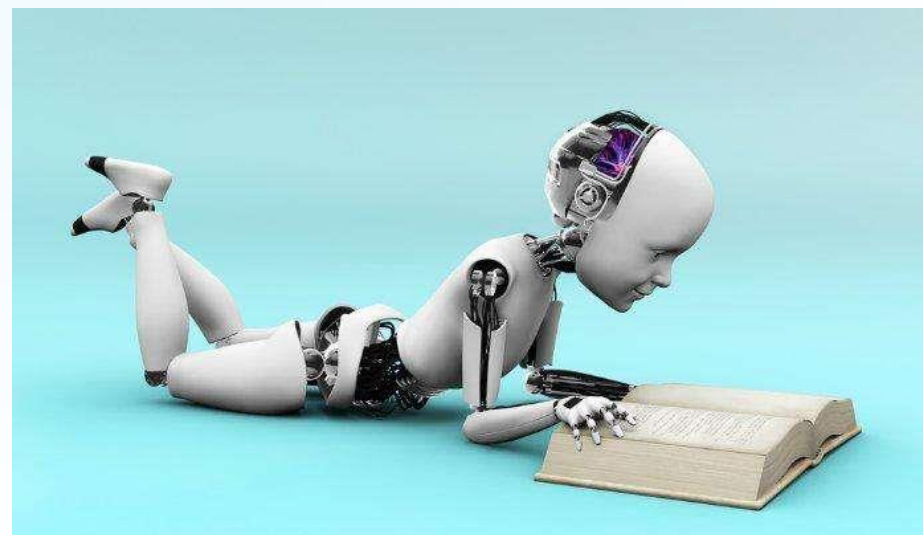




## 34机器学习概论



生信宝典 1: 下面是一段 Linux 代码，请帮我解读一下：

```
awk 'BEGIN{OFS=FS="\t"}{if(FNR==1) {for(i=3;i<=NF;i++) a[i]=$i; print "OTU","Group";} \
    else {for(i=3;i<=NF;i++) if($i>0.1) print $1, a[i];}}' \
    result/otutab_mean.txt > result/alpha/otu_group_exist.txt
```



微信扫一扫  
关注该公众号

## ChatGPT 1:

这段代码是一个 awk 命令，用于处理数据文件 result/otutab\_mean.txt，并将结果输出到文件 result/alpha/otu\_group\_exist.txt。

具体解读如下：

1. awk 命令：awk 是一种处理文本文件的工具，可以按行读取文件并根据指定的规则进行处理。
2. 'BEGIN{OFS=FS="\t"}'：BEGIN 表示在 awk 处理数据之前执行的代码块，本行代码设置输出字段分隔符 OFS 和输入字段分隔符 FS 为制表符 (\t)。
3. {if(FNR==1) {for(i=3;i<=NF;i++) a[i]=\$i; print "OTU","Group";}}：if 语句表示如果当前处理的行号为 1，即处理的是表头，则执行花括号内的代码块。该代码块使用 for 循环将第 3 列到最后一列的值存储到数组 a 中，然后输出“OTU”和“Group”两个字符串作为表头。
4. else {for(i=3;i<=NF;i++) if(\$i>0.1) print \$1, a[i];}}：else 语句表示如果当前处理的行号不是 1，即处理的是数据行，则执行花括号内的代码块。该代码块使用 for 循环遍历第 3 列到最后一列的值，如果该列的值大于 0.1，则输出该行第 1 列和对应的数组 a 中的值。
5. result/otutab\_mean.txt：要处理的输入文件路径。

<https://mp.weixin.qq.com/s/9CSSpVqOutdgQLbRF6xH8g>



# ChatGPT – 你问我答

生信宝典 2: 请帮我优化一下这段 Python 代码

```
aDict = {}
for line in open("GRCh38.fa"):
    if line[0] == '>':
        key = line[1:-1]
        aDict[key] = ''
    else:
        aDict[key] += line.strip()
```

**ChatGPT 2:**

可以通过使用 with 语句来打开文件，同时使用 rstrip() 方法代替 strip() 方法来剔除行末可能存在的空格、制表符等空白符号。另外，对于字符串拼接，可以使用 join() 方法提高效率。优化后的代码如下：

```
aDict = {}
with open("GRCh38.fa") as f:
    key = ""
    for line in f:
        if line.startswith(">"):
            key = line[1:-1]
            aDict[key] = []
        else:
            aDict[key].append(line.rstrip())
    for key, value in aDict.items():
        aDict[key] = ''.join(value)
```

生信宝典

<https://mp.weixin.qq.com/s/9CSSpVqOutdgQLbRF6xH8g>



# ChatGPT 的训练

- GPT3 的训练数据：1750亿的参数，31个分工明确的作者，超强算力的计算机（28500个CPU， 10,000个GPU），1200万的训练费用，45TB的训练数据（维基百科的全部数据只相当于其中的0.6%）。

| 模型    | 发布时间       | 参数量     | 预训练数据量 |
|-------|------------|---------|--------|
| GPT   | 2018 年 6 月 | 1.17 亿  | 约 5GB  |
| GPT-2 | 2019 年 2 月 | 15 亿    | 40GB   |
| GPT-3 | 2020 年 5 月 | 1,750 亿 | 45TB   |



# 模型就是函数集合，学习就是确定函数的最优参数

Machine Learning is so simple .....

$$y = b + wx_1$$

Step 1:  
function with  
unknown



Step 2: define  
loss from  
training data



Step 3:  
optimization

$$w^* = 0.97, b^* = 0.1k$$

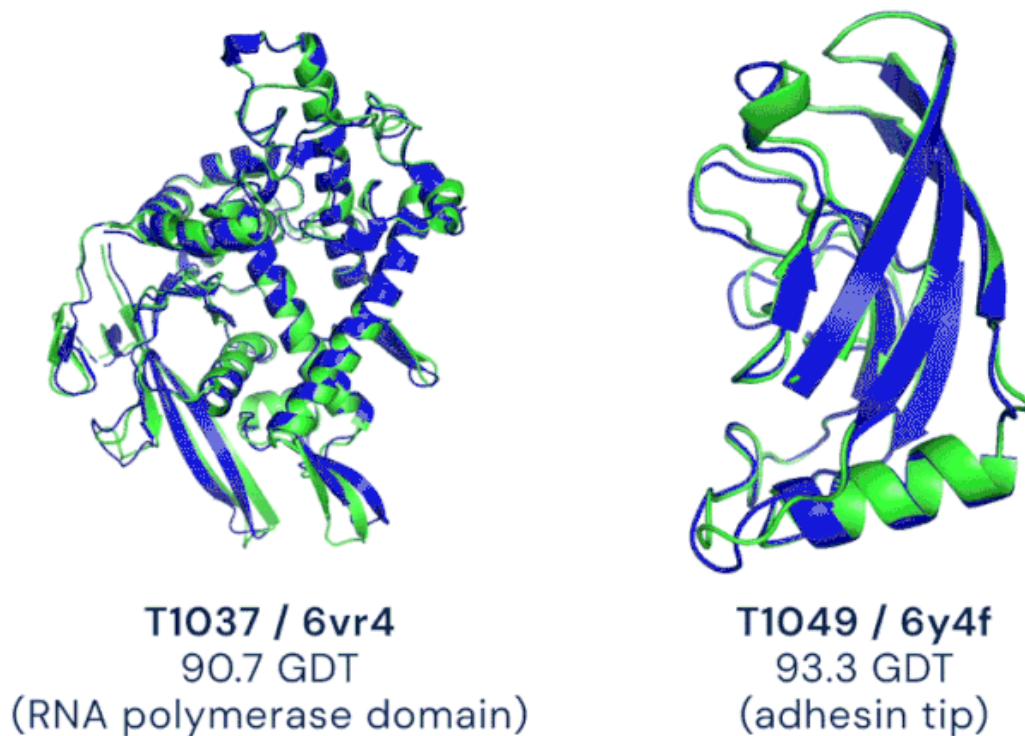
$$L(w^*, b^*) = 0.48k$$

Training

$y = 0.1k + 0.97x_1$  achieves the smallest loss  $L = 0.48k$   
on data of 2017 – 2020 (training data)

宏基因组

# AlphaFold – 机器学习预测蛋白结构



● Experimental result  
● Computational prediction

宏基因组

AlphaFold2开源了，不是土豪也不会编程的你怎么蹭一波？



# AlphaFold的硬件需求

- 硬盘 3T。Alpha fold训练好的模型加数据库下载下来是428 GB大小的文件，解压后需要2.2T的空间。
- CPU: 12个虚拟CPU（可理解为12个超线程）
- 内存：85 GB
- GPU: 1个Nvidia A100 GPU；NVIDIA A100 GPU拥有40gb的高速HBM2内存，6912个CUDA核心（大约7万元）

易生信 毕生缘 培训版权所有



# RoseTTAFold – 受限于计算资源的学术版

relationships within and between sequences, distances, and coordinates. In contrast, reasoning about 3D atomic coordinates in the two-track AlphaFold2 architecture happens after processing of the 1D and 2D information is complete (although end-to-end training does link parameters to some extent). Because of computer hardware memory limitations, we could not train models on large proteins directly as the three-track models have many millions of parameters; instead, we presented to the network many discontinuous crops of the input sequence consisting of two discontinuous sequence segments spanning a total of 260 residues. To generate final models, we combined and averaged the 1D features and 2D distance and orientation predictions produced for

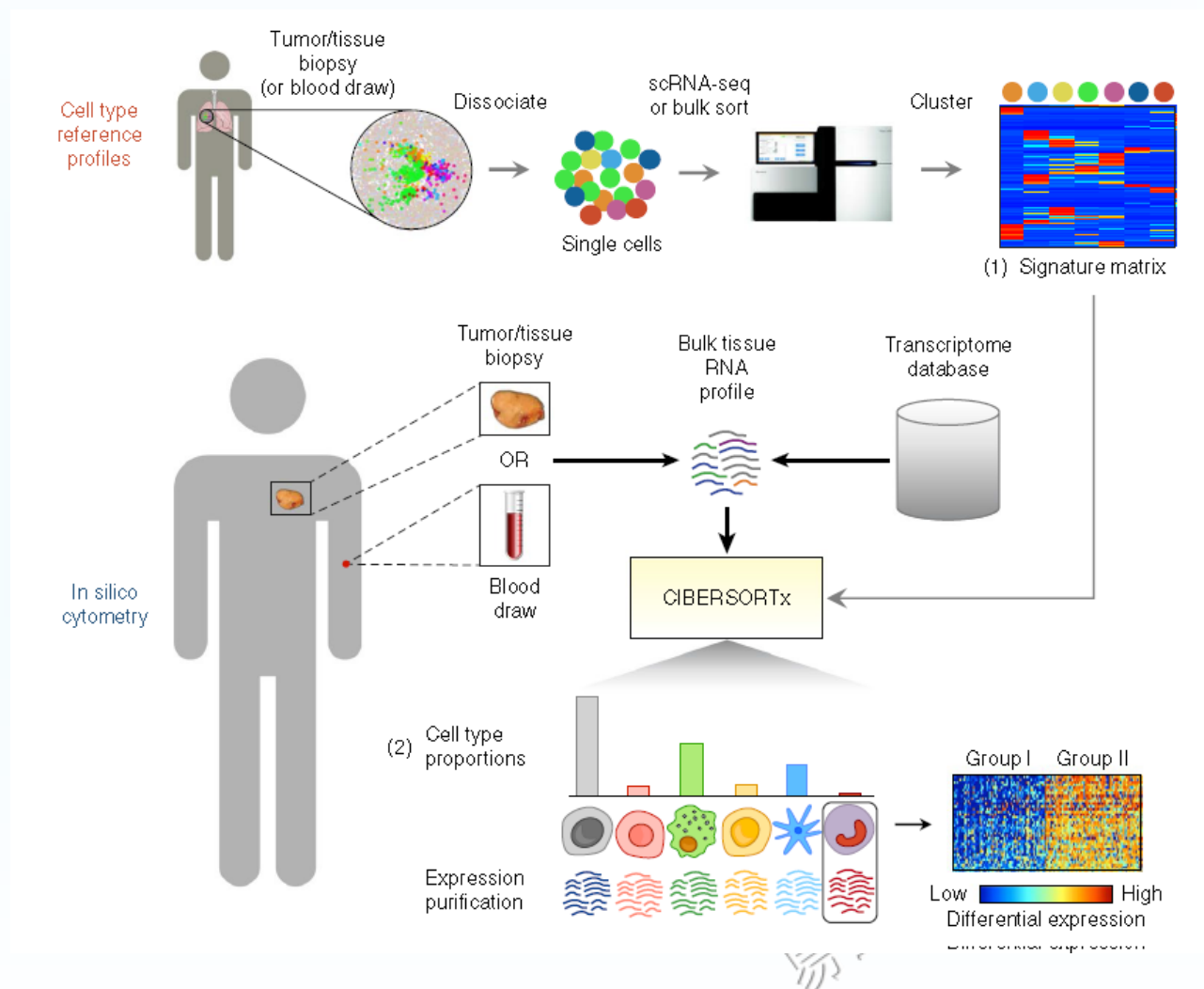
the top 2 server groups (Ensemble Server and BRACKEN-ROSETTA-ROSETTA, BRACKEN-human group (ranked second among all groups), and our two-track attention models on CASP14 targets. As in the case of AlphaFold2, the correlation between multiple sequence alignment depth and model accuracy is lower for RoseTTAFold than for trRosetta and other methods tested at CASP14 (fig. S2). The performance of the three-track model on the CASP14 targets was still not as good as AlphaFold2 (Fig. 1B). This could reflect hardware limitations that limited the size of the models we could explore, alternative architectures or loss formulations, or more intensive use of the network for inference. DeepMind reported using several GPUs for days to make individual predictions, whereas our predictions are made in a single pass through the network in the same manner that

AlphaFold2开源了，不是土豪也不会编程的你怎么蹭一波？



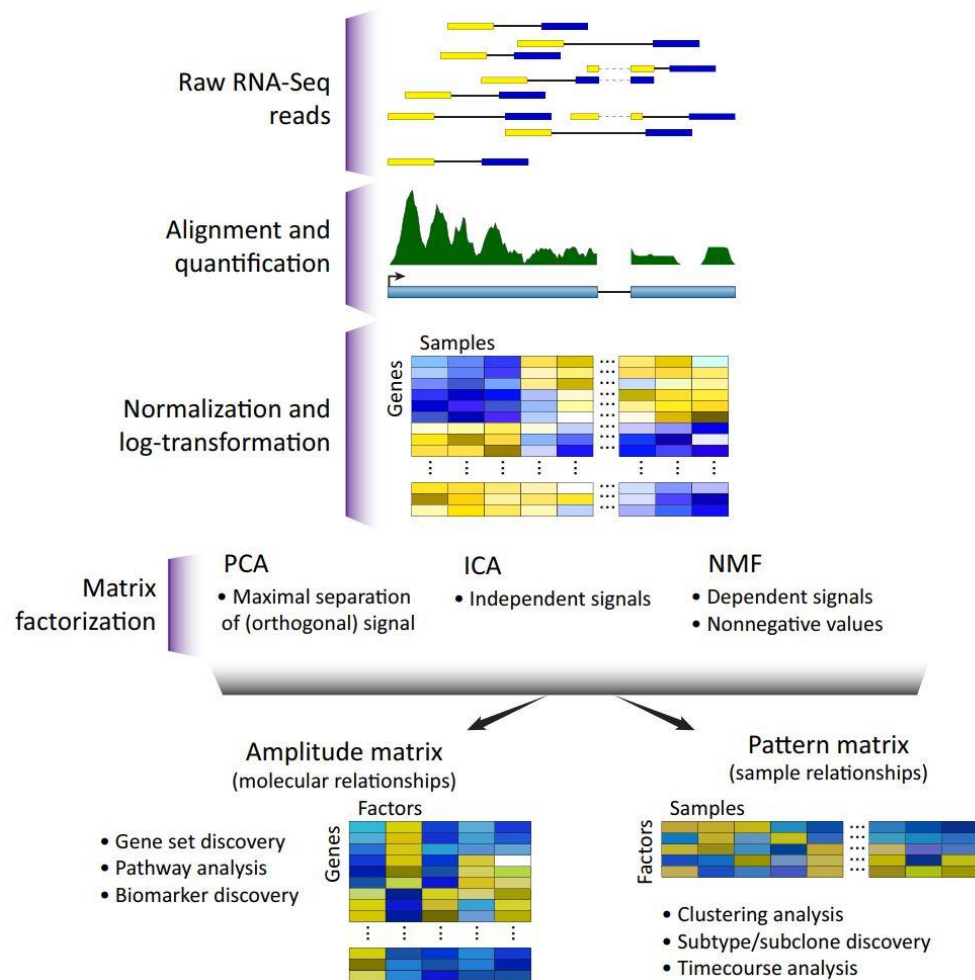


# 免疫浸润 – 一种基于机器学习的细胞分型方法



宏基因组

# 矩阵因子分解 – PCA, ICA, NMF



- **Amplitude matrix:** 每一列代表每个分子（基因）对分解出的因子的贡献。
- **Pattern matrix:** 每一行代表每个样本对分解出的因子的贡献。
- 理想上 **amplitude** 矩阵的每一列和对应的 **pattern** 矩阵的每一行代表影响样本表达状态的一个或生物、或实验、或技术的差异来源。

## Why Use DeepVariant?

- **High accuracy** - DeepVariant won 2020 [PrecisionFDA Truth Challenge V2](#) for All Benchmark Regions for ONT, PacBio, and Multiple Technologies categories, and 2016 [PrecisionFDA Truth Challenge](#) for best SNP Performance. DeepVariant maintains high accuracy across data from different sequencing technologies, prep methods, and species. For [lower coverage](#), using DeepVariant makes an especially great difference. See [metrics](#) for the latest accuracy numbers on each of the sequencing types.
- **Flexibility** - Out-of-the-box use for [PCR-positive](#) samples and [low quality sequencing runs](#), and easy adjustments for [different sequencing technologies](#) and [non-human species](#).
- **Ease of use** - No filtering is needed beyond setting your preferred minimum quality threshold.
- **Cost effectiveness** - With a single non-preemptible n1-standard-16 machine on Google Cloud, it costs ~\$11.8 to call a 30x whole genome and ~\$0.89 to call an exome. With preemptible pricing, the cost is \$2.84 for a 30x whole genome and \$0.21 for whole exome (not considering preemption).
- **Speed** - See [metrics](#) for the runtime of all supported datatypes on a 64-core CPU-only machine. Multiple options for acceleration exist, taking the WGS pipeline to as fast as 40 minutes (see [external solutions](#)).
- **Usage options** - DeepVariant can be run via Docker or binaries, using both on-premise hardware or in the cloud, with support for hardware accelerators like GPUs and TPUs.

基因组



nature  
medicine

TECHNICAL REPORT

<https://doi.org/10.1038/s41591-019-0539-7>

## An augmented reality microscope with real-time artificial intelligence integration for cancer diagnosis

Po-Hsuan Cameron Chen<sup>1,4</sup>, Krishna Gadepalli<sup>1,4</sup>, Robert MacDonald<sup>1,4</sup>, Yun Liu<sup>1</sup>, Shiro Kadowaki<sup>1</sup>, Kunal Nagpal<sup>1</sup>, Timo Kohlberger<sup>1</sup>, Jeffrey Dean<sup>1</sup>, Greg S. Corrado<sup>1</sup>, Jason D. Hipp<sup>1,2</sup>, Craig H. Mermel<sup>1\*</sup> and Martin C. Stumpe<sup>1,3\*</sup>

The microscopic assessment of tissue samples is instrumental for the diagnosis and staging of cancer, and thus guides therapy. However, these assessments demonstrate considerable variability and many regions of the world lack access to trained pathologists. Though artificial intelligence (AI) promises to improve the access and quality of healthcare, the costs of image digitization in pathology and difficulties in deploying AI solutions remain as barriers to real-world use. Here we propose a cost-effective solution: the augmented reality microscope (ARM). The ARM overlays AI-based information onto the current view of the sample in real time, enabling seamless integration of AI into routine workflows. We demonstrate the utility of ARM in the detection

宏基因组

易生信

易生信





# 未来最热门的方法——深度学习

众所周知，目前最火的研究领域，莫过于肠道菌群和深度学习，而此项研究的作者，完美的将这两项研究结合，为我们带来一项非常有趣的研究。该研究通过深度学习的算法分析肠道菌群组成来预测人类年龄，误差在4岁以内，说明菌群可作为生物标志物用于衰老相关的研究。

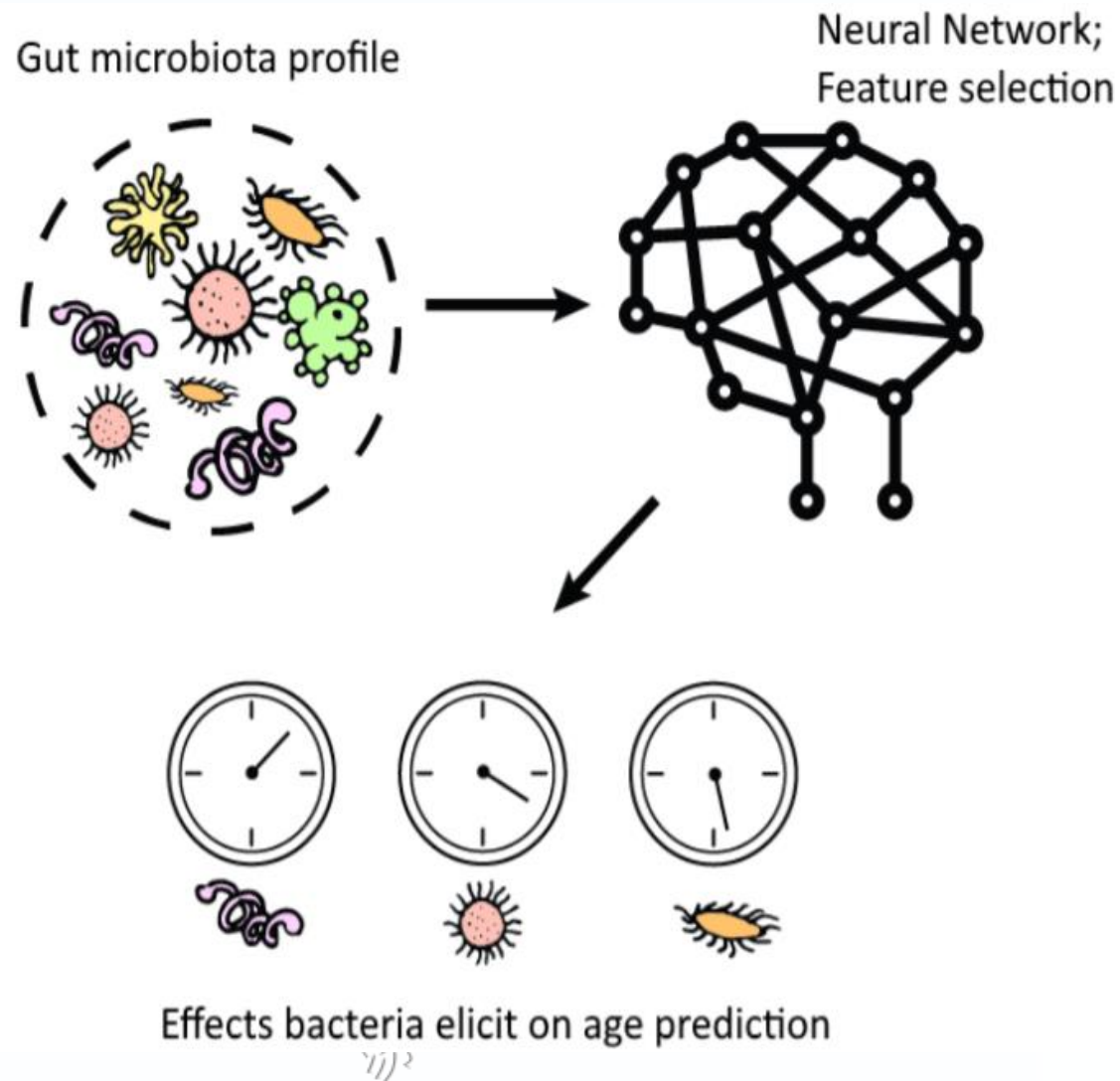


*Bacteroides* are the most common bacteria species found in the human intestinal tract. DENNIS KUNKEL  
MICROSCOPY/SCIENCE SOURCE

The bacteria in your gut may reveal your true age

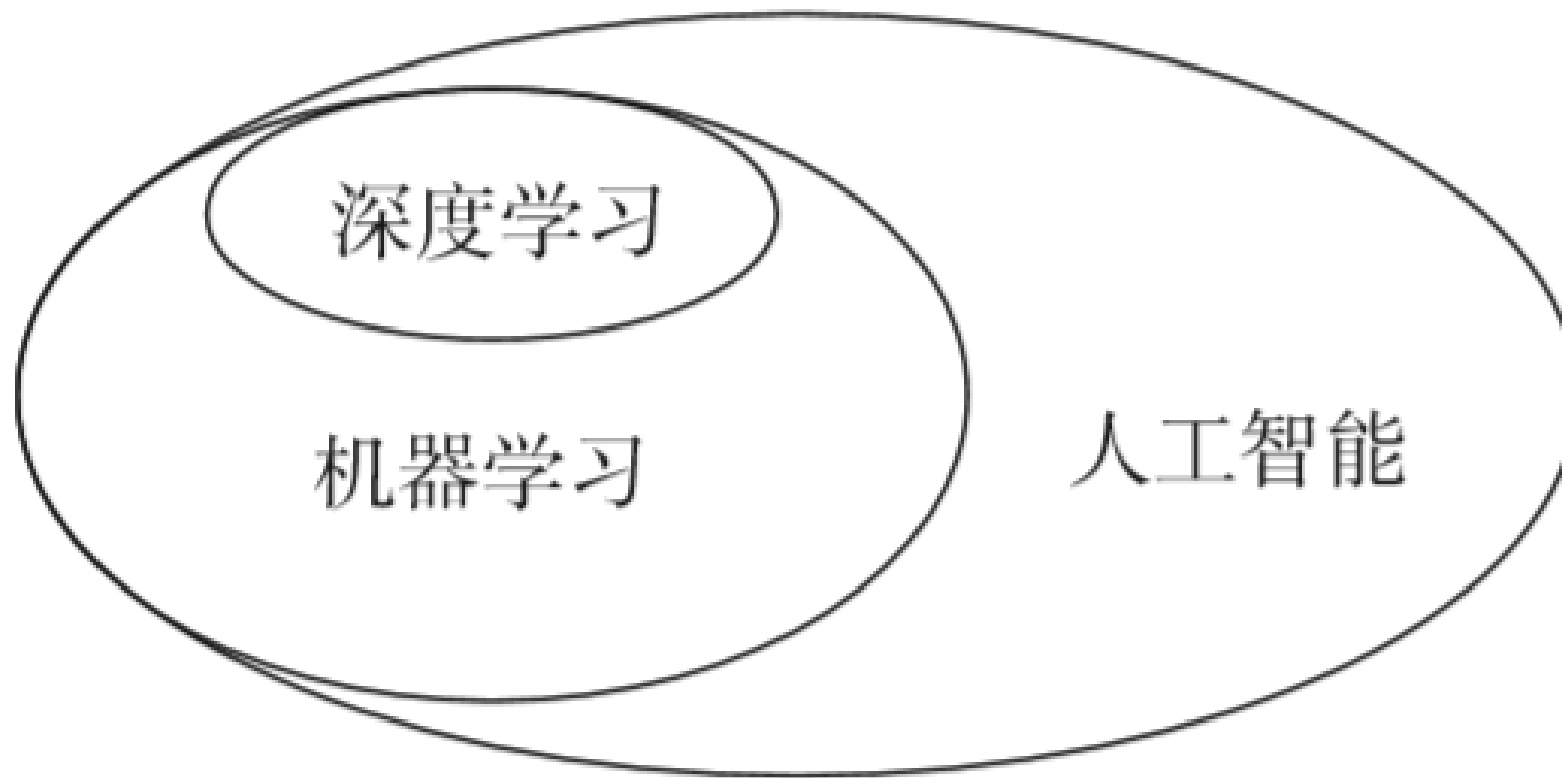
By Emily Mullin | Jan. 11, 2019, 11:50 AM

易汉博基因科技(北京)有限公司  
EHBIO Gene Technology (Beijing) co., LTD



Science: 肠道菌群又添一大功能，揭示你的真实年龄  
Nature Medicine同期8篇论文，聚焦人工智能在医学领域的应用

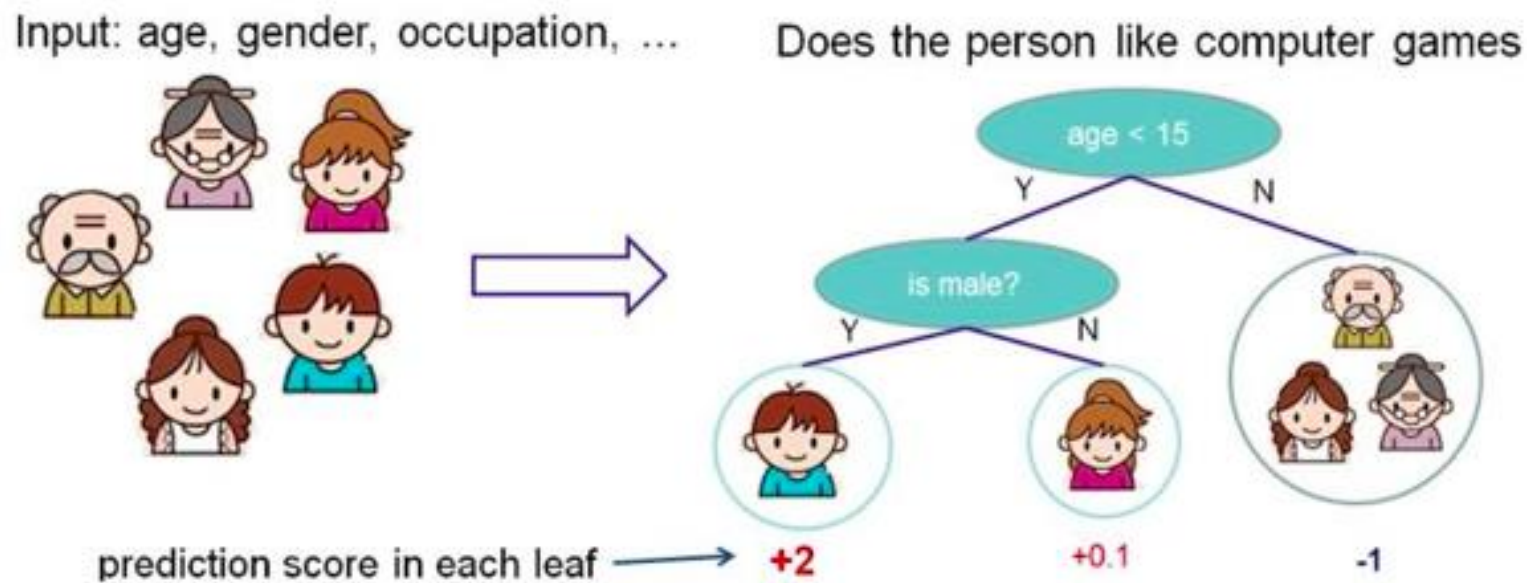
# 人工智能、机器学习、深度学习的关系



▲图1-1 人工智能、机器学习和深度学习的关系

# 机器学习：十大常用算法

- 决策树、随机森林、逻辑回归、SVM、朴素贝叶斯、K最近邻、K均值、Adaboost、神经网络、马尔可夫



决策树：根据一些特征(feature)进行分类，每个节点提一个问题，通过判断，将数据分为两类，再继续提问。这些问题是根据已有数据学习出来的，再投入新数据的时候，就可以根据这棵树上的问题，将数据划分到合适的叶子上。



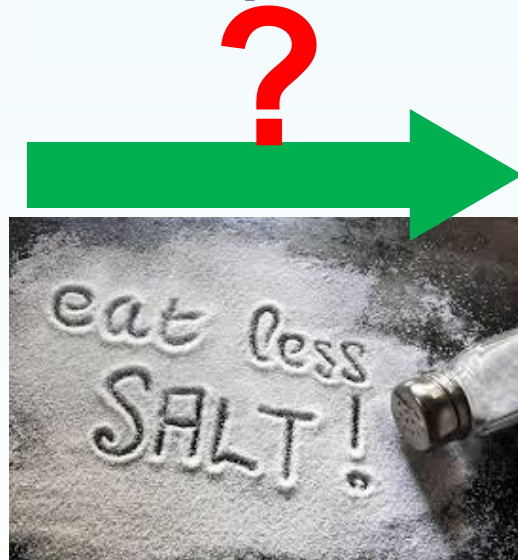


## 应用案例

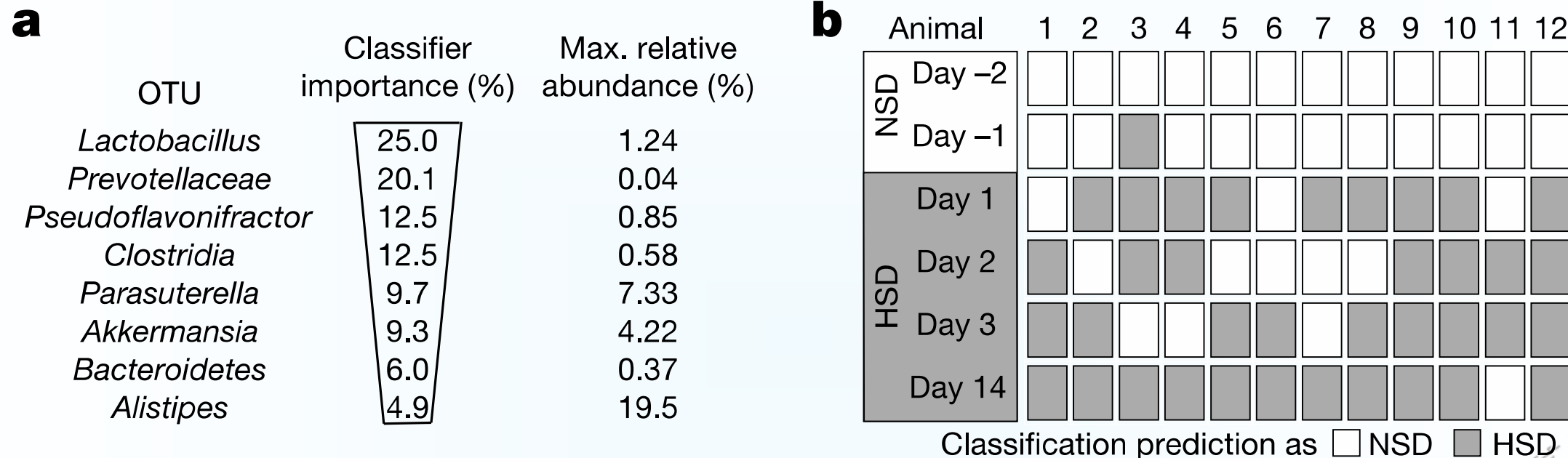


# Salt-responsive gut commensal modulates $T_H17$ axis and disease

Nicola Wilck<sup>1,2,3,4,5</sup>, Mariana G. Matus<sup>6,7</sup>, Sean M. Kearney<sup>6</sup>, Scott W. Olesen<sup>6</sup>, Kristoffer Forslund<sup>8</sup>, Hendrik Bartolomaeus<sup>1,2,3,4</sup>, Stefanie Haase<sup>9</sup>, Anja Mähler<sup>1,5</sup>, András Balogh<sup>1,2,3,4,5</sup>, Lajos Markó<sup>1,2,3,4,5</sup>, Olga Vvedenskaya<sup>3,10,11</sup>, Friedrich H. Kleiner<sup>1</sup>, Dmitry Tsvetkov<sup>1,2</sup>, Lars Klug<sup>1,5</sup>, Paul I. Costea<sup>8</sup>, Shinichi Sunagawa<sup>8,12</sup>, Lisa Maier<sup>13</sup>, Natalia Rakova<sup>1,9</sup>, Valentin Schatz<sup>14</sup>, Patrick Neubert<sup>14</sup>, Christian Frätzer<sup>15</sup>, Alexander Krannich<sup>5</sup>, Maik Gollasch<sup>1,2,3</sup>, Diana A. Grohme<sup>16</sup>, Beatriz F. Côrte-Real<sup>17</sup>, Roman G. Gerlach<sup>18</sup>, Marijana Basic<sup>19</sup>, Athanasios Typas<sup>13</sup>, Chuan Wu<sup>20</sup>, Jens M. Titze<sup>21</sup>, Jonathan Jantsch<sup>14</sup>, Michael Boschmann<sup>1,5</sup>, Ralf Dechend<sup>1,2,5</sup>, Markus Kleinewietfeld<sup>16,17,22</sup>, Stefan Kempa<sup>3,5,10</sup>, Peer Bork<sup>3,8,23,24</sup>, Ralf A. Linker<sup>9</sup>§, Eric J. Alm<sup>6</sup>§ & Dominik N. Müller<sup>1,2,3,4,5</sup>§

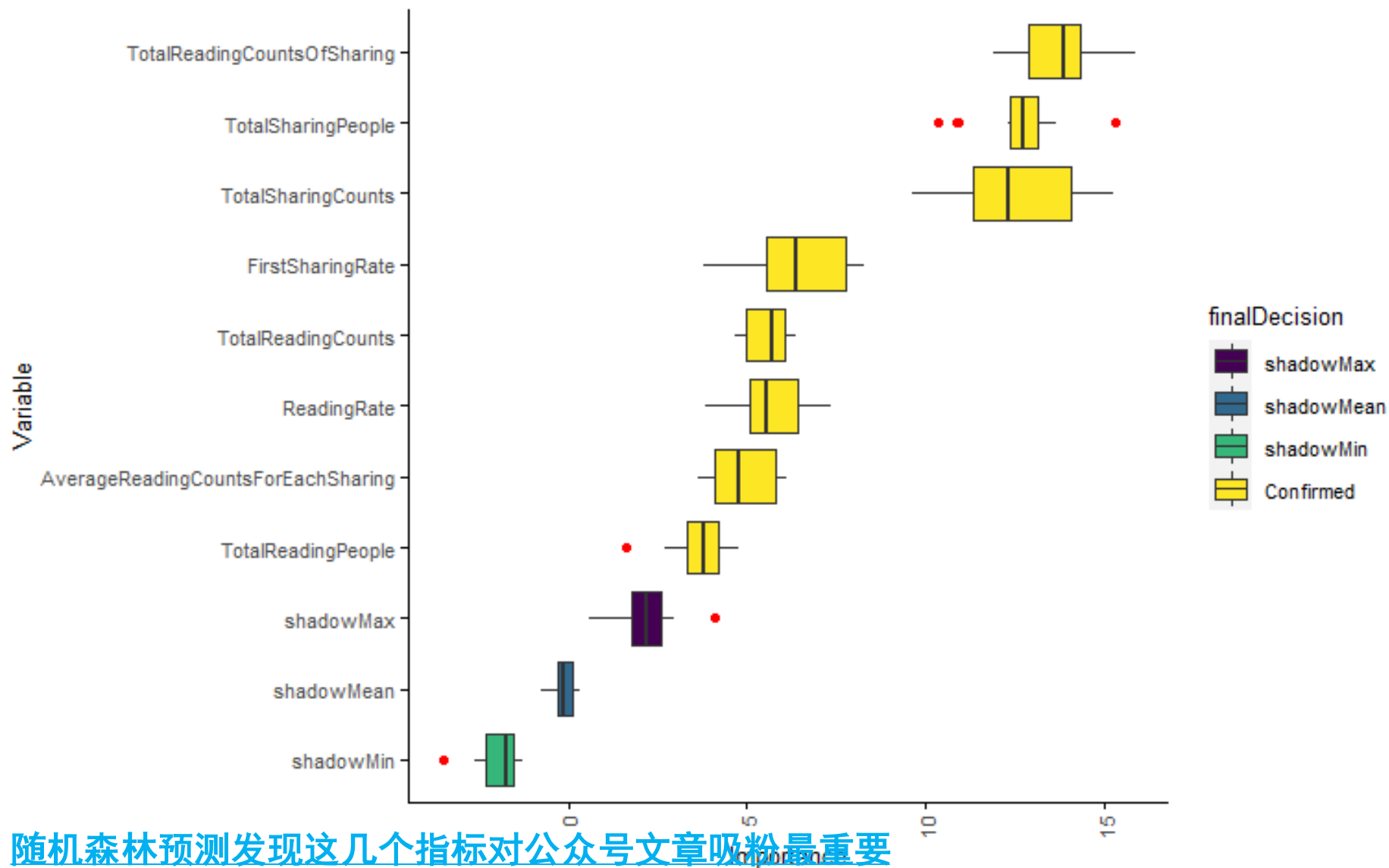


# 预测人高盐/低盐组生物标记(biomarker)+动物验证



- 随机森林和Adabooster分类，Adabooster结果重要性importance值更大，且对动物验证数据结果准确性也很高
- 软件：使用Python3的slime2计算决定分组特征的贡献率/重要性

# 随机森林预测：文章被分享才能快速涨粉



易基因组



## 决策树



# 决策树 - 把你做选择的过程呈现出来

- 如果有一个数据集如下，那么假如加入一个新的点，其x值为1，那么该点对应的最可能的颜色是什么？

| ##    | x   | color |
|-------|-----|-------|
| ## 1  | 0.0 | blue  |
| ## 2  | 0.5 | blue  |
| ## 3  | 1.1 | blue  |
| ## 4  | 1.8 | blue  |
| ## 5  | 1.9 | blue  |
| ## 6  | 2.0 | green |
| ## 7  | 2.5 | green |
| ## 8  | 3.0 | green |
| ## 9  | 3.6 | green |
| ## 10 | 3.7 | green |

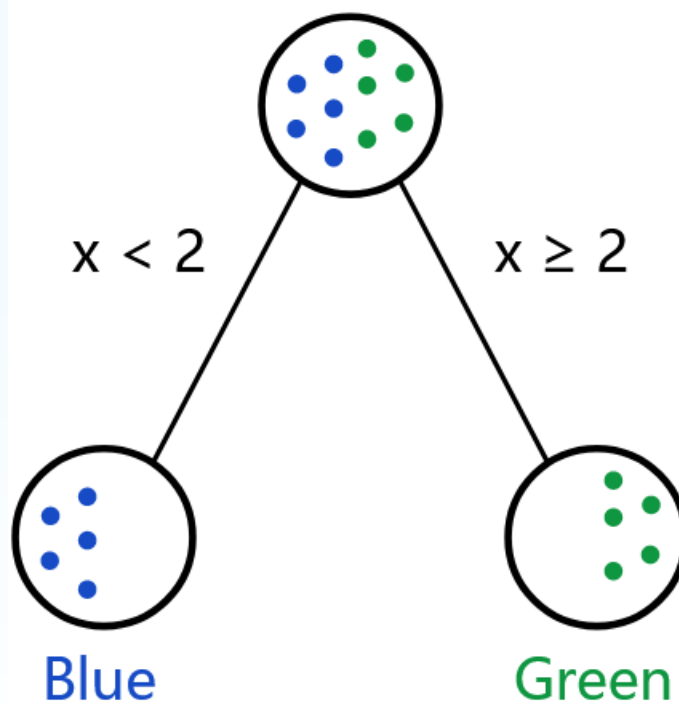
易生信 生信宝典 宏基因组



# 决策树 - 把你做选择的过程呈现出来

- 如果有一个数据集如下，那么假如加入一个新的点，其 $x$ 值为1，那么该点对应的最可能的颜色是什么？

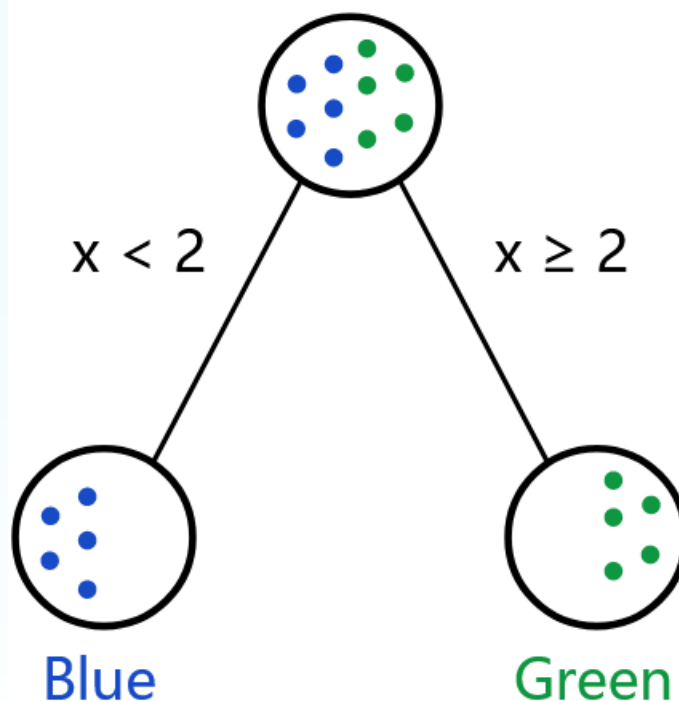
| ##    | x   | color |
|-------|-----|-------|
| ## 1  | 0.0 | blue  |
| ## 2  | 0.5 | blue  |
| ## 3  | 1.1 | blue  |
| ## 4  | 1.8 | blue  |
| ## 5  | 1.9 | blue  |
| ## 6  | 2.0 | green |
| ## 7  | 2.5 | green |
| ## 8  | 3.0 | green |
| ## 9  | 3.6 | green |
| ## 10 | 3.7 | green |



# 决策树 - 把你做选择的过程呈现出来

- 如果有一个数据集如下，那么假如加入一个新的点，其 $x$ 值为1，那么该点对应的最可能的颜色是什么？

| ##    | x   | color |
|-------|-----|-------|
| ## 1  | 0.0 | blue  |
| ## 2  | 0.5 | blue  |
| ## 3  | 1.1 | blue  |
| ## 4  | 1.8 | blue  |
| ## 5  | 1.9 | blue  |
| ## 6  | 2.0 | green |
| ## 7  | 2.5 | green |
| ## 8  | 3.0 | green |
| ## 9  | 3.6 | green |
| ## 10 | 3.7 | green |



决策树常用来回答这样的问题：给定一个带标签的数据集(标签这里对应我们的color列)，怎么来对新加入的数据集进行分类？

# 决策树 - 把你做选择的过程呈现出来

- 如果有一个数据集如下，那么假如加入一个新的点，其(x,y)值为(1.3, 1.9)，那么该点对应的最可能的颜色是什么？

|    | x   | y   | color |
|----|-----|-----|-------|
| 1  | 0.0 | 1.0 | blue  |
| 2  | 0.5 | 0.5 | blue  |
| 3  | 1.1 | 1.5 | blue  |
| 4  | 1.8 | 2.1 | red   |
| 5  | 1.9 | 2.8 | red   |
| 6  | 2.0 | 2.0 | green |
| 7  | 2.5 | 2.2 | green |
| 8  | 3.0 | 3.0 | green |
| 9  | 3.6 | 3.3 | green |
| 10 | 3.7 | 3.5 | green |

易生信 生信宝典 宏基因组

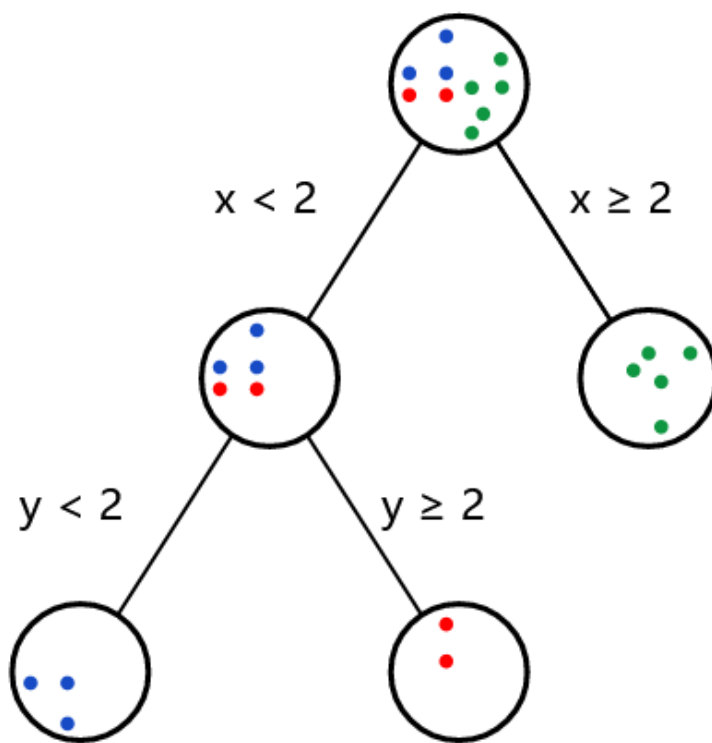




# 决策树 - 把你做选择的过程呈现出来

- 如果有一个数据集如下，那么假如加入一个新的点，其(x,y)值为(1.3, 1.9)，那么该点对应的最可能的颜色是什么？

|    | x   | y   | color |
|----|-----|-----|-------|
| 1  | 0.0 | 1.0 | blue  |
| 2  | 0.5 | 0.5 | blue  |
| 3  | 1.1 | 1.5 | blue  |
| 4  | 1.8 | 2.1 | red   |
| 5  | 1.9 | 2.8 | red   |
| 6  | 2.0 | 2.0 | green |
| 7  | 2.5 | 2.2 | green |
| 8  | 3.0 | 3.0 | green |
| 9  | 3.6 | 3.3 | green |
| 10 | 3.7 | 3.5 | green |



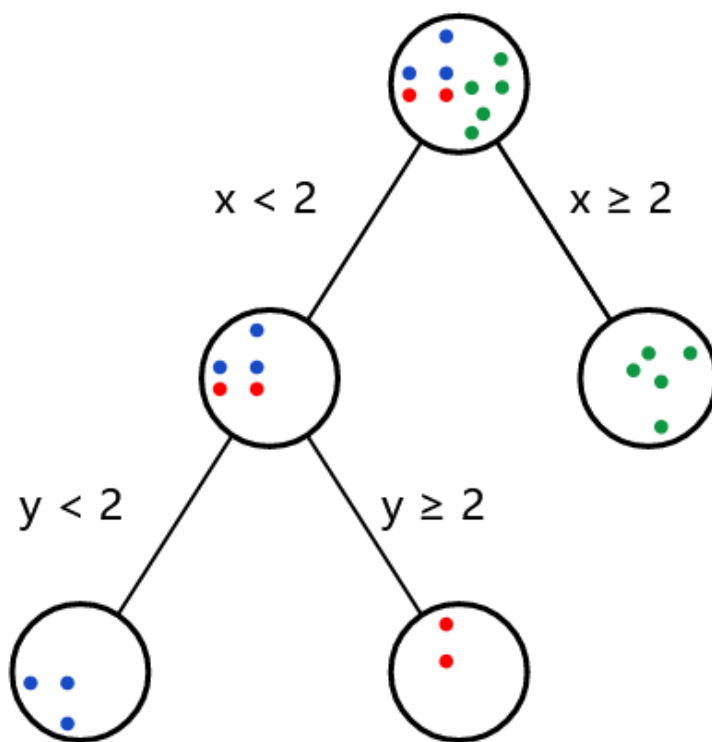
易生信，毕生缘；培训版权所有。

易生信 毕生缘 宏基因组

# 决策树 - 把你做选择的过程呈现出来

- 如果有一个数据集如下，那么假如加入一个新的点，其(x,y)值为(1.3, 1.9)，那么该点对应的最可能的颜色是什么？

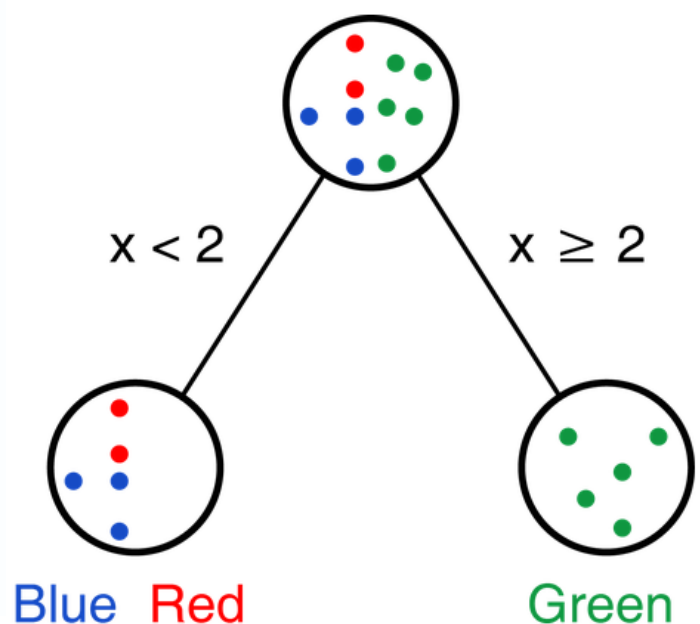
|    | x   | y   | color |
|----|-----|-----|-------|
| 1  | 0.0 | 1.0 | blue  |
| 2  | 0.5 | 0.5 | blue  |
| 3  | 1.1 | 1.5 | blue  |
| 4  | 1.8 | 2.1 | red   |
| 5  | 1.9 | 2.8 | red   |
| 6  | 2.0 | 2.0 | green |
| 7  | 2.5 | 2.2 | green |
| 8  | 3.0 | 3.0 | green |
| 9  | 3.6 | 3.3 | green |
| 10 | 3.7 | 3.5 | green |



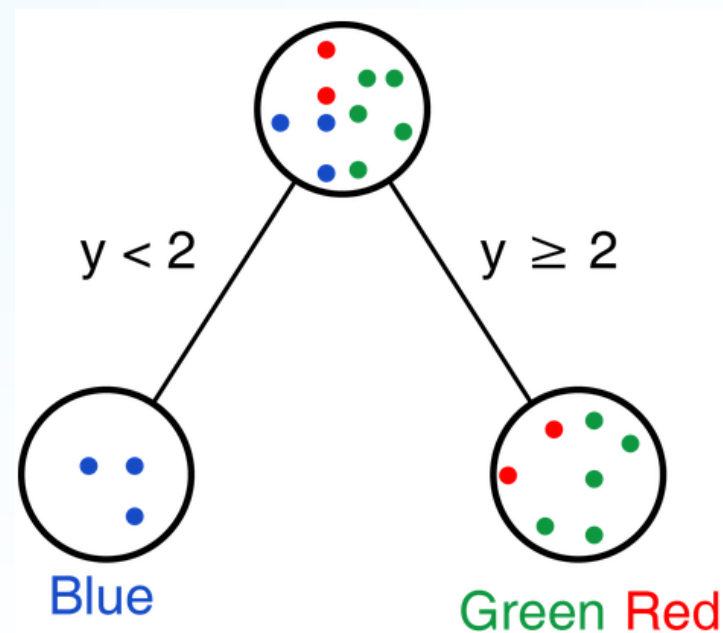
这就是决策树，也是我们日常推理问题的一般方式。

# 训练决策树 - 确定决策树的根节点，选择哪个变量和对应阈值选择多少能给数据做出最好的区分

先处理变量  $x$ ，选择阈值为 2 (为什么选 2，是不是有比 2 更合适阈值，我们后续再说)，则可获得如下决策树：



先处理变量  $y$ ，选择阈值为 2，则可获得如下决策树：



宏基因组

# 评价决策树的标准：Gini Impurity

- 在数据集中随机选择一个数据点，并随机分配给它一个数据集中存在的标签，分配错误的概率即为Gini impurity。
- 第一套数据集的初始状态，10个数据点，5个blue，5个green。从中随机选一个数据点，再随机选一个分类标签作为这个数据点的标签，分类错误的概率是多少？如下表，分类错误概率为 $0.25+0.25=0.5$ 。

| Event                      | Probability | Type  |
|----------------------------|-------------|-------|
| Pick Blue, Classify Blue   | 0.25        | TRUE  |
| Pick Blue, Classify Green  | 0.25        | FALSE |
| Pick Green, Classify Blue  | 0.25        | FALSE |
| Pick Green, Classify Green | 0.25        | TRUE  |





# 评价决策树的标准：Gini Impurity

- 我们再看第二套数据集，10个数据点，2个red，3个blue，5个green。从中随机选一个数据点，再随机选一个分类标签作为这个数据点的标签，分类错误的概率是多少？0.62。

|                            | Event Probability | Type  |
|----------------------------|-------------------|-------|
| Pick Blue, Classify Blue   | 0.09              | TRUE  |
| Pick Blue, Classify Green  | 0.15              | FALSE |
| Pick Blue, Classify Red    | 0.06              | FALSE |
| Pick Green, Classify Blue  | 0.15              | FALSE |
| Pick Green, Classify Green | 0.25              | TRUE  |
| Pick Green, Classify Red   | 0.10              | FALSE |
| Pick Red, Classify Blue    | 0.06              | FALSE |
| Pick Red, Classify Green   | 0.10              | FALSE |
| Pick Red, Classify Red     | 0.04              | TRUE  |

[微信推文](#)



# Gini Impurity计算公式

假如我们的数据点共有 $C$ 个类， $p(i)$ 是从中随机拿到一个类为 $i$ 的数据，Gini Impurity 计算公式为：

$$G = \sum_{i=1}^C p(i) * (1 - p(i))$$

对第一套数据集，10个数据点，5个 blue，5个 green。从中随机选一个数据点，再随机选一个分类标签作为这个数据点的标签，分类错误的概率是多少？错误概率为  $0.25+0.25=0.5$ 。

$$\begin{aligned}\text{错误概率 } G &= p(\text{blue}) * (1 - p(\text{blue})) + p(\text{green}) * (1 - p(\text{green})) \\ &= 5/10 * (1 - 5/10) + 5/10 * (1 - 5/10) \\ &= 0.5\end{aligned}$$

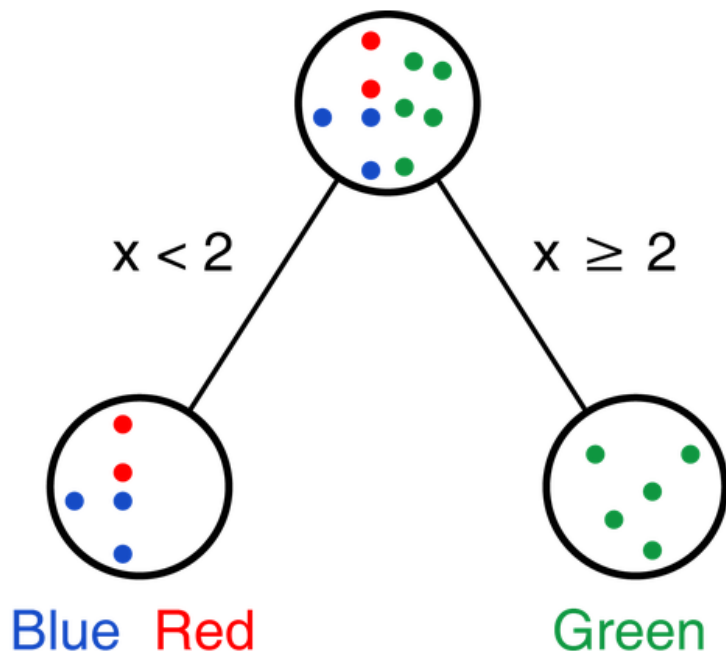
对第二套数据集，10个数据点，2个 red，3个 blue，5个 green。从中随机选一个数据点，再随机选一个分类标签作为这个数据点的标签，分类错误的概率是多少？ $0.62$ 。

$$\begin{aligned}\text{错误概率 } G &= p(\text{blue}) * (1 - p(\text{blue})) + p(\text{green}) * (1 - p(\text{green})) + p(\text{red}) * (1 - p(\text{red})) \\ &= 3/10 * (1 - 3/10) + 5/10 * (1 - 5/10) + 2/10 * (1 - 2/10) \\ &= 0.21 + 0.25 + 0.16 = 0.62\end{aligned}$$



# 比较两种决策的Gini impurity, 确定根节点分类变量

1. 我们可以先处理变量  $x$ , 选择阈值为 2, 则可获得如下分类:



每个分支的 Gini Impurity 可以如下计算:

$$G_{left} = 3/5 * (1 - 3/5) + 2/5 * (1 - 2/5) = 0.48$$

$$G_{right} = 5/5 * (1 - 5/5) = 0$$

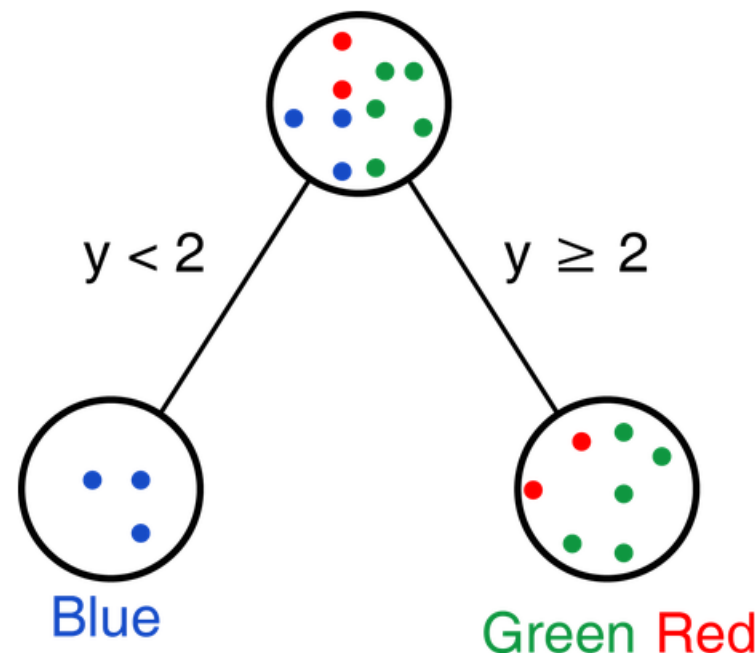
当前决策的 Gini impurity 需要对各个分支包含的数据点的比例进行加权, 即

$$G_{x-root} = G_{left} * \frac{C_{blue} + C_{red}}{C_{total}} + G_{right} * \frac{C_{green}}{C_{total}}$$

$$= 0.48 * (2 + 3)/10 + 0$$

$$= 0.24$$

2. 我们也可以先处理变量  $y$ , 选择阈值为 2, 则可获得如下分类:



每个分支的 Gini Impurity 可以如下计算:

$$G_{left} = 3/3 * (1 - 3/3) = 0$$

$$G_{right} = 2/7 * (1 - 2/7) + 5/7 * (1 - 5/7) = 0.41$$

当前决策的 Gini impurity 需要对各个分支包含的数据点的比例进行加权, 即

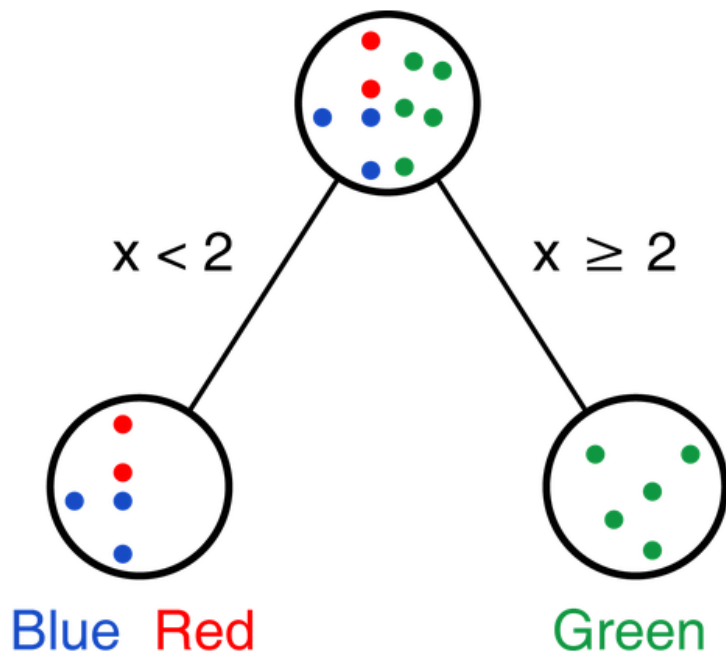
$$G_{y-root} = G_{left} * \frac{C_{blue}}{C_{total}} + G_{right} * \frac{C_{green} + C_{red}}{C_{total}}$$

$$= 0 + 0.41 * (2 + 5)/10$$

$$= 0.29$$

# 比较两种决策的Gini impurity, 确定根节点分类变量

1. 我们可以先处理变量  $x$ , 选择阈值为 2, 则可获得如下分类:



每个分支的 Gini Impurity 可以如下计算:

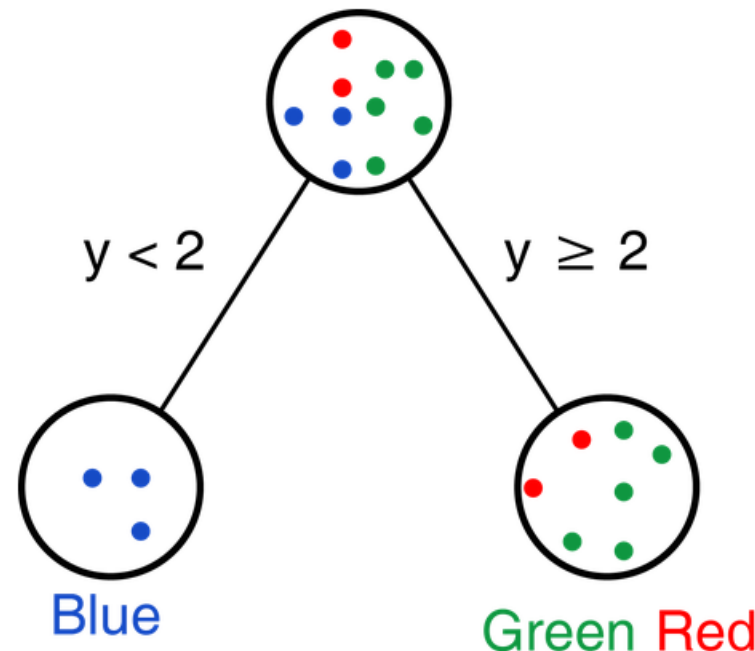
$$G_{left} = 3/5 * (1 - 3/5) + 2/5 * (1 - 2/5) = 0.48$$
$$G_{right} = 5/5 * (1 - 5/5) = 0$$

当前决策的 Gini impurity 需要对各个分支包含的数据点的比例进行加权, 即



$$G_{x-root} = G_{left} * \frac{C_{blue} + C_{red}}{C_{total}} + G_{right} * \frac{C_{green}}{C_{total}}$$
$$= 0.48 * (2 + 3)/10 + 0$$
$$= 0.24$$

2. 我们也可以先处理变量  $y$ , 选择阈值为 2, 则可获得如下分类:



每个分支的 Gini Impurity 可以如下计算:

$$G_{left} = 3/3 * (1 - 3/3) = 0$$
$$G_{right} = 2/7 * (1 - 2/7) + 5/7 * (1 - 5/7) = 0.41$$

当前决策的 Gini impurity 需要对各个分支包含的数据点的比例进行加权, 即

$$G_{y-root} = G_{left} * \frac{C_{blue}}{C_{total}} + G_{right} * \frac{C_{green} + C_{red}}{C_{total}}$$
$$= 0 + 0.41 * (2 + 5)/10$$
$$= 0.29$$



# 训练决策树 - 确定根节点的分类阈值

- Gini impurity可以用来判断每一步最合适的决策分类方式，那么怎么确定最优的分类变量和分类阈值呢？
- 最粗暴的方式是，我们遍历每个变量的每个可能的阈值来进行决策分类，选择具有最低Gini impurity值的分类组合。这不是最快速的解决问题的方式，但是最容易理解的方式。

易生信 生信宝典 宏基因组



# 变量X使用不同阈值决策计算的Gini impurity

| Variable | Threshold | Left_branch                  | Right_branch                 | Gini_impurity |
|----------|-----------|------------------------------|------------------------------|---------------|
| x        | 1.95      | blue x 3; red x 2            | green x 5                    | 0.24          |
| x        | 1.45      | blue x 3                     | green x 5; red x 2           | 0.285712      |
| x        | 1.85      | blue x 3; red x 1            | green x 5; red x 1           | 0.316668      |
| x        | 2.25      | blue x 3; green x 1; red x 2 | green x 4                    | 0.366666      |
| x        | 0.8       | blue x 2                     | blue x 1; green x 5; red x 2 | 0.425         |
| x        | 2.75      | blue x 3; green x 2; red x 2 | green x 3                    | 0.457142      |
| x        | 3.3       | blue x 3; green x 3; red x 2 | green x 2                    | 0.525         |
| x        | 0.25      | blue x 1                     | blue x 2; green x 5; red x 2 | 0.533331      |
| x        | 3.65      | blue x 3; green x 4; red x 2 | green x 1                    | 0.577782      |

易生信

# 变量X使用不同阈值决策计算的Gini impurity

| Variable | Threshold | Left_branch                  | Right_branch                 | Gini_impurity |
|----------|-----------|------------------------------|------------------------------|---------------|
| x        | 1.95      | blue x 3; red x 2            | green x 5                    | 0.24          |
| x        | 1.45      | blue x 3                     | green x 5; red x 2           | 0.285712      |
| x        | 1.85      | blue x 3; red x 1            | green x 5; red x 1           | 0.316668      |
| x        | 2.25      | blue x 3; green x 1; red x 2 | green x 4                    | 0.366666      |
| x        | 0.8       | blue x 2                     | blue x 1; green x 5; red x 2 | 0.425         |
| x        | 2.75      | blue x 3; green x 2; red x 2 | green x 3                    | 0.457142      |
| x        | 3.3       | blue x 3; green x 3; red x 2 | green x 2                    | 0.525         |
| x        | 0.25      | blue x 1                     | blue x 2; green x 5; red x 2 | 0.533331      |
| x        | 3.65      | blue x 3; green x 4; red x 2 | green x 1                    | 0.577782      |

易生信

# 重复迭代继续决策后续节点

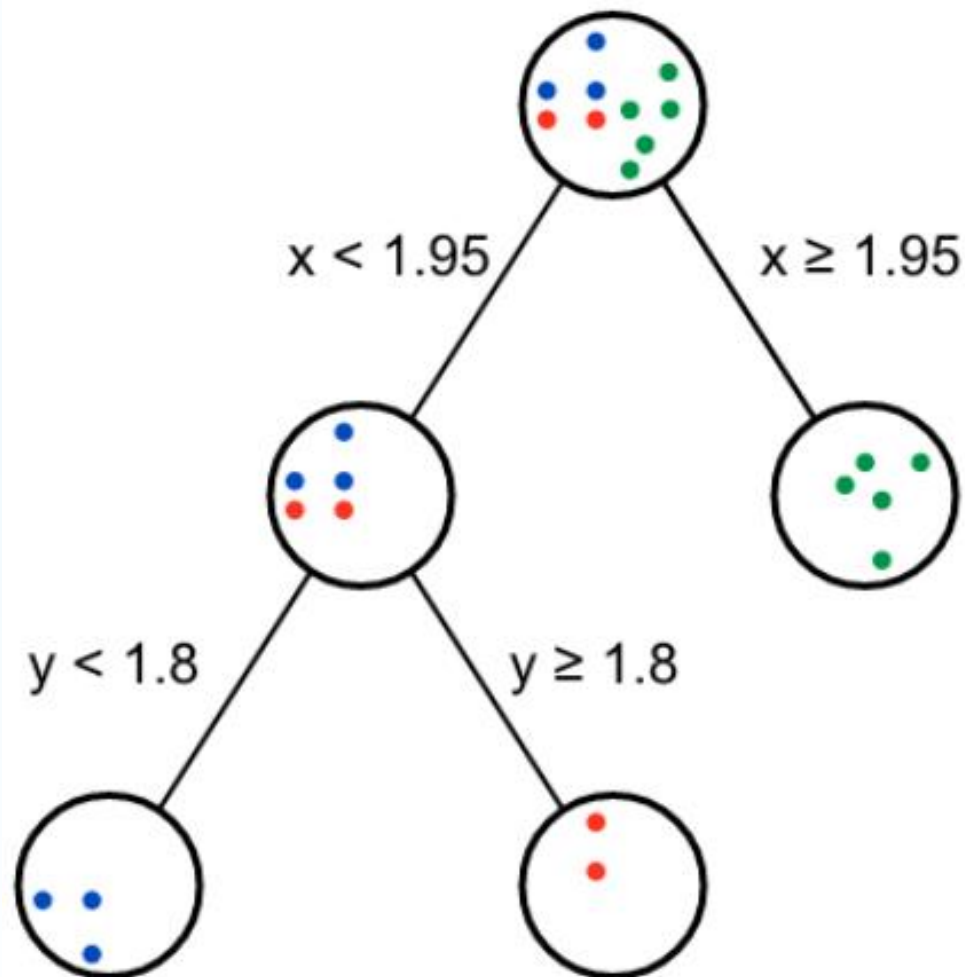
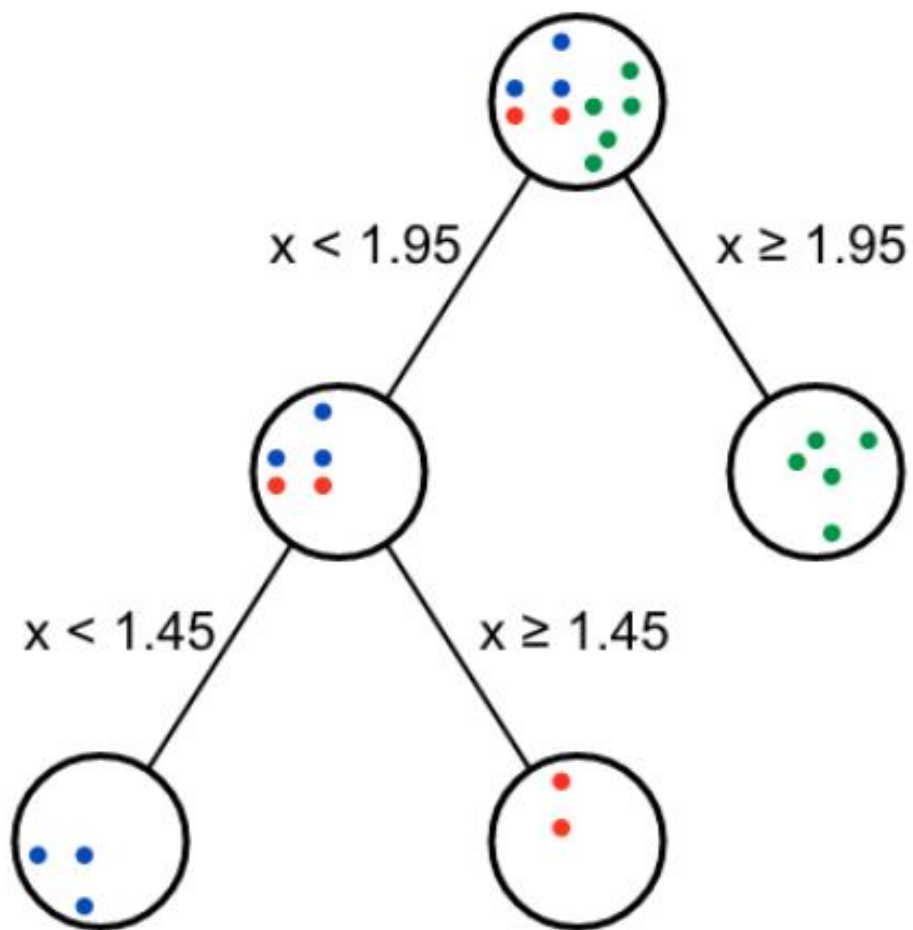
- 第一个决策节点找好了，后续再找其它决策节点。如果分配到某个分支的点从属于多个**class**，则递归决策。
- 递归决策终止的条件是：
  - 再继续向下分支不会降低**Gini impurity**
  - 该分支的数据点属于同一分类组 ( $\text{Gini impurity} = 0$ )

易生信 生信宝典 宏基因组





# 获得最终决策树

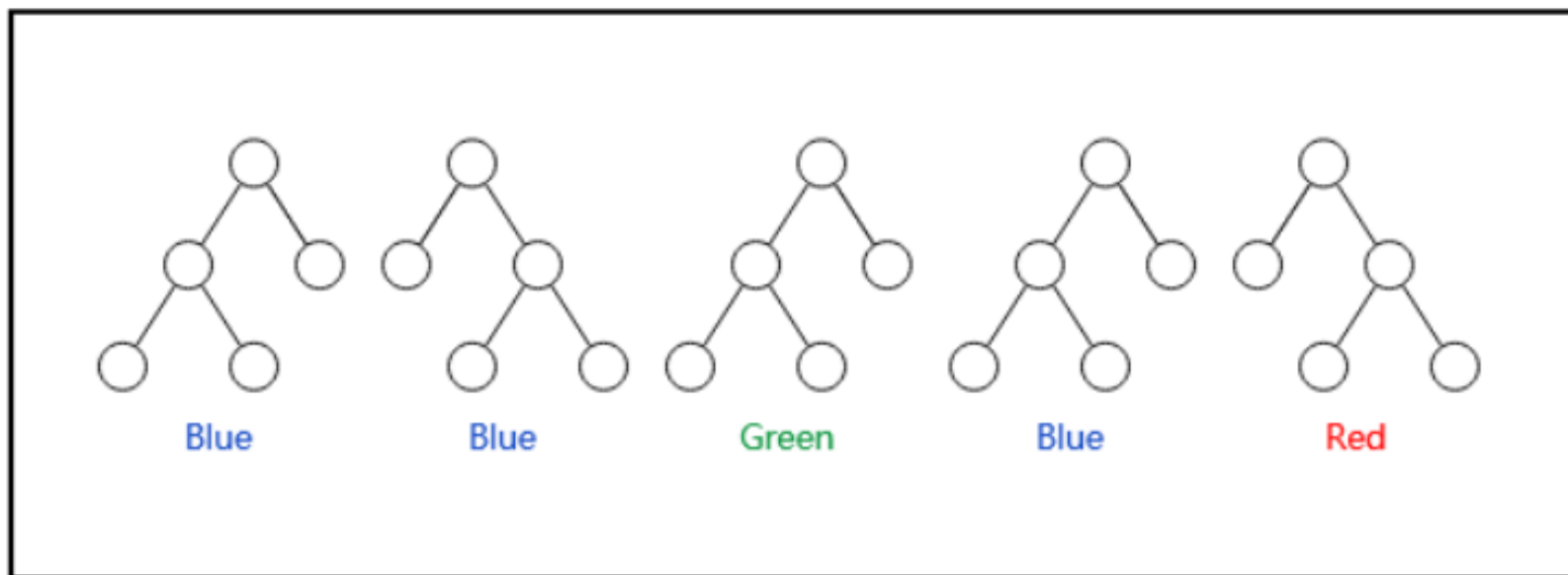


基因组



## 随机森林

# 随机森林实际是一堆决策树的组合（正如其名，树多了就是森林了）



Blue

Bagged Decision Trees predicting color

宏基因组

# 随机森林 – 鉴定分类Marker

- 随机森林 是一种基于决策树的高效的机器学习算法，可以用于对样本进行分类或回归分析。它属于非线性分类器，因此可以挖掘变量之间复杂的非线性的相互依赖关系。通过随机森林分析，可以找出能够区分两组样本间差异的关键成分(基因/代谢物ASV/等)。

易生信 生信宝典 宏基因组





# 随机森林是基于集体智慧的一个机器学习算法

- 准确率最好的算法之一
- 对大数据集也能有效运算 (得益于第2次随机)
- 可以处理高维数据, 且无需降维 (得益于第2次随机)
- 能够评估各个变量在分类问题中的重要性 (数据置换评估)
- 自带训练集和测试集, 在构建模型时即可评估分类错误 (得益于第1次随机)
- 即便在大量缺失值情况下, 也可准确进行缺失值插补
- 产生的树可以保存用于后续预测等使用
- 随机森林不会过拟合, 只要内存足够, 可以设置更多的树。一个 50,000 cases X 100 variables 的数据集, 普通电脑可以在3分钟构建100棵树
- 可以平衡不同分组样品多少引入的预测错误率
- 可以鉴定异常样品、可视化样品的相似性关系



# 随机森林的2次随机

假设有一个数据集包含  $n$  个样品,  $p$  个变量, 也就是一个  $n \times p$  的矩阵, 采用下面的算法获得一系列的决策树:

1. 从数据集中有放回地取出  $n$  个样品作为训练集, 训练 1 棵决策树。假如数据集有 6 个样品  $[a,b,c,d,e,f]$ , 第一次有放回地取出 6 个样品可能是  $[a,a,c,d,d,e]$ , 第二次有放回地取出 6 个样品可能是  $[a,c,c,d,e,e]$ 。每棵树的训练集是不完全相同的, 但样品数目是一样的, 因为每个训练集内部包含重复样本。这一步称为 Bagging (Bootstrap aggregating) 自助聚合。
2. 每棵决策树构建时, 通常随机抽取  $m$  ( $m < p$ ) 个变量进行每个节点决策变量的选择,  $m$  在选择每一层节点时不变。其意义是每次进行决策时只在随机选择的  $m$  个变量而不是全部变量中选择最好的分类变量, 也是为了增强随机性。
3. 每棵决策树野蛮生长, 不剪枝。
4. 重复第 1,2,3 步  $t$  次, 获得  $t$  棵决策树。

通过聚合  $t$  棵决策树做出最终决策:

- 分类问题: 选择  $t$  棵决策树中支持最多的类作为最终分类 (服从多数, majority vote)
- 回归问题: 计算  $t$  棵决策树预测的数值的均值作为最终预测值



# 随机选取的变量数m怎么确定？

在数据科学中，很大数目的相对**不相关**的模型的群体决策**优于**任何单个模型的决策。

随机森林的错误率取决于两个因素：

- 不同树之间的相关性越高，则错误率越大。这要求  $m$  尽可能小。
- 每个决策树的分类强度越大，则错误率越小。这要求  $m$  尽可能大。

$m$  通常为  $\sqrt{\text{num\_variable}}$  (用于分类问题)或  $\frac{\text{num\_variable}}{3}$  (用于回归问题)。也可以通过 `oob (out-of-bag) error rate` (自助聚合错误率) 进行迭代调参选择分类效果最好的  $m$ 。

## 机器学习算法-随机森林之理论概述



# oob (out-of-bag) error rate (聚合错误率)

1. 每一个决策树构建时使用到的样本是不同的。因为是有放回地随机采样，在构建每棵树时，大约  $1/3$  的样品不会被用到（验证集）。
2. 这些没有被抽样到用于构建决策树的样品会用来测试该决策树的分类能力。理论上每个样品在  $1/3$  的决策树中可作为验证样品获得一个分类结果和分类错误率。
3. 聚合每个没有被抽样到的样品在所有其不被用于训练集构建的决策树的分类结果作为该样品的最终分类结果 (Majority vote, 可以理解为是一个小的随机森林对该样品的分类结果)，与数据的原始分类一致则是分类正确，否则是分类错误。
4. 所有分类错误的样品除以总样品即为构建的随机森林模型的错误率。(  $i$  表示第  $i$  棵树)

$$OOB_{case_i} = \begin{cases} 1 & \text{if predicted class for case } i \text{ is wrong} \\ 0 & \text{if predicted class for case } i \text{ is right} \\ 0 & \text{case } i \text{ always in train sample} \end{cases}$$

$$OOB_{tree} = \frac{\sum_{i=1}^n OOB_{case_i}}{n}$$

# 随机森林分类变量重要性

对于每一棵随机决策树，统计其分类 OOB 数据集中所有样品正确分类的次数 (  $O$  )。随机排列 OOB 数据集中变量  $m$  的值 (打乱变量  $m$  在所有样品的原始值)，再用该决策树分类，统计所有样品被正确分类的次数 (  $P$  )。这两个次数的差值 (  $O-P$  ) 即为变量  $m$  在单棵决策树的分类重要性得分 (  $CS$ , *classification score* )，其在所有分类树中的均值即为该变量的整体重要性得分 (  $ACS$ , *avraged classification score* )。

$$CS_m^{tree\_i} = Count_{rightly\_classified} - Count_{rightly\_classified\_for\_random\_permutation}$$

$$ACS_m = \frac{\sum_{i=1}^t CS_m^{tree\_i}}{t}$$

$$ACS_m^{zscore} = \frac{ACS_m}{sd(CS_m^{tree\_1}, CS_m^{tree\_2}, \dots, CS_m^{tree\_t})}$$

通过大量数据测试发现，同一个变量不同决策树获得的  $CS$  的相关性很低，可以认为是彼此独立的。随后按照常规方式计算所有  $CS$  的标准差， $ACS/sd(CS)$  获得  $Z-score$  值。假设  $Z-score$  服从正态分布，即可根据  $Z-score$  估计该变量的重要性程度。（后面的特征变量选择还有其他方式可进行筛选）

如果变量数目很多，也可只在第一次用所有变量评估，后续只评估第一次选出的最重要的一部分变量。

但是如果两个变量相关性比较强，随机置换一个变量时其对分类的影响效果有限，会高估变量的重要性。



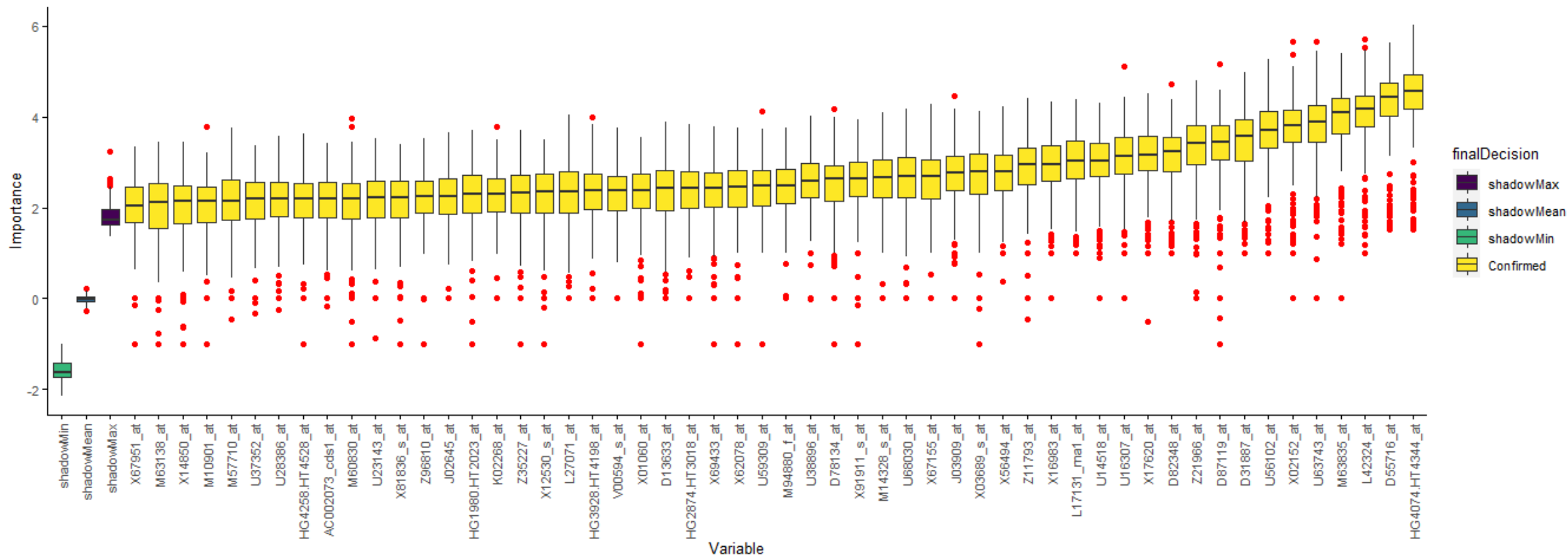


# Boruta算法进行特征选择

- **Boruta**得名于斯拉夫神话中的树神，可以识别所有对分类或回归有显著贡献的变量。其核心思想是统计比较数据中真实存在的特征变量与随机加入的变量（也称为影子变量）的重要性。
- 初次建模时，把原始变量拷贝一份作为影子变量。
- 原始变量的值随机化后作为对应影子变量的值(随机化就是打乱原始变量值的顺序)。
- 使用随机森林建模并计算每个变量的重要性得分。
- 对于每一个真实特征变量，统计检验其与所有影子变量的重要性最大值的差别。重要性显著高于影子变量的真实特征变量定义为重要。重要性显著低于影子变量的真实特征变量定义为不重要。
- 所有不重要的变量和影子变量移除。
- 基于新变量构成的数据集再次重复刚才的建模和选择过程，直到所有变量都被分类为重要或不重要，或达到预先设置的迭代次数。



# Boruta算法进行特征选择





## 模型评估指标

# 准确率不适合不平衡数据

- 分类问题评估指标有：准确率 (Accuracy)、精准率 (Precision)、灵敏度 (Sensitivity)、ROC 曲线、AUC 值。

|                                | Predicted_as_DLBCL | Predicted_as_FL |
|--------------------------------|--------------------|-----------------|
| Originally_classified_as_DLBCL | 56                 | 2               |
| Originally_classified_as_FL    | 8                  | 11              |

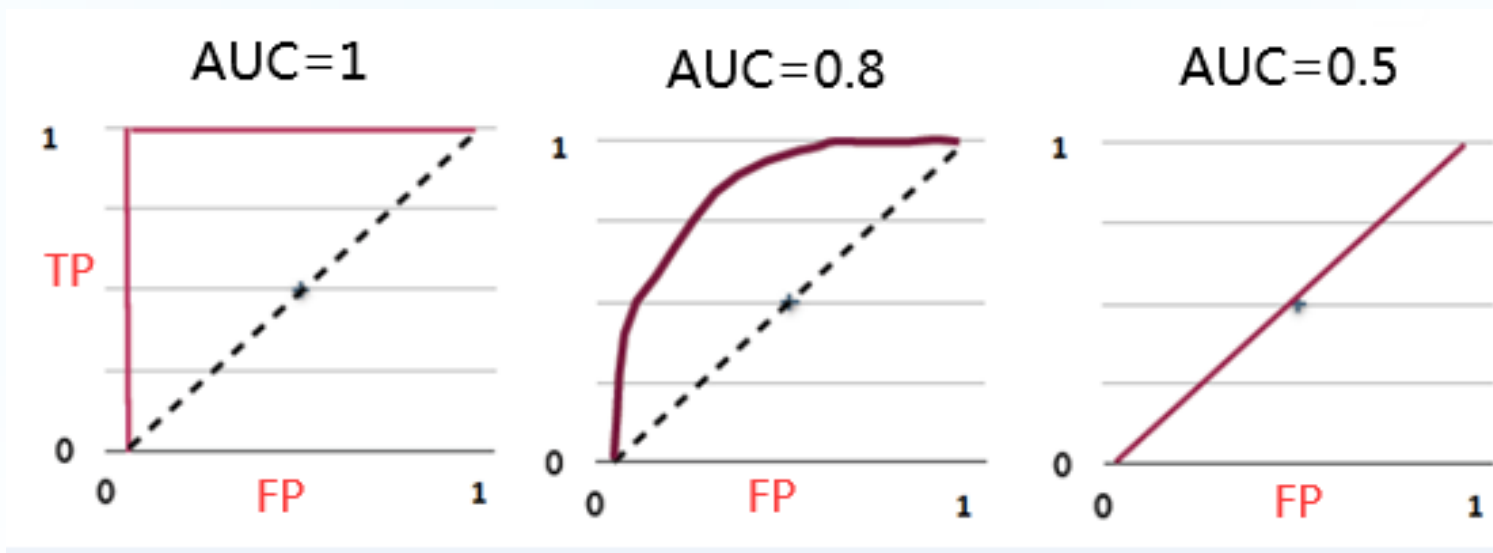
$$\text{准确率} = \frac{56 + 11}{56 + 2 + 8 + 11} = 0.870$$

准确率可以判断总的正确率，但在各个分组样本数目差别较大时不能作为一个很好的评价标准。比如上面 confusion matrix 中，所有样品全部预测为 DLBCL 时准确率可达 75.3%。这一不负责任的预测方式 (也称为 No information rate) 跟预测模型的准确率 ( 87.0% ) 相差不是太大。



# ROC (Receiver Operating Characteristic)特征曲线

- 横轴为假阳性率，纵轴为真阳性率的一条曲线。这条曲线越陡越好，说明在较低的假阳性率时可以获得较高的真阳性率。
- 一般通过曲线下面积AUC (Area under curve)评估一个ROC曲线的好坏。一般模型的AUC值在0.5-1之间，值越大越好。

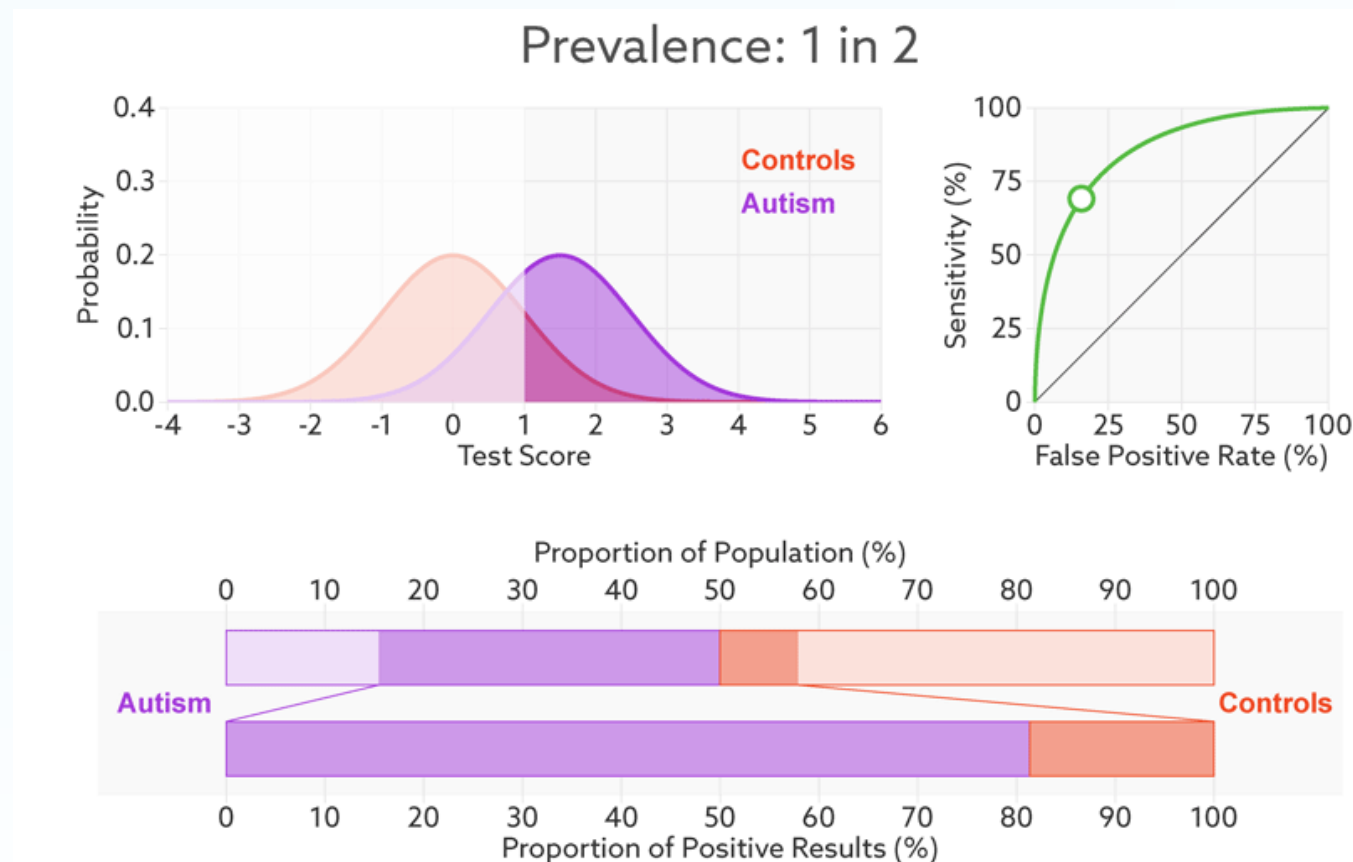


宏基因组  
转录组



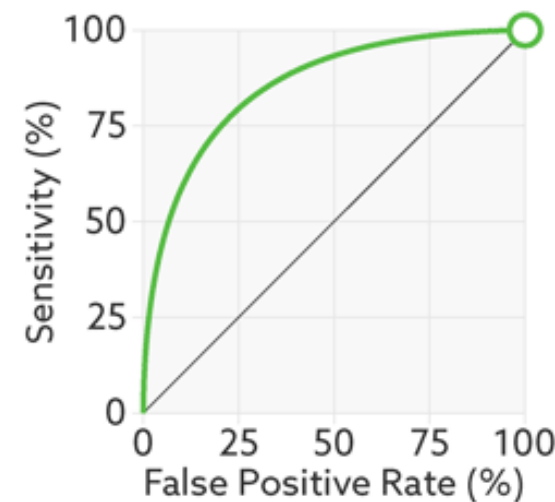
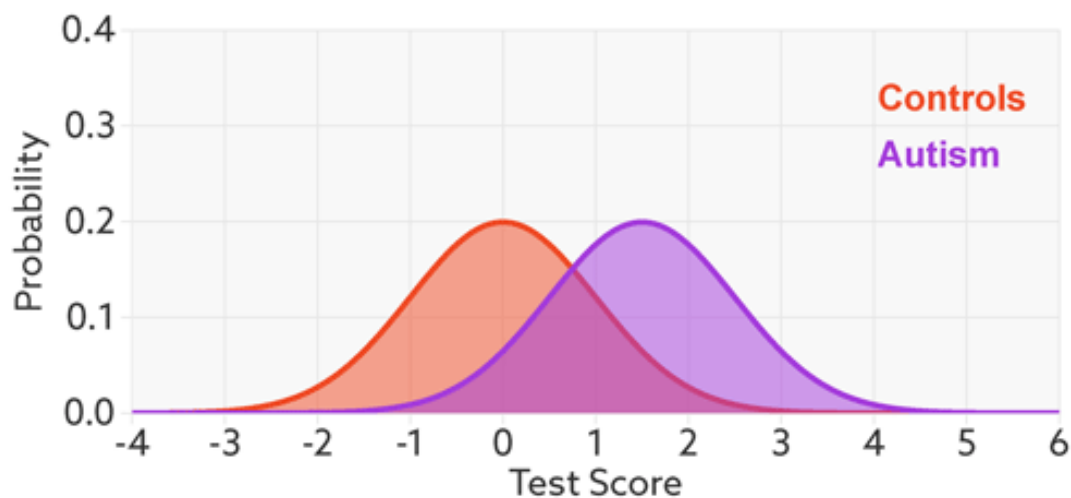
# ROC曲线不因样品不均衡而受影响

- 当样品中自闭症病人的比例从1/2降到1/68时，精准率从81%降到6%



宏基因组

# 模型阈值的改变只会改变真阳性率（灵敏度）和假阳性率，但是ROC曲线却不会变化



模型是模型， 阈值是阈值

# 分类模型、预测模型和决策指定

- 在很多决策应用中，分类模型代表着一个“不成熟”的决定，它组合了预测模型和决策制定，但剥夺了决策者对错误决定带来的损失的控制权（如随机森林中的服从**大多数**原则，**51**棵树预测结果为患病**49**棵树预测结果为正常与**91**棵树预测结果为患病**9**棵树预测结果为正常返回的结果都是患病）。
- 如果采样标准或损失/收益规（在预测疾病时，更看重敏感性而非假阳性）则发生改变，分类模型也需要相应的改变。而预测模型是与决策分开的，可用于任何决策制定。

值得思考，机器学习模型做出的决策是你想要的吗？





## 训练集、验证集、测试集

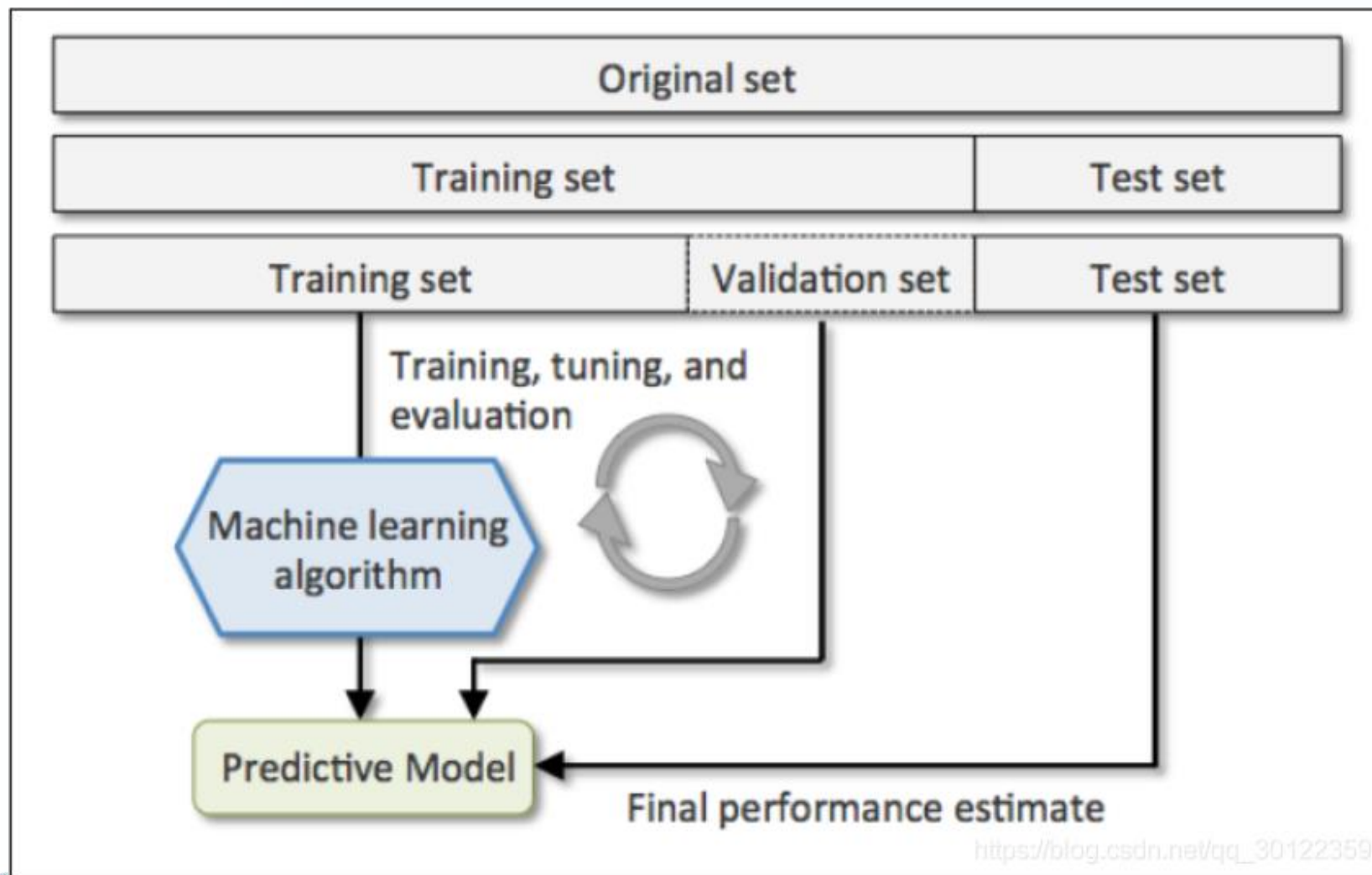
# 训练集、验证集、测试集

- 通常在有了一套数据时，需要拆分为训练集、测试集。数据集一般按比例**8:2**，**7:3**，**6:4**等分为训练集和测试集。如果数据集很大，测试集不需要完全按比例分配，够用就好。
- 测试集完全不用于训练模型，是模型从未见过的数据。训练集在训练模型时可能会出现过拟合问题（过拟合指模型可以很好的匹配训练数据但预测其它数据时效果不好），所以一般需要在训练集中再分出一部分作为验证集，用于评估模型的训练效果和调整模型的超参数(hyperparameter)，如随机森林的**ntree**和**mtry**。





# 训练集、验证集、测试集



训练集等同于上课学习知识，  
验证集等同于课后测验检测学习效果并且查漏补缺。  
测试集是期末考试评估这个模型到底怎样，考试之前不能泄题。



## 交叉验证

# 简单交叉验证

- 简单交叉验证是从训练集中选择一部分（如**70%**）作为新训练集，剩余一部分（如**30%**）作为验证集。基于此选出最合适模型或最优模型参数。然后再用全部的训练集训练该选择的模型。其在一定程度上可以避免过拟合事件的发生。
- 但基于**70%**训练集评估的最优模型是否等同于基于所有训练集的最优模型存疑。不同大小的验证集获得的评估结果差异较大，单纯按比例划分会导致无法选到最好的模型。
- 另外如果训练集较小再如此分割后，训练集数目就更少了，不利于获得较好的训练模型。

# K-fold交叉验证

其目的是重复使用原始训练集中的数据，每一个样品都会被作为训练集参与训练模型，也会作为测试集参与评估模型。最大程度地利用了全部数据，当然也消耗了更多计算时间。

其操作过程如下：

1. 将训练集分成  $K$  份（如果训练集有  $m$  个样本，则每一份子集有  $m/K$  的样本；若不能整除其中一个或数个集合会样本少1个。）
2. 对于每一个模型（如随机森林中不同的  $mtry$  值， $mtry=2$ ， $mtry=10$  时分别会构建出不同的模型；或不同的算法如随机森林、支持向量机、logistic 回归等）  
for  $j$  in 1, 2, 3, ...  $K$ : 将训练集中除去第 $j$ 份数据作为新训练集，用于训练模型用第 $j$ 份数据测试模型，获得该模型错误率
3. 经过第2步就得到了 1 个模型和  $K$  个错误；这  $K$  个错误率的均值和方差就是该模型的一般错误率。
4. 对每个模型重复 2 和 3 步骤，选择一般错误率最小的模型为当前最优模型。
5. 用所有的训练集数据训练当前的最优参数模型，获得最终训练结果。
6. 用独立的测试集评估模型错误率。

这一操作的优点是：

1. 所有训练集的样品都参与了每个模型的训练和评估，最大程度利用了数据。
2. 多个验证集多次评估，能更好的反应模型的预测性能。

1/12





# K-fold交叉验证

- 通常为5或10（10是最常用的经验值，但根据自己的数据集5，20也都有可能获得比较好的结果）

## Setting Hyperparameters

Your Dataset

**Idea #4: Cross-Validation:** Split data into **folds**, try each fold as validation and average the results

|        |        |        |        |        |      |
|--------|--------|--------|--------|--------|------|
| fold 1 | fold 2 | fold 3 | fold 4 | fold 5 | test |
| fold 1 | fold 2 | fold 3 | fold 4 | fold 5 | test |
| fold 1 | fold 2 | fold 3 | fold 4 | fold 5 | test |

Useful for small datasets, but not used too frequently in deep learning

易生信，毕生缘；培训版权所有。





## Caret: 一个函数统一238个机器学习R包

# Caret 是一个试图标准化机器学习过程的一个包

- Caret 对 R 中最常用的机器学习方法 (目前支持238个R包, 具体列表见 <https://topepo.github.io/caret/available-models.html>, 也可以自己根据已有方法的模板快速添加新的方法)提供了统一的接口。
- 进行数据预处理
- 实现机器学习方法流程化模型构建
- 通过参数组合和交叉验证评估模型的参数
- 选择最优模型
- 评估模型性能

易生信 生信宝典 宏基因组

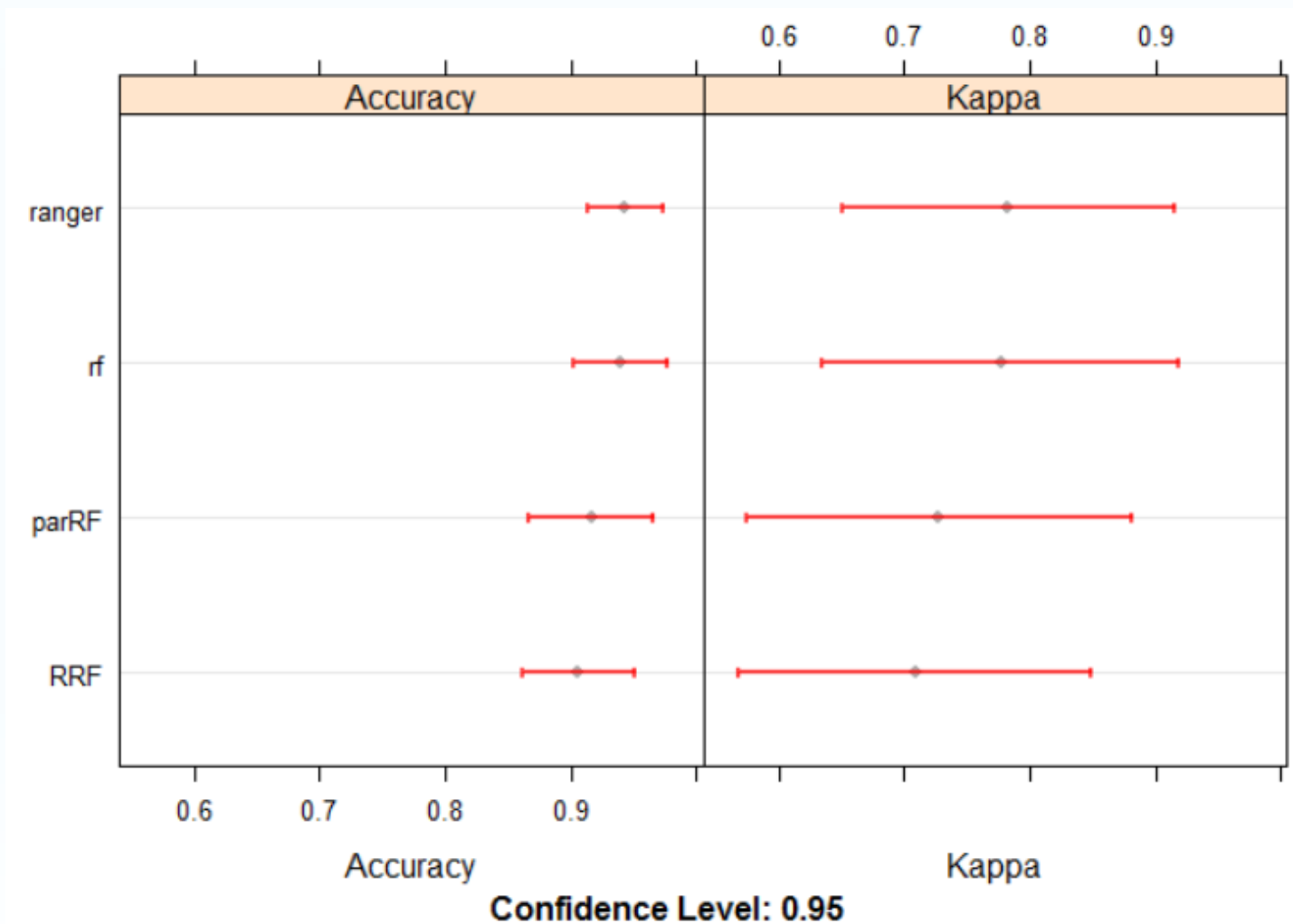


Caret依赖 `trainControl` 函数设置交叉验证参数, `train` 函数具体训练和评估模型。首先是选择一系列需要评估的参数和参数值的组合, 然后设置重采样评估方式, 循环训练模型评估结果、计算模型的平均性能, 根据设定的度量值选择最好的模型参数组合, 使用全部训练集和最优参数组合完成模型的最终训练。

```
1 Define sets of model parameter values to evaluate
2 for each parameter set do
3   for each resampling iteration do
4     Hold-out specific samples
5     [Optional] Pre-process the data
6     Fit the model on the remainder
7     Predict the hold-out samples
8   end
9   Calculate the average performance across hold-out predictions
10 end
11 Determine the optimal parameter set
12 Fit the final model to all the training data using the optimal parameter set
```

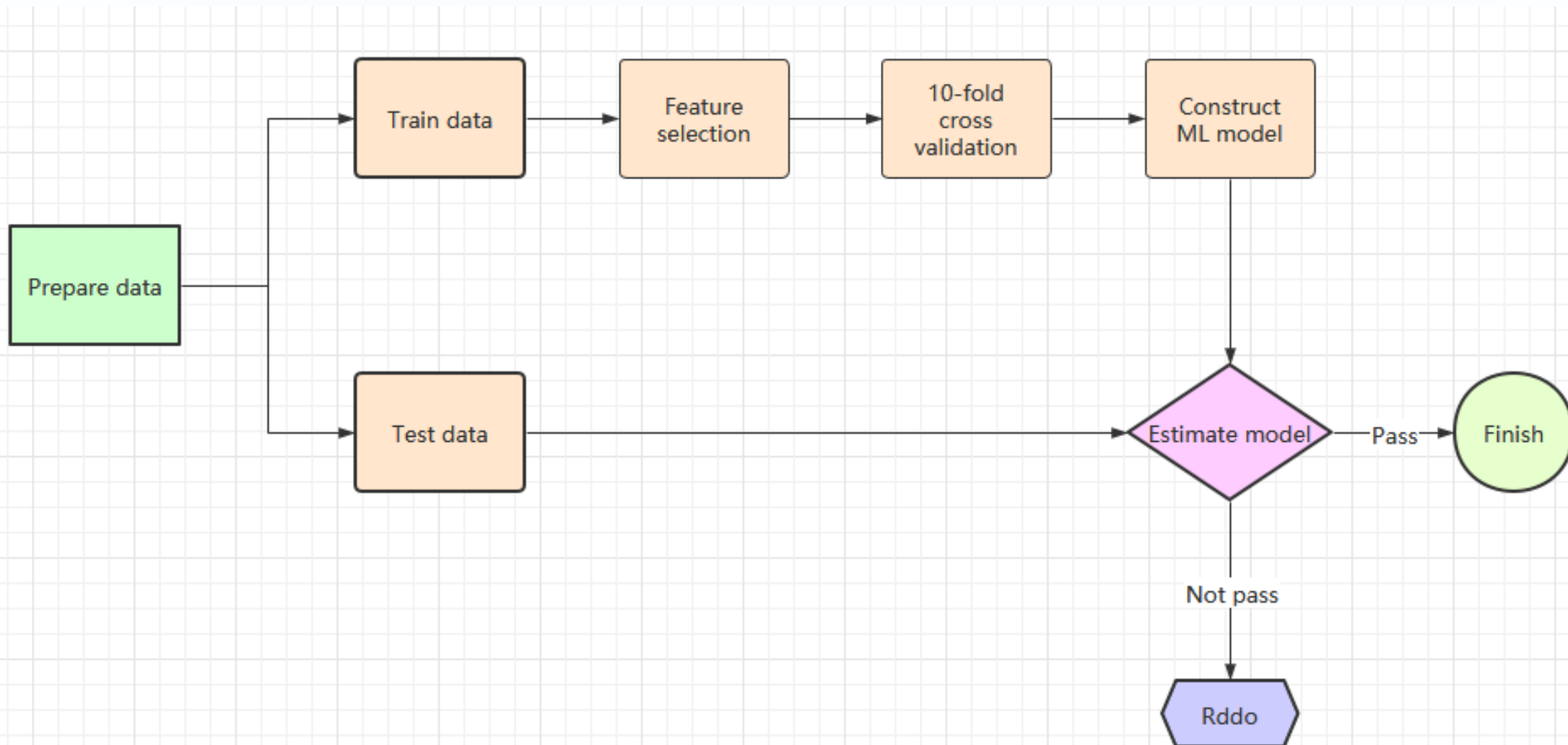


# Caret可以很方便地比较多个模型的质量



宏基因组

# 机器学习分析的基本流程





1. [机器学习算法-随机森林之决策树初探（1）](#)
2. [机器学习算法-随机森林之决策树R代码从头暴力实现（2）](#)
3. [机器学习算法-随机森林之决策树R代码从头暴力实现（3）](#)
4. [机器学习算法-随机森林之理论概述](#)
5. [随机森林拖了这么久，终于到实战了。先分享很多套用于机器学习的多种癌症表达数据集 <https://file.biolab.si/biolab/supp/bi-cancer/projections/>。](#)
6. [机器学习算法-随机森林初探（1）](#)
7. [机器学习 模型评估指标 - ROC曲线和AUC值](#)
8. [机器学习 - 训练集、验证集、测试集](#)
9. [机器学习 - 随机森林手动10折交叉验证](#)
10. [一个函数统一238个机器学习R包，这也太赞了吧](#)

易生信 生信宝典 宏基因组



1. [基于Caret和RandomForest包进行随机森林分析的一般步骤（1）](#)
2. [Caret模型训练和调参更多参数解读\(2\)](#)
3. [机器学习相关书籍分享](#)
4. [基于Caret进行随机森林随机调参的4种方式](#)
5. [送你一个在线机器学习网站，真香！](#)
6. [UCI机器学习数据集](#)
7. [机器学习第17篇 - 特征变量筛选（1）](#)

易生信 生信宝典 宏基因组





扫码关注生信宝典，学习更多生信知识



扫码关注宏基因组，获取专业学习资料

# 易生信，没有难学的生信知识

