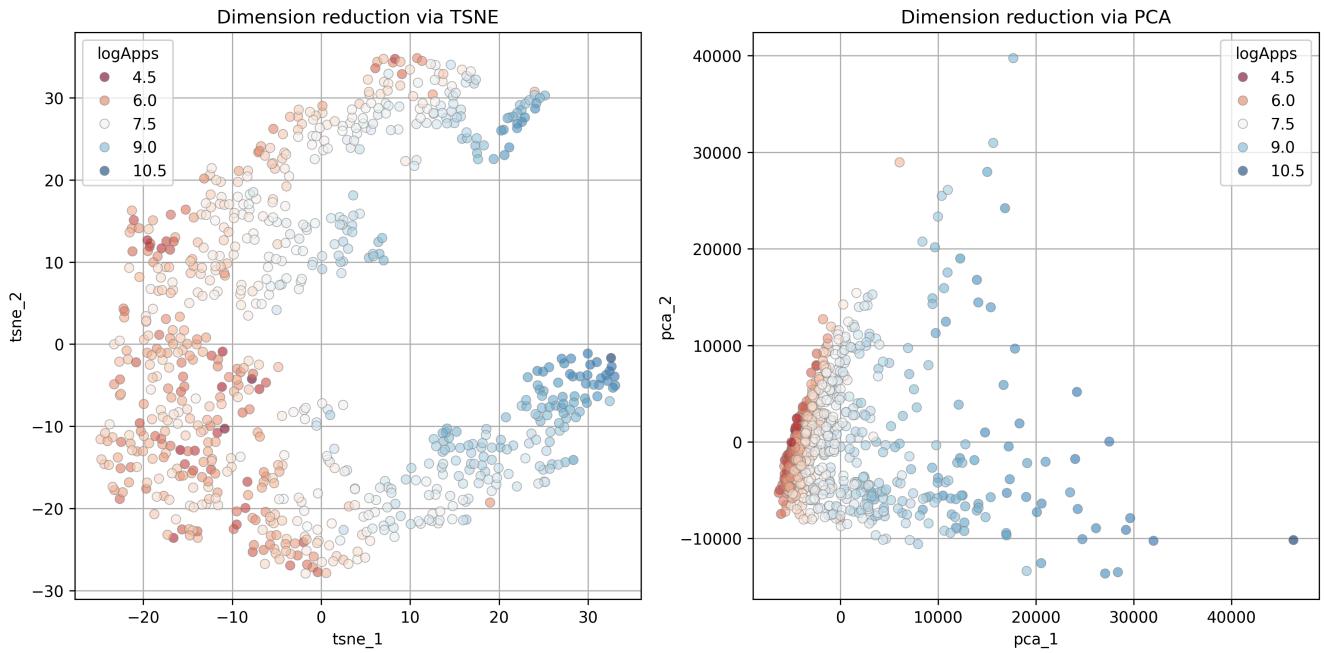


图 4 college 数据集在 t-SNE (左) 和 PCA (右) 两个降维算法下的降维结果

通过该图可以看到，高申请人数学校降维后的特征点大多位于图的右侧，低申请人数学校聚集于左侧；另外我们通过降维可以猜测，低申请人数的学校可能其特征更为相近，而高申请人数的名校也许各有特色，因为降维后的特征点分布较为分散。

## 2.6. 正态性检验

我们对数据的每个特征列都做 K-S 检验，得到所有的  $p$  值都为零：这说明所有列都不遵从正态分布。

### 3. 方法和实验

#### 3.1. 数据预处理

通过小节 2.2 可以看到，数据中各域的数量级差距悬殊，这会导致在线性回归等训练过程中权重参数被压制接近于 0。于是可以统一将所有数据域放缩至  $[-1, 1]$  这个区间。实现这个要求可以使用 `sklearn` 模块的 `MaxAbsScalar`，其原理为该数据域的数据统一除以最大的绝对值，即

$$x'_i \leftarrow \frac{x_i}{\max_i(x_i)}. \quad (1)$$

这样我们就得到了一系列数量级相同的归一化特征。归一化后对数据集进行的随机切分，为训练集占 70%，验证集占 30%。至于回归的目标，综合考虑先前分析得到的申请数量的分布，我们将申请人数取对数，得到对数申请人数 (`logApp`)，并将其作为回归目标。

#### 3.2. 线性回归及其变体

下面简要介绍所用的线性回归基线方法族。线性回归可以用直线拟合、向数据矩阵  $\mathbf{X}$  的列空间做正交投影、极大似然估计等视角进行理解和解释。记数据集为  $D = \{\mathbf{x}_{\{n\}}, y_n\}_{n=1}^N$ ，定义对标签的预测函数为下面的线性形式

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \cdots + w_D x_D = \mathbf{w}^\top \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix} \quad (2)$$

其中  $\mathbf{x} \in \mathbb{R}^D$ ,  $\mathbf{w} \in \mathbb{R}^{D+1}$ . 这是线性回归的基本形式。若将特征和对应的标签堆叠起来，即

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_N^\top \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \quad (3)$$

则线性回归的预测结果为  $\hat{\mathbf{y}} = \mathbf{X}\omega$ ，我们要求预测结果  $\hat{\mathbf{y}}$  距离真实目标  $\mathbf{y}$  越近越好，这就得到了线性回归的目标函数：

$$J(\mathbf{w}) = \frac{1}{2N} \sum_{n=1}^N (y_n - \hat{y}_n)^2 = \frac{1}{2N} \|\hat{\mathbf{y}} - \mathbf{y}\|_{[2^2]}^2 = \frac{1}{2N} \|\mathbf{X}\omega - \mathbf{y}\|_2^2, \quad (4)$$

这是一个拥有光滑凸目标函数的优化问题，若  $\mathbf{X}^\top \mathbf{X}$  可逆，令  $J(\omega)$  的梯度为零，我们能得到其解析解

$$\omega^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}. \quad (5)$$

实际操作中，我们会在目标  $J(\omega)$  中加入正则项  $R(\omega)$ ，通过对权重向量进行软限制的方式防止过拟合。修改过后的目标函数为

$$J(\omega) = \frac{1}{2N} \|\mathbf{X}\omega - \mathbf{y}\|_2^2 + \lambda R(\omega), \quad (6)$$

其中  $\lambda \geq 0$  是需要人为给定的权重超参数。常用的正则项可以是权重向量的 L2 范数（这将得到岭回归）或 L1 范数（这将得到 LASSO 回归）。若选取的 L2 范数，问题总是存在整洁的解析解：

$$\omega^* = (\mathbf{X}^\top \mathbf{X} + \lambda I)^{-1} \mathbf{X}^\top \mathbf{y}. \quad (7)$$

从计算上看，这会使得括号中的矩阵可逆，此时该模型总是有解析解。

ElasticNet 方法结合了岭回归和 LASSO 回归，其目标函数为

$$J(\omega) = \frac{1}{2N} \|X\omega - y\|_2^2 + \lambda[\alpha\|\omega\|_1 + (1-\alpha)\|\omega\|_2^2]. \quad (8)$$

我们使用标准线性回归、岭回归、LASSO 回归 和 ElasticNet 回归这四种方法为基准方法，使用其余特征预测对数申请人数。对于标准线性回归，允许其拟合截距项 (interception)；对岭回归，设置正则项权重为  $\alpha = 0.01$ ；对 LASSO 回归，设置正则项权重为  $\alpha = 0.001$ ，对 ElasticNet，设置正则项权重为  $\alpha = 0.001$ ，且设置 L1 和 L2 正则项的权重相同 (`l1_ratio=0.5`)。其他为 sklearn 中的默认参数，详见附录。

四种基准模型在测试集的分数如表 3

	模型	$R^2$	均方损失 (MSE)
	LinearRegression	0.75	0.30
	Ridge	0.75	0.30
	Lasso	0.74	0.31
	ElasticNet	0.74	0.31

可见四个基准模型的预测损失 MSE 在 0.3 左右， $R^2$  分数在 0.75 左右，这说明，一定程度上线性模型可以拟合数据的大致趋势。我们进一步观察其拟合得到的权重和截距值，得到图 5。

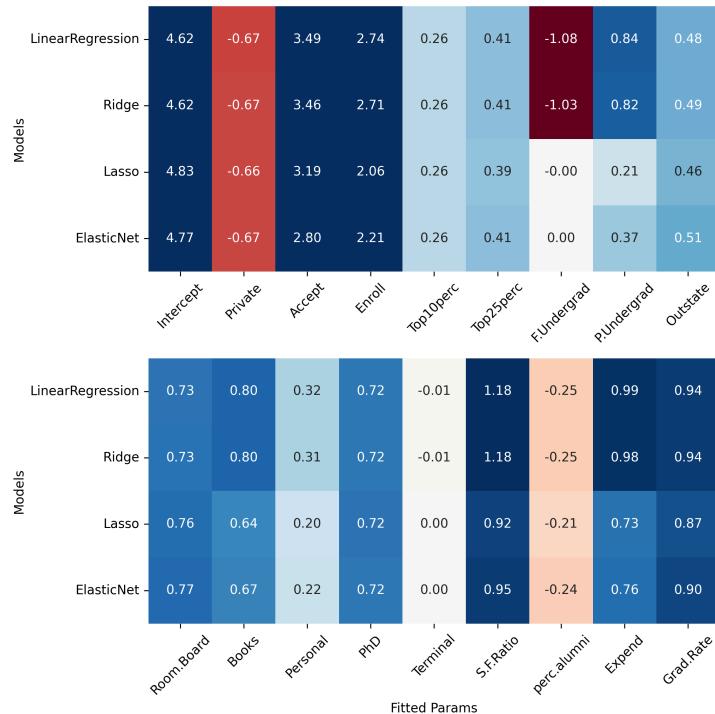


图 5 四种线性基线模型在训练后对应于各特征的权重和截距项 (Intercept)

其中的颜色反映系数正负及大小：颜色越深说明权重绝对值越大，颜色越浅说明权重绝对值越小；蓝色对应正值，红色对应负值。通过该图可见，四种线性基线模型在训练集上学习得到的大部分权重相差不大。由于 LASSO 和 ElasticNet 中含有 L1 范数，这会导致学习得到的权重更加稀疏。如对应于 F.Undergrad、P.Undergrad 等特征的权重。除此之外，我们可以看出四种模型对学校是否为私立学校、录取率、入学率、师生比例、生均开支和毕业率有大权重，其中录取率的权重为除去偏置项后最大，这一定程度上可以反应它和择校策略之间的联系。录取率越高、毕业率、生均开销、教师 PhD 占比、师生比例越高的学校，对应申请人数越多。

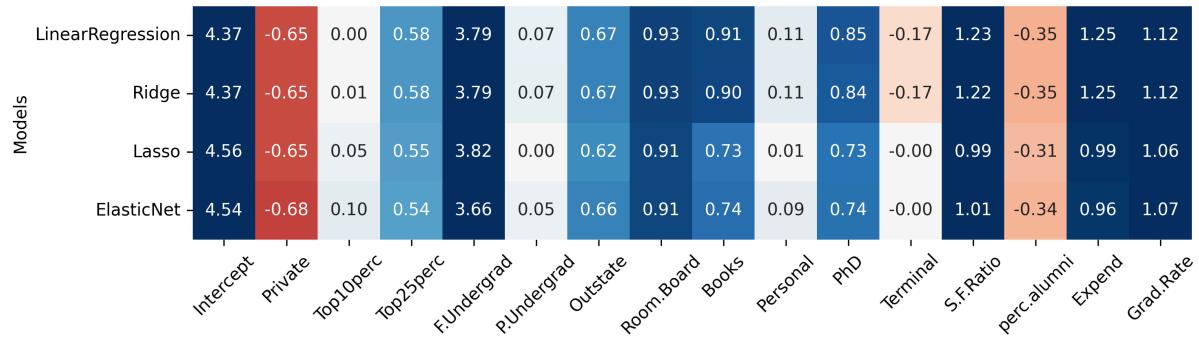


图 6 四种线性基线模型在训练后对应于各特征的权重和截距项 (Intercept)

如果避免上述倒果为因的问题，将 Accept 和 Enroll 两列删去，得到的训练结果如图所示，四个模型的  $R^2$  分数均为 0.68，MSE 分数均为 0.39。可以看到除了被删去的两列外，大体上贡献较大的特征和删除之前相同，即是否为私立、师生比、生均花费和毕业率。

### 3.3. 决策树

最后我们考虑使用决策树算法对同样的数据集进行预测。和分类决策树类似，决策树对节点中的数据按照最优划分条件递归划分，直至满足停机条件。回归决策树的预测方法为将叶节点对应的训练集的目标值的平均作为该叶节点的预测目标值。我们同样采用 sklearn 中的 DecisionTreeRegressor 类进行训练，并约束其分裂指标为平方损失 (squared\_error)、最大深度为 2、最小可分裂数据量为 10，以保证模型的泛化性能。该模型在测试集上达到 0.85 的  $R^2$  分数和 0.178 的 MSE 分数，相比于线性回归基准模型低，效果较回归模型更好。

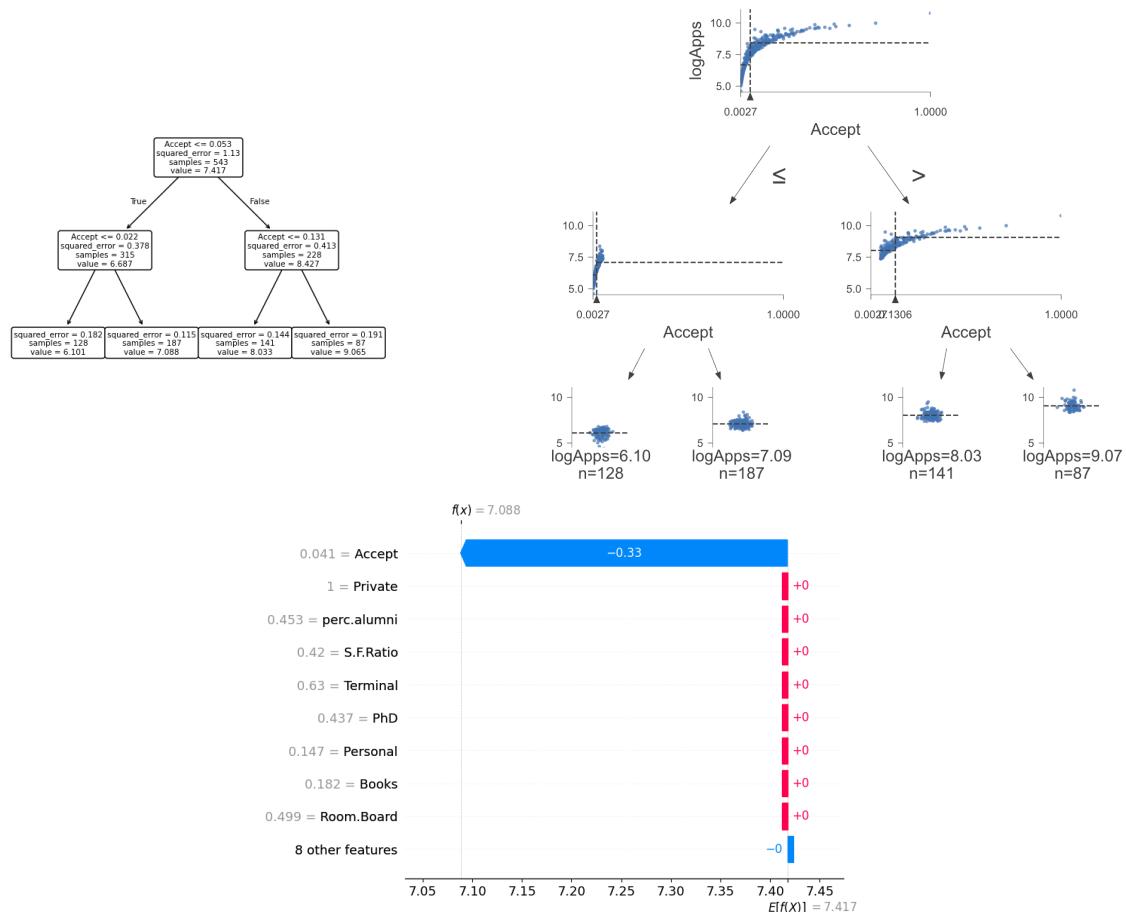


图 7 决策树模型的可视化结果（上）和各特征的 Shap 值（右）

我们对决策树进行可视化，并计算各特征的 Shap 值。Shap 值是一种衡量特征在模型中重要性的指标，

如图 7 所示。可见 Accept 一列显然对整个训练数据造成了某种污染，因为高校的录取率一般不变，因此与填报率相关。如果像线性模型那样去除相关的两列，并使用 optuna 模块搜索最优参数，得到的模型在测试集上可以达到 0.28 的 MSE 值，此时 F.Undergrad 拥有最高的 Shap 值。

## 4. 结论和讨论

通过数据分析和建模预测可以看出，各个学校的申请人数和学校的实力、师资等等有重要关系。申请者同时也关注在学校的开销、学校倾注在学生身上的花费，以及师生比，这可以看出学校对学生的重视程度。申请者还看重毕业率，这预示着他们将来成功从这所大学毕业拿到文凭的概率。

由于本数据的记录时间很早，到现在已有近三十年，上述结果可能不足以说明当今世界的高校申请形势。同时在分析中遇到的诸多疑似“倒果为因”的现象，使对相关数据的因果联系的建模与发现变得必要。

## 5. 附录

```
# % [markdown],  
# ## 0 读取数据  
  
# 我选择的是 `college` 这个数据集。该数据集包含摘自《美国新闻与世界报道》1995年刊的美国高校统计数  
据：  
  
# | 数据域名称 | 含义 |  
# |-----|-----|  
# | `Private` | 二分变量，以 "No" 和 "Yes" 标识私立或公立学校 |  
# | `Apps` | 收到的申请数量 |  
# | `Accept` | 录取的申请数量 |  
# | `Enroll` | 入学新生数量 |  
# | `Top10perc` | 高中排名前 10% 的新生百分比 |  
# | `Top25perc` | 高中排名前 25% 的新生百分比 |  
# | `F.Undergrad` | 全日制本科学生人数 |  
# | `P.Undergrad` | 非全日制本科学生人数 |  
# | `Outstate` | 外州学生学费 |  
# | `Room.Board` | 食宿费用 |  
# | `Books` | 预估书本费用 |  
# | `Personal` | 预估个人开销 |  
# | `PhD` | 拥有博士学位的教师比例 |  
# | `Terminal` | 拥有最高学位的教师比例 |  
# | `S.F.Ratio` | 师生比例 |  
# | `perc.alumni` | 捐赠校友比例 |  
# | `Expend` | 生均教学支出 |  
# | `Grad.Rate` | 毕业率 |  
  
# 我们希望了解其他特征域的值对毕业率的影响。  
  
# %%  
import pandas as pd # type: ignoreselect  
college = pd.read_csv('college.csv')  
  
# %%  
college.head()  
  
# % [markdown],  
# 将第一列的列名改为 `College`  
  
# %%  
college.rename({'Unnamed: 0': 'College'}, axis=1, inplace=True)  
college.set_index('College', inplace=True)  
college.head()  
  
# % [markdown],  
# 将 `Private` 字段映射为 `bool` 值变量  
  
# %%  
college['Private'] = college['Private'].map({'Yes': 1, 'No': 0})  
college.head()  
  
# % [markdown],  
# 使用 `DataFrame.describe()` 函数获取每个特征域的各统计量
```

```
# %%
college.describe(include= "all")

# %% [markdown] ,
# ## 1 探索性数据分析

# %% [markdown] ,
# (d) Use the `pd.plotting.scatter_matrix()` function to produce a
# scatterplot matrix of the first columns `[Top10perc, Apps, Enroll]`.
# Recall that you can reference a list `C` of columns of a data frame
# `A` using `A[C]`.
# %%
pd.plotting.scatter_matrix(college[
    "Top10perc",
    "Apps",
    "Enroll",
    "Grad.Rate"
]);

# %%
college.corr()

# %%
college.corr()["Grad.Rate"].abs() > [0.4],

# %%
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# %%
plt.figure(dpi=200)
sns.heatmap(college.corr(), ax=plt.gca(), cmap="RdBu")

# %%
from typing import List
from matplotlib.axes import Axes

fig, axs = plt.subplots(6, 3, figsize=(16, 16), dpi=300)
axs: List[Axes] = axs.flatten()

for (idx, col) in enumerate(college.columns[1:]):
    sns.histplot(data=college, x=col, hue='Private', stat='density',
common_norm=False, ax=axs[idx])
    axs[idx].set_yticks([])
    # if college[col].max() > [100] * college[col].min():
    #     axs[idx].semilogy()
axs[-1].axis("off")

plt.tight_layout()

# %%
def clip_graduate_rate(a: int):
    if a < 0:
        return 0
```

```
    elif a > 100:
        return [100],
    else:
        return a

# %%
college["Grad.Rate"] = college["Grad.Rate"].apply(clip_graduate_rate)

# %%
def classify_graduate_slot(a: int) -> str:
    if a > 80:
        type_ = 4
    elif a > 60:
        type_ = 3
    elif a > 40:
        type_ = 2
    elif a > 20:
        type_ = 1
    else:
        type_ = 0
    return f"{{(type_)} * 20:{>3d}}-{{(type_+1)} * 20:{>3d}}%"

# %%
list(college["Grad.Rate"])

# %%
college["graduate_slot"] = college["Grad.Rate"].apply(classify_graduate_slot)

# %%
set(college["graduate_slot"])

# %%
# Top10perc      True
# Top25perc      True
# Outstate       True
# Room.Board     True
# perc.alumni    True
# Grad.Rate      True

# %%
college_distilled = college[["Private", "Top10perc", "Outstate", "Room.Board",
"perc.alumni", "graduate_slot", "Apps"]],
# %%
college_distilled["Outstate/250"] = college_distilled["Outstate"] / [250],
college_distilled["Room.Board/80"] = college_distilled["Room.Board"] / 80
college_distilled["Apps/500"] = college_distilled["Apps"] / [500],

college_distilled.drop(columns=["Outstate", "Room.Board", "Apps"], inplace=True)

# %%
from collections import Counter

Counter(college_distilled["Private"])

# %%
```

```
weights = college_distilled["Private"].map({1: 0.273, 0:0.727})

# %%
from functools import partial
import warnings

warnings.filterwarnings("ignore")

g = sns.PairGrid(college_distilled, hue="Private");

plt.legend()

plt.gcf().set_dpi(300)

g.map_diag(partial(sns.histplot, stat="density", weights=weights, kde=True));
g.map_lower(sns.scatterplot,
            size=college_distilled["graduate_slot"],
            size_order=[' 0- 20%', ' 20- 40%', ' 40- 60%', ' 60- 80%', ' 80-100%']
           [::-1],
            alpha=0.4);
g.map_upper(sns.kdeplot, alpha=0.8);
g.add_legend(title="", adjust_subtitles=True);

# %% [markdown],
# # 正态性检验
#

# %%
from scipy.stats import kstest

for col in college.columns[:-1]:
    kstest_result = kstest(college[col], cdf="norm")
    print(f"[ {col} ], [{kstest_result.statistic:.2f}],
[ {kstest_result.pvalue:.4f} ], ")

# %% [markdown],
# # 多重共线性检测

# %%
for c in college.columns:
    print(college[c].dtype)

# %%
# perform VIF

from statsmodels.stats.outliers_influence import variance_inflation_factor

def get_vif(df: pd.DataFrame) -> pd.DataFrame:
    vif_ = pd.DataFrame()
    vif_[‘index’] = df.columns
    vif_[‘VIF’] = [variance_inflation_factor(df.values,i) for i in
range(df.shape[1])],
    return vif_

college_dvif = college.drop(columns=["Apps",
"graduate_slot"]).copy(deep=True).astype(float)
```

```
vif = get_vif(college_dvif)
vif

# one can see that there are in the feature columns there are features have high VIF
values

# %%
ls = [],
while True:
    vif =
get_vif(college_dvif.drop(columns=list(set(ls).intersection(college_dvif.columns))))
    ls = list(vif[vif["VIF"] > 20]["index"])

    print(vif, end="\n\n")

    if len(ls) == 0:
        break

# %% [markdown],
# # 多元线性回归

# %%
from sklearn.model_selection import train_test_split

college["logApps"] = np.log(college["Apps"] + 1e-5)

X, y = college.drop(columns=["graduate_slot", "Apps", "logApps"]),
college["logApps"],
X_columns = X.columns

# %%
# normalize fields
from sklearn.preprocessing import MaxAbsScaler

max_abs_scalar = MaxAbsScaler()

X = max_abs_scalar.fit_transform(X)
print(*max_abs_scalar.max_abs_, sep='\t')

# %%
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# %%
from sklearn.linear_model import LinearRegression, Ridge, Lasso, ElasticNet
from sklearn.metrics import r2_score, mean_squared_error

models: LinearRegression|Ridge|Lasso|ElasticNet = [
    LinearRegression(),
    Ridge(alpha=1e-2),
    Lasso(alpha=1e-3),
    ElasticNet(alpha=1e-3, l1_ratio=0.5),
],
for model in models:
    model = model.fit(X_train, y_train)
    yhat = model.predict(X_test)
```

```
r2, mse = r2_score(y_test, yhat), mean_squared_error(y_test, yhat)

print(f"{model.__class__.__name__:>20}, {r2:.2f}, {mse:.2f}")

# %%
[model.intercept_] + list(model.coef_)

# %%
plt.figure(dpi=300, figsize=(8, 8))

ax = plt.subplot(2, 1, 1)
sns.heatmap(np.stack([[i.intercept_] + list(i.coef_)[8:] for i in models]),
            annot=True, fmt=".2f", cmap="RdBu", vmin=-1, vmax=1, cbar=False, ax=ax)
ax.set_xticks(np.arange(len(X_columns)-8)+0.5, ["Intercept"] + list(X_columns)[8:], rotation=45)
ax.set_ylabel("Models")
ax.set_yticks(np.arange(4)+0.5, [i.__class__.__name__ for i in models], rotation=0)

ax=plt.subplot(2, 1, 2)
sns.heatmap(np.stack([list(i.coef_[8:]) for i in models]), annot=True, fmt=".2f",
            cmap="RdBu", vmin=-1, vmax=1, cbar=False, ax=ax)
ax.set_xticks(np.arange(len(X_columns)-8)+0.5, list(X_columns)[8:], rotation=45)
ax.set_ylabel("Models")
ax.set_xlabel("Fitted Params")
ax.set_yticks(np.arange(4)+0.5, [i.__class__.__name__ for i in models], rotation=0)

plt.tight_layout()

# %%
# Tree-based and Random forest with parameter search

from sklearn.tree import DecisionTreeRegressor, plot_tree

reg = DecisionTreeRegressor(criterion="squared_error", max_depth=2,
                           min_samples_split=10)

reg.fit(X_train, y_train)
yhat = reg.predict(X_test)

print(r2_score(y_test, yhat), mean_squared_error(y_test, yhat))

# %%
# search paramms
import optuna

def objective(trial: optuna.trial.Trial):
    params = {
        # 固定项
        'criterion': 'squared_error',           # 也可放进搜索空间
        'splitter': 'best',                     # 也可放进搜索空间
        'random_state': 42,
        # Optuna 搜索空间
        'max_depth': trial.suggest_int('max_depth', 2, 3),
        'min_samples_split': trial.suggest_int('min_samples_split', 2, 30),
```

```
'min_samples_leaf': trial.suggest_int('min_samples_leaf', 1, 20),
'min_weight_fraction_leaf': trial.suggest_float('min_weight_fraction_leaf',
0.0, 0.5, step=0.05),
'max_features': trial.suggest_categorical(
    'max_features', ['sqrt', 'log2', None],
),
'max_leaf_nodes': trial.suggest_int('max_leaf_nodes', 10, 1000, log=True),
'min_impurity_decrease': trial.suggest_float('min_impurity_decrease', 0.0,
0.1, step=0.001),
'ccp_alpha': trial.suggest_float('ccp_alpha', 0.0, 0.05, step=0.001),
}

model = DecisionTreeRegressor(**params)
model.fit(X_train, y_train)

yhat = model.predict(X_test)

return mean_squared_error(y_test, yhat)

study = optuna.create_study(direction="minimize")
study.optimize(objective, n_trials=1500, show_progress_bar=True)

# %%
reg = DecisionTreeRegressor(**study.best_params)
reg.fit(X_train, y_train)

# %%

# %%
from dtreeviz import model

viz = model(reg, X_train=pd.DataFrame(X_train, columns=X_columns), y_train=y_train,
            target_name='logApps',
            feature_names=X_columns
        ) # 中文

viz.view(scale=2)

# %%
plot_tree(reg, rounded=True, feature_names=X_columns)

# %%
import shap

explainer = shap.TreeExplainer(reg)

# %%
shap_values = explainer(pd.DataFrame(X_train, columns=X_columns))

# %%
shap.plots.waterfall(shap_values[0])

# %%
shap.plots.beeswarm(shap_values)
```

```
# %%
from sklearn.manifold import TSNE

X_tsne = TSNE(n_components=2).fit_transform(X)

# %%
from sklearn.decomposition import PCA

X_pca = PCA(n_components=2).fit_transform(X)

# %%
college[["tsne_1", "tsne_2"]] = X_tsne
college[["pca_1", "pca_2"]] = X_pca

# %%
plt.figure(dpi=300, figsize=(12, 6))

ax = plt.subplot(1, 2, 1)
sns.scatterplot(data=college, x="tsne_1", y="tsne_2", hue="logApps", palette="RdBu",
edgecolor="gray", alpha=0.6, ax=ax, zorder=10)
ax.grid(zorder=0)
ax.set_title("Dimension reduction via TSNE")

ax = plt.subplot(1, 2, 2)
sns.scatterplot(data=college, x="pca_1", y="pca_2", hue="logApps", palette="RdBu",
edgecolor="gray", alpha=0.6, ax=ax, zorder=10)
ax.grid(zorder=0)
ax.set_title("Dimension reduction via PCA")

plt.tight_layout()

# %%
```