用户画像

一、课前准备

- 熟悉Python的使用
- 熟悉用户画像的基本用法

二、课堂主题

- 本节课主要讲授模型融合方法及用户画像知识。

三、课堂目标

- 用户画像的概念
- 用户画像的制作
- 模型融合方法
- 算法总结

四、知识要点

4.1 用户画像

4.1.1 何为用户画像

- 虚拟的
- 基于大量用户积累下的数据而产生
- 结合相应的需求和场景沉淀出的一些列标签

4.1.2 用户画像详解

见PPT

4.2 模型融合

4.2.1 Voting

假设对于一个二分类问题，有3个基础模型，那么就采取投票制的方法，投票多者确定为最终的分类。

对于分类任务来说，学习器h_i将从类别标记集合$c_1, c_2, \ldots, c_N$中预测出一个标记，最常见的结合策略是使用投票法。为了便于讨论，我们将h_i在样本x上的预测输出表示为一个N维向量$(h_i^1(x); h_i^2(x); \ldots; h_i^N(x);)$，其中$h_i^j(x)$是$h_i$在类别标记$c_j$上的输出。

- 绝对多数投票法

$$H(x) = \begin{cases} c_j, if \sum_{i=1}^{T} h_i^j(x) > 0.5 \sum_{k=1}^{N} \sum_{i=1}^{T} h_i^k(x) \\ reject, otherwise \end{cases}$$

若某标记投票数过半，则预测为该标记，否则拒绝预测。

- 相对多数投票法

$$H(x)=C_{arg_j max \sum_{i=1}^{T} h_i^j(x)}$$

预测为得票最多的标记，若同时有多个标记获得高票，则从中随机选取一个。

- 加权投票法

$$H(x)=C_{arg_j max \sum_{i=1}^{T} w_i h_i^j(x)}$$

与加权平均法类似，w_i是h_i的权重，通常$w_i \geqslant 0, \sum_{i=1}^{T} w_i = 1$。

标准的绝对多数投票法提供了“拒绝预测”选项，这在可靠性要求较高的学习任务中是一个很好的机制，但若学习任务要求必须提供预测结果，则绝对多数投票法将退化为相对多数投票法，因此在不允许拒绝预测的任务中，绝对多数、相对多数投票法统称为“多数投票法”。

我们在上面的几个投票法中没有限制个体学习器输出值的类型，在现实任务中，不同类型个体学习器可能产生不同类型的$h_i^j(x)$值，常见的我们根据输出值把投票法分为“硬投票”和“软投票”。

- 硬投票：若h_i将样本x预测为类别c_j则取值为1，否则为0，那么这种使用类标记的投票就叫做“硬投票”。
- 软投票：相当于对后验概率$P(c_j|x)$的一个估计，这种使用累概率的投票就叫做”软投票“。

4.2.2 Averaging

对于回归问题，一个简单直接的思路是取平均。稍稍改进的方法是进行加权平均。权值可以用排序的方法确定，举个例子，比如A、B、C三种基本模型，模型效果进行排名，假设排名分别是1，2，3，那么给这三个模型赋予的权值分别是3/6、2/6、1/6。

对于数值型输出$h_i(x) \in R$:

- 简单平均法：

$$H(x)=\frac{1}{T}\sum_{i=1}^{T} h_i(x)$$

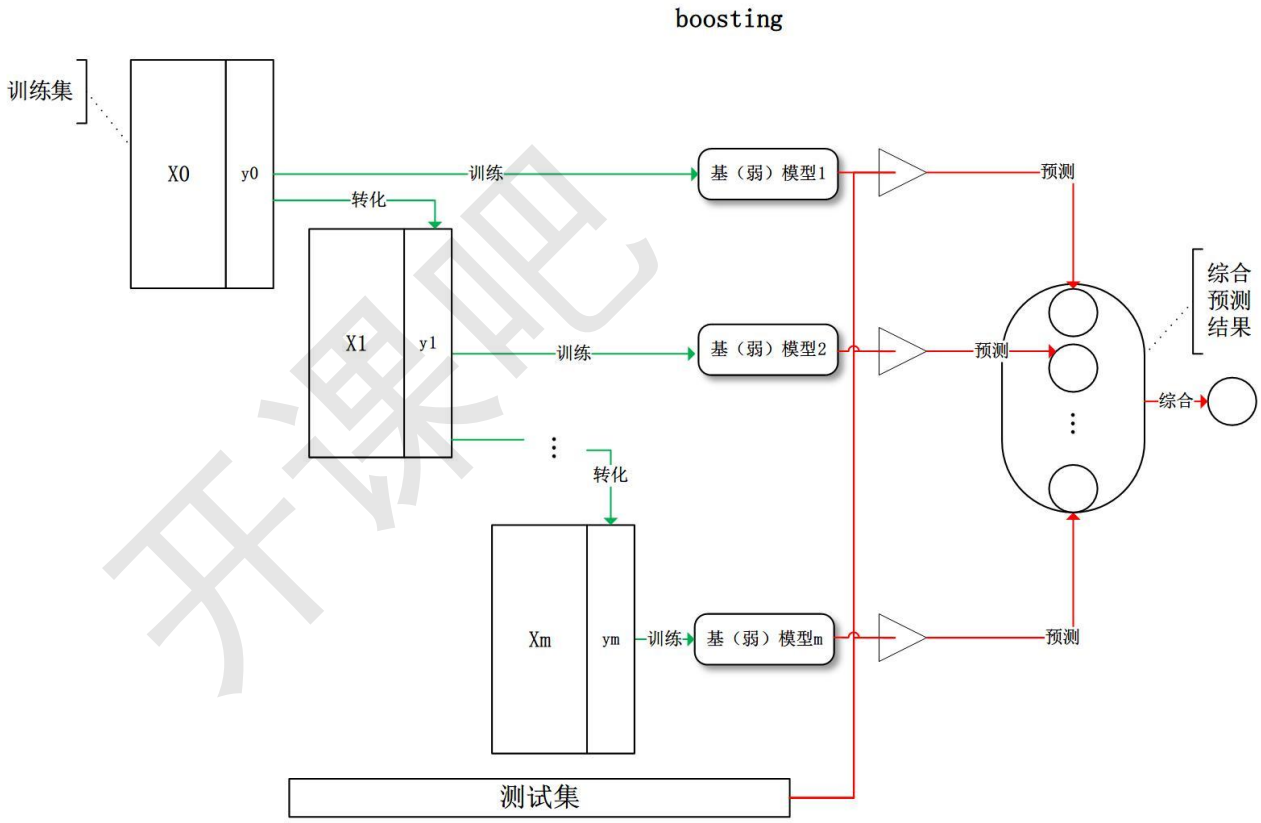
- 加权平均法：

$$H(x)=\sum_{i=1}^{T} w_i h_i(x)$$

简单平均法可以看作是加权平均法令$w_i=\frac{1}{T}$的特例，而加权平均的权值都是从训练数据中学习的，由于真实情况下训练数据有缺失或者噪声，导致其得到的权值不一定真实可靠，所以从这个角度来说，加权平均法不一定优于简单平均法。其实有这样的一个思想：对于个体学习器的性能相近时，我们采用简单平均；而个体学习器差异较大的，我们采用加权平均。

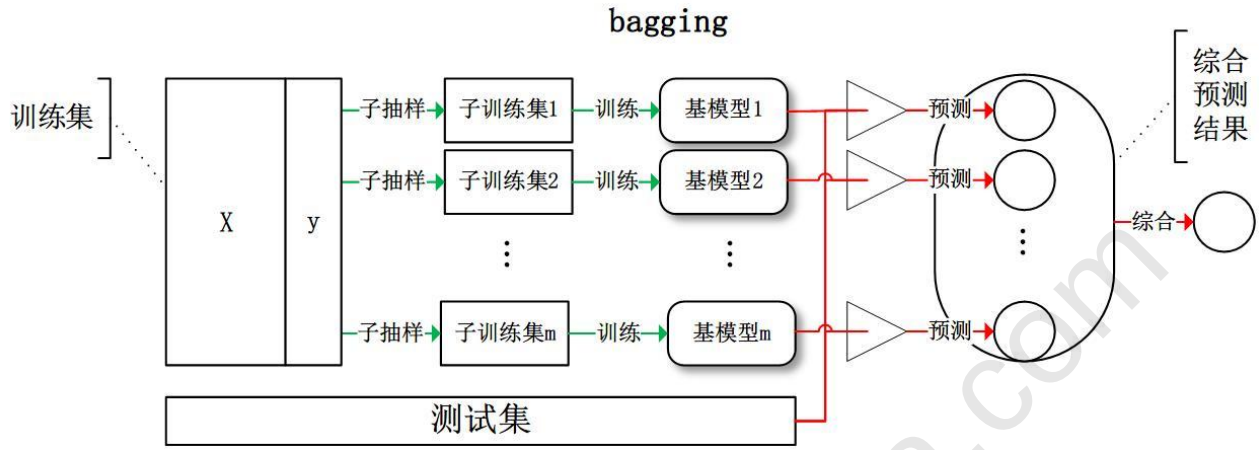
4.2.3 Bagging

Bagging是一种并行的集成学习方法，基学习器的训练没有先后顺序，同时进行。Bagging 采用“有放回”采样，对于包含m 个样本的训练集，进行m次有放回的随机采样操作，从而得到m个样本的采样集，按照这样的方式重复进行，我们就可以得到 T 个包含m个样本的训练集，训练出来T个基学习器，然后对这些基学习器的输出进行结合。



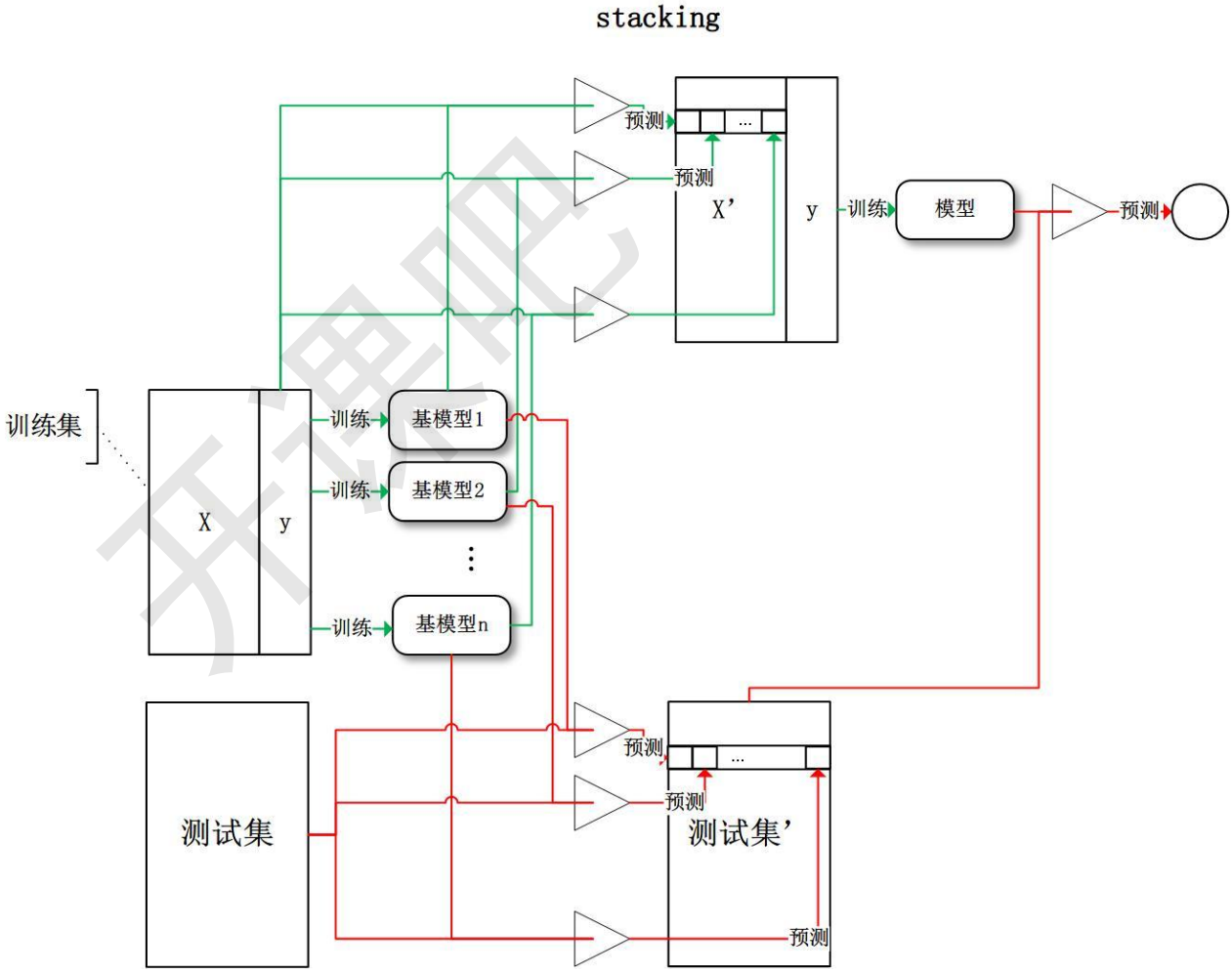
4.2.4 Boosting

Boosting 是一种可以将弱学习器提升为强学习器的算法。这是一种串行的思想，序列化进行。基本思想是：增加前一个基学习器预测错误的样本的权值，使得后续的基学习器更加关注于这些打错标注的样本，尽可能的纠正这些错误。直到训练出了T个基学习器，最终将这T个基学习器进行加权结合。



4.2.5 Stacking

训练好的所有基模型对训练集进行预测，第j个基模型对第i个训练样本的预测值将作为新的训练集中第i个样本的第j个特征值，最后基于新的训练集进行训练。同理，预测的过程也要先经过所有基模型的预测形成新的测试集，最后再对测试集进行预测，Stacking的流程图如下所示：



stacking 代码演示

```
from sklearn import datasets

iris = datasets.load_iris()
X, y = iris.data[:, 1:3], iris.target

from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from mlxtend.classifier import StackingClassifier
import numpy as np

clf1 = KNeighborsClassifier(n_neighbors=1)
clf2 = RandomForestClassifier(random_state=1)
clf3 = GaussianNB()
lr = LogisticRegression()
sclf = StackingClassifier(classifiers=[clf1, clf2, clf3],
                          use_probas=True,
                          average_probas=False,
                          meta_classifier=lr)
```

```
print('3-fold cross validation:\n')

for clf, label in zip([clf1, clf2, clf3, sclf],
                      ['KNN',
                       'Random Forest',
                       'Naive Bayes',
                       'StackingClassifier']):

    scores = model_selection.cross_val_score(clf, X, y,
                                             cv=3, scoring='accuracy')
    print("Accuracy: %0.2f (+/- %0.2f) [%s]"
          % (scores.mean(), scores.std(), label))
```

```
3-fold cross validation:

Accuracy: 0.91 (+/- 0.01) [KNN]
Accuracy: 0.95 (+/- 0.01) [Random Forest]
Accuracy: 0.91 (+/- 0.02) [Naive Bayes]
Accuracy: 0.92 (+/- 0.02) [StackingClassifier]
```

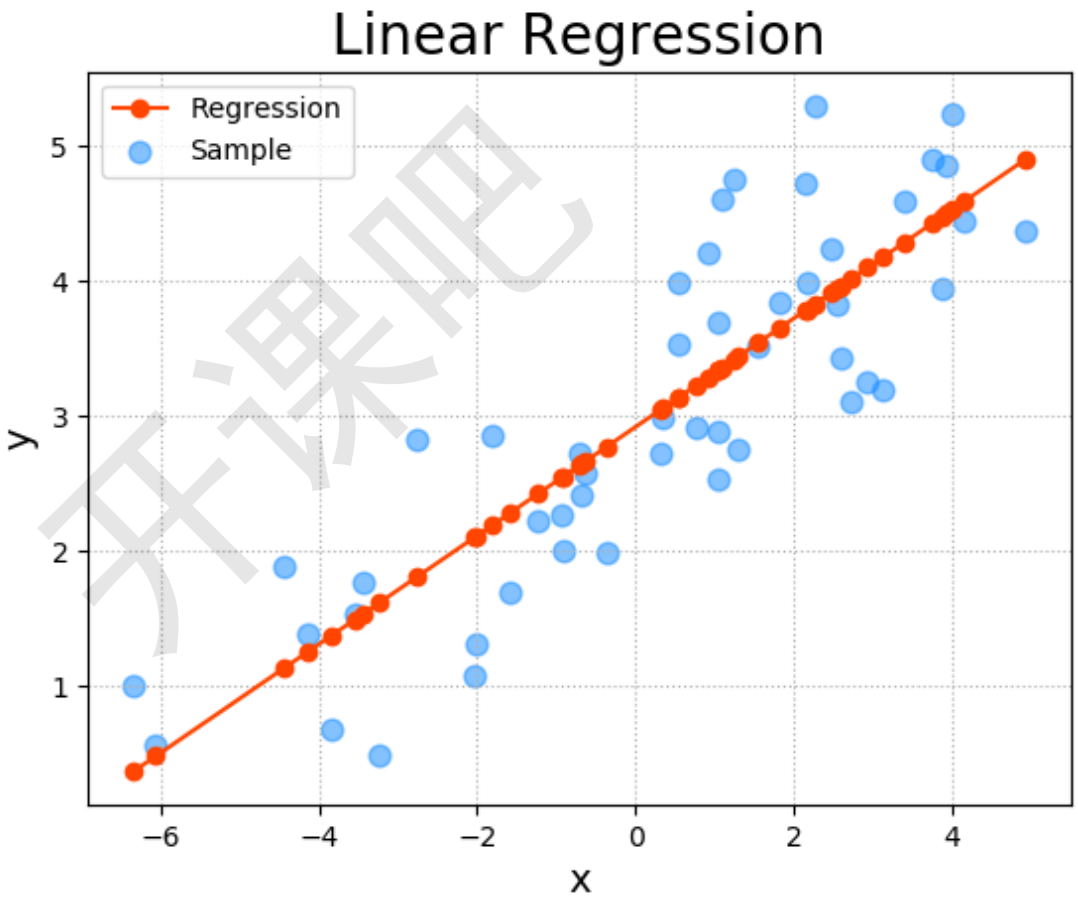
4.3 基于用户输入信息的用户画像

- 目标：根据用户的（输入法）输入信息，进行分类，预测出用户的年龄，性别，教育程度
- 算法：LR、SVM、Stacking对比
- 实现：见代码

五、总结

5.1 模型对比

线性回归



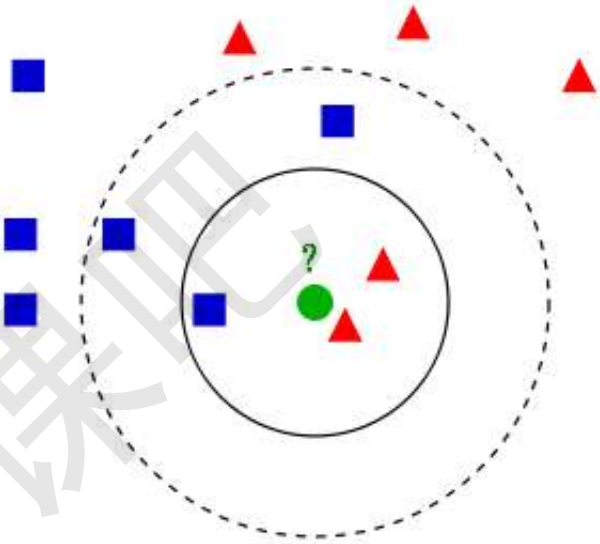
- 思路：线性回归假设目标值与特征之间线性相关，即满足一个多元一次方程。通过构建损失函数，来求解损失函数最小时的参数w和b。

$\hat{y} = wx + b$

- 优点：
  1.模型简单，容易实现
  2.许多非线性模型的基础
  3.机器学习的基石
- 缺点：
  1.对于非线性数据或者数据特征间具有相关性多项式回归难以建模
  2.难以很好地表达高度复杂的数据

使用场景：一般简单的回归模型

KNN（K近邻算法）



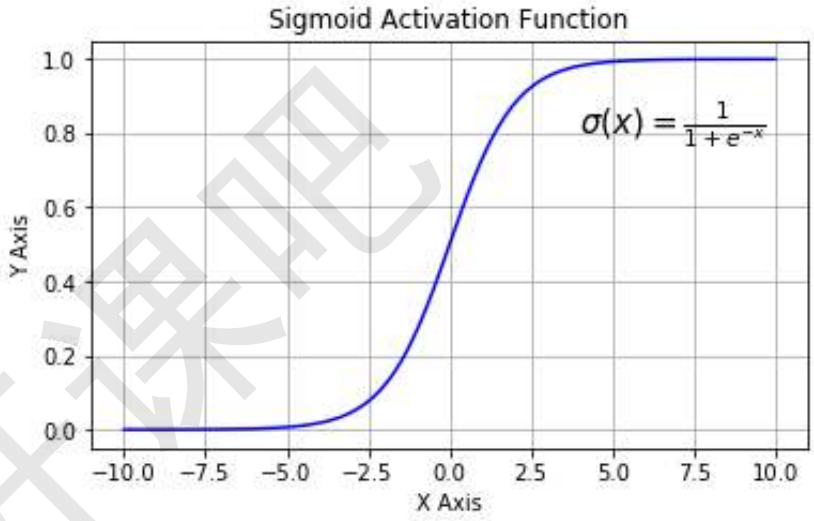
- 思路：对于待判断的点，找到离他最近的几个数据点，根据他们的类型决定待判断点的类型。
- 特点：完全跟着数据走，没有什么数学模型。
- 优点：
  1.理论成熟，思想简单；
  2.可用于非线性；
  3.准确度高；
  4.对异常值不敏感。
- 缺点：
  1.计算量大;
  2.样本不均衡的问题;
  3.需要大量的内存。
- 适用场景：需要一个好解释的模型的时候。

NB（朴素贝叶斯）

	瓜蒂	形状	颜色	类别
1	脱落	圆形	深绿	瓜熟
2	未脱落	尖形	浅绿	瓜生
3	未脱落	圆形	浅绿	瓜生
4	脱落	尖形	青色	瓜熟
5	脱落	圆形	浅绿	瓜熟
6	未脱落	尖形	青色	瓜生
7	脱落	尖形	深绿	瓜熟
8	未脱落	圆形	青色	瓜熟
9	脱落	尖形	浅绿	瓜生
10	未脱落	圆形	深绿	瓜熟

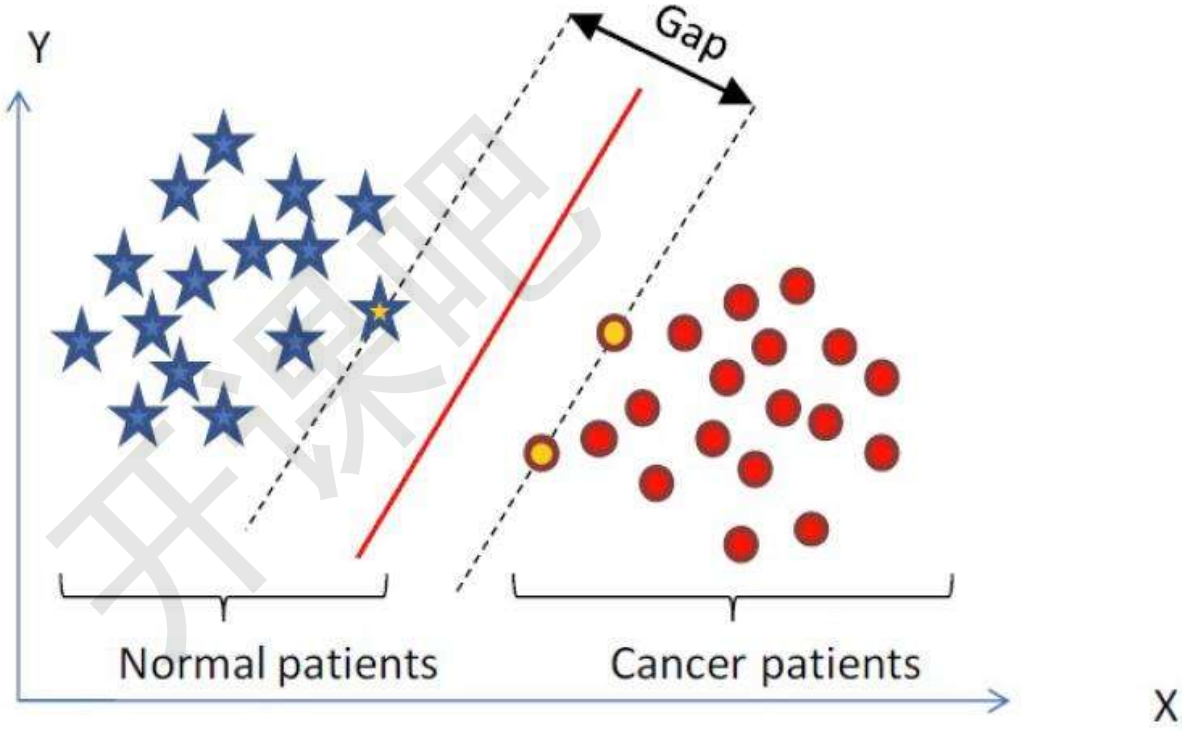
- 条件概率： $P(A|B) = \frac{P(AB)}{P(B)}$
- 全概率： $P(A) = \sum_{i=1}^{n} P(B_i)P(A|B_i)$
- 贝叶斯公式： $P(B_i|A) = \frac{P(B_i)P(A|B_i)}{\sum_{i=1}^{n} P(B_i)P(A|B_i)}$
- 优点：
  1.朴素贝叶斯起源于古典数学理论，有着坚实的数学基础，以及稳定的分类效率；
  2.对小规模的数据表现很好，能进行多分类；
  3.对缺失值不敏感，算法简单。
- 缺点：
  1.需要计算先验概率；
  2.对特征间强相关的模型分类效果不好。
- 适用场景，容易解释，不同维度之间相关性小的模型，不计后果的前提下可以处理高维数据。

LR（逻辑回归）



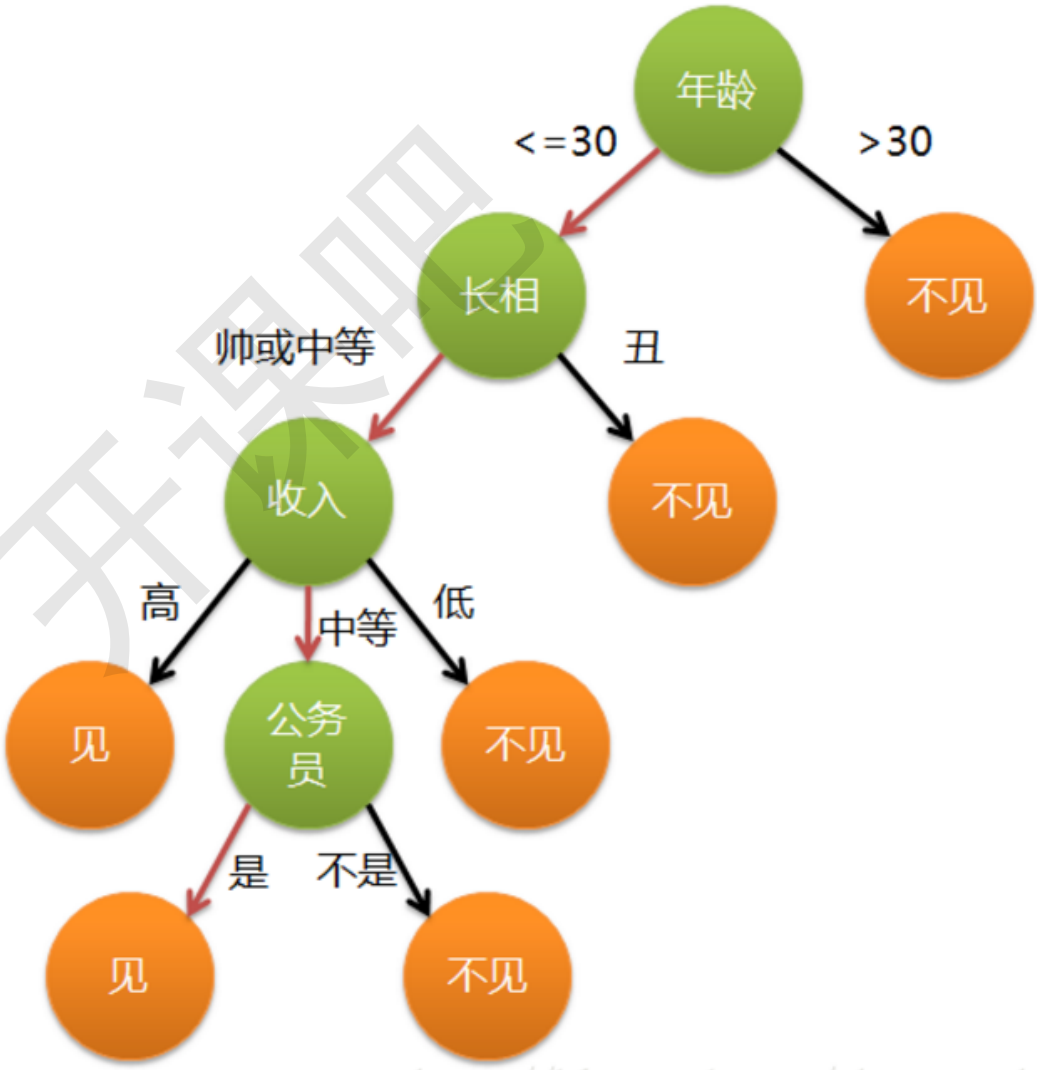
- 核心：
  $sigmoid = \frac{1}{1+e^{-z}}$
- 优点：
  1.实现简单，广泛应用于工业上；
  2.分类时计算量非常小，速度很快，存储资源少；
  3.可观测样本的概率分数。
- 缺点：
  1.特征空间很大时，性能不是很好；
  2.容易前拟合，一般准确度不高；
  3.只能处理二分类线性可分问题。
- 适用场景：很多分类算法的基础组件；用于分析单一因素对某一事件发生的影响因素；用于预测事件发生的概率。

SVM（支持向量机）



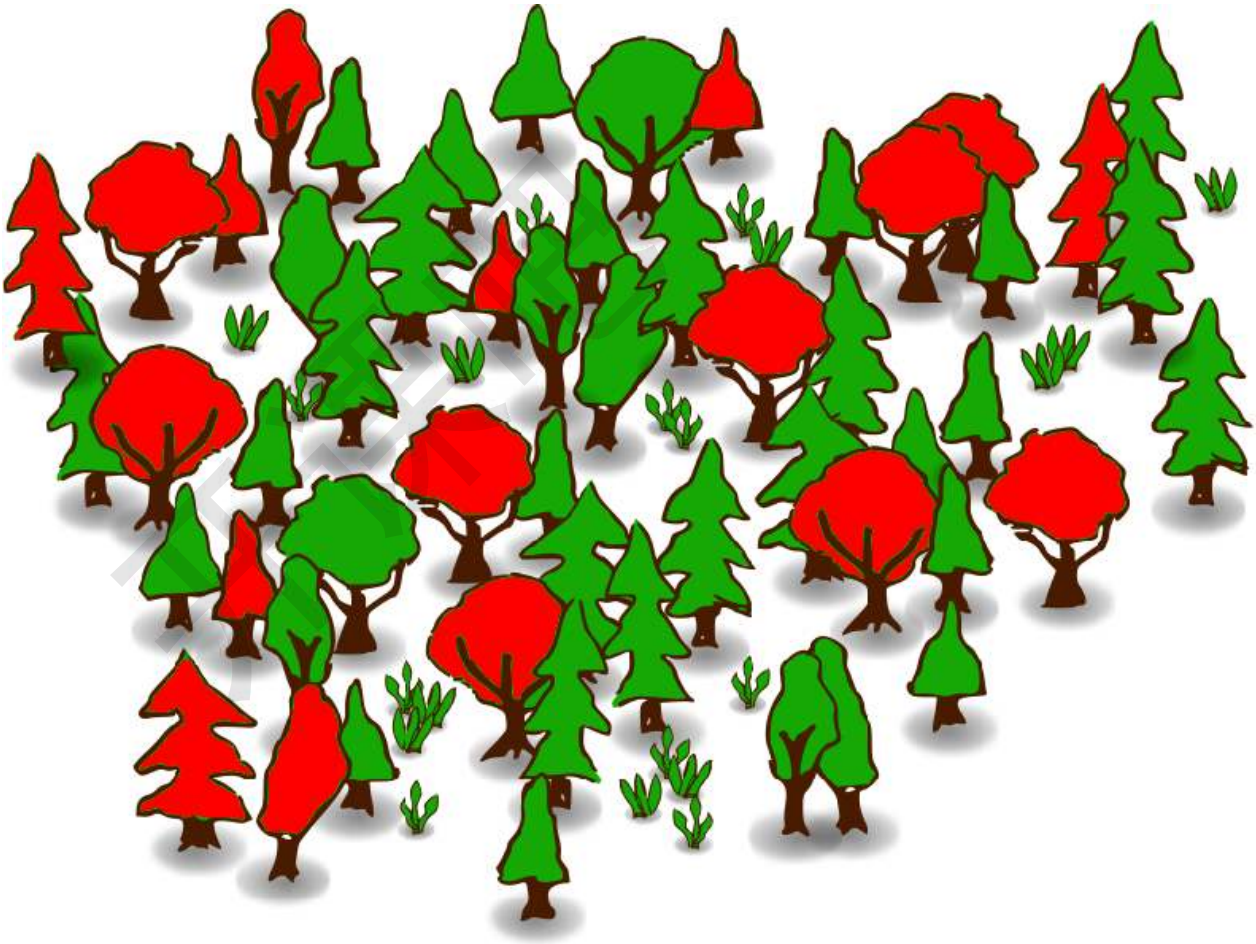
- 核心：找到不同类别之间的分类面，使得两类样本尽量落在面的两边，且离分类面尽量远。
- 优点：
  1.可以解决高维问题，即大型特征空间;
  2.能够处理非线性特征的相互作用；
  3.无需依赖整个数据。
- 缺点：
  1.当观测样本很多的时候，效率不是很高；
  2.对非线性问题没有通用的解决方案，很难找到一个合适的核函数；
  3.对缺失数据敏感。
- 适用场景：在很多数据集上都有优秀的表现，拿到数据就可以尝试一下SVM。

DT（决策树）



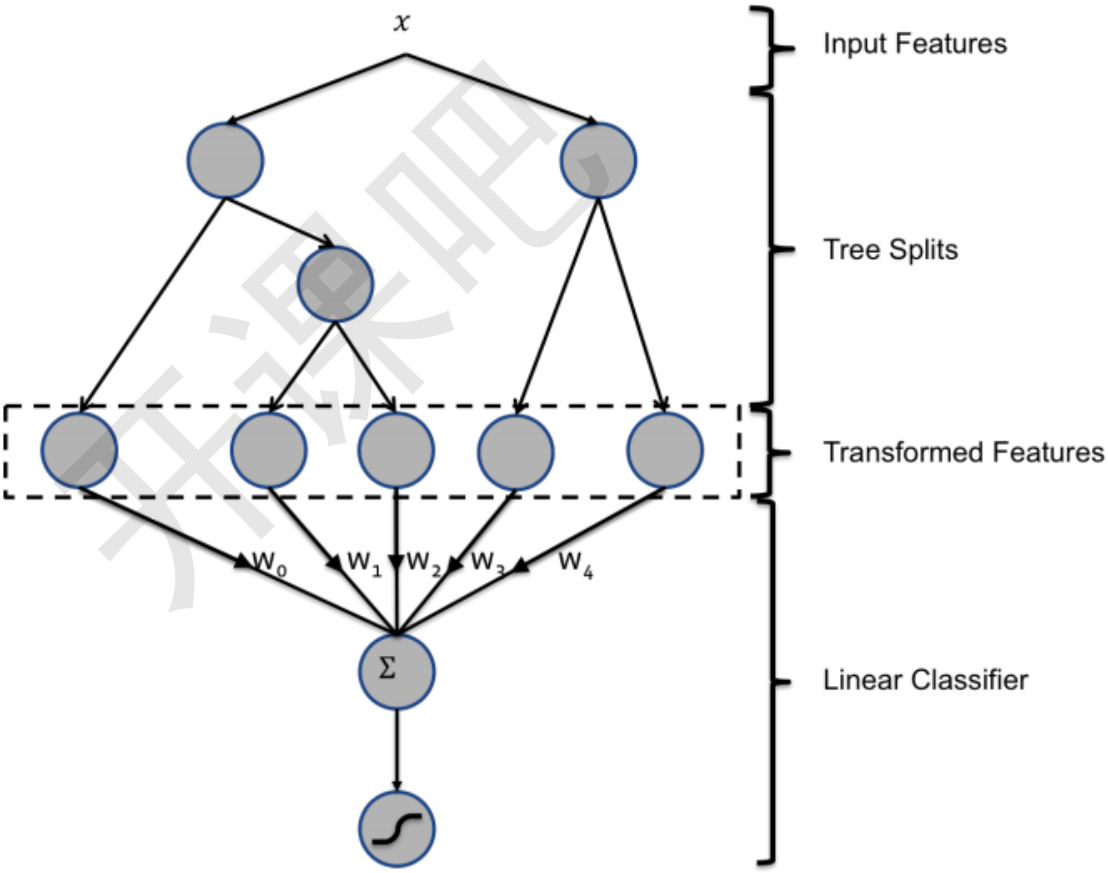
- 核心：信息增益；信息增益比；Gini系数。
- 优点：
  1.计算简单，易于理解，可解释行强；
  2.比较适合有缺失属性的样本；
  3.能够处理不相关的特征；
  4.在短时间内可以对大型数据做出好的结果。
- 缺点：
  1.容易发生过拟合；
  2.易被攻击；
  3.忽略了数据之间的相关性；
  4.各个类别样本数量不一致的数据，信息增益偏向具有更多数值的特征。
- 适用场景：常作为一些算法的基石；它能够生成清晰的基于特征(feature)选择不同预测结果的树状结构，数据分析师希望更好的理解手上的数据的时候往往可以使用决策树。

RF（随机森林）



- 核心：两个随机（随机选取训练样本，随机选取特征），由决策树形成。
- 优点：
  1.可以解决分类和回归问题；
  2.抗过拟合能力强；
  3.稳定性强。
- 缺点：
  1.模型复杂；
  2.计算成本高；
  3.计算时间长。
- 适用场景：数据维度相对低（几十维），同时对准确性有较高的要求；使用随机森林时，不需要调节很多的参数就可以达到很好的效果，所以不知道用什么方法时可以尝试一下。

GBDT



- 原理：计算树的残差，通过前一棵树的残差拟合下一棵树，最终进行残差的加和。
- 优点

  1.预测精度高；

  2.适合低维数据；

  3.能处理非线性数据；

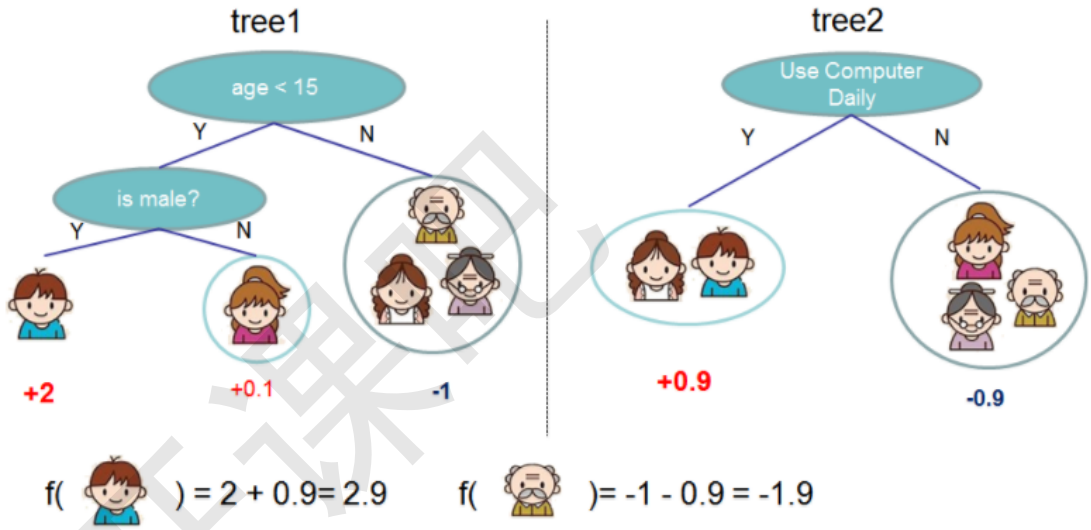
  4.可以灵活处理各种类型的数据，包括连续值和离散值；

  5.在相对少的调参时间情况下，预测的准备率也可以比较高。
- 缺点

  1.由于弱学习器之间存在依赖关系，难以并行训练数据。不过可以通过自采样的SGBT来达到部分并行；

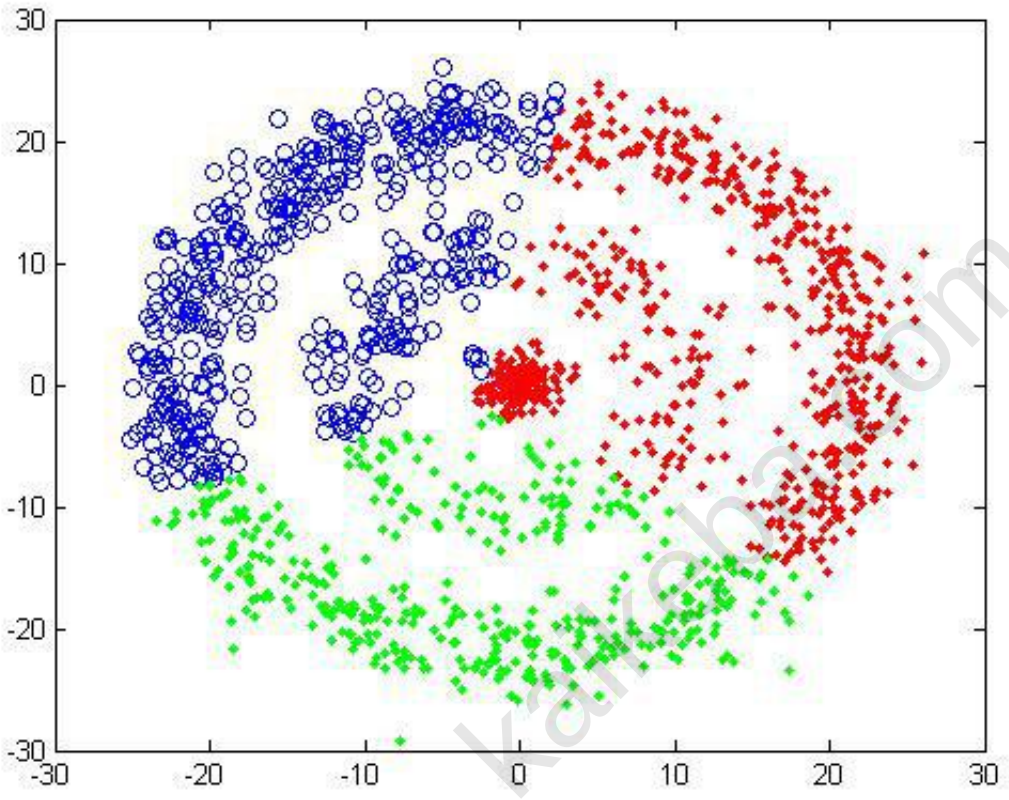
  2.如果数据维度较高时会加大算法的计算复杂度。
- 适用场景：不知道用什么模型时候可以使用的回归/分类模型

XGBoost



- 原理：通过计算伪残差，计算加和
- 传统GBDT以CART作为基分类器，xgboost还支持线性分类器，这个时候xgboost相当于带L1和L2正则化项的逻辑回归（分类问题）或者线性回归（回归问题）。
- 传统GBDT在优化时只用到一阶导数信息，xgboost则对代价函数进行了二阶泰勒展开，同时用到了一阶和二阶导数（能自定义损失函数）。
- gboost在代价函数里加入了正则项，用于控制模型的复杂度。正则项里包含了树的叶子节点个数、每个叶子节点上输出的score的L2模的平方和。正则项降低了模型的复杂度，使学习出来的模型更加简单，防止过拟合，这也是xgboost优于传统GBDT的一个特性。

K-Means



- 原理：物以类聚，人以群分

- 优点：
  1. 原理简单，容易实现
  2. 内存占用小
- 缺点:
  1. K值需要预先给定，属于预先知识，很多情况下K值的估计是非常困难的，对于像计算全部微信用 户的交往圈这样的场景就完全的没办法用K-Means进行。
  2. K-Means算法对初始选取的聚类中心点是敏感的，不同的随机种子点得到的聚类结果完全不同。
  3. K均值算法并不适合所有的数据类型。
  4. 对离群点的数据进行聚类时，K均值也有问题，这种情况下，离群点检测和删除有很大的帮助。
- 应用场景：没有明确标签的情况下。

六、作业

无了