算法导论报告

Random Forest: a C++ implementation

2023 年 6 月 9 日

摘要

介绍了 Random Forest 算法的理论，并且给出了实现细节。

关键字：随机森林，树形分类器

目录

一、 A Breif Introdution

Random Forest 是一种经典的集成学习（Ensemble Learning），鲁棒性、准确度都很好。Random Forest 是典型的计算模型（algorithm model），相较于数据模型（data model）它没有对数据的分布做出假设 [2]，同时 Random Forest 具有比较简单的数学推导，然而，Random Forest 具有不亚于数据模型的预测准确度和很高的鲁棒性。通常的 Random Forest 适用于在以类别变量为特征的数据上做分类任务，但如果做一些改进，它将可以适用于连续变量的分类任务以及回归任务 [3]。本报告中涉及的数据集是以类别变量为特征的二分类数据集，这也是 Leo Breiman 在"Random Forest" 中主要讨论的。我们将使用 Random Forest 来预测 Kaggle 上的 Titanic 问题以达到代码准确性验证的目的。

二、 Theory Part

由于有的定理的证明篇幅较长，所以此处并未给出所有定理的证明。

(一) Decision Tree

在机器学习中，决策树是一种有监督的分类算法，它接受表格类型的数据并输出数据对应的类别。决策树一般是一种树形结构，从根节点开始，它在每个节点根据特定的规则对输入数据进行划分，最终在叶结点处给出数据所属的类别 [4]。本文中涉及的决策树为二叉决策树，这意味着决策树对数据的每次划分都至多产生两类结果

如图2所示是一颗简单的二叉决策树。我们假定输入数据至少包含有无车和有无房子两个特征，在根节点处，我们首先对数据在有无房子上进行划分，如果满足规则，即有自己的房子，则到达类别“是”；如果不满足规则，即没有自己的房子，就进入下一个按规则划分，即有无工作，依次类推。关于我们应该使用何种分类规则，给出分类规则集 S 的定义，在本小节被使用的规则应该被包含于规则集 S 中。

Definition 1. 一个分类规则集 S 中包含的任意一个分类规则 s 应该满足：

1.s **仅对数据的某个特定的特征进行分类**

2. **对于连续的有序变量** x_m**，** s **应该具备形式：** $\{Is\ x_m \leq c\ ?\}\ \forall c \in (-\infty, +\infty)$

3. **对于类别变量** x_m**，若** $x_m \in B = \{b_1, b_2, \ldots, b_n\}$ **，** s **应该具备形式** $\{Is\ x_m \in B' |\ B' \subset B\ ?\}$

一般地，我们记 T 为一颗决策树包含的所有节点的集合，记 $\widetilde{T}$ 为决策树的叶子节点组成的集合。对于一个分类器，我们希望它分类的误差尽可能小，具体地，对于决策树，我们希望训练数据中，它的每个叶子节点都只有特定的某种类别的数据能够到达。这就是说，我们希望决策树的划分规则能够让数据越来越“纯粹”，以至于到达叶结点时，每个叶结点仅有一种类别能够到达。为了能够定量地比较数据之间的“不纯粹度”，我们希望找到一个杂度函数（impurity function），帮助我们将数据映射到实数域上。

如图1，在再代入误差（resubstitution error）相同时，杂度函数应该考虑到树的后续构建步骤，所以在 split1 和 split2 中，应当给予 split2 更低的杂度。这要求杂度函数是一个严格凹函数，即在定义域的在各维度上，它的二阶导应小于零（这里假设其存在二阶导）。

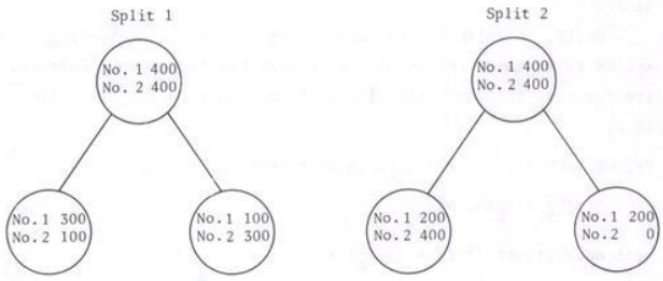

图 1: Two split example

Definition 2. 一个杂度函数 φ 是定义在 j 维向量 $P=(p_1,p_2,\ldots,p_j)$ 上的函数，其中 P 满足：$\forall j\in\{1,2,\ldots,j\},p_j\geq 0$ 且 $\sum_{k=1}^{j}p_k=1$。φ 满足以下性质：

1. **当且仅当** $P=(\frac{1}{j},\frac{1}{j},\ldots,\frac{1}{j})$ **时，φ 取到最大值**;

2. **当且仅当** $P\in\{(1,0,\ldots,0),(0,1,\ldots,0),\ldots,(0,0,\ldots,1)\}$ **时，φ 取最小值**;

3.φ **在** $p_1,p_2,\ldots,p_j$ **上对称**

4.$\varphi(p)<0\ \forall p\in[0,1]$

借助定义4，在训练过程中，对于某一个节点或某一棵决策树，我们可以给出它的杂度定义，但在定义杂度之前，需要先定义决策树中某个节点的数据数量。

Definition 3. $\forall t\in T$，定义 t 的数据数量 num_t 为训练决策树时所有到达过 t 的数据个数。特别地，num_{Root} 表示所有到达过根节点的数据个数，即训练集内的数据个数。

有时我将混用 t 来表示节点或该节点的数据数量，具体 t 的含义可以通过上下文来推断。

Definition 4. 对于给定的杂度函数 φ，定义某个节点 $t\in T$ 的杂度 $i(t)$ 为：

$$i(t)=\varphi(p(1|t),p(2|t),\ldots,p(j|t))$$

定义某棵树 T 的杂度 $I(T)$ 为：

$$I(T)=\sum_{t\in\widetilde{T}}i(t)p(t)$$

其中 $p(t)=num_t/num_{Root}$

自然地，我们用分类后节点杂度的加权与原先节点杂度作差来定义分类规则带来的杂度下降，我称它为杂度降:

Definition 5. 如果一个分类规则 s 使得具有 N_0 个数据的节点 t 分裂为 m 个节点 $t_1,t_2,\ldots,t_m$，它们分别具有 $num_1,num_2,\ldots,num_m$ 个数据，则 s 在 t 上的杂度降定义为：

$$\Delta(i(s,t))=i(t)-\sum_{k=1}^{m}P_ki_k$$

其中 $P_k=num_{t_k}/num_t$

s 在树 T 上的杂度降定义为:

$$\Delta(I(s,T)) = I(T) - I(T')$$

其中 T' 是分裂后得到的决策树

可以使用杂度降来评估某个分类规则在某个节点的分类效果。特别地, 对于二叉决策树, 有: $\Delta(i(s,t)) = i(t) - P_R i(t_R) - P_L i(t_L)$, 其中 $P_R = num_{t_R}/num_t$; $P_L = num_{t_L}/num_t$。

此时我们思考, 是否所有的分类规则都能够帮助决策树提高分类效果? 可以证明, 所有满足定义1 的分类规则 s 都能够带来正的杂度降。

Theorem 2.1. 当杂度函数 $\varphi(p_1, p_2, \ldots, p_j)$ 在维度 $p_1, p_2, \ldots, p_j$ 上严格凹时，其中 $p_k \geq 0 \forall k \in 1, 2, \ldots, j and \sum_{k=1}^{j} p_k = 1$ ，那么对于任何符合定义t的分裂规则 s，都有：

$$\Delta(i(s,t)) \leq 0$$

当且仅当 $p(i|t_L) = p(i|t_R) = p(i|t) \forall i \in \{1, 2, \ldots, j\}$ 时取到等号

定理的证明请参考论文。

现在给出一个符合直觉的贪婪做法, 在节点 t 的分类时, 都选择 $s' = \underset{s}{argmax}\, \Delta(i(s,t))$ 作为分裂规则。这种选择仅考虑单个节点 t 时是最优的, 下面的定理保证了它也是在全局下对 t 的最好分裂。

Theorem 2.2. 对于决策树 T，如果存在一个节点 $t \in \widetilde{T}$ 可以被分裂，那么存在节点 t 的单步最优分裂 s' 同时满足：$s' = \underset{s}{argmax}\, \Delta(i(s,t))$ 和 $s' = \underset{s}{argmax}\, \Delta(I(s,T))$

Proof. 对于树 T, 其杂度为:

$$I(T) = \sum_{t \in \widetilde{T}} p(t) i(t)$$

$\forall t_0 \in \widetilde{T}$, 节点分裂后的树 T' 的杂度为:

$$I(T') = \sum_{t \in T - t_0} I(t) + p(t_L) i(t_L) + p(t_R) i(t_R)$$

所以有:

$$\begin{aligned} \Delta(I(T)) &= p(t_0) i(t_0) - p_{t_L} i(t_L) - p(t_R) i(t_R) \\ &= p(t_0) \left[i(t_0) - \frac{i(t_L p(t_L)}{p(t_0)} - \frac{i(t_R p(t_R)}{t_0} \right] \\ &= p(t_0) \Delta(i(s,t)) \end{aligned} \tag{1}$$

由于对于确定的树 T, $P(t_0)$ 是定值, 所以 $\Delta(I(T))$ 和 $\Delta(i(s,t))$ 同时取到最大值 □

观察分类集合 S, 可以发现对于类别变量, 平凡算法需要遍历它取值集合 B 的所有子集才能确定最优的集合 B'。因为子集个数随集合的势呈现指数级增长, 所以当 B 的势较大时, 平凡算法显然是低效的。针对二分类问题, Breiman 提出了优化方法, 结合快速排序能够达到 $O(nlogn)$ 的时间复杂度。这个优化方法基于下面的定理:

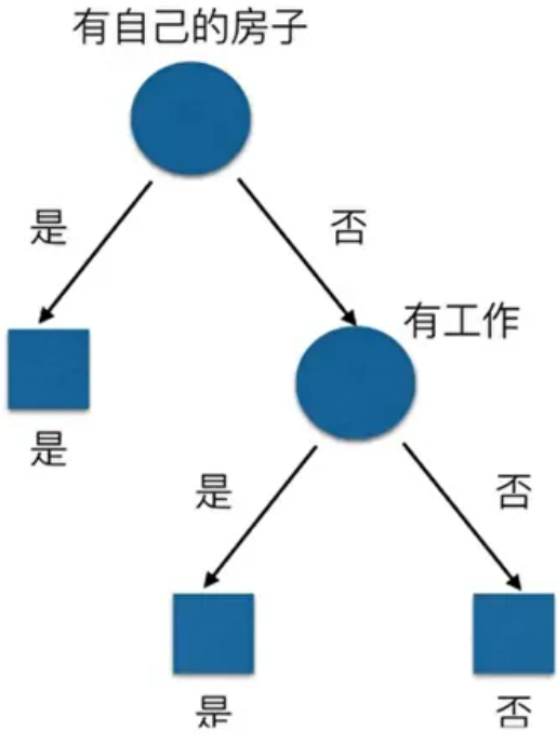

图 2: 决策树

Theorem 2.3. 在节点 t 上，对于变量 x，假设其所有可能的取值组成的集合为 $B = \{b_1, b_2, \ldots, b_L\}$，如果标签 y 取值为 0 或 1，有：

对 $p(1|x = b_l)$ 排序，得到：

$$p(1|x = b_{l_1}) \leq p(1|x = b_{l_2}) \leq \cdots \leq p(1|x = b_{l_L})$$

对于 $\forall \varphi \in F$，存在 B 的一个子集

$$\{p(1|x = b_{l_1}) \leq p(1|x = b_{l_2}) \leq \cdots \leq p(1|x = b_{l_k})\}$$

可以使分类规则 s 取到$:s = \underset{s}{argmax}\, \Delta(i(s,t)$

定理的证明请参考论文。

至此，我们已经成功构建了用于决策树的理论体系。

(二) Random Forest

随机森林（Random Forest）是一种集成分类器，它集成了一定数量的随机决策树。随机决策树是在决策树的基础上引入了一些随机向量产生的。[1]

Definition 6. 随机森林 $H(x, \theta)$ 是用投票（$voting$）的方式集成了 K 个树形分类器 $h(x, \theta_k)$ 的分类器，其中 x 是给定的数据，$\{\theta_k\}$ 由随机向量 θ 产生且它们两两独立同分布。

对于给定的数据 x，下文也会使用 $h(x, \theta_k)$ 表示一个树形分类器分类的结果，一般 $h(x, \theta_k)$ 的含义可以通过上下文确定。

在讨论随机森林之前，需要先建立一些评价指标。

Definition 7. 对于数据对（*x, y*），定义随机森林的置信函数（*margin function*）为*:*

$$mg(x,\ y) = \underset{k}{av}\, I(h(x,\theta_k) = k) - \underset{j\neq y}{max}\, \underset{k}{av}\, I(h(x,\theta_k) = j)$$

进一步地，可以给出随机森林的泛化误差（generalization error）的定义：

Definition 8. 假设数据 x，y 符合某个分布，定义随机森林的泛化误差为*:*

$$PE^* = \underset{x,y}{P}\,(mg(x,y) < 0)$$

其中 P 的下标表示这是在 x，y 空间上的概率。

现在我们可以给出，对于一个随机森林，在包含的决策树个数 K 足够多时，其泛化误差将几乎必然收敛（almost sure convergence）到与 θ 有关的确值。

Theorem 2.4. 随着随机森林中树形分类器数量增长，对所有序列 $\theta_1, \theta_2, \ldots$，$PE^*$ 几乎必然收敛到*:*

$$\underset{x,y}{P}(\underset{\theta}{P}(h(x,\theta) = y) - \underset{j\neq y}{max}\, \underset{\theta}{P}(h(x,\theta) = j))$$

Proof.

Lemma 2.5. 随着森林中树形决策器数量上升，在序列空间 $\theta_1, \theta 2, \ldots$ 上存在一个概率为零的集合 C，使得对于 C 之外的任何序列，对任意 x 均有*:*

$$\frac{1}{N}\sum_{n=1}^{N} I(h(x,\theta_n) = j) \to P_\theta(h(x,\theta) = j)$$

Proof. 对于确定的训练集和特定的随机向量 θ，集合 $\{x : h(x,\theta) = j\}$ 是超长方体的并集。

由于随机森林中，序列 $\theta_1, \theta 2, \ldots, \theta_N$ 是有限的，所以集合 $\{x : h(x,\theta) = j\}$ 也是有限的，记它们为: $S_1, S_2, \ldots, S_K$

令 $\varphi(\theta) = k\ iff\ \{x : h(x,\theta) = k\}$

令 $N_k = \sum\limits_{n=1}^{N} I(\varphi(\theta_n) = k)$ 于是我们有：

$$\frac{1}{N}\sum_{n=1}^{N} I(h(x,\theta) = j) = \frac{1}{N}\sum_{k=1}^{K} N_k I(x \in S_k)$$

其中 $N_k = \sum\limits_{n=1}^{N} I(\varphi(\theta_n) = S_k)$

因为对于不同的 n，$I(\varphi(\theta_n) = S_k)$ 独同分布，所以，依据大数定理，有：

$$\begin{aligned}\frac{1}{N}\sum_{k=1}^{K} N_k I(\varphi(\theta) = k) &= \sum_{k=1}^{K}\frac{1}{N}\sum_{n=1}^{N} I(\varphi(\theta_n) = k) I(x \in S_k) \\ &= P_\theta(h(x,\theta) = j)\end{aligned} \tag{2}$$

□

根据引理，$mg(x,y)$ 几乎必然收敛到：

$$P_\theta(h(x,\theta) = y) - \underset{j\neq y}{max}\, P_\theta(h(x,\theta) = j)$$

所以，PE* 几乎必然收敛到：

$$P(P_\theta(h(x,\theta)=y) - \max_{j\neq y} P_\theta(h(x,\theta)=j) < 0)$$

□

这个定理确保了随机森林的泛化误差的收敛性，即对于足够大的树形分类器数目 K，我们几乎必然能够得到稳定的泛化误差。接下来我们考察随机森林分类器的效果，有以下定理：

Theorem 2.6. 泛化误差的一个上界是： $PE^* \leq \hat{\rho}(1-s^2)/s^2$

其中

$\hat{\rho} = \mathbb{E}_{\theta,\theta'}\{\rho(\theta,\theta')sd(\theta)sd(\theta')\}/\mathbb{E}_{\theta,\theta'}\{sd(\theta)sd(\theta')\}$

$s = \mathbb{E}\{P_\theta(h(x,\theta)=y) - \max_{j\neq y} P_\theta(h(x,\theta)=j)\}$

证明比较冗长，限于篇幅此处不再给出，请参考"Random Forest", Leo Breiman

$\hat{\rho}$ 可以被近似地看作随机森林中两个树形分类器的相关性（分类结果的一致性），以皮尔逊相关系数给出，相关性越强，越接近 1，越不相关，越接近 0。s 可以被理解为森立中单个树形分类器的预期分类效果（具体为单棵树泛化误差的期望），效果越好，越接近 1，效果越差，越接近 0。

上界的一个常用近似是 c/s^2 $ratio$。定义如下

Definition 9. 随机森林的 $c/s^2 ratio$ **定义为**: $c/s^2 \triangleq \hat{\rho}/s^2$

由此可以知道，随机森林的分类效果主要受其中树形分类器的相关度和树形分类器的分类效果影响。如果想要提升随机森林的分类效果，应该降低树形分类器的相关度，同时提升其分类效果。

(三) Bonus

个人有以下简单思考：

1. 参考证明过程可以发现,分类效果一个更加直接(也更好)的上界是:$PE^* \leq Var(mg(x,y)/s^2$。事实上，论文在 Intro 部分提到构建随机森林的"Another approch is to select the training set from a random set of weights on the examples in the training set"。这种做法的 idea 或许就来自这个上界。但是它可能具有一些缺陷，比如选取 random set 的大小难以确定。也有结合抽取特征向量和抽取训练集的方式，被称为 random subspace。

2. 随机森林或是 Ensemble Learning 能够在理论上解释“大选”或“普选”制度的合理性，同时也可能是对精英决策制度的一个有力的驳斥。民众之间在学历、经验等方面的差异正是降低误差上界的重要因素，相反的，如果精英层出现了同质化，将会加剧决策效果的不稳定性，使得决策变为更加冒险。关于民众决策能力，Leo Breiman 在论文中讨论了 weak input 对随机森林造成的影响，在 weak input 场景下，虽然单个决策器决策能力很弱，但如果数量足够多，仍能获得较好的效果。进一步地，随机森林对于错误输入具有高鲁棒性，相应的，具有足够多民众参与的决策体系可能很难被错误信息引导。在实践中的具体讨论超出了此报告的范围。

3. 我们不能确保使用随机森林得到的结果一定比按照 CART 构建的决策树效果更好，至少没有定理能够确保我们做到这一点，我曾经利用 CART 的方法构建过一颗没有经过剪

图 3: 效果比较

枝的决策树，如图3，在 Titanic 数据集上它的效果与利用 scikit learn 库构建的随机森林效果差距并不大（仅有 0.3%）。随机森林带来的启发主要是它对弱分类器的效果提升和它的鲁棒性，它们在 Ensemble Learning 追求的目标之中。

三、 Implementaion Details

囿于个人能力,我没有复现论文中的所有内容,未复现的内容包括随机组合森林(RandomForest-CI）和袋外估计（Out-of-Bag Estimate）以及论文中基于许多数据集的各种实验，相应地，我也没有在理论部分提到它们。

(一) DataLoader

由于训练集和测试集都储存在 csv 文件中，所以需要一个 DataLoader 将数据加载到内存中。csv 的格式以换行符分割不同的记录，在记录内部以逗号分割不同的字段。所以在读入时，首先以行为单位读入，然后再以逗号来分割字段并存储在 vector<vector<string» 中。需要注意的要点有：

1.csv 中可能会有文本类型的字段，文本字段以英文双引号包裹，中间可能含有逗号，而英文双引号又是不区分左引号和右引号的，所以在代码中要特别注意配对的过程。

2.csv 中字段值可能缺失，当采用 getline 函数来做字段分割时，如果某条记录最后一个或数个字段缺失，那么它们将会被忽略，这也需要特别留意。

读取时，从输入流读入默认是字符串类类型，考虑到后续可能会需要对表中某几列做类型变换，这时创建 Field 类来存储字段值，然后将文件存储在 vector<vector<Field» 中可以让我们很方便地字段值进行类型转换。Algorithm 1给出 DataLoader 读取 csv 的核心代码

Algorithm 1 Read data from csv file

Require: 数据文件 *csv_data*，空字符串数组 WordRes
Output 记录文件字符串的序列 WordRes

```
count = 0
while 从 csv_data 中获取一行 line do
    解析 line 为序列 words
    for i = 0 to size of words do
        if words[i] 为空字符串 then
            处理空字符串并加入 WordsRes
            continue
        else if words[i] 的第一个字符为双引号 then
            i ← i + 1
            str ← words[i]
            while words[i] 不以双引号结尾 do
                str ← str + words[i]
                i ← i + 1
            end while
            str ← str + words[i]
            将 str 加入 WordRes
            continue
        else
            处理一般字符串并加入 WordsRes
        end if
    end for
    if count = 0 then
        设置 header 的大小
        设置 file 的大小
    else
        补充空字符串并将字段添加到 file
    end if
    count ← count + 1
end while
关闭 csv_data
```

(二) Tree-structured Classifier

RandomTree 以树的深度作为停止条件，按照 CART 内描述的方法递归地创建树形分类器的节点，主要步骤就是利用随机抽取和 criteria 选择特征然后分裂节点。具体地，每次分裂节点前，需要从给定的所有特征中不放回地抽取指定个数，然后根据评价函数集 F 中的函数来选择最优的作为当前节点的选定特征。在不放回地抽取特征时，可以利用 Knuth Durstenfeld shuffle 算法来避免繁琐的元素插入/删除操作。Algorithm2 是 Knuth 算法的一个伪代码。

Algorithm 2 Knuth Durstenfeld Shuffle Algorithm

Require: 一个数组或列表 $array$，需要抽取的元素个数 m

```
n = 数组或列表的长度
for i ← n − 1 to n − m do
    生成一个随机数 j，范围为 [0, i]
    交换 array[i] 和 array[j]
end for
```

依 Theory Part 中所述, 代码中选择了方差指标（二次函数）作为 criteria。如 Algorithm 3所示是节点分裂的伪代码

Algorithm 3 Node Splitting Algorithm

Input 数据记录 X, 数据标签 Y, 树深度 $depth = 0$

Output 指向树根节点的指针

```
procedure SPLIT(x, y, depth) {}
    if depthmaxDep then
        创建叶结点 leafnode
        return 指向 leafnode 的指针
    end if
    if size of y = 0 then
        return NULL
    end if
    运行 Knuth 洗牌算法，得到选择的特征序列 selected
    计算 selected 中各特征的 variance criteria value，存储在序列 CriRes 中
    FeaSel ← min(CriRes)
    创建节点 currNode
    向 currNode 中保存必要的信息
    depth ← depth + 1
    根据 FeaSel 对 x, y 分组得 x0, x1 和 y0, y1
    split(x0, y0, depth)
    split(x1, y1, depth)
    return 指向 currNode 的指针
end procedure
```

树形分类器用于决策时，需要循环访问其节点，并且在没有子节点/遇到叶结点时返回分类结果，Algorithm 4给出了对单个数据点分类的伪代码

(三) RandomForest

RandomForest 类中包含有 n 个 RandomTree 类，这些 RandomTree 类被放置在一个队列中，在训练/预测时，通过 n 次出队、入队来实现对它们的遍历。投票时决定最终的输出时，将按照“少数服从多数的原则”，选择多数分类器的结果作为最终的结果。由于这部分实现比较简单，所以不给出伪代码。

Algorithm 4 Tree-structed Classifier for Prediction

REQUIRE data x

$state \leftarrow root\ of\ tree$
while state is not a leaf node **do**
  if state belongs to right child **then**
    $state \leftarrow state's\ right\ child$
  else
    $state \leftarrow state's\ left\ child$
  end if
end while
output class of state

四、 Complexity Results

结合伪代码和源码可知，一般情况下（以树的深度为终止条件时），在训练阶段，RandomForest 的时间复杂度是：

$$\mathcal{O}\left(Features \times Columns \times Dataset \times Depth\right)$$

对于预测，RandomForest 的时间复杂度是：

$$\mathcal{O}\left(numberoftrees \times datasetsize \times numberofattributevalues\right)$$

五、 Code Availability

代码在 GitHub 项目Faraway: RandomForest上开源。也可以参考电子版作业中附上的代码。数据集可以从 Kaggle 上下载，Github 项目和电子版作业也附上了数据集。

运行时，请注意超参数的设置，随机森林规模上，建议不要设置过大的超参数以免运行时间太长，可以参考默认参数。建议选择删除数据集中缺失值过多及对分类没有直接意义的列（第 3，5，10 列），如果希望调整删除的列，请注意这可能影响超参数"numfea" 的设置。

如果希望替换数据集，请注意文件格式应为 csv，同时请注意目前只支持二分类的数据集，数据集的表头应当存在且不含空值。

由于没有调整超参，同时没有加入 bagging 过程，树的相关性比较大。所以运行的准确率不太好。在 Kaggle 提交可以发现准确率较单个决策器有提升但是提升不多，后续可能继续在 GitHub 更新论文的其他部分复现/探究准确率不高的原因。

六、 Acknowledgement

特别感谢老师本学期的授课、指导。

参考文献

[1] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[2] Leo Breiman. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical science*, 16(3):199–231, 2001.

[3] Leo Breiman. *Classification and regression trees*. Routledge, 2017.

[4] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1:81–106, 1986.