这蘑菇有毒！

蒋文馨　16342067　jiangwx7@mail2.sysu.edu.cn
中山大学数学学院 17 级统计学

日期：2020 年 7 月 7 日

摘　要

本文基于 UCI 蘑菇数据集，建立判别蘑菇是否可食用的分类器。数据集样本量为 8124，其中可食用和不可食用分别为 4208 和 3916，自变量有 22 个，均为因子型变量。首先对数据进行详细的探索性分析。对于缺失值，尝试决策树、多重填补和 kNN 三种填补方法，最终选用 kNN 对训练集和测试集分别进行填补。之后，根据模型需求，将因子型变量编码为哑变量。使用主成分回归、线性判别分析、LASSO 回归、逐步回归、决策树（CART、C4.5、C5.0）、随机森林、XGBoost、kNN、SVM 、NN、RIPPER 和 PART，共 14 种模型建立了性能优秀的分类器，并基于树模型给出了特征的重要性。

关键词： Mushroom　缺失值　LASSO 回归　随机森林　XGBoost　NN

目录

0 引言

每年都有很多人因为摄入有毒野生蘑菇生病，甚至死亡[1]。由于许多蘑菇在外观上非常相似，所以即使是有经验的蘑菇采集者也会中毒。百度百科上鉴别有毒蘑菇的方法为[2]：

1. 观形状，毒蘑菇般比较黏滑，菌盖上常黏些杂物，或生长一些像补丁状的斑块，菌柄上常有菌环; 无毒蘑菇很少有菌环。
2. 察色味，有毒磨菇的颜色比较浓艳，菌伞带有红、紫、黄或其他杂色斑点，基底红色，形状异常; 无毒蘑菇为苦杏或水果味。毒蘑菇有土豆或萝卜味，而且有辛辣、恶臭和苦味。
3. ...

本文使用 UCI 机器学习库的 Mushroom 数据集，完成以下两个任务：

- 建立判断野生蘑菇是否可食用的分类器；
- 探究蘑菇是否可食用与哪些特征相关。

需要注意的是，关于毒蘑菇的识别，迄今为止还没有找到一种既科学可靠，又简单易行的具体适用方法[3]。所以，为确保安全，**不建议食用野生蘑菇**。

1 数据说明

本文所用数据集属于 UCI 机器学习库的 Mushroom 数据集[1]。本文使用 Kaggle 版数据[2]，网址为 https://www.kaggle.com/uciml/mushroom-classification。该数据集包括了列于《Audubon Society Field Guide to North American Mushrooms (1981)》上的 23 个带菌褶的磨菇品种的 8124 个磨菇案例信息。在食用指南中，每种蘑菇被鉴定为肯定可以食用”，“肯定是有毒的”和“可能有毒，不建议食用”。“可能有毒，不建议食用”和“肯定是有毒的”合并到一起，形成最终的两个类：不可食用（poisonous）和可食用（edible），分别有 4208 和 3916 个样本。

因数据集只记录了北美 Agaricus 和 Lepiota 科的蘑菇信息，许多常见蘑菇种类及特征未包含在数据集中，所以不建议将本文所得模型用于判断其它地区或其它科蘑菇是否可食用。

2 探索性分析

首先，对数据集进行概述。然后，再分别介绍每一个特征。

0. 数据概览：

   总样本数为 8124。特征包括 1 个因变量和 22 个自变量，均为名义型变量，在 R 中储存为因子型变量。缺失值只存在于自变量菌托（stalk.root）中，用“?”表示。缺失情况说明见12，对缺失值的处理见 3.4。变量的中英文对照及取值见表1，示意图如图1所示。表格中的取值为可能取值的简写，实际取值是可能取值的子集，例如菌褶剖面 4 种可能取值为 a,d,f,n，但数据集中菌褶剖面只有 a,f 这 2 种取值。菌幕性状包括菌托和菌环的情况。

1. 类别：

[1]也是 XGBoost 安装包中的 demo 数据集
[2]Kaggle 版的优势在于拥有表头

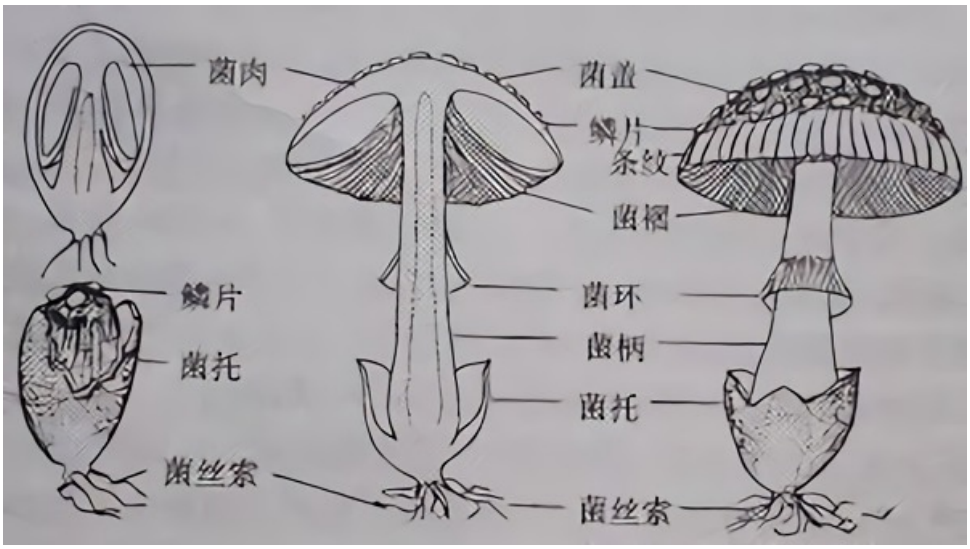


图 1: 蘑菇结构示意图

表 1: 数据概览

序号	英文名	中文名	取值
1	class	类别	e,p
2	cap.shape	菌盖形状	b,c,x,f,k,s
3	cap.surface	菌盖表面特征	f,g,y,s
4	cap.color	菌盖颜色	n,b,c,g,r,p,u,e,w,y
5	bruises	擦伤反应	t,f
6	odor	气味	a,l,c,y,f,m,n,p,s
7	gill.attachment	菌褶剖面	a,d,f,n
8	gill.spacing	菌褶间隔	c,w,d
9	gill.size	菌褶大小	b,n
10	gill.color	菌褶颜色	k,n,b,h,g,r,o,p,u,e,w,y
11	stalk.shape	菌柄形状	e,t
12	stalk.root	菌托	b,c,u,e,z,r,?
13	stalk.surface.above.ring	环上柄面	f,y,k,s
14	stalk.surface.below.ring	环下柄面	f,y,k,s
15	stalk.color.above.ring	环上柄色	n,b,c,g,o,p,e,w,y
16	stalk.color.below.ring	环下柄色	n,b,c,g,o,p,e,w,y
17	veil.type	菌幕类型	p,u
18	veil.color	菌幕颜色	n,o,w,y
19	ring.number	菌环数量	n,o,t
20	ring.type	菌环类型	c,e,f,l,n,p,s,z
21	spore.print.color	孢子印颜色	k,n,b,h,r,o,u,w,y
22	population	生长分布	a,c,n,s,v,y
23	habitat	生长环境	g,l,m,p,u,w,d

类别变量为分析的目标变量，有 2 种取值，分别为：可食用（edible=e），样本数为 4208 个，占比为 51.8%；不可食用（poisonous=p），样本数为 3916 个，占比为 48.2%，如图2所示。可以看出两种类型数目大致相当，数据平衡。

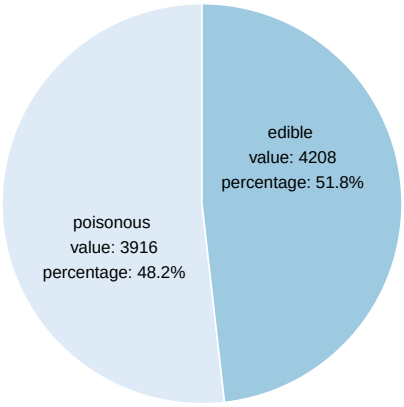
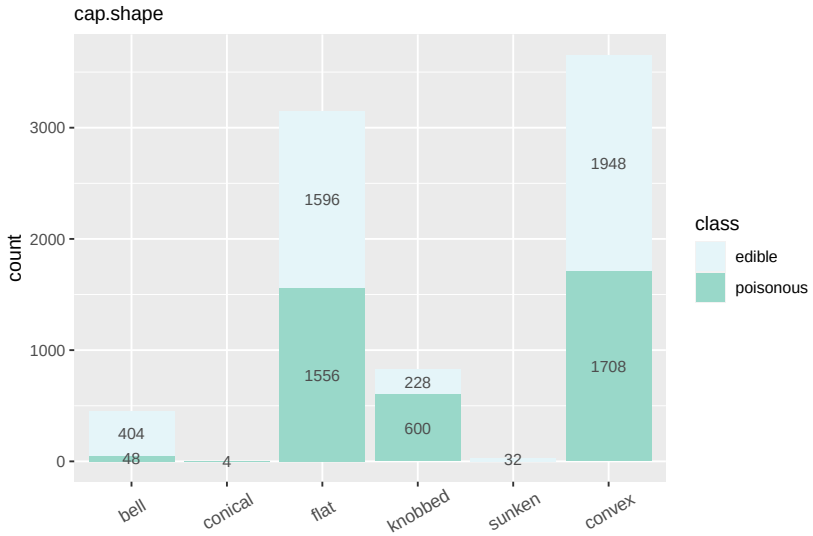
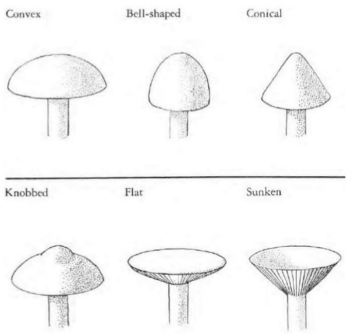


图 2: 类别的分布情况

2. 菌盖形状：

有 6 种取值，分别为 bell=b，conical=c，convex=x，flat=f，knobbed=k，sunken=s（等号表示缩写后符号，下同），如图3所示 [3]。





(a) 菌盖形状分布情况　　**(b)** 菌盖形状示意图

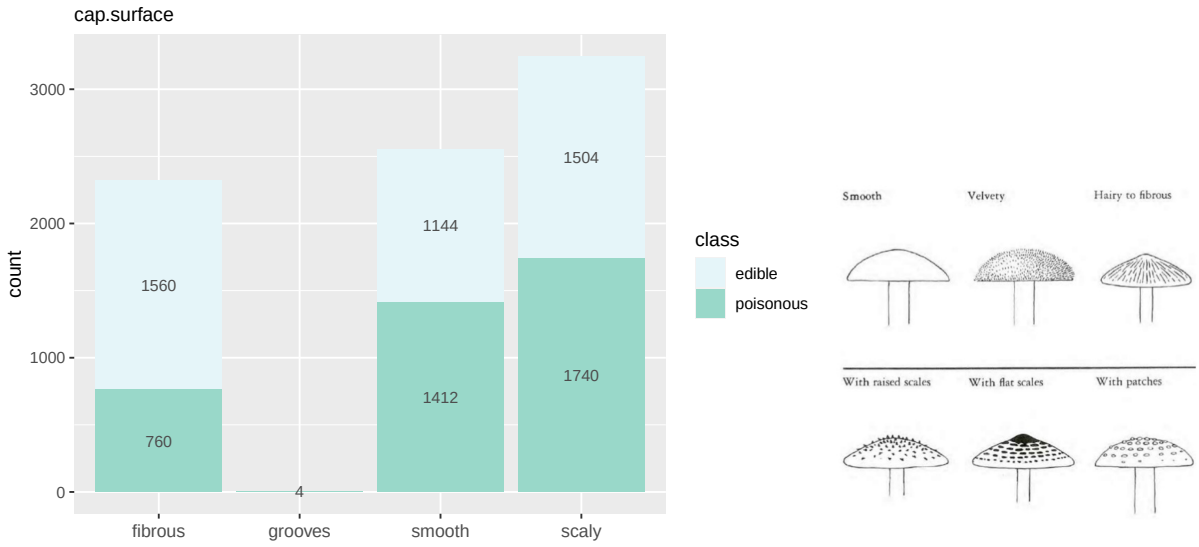
图 3: 菌盖形状

3. 菌盖表面特征：

有 4 种取值，分别为 fibrous=f，grooves[4] =g，scaly=y，smooth=s，如图4所示。

[3]图片由 ggplot2 在 Rmd 中绘制并生成 pdf 文件，示意图来自网络[4]

[4]表示菌盖表面有凹槽，未找到相应的图示

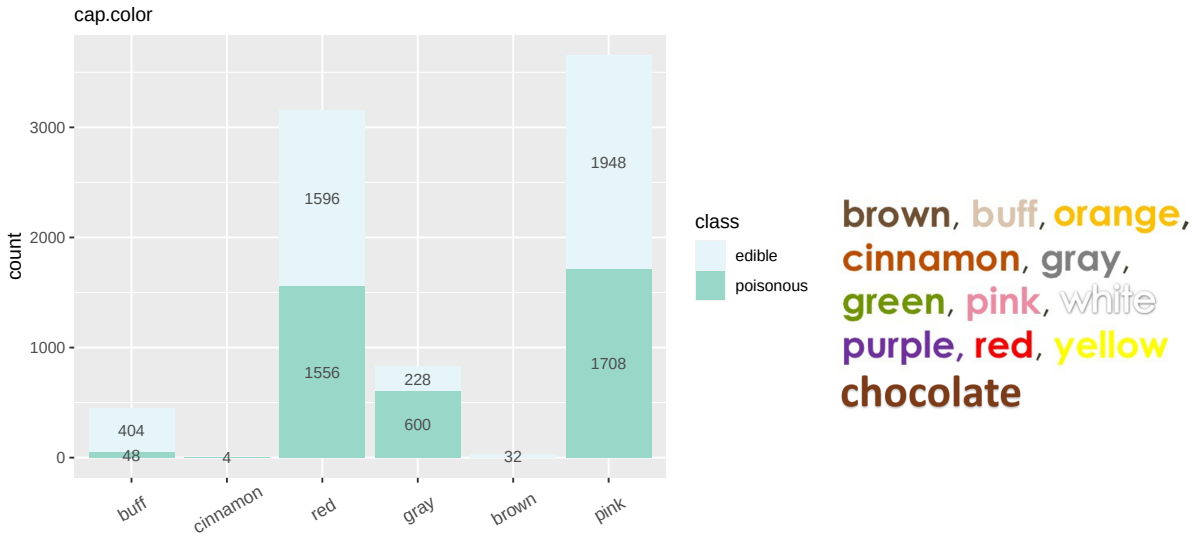


(a) 菌盖表面特征分布情况　　**(b)** 菌盖表面特征示意图

图 4: 菌盖表面特征

4. 菌盖颜色：

   有 6 种取值，分别为 brown=n，buff=b，cinnamon=c，gray=g，pink=p，red=e，如图5所示。实际颜色与名称的对比如图11所示。



(a) 菌盖颜色分布情况　　**(b)** 颜色示意图

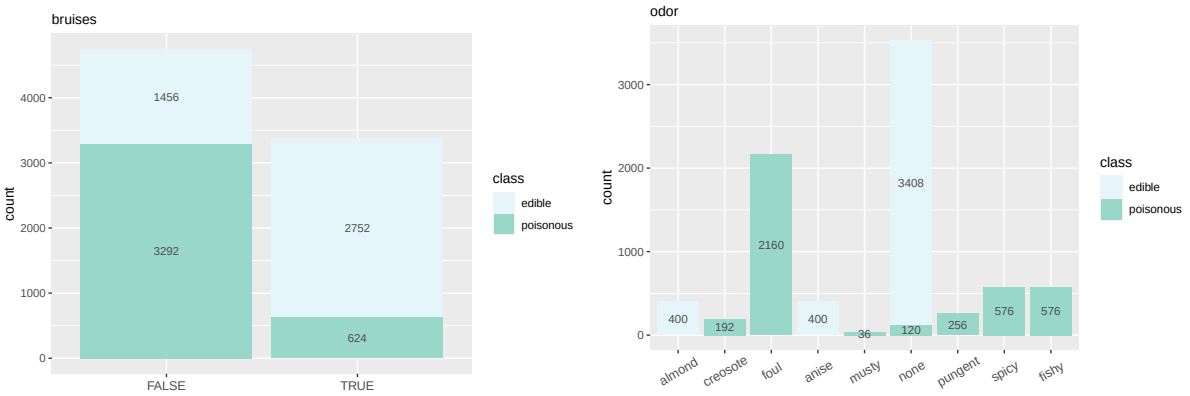
图 5: 菌盖颜色

5. 擦伤反应：

   有 2 种取值，分别为 TRUE 表示存在擦伤反应，FALSE 表示不存在擦伤反应，如图6a所示。擦伤反应即当蘑菇受到损伤时，伤口出现的特殊颜色，一般为蓝色或者绿色。存在擦伤反应表明该蘑菇含有一些特殊化合物，可能与致幻（即不可食用）相关[5]。但数据集中大多数可食用蘑菇也存在擦伤反应。

6. 气味：

有 9 种取值，分别为 almond=a，anise=l，creosote=c，fishy=y，foul=f，musty=m，none=n，pungent=p，spicy=s，如图6b所示。可以看出，特殊气味与不可食用有很大的相关性。无味（none）基本为可食用菌，但也有例外。因此单凭无味不可以断定蘑菇可食用。

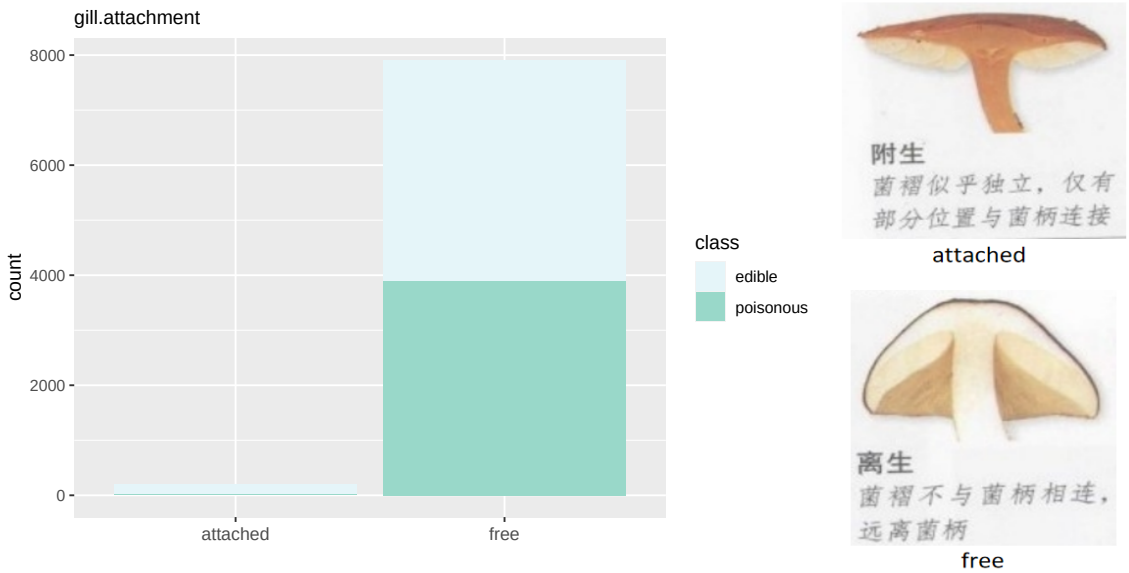


(a) 擦伤反应分布情况　　**(b)** 气味分布情况

图 6: 擦伤反应和气味

7. 菌褶剖面：

有 2 种取值，分别为 attached=a，free=f，如图7所示[6]。



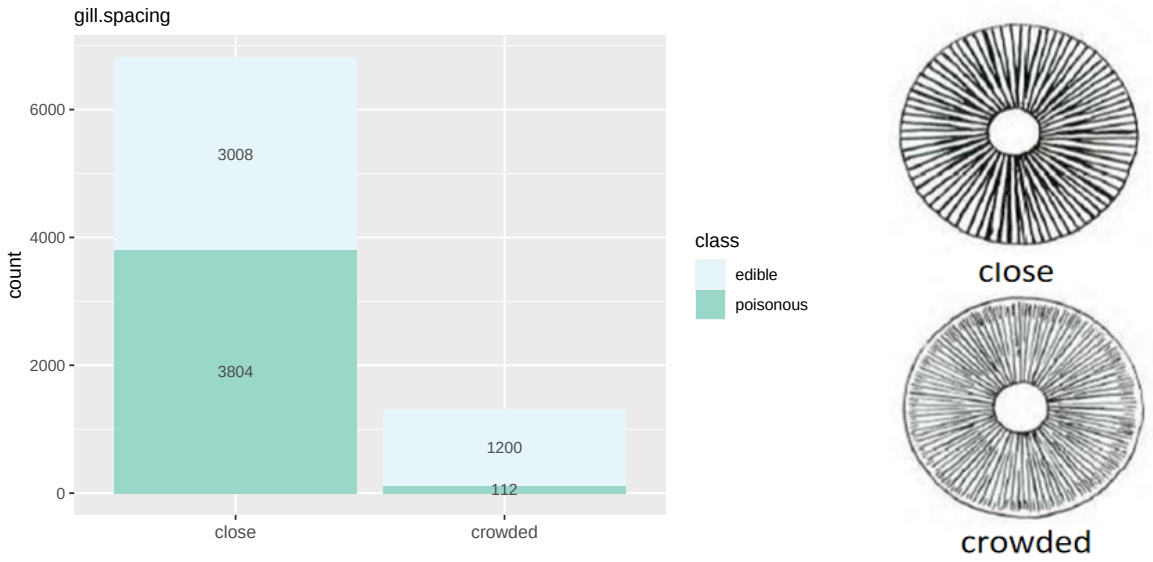
(a) 菌褶剖面分布情况　　**(b)** 菌褶剖面示意图

图 7: 菌褶剖面

8. 菌褶间隔：

有 2 种取值，分别为 close=c，crowded=w，如图8所示。可以看出菌褶细、间隔密集的蘑菇大多数是可以食用的。

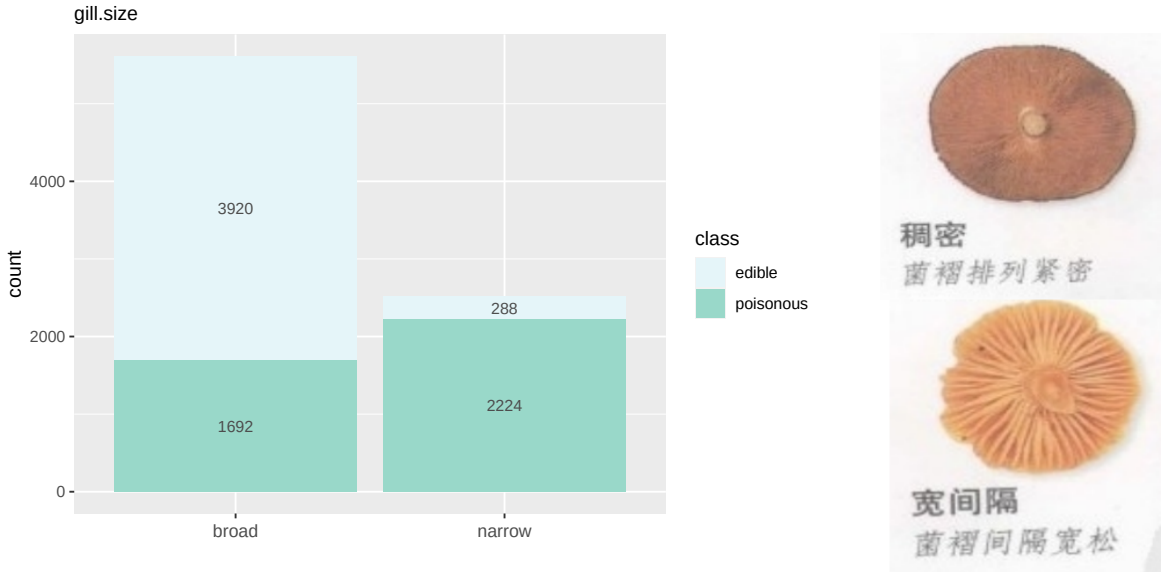
9. 菌褶大小：



(a) 菌褶间隔分布情况　　**(b)** 菌褶间隔面示意图

图 8: 菌褶间隔

有 2 种取值，分别为宽间隔（broad=b），稠密（narrow=n），如图9所示。菌褶稠密的蘑菇大都是不可食用的。



(a) 菌褶大小分布情况　　**(b)** 菌褶大小面示意图

图 9: 菌褶大小

10. 菌褶颜色：

    有 12 种取值，分别为 black=k，brown=n，buff=b，chocolate=h，gray=g，green=r，orange=o，pink=p，purple=u，red=e，white=w，yellow=y，如图10所示，实际颜色如图11所示。菌褶颜色为 buff 的全部不可食用；而菌褶颜色为 red 的全部可食用。

11. 菌柄形状：

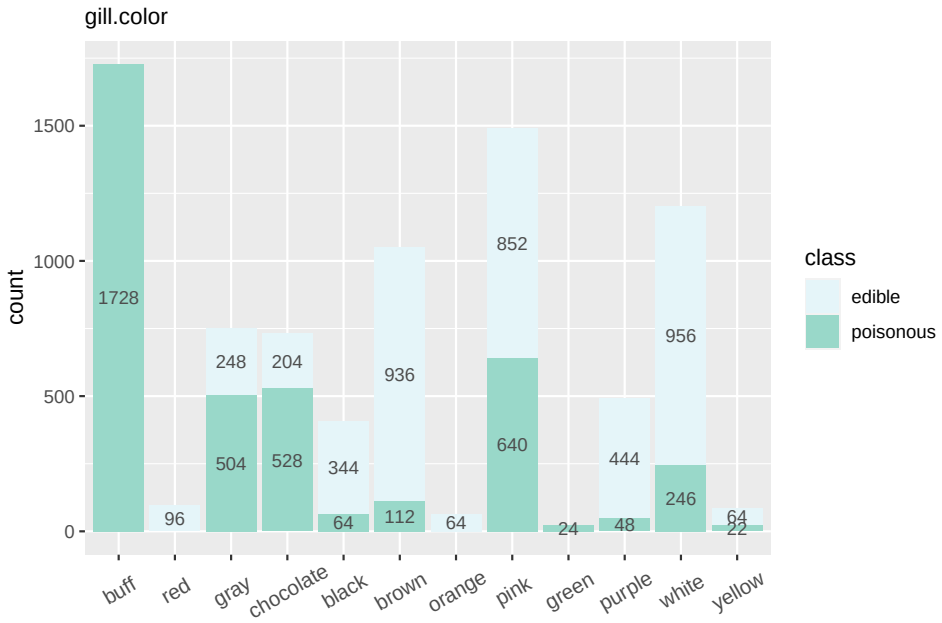


图 10: 菌褶颜色分布情况

有 2 种取值分别为上细下粗（enlarging=e），上粗下细（tapering=t），如图12所示。

12. 菌托：

有 5 种取值，分别为 bulbous=b，club=c，equal=e，rooted=r，missing=?，其中 “?” 表示缺失值，两种类型蘑菇共 2480 个缺失值，在所有样品中占比超过 30%。如图13所示。可以看出，rooted 型菌托的蘑菇均可以食用。出现缺失值的原因可能为：

- 无法判断菌托类型：从13b中可以看出 bulbous 型和 cup 型菌托长相相似，很难分辨。
- 损伤根部：采集的过程中损伤根部以至于无法判断菌托类型。缺失值中不可食用比例高于其它菌托类型，这可能是因为有毒的蘑菇更容易受到损伤。
- 蘑菇本身未发育完全。

判断缺失类型为完全非随机缺失 (Missing Not At Random，MNAR) 和任意缺失。关于缺失值的介绍及处理在章节3.4 [5]。

13. 环上柄面：

有 4 种取值，分别为 fibrous=f，scaly=y，silky=k，smooth=s，分布情况如图14a所示，示意图如图15所示。

14. 环下柄面：

有 4 种取值，分别为 fibrous=f，scaly=y，silky=k，smooth=s，分布情况如图14b所示，示意图如图15所示。可以看出环上柄面和环下柄面分布情况大致相同。计算发现共有 1868 个样品环上柄面和环下柄面不同，约占总样品数的 23.0%。

[5]Kaggle 和其他平台上很多解答把 “?” 当成一个独立的类别，因为不处理缺失值，一些分类器（例如基于树的随机森林）依然可以有很好的表现。更何况缺失值中不可食用的比例高于其它菌托类型，这有助于分类。但本文认为不处理缺失值是不合理的，即使缺失和不可食用确有关联。

图 11: 实际颜色对照图

15. 环上柄色：
    有 9 种取值，分别为 brown=n，buff=b，cinnamon=c，gray=g，orange=o，pink=p，red=e，white=w，yellow=y，如图16a所示。

16. 环下柄色：
    有 9 种取值，分别为 brown=n，buff=b，cinnamon=c，gray=g，orange=o，pink=p，red=e，white=w，yellow=y，如图16b所示。可以看出环上柄色和环下柄色分布情况大致相同。具体地，9 种取值中有 6 种数量上完全相同。猜想环上柄色和环下柄色具有很强的相关性。然而计算发现共有 3056 个样品环上柄色和环下柄色不同，约占总样品数的 37.6%。

17. 菌幕类型：
    只有 1 种取值，为 partial=p。这一特征对分类完全没有帮助，因此在章节3.1中删除这一特征[6]。

18. 菌幕颜色：
    有 4 种取值，分别为 brown=n，orange=o，white=w，yellow=y，如图17所示。大多数蘑菇菌幕为白色。菌幕为棕色或橘色的蘑菇均可食用，为黄色的均不可食用。

19. 菌环数量：
    有 3 种取值，分别为无,1,2，如图18a所示。这份数据集中大部分是有菌环的蘑菇品种，事实上常见的食用蘑菇是没有菌环的。可以看出数据集中没有菌环的蘑菇均不可食用。这里可以选择理解为有序因子或无序因子，但我认为取无序因子更合适，因为不同菌环数目代表不同类别。

20. 菌环类型：

[6]Kaggle 上很少有人注意到这类细节。

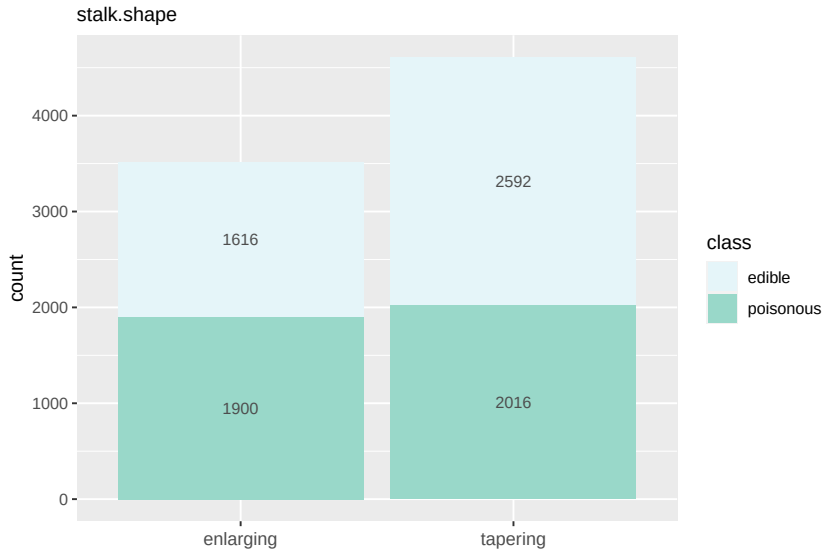
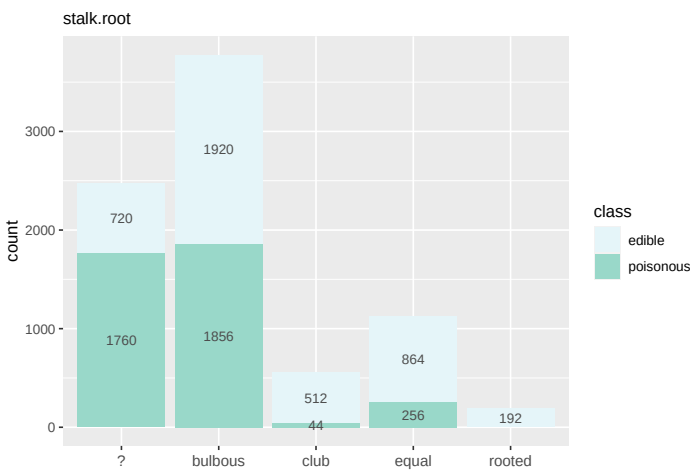
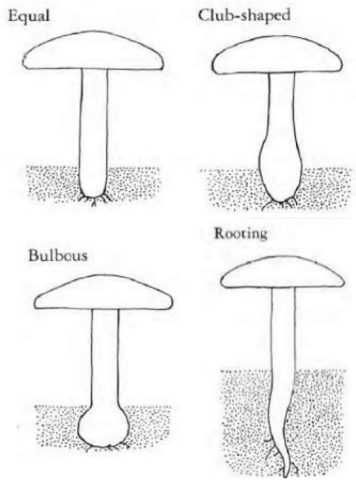


图 12: 菌柄形状分布情况



(a) 菌托分布情况



(b) 菌托示意图

图 13: 菌托

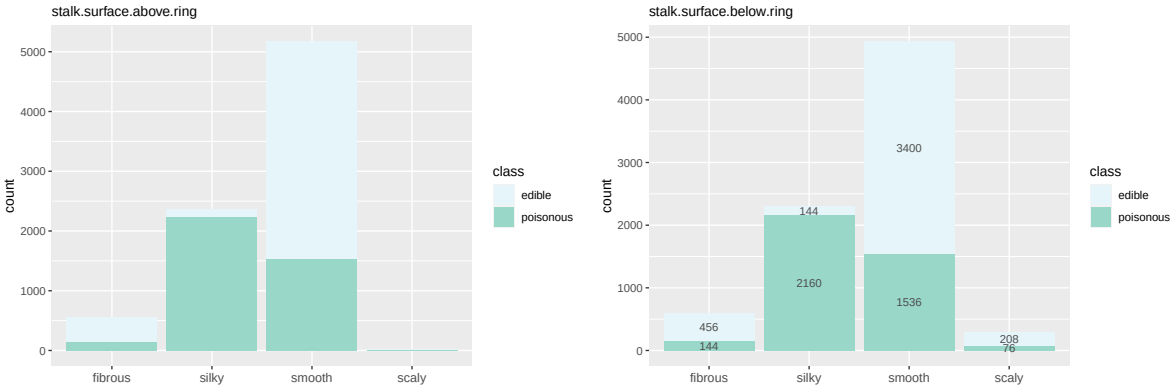
有 5 种取值，分别为 cobwebby=c，evanescent=e，flaring=f，large=l，none=n，pendant=p，sheathing=s，zone=z，如图18b所示。菌环类型为 flaring 的蘑菇均可食用，有大菌环和没有菌环的均不可食用。

21. 孢子印颜色：

有种取值，分别为 black=k，brown=n，buff=b，chocolate=h，green=r，orange=o，purple=u，white=w，yellow=y，如图19所示。颜色参见图17b。孢子印颜色为 buff、黄色、紫色或橘色的蘑菇均可食用，为黑色或棕色的大多数可以食用，为巧克力色和白色的大都不可食用。

22. 生长分布：

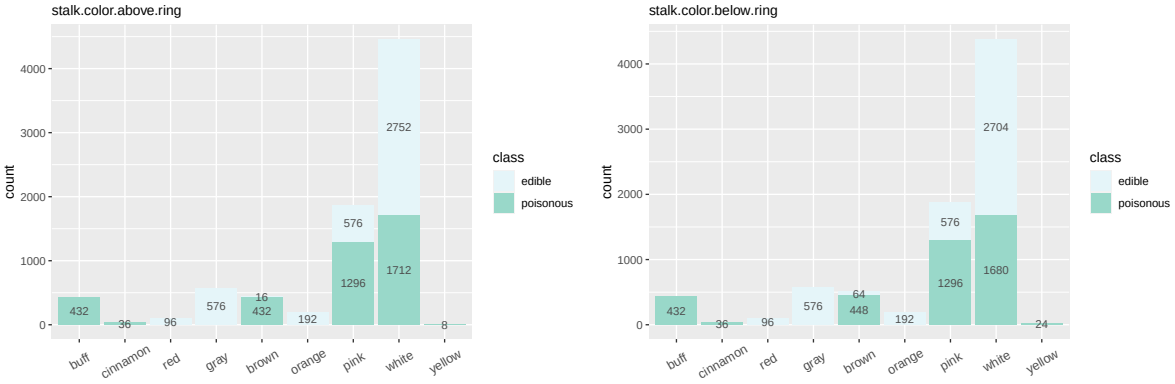
有 6 种取值，分别为群生（abundant=a），簇生（clustered=c），叠生（numerous=n），散生（scattered=s），丛生（several=v），单生（solitary=y）[7]，如图20a所示。生长分布为 abundant



(a) 环上柄面分布情况　　**(b)** 环下柄面分布情况

图 14: 柄面

图 15: 菌柄形状示意图



(a) 环上柄色分布情况　　**(b)** 环下柄色分布情况

图 16: 柄色

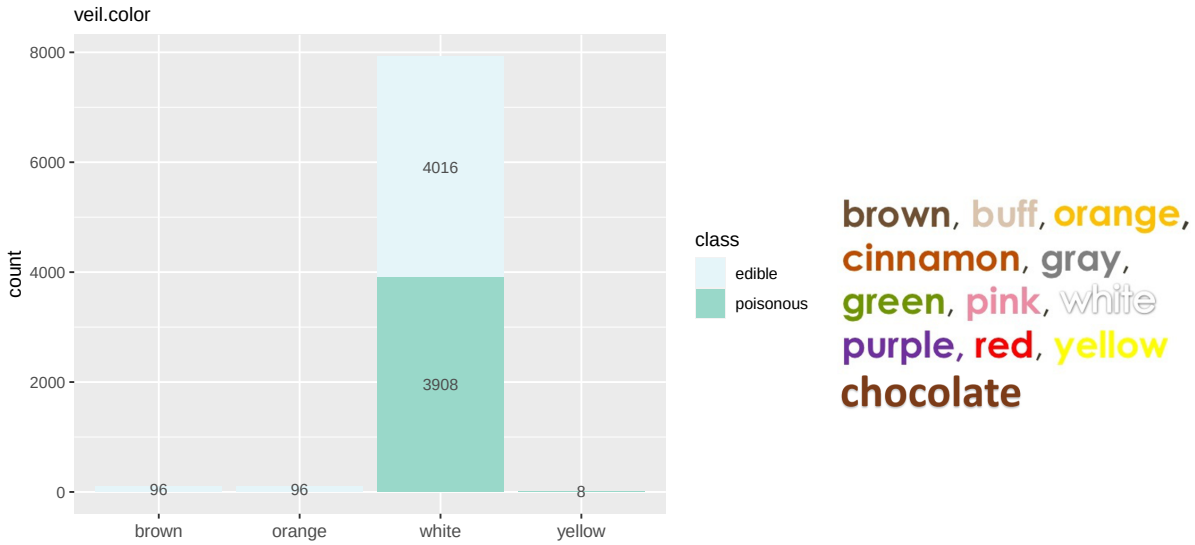
和 numerous 的蘑菇均可食用。

23. 生长环境：

有 7 种取值，分别为 grasses=g，leaves=l，meadows=m，paths=p，urban=u，waste=w，woods=d，如图21所示。生长在有污染的环境的蘑菇均可食用[7]，生长在路边的蘑菇大多不可食用。

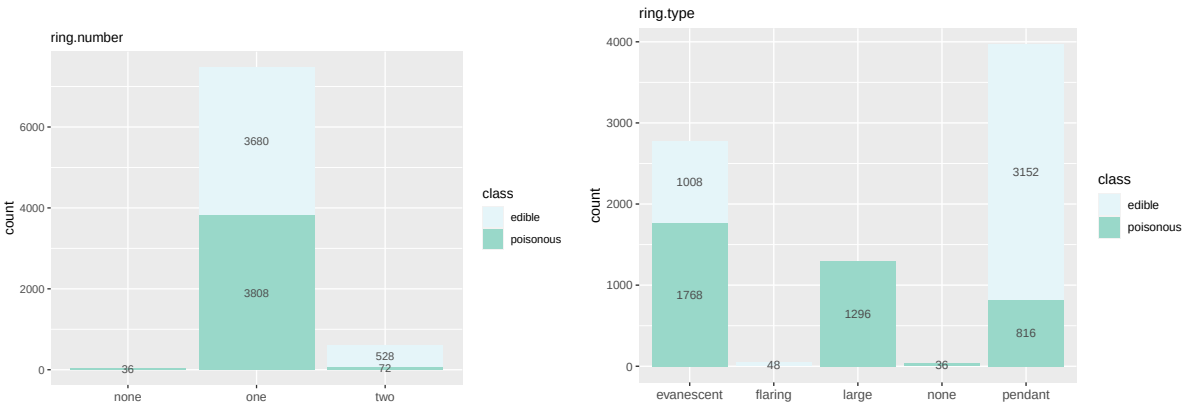
至此，23 个特征全部介绍完毕。从上述分析中可以感受到，很多特征都和可以食用或不可以食用有明显的相关性。这说明利用蘑菇的外形特征判断是否可食用是可行的。然而，仅根据上面的直方图无法找出准确判断所有蘑菇是否可食用的方法。因此，还需要进一步的数据分析。

[7]感觉违背常识，我专门确认了类别没有标反。



(a) 菌幕颜色分布情况　　**(b)** 颜色示意图

图 17: 菌幕颜色



(a) 菌环数量分布情况　　**(b)** 菌环类型分布情况

图 18: 菌环数量和菌环类型分布情况

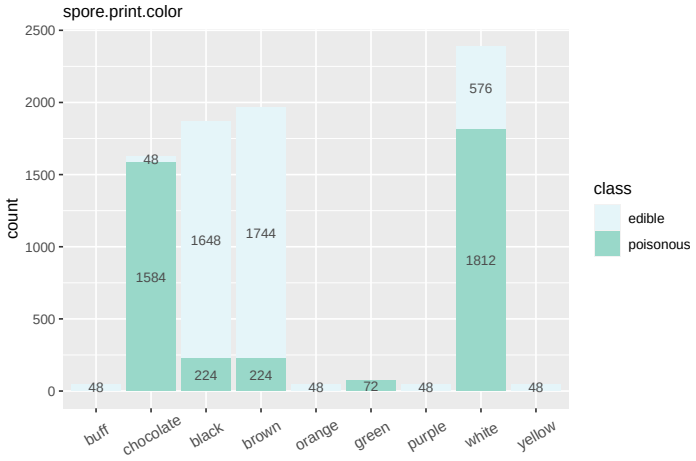
此外，注意到表1中很多取值并未在数据集中出现，意味着基于此数据集建立的模型可能不适用于除北美地区 Agaricus 和 Lepiota 科之外的蘑菇食用性鉴别。一些特征的部分取值样本量较少，可能对模型有影响。

接下来，先根据探索性分析对数据进行预处理，再尝试建立判断蘑菇是否可食用的模型。

注：正文未展示的探索性分析见附录A。

3 预处理

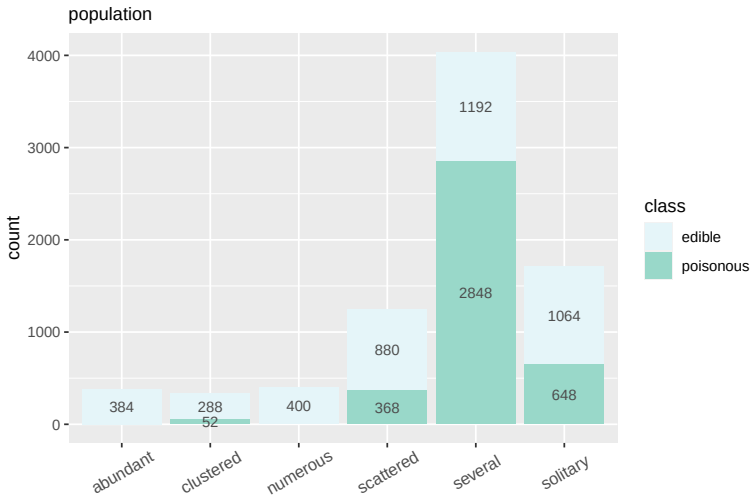
在这一部分中，本文根据探索性分析对数据进行预处理。读入数据后，删除取值全部相同的特征。之后，利用随机森林计算特征重要性，进一步确认需要填补缺失值。为避免训练集与测试集之间的数据泄露，在填补缺失值之前进行训练集与测试集的划分，之后对训练集与测试集分别进行填补。由于数据集中均为因子变量，为了使用逻辑回归等模型，将因子变量编码为哑



(a) 孢子印颜色分布情况

(b) 孢子印获取方式示意图

图 19: 孢子印颜色



(a) 生长分布分布情况

(b) 4 种较难区分的生长分布

图 20: 生长分布

变量。

首先，读入附件中的数据集并命名为“mushroom”。

3.1 删除不必要的特征

由于所有样本的 veil.type 取值相同，这一个特征对分类没有价值，删去这列。删去后数据保存在 data.o 中，mushroom 可以作为备份数据。

```
data.o <-  mushroom
data.o$veil.type <- NULL
```

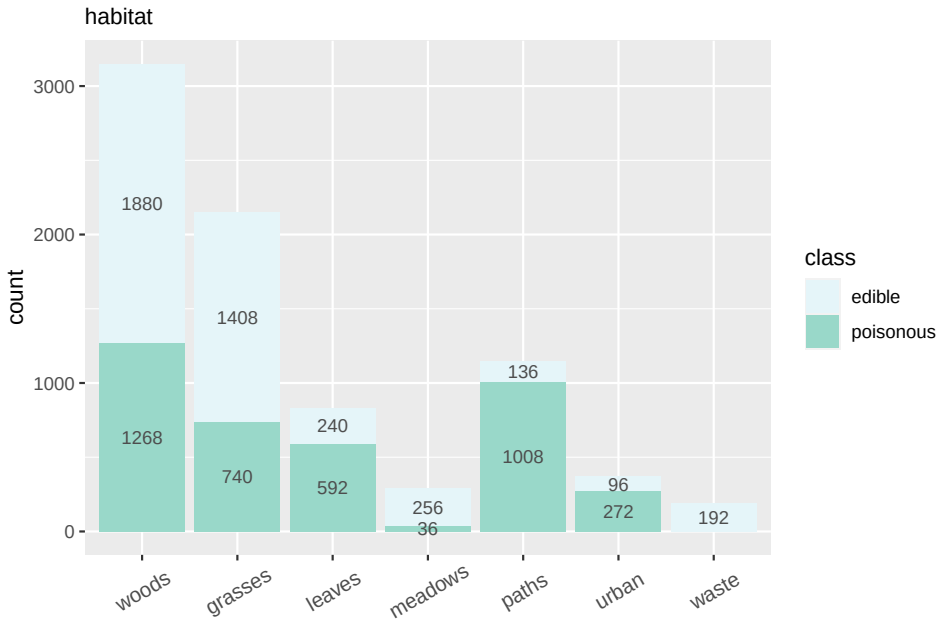


图 21: 生长环境

3.2 计算变量的重要性

利用 randomForest 包的随机森林模型，计算各个变量的 MeanDecreaseGini，用于评判变量的重要性。计算时忽略 stalk.root（菌托）缺失的样本。如图22所示，odor（气味）的重要性远超出其它特征；stalk.root 的重要性不该被忽视，因此在3.4 中将对缺失值进行补全。

3.3 划分训练集与测试集

如果先进行缺失值填补，训练集的数据情况必将影响测试集。所以，为避免训练集与测试集之间的数据泄露，本文在填补缺失值之前进行训练集与测试集的划分。考虑到数据量等因素，划分比例取训练集：测试集 = 8：2[8]。划分后，训练集 6499 个样本，测试集 1625 个样本。此后，模型训练和参数调整将在训练集中进行，测试集用于计算模型最终的精度。划分代码如下所示，test.index 表示抽取作为测试集的样本序号。

```
set.seed(0)
test.index <- sample(1:nrow(data.o), replace = F, size = ceiling
   (0.2*nrow(data.o)))
```

3.4 缺失值处理

缺失值从缺失的分布来讲可以分为：

- 完全随机缺失（missing completely at random，MCAR）：数据的缺失是随机的，数据的缺失不依赖于任何不完全变量或完全变量；

[8]与 XGBoost 划分比例相同

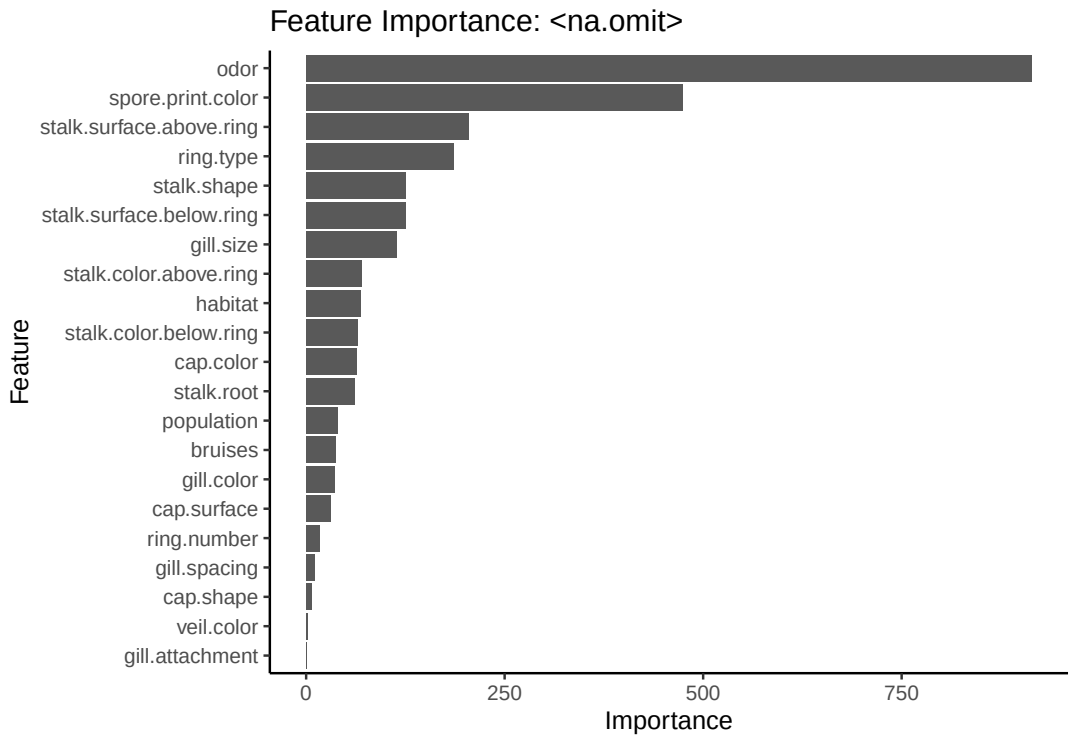


图 22: 变量重要性

- 随机缺失 (missing at random，MAR)：数据的缺失不是完全随机的，即该类数据的缺失依赖于其他完全变量；
- 完全非随机缺失 (missing not at random，MNAR)：数据的缺失依赖于不完全变量自身。

缺失值从缺失值的所属属性来讲可以分为：

- 单值缺失：如果所有的缺失值都是同一属性；
- 任意缺失：缺失值属于不同的属性；
- 单调缺失：对于时间序列类的数据，可能存在随着时间的缺失[8]。

根据探索性分析第12条中的分析，认为菌托的缺失属于完全非随机缺失和任意缺失。根据数据集的特点，选取如下 3 种填补方法：

1. 决策树：使用 rpart 包的决策树模型预测缺失值；
2. 多重填补（Multivariate Imputation，MI）：一种基于重复模拟的处理缺失值方法，处理复杂的缺失值问题最常选用的方法。这里采用 mice 包利用链式方程的多元插补（Multivariate Imputation via Chained Equations，MICE）完成。mice() 中，缺失值的插补通过 Gibbs 抽样完成。每个包含缺失值的变量都默认可通过数据集中的其他变量预测得来，于是这些预测方程便可用来预测缺失数据的有效值 [9]。mice() 填补过程示意图如图23所示。
3. k 近邻（k Nearest Neighbors，kNN）：使用 VIM 包的 kNN() 预测缺失值。

为选取最适合数据集的填补方法，将随机地在无缺失样本中人为制造缺失值，再分别用上述 3 种方法预测，以此判断填补方法的效果。具体步骤如下：

1. 取出不含缺失值的样本，同时把目标变量 class 排除在外 [9]；
2. 将上述样本按 7：3 划分为训练集和测试集。选取这个比例的原因是：原数据中缺失比例

[9]删去目标变量是有必要的，因为后续需要用菌托来预测蘑菇是不是可食用。

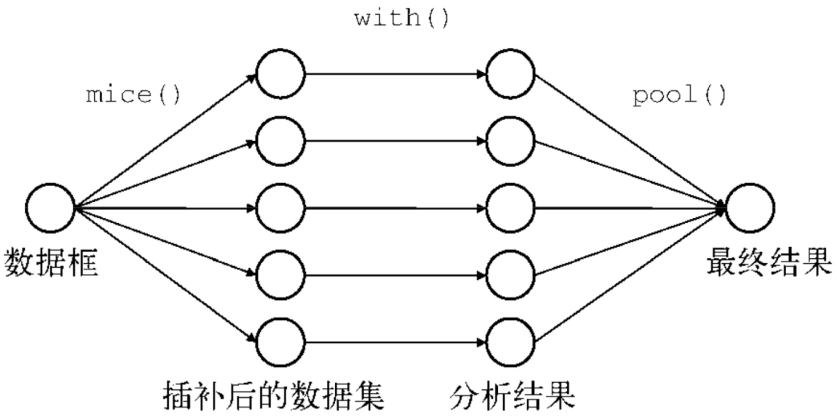


图 23: 通过 MICE 包应用多重插补的步骤：函数 mice() 首先从一个包含缺失数据的数据框开始，然后返回一个包含多个 (默认为 5 个) 完整数据集的对象。每个完整数据集都是通过对原始数据框中的缺失数据进行插补而生成的。由于插补有随机的成分，因此每个完整数据集都略有不同。然后，with() 函数可依次对每个完整数据集应用统计模型 (如线性模型或广义线性模型) 。最后，pool() 函数将这些单独的分析结果整合为一组结果。最终模型的标准误和 p 值都将准确地反映出由于缺失值和多重插补而产生的不确定性。

约为 30%；

3. 分别用 3 种方法在训练集上训练[10]；
4. 分别用 3 种方法在测试集上测试，并记录结果。

结果如表2所示。其中，kNN 的 k 值为 3。可以看出 kNN 效果最好，在测试集上全部预测正确。同时其它两个方法效果也很好。最终选用的方法是 kNN（k = 3）。使用 KNN 在3.3中确定的

表 2: 3 种方法的填补准确率

方法	kNN	mice	rpart
准确率	1	0.9994	0.9947

训练集和测试集上进行填补。选用 kNN 不仅是因为这个方法效果最好，还因为它满足模型在进化论上的意义。理论上其它特征相似的蘑菇应该有相似的菌托。同时 k 选取 3 也允许这一情况的存在：两种蘑菇除了菌托不同，其它特征相同。

```
data.o$stalk.root[na.index] = NA
data.o$stalk.root = factor(data.o$stalk.root)
train.x = kNN(data.o[-test.index, -1], variable = 'stalk.root', k
    = 3)
train.x$stalk.root_imp = NULL
train.y = data.o[-test.index, 1]
train.data = train.x
train.data$class = train.y
test.x = kNN(data.o[test.index, -1], variable = 'stalk.root', k =
    3)
```

[10]这里 kNN 参数 k 的确定最好使用交叉检验

```
test.x$stalk.root_imp = NULL
test.y = data.o[test.index, 1]
test.data = test.x
test.data$class = test.y
```

填补后 stalk.root 分布变化如图24所示。填补主要集中在 bulbous 与 equal，这说明探索性分析12 中的猜测应该是正确的，缺失值的出现很可能是因为这两类菌托难以分辨。

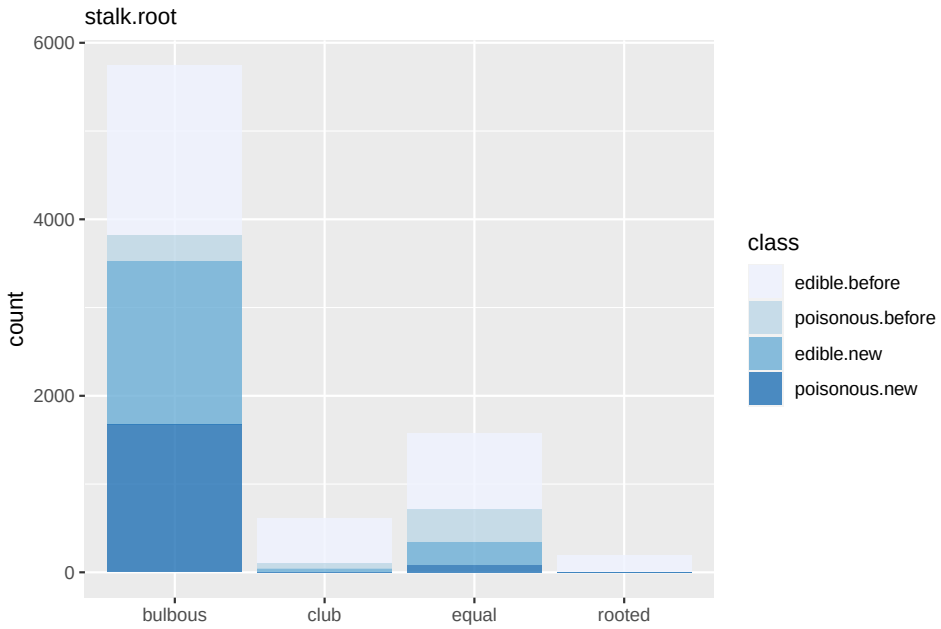


图 24: 填补前后菌托分布情况

3.5 编码为哑变量

填补完成后，将因子变量编码为哑变量。注意到由于个别特征取值的数量过少，可能全部被分到训练集中，导致测试集中不存在该取值。编码成哑变量会使得训练集和测试集变量数目不同，使得模型无法运行。因此需先检查是否存在上述取值，编码后手动调整，为测试集添加变量 [11]。

使用 dummies 包编码哑变量代码如下所示。

```
library(dummies)
train.xd = dummy.data.frame(train.x, sep = ".")
train.xd = train.xd[, order(colnames(train.xd))]
train.yd = as.numeric(train.y) - 1
train.d = train.xd
train.d$class = train.yd
```

[11]我有考虑过先编码哑变量，再划分训练集和测试集，最后进行填补。这样就不会出现这个问题。但是编码成哑变量后很难填补，所以最终决定最后编码哑变量。

```
test.xd = dummy.data.frame(test.x, sep = ".")
test.xd$cap.surface.g = 0
test.xd = test.xd[, order(colnames(test.xd))]
test.yd = as.numeric(test.y) - 1
test.d = test.xd
test.d$class = test.yd
```

最终数据集变为：

```
  bruises cap.color.b cap.color.c cap.color.e cap.color.g  ...
1    TRUE           0           0           0           0  ...
2    TRUE           0           0           0           0  ...
3    TRUE           0           0           0           0  ...
4    TRUE           0           0           0           0  ...
5   FALSE           0           0           0           1  ...
6    TRUE           0           0           0           0  ...
```

即由 23 个变量，扩展为 114 个可以由 {0，1} 编码的变量。

编码后可以计算变量之间相关性[12]，如图25所示。右边第一列是类别变量，和几乎所有哑变量都有较强的相关性。部分变量之间相关性较强，提示在回归时可能有多重共线性的问题。

4 数据分析

这一部分将使用降维（主成分回归、线性判别分析、LASSO 回归、逐步回归）、树（决策树、随机森林、XGBoost）和其它可以使用的方法（kNN、SVM、神经网络、JPip、PART）对蘑菇数据集进行建模分析。

4.1 降维

根据图25的分析，考虑到编码为哑变量的数据集可能存在多重共线性，同时 114 个自变量如果直接进行逻辑回归会造成维度爆炸。所有在进行回归之前，要么选择变量，要么尝试用较低的维度表示高维数据。可选的方法有主成分回归、线性判别分析、LASSO 回归和逐步回归。这几种方法都需要用哑变量编码的数据。

1. 主成分回归：

   参考《应用多元统计分析》进行主成分回归。首先将 114 个自变量用几个主成分表示，进行逻辑回归，再利用 PCA 的旋转矩阵将回归系数转换成原始变量的回归系数，由此建立逻辑回归模型[10]。选取主成分为 4,5,6,7 [13] 利用 pROC 包绘制 ROC 曲线并计算 AUC，如

[12] Kaggle 上很多教程默认变量为有序因子，然后计算相关性。这样显然不对。

[13] 原始变量间多重共线性并不强，并且这里的主成分也找不到特殊的含义。同时，20 个 PCs 才解释 80% 的方差，31 个 PCs 才解释 90% 的方差。说明 PCR 不是合适的模型。

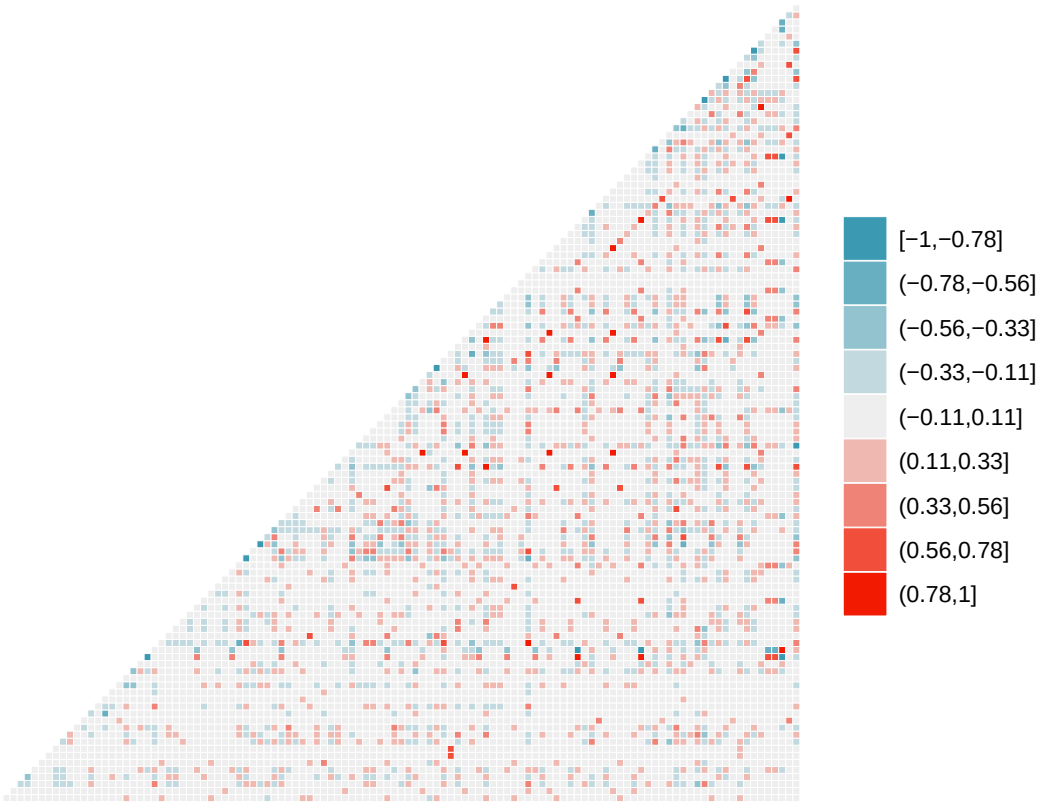


图 25: 114 个变量的相关性

图26所示。AUC 随着 PCs 数量的增加而增加。当 PCs=7 时，在测试集上分类正确率最高为 93.85% 。

2. 线性判别分析：使用 MASS 包的 lda()，在测试集上分类正确率为 99.88%。
3. LASSO 回归：

   LASSO 回归在普通线性回归上增加了约束 $\sum_{j=1}^{r}|\beta_j| \leq c$ 。问题的正则形式是寻找 $\boldsymbol{\beta}$ 最小化

   $$\phi(\boldsymbol{\beta}) = (\mathcal{Y} - \mathcal{X}\boldsymbol{\beta})^{\tau}(\mathcal{Y} - \mathcal{X}\boldsymbol{\beta}) + \lambda \sum_{j=1}^{r} |\beta_j| \,. \text{ [11]}$$

   使用 glmnet 包的 cv.glmnet() 在测试集上通过交叉验证选择合适的参数。将合适的参数代入 glmnet() 获得最终的模型，模型在测试集上分类正确率为 100%[14]。
4. 逐步回归：

   使用 step() 对逻辑回归模型进行后向逐步回归[15]，在测试集上分类正确率为 100% 。

4.2 树

使用决策树、随机森林等模型的优势在于不需要使用哑变量编码，对于因子型数据操作方便。同时，决策树模型可以给出一个明确且可以在现实中推广的判别模型。而随机森林和 XGBoost

[14]LASSO 的优势在于可以变量选择。它在 114 个自变量（哑变量）中选择了 28 个用于回归。选出的 28 个变量涉及到原有 22 个自变量中的 21 个。

[15]运行了将近 2h。

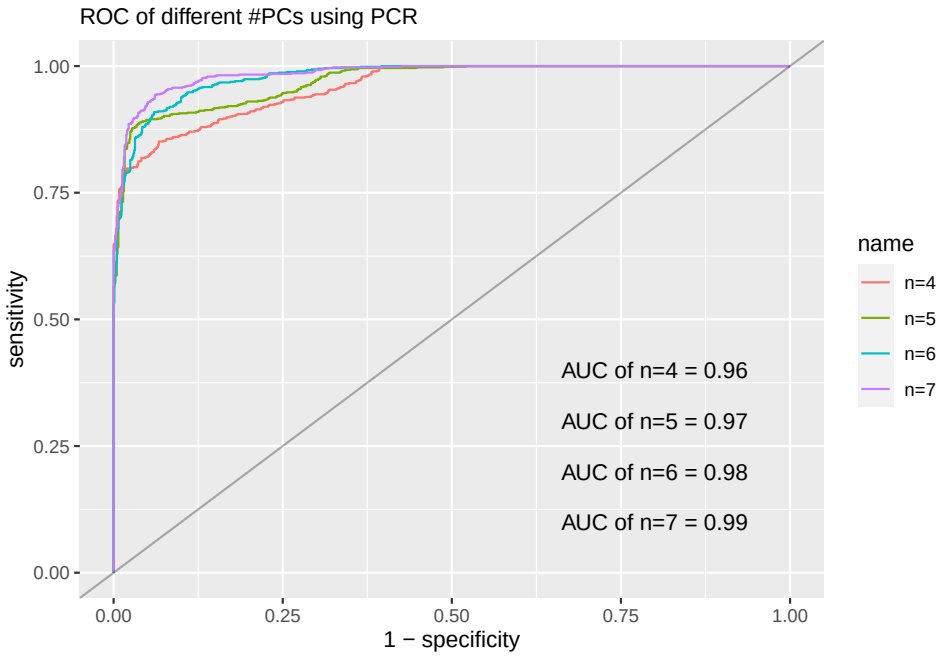


图 26: PCR 的 ROC 曲线及 AUC

都可以给出变量的重要性，由此可知哪些特征对判断蘑菇是否可食用最有帮助。接下来将使用决策树、随机森林和 XGBoost 建立模型。

1. CART 决策树：

   使用 rpart 包的 rpart() 建立基于 CART 算法的决策树模型，调整 cp 值使树尽可能生长，得到如图 27所示的决策树[16]，选择的特征为：odor、spore.print.color、stalk.color.below.ring 和 stalk.surface.above.ring 。在测试集上正确率为 99.88%。

2. C4.5 决策树：

   相对于CART 算法使用结点不纯度选择最佳的分裂特征,C4.5 算法选择信息增益比 $g_R(D,A)$ 完成这一过程。为定义信息增益比，首先定义信息增益：

   定义 4.1. (信息增益) 特征 A 对训练数据集 D 的信息增益 $g(D,A)$，定义为集合 D 的经验熵 $H(D)$ 与特征 A 给定条件 D 下的经验条件熵 $D(D|A)$ 之差，即

   $$g(D,A) = H(D) - H(D|A).$$

   再定义信息增益比：

   定义 4.2. (信息增益比) 特征 A 对训练数据集 D 的信息增益比 $g_R(D,A)$ 定义为其信息增益 $g(D,A)$ 与训练数据集 D 关于特征 A 的值的熵 $H_A(D)$ 之比，即

   $$g_R(D,A) = \frac{g(D,A)}{H_A(D)}$$

   其中，$H_A(D) = -\sum_{i=1}^{n} \frac{|D_i|}{|D|} \log_2 \frac{|D_i|}{|D|}$ ，n 是特征 A 取值的个数[12]。

   同时，C4.5 在构造树的过程中剪枝且可以处理缺失值[13]。使用 RWeka 包的 J48() 绘制的决策树如下所示。选择的特征为：odor、spore.print.color、gill.size、gill.spacing 和 population。

[16]决策树左下角 6 个错误分类的样品情况比较复杂，为避免过拟合，树不继续生长

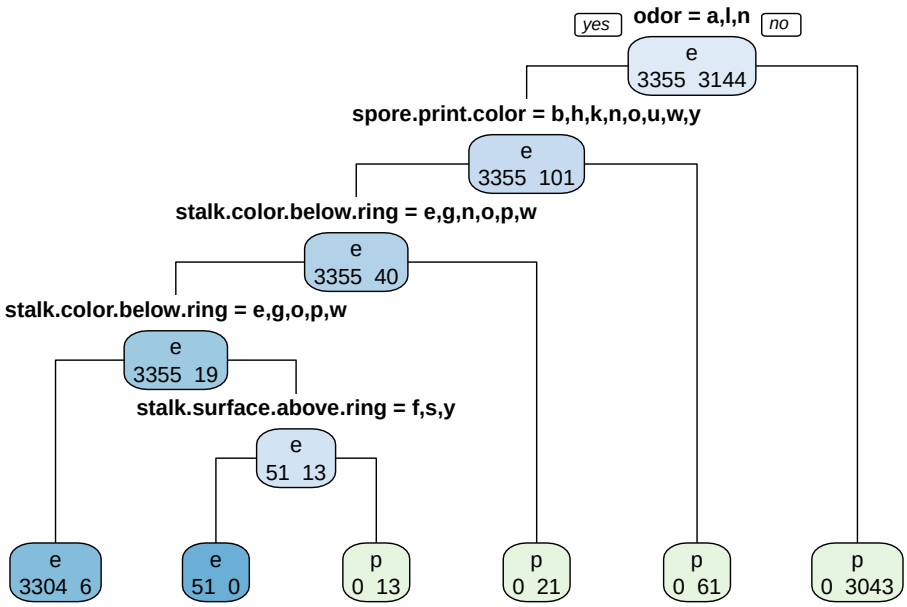


图 27: CART 决策树

前两个特征和 CART 决策树选择的一致。模型在测试集上正确率为 100%。使用含有缺失值的数据（即未经过补全，且缺失值设置为 NA 的数据）时，测试集上正确率为 85.29%。

```
J48 pruned tree
------------------

odor = a: e (307.0)
odor = c: p (154.0)
odor = f: p (1741.0)
odor = l: e (316.0)
odor = m: p (33.0)
odor = n
|   spore.print.color = b: e (37.0)
|   spore.print.color = h: e (39.0)
|   spore.print.color = k: e (1053.0)
|   spore.print.color = n: e (1081.0)
|   spore.print.color = o: e (34.0)
|   spore.print.color = r: p (61.0)
|   spore.print.color = u: e (0.0)
|   spore.print.color = w
|   |   gill.size = b: e (405.0)
|   |   gill.size = n
|   |   |   gill.spacing = c: p (28.0)
|   |   |   gill.spacing = w
|   |   |   |   population = a: e (0.0)
|   |   |   |   population = c: p (12.0)
|   |   |   |   population = n: e (0.0)
```

```
|   |   |   |   population = s: e (0.0)
|   |   |   |   population = v: e (39.0)
|   |   |   |   population = y: e (0.0)
|   spore.print.color = y: e (44.0)
odor = p: p (198.0)
odor = s: p (471.0)
odor = y: p (446.0)

Number of Leaves  :  24

Size of the tree :  29
```

3. C5.0 决策树：

   C5.0 比 C4.5 更快更高效，同时采用 Boosting 方式提高模型准确率。使用 C50 包的 C5.0() 绘制的决策树如图29所示，变量重要性如图**??**所示，选择的特征为：odor、spore.print.color、cap.surface、stalk.color.below.ring 和 stalk.surface.above.ring，和 CART 相同。其在测试集上正确率为 99.88%。使用含有缺失值的数据时，测试集上正确率为 99.88%。相对于 C4.5 算法，C5.0 在含缺失值数据上有很好的表现。注意到 C4.5 和 C5.0 都是多叉树。

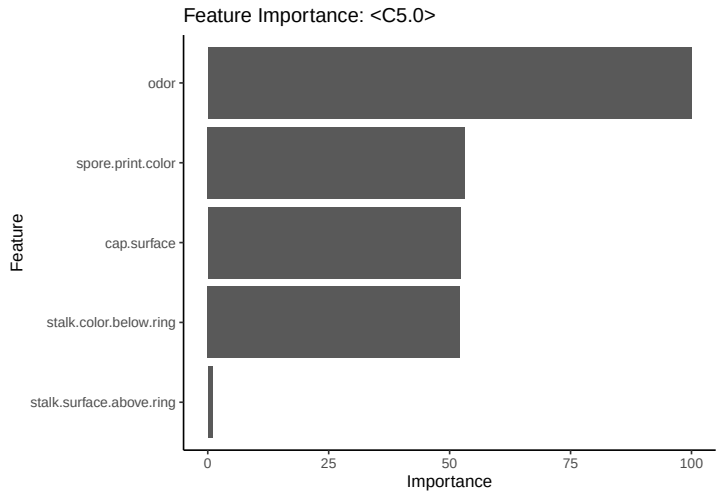


图 28: C5.0 变量重要性

4. 随机森林：

   使用 randomForest 包的 randomForest()，测试集特征的重要性如图30所示。重要性较高的特征有：odor、spore.print.color、gill.color、gill.size、stalk.color.below.ring 和 stalk.surface.above.ring。无论是否含有缺失值，在测试集上正确率均为 100%。

5. XGBoost：[17]

   XGBoost（eXtremeGradient Boosting）将多个 CART 模型集成在一起，形成一个很强的分类器。通过不断地添加树，不断地进行特征分裂来生长一棵树。测试样本在每棵树中都会落到一个对应于特定分数的叶子节点，将分数相加就是预测值。

[17]我原本以为这一部分会相对轻松，因为 Mushroom 是 XGBoost 的示例数据集，但还是因为数据格式等问题，调试很久才做出来。

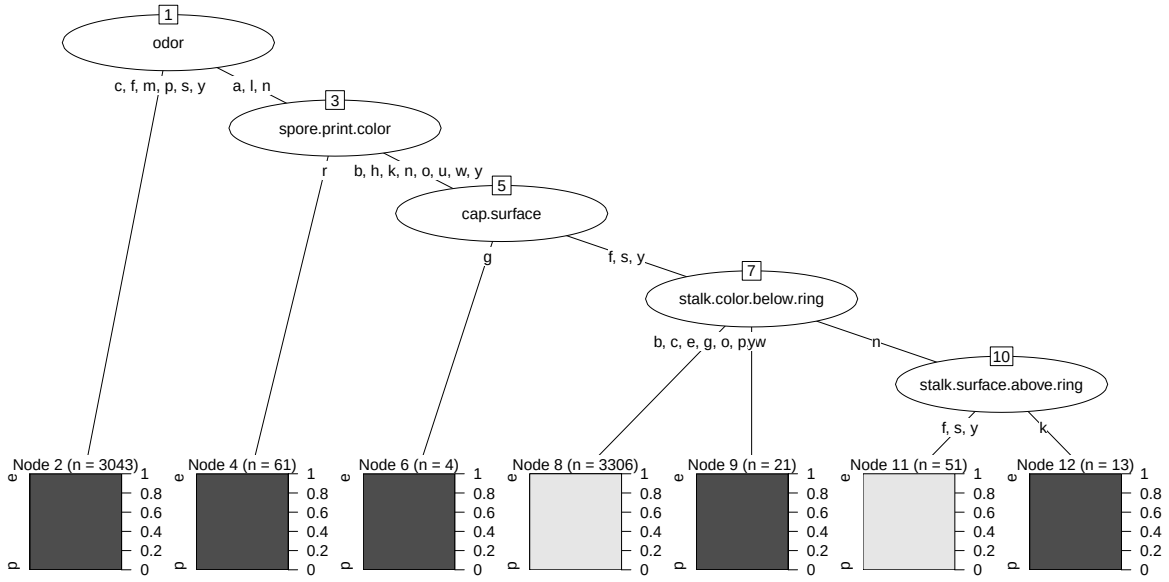


图 29: C5.0 决策树

XGBoost 的目标函数为

$$\mathcal{L}(\phi) = \sum_i l\left(\hat{y}_i, y_i\right) + \sum_k \Omega\left(f_k\right)$$
$$\text{where } \Omega(f) = \gamma T + \frac{1}{2}\lambda\|w\|^2.$$

其中，$l\left(\hat{y}_i, y_i\right)$ 为可微凸损失函数，用于测量预测和目标之间的差异；T 表示叶子节点数；w 表示叶子节点权重。

利用泰勒展开可以近似得到第 t 次迭代的目标函数

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n}\left[g_i f_t\left(\mathbf{x}_i\right) + \frac{1}{2}h_i f_t^2\left(\mathbf{x}_i\right)\right] + \Omega\left(f_t\right)$$

其中，$g_i = \partial_{\hat{y}(t-1)} l\left(y_i, \hat{y}^{(t-1)}\right)$，$h_i = \partial^2_{\hat{v}(t-1)} l\left(y_i, \hat{y}^{(t-1)}\right)$。对于给定的实例，计算叶子节点权重的公式为

$$w_j^* = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda},$$

目标函数可以化简为

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T}\frac{\left(\sum_{i\in I_j} g_i\right)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T,$$

结点分裂可通过计算

$$\mathcal{L}_{split} = \frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma$$

选择。

XGBoost 还提出了两种防止过拟合的方法：Shrinkage 和 Column(Feature) Subsampling。Shrinkage 方法是在每次迭代中对树的每个叶子结点的分数乘上一个缩减权重，类似随即优化中的学习率。Column Subsampling 也在随机森林中使用[14]。

使用 xgboost 包中 xgb.cv() 先在训练集上通过 5 折交叉检验寻找最佳参数，之后使用该参数用 xgboost() 建立模型，并在测试集上进行预测，得到正确率为 99.63%，绘制的变量重

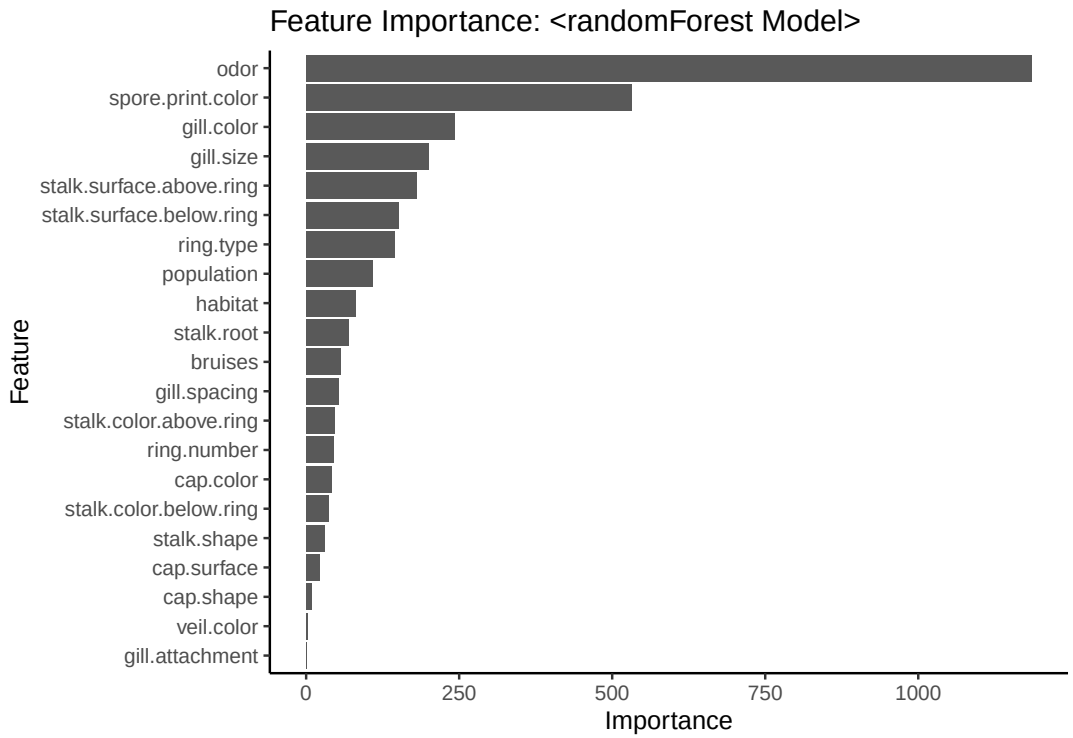


图 30: 随机森林特征重要性

要性如图31所示。

综合上述讨论，基于树的模型表现都相当不错。几种算法都认为 odor（气味）是最重要的指标，除了 XGBoost 之外的 4 种方法认为 spore.print.color（孢子印颜色）是第二重要的指标。

4.3 其它

这一部分讨论其它可以使用的方法，包括 kNN、SVM、NN 和规则学习算法 RIPPER 与 PART。

1. kNN：

   使用 class 包的 knn() 选择 k=1，在测试集上正确率为 100%。

2. SVM：

   使用 e1071 包的 svm()[18]，在测试集上正确率为 100%。

3. NN[15]：

   神经网络（Neural Network，NN）的设计模拟了生物体神经元间信号传递的过程。图32为一个单一的神经元模型。有向网络图定义了树突接收的输入信号（变量 x）和输出信号（变量 y）之间的关系。与生物神经元一样，每一个树突的信号都根据其重要性被加权（W_i），输入信号由细胞体求和，然后该信号根据一个用 f 表示的激活函数（Activation Function）来传递。神经网络像搭积木一样使用神经元来构造复杂的模型，可由以下 3 种特征定义：

   - **激活函数**：将高维输入信号转为一维输出信号，最常用的是 S 形激活函数（Sigmoid Activation function），如图33a所示；

[18]SVM 参数没有用 tune 选择，因为这个函数对 3*5 个参数情况运行了超过 2h 还没有结果。并且默认参数在训练集上正确率为 99.98% 。

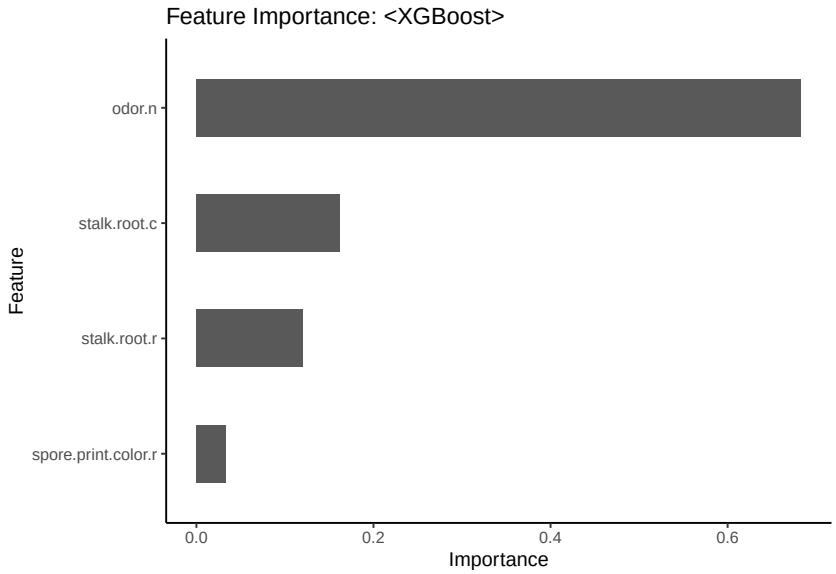


图 31: XGBoost 特征重要性

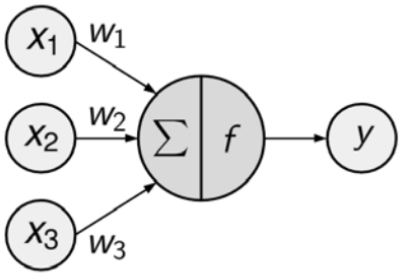


图 32: 神经元模型示意图

- **网络拓扑（Network Topology ）或结构（Architecture）**：描述模型中神经元的数量、层数和连接方式，图33b所示的多层网络添加了一个隐藏层使得总层数达到 2 层；
- **训练算法（Training Algorithm ）**：指定如何设置连接权重。

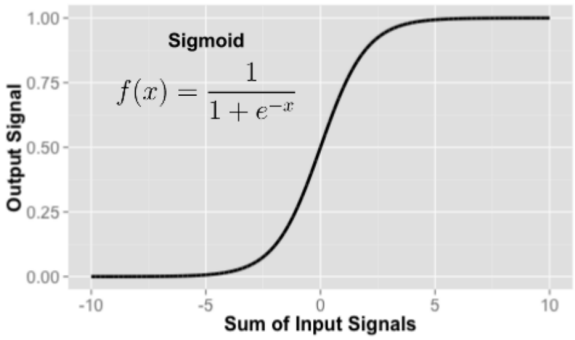
使用 neuralnet 包的 neuralnet() 对哑变量编码的数据进行处理，设置一层隐藏层，隐藏层有一个节点，在测试集上正确率为 100%。

4. RIPPER[15][16]：重复增量修剪（Repeated Incremental Pruning to Produce Error Reduction，RIPPER）由 Cohen 于 1995 年提出[17]，是增量减少误差修剪算法（Incremental Reduced Error Pruning algorithm ，IREP）的改进版，修复了效率低下的问题，性能与决策树相当。这类规则算法易于生成可理解的规则，且适合处理因子型数据，所以非常适合蘑菇数据集。算法与决策树非常相似，可以简单地分为三个步骤：
   (a) 生长：选择信息增益最高的条件，贪婪添加规则；
   (b) 修剪：当增加规则，而熵不减时，剪枝；
   (c) 优化：重复前两步，直到满足停止准则，再使用探索方法优化。

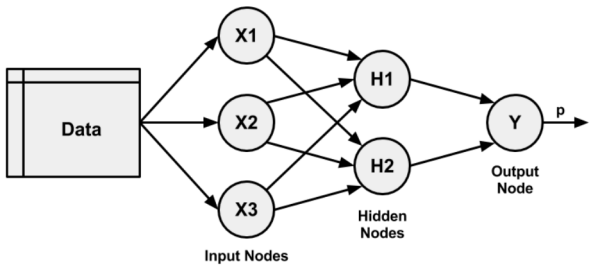
   最终得到形如

$$\oplus \leftarrow \mathbf{f}_1 \wedge \mathbf{f}_2 \wedge \cdots \wedge \mathbf{f}_L$$

   的规则。右边的部分称为规则体，表示该条规则的前提，由一系列逻辑文字 f_i 组成的合取



(a) S 形激活函数



(b) 网络拓扑示意图

图 33: 神经网络

式。左边的部分称为逻辑头，表达该条规则的结果，也是逻辑文字，一般用来表示规则所判定的目标类别或概念。

使用 RWeka 包的 JRip()，在测试集上正确率为 100%。建立规则如下：

```
JRIP rules:
===========

(odor = f) => class=p (1741.0/0.0)
(gill.size = n) and (gill.color = b) => class=p (917.0/0.0)
(gill.size = n) and (odor = p) => class=p (198.0/0.0)
(odor = c) => class=p (154.0/0.0)
(spore.print.color = r) => class=p (61.0/0.0)
(stalk.surface.above.ring = k) and (gill.spacing = c) => class=p
    (61.0/0.0)
(habitat = l) and (gill.attachment = f) and (population = c) => class=p
    (12.0/0.0)
 => class=e (3355.0/0.0)

Number of Rules : 8
```

JRip() 从训练集中学到了 8 条规则。前两条规则为：

(a) 如果气味 = f，则不可食用；

(b) 如果菌褶大小 = n 且菌褶颜色 = b，则不可食用。

其它规则的解读以此类推。最后一条规则表明：若蘑菇不满足之前的条件，则可食用。每个规则后面的数字表示被规则覆盖的样本数和分类错误的样本数。最终 8 条规则覆盖了所有样本且没有分类错误。

5. PART[19] ：

使用 RWeka 包的 PART()，在测试集上正确率为 100%。建立规则如下：

```
PART decision list
------------------
odor = f: p (1741.0)
```

[19]在看 JRip 帮助文档时发现的函数，关于这个函数的资料很少

```
gill.size = b AND ring.number = o: e (2725.0)
ring.number = t AND spore.print.color = w: e (405.0)
odor = s: p (471.0)
odor = y: p (446.0)
stalk.shape = e AND
stalk.surface.below.ring = s AND odor = p: p (198.0)
stalk.shape = e AND stalk.root = b AND ring.type = p: p (221.0)
stalk.surface.above.ring = s: e (206.0)
stalk.surface.below.ring = y: p (67.0)
: e (19.0)
Number of Rules  :  10
```

规则的解读与 JRip() 类似，PART() 使用 10 条规则覆盖了所有样本且没有分类错误。

4.4 总结

上面讨论的 14 种模型正确率如表3,4,5所示。对于可以生成可理解规则的模型：基于树的模型普遍认为气味（odor）、孢子印颜色（spore.print.color）、环下柄色（stalk.color.below.ring）、菌盖表面特征（cap.surface）和菌褶颜色（gill.color）是判断蘑菇是否可食用重要的指标。

表 3: 降维模型正确率

模型类型	降维			
模型名称	主成分回归	线性判别分析	LASSO 回归	逐步回归
正确率 (%)	93.85	99.88	100	100

表 4: 树模型正确率

模型类型	树				
模型名称	CART	C4.5	C5.0	随机森林	XGBoost
正确率 (%)	99.88	100	99.88	100	99.63

表 5: 其它模型正确率

模型名称	kNN	SVM	NN	RIPPER	PART
正确率 (%)	100	100	100	100	100

这 14 种模型中，我认为最实际可行的是决策树模型和基于规则学习的 JRIP 模型。几乎所有模型正确率都很高，但重要的是，这两个模型建立了简单具体且易于理解的规则。

5 结论

数据分析结束，回顾0引言提到的两个任务。到这里，本文已建立对于蘑菇数据集性能优良的分类器；根据基于树模型变量重要性的分析，气味（odor）、孢子印颜色（spore.print.color）、

环下柄色（stalk.color.below.ring）、菌盖表面特征（cap.surface）和菌褶颜色（gill.color）是判断蘑菇是否可食用重要的指标。对比网络上流传的鉴别蘑菇是否可食用的方法：观形状、察色味。根据数据分析，形状、颜色和气味确实对判别蘑菇是否可食用最有帮助！

尽管本文所得模型大都非常可靠，但由于数据集只记录北美两个科的蘑菇信息，许多实际存在的特征未涵盖在数据集中，所以上述模型在其它蘑菇数据集上的准确性有待探究。

5.1 后续工作

由于时间有限，仍有一些工作有待进一步讨论：

- 由于食用有毒蘑菇的代价远高于食用无毒蘑菇，在建模时可以考虑以代价最低为优化目标；
- 探索特征之间的相关性；
- 比较在含有缺失值时分类器的性能（比如使用 XGBoost 处理有缺失值的数据）；
- 比较不同的缺失值处理方法填补的缺失值，以及这些方法对分类器的影响；
- 由于多数分类器正确率都为 1，可以试试对所有样本进行判别，以更好的比较这些分类器的分类效果；
- 如果有时间，应该进一步调整各个模型的参数，探索它们的输出。

5.2 收获

蘑菇数据集是我处理的第二个 kaggle 上的数据集[20]。这份数据集并不复杂，甚至比入门的泰坦尼克号数据还要简单。虽然有缺失值，但把缺失值单独作为一类仍可以有很好的表现。特征也全部可以理解为因子型，也就没有离群值这个概念。比较麻烦的地方在于哑变量编码之后达到了 114 维，但也有很多方法可以降维、选择变量，不难处理。从探索性分析和数据分析的结果都可以看出，这份数据集质量相当好。同时，由于样本量足够（总样本数为 8124），分类效果也会好，也可以拿出足够的样本作为测试集。

和第一次处理相比，无论是对数据集的理解，还是处理（包括缺失值处理和数据分析），都有了明显的进步。除此之外，最大的收获应该是：第一次真正上手使用语法复杂但很 fancy 的 ggplot2，以及第一次尝试 XGBoost 等课本上没有，但是总在 kaggle 上看见的模型。尽管在这份简单的数据集上 XGBoost 的表现有些对不起我调参的时间与精力。

尽管正文中考虑到篇幅，对新方法的解释并不详细，但是通过阅读资料和自行总结，对新方法也有一定的了解。

[20]第一个当然是泰坦尼克号生存预测