决策树(Decision Tree)

基本概念

决策树是以树状图为基础的、基于特征的、有监督的、贪心的、学习算法。决策树可以是二叉树也可以是非二叉树，其输出结果是一些进行判别的规则。
决策树由节点和有向边组成，内部的节点表示一个特征（属性），叶子节点表示一个分类。决策树可以用于分类问题也可以用于回归问题。对于分类问题，利用决策树进行预测时，将样本实例输入决策树，经过决策树内部的判别规则，最终会将样本实例分配到某一个叶节点的类中，该叶节点的类就是样本实例所属的类别。
例如，刘某需要贷款买房，银行需要评估其贷款风险，评估项有：Credit、Term、Income三项。根据用户的数据（样本）构造出决策树，再将刘某的信息作为决策树的输入，经过判定得出风险值。解决该问题的整体框架为：

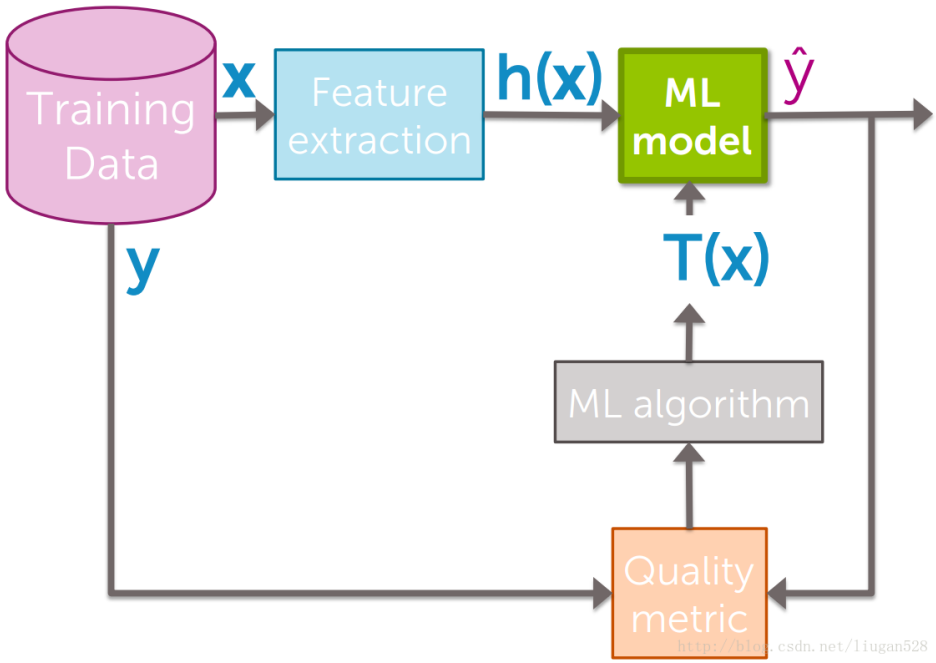

图1

如果ML model采用决策树算法，则可构造出类似图2的决策树：

Credit	Term	Income	y
excellent	3 yrs	high	safe
fair	5 yrs	low	risky
fair	3 yrs	high	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	risky
fair	5 yrs	low	safe
poor	3 yrs	high	risky
poor	5 yrs	low	safe
fair	3 yrs	high	safe

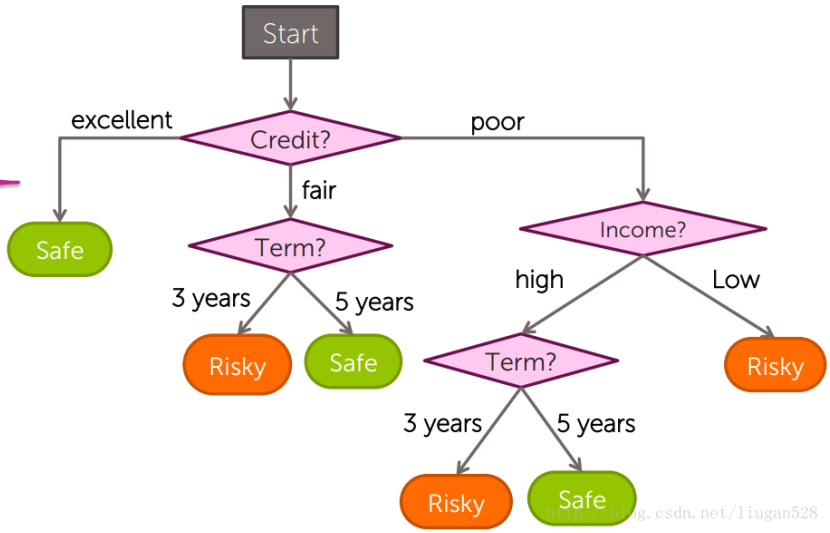

图2

决策树构造好之后对于具体实例预测方法为将实例作为输入使之贯穿整个决策树得出最终的判定结果，例如，对于刘某，假设其Credit=poor，Income=high，Term=5 years，则风险预测方法为：

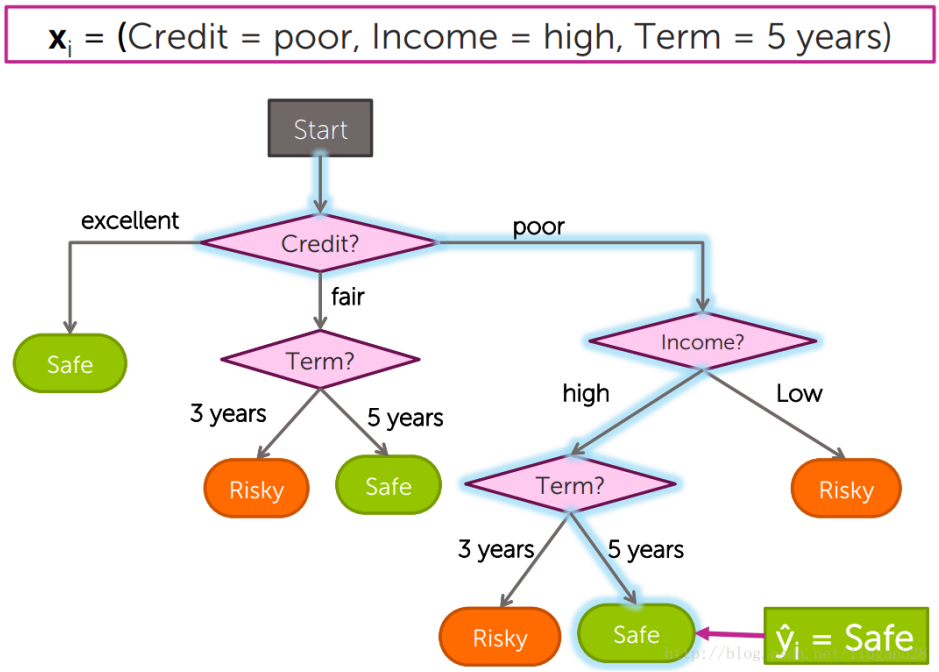

图3

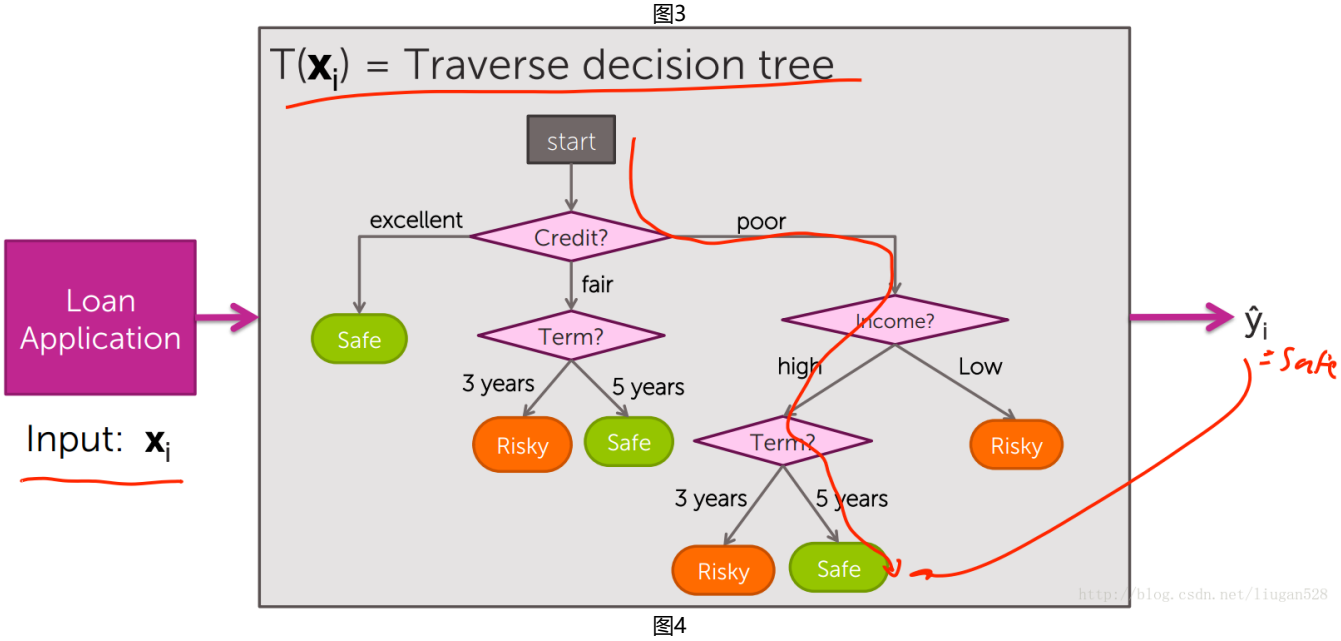

图4

构造决策树的算法

决策树的生成是一个递归过程，在决策树基本算法中，有三种情形会导致递归返回：
1. 当前结点包含的样本全属于同一类别，无需划分；
2. 当前属性集为空，或是所有样本在所有属性上取值相同，无法划分；
3. 当前结点包含的样本集合为空，不能划分。

ID3

根据信息论的知识，系统的信息增益越大那么纯度就越高。ID3算法的核心就是根据信息增益作为选择特征（属性）的标准。每次都选择可以使系统信息增益最大的特征（属性）。
给定训练集$D = (\vec{x_1}, y_1), (\vec{x_2}, y_2), \ldots, (\vec{x_N}, y_N)$，其中$\vec{x_i} = (x_{i1}, x_{i2}, \ldots, x_{in})$，n为特征个数，$y_i \in 1, 2, \ldots, K$为类别标记，$i = 1, 2, \ldots, N$，N为样本数量。假设每个类别有$C_k$个样本。对于数据集D，可以用熵Entropy(D)来描述数据集的不确定程度，熵越大表示越混乱，熵越小表示越有序，因此信息增益表示混乱的减少程度。当数据集中的所有样本都属于同一类别时，$Entropy(D) = 0$，当数据集中的各个类别的样本分别均匀时Entropy(D)最大。给定特征F，信息增益定义为：

$$Gain(D, F) = Entropy(D) - Entropy(D, F) \tag{1}$$

其中，$Gain(D, F)$表示信息增益，$Entropy(D)$表示利用特征F对数据集进行划分之前系统的熵，$Entropy(D, F)$表示利用特征F对数据集进行划分的条件熵。

$$Entropy(D) = -\sum_{i=1}^{K} \frac{C_k}{N} \log \frac{C_k}{N} \tag{2}$$

$$Entropy(D,F)=\sum_{i=1}^{n}\frac{N_i}{N}\sum_{k=1}^{K}-(\frac{N_{ik}}{N_i}\log\frac{N_{ik}}{N_i}) \tag{3}$$

举例说明公式含义：
根据图2的数据，一共9个样本，包括5个safe，4个risky，则：

$$\begin{aligned}Entropy(D)&=-\frac{5}{9}*\log_2\frac{5}{9}-\frac{4}{9}*\log_2\frac{4}{9}\\&=0.991076059838222\end{aligned}$$

如果根据特征Income来划分：

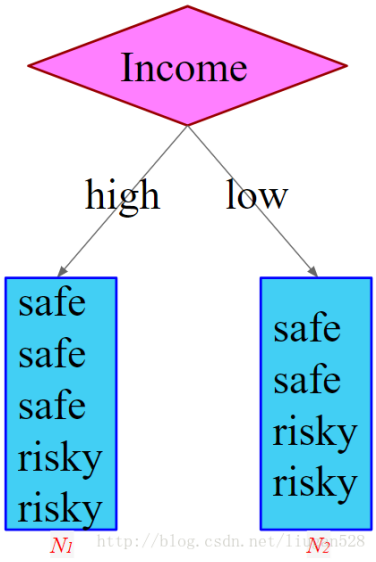

图5

划分后，数据D被分为两部分，high分支、low分支的熵分别为：

$$\begin{aligned}Entropy(D,high)&=-\frac{3}{5}*\log_2\frac{3}{5}-\frac{2}{5}*\log_2\frac{2}{5}\\&=0.970950594454669\\Entropy(D,low)&=-\frac{2}{4}*\log_2\frac{2}{4}-\frac{2}{4}*\log_2\frac{2}{4}\\&=1\end{aligned}$$

那么根据Income划分之后的条件熵为：

$$\begin{aligned}Entropy(D,Income)&=\frac{5}{9}*0.970950594454669+\frac{4}{9}*1\\&=0.983861441363705\end{aligned}$$

那么根据特征Income划分的信息增益为

$$\begin{aligned}Gain(Income)&=Entropy(D)-Entropy(D,Income)\\&=0.991076059838222-0.983861441363705\\&=0.007214618474517\end{aligned}$$

根据Term进行划分：

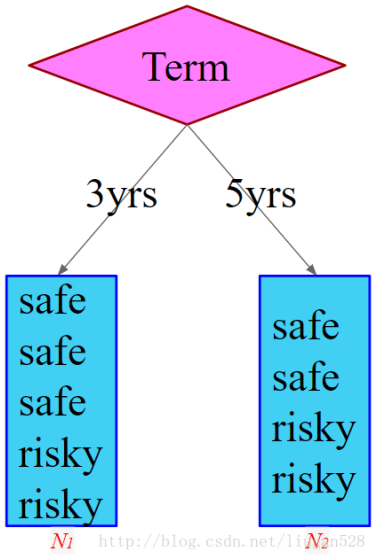

图6

比较图5和图6可知根据Term进行划分和根据Income进行划分的信息增益是相同的，因此：

$$Gain(Term) = 0.007214618474517$$

根据Credit进行划分：

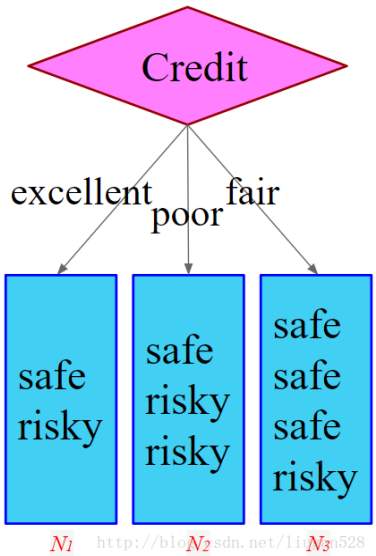

图7

$$\begin{aligned}
Entropy(D, excellent) &= -\frac{1}{2} * \log_2 \frac{1}{2} - \frac{1}{2} * \log_2 \frac{1}{2} \\
&= 1 \\
Entropy(D, poor) &= -\frac{1}{3} * \log_2 \frac{1}{3} - \frac{2}{3} * \log_2 \frac{2}{3} \\
&= 0.918295834054489 \\
Entropy(D, fair) &= -\frac{3}{4} * \log_2 \frac{3}{4} - \frac{1}{4} * \log_2 \frac{1}{4} \\
&= 0.811278124459133 \\
Entropy(D, Credit) &= 2/9 * 1 + 3/9 * 0.918295834054489 + 4/9 * 0.811278124459133 \\
&= 0.888888888888889 \\
Gain(Credit) &= Entropy(D) - Entropy(D, Credit) \\
&= 0.991076059838222 - 0.888888888888889 \\
&= 0.102187170949333
\end{aligned}$$

比较Gain(Income)、Gain(Term)、Gain(Credit)可知按照Credit进行划分的信息增益最大，即Credit在第一步使信息熵下降得最快，所以决策树的根节点就取Credit。
接下来，需要根据特征Term和Credit来对N_1、N_2、N_3进行划分，方法如上。对N_1、N_2、N_3分别进行一次划分就没有特可用了，算法终止。
（对于本例为展示方便只选择了三个特征的例子，对于其它实际问题往往有更多的特征，就需要不断的往下划分。）

ID3缺点：
1. 以信息增益对训练集的特征进行划分，会产生偏向于选择取值较多的特征的问题。
2. ID3只有树的生成算法，没有剪枝，生成的树容易产生过拟合，即对训练集匹配的很好但是对于测试集效果较差。

例如，对于图8，当选择Day作为特征进行划分的时候可以使信息增益最大，（此时条件熵为0，信息增益 $Gain(D,F) = Entropy(D) - 0 = Gain(D,F)$ 。也就是说在极限情况下特征Day将样本一一对应到一个叶节点中去，这显然不是最佳的选择。

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

图8

C4.5

C4.5使用信息增益率作为选择特征的标准。给定特征F，信息增益率定义为：

$$GainRatio(D,F) = \frac{Gain(D,F)}{SplitInformation(D,F)} \tag{4}$$

其中，$GainRatio(D,F)$ 是信息增益率，$SplitInformation(D,F)$ 是分离信息（Split Information）。
例如，对于图8，根据特征Day计算信息增益率：

$$\begin{aligned} SplitInformation(D,Day) &= -\frac{1}{14} * \log_2 \frac{1}{14} * 14 \\ &= 3.807354922057603 \\ Gain(D,Day) &= -\frac{5}{14} * \log_2 \frac{5}{14} - \frac{9}{14} * \log_2 \frac{9}{14} \\ &= 1.485426827170242 \\ GainRatio(D,Day) &= \frac{1.485426827170242}{3.807354922057603} \\ &= 0.390146665488038 \end{aligned}$$

而以Outlook作为特征进行划分的信息增益率：

$$\begin{aligned} SplitInformation(Outlook) &= -5/14 * \log_2 \frac{5}{14} - \frac{4}{14} * \log_2 \frac{4}{14} - \frac{5}{14} * \log_2 \frac{5}{14} \\ &= 1.577406282852345 \\ GainRatio(D,Outlook) &= \frac{1.485426827170242}{1.577406282852345} \\ &= 0.941689432404326 \end{aligned}$$

显然$GainRatio(D,Outlook)$要比$GainRatio(D,Day)$ 大，因此就不会选择信息增益最大的特征Day。
需要注意的是，$SplitInformation(D,F)$ 描述的是特征对训练集的分辨能力，$SplitInformation(D,F)$ 越大说明其对应的特征种类越多。并不表征其对类别的分辨能力。
利用C4.5构造决策树的过程和利用ID3是一样的，只需要将选择特征的标准由信息增益换成信息增益率即可。另外，C4.5可以处理连续数据，例如：

Day	Outlook	Temperature	Humidity	Windy	Play Golf?
1	Sunny	85	85	False	No
2	Sunny	80	90	True	No
3	Overcast	83	78	False	Yes
4	Rainy	70	96	False	Yes
5	Rainy	68	80	False	Yes
6	Rainy	65	70	True	No
7	Overcast	64	65	True	Yes
8	Sunny	72	95	False	No
9	Sunny	69	70	False	Yes
10	Rainy	75	80	False	Yes
11	Sunny	75	70	True	Yes
12	Overcast	72	90	True	Yes
13	Overcast	81	75	False	Yes
14	Rainy	71	80	True	No

图9

对于图9训练集，Temperature特征和Humidity特征均属于连续特征。
C4.5处理连续属性的方法是先把连续属性转换为离散属性再进行处理。虽然本质上属性的取值是连续的，但对于有限的采样数据它是离散的，如果有N个样本，那么我们有N − 1种离散化的方法，给定分界点Value，小于等于Value的分到左子树，大于Value的分到右子树。计算这N − 1种情况下最大的信息增益率。
在离散属性上只需要计算1次信息增益率，而在连续属性上却需要遍历计算N − 1次以确定最优的分割点，计算量是相当大的。有办法可以减少计算量，对于连续属性先进行排序，只有在决策属性发生改变的地方才需要切开。比如对Temperature进行排序：

Outlook	Temperature	Humidity	Windy	PlayGolf?
overcast	64	65	True	yes
rainy	65	70	True	no
rainy	68	80	False	yes
sunny	69	70	False	yes
rainy	70	96	False	yes
rainy	71	91	True	no
sunny	72	95	False	no
overcast	72	90	True	yes
rainy	75	80	False	yes
sunny	75	70	True	yes
sunny	80	90	True	no
overcast	81	75	False	yes
overcast	83	86	False	yes
sunny	85	85	False	no

图10

本来需要计算13次来确定分界点，现在只需计算7次。一般使用增益来选择连续值特征的分界点，因为如果利用增益率来选择连续值特征的分界点，会有一些副作用。分界点将样本分成两个部分，这两个部分的样本个数之比也会影响增益率。根据增益率公式可以发现，当分界点能够把样本分成数量相等的两个子集时（此时的分界点为等分分界点），增益率的抑制会被最大化，因此等分分界点被过分抑制了。子集样本个数能够影响分界点，显然不合理。因此在确定有连续值的特征的分界点时还是采用增益，而在分界点确定之后选择特征的时候才使用增益率。这个改进能够很好得抑制连续值属性的倾向。
对有连续值的特征构造出的决策树形如：

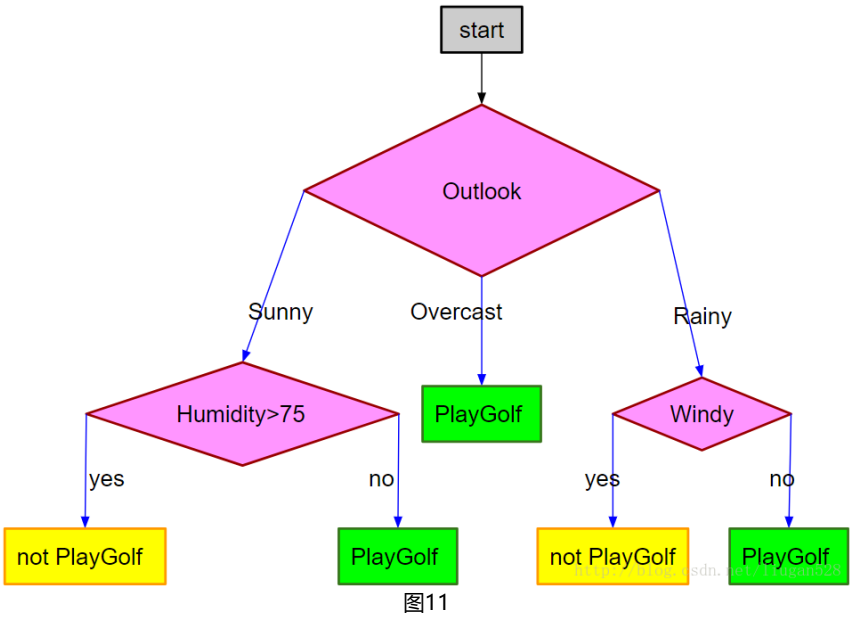

图11

决策边界

决策树的决策边界是一些垂直于特征的线，例如对于一维特征，决策边界类似：

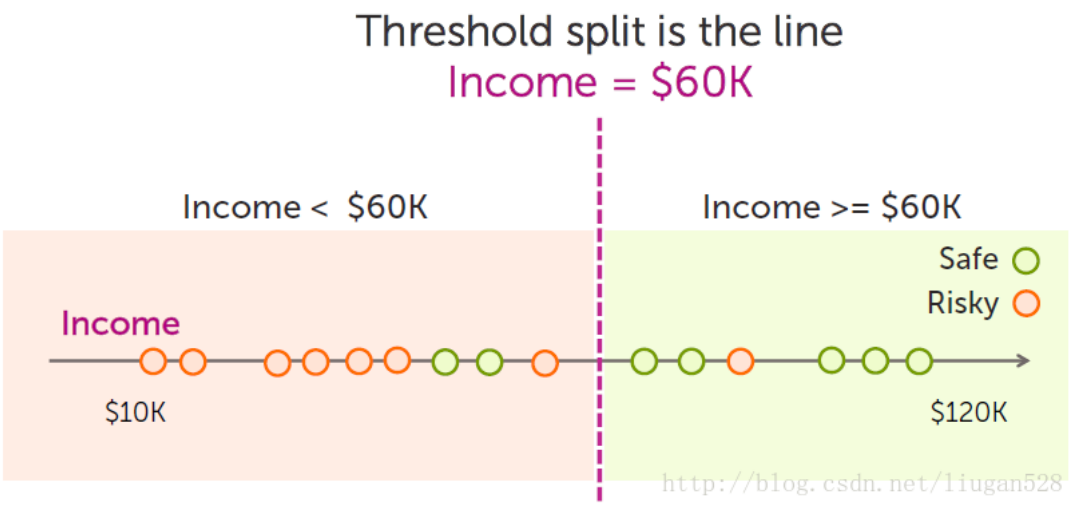

图12

对于二维特征，决策边界类似：

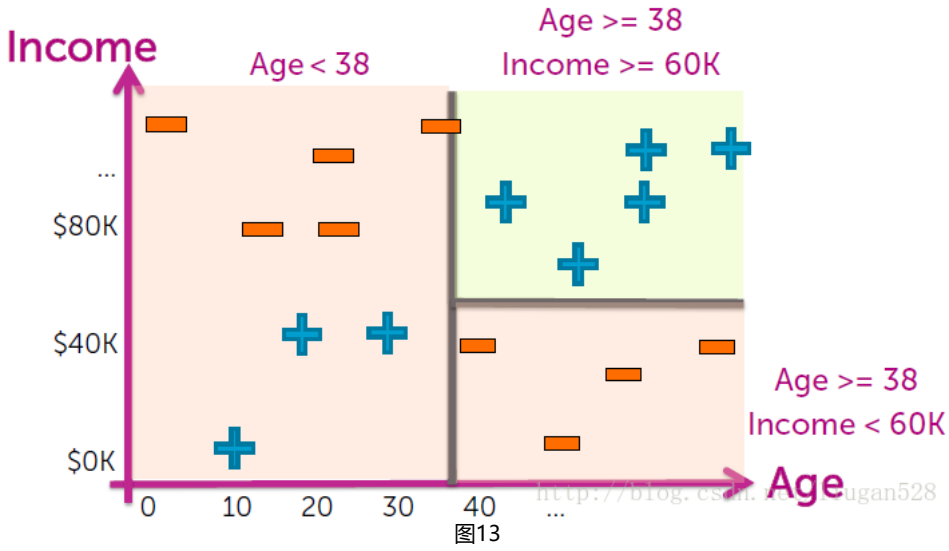

图13

C4.5优点：
a. 使用信息增益率作为划分的标准，克服了ID3使用信息增益带来的选择特征时偏向于选择取值多的特征的问题；
b. 可以处理连续的特征；
c. 在构造树的同时进行剪枝；
d. 可以处理不完整的数据
对于如何进行剪枝、如何处理不完整的数据，后续会有专题文章。
C4.5缺点：1.在构造决策树的过程中，需要对数据集进行多次顺序扫描和排序，因此算法比较低效；2.C4.5只适合处理训练集小的样本，如果训练集样本过大，内存无法容纳所有的数据集是无法完成决策树的构造的。

CART(Classification and Regression Tree)

分类与回归树（CART）算法也可以用来构造决策树，并且CART构造出的决策树是二叉树。CART算法既可以用于分类又可以用于回归。
分类与回归树模型采用不同的标准来选择最优的特征，CART分类树采用基尼指数，CART回归树采用最小加权平均方差。

CART分类树

对于给定样本集合D，有K 个类别，每个类别样本个数为C_k，$k = (0, 1, \dots, K)$ ，则基尼指数可定义为：

$$\text{Gini}(D) = 1 - \sum_{k=1}^{K} (\frac{C_k}{D})^2 \tag{5}$$

如果根据特征F 来进行划分，则根据F 的取值可将D 划分为两个子集D_1、D_2，则在特征F 的条件下，集合D的基尼指数为：

$$\text{Gini}(D, F) = \frac{D_1}{D}\text{Gini}(D_1) + \frac{D_2}{D}\text{Gini}(D_2) \tag{6}$$

其中，D，D_1，D_2在公式(5)中用作集合中样本数目。

为展示方便还拿图2的数据说明利用基尼指数构造CART分类树的过程。

- **以Credit为特征进行划分**：
  CART算法构造的决策树是二叉树，而特征Credit有三个取值，因此需要将Credit的三个取值中的两个进行合并，因此需要进行遍历合并求基尼指数来确定哪两个值合并是最好的。

1. excellent与 fair 合并:

$$\begin{aligned}\text{Gini}(D, \text{excellent} + \text{fair}) &= 1 - (\frac{4}{6})^2 - (\frac{2}{6})^2 \\ &= 0.4444 \\ \text{Gini}(D, \text{poor}) &= 1 - (\frac{1}{3})^2 - (\frac{2}{3})^2 \\ &= 0.4444 \\ \text{Gini}(D, \text{Credit}) &= \frac{6}{9} * 0.4444 + \frac{3}{9} * 0.4444 \\ &= 0.4444\end{aligned}$$

2. excellent与 poor合并:

$$\begin{aligned}\text{Gini}(D, \text{excellent} + \text{poor}) &= 1 - (\frac{2}{5})^2 - (\frac{3}{5})^2 \\ &= 0.48 \\ \text{Gini}(D, \text{fair}) &= 1 - (\frac{3}{4})^2 - (\frac{1}{4})^2 \\ &= 0.375 \\ \text{Gini}(D, \text{Credit}) &= \frac{5}{9} * 0.48 + \frac{4}{9} * 0.375 \\ &= 0.4333\end{aligned}$$

3. fair 与 poor合并:

$$\begin{aligned}\text{Gini}(D, \text{fair} + \text{poor}) &= 1 - (\frac{4}{7})^2 - (\frac{3}{7})^2 \\ &= 0.4898 \\ \text{Gini}(D, \text{excellent}) &= 1 - (\frac{1}{2})^2 - (\frac{1}{2})^2 \\ &= 0.5 \\ \text{Gini}(D, \text{Credit}) &= \frac{7}{9} * 0.4898 + \frac{2}{9} * 0.5 \\ &= 0.4921\end{aligned}$$

- **以Term为特征进行划分**:

$$\begin{aligned}\text{Gini}(D, 3\text{yrs}) &= 1 - (\frac{3}{5})^2 - (\frac{2}{5})^2 \\ &= 0.48 \\ \text{Gini}(D, 5\text{yrs}) &= 1 - (\frac{2}{4})^2 - (\frac{2}{4})^2 \\ &= 0.5 \\ \text{Gini}(D, \text{Term}) &= \frac{5}{9} * 0.48 + \frac{4}{9} * 0.5 \\ &= 0.4889\end{aligned}$$

- **以Income为特征进行划分**:

$$\begin{aligned}\text{Gini}(D, \text{high}) &= 1 - (\frac{3}{5})^2 - (\frac{2}{5})^2 \\ &= 0.48 \\ \text{Gini}(D, \text{low}) &= = 1 - (\frac{2}{4})^2 - (\frac{2}{4})^2 \\ &= 0.5 \\ \text{Gini}(D, \text{Income}) &= \frac{5}{9} * 0.48 + \frac{4}{9} * 0.5 \\ &= 0.4889\end{aligned}$$

比较$\text{Gini}(D, \text{Credit})$ 、$\text{Gini}(D, \text{Term})$ 、$\text{Gini}(D, \text{Income})$ 可知按照特征Credit进行划分且将excellent与poor合并的基尼指数最小，所以决策树的根节点就取Credit。
此时的树为:

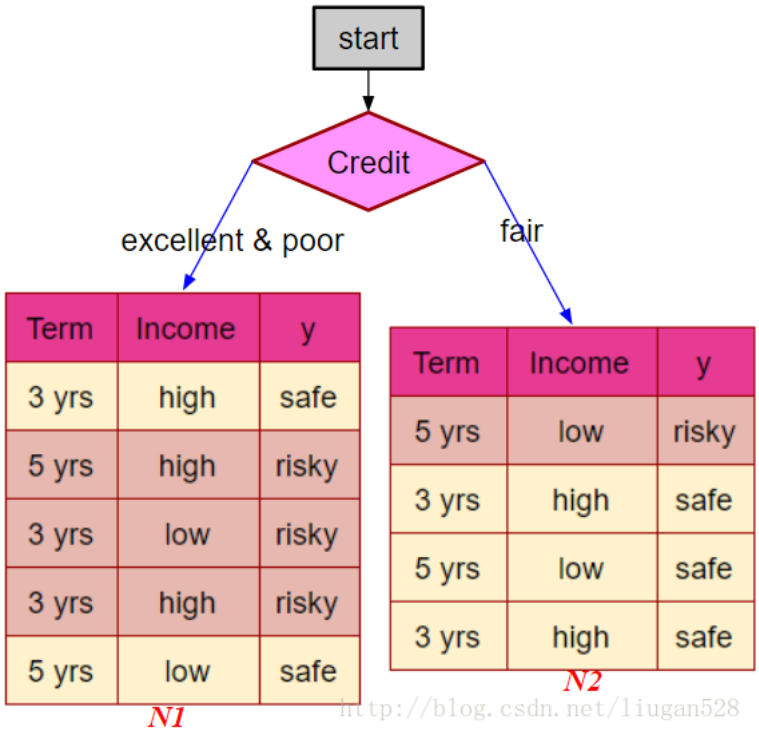

图14

下面需要分别对N_1和N_2进行划分：

- 对于N_1：
  以Term为特征进行划分：

$$\begin{aligned} Gini(N_1, 3yrs) &= 1 - (\frac{1}{3})^2 - (\frac{2}{3})^2 \\ &= 0.4444 \\ Gini(N_1, 5yrs) &= 1 - (\frac{1}{2})^2 - (\frac{1}{2})^2 \\ &= 0.5 \\ Gini(N_1, Term) &= \frac{3}{5} * 0.4444 + \frac{2}{5} * 0.5 \\ &= 0.4666 \end{aligned}$$

  以Income为特征进行划分：

$$\begin{aligned} Gini(N_1, high) &= 1 - (\frac{1}{3})^2 - (\frac{2}{3})^2 \\ &= 0.4444 \\ Gini(N_1, low) &= 1 - (\frac{1}{2})^2 - (\frac{1}{2})^2 \\ &= 0.5 \\ Gini(N_1, Income) &= \frac{3}{5} * 0.4444 + \frac{2}{5} * 0.5 \\ &= 0.4666 \end{aligned}$$

- 对于N_2：
  以Term为特征进行划分：

$$\begin{aligned} Gini(N_2, 3yrs) &= 1 - (\frac{2}{2})^2 - (\frac{0}{2})^2 \\ &= 0 \\ Gini(N_2, 5yrs) &= 1 - (\frac{1}{2})^2 - (\frac{1}{2})^2 \\ &= 0.5 \\ Gini(N_2, Term) &= \frac{2}{4} * 0 + \frac{2}{4} * 0.5 \\ &= 0.25 \end{aligned}$$

  以Income为特征进行划分：

$$\begin{aligned} Gini(N_2, high) &= 1 - (\frac{2}{2})^2 - (\frac{0}{2})^2 \\ &= 0 \\ Gini(N_2, low) &= 1 - (\frac{1}{2})^2 - (\frac{1}{2})^2 \\ &= 0.5 \\ Gini(N_2, Income) &= \frac{2}{4} * 0 + \frac{2}{4} * 0.5 \\ &= 0.25 \end{aligned}$$

根据上面的计算可知对于N1和N2利用Term进行划分和利用Income进行划分基尼指数都是相等的。此时满足了构造决策树终止的条件，因此算法终止。从这里我们也可以知道，该问题使用决策树算法并不能得到很好的解决。

CART回归树

学习决策树可归结为对实例空间进行划分，使得每个隔离的空间都具有较小的方差。在回归问题中，特征值是连续型而非二值型的，CART构造回归树就是找到合适的特征对数据集D进行划分使得每个划分后的子数据集方差最小。
定义数据集D的方差为各个元素到该数据集均值的均方距离：

$$Var(D) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2 \tag{7}$$

其中，N表示数据集中元素的个数，y_i为每个样本的取值，$\bar{y}$表示数据集D的均值。
如果根据特征F对数据集D进行划分，将数据集D划分成了m个互斥子集$\{D_1, D_2, \dots, D_m\}$，则加权平均方差定义为：

$$\begin{aligned} Var(\{D_1, D_2, \dots, D_m\}) &= \sum_{j=1}^{m}\frac{|D_j|}{N}Var(D_j) \\ &= \sum_{j=1}^{m}\frac{|D_j|}{N}(\frac{1}{|D_j|}\sum_{k=1}^{|D_j|}y^2 - \bar{y_j}^2) \\ &= \frac{1}{N}\sum_{i=1}^{N}y_i^2 - \sum_{j=1}^{m}\frac{|D_j|}{N}\bar{y_j}^2 \\ &= \frac{1}{N}\sum_{j=1}^{m}y_j^2 - \bar{y}^2 \end{aligned} \tag{8}$$

其中$|D_j|$用作第j个划分子集的元素个数。
从公式(7)可以看出，方差是集合中元素平方的均值与均值的平方之差。因此，选择特征使得所有可能的划分子集的方差最小化等价于选择特征使得所有可能的划分子集的加权平均最大化。
利用下图中的样本来说明CART算法构造回归树的过程：

#	Model	Condition	Leslie	Price
1	B3	excellent	no	4513
2	T202	fair	yes	625
3	A100	good	no	1051
4	T202	good	no	270
5	M102	good	yes	870
6	A100	excellent	no	1770
7	T202	fair	no	99
8	A100	good	yes	1900
9	E112	fair	no	77

图15

图15的数据集D总共涉及三种特征，因此三种可能的划分方案为：

$$\begin{aligned} Model &= [A100, B3, E112, M102, T202] \rightarrow [1051, 1770, 1900][4513][77][870][99, 270, 625] \\ Condition &= [excellent, good, fair] \rightarrow [1770, 4513][270, 870, 1051, 1900][77, 99, 625] \\ Leslie &= [yes, no] \rightarrow [625, 870, 1900][77, 99, 270, 1051, 1770, 4513] \end{aligned}$$

用Ave表示均值，WeiSquAve表示方均值的加权平均，则对于图15的数据：

- 根据特征Model进行划分：

$$\begin{aligned}
Ave(A100) &= \frac{1051+1770+1900}{3} \\
&= 1573.6667 \\
Ave(B3) &= 4513 \\
Ave(E112) &= 77 \\
Ave(M102) &= 870 \\
Ave(T202) &= \frac{99+270+625}{3} \\
&= 331.3333 \\
WeiSquAve(D,Model) &= \frac{3}{9}*Ave^2(A100)+\frac{1}{9}*Ave^2(B3)+\frac{1}{9}*Ave^2(E112)+\frac{1}{9}*Ave^2(M102) \\
&\quad +\frac{3}{9}*Ave^2(T202) \\
&= 3.2098\cdot 10^6
\end{aligned}$$

- 根据特征Condition进行划分：

$$\begin{aligned}
Ave(excellent) &= \frac{1770+4513}{2} \\
&= 3141.5 \\
Ave(good) &= \frac{270+870+1051+1900}{4} \\
&= 1022.75 \\
Ave(fair) &= \frac{77+99+625}{3} \\
&= 267 \\
WeiSquAve(D,Condition) &= \frac{2}{9}*Ave^2(excellent)+\frac{4}{9}*Ave^2(good)+\frac{3}{9}*Ave^2(fair) \\
&= 2.6818\cdot 10^6
\end{aligned}$$

- 根据特征Leslie进行划分：

$$\begin{aligned}
Ave(yes) &= \frac{625+870+1900}{3} \\
&= 1131.6667 \\
Ave(no) &= \frac{77+99+270+1051+1770+4513}{6} \\
&= 1296.6667 \\
WeiSquAve(D,Leslie) &= \frac{3}{9}*Ave^2(yes)+\frac{6}{9}*Ave^2(no) \\
&= 1.5478\cdot 10^6
\end{aligned}$$

比较WeiSquAve(D, Model) WeiSquAve(D, Condition) WeiSquAve(D, Leslie)可知WeiSquAve(D, Model)最大，因此应该选择特征Model进行划分，该步划分的结果为：

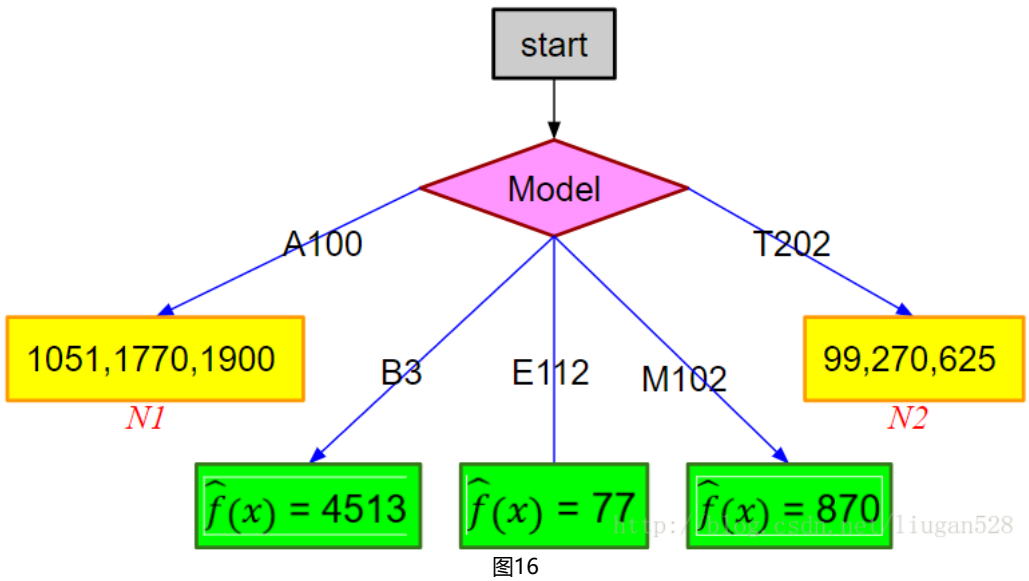

图16

接下来需要对数据集N_1和N_2分别选择特征继续划分。
对于N_1

- 根据特征Condition进行划分：

$$\begin{aligned} \text{Ave(excellent)} &= 1770 \\ \text{Ave(good)} &= \frac{1051+1900}{2} \\ &= 1475.5 \\ \text{Ave(fair)} &= 0 \\ \text{WeiSquAve}(N_1, \text{Condition}) &= \frac{1}{3}*\text{Ave}^2(\text{excellent}) + \frac{2}{3}*\text{Ave}^2(\text{good}) + \frac{0}{3}*\text{Ave}^2(\text{fair}) \\ &= 2.4957 \cdot 10^6 \end{aligned}$$

- 根据特征Leslie进行划分：

$$\begin{aligned} \text{Ave(yes)} &= 1900 \\ \text{Ave(no)} &= \frac{1051+1770}{2} \\ &= 1410.5 \\ \text{WeiSquAve}(N_1, \text{Leslie}) &= \frac{1}{3}*\text{Ave}^2(\text{yes}) + \frac{2}{3}*\text{Ave}^2(\text{no}) \\ &= 2.5297 \cdot 10^6 \end{aligned}$$

比较$\text{WeiSquAve}(N_1, \text{Condition})$ $\text{WeiSquAve}(N_1, \text{Leslie})$可知$\text{WeiSquAve}(N_1, \text{Leslie})$最大，因此应该选择特征Leslie进行划分。

对于N_2

- 根据特征Condition进行划分：

$$\begin{aligned} \text{Ave(excellent)} &= 0 \\ \text{Ave(good)} &= 270 \\ \text{Ave(fair)} &= \frac{99+625}{2} \\ &= 362 \\ \text{WeiSquAve}(N_2, \text{Condition}) &= \frac{0}{3}*\text{Ave}^2(\text{excellent}) + \frac{1}{3}*\text{Ave}^2(\text{good}) + \frac{2}{3}*\text{Ave}^2(\text{fair}) \\ &= 111,662.6667 \end{aligned}$$

- 根据特征Leslie进行划分：

$$\begin{aligned} \text{Ave(yes)} &= 625 \\ \text{Ave(no)} &= \frac{99+270}{2} \\ &= 184.5 \\ \text{WeiSquAve}(N_2, \text{Leslie}) &= \frac{1}{3}*\text{Ave}^2(\text{yes}) + \frac{2}{3}*\text{Ave}^2(\text{no}) \\ &= 152,901.8333 \end{aligned}$$

比较$\text{WeiSquAve}(N_2, \text{Condition})$ $\text{WeiSquAve}(N_2, \text{Leslie})$可知$\text{WeiSquAve}(N_2, \text{Leslie})$最大，因此应该选择特征Leslie进行划分。

因此，最终构造的决策回归树为：

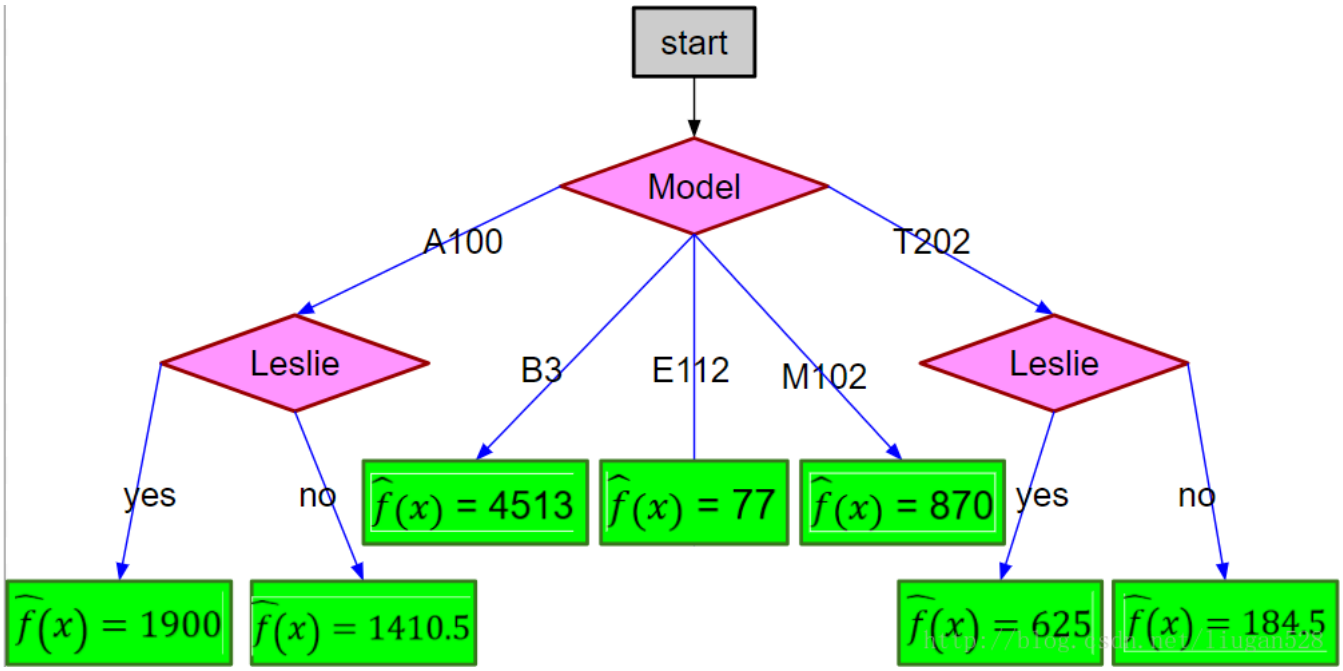

图17

- scikit-learn实现CART分类决策树：
  使用图9的数据进行演示：

```
import numpy as np
import pandas as pd
from sklearn import tree
#导出为pdf所需package
import graphviz

#scikit learn中CART分类树
CARTClassificationTree = tree.DecisionTreeClassifier()

#准备数据
adict = {'Outlook':'Sunny','Sunny','Overcast','Rainy','Rainy','Rainy','Overcast',
                    'Sunny','Sunny','Rainy','Sunny','Overcast','Overcast','Rainy'],
         'Temperature':[85,80,83,70,68,65,64,72,69,75,75,72,81,71],
         'Humidity':[85,90,78,96,80,70,65,95,70,80,70,90,75,80],
         'Windy':False,True,False,False,False,True,True,
                 False,False,False,True,True,False,True]}
dfx = pd.DataFrame(adict)
#进行one-hot编码
onehot_dfx = pd.get_dummies(dfx)
#给X赋值
dataX = onehot_dfx.values
#给Y赋值
dataY = np.array(['No','No','Yes','Yes','Yes','No','Yes',
                  'No','Yes','Yes','Yes','Yes','Yes','No'])

#训练模型
CARTClassificationTree.fit(dataX,dataY)

#输出CART分类决策树并导出为pdf格式
dot_data = tree.export_graphviz(CARTClassificationTree,out_file=None,
                                class_names=npY,feature_names=onehot_dfx.columns,
                                filled=True,rounded=True,special_characters=True)
graph = graphviz.Source(dot_data)
graph.render('PlayGolf')
```

最终的CART分类决策树为：

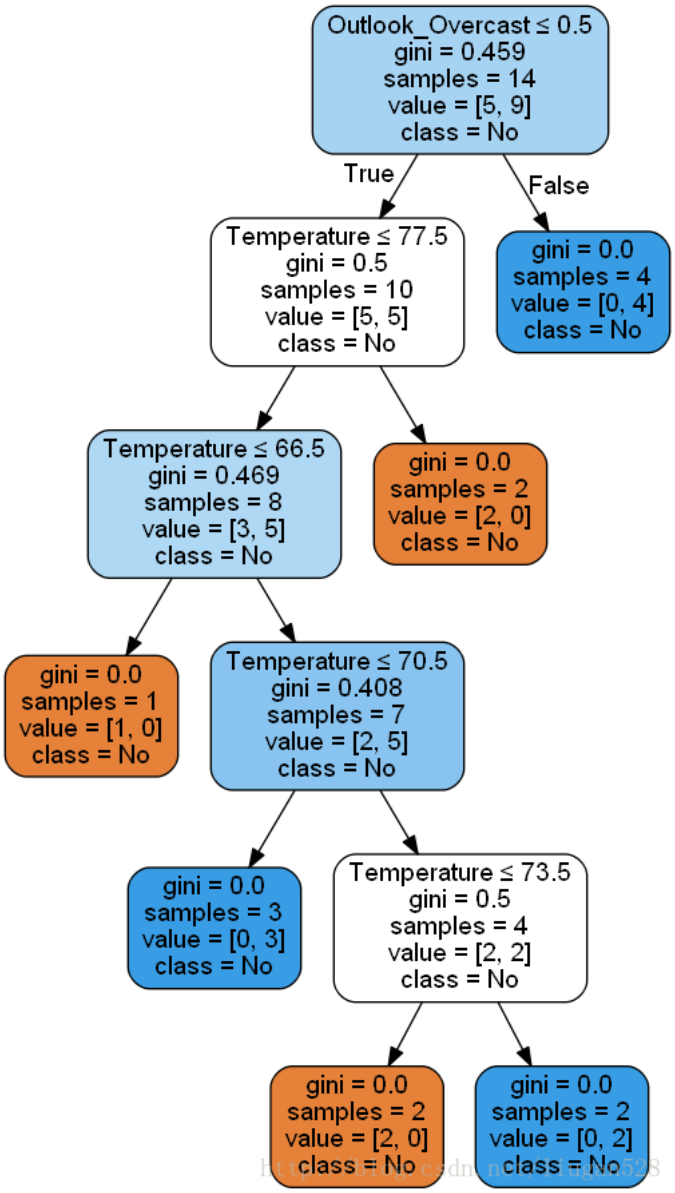

图18

- scikit-learn实现CAR回归决策树：
  使用图15的数据进行演示：

```
import numpy as np
import pandas as pd
from sklearn import tree
#导出为pdf所需package
import graphviz

#scikit learn中CART回归树
CARTRegressionTree = tree.DecisionTreeRegressor()

#准备数据
adict = {'Model':['B3','T202','A100','T202','M102','A100','T202','A100','E112'],
         'Condition':['excellent','fair','good','good','good','excellent','fair','good','fair'],
         'Leslie':['no','yes','no','no','yes','no','no','yes','no']}
dfx = pd.DataFrame(adict)
#进行one-hot编码
onehot_dfx = pd.get_dummies(dfx)
#给X赋值
dataX = onehot_dfx.values
#给Y赋值
dataY = np.array([4513,625,1051,270,870,1770,99,1900,77])

#训练模型
CARTRegressionTree.fit(dataX,dataY)

#输出CART分类决策树并导出为pdf格式
dot_data = tree.export_graphviz(CARTRegressionTree,out_file=None,feature_names=onehot_dfx.columns,
                                filled=True,rounded=True,special_characters=True)
graph = graphviz.Source(dot_data)
graph.render('Price')
```

最终的CART回归决策树为：

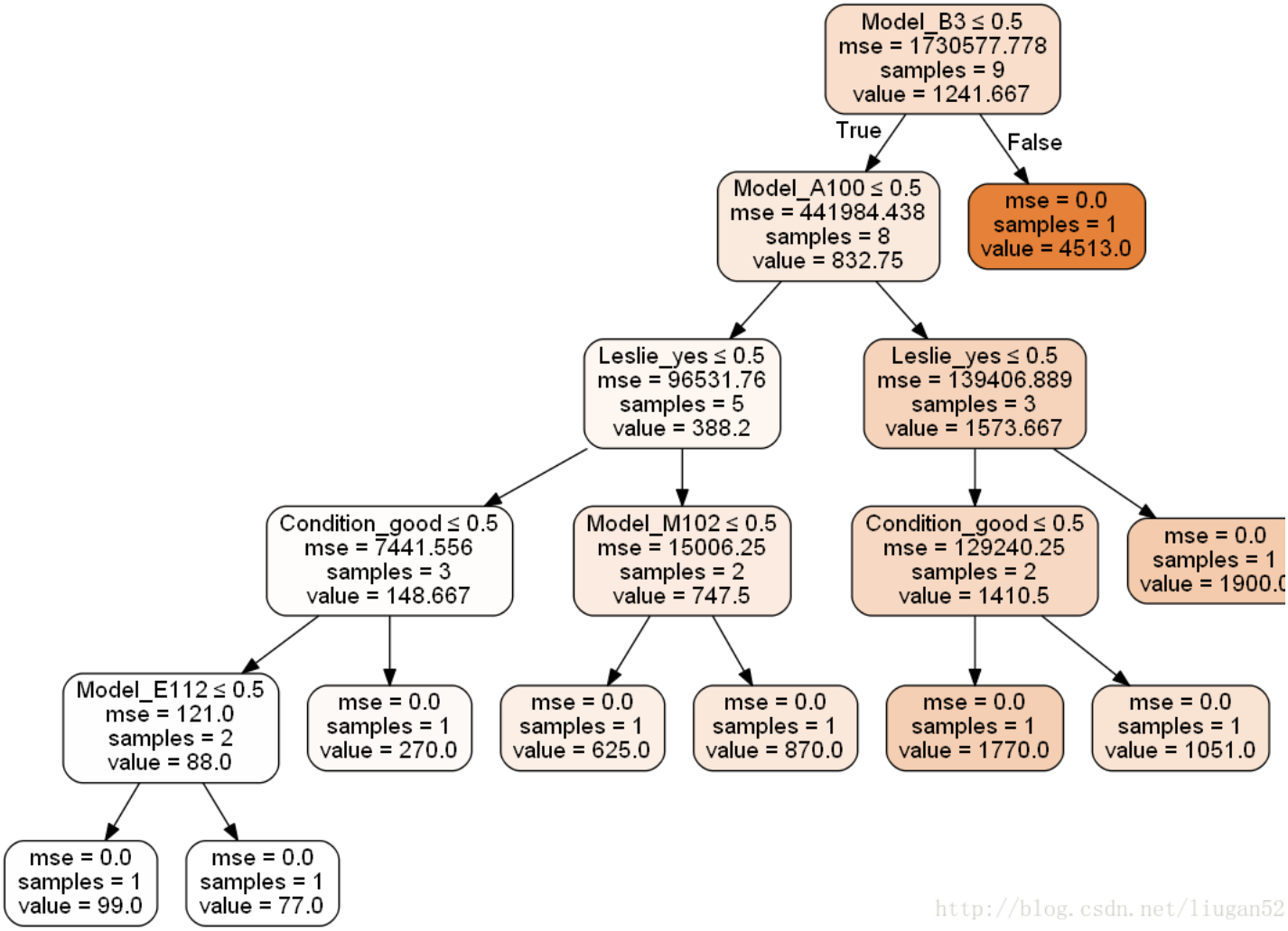

图19

图19

更多完整资料请移步github：
https://github.com/GarryLau/MachineLearning