《实战机器学习应用开发》

实战机器学习经典预测算法

回顾

上一章主要对机器学习的流行框架 sk-learn 进行一些梳理，目的是让大家能从更高层次去理解 sklearn 的设计思路，主要内容包括：

- sklearn 是什么？
- sklearn 使用流程
- sklearn 覆盖的机器学习问题
- sklearn 算法选择路径图
- sklearn 的API结构
- sklearn datasets模块
- sklearn 模型训练的基本套路
- 如何构建sklearn机器学习流水线 (Pipline)

本章导读

通过上一章的学习，我相信大家应该对一个 完整的机器学习开发流程 有了一些了解，同时也看到了机器学习算法在建模过程中的作用。前面介绍的预测模型我们主要以逻辑回归这种算法为例，偶尔也提及到了像 决策树，KNN,它们都属于 预测类算法。除了它们以外还有哪些常用算法呢？接下来就让我们开始学习。

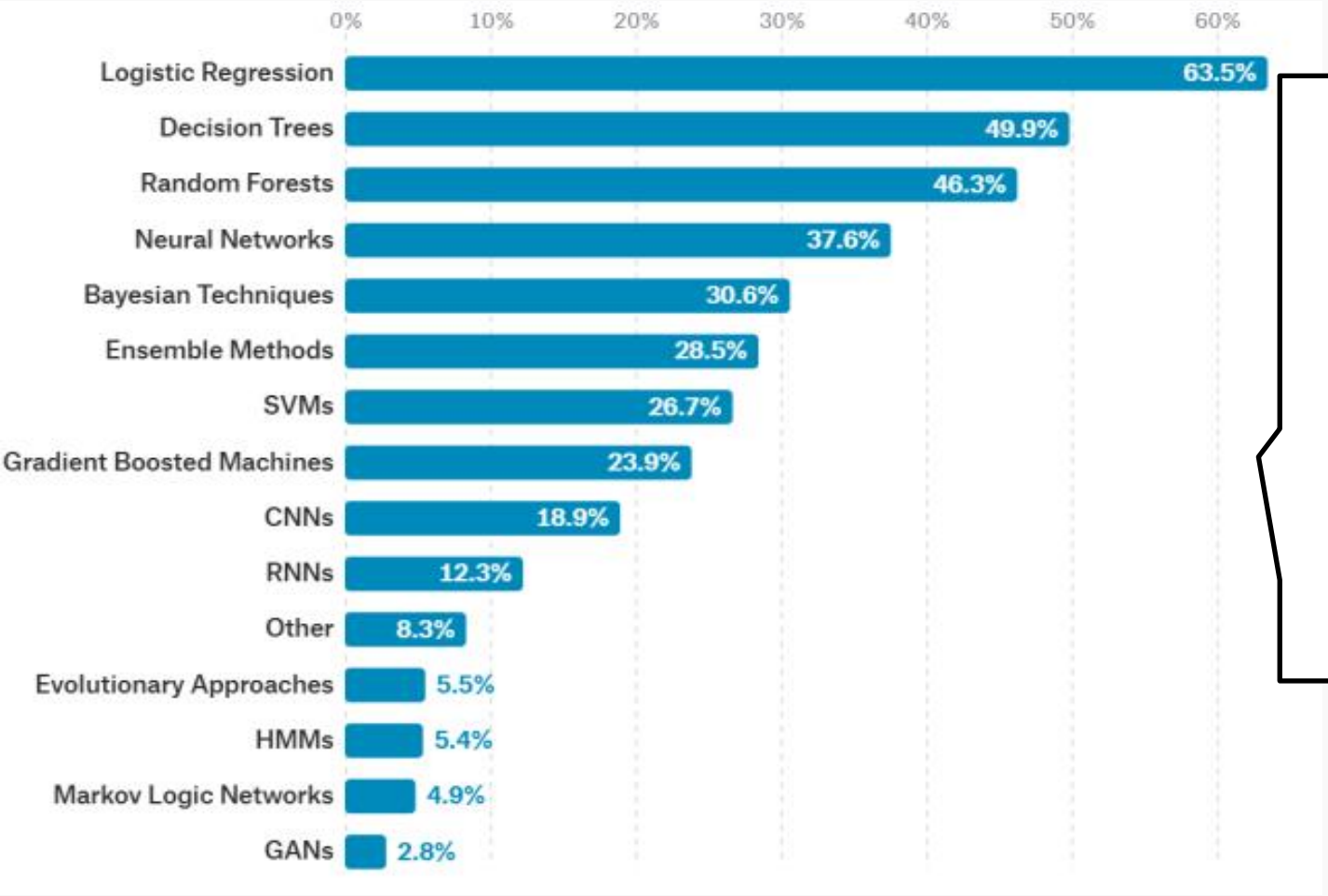
本章目标

通过本章的学习，可以让大家学习到：

- 了解机器学习中常用的预测算法有哪些
- 每个算法的核心原理是什么，从更高层面去解释和理解它们
- 每个算法的优缺点以及使用场景
- 每个算法的 Python 代码实现

一、引言

1.2、Kaggle 调研常用算法



左图来自 **Kaggle** 在 **2017年12月份** 发布的一份关于数据科学家的调研问卷中的数据科学家最常使用的算法。我觉得 **CNN** 算是个分水岭，之前的算法基本属于 **传统机器学习算法**，CNN 及后续的算法则是偏重人工智能和深度学习的算法，这也说明数据科学领域目前还是 **从事传统机器学习** 工作居多。**逻辑回归** 是 **最受欢迎** 的算法，这也是为什么本课程一开始就以逻辑回归算法为例来建模的主要原因。

经典预测算法（一）
—— 线性回归

核心原理


线性回归
什么是线性回归?
X和y 的相关性
皮尔逊相关系数
最小二乘法
损失函数
梯度下降法求解参数
回归模型效果评估

二、 经典预测算法

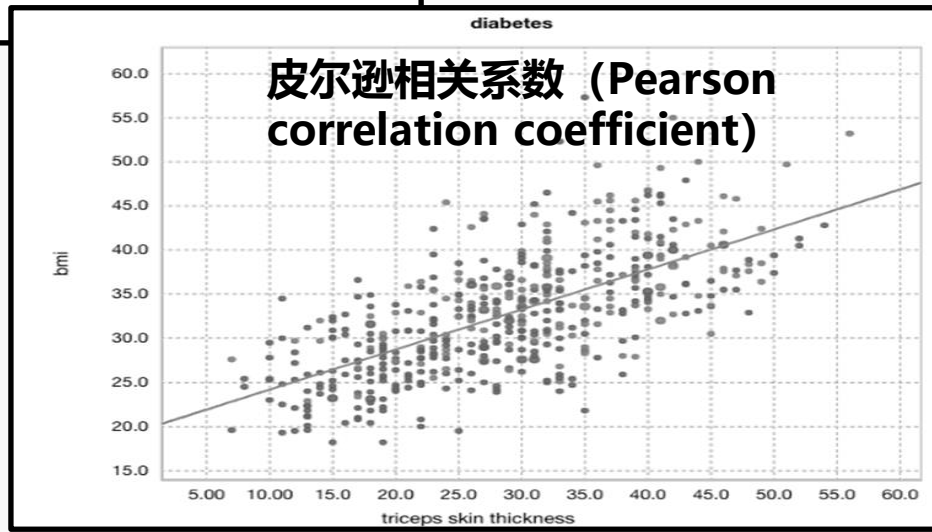
- **2.1.1 线性回归算法核心原理 （1/8）**

什么是线性回归?

回归问题 就是拟合 **输入变量X** 与 **数值型** 的 **目标变量y** 之间的 **关系**，而线性回归就是假定了X与y之间是线性关系，数学表示如下：

$$y=f(x1,x2,x3...)=w1x1+w2x2+w3x3+...+b$$

如下图所示，我们可以通过绘制（X,y）散点图的方式来查看X和y之间是否有线性关系，**回归模型的目标是寻找到一条穿过这些散点的直线，让所有的点离这条直线的距离最近。**这条完美直线对应的参数就是我们要寻找的 **回归模型参数 w1,w2,w3...b** 。



二、 经典预测算法

- **2.1.1 线性回归算法核心原理（2/8）**

X和y 的相关性

回归建模之前，往往先要研究 X和y 的相关性，**X和y具有相关性是进行回归建模的前提**，相关系数是反映相关性强弱的量化指标。相关性包含 **线性相关** 和 **非线性相关** ，而且X和y变量的类型不同，相关系数的计算方法也不同：

- **X和y都是数值型变量** ： **皮尔逊相关系数** 是衡量它们之间线性相关性的最佳选择
- **X和y都是类别型变量：** 由于y是类别型变量，这个时候就是 **分类问题** 了，用 **卡方值（chi-square）** 来衡量相关性
- **X和y中一个数值变量，一个类别变量**：可以考虑用 **Spearman相关系数**

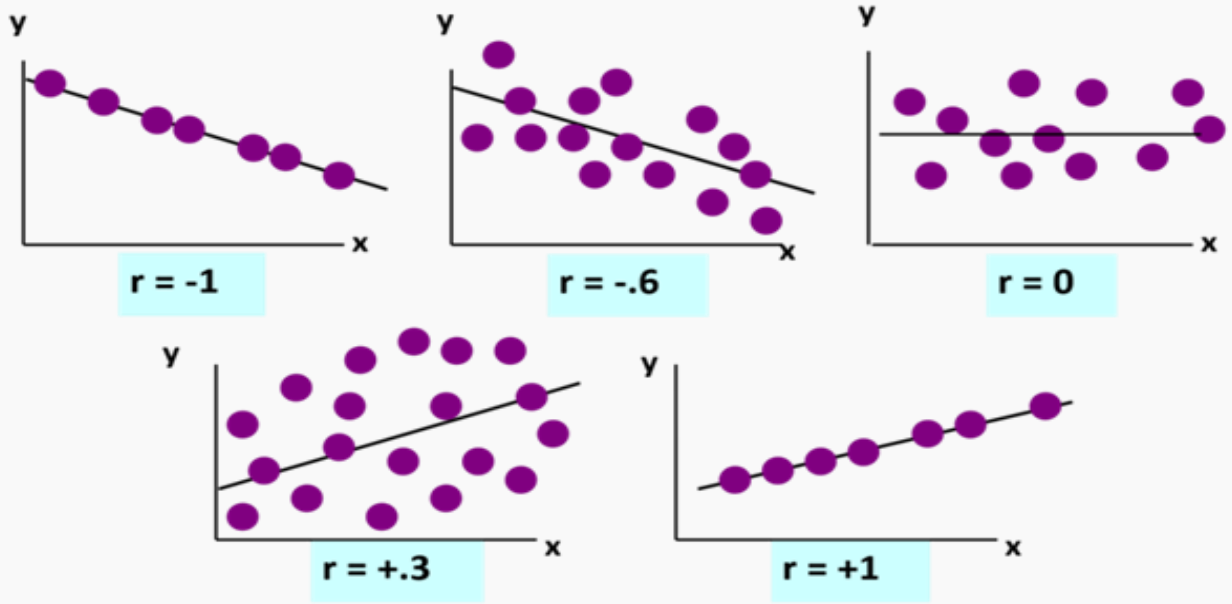
二、 经典预测算法

- **2.1.1 线性回归算法核心原理（3/8）**

在**线性回归** 问题中，我们最常用的是 **皮尔逊相关系数**，下图给出了皮尔逊相关系数的计算方法，以及不同的相关系数值对应的相关关系示意。皮尔逊相关系数的值在**-1~1**之间，**正数表示正相关**，**负数表示负相关**，**1和-1表示绝对相关**，就是说X和y完全落在一条直线上，这样的情况在机器学习中往往没有意义。

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

皮尔逊相关系数



二、 经典预测算法

- **2.1.1 线性回归算法核心原理 (4/8)**

最小二乘法

最小二乘法 是一种求解 **回归模型参数** w1,w2,w3...b 的方法。前面已经介绍了，**回归就是为了寻找一条完美直线** ，如何用数学的方式来精确定义这样的一条完美直线呢 ？ 200 多年前的数学家们发明了最小二乘法来解决这个问题。能让**预测值**与**真实值**的**误差平方和**最小的这条直线就是**完美直线**，用数学符号表示就是：

$$min \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

y^i 表示第 i 个数据点的预测值，也就是对应完美直线上的 y 值。最小二乘法通过**求偏导数**的方法找到让误差平方和取得最小值的 **w1,w2,w3...b** 。今天我们通过 **sklearn** 中**线性回归算法的 fit()**方法可以很快的求得模型参数，不需要我们再手动去计算了。

二、 经典预测算法

- **2.1.1 线性回归算法核心原理 (5/8)**

损失函数

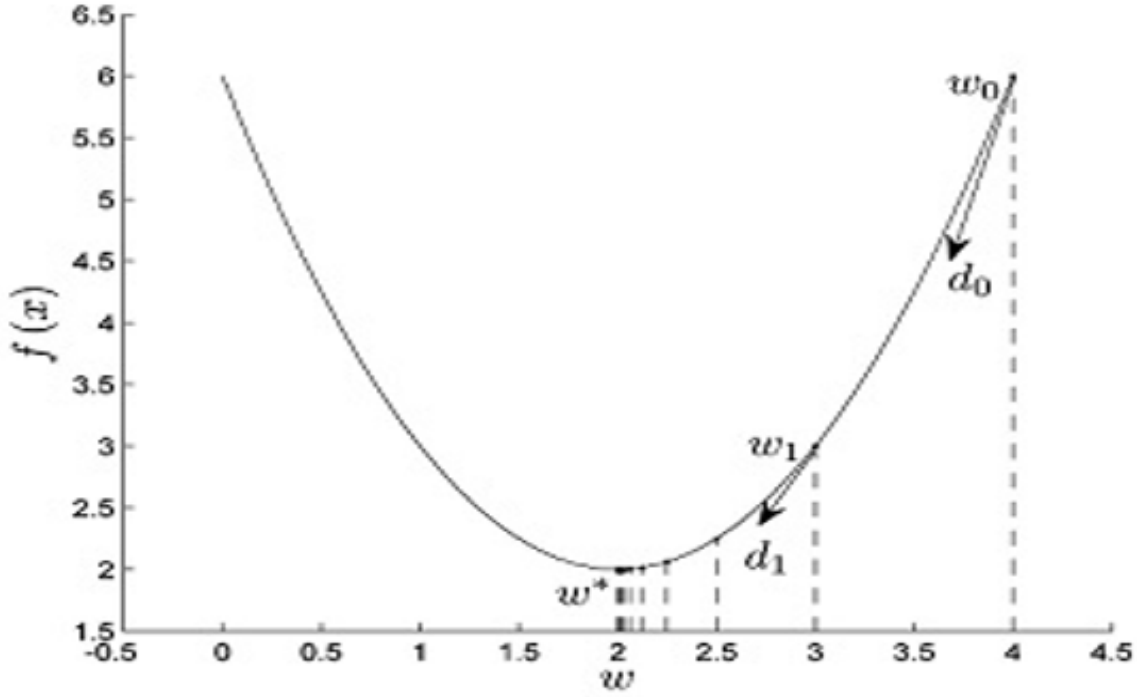
这里我们再引入一个术语叫 **损失函数**。我们知道，**监督学习算法的目标就是为了让目标变量 y 的预测值与真实值尽量的吻合**，那么两者之间的**差异**如何定义呢？**定义预测值与真实值之间的差异的方法就叫损失函数。损失函数值越小**，说明差异越小，模型的 **预测效果也就越好**。所以最小二乘法定义的误差平方和 $\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$ 就是损失函数。针对于 **分类模型** ，损失函数的定义就 **不是最小二乘法** 这样的形式了，但从 **本质上** 来说，无论 **损失函数** 如何定义，它都是在 **反映预测值和真实值之间的差异大小**。

二、 经典预测算法

- **2.1.1 线性回归算法核心原理 (6/8)**

梯度下降法求解参数（1/2）

最小二乘法 是 **一种** 求解参数的方法，**另外一种** 在机器学习领域 **更加通用** 的方法是 **梯度下降法。**它的**核心思想**是 **通过迭代逼近的方法寻找到让损失函数取得最小的参数** w1,w2,w3...b 。如下图所示，梯度下降法通过逐步迭代的方式寻找到让损失函数最小的参数 w 值。

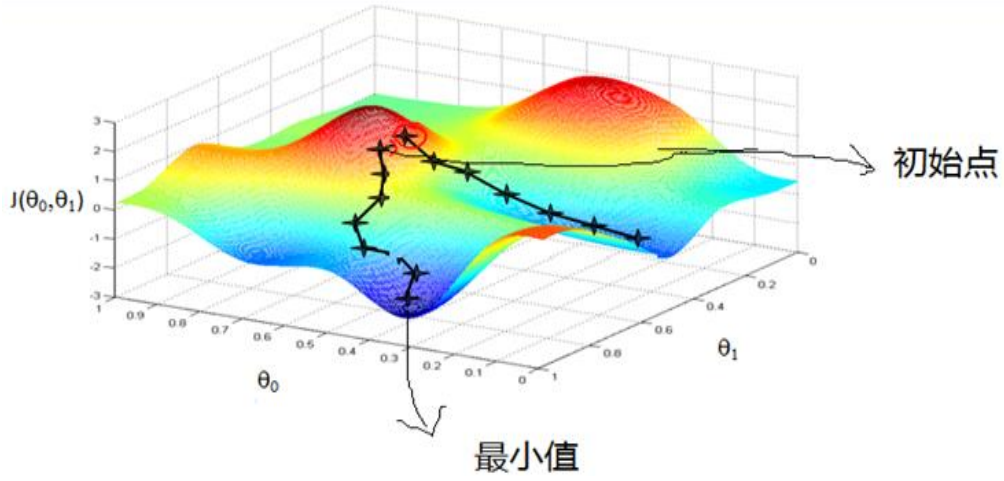


二、 经典预测算法

- **2.1.1 线性回归算法核心原理 （7/8）**

梯度下降法求解参数（2/2）

梯度下降法的 **基本思想** 可以类比为一个 **下山的过程** 。假设这样一个场景：一个人被困在山上，需要从 **山上** 下来(i.e. 找到山的最低点，也就是 **山谷**)。但此时山上的浓雾很大，导致可视度很低。因此，下山的路径就无法确定，他必须利用自己周围的信息去找到下山的路径。这个时候，他就可以 **利用梯度下降算法** 来帮助自己下山。具体来说就是，**以他当前的所处的位置为基准，寻找这个位置最陡峭的地方，然后朝着山的高度下降的地方走**，同理，如果我们的目标是上山，也就是爬到山顶，那么此时应该是朝着最陡峭的方向往上走。然后 **每走一段距离**，都 **反复采用同一个方法**，**最后** 就能 **成功** 的 **抵达山谷**。



从上面的描述可以看出，梯度下降 **不一定** 能够找到 **全局的最优解**，有可能只是一个 **局部最优解**。Sklearn 中有基于 **梯度下降优化算法的预测模型**，例如 **linear_model.SGDClassifier**，你不需要自己去实现它。

二、 经典预测算法

- 2.1.1 线性回归算法核心原理 (7/8)

梯度下降法求解参数（2/2）

```
 8 x = 2
 9 lr = 0.01
10
11 epochs = 1000
12
13 #fun()函数的定义
14 def fun(x):
15     return x ** 2 + 2 * x + 1
16
17 #fun()函数的导数
18 def dfun(x):
19     return x * 2 + 2
20
21 for epoch in range(epochs):
22     x = x - lr * dfun(x)
23
24 print('x =', x)
25 print('y =', fun(x))
```

二、 经典预测算法

- **2.1.1 线性回归算法核心原理 (8/8)**

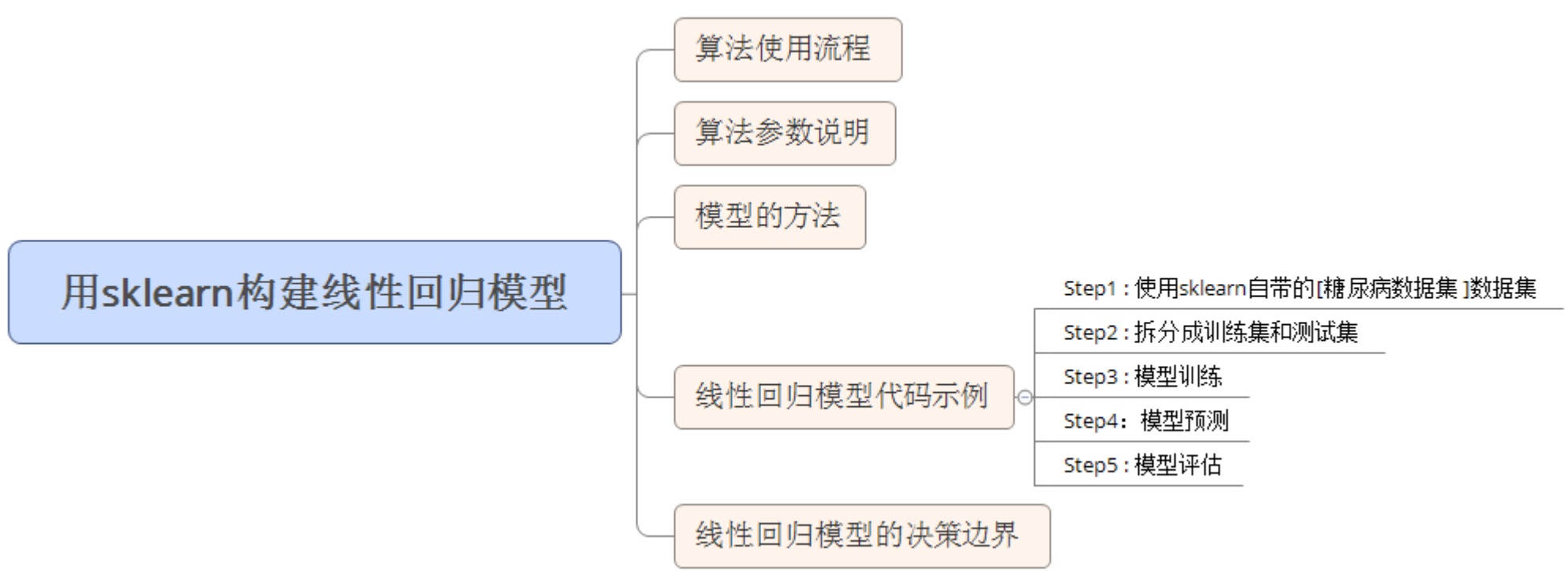
回归模型效果评估

- 损失函数如何定义，理论上模型就如何评估，因为模型拟合的目标就是为了让损失函数最小，所以 **模型评估本质上就是看损失函数的值**，我们经常把损失函数值称为 **loss**，哪一个模型的 **loss 最小**，**这个模型就是最好** 的。
- 当然，有时候我们也不是只看loss，因为如果**缺乏参照**，我们也不知道loss多小算好。这个时候往往我们需要一些 **类似皮尔逊相关系数** 这样的更加直观的 **评估指标。**
- **回归模型评估** 最常用的一个指标叫 **决定系数 r2_score。** 关于预测模型的评估，我们后面会专门的讲解，这里暂不展开。

线性回归 是我们 **第一个** 讲解的 **机器学习预测算法**，也是第一次提到了**监督学习** 中非常重要的两个概念 **损失函数** 和 **梯度下降**。这里只是做了非常简短的描述，如果大家对它们有兴趣，可以再自行深入学习。简单对线性回归做一个总结：

- **线性回归** 研究的是 **x和y** 之间的 **线性关系**
- **线性回归** 和 **相关分析** 有很强的关系，我们往往先通过相关分析发现 x和y 有相关性之后再进行回归
- 线性回归模型使用 **最小二乘法** 来估计参数值，也可以通过 **损失函数+梯度下降** 的方法来估计参数，后者是机器学习领域中更通用性的一种方法
- 线性回归模型的评估指标和分类模型不太一样，**分类模型** 常用 **准确率**，而回归模型常用 **R2 （r2_score）**

算法实现


用sklearn构建线性回归模型
算法使用流程
算法参数说明
模型的方法
线性回归模型代码示例
Step1：使用sklearn自带的[糖尿病数据集]数据集
Step2：拆分成训练集和测试集
Step3：模型训练
Step4：模型预测
Step5：模型评估
线性回归模型的决策边界

二、 经典预测算法

- **2.1.2 用sklearn构建线性回归模型**

Sklearn 中的算法使用流程

在 **sklearn** 中调用某个算法来构建模型的 **基本套路** 是这样的：

- 用 **算法类 实例化** 一个 **模型对象**，例如 model = LinearRegression(),
- 调用模型对象的 **model.fit(X,y)** 方法，传入数据集X和y来 **训练模型**
- 调用模型对象的一些属性来 **查看** 训练好的 **模型结果**，例如查看回归模型的系数model.coef_
- 调用模型对象的 **model.predict(X)** 方法，对数据集X **进行预测**，得到y的预测值y_pred
- 调用sklearn中的 **评估函数** 进行 **评估**，例如回归模型的R2系数 r2_score(y,y_pred)

以上套路基本上 **对所有的预测类算法都适用**，差别只是换一下算法的类名而已，例如把 **LinearRegression** 换成 **LogisticRegression**，当然不同模型对象的属性 **可能会有一些差别**，例如 **回归模型** 有 **coef_** 这个属性，但是 **决策树算法** 就没有，具体到每一种算法介绍的时候我们再详细讲解。

二、 经典预测算法

- **2.1.2 用sklearn构建线性回归模型**

首先：线性回归算法使用 **sklearn.linear_model** 模块**的 LinearRegression** 类，使用算法之前我们需要先要先导入算法类：
from **sklearn.linear_model** import **LinearRegression**

线性回归算法参数说明

其次：通过LinearRegression类实例化一个模型对象:
model = LinearRegression()。
说明:
1、在实例化对象的时候，可以 **指定** 该算法对应的一些 **参数**
2、这个参数指的是模型训练的一些 **选项和配置**，例如回归模型是否要 **拟合截距**，是否要对 **变量** 进行 **标准化** 等。
3、有时候模型训练完成之后得到的结果，例如 **回归系数值** ，我们有时候也叫它 **模型参数**
4、为了把两者区分开来，**模型训练前的选项和配置我们称为 超参数。**
5、超参数的设置对模型结果的影响比较大，选择合适的超参数是建模的核心工作之一，这个过程我们又称为 **调参**
6、有经验的应用机器学习工程师往往都调得一手好参。

二、 经典预测算法

- **2.1.2 用sklearn构建线性回归模型**

sklearn.linear_model.**LinearRegression**（**fit_intercept**=True, **normalize**=False, **copy_X**=True, **n_jobs**=1）

参数说明

fit_intercept：是否拟合截距项，默认是True
normalize：是否对X变量进行标准化处理，标准化处理对回归模型来说不是必须的，默认是False
copy_X：主要是针对normalize后的X变量是否要覆盖始的变量，默认是True

二、 经典预测算法

- **2.1.2 用sklearn构建线性回归模型**

线性回归模型的结果

模型训练好之后，可以查看线性回归模型的结果，主要是如下两个属性：

coef_：回归系数，即回归表达式中每一个X变量对应的w值
intercept_：回归表达式中的截距项b值

二、 经典预测算法

- **2.1.2 用sklearn构建线性回归模型**

线性回归模型的方法

方法指的是对模型对象可以进行的一系列操作，例如训练模型使用 fit() 方法，预测使用predict()方法。线性回归模型对象常用的方法有：

- **fit(X,y)**：输入数据集X和y对模型进行训练
- **predict(X)**：输入数据集X，返回模型对y的预测值
- **score(X,y)**：输入数据集X和y，返回模型准确性评估的决定系数R^2值

二、 经典预测算法

- **2.1.2 用sklearn构建线性回归模型**

线性回归代码实现(1/7)

Step1:使用 sklearn 自带的糖尿病数据集

diabetes 是一个关于糖尿病的数据集，该数据集包括 **442** 个病人的生理数据及一年以后的病情发展情况。数据集中 **X变量** 有 **10** 个，都经过了 **标准化** 处理：

- **age:** 年龄
- **sex:** 性别
- **bmi:** 体质指数
- **bp:** 血压
- **s1,s2,s3,s4,s4,s6** (六种血清的化验数据)
- 目标变量 **target** 表示的是一年以后的糖尿病病情情况，是一个 **介于25到346之间** 的连续数值

>> 更多扫码

二、 经典预测算法

- **2.1.2 用sklearn构建线性回归模型**

线性回归代码实现(2/7)

Step1:使用 sklearn 自带的糖尿病数据集

```
from sklearn import datasets
diabetes = datasets.load_diabetes()
# 查看data中的内容
dir(diabetes)  #['DESCR', 'data', 'feature_names', 'target']
# 查看自变量名称，共10个
diabetes.feature_names  #['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5', 's6']
# 查看数据集X变量
diabetes.data
# 查看目标变量y前100行
diabetes.target[:100]
```

```
array([[ 0.03807591,  0.05068012,  0.06169621, ..., -0.00259226,
         0.01990842, -0.01764613],
       [-0.00188202, -0.04464164, -0.05147406, ..., -0.03949338,
        -0.06832974, -0.09220405],
       [ 0.08529891,  0.05068012,  0.04445121, ..., -0.00259226,
         0.00286377, -0.02593034],
       ...,
       [ 0.04170844,  0.05068012, -0.01590626, ..., -0.01107952,
        -0.04687948,  0.01549073],
       [-0.04547248, -0.04464164,  0.03906215, ...,  0.02655962,
         0.04452837, -0.02593034],
       [-0.04547248, -0.04464164, -0.0730303 , ..., -0.03949338,
        -0.00421986,  0.00306441]])
```

二、 经典预测算法

- **2.1.2 用sklearn构建线性回归模型**

线性回归代码实现(3/7)

Step2: 拆分成训练集和测试集

```
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test=train_test_split(diabetes.data,diabetes.target,test_size=0.3,random_state=123456)
```

二、 经典预测算法

- **2.1.2 用sklearn构建线性回归模型**

线性回归代码实现(4/7)

Step3: 训练模型

```
from sklearn.linear_model import LinearRegression
# 实例化模型并训练
model = LinearRegression()
model.fit(X_train,y_train)
# 查看模型结果
print(model.coef_)
print(model.intercept_)
```

```
[   0.88344869 -228.95940929  586.51882755  291.48268133 -720.87460703
  395.52012225  226.25389411  374.92192841  650.93489885   12.95543774]
150.232354716
```

二、 经典预测算法

- **2.1.2 用sklearn构建线性回归模型**

线性回归代码实现(5/7)

Step4: 模型评估

```
# 方法1：直接使用模型的score方法计算r2值
print(model.score(X_test,y_test))
# 方法2：使用sklearn.metrics下的r2_score函数
# 先对测试集进行预测
y_pred = model.predict(X_test)
# 使用r2_score计算r2值
from sklearn.metrics import r2_score
print(r2_score(y_test,y_pred))
```

对这个结果我们暂不做评价，后继会有专题来讲解这块内容！

```
0.516704402798
0.516704402798
```

- **2.1.2 回归模型评估指标**

MSE

$$\frac{1}{m}\sum_{i=1}^{m}\left(y_{\text{test}}^{(i)}-\hat{y}_{\text{test}}^{(i)}\right)^2=MSE$$

RMSE

$$\sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(y_{\text{test}}^{(i)}-\hat{y}_{\text{test}}^{(i)}\right)^2}=\sqrt{MSE_{\text{test}}}=RMSE_{\text{test}}$$

MAE

$$\sum_{i=1}^{m}\left|y_{iest}^{(i)}-\hat{y}_{tst}^{(i)}\right|$$

R2_score

$$R^2=1-\frac{\sum_i\left(\hat{y}^{(i)}-y^{(i)}\right)^2}{\sum_i\left(\overline{y}-y^{(i)}\right)^2}$$

二、 经典预测算法

- **2.1.2 用sklearn构建线性回归模型**

线性回归代码实现(6/7)

PS：完整代码见PPT备注

 # 通过图表做探索式分析

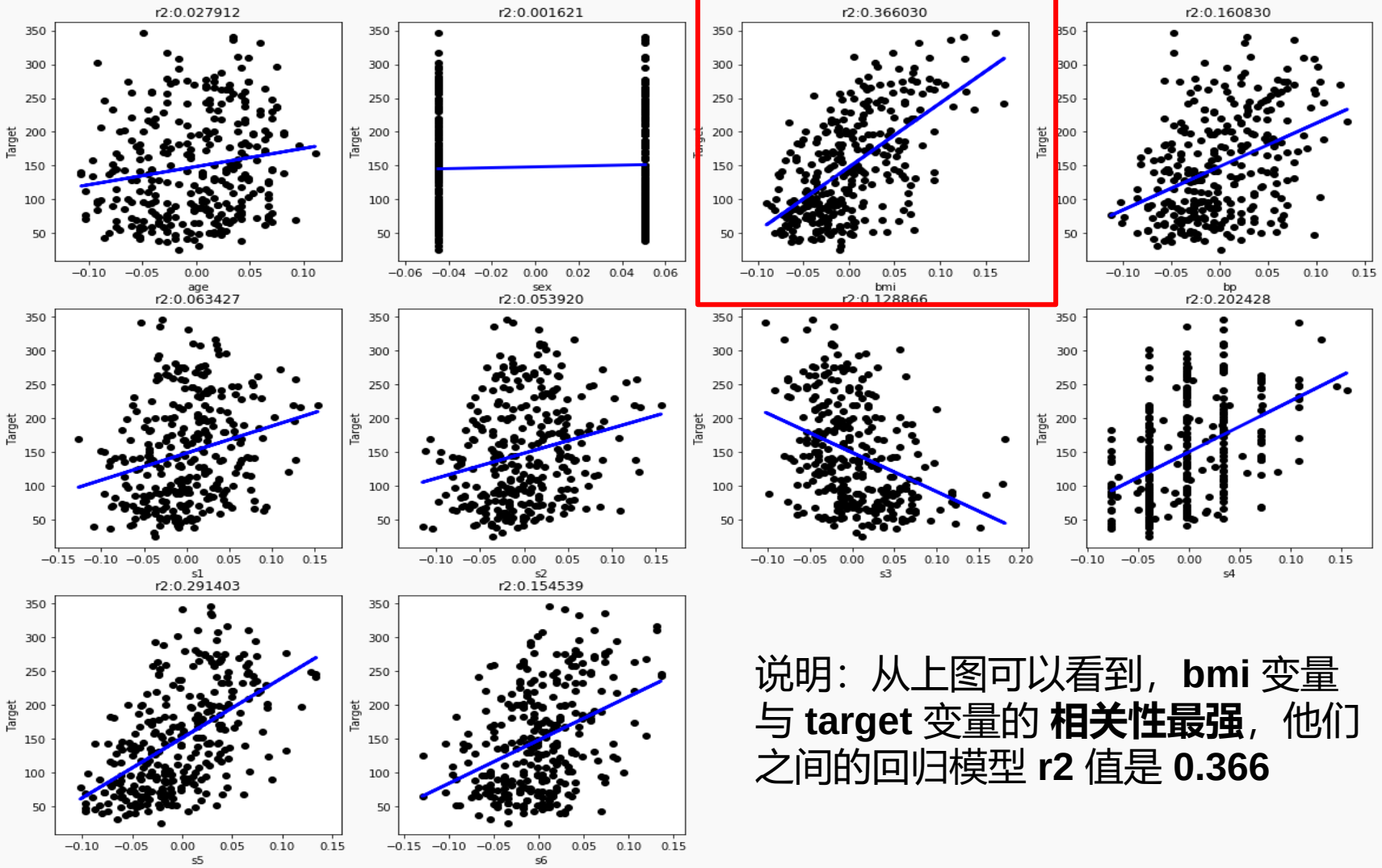
总共有10个X变量，每次取1个X变量来与y绘制(x,y)散点图，并画出x和y之间的线性回归线。一般来说，这是在构建模型之前做的变量探索性分析，通过散点图可以查看每一个X变量与y变量的相关性，可以根据相关性大小进行变量筛选。下面我们针对拆分后的训练数据集来进行分析：

>>获取完整代码

二、 经典预测算法

- **2.1.2 用sklearn构建线性回归模型**
  - 探索性分析

线性回归代码实现（7/7）



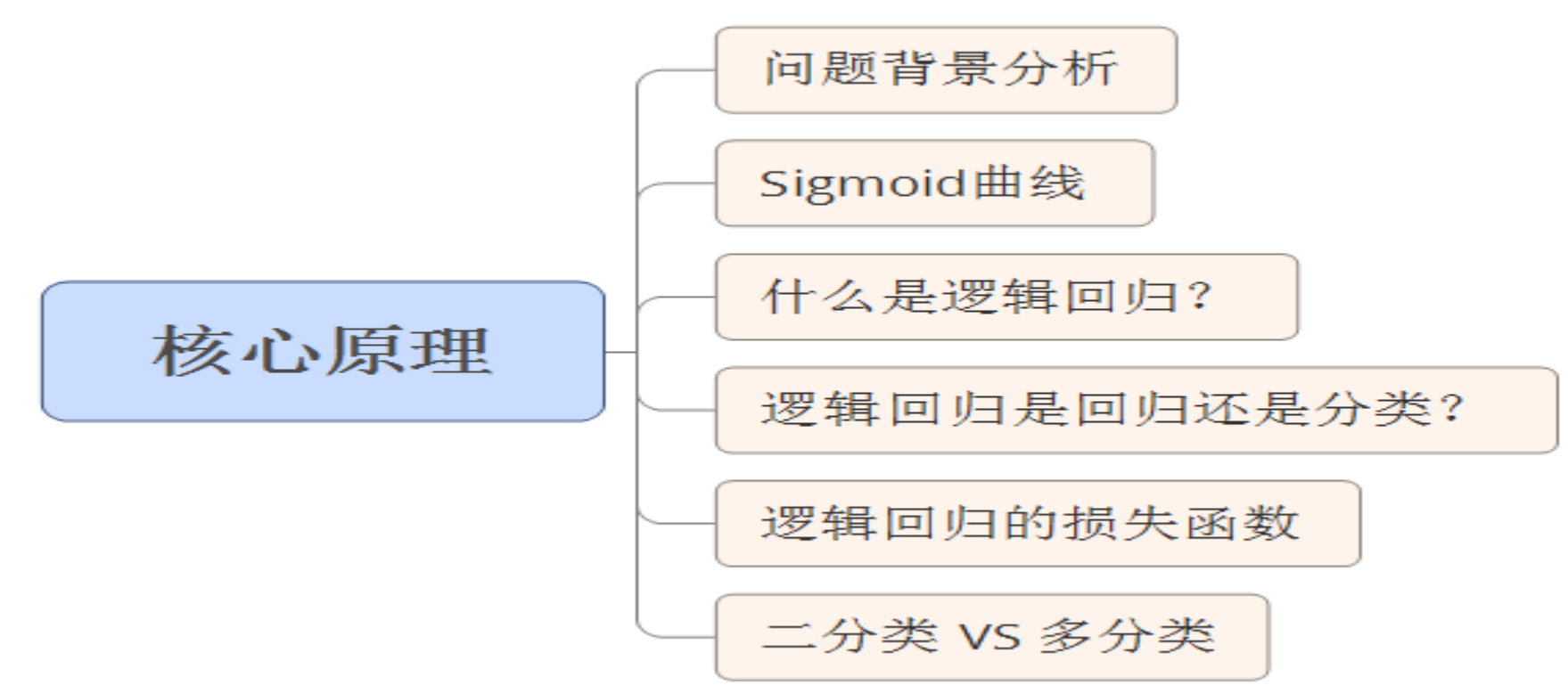
说明：从上图可以看到，**bmi** 变量与 **target** 变量的 **相关性最强**，他们之间的回归模型 **r2** 值是 **0.366**

使用skleanr.linear_model的LinearRegression类来构建线性回归模型

- 线性回归模型构建的基本套路（其他模型也基本类似）：
- 先实例化模型对象：model= LinearRegression()
- 训练模型：model.fit(X_train,y_train)
- 对测试集预测：model.predict(X_test)
- 评估模型准确度：model.score(X_test,y_test)
- 构建回归模型之前一般先要进行单变量与目标变量的相关性分析，通过散点图和单变量回归的拟合直线

经典预测算法（二）
—— 逻辑回归

核心原理

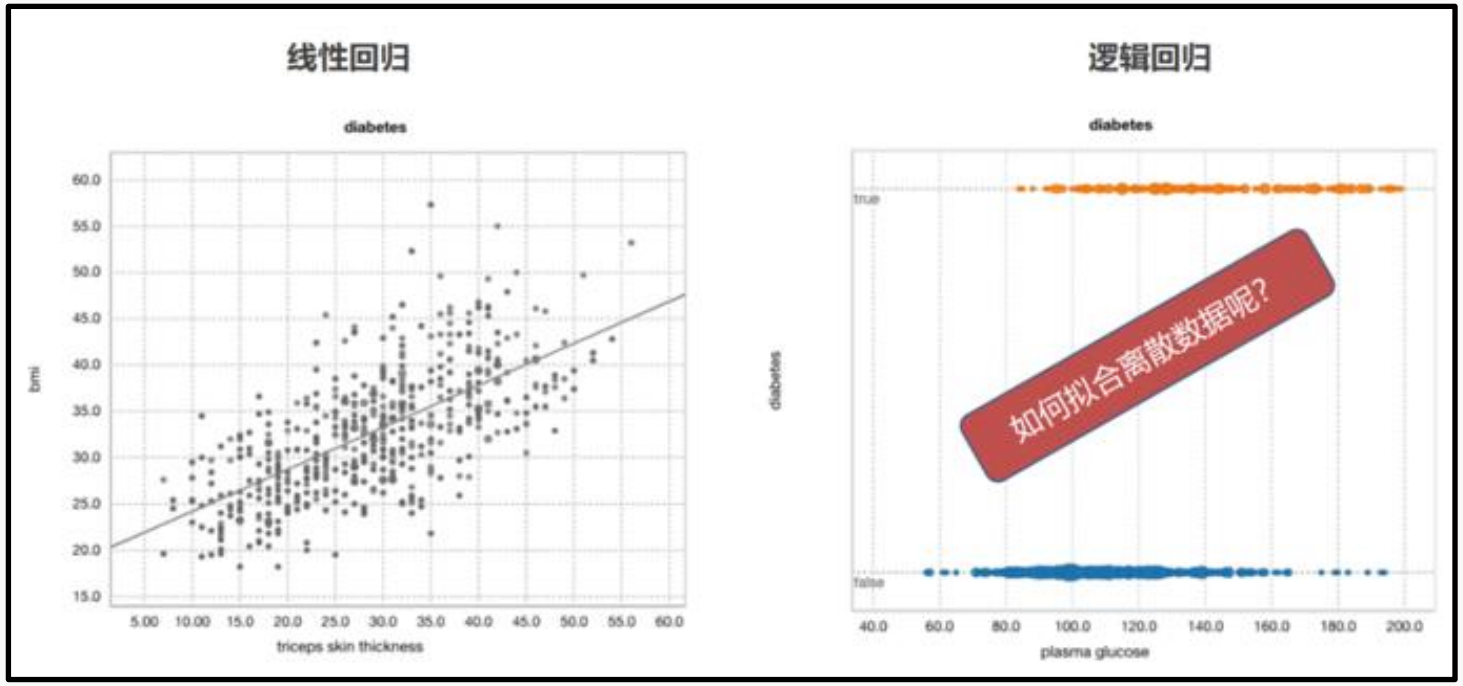


二、 经典预测算法

- **2.2.1 逻辑回归算法核心原理(1/11)**

我们知道了 **线性回归模型** 可以 **拟合X** 与 **y** 之间的函数关系，但是 **回归模型** 中的 **y 值** 是一个 **连续型的数值** ， 如果把 **y的值** 换成一个**二分类的标签**，例如y只有2个取值：**0**和**1**，**那该如何处理呢**？还能继续使用线性回归吗？这个时候 **逻辑回归** 就要 **登场** 了。

问题背景（1/2）

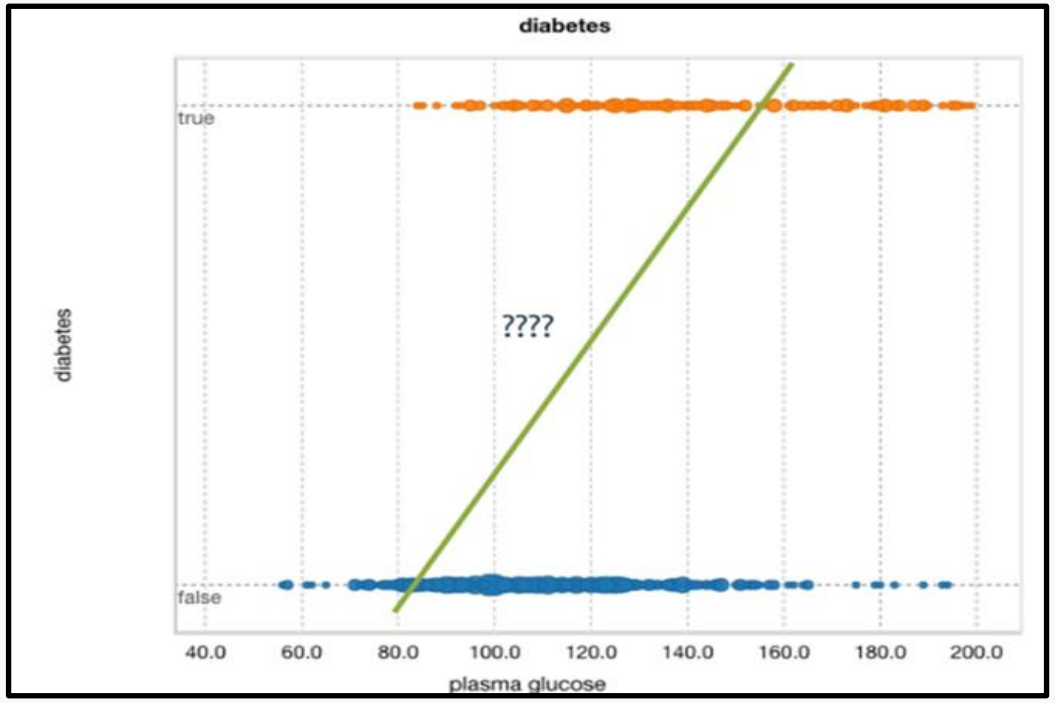


二、 经典预测算法

- **2.2.1 逻辑回归算法核心原理(2/11)**

问题背景（2/2）

很显然，针对y的取值只有0和1这样的情况，我们不能像线性回归一样，寻找一条完美直线穿过所有的数据点（如下图所示），因为y的值域在[0-1]之间，这条直线很容易就穿破了下限0和上限1，这样得到的预测结果肯定是不靠谱的。



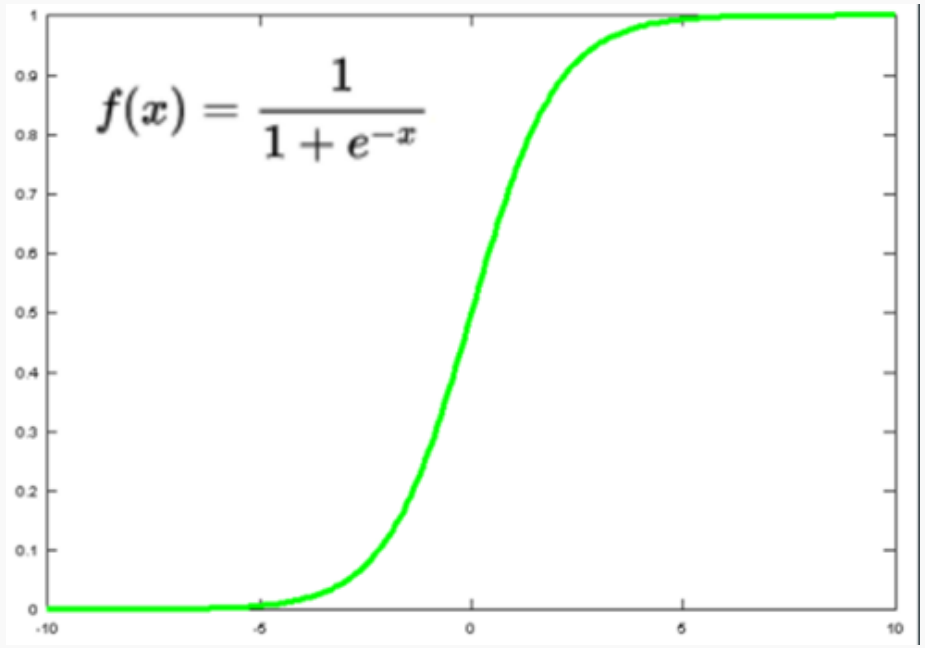
二、 经典预测算法

- **2.2.1 逻辑回归算法核心原理(3/11)**

很明显，针对y的值域在[0-1]的问题，我们不能寻找一条完美直线，但是换个思路，我们可以寻找一条完美的 **S型曲线** 。S型曲线全称叫 **Sigmoid** 曲线，它的函数表达式是这样的：

$$f(x)=\frac{1}{1+e^{-x}}$$

Sigmoid曲线



说明：我们可以看到S曲线具有如下特性：

当x->+∞ 时，f(x)趋向于1

当x->-∞ 时，f(x)趋向于0

当x=0 时，f(x)=0.5

二、 经典预测算法

- **2.2.1 逻辑回归算法核心原理(4/11)**

什么是逻辑回归?(1/2)

我们以 **二分类问题** 为例，**逻辑回归** 就是假设 y 取值为 1 的概率 **p(y=1)与X之间是S曲线关系**（这个是容易理解的，p(y=1)和S曲线的值域是吻合的，都在[0-1]之间），用数学符号表示就是：

$$p(y=1)=\frac{1}{1+e^{-(wx+b)}}$$

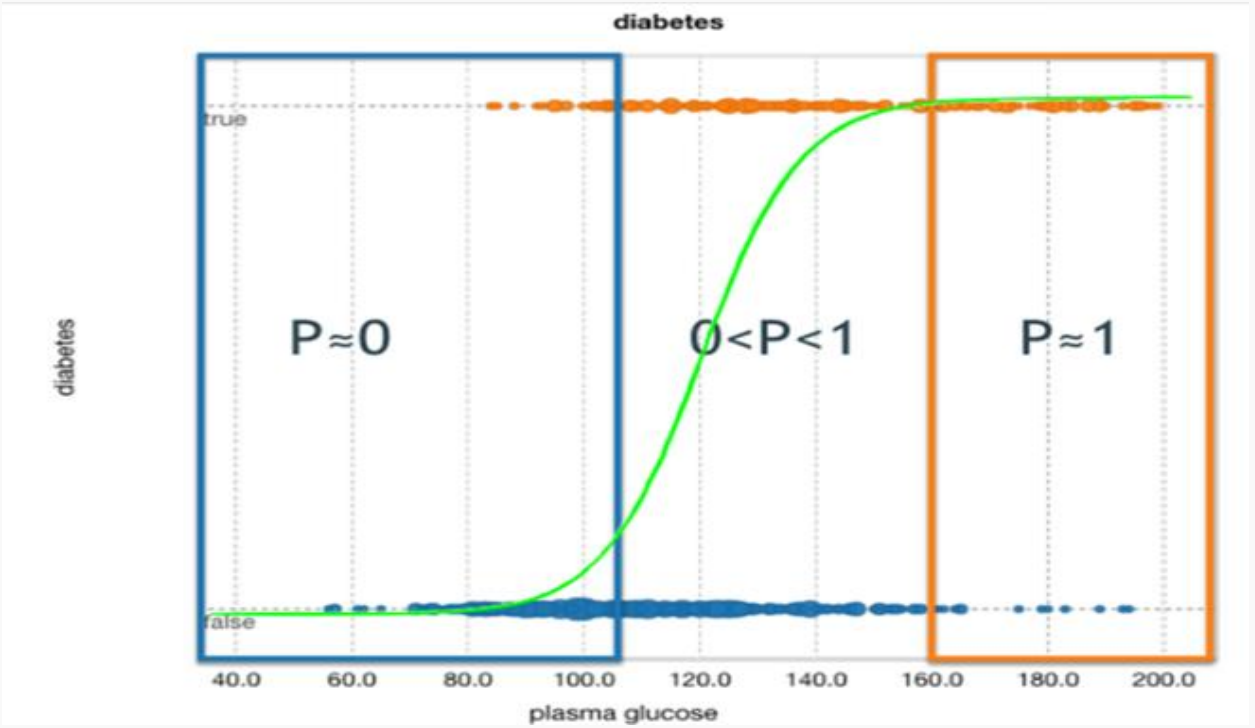
w，b是逻辑回归模型的参数。为什么是p(y=1)，而不是p(y=0)？其实这个没有本质区别，因为 **p(y=1)+p(y=0)=1** ，它们是可以互相转换的，所以拟合 p(y=1) 和 p(y=0) 的效果是一样的。我们前面也介绍了，一般来说，二分类问题我们习惯把少的那一个类赋值为1，也叫 **正例**，它是我们比较关注的类，所以逻辑回归模型也习惯拟合 p(y=1)。

二、 经典预测算法

- **2.2.1 逻辑回归算法核心原理(5/11)**

从下图中我们可以看到：S曲线右侧就表示 **y=1 的概率高**，我们就会把y的 **类别值预测为1**；S曲线左边表示 **y=1的概率低**，反过来就是 **y=0的概率高**，那么我们就会把 **y的类别值预测为0** 。

什么是逻辑回归?
(2/2)



二、 经典预测算法

- **2.2.1 逻辑回归算法核心原理(6/11)**

不要只看看名字，我们要看问题的本质，逻辑回归模型的 **y取值是类别值**，所以它解决的问题一定是**分类问题。**之所以这个算法名词包含了“回归”二字，那是因为它是**广义线性回归模型**的**变形**：

逻辑回归是回归还是分类?

$$p = \frac{1}{1 + e^{-(wx+b)}}$$

等价于

$$ln(\frac{p}{1-p}) = wx + b$$

- 如果我们把 ln(p/1−p) 当做 y 看待，那么 y=wx+b 就是**线性回归**了。
- 我们习惯把 p/1−p 称为 odds，中文可以叫 **几率**，就是代表类别发生和不发生的概率比值
- 这样逻辑回归又可以表达为： ln(odds)=wx+b
- 所以：**逻辑回归算法**属于**分类算法**而不是回归算法，但是它的**算法形式**与**线性回归很相似**。

二、 经典预测算法

- **2.2.1 逻辑回归算法核心原理(7/11)**

同线性回归模型一样，**逻辑回归模型**也是要求解参数 **w,b** 的值，同样的原理，逻辑回归模型也需要定义一个 **损失函数**，然后找到 **让损失函数最小的w,b 值**

逻辑回归的损失函数(1/3)

- 在目标变量 **y是分类值** 的情况下，我们经常使用一种叫 **最大似然** 的方法来定义 **损失函数**，**简单来说就是每一条数据所属类别的预测概率值的乘积就叫最大似然，如果每条数据被预测的概率值乘积越大，那么说明我们模型拟合的效果越好。**
- **max(p1*p2*p3...*pn)**，n代表的是记录数

二、 经典预测算法

- **2.2.1 逻辑回归算法核心原理(8/11)**

逻辑回归的损失函数(2/3)

由于数学中乘法不太好求解，取个 **对数就变成了加法**，另外，我们一般习惯定义**损失函数**，而不是 **目标函数**，他们的差别是 **损失函数loss值越小越好**，而上面定义的是 **目标函数**，它的 **值是越大越好**。这个很好办，我们在 **目标函数前面加个负号就变成了损失函数**了。所以逻辑回归模型的常用的损失函数就是如下定义的**对数似然损失函数**：

$$loss=-(\ln(p1)+\ln(p2)+\ln(p3)+\ldots+\ln(pn))$$

- 回顾一下**线性回归模型**的**损失函数**用的是什么方法？

---**最小二乘法**（误差平方和）

二、 经典预测算法

- **2.2.1 逻辑回归算法核心原理(9/11)**

逻辑回归的损失函数(3/3)

obs	x	y	p(y=1)	p(y=0)	p(y)	-ln(p)
1	72.74	1	0.95	0.05	0.95	0.05129
2	38.51	0	0.23	0.77	0.77	0.26136
3	9.00	1	0.67	0.33	0.67	0.40048
4	36.00	1	0.85	0.15	0.85	0.16252
5	33.38	0	0.23	0.77	0.77	0.26136
6	9.31	0	0.34	0.66	0.66	0.41552
7	2.80	0	0.13	0.87	0.87	0.13926
8	50.09	1	0.87	0.13	0.87	0.13926
9	34.18	0	0.45	0.55	0.55	0.59784
10	53.85	0	0.32	0.68	0.68	0.38566
11	65.06	1	0.69	0.31	0.69	0.37106
12	75.32	1	0.78	0.22	0.78	0.24846
13	10.73	0	0.32	0.68	0.68	0.38566
14	53.45	0	0.26	0.74	0.74	0.30111
					avg	0.29435

- 如左图所示，我们用一个表格来示例可能会更容易理解一些。表格的 **最后一列** 就表示每一条数据的 **对数似然损失值**，把所有数据的对数似然损失值 **求平均** 就是最终的 **总损失值**。注意：每一条数据有两个概率p(y=1)和p(y=0)，我们在计算每一条数据的对数似然损失值的时候不是固定取p(y=1)，而是该条数据被预测为哪一个类就使用该类的概率，你也可以简单的理解是p(y=1)和p(y=0)中的最大值。

二、 经典预测算法

- **2.2.1 逻辑回归算法核心原理(10/11)**

二分类 VS 多分类 (1/2)

多分类就是指 **目标变量y** 的取值有**多个类别标签**，以下这些问题都是多分类问题：

- **文本分类**： 文章所属的类别较多，例如财经类，体育类，娱乐类，健康类等
- **物体识别**： 物体的种类非常多，例如ImageNet大赛的物体识别有1000个类
- **手写数字识别**： 包含0-9共计10个数字的识别

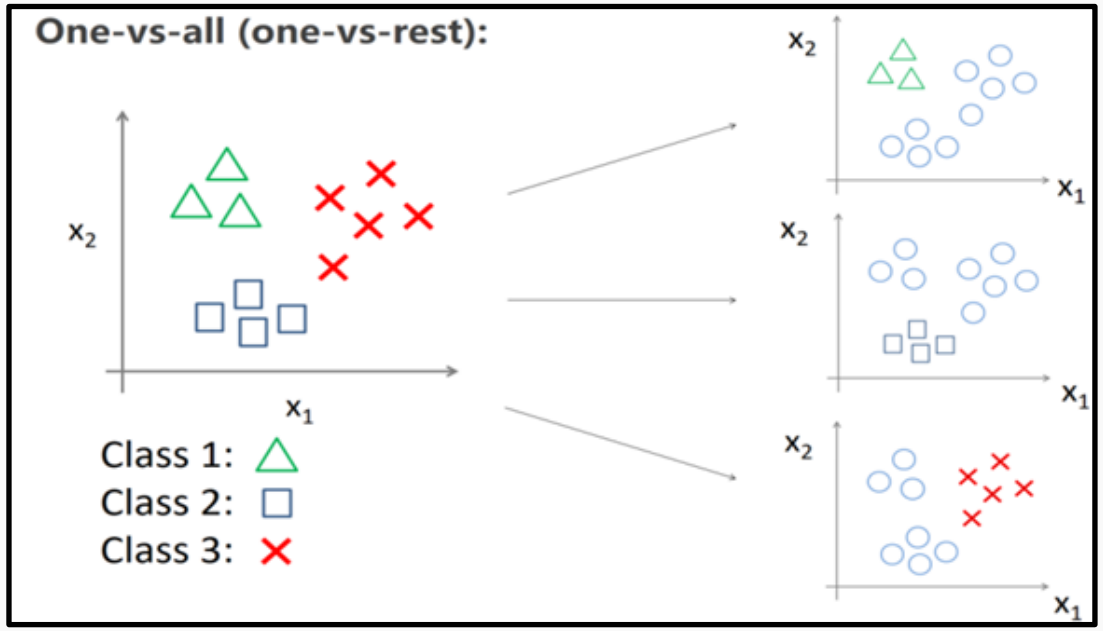
有一些算法天然可以解决多分类问题，例如**KNN**，**决策树。**对于**逻辑回归算法**来说，我们从一开始介绍它的时候就只针对**二分类**问题，的确，传统的逻辑回归只能解决二分类问题。但是也不用担心，只需要简单的调整，**逻辑回归算法就能解决多分类问题。**

二、 经典预测算法

- **2.2.1 逻辑回归算法核心原理(11/11)**

二分类 VS 多分类 (2/2)

解决方案就是：把 **多分类问题拆分成若干个二分类问题** 就好了，若干个是多少个？**多分类的类别数有几个那就是几个。**
如下图所示，多分类问题拆成之后，每一个二分类模型轮流挑选一个类别作为正例1，剩余的作为负例0。这样有多少个类别，就有多少个二分类模型，这种方法叫 **one vs rest** 或者 **one vs all** 。



sklearn很强大，当你在调用逻辑回归算法的时候，它已经默认设置了multi_class=’ovr’这个超参数，你什么也不需要做，对，你只需要知道就行了。所以sklearn中的逻辑回归算法默认是可以处理多分类问题。

逻辑回归 虽然名字中包含“回归”二字，但是它 **解决** 的是 **分类问题** 而不是回归问题。**逻辑回归是最经典和最常用的一个分类算法**，简单回顾一下：

- 因为逻辑回归解决的是二分类问题，所以 **逻辑回归拟合的是一条S型曲线**，而不是一条直线，S型曲线的值表示的是 **y=1的预测概率**
- 逻辑回归模型的 **损失函数** 是 **对数似然损失函数**，而不是误差平方和
- 标准的逻辑回归模型 **解决的是二分类** 问题，但是 **可以把多分类问题拆分为多个二分类** 问题，**sklearn** 的逻辑回归算法 **支持多分类** 问题

二、 经典预测算法

- **2.2.2 用sklearn构建逻辑回归模型**

算法使用流程

在sklearn中，选择不同的算法构建预测模型的流程是类似的，主要是算法的类名和模型的结果属性有一些差别，逻辑回归建模的基本流程：

- 实例化一个逻辑回归模型 **model = LogisticRegression()**
- 调用模型的 **model.fit(X,y)** 方法，传入数据集X和y来训练模型
- 查看模型的一些属性，例如 **model.coef_**查看模型的回归系数
- 调用模型的 **model.predict(X)** 方法，返回预测的标签值，还可以调用 **model.predict_proba(X)** 方法，返回预测的概率
- 调用模型的 **model.score(X,y)** 方法，返回模型的正确率

二、 经典预测算法

- **2.2.2 用sklearn构建逻辑回归模型**

逻辑回归算法使用sklearn.linear_model模块的LogisticRegression类，使用之前需要先导入算法类：

from sklearn.linear_model import LogisticRegression

逻辑回归算法参数说明

通过model = LogisticRegression()来实例化一个逻辑回归模型，可以指定如下的参数：

```
sklearn.linear_model.LogisticRegression ( penalty='l2', dual=False, tol=0.0001, C=1.0,
fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver=
'liblinear', max_iter=100, multi_class='ovr', verbose=0, warm_start=False, n_jobs=1)
```

二、 经典预测算法

- **2.2.2 用sklearn构建逻辑回归模型**

逻辑回归算法参数说明

>> 了解更多

```
sklearn.linear_model.LogisticRegression ( penalty='l2', dual=False, tol=0.0001, C=1.0,
fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver=
'liblinear', max_iter=100, multi_class='ovr', verbose=0, warm_start=False, n_jobs=1)
```

参数说明：

penalty ：回归模型训练过程中加入惩罚项是一种防止模型过拟合的高级技巧，之前没有介绍过，我们会安排在模型评估那一章来讲解。大家知道这个参数有两个取值就好了：l1 和 l2。一般又叫 l1 正则化和 l2 正则化

C：和 penalty 配合的一个参数，是正则化强度（惩罚力度）的倒数，C 越小意味着更强的正则化（更强的惩罚），默认是 1.0

fit_intercept：是否拟合截距项 b，默认 True

class_weight：当目标变量 y 的 0 和 1 两个类别样本数量不均衡的时候，可以考虑给不同的类别加权，默认不加权

solver：模型训练的优化算法选择，默认是 liblinear

multi_class：指定多分类的处理方法，默认是 ovr（1 对多的方法）

二、 经典预测算法

- **2.2.2 用sklearn构建逻辑回归模型**

逻辑回归模型的结果

逻辑回归模型的结果和线性回归一样（毕竟大家都是属于回归家族的算法），主要就是如下两个属性：

- coef_：回归系数，即回归表达式中每一个X变量对应的w值
- intercept_：回归表达式中的截距项b值

二、 经典预测算法

- **2.2.2 用sklearn构建逻辑回归模型**

逻辑回归模型的方法

逻辑回归模型常用的方法有：

1) fit(X,y)：输入数据集X和y对模型进行训练
2) predict(X)：输入数据集X，返回目标变量y的预测标签
3) predict_proba(X)：输入数据集X，返回目标变量y每一个类别的概率
4) score(X,y)：输入数据集X和y，返回模型预测结果的正确率（accuracy)

二、 经典预测算法

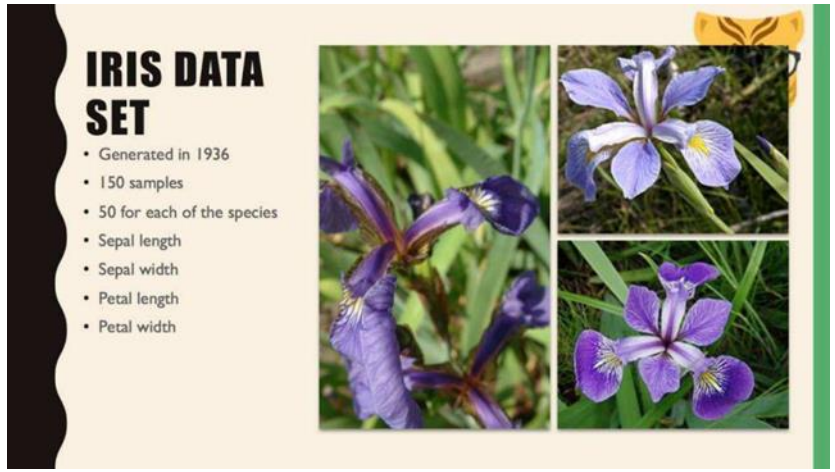
- **2.2.2 用sklearn构建逻辑回归模型**

逻辑回归代码示例

- Step1:使用sklearn自带的iris数据集

iris（鸢尾花数据集）可以说是机器学习中最经典的演示数据集了。通过花萼长度，花萼宽度，花瓣长度，花瓣宽度4个属性来预测鸢尾花属于（Setosa，Versicolour，Virginica）三个种类中的哪一类。它是一个多分类问题。

特征变量说明：
1) sepal length (cm)： 花萼长度
2) sepal width (cm)： 花萼宽度
3) petal length (cm)： 花瓣长度
4) petal width (cm)： 花瓣宽度
目标变量 target 值是 [0,1,2] 代表 [Setosa，Versicolour，Virginica] 三个类



二、 经典预测算法

- **2.2.2 用sklearn构建逻辑回归模型**

逻辑回归代码示例

- Step1:使用sklearn自带的iris数据集

```
from sklearn import datasets
# 导入sklearn自带的iris数据集： iris = datasets.load_iris()
# dir()查看一下iris中有哪些内容：dir(iris)
# ['DESCR', 'data', 'feature_names', 'target', 'target_names']
# 查看一下iris的shape： print(iris.data.shape)
# 查看一下feature_names： print(iris.feature_names)
# 查看iris数据前10行： print(iris.data[:10])
# 查看target前100个值：
print(iris.target[:100])
# 查看目标变量的类别名称：
print(iris.target_names)
```

```
['DESCR', 'data', 'feature_names', 'target', 'target_names']
(150, 4)
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
[[ 5.1  3.5  1.4  0.2]
 [ 4.9  3.   1.4  0.2]
 [ 4.7  3.2  1.3  0.2]
 [ 4.6  3.1  1.5  0.2]
 [ 5.   3.6  1.4  0.2]
 [ 5.4  3.9  1.7  0.4]
 [ 4.6  3.4  1.4  0.3]
 [ 5.   3.4  1.5  0.2]
 [ 4.4  2.9  1.4  0.2]
 [ 4.9  3.1  1.5  0.1]]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
['setosa' 'versicolor' 'virginica']
```

二、 经典预测算法

- **2.2.2 用sklearn构建逻辑回归模型**

逻辑回归代码示例

- Step2：拆分成训练集和测试集

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(iris.data,iris.target,test_size=0.3,
random_state=123456)
```

```
[[ 0.36538529  1.37198083 -2.06564697 -0.94881488]
 [ 0.27680493 -1.33031335  0.5338162  -1.03778136]
 [-1.40278862 -1.46945218  2.12708947  2.18107388]]
[ 0.23501757  0.84472013 -1.0328022 ]
```

- Step3：模型训练

```
from sklearn.linear_model import LogisticRegression
# 实例化模型并训练
model = LogisticRegression()
model.fit(X_train,y_train)
# 查看模型结果
print(model.coef_)
print(model.intercept_)
```

因为iris数据集中target有3个类别，所以coef_是 3*4 的2D数组，intercept_是一个长度为3的1D数组。我们前面有说过，多分类的逻辑回归，有几个类就会构建几个逻辑回归模型。

二、 经典预测算法

- **2.2.2 用sklearn构建逻辑回归模型**

逻辑回归代码示例

- Step4: 模型评估

```
# 方法1：直接使用模型的score方法计算正确率
print(model.score(X_test,y_test))
# 方法2：使用sklearn.metrics下的classification_report方法
# 先对测试集进行预测
y_pred = model.predict(X_test) #预测类别标签
y_pred_prob = model.predict_proba(X_test) #预测类别概率
# 分类评估报告 classification_report
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
#可以通过以下查看预测试结果
print(y_pred)
print(y_pred_prob)
# 验证三个概率相加等于1
y_pred_prob[0].sum()
```

二、 经典预测算法

- **2.2.2 用sklearn构建逻辑回归模型**

逻辑回归代码示例

- 完整代码

```
# 导包
from sklearn import datasets
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
# 导入sklearn自带的iris数据集：
iris = datasets.load_iris()
# 实例化模型并训练
model = LogisticRegression()
model.fit(X_train,y_train)
y_pred = model.predict(X_test) #预测类别标签
# 分类评估报告 classification_report
print(classification_report(y_test,y_pred))
```

>>完整代码

二、 经典预测算法

- **2.2.2 用sklearn构建逻辑回归模型**

绘制分类模型的决策边界图（了解）(1/2)

- 分析

绘制决策边界图不是分类模型的必需步骤，主要目的是用来更好的演示分类模型是如何对数据进行分类的，便于理解不同的分类算法的原理。
决策边界图的绘制方法是这样的：

- 一般选择2个维度来绘制一个2D的散点图，这样容易理解
- 不同的类别在散点图中用不同的颜色标记出来
- 使用绘制散点图的2个维度来训练分类模型

在散点图中确定x和y的取值范围，并在该范围内按照一个最小间隔来绘制网格，可以理解网格交叉处的那些点是虚拟的点，基于他们的坐标值(x,y)可以使用模型对它进行预测，得到预测结果z
基于网格交叉点的(x,y)和z的结果来绘制出决策边界
网格划分得越细致，那么决策边界也越精细。

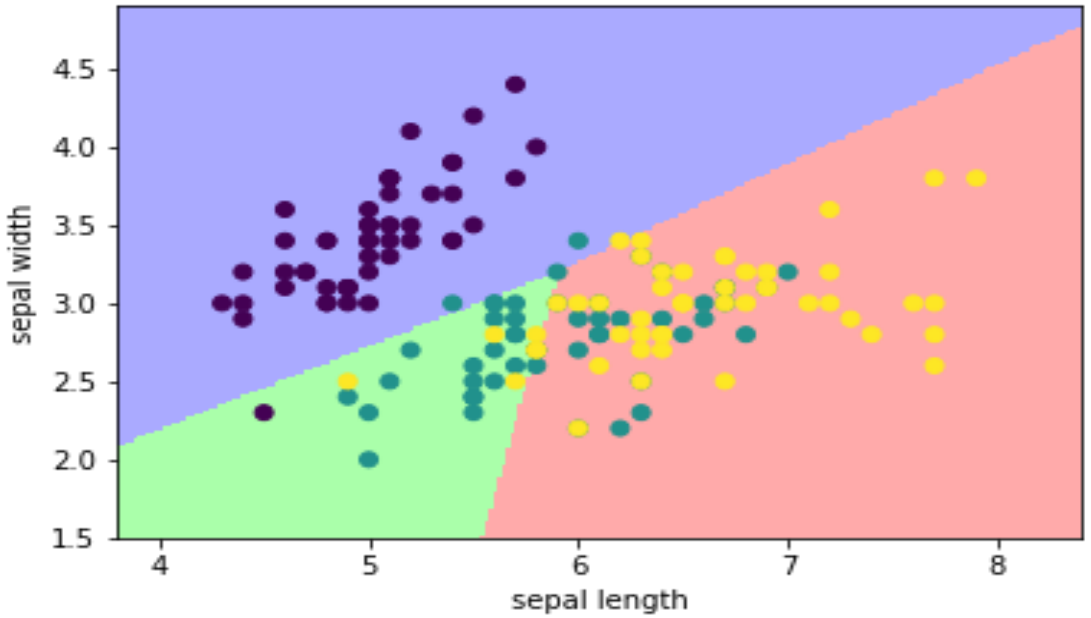
二、 经典预测算法

- **2.2.2 用sklearn构建逻辑回归模型**

绘制分类模型的决策边界图（了解）（2/2）

PS：完整代码见PPT备注

- 代码及图表分析



>> 完整代码

从上图可以明显的看到逻辑回归的决策边界是线性的：想象一下，逻辑回归算法就是在这个2D平面上画了三条直线，然后把三个类别就区分出来了。

通过对本小节的学习，我们了解到：

- 使用skleanr.linear_model的LogisticRegression类来构建逻辑回归模型
- 逻辑回归模型构建的基本流程：
- 先实例化模型：model= LogisticRegression()
- 训练模型：model.fit(X_train,y_train)
- 对测试集预测：model.predict(X_test)，model.predict_proba(X_test)
- 评估模型准确度：model.score(X_test,y_test)
- 逻辑回归的决策边界是线性的

二、 经典预测算法

启示

我们发现，在sklearn中进行机器学习建模，更换一个算法需要调整的核心代码其实非常少，基本上就是这一行代码model = LogisticRegression()，把 LogisticRegression 换成其他的算法类名就可以了。这给我们一些启示：的确，**大量的时间花在了数据准备上，尝试不同的算法成本极低，机器学习建模门槛并没有想象的那么高，我们不需要重复造算法的轮子**

sklearn的算法设计抽象得非常好，训练用**fit()**，预测用**predict()**，评估用**score()**，所有**预测类算法基本通用**

学习机器学习不仅仅只是学算法，更重要的学习用机器学习来解决问题的思路

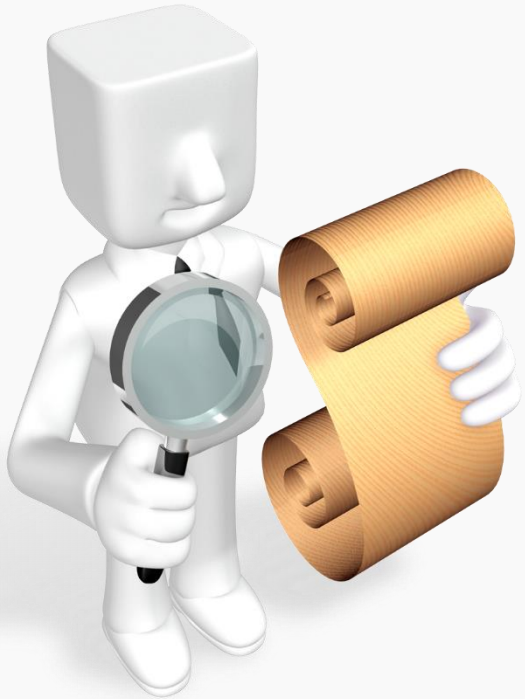
经典预测算法（三）
--K近邻（KNN）

核心原理

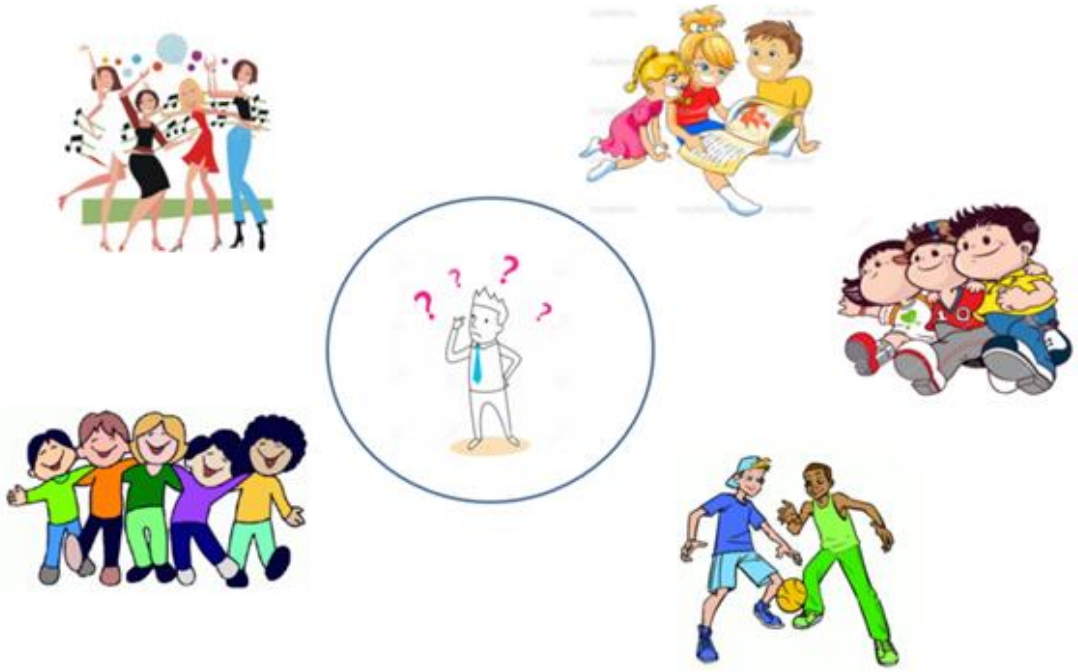
二、 经典预测算法

- **2.5.1 K近邻算法核心原理 (1/9)**

什么是 K近邻?



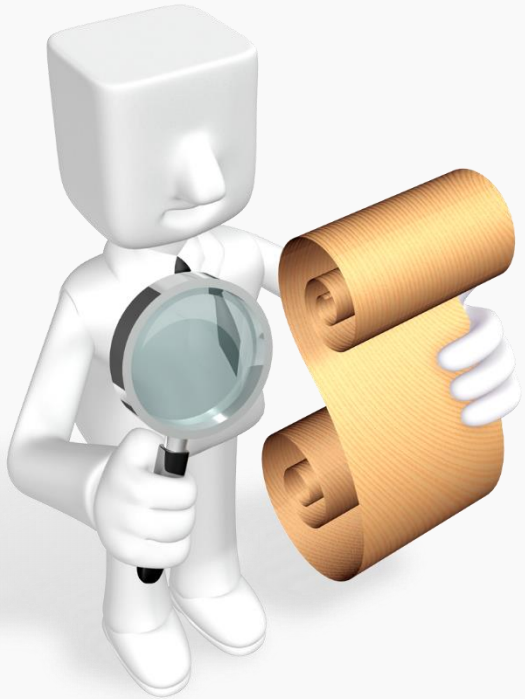
K近邻（K Nearest Neighbors，简称为KNN）算法的核心思想是：只要知道你的朋友（邻居）是什么样的人，就能知道你是什么样的人。”



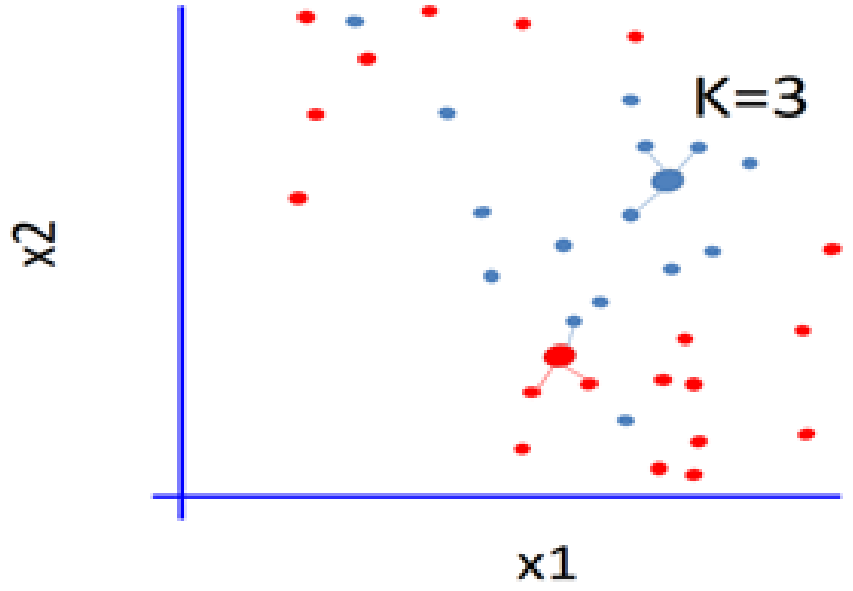
二、 经典预测算法

- **2.5.1 K近邻算法核心原理 (2/9)**

理解K近邻



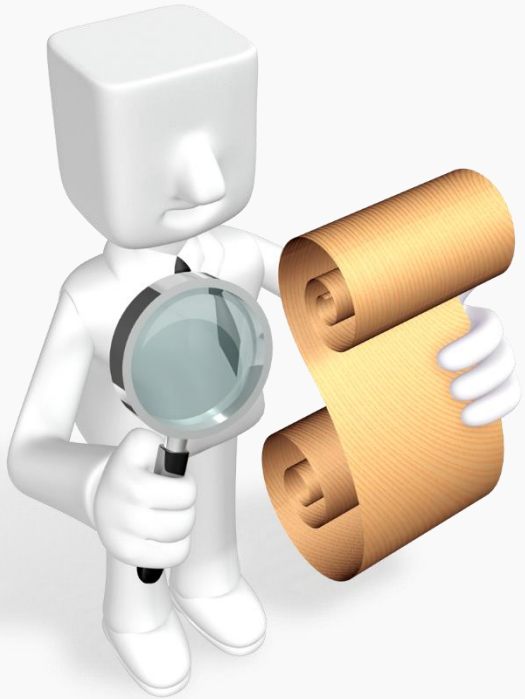
K近邻的K表示要选择多少个邻居点来对一个数据点进行预测。如下图所示，我们假设K=3，上方较粗的蓝色点由于它的3个邻居都是蓝色的，所以我们就预测它也是蓝色；下方较粗的那个红色点，由于它的3个邻居点中，有2个邻居点都是红色（超过1半），所有我们就预测它也是红色。



二、 经典预测算法

- **2.5.1 K近邻算法核心原理 (3/9)**

如何寻找K近邻



邻居就是与给定的数据点最相似的点，如何衡量两个数据点的相似性呢?

0.23

3

342.7

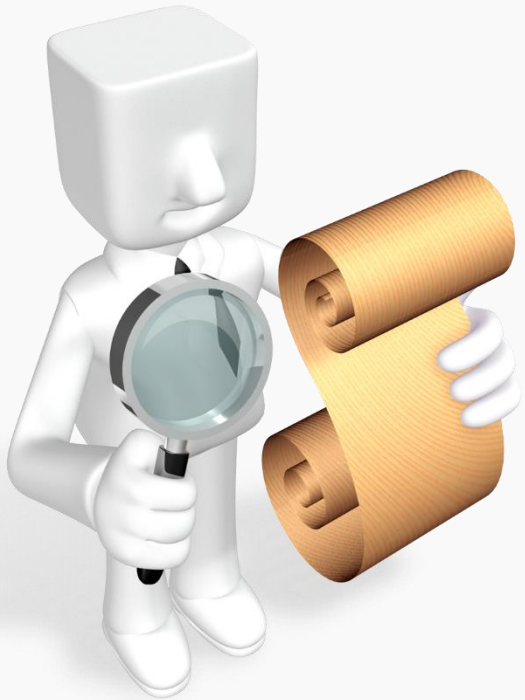
距离（Distance）

在数学上我们可以用两点之间的距离来衡量相似度，距离越小说明越相似

二、 经典预测算法

- **2.5.1 K近邻算法核心原理 (4/9)**

常见距离度量公式



- 欧氏距离

$$d(x,y) := \sqrt{(x_1-y_1)^2+(x_2-y_2)^2+\cdots+(x_n-y_n)^2} = \sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$$

- 曼哈顿距离

$$d_{12} = |x_1-x_2| + |y_1-y_2|$$

- 切比雪夫距离

$$D_{\text{Chebyshev}}(p,q) := \max_i(|p_i-q_i|).$$

- 闵可夫斯基距离

$$d_{12} = \sqrt[p]{\sum_{k=1}^{n}|x_{1k}-x_{2k}|^p}$$

- 马氏距离

$$\mathrm{D}(X_i,X_j) = \sqrt{(X_i-X_j)^T S^{-1}(X_i-X_j)}$$

- 汉明距离

两个等长字符串 s1 与 s2 之间的汉明距离定义为将其中一个变为另外一个所需要作的最小替换次数。例如字符串“1111”与“1001”之间的汉明距离为 2

- 夹角余弦

$$\cos\theta = \frac{x_1x_2+y_1y_2}{\sqrt{x_1^2+y_1^2}\sqrt{x_2^2+y_2^2}}$$

或者

$$\cos(\theta) = \frac{a\cdot b}{|a|\,|b|}$$

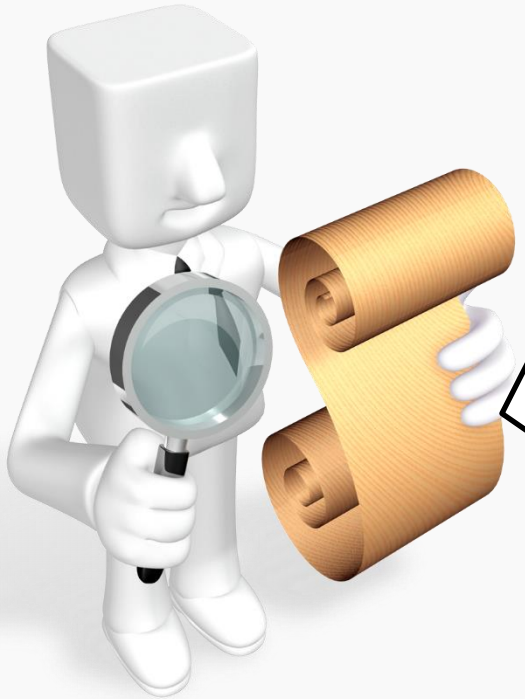
- 皮尔逊相关系数

$$\rho_{XY} = \frac{\mathrm{Cov}(X,Y)}{\sqrt{D(X)}\sqrt{D(Y)}} = \frac{E((X-EX)(Y-EY))}{\sqrt{D(X)}\sqrt{D(Y)}}$$

二、 经典预测算法

- **2.5.1 K近邻算法核心原理 (5/9)**

K近邻算法示例(1/4)



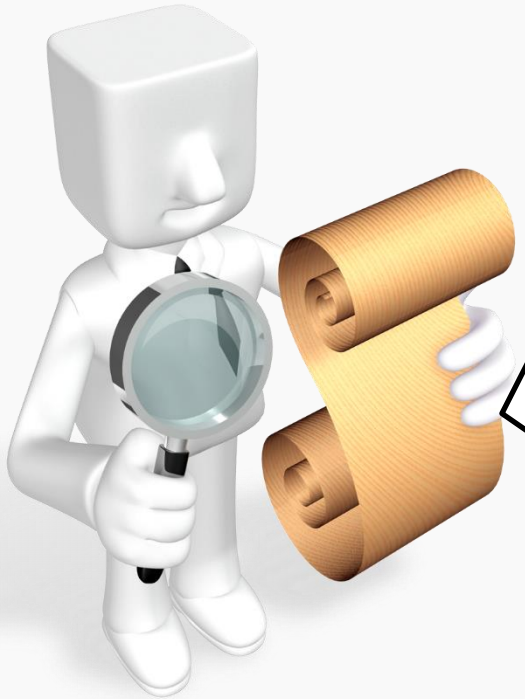
K近邻算法是一个非常简单易懂的算法，我们通过如下的一个例子来详细讲解它的计算过程

K近邻算法流程

二、 经典预测算法

- **2.5.1 K近邻算法核心原理 (6/9)**

K近邻算法示例(2/4)



- **任务描述：**
  对John的违约标记（Yes/No)进行预测。

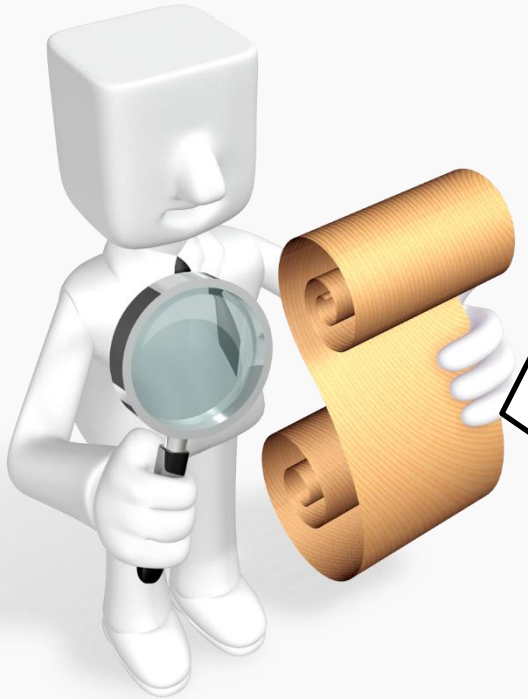
数据集

客户ID	年龄	收入	信用卡张数	违约标记
George	35	35K	3	No
Rachel	22	50K	2	Yes
Steve	63	200K	1	No
Tom	59	170K	1	No
Anne	25	40K	4	Yes
John	37	50K	2	?

二、 经典预测算法

- 2.5.1 K近邻算法核心原理 (7/9)

K近邻算法示例(3/4)



- Step1,2：指定K值，计算预测样本与训练样本的距离

Step 1：选择k值：k=3

Step 2：计算距离

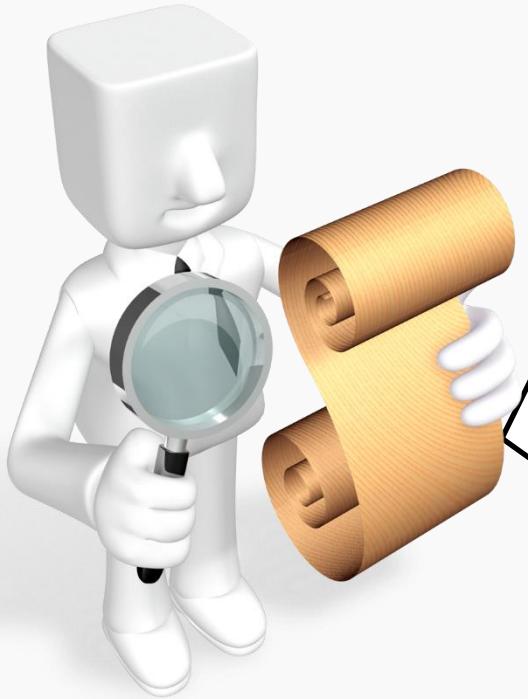
John	37	50K	2	?

客户ID	年龄	收入	信用卡张数	违约标记	John与其他客户的距离（欧式距离）
George	35	35K	3	No	sqrt $[(35-37)^2+(35-50)^2+(3-2)^2]=15.16$
Rachel	22	50K	2	Yes	sqrt $[(22-37)^2+(50-50)^2+(2-2)^2]=15$
Steve	63	200K	1	No	sqrt $[(63-37)^2+(200-50)^2+(1-2)^2]=152.23$
Tom	59	170K	1	No	sqrt $[(59-37)^2+(170-50)^2+(1-2)^2]=122$
Anne	25	40K	4	Yes	sqrt $[(25-37)^2+(40-50)^2+(4-2)^2]=15.74$

二、 经典预测算法

- 2.5.1 K近邻算法核心原理 (8/9)

K近邻算法示例(4/4)



- Step3：对距离排序，找出距离最近的k个邻居

客户ID	年龄	收入	信用卡张数	违约标记	距离排序	John与其他客户的距离（欧式距离）
George	35	35K	3	No	2	sqrt [(35-37)^2+(35-50)^2 +(3-2)^2]=15.16
Rachel	22	50K	2	Yes	1	sqrt [(22-37)^2+(50-50)^2 +(2-2)^2]=15
Steve	63	200K	1	No	5	sqrt [(63-37)^2+(200-50)^2 +(1-2)^2]=152.23
Tom	59	170K	1	No	4	sqrt [(59-37)^2+(170-50)^2 +(1-2)^2]=122
Anne	25	40K	4	Yes	3	sqrt [(25-37)^2+(40-50)^2 +(4-2)^2]=15.74

- Step4：根据k个邻居的类别值投票做出预测判断

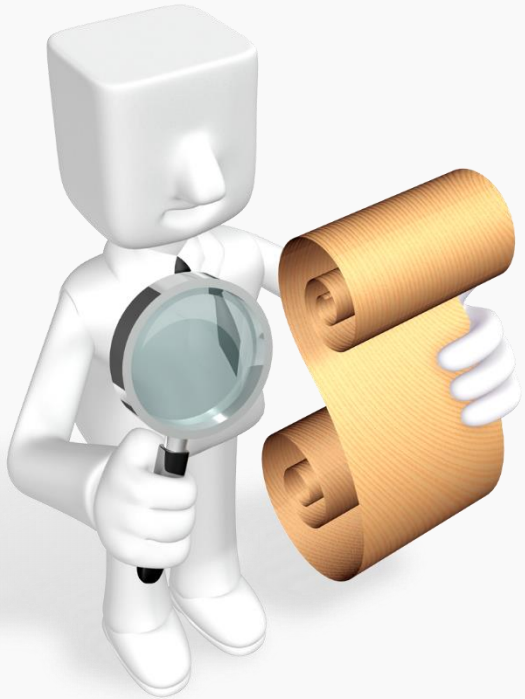
客户ID	年龄	收入	信用卡张数	违约标记	距离排序	John与其他客户的距离（欧式距离）
George	35	35K	3	No	2	sqrt [(35-37)^2+(35-50)^2 +(3-2)^2]=15.16
Rachel	22	50K	2	Yes	1	sqrt [(22-37)^2+(50-50)^2 +(2-2)^2]=15
Steve	63	200K	1	No	5	sqrt [(63-37)^2+(200-50)^2 +(1-2)^2]=152.23
Tom	59	170K	1	No	4	sqrt [(59-37)^2+(170-50)^2 +(1-2)^2]=122
Anne	25	40K	4	Yes	3	sqrt [(25-37)^2+(40-50)^2 +(4-2)^2]=15.74

John	37	50K	2	YES

二、 经典预测算法

- **2.5.1 K近邻算法核心原理 (9/9)**

为什么说K近邻是一种懒惰学习算法?



- **K近邻算法只有模型预测，没有模型训练**

  通过前面的描述，我们发现K近邻这种算法和其他分类算法截然不同，它没有模型训练，只有模型预测。待预测的数据在训练数据中进行遍历搜索，寻找到它的K个邻居，然后根据邻居的分类值来对新数据进行预测。正因为它这种只有需要预测的时候才干活的特性，所以我们称它为懒惰算法，而别的算法都是要事先基于训练样本训练模型的。K近邻算法也被称为基于实例的学习算法。

- **懒惰学习算法的优缺点**

  优点当然是简单，缺点是对存储空间的需求量很大，需要占用的空间直接取决于训练样本数据量的大小，并且预测的时候需要对训练样本中的所有数据进行遍历计算，所以运行时间会较长。

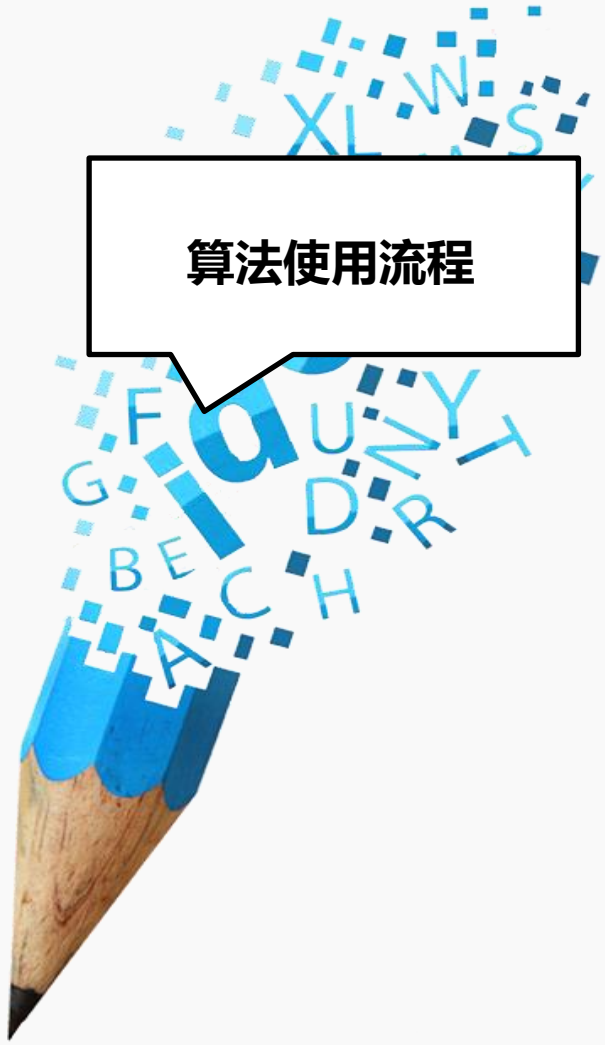
通过对本小节的学习，我们了解到：

- K近邻是一个非常简单的算法，它基于距离来判断两个数据点之间的相似度并依此来寻找邻居，根据邻居的值对待预测数据的值进行预测
- K近邻是一种懒惰学习的算法，没有训练过程，只有预测过程
- K近邻算法需要调节的最核心参数就是K值，不同的K值对预测结果影响较大，K太大准确率会降低，K太小，预测结果不稳定。

算法实现

二、 经典预测算法

- **2.5.2 用sklearn构建 KNN 模型（1/6）**



在sklearn中使用KNN算法的流程：

- 实例化一个KNN模型model = KNeighborsClassifier()
- 调用模型的model.fit(X,y)方法，传入数据集X和y来训练模型，其实这个方法对于KNN算法来说什么也没干。
- 其他算法这一步是查看模型结果，KNN由于没有模型训练过程，自然也没有模型结果可查看
- 调用模型的model.predict(X)方法，返回预测的标签值，还可以调用model.predict_proba(X)方法，返回预测的概率
- 调用模型的model.score(X,y)方法，返回模型的正确率

二、 经典预测算法

- **2.5.2 用sklearn构建 KNN 模型（2/6）**

KNN算法参数说明

KNN分类算法使用sklearn.neighbors模块的KNeighborsClassifier类，使用前先导入算法类:from sklearn.neighbors import KNeighborsClassifier

参数说明：

通过model = KNeighborsClassifier()来实例化一个KNN模型，可以指定如下的参数：

sklearn.neighbors.KNeighborsClassifier(n_neighbors=5,
weights=’uniform’, algorithm=’auto’, leaf_size=30, p=2,
metric=’minkowski’, metric_params=None, n_jobs=1, **kwargs)

几个重要的参数说明：

n_neighbors：指定邻居的个数，就是KNN算法中的k值，默认值=5

weights：基于邻居给出预测值时考虑邻居的权重，'uniform'表示邻居权重一样大，'distance'表示距离小的权重大

metric：距离计算公式，默认是'minkowski' 闵氏距离

p：闵氏距离的参数，p=2就是欧式距离，p=1就是曼哈顿距离

二、 经典预测算法

- **2.5.2 用sklearn构建 KNN 模型（3/6）**

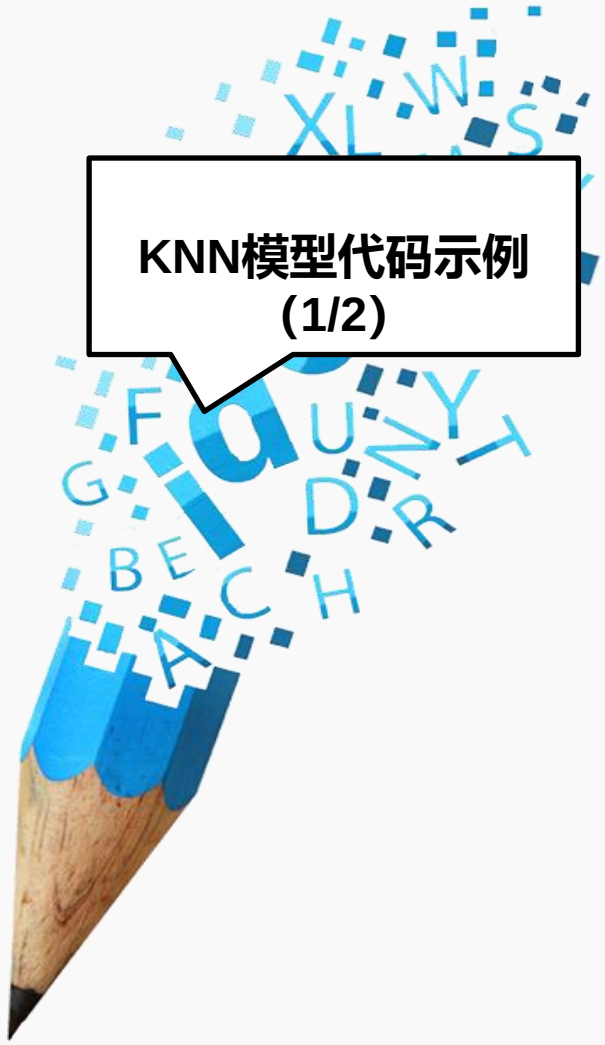
- **KNN模型的结果**

  KNN算法不需要模型训练，所以没有什么结果可以查看

- **KNN模型的方法**
  - ✓ fit(X,y)：输入数据集X和y对模型进行训练
  - ✓ predict(X)：输入数据集X，返回目标变量y的预测标签
  - ✓ predict_proba(X)：输入数据集X，返回目标变量y每一个类别的概率
  - ✓ score(X,y)：输入数据集X和y，返回模型预测结果的正确率（accuracy）

二、 经典预测算法

- **2.52 用sklearn构建 KNN 模型（4/6）**

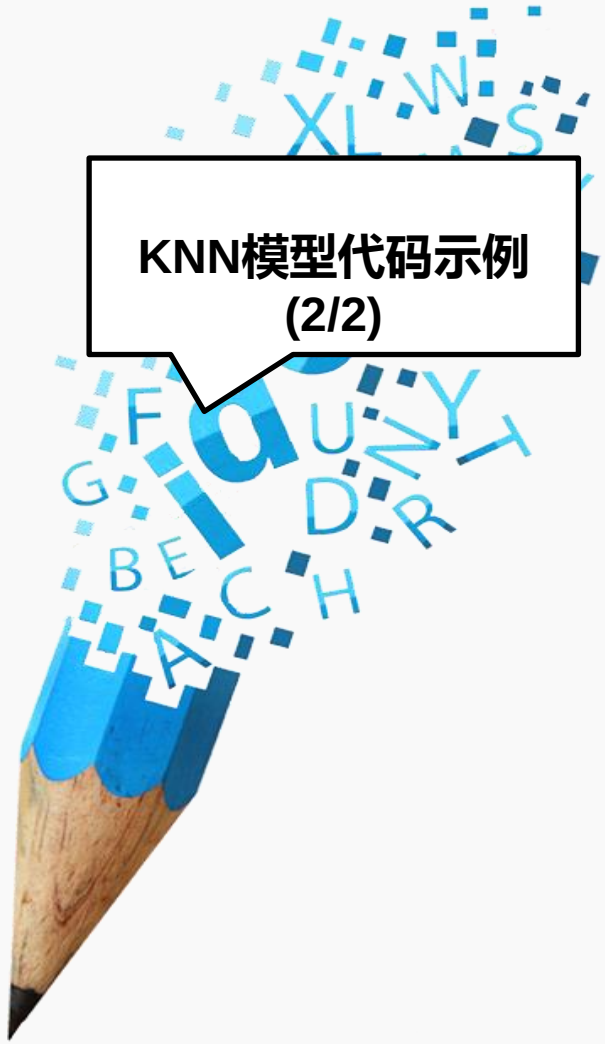


KNN模型代码示例 (1/2)

```
# Step1: 使用sklearn自带的iris数据集
from sklearn import datasets
# 导入sklearn自带的iris数据集
iris = datasets.load_iris()
# Step2: 拆分成训练集和测试集
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(iris.data,iris.target,test_size=0.3,
random_state=123456)
# Step3: 模型训练
from sklearn.neighbors import KNeighborsClassifier
# 实例化模型并训练
model = KNeighborsClassifier()
model.fit(X_train,y_train)
```

二、 经典预测算法

- **2.5.2 用sklearn构建 KNN 模型（5/6）**



KNN模型代码示例 (2/2)

```
# Step4：模型评估
# 方法1：直接使用模型的score方法计算正确率
print(model.score(X_test,y_test))
# 方法2：使用sklearn.metrics下的classification_report方法
# 先对测试集进行预测
y_pred = model.predict(X_test)   #预测类别标签
y_pred_prob = model.predict_proba(X_test)  #预测类别概率
# 分类评估报告classification_report
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

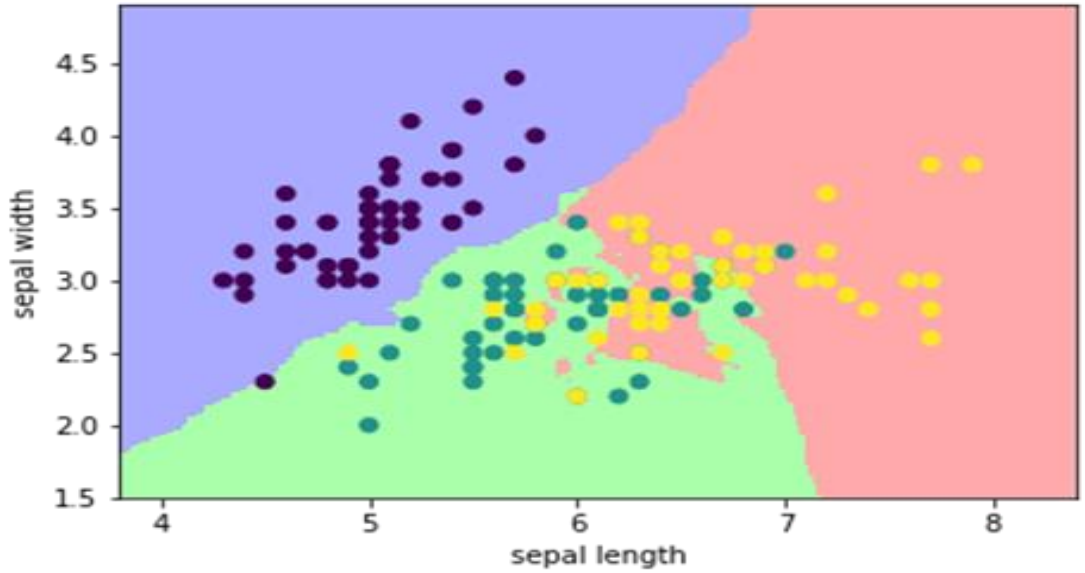
```
0.977777777778
             precision    recall  f1-score   support

          0       1.00      1.00      1.00        15
          1       1.00      0.93      0.97        15
          2       0.94      1.00      0.97        15

avg / total       0.98      0.98      0.98        45
```

二、 经典预测算法

- **2.5.2 用sklearn构建 KNN 模型（6/6）**

- **使用花萼的长度和宽度绘制散点图，并绘出分类决策边界**

从KNN模型的决策边界可以看到：KNN算法的决策边界不像其他算法有规律可循，它完全是基于实例的学习而得到预测结果，我们会得到什么样的决策边界在预测完成之前不可预期。



>> 完整代码

>> 完整代码见 PPT 备注

通过对本小节的学习，我们了解到：

- 使用skleanr.neighbors的KNeighborsClassifier类来构建KNN分类模型
- KNN模型构建的基本流程：
- 先实例化模型：model= KNeighborsClassifier()
- 训练模型：model.fit(X_train,y_train)
- 对测试集预测：model.predict(X_test)，model.predict_proba(X_test)
- 评估模型准确度：model.score(X_test,y_test)
- KNN模型的决策边界不像逻辑回归，决策树之类的方法那么有明显的规律（线性划分或者垂直划分）

经典预测算法（三）
-- Kmeans

- 何谓聚类分析
- K均值聚类分析原理
- K均值聚类分析代码实现

一、聚类分析概述

聚类是一种把相似数据合并在一起的方法，就是我们常说的“人以群分，物以类聚”。聚类是一种“非监督的学习算法”——事先并不需要有类别标注的样本来监督学习，而是直接从数据中学习模式。聚类是一种“数据探索”的分析方法，它帮助我们在大量的数据中探索发现数据结构。

什么是聚类分析？

分类与聚类的区别是：

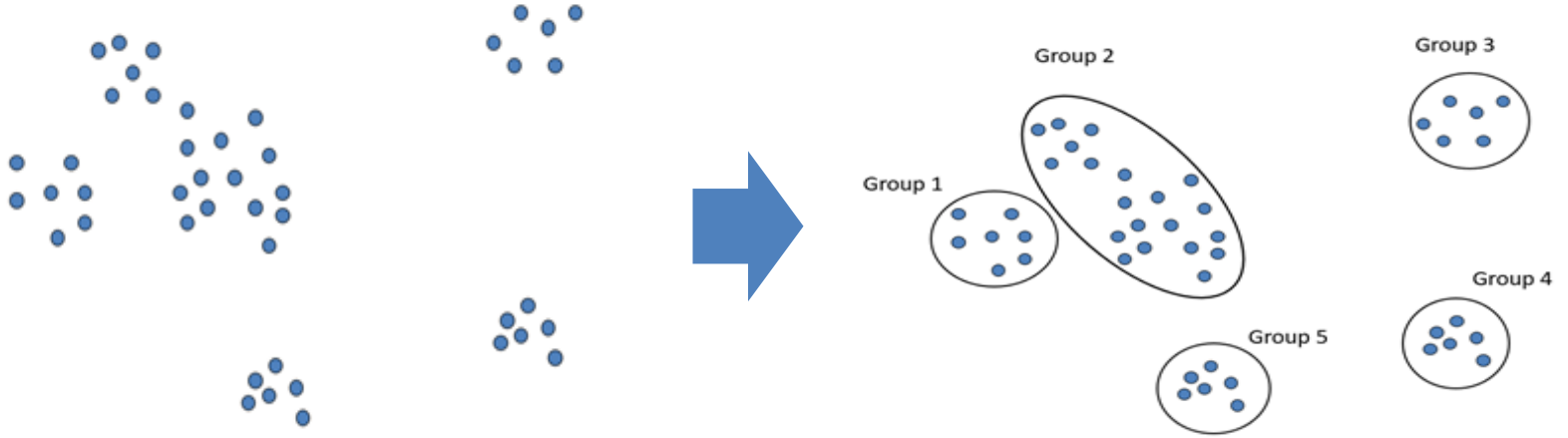
- 分类问题中的类别标签是事先确定的，要解决的问题是对样本所属的标签进行预测
- 聚类问题中数据对应的类是未知的，我们事先并不知道数据中可能会包含多少个类，聚类要解决的问题正是通过算法来发现隐藏在数据中的这些类

一、聚类分析概述

什么是聚类分析？

举个栗子：

下图中有规律的分布着一些数据，通过我们的肉眼观察，我们大概就能知道应该如何对这些数据进行分类



聚类分析的目标就是通过算法把这些数据自动的聚为不同的类：

一、聚类分析概述

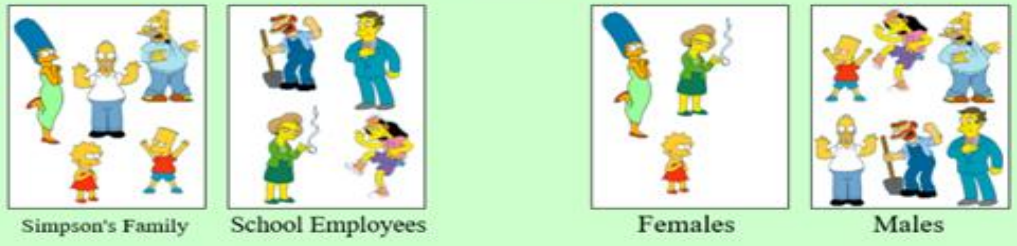
聚类是主观的

同一批数据，使用不同的算法或者聚类标准，得到的聚类结果是不同的，所以大家一定要记住：聚类是主观的！

如下图所示，我们对《辛普森一家》中的人物进行聚类，如果采用不同的分组标准，得到的结果就不一样。

在下面的这些人物中，天然的分组是什么？

聚类是主观的！



一、聚类分析概述

聚类的核心是
距离度量

聚类分析的核心原理是基于数据之间的相似度即距离来把相似的数据合并在一起，这与之前介绍的分类算法中的K近邻的原理是比较类似的。如何衡量两个数据点的相似性？

在数学上我们可以用两点之间的距离来衡量相似度，距离越小说明越相似

一、聚类分析概述

有哪些距离度量?

- 欧氏距离

$$d(x,y) := \sqrt{(x_1-y_1)^2+(x_2-y_2)^2+\cdots+(x_n-y_n)^2} = \sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$$

- 曼哈顿距离

$$d_{12} = |x_1-x_2| + |y_1-y_2|$$

- 切比雪夫距离

$$D_{\text{Chebyshev}}(p,q) := \max_i(|p_i-q_i|).$$

- 闵可夫斯基距离

$$d_{12} = \sqrt[p]{\sum_{k=1}^{n}|x_{1k}-x_{2k}|^p}$$

- 马氏距离

$$\mathrm{D}(X_i,X_j) = \sqrt{(X_i-X_j)^T S^{-1}(X_i-X_j)}$$

- 汉明距离

两个等长字符串 s1 与 s2 之间的汉明距离定义为将其中一个变为另外一个所需要作的最小替换次数。例如字符串“1111”与“1001”之间的汉明距离为 2

- 夹角余弦

$$\cos\theta = \frac{x_1x_2+y_1y_2}{\sqrt{x_1^2+y_1^2}\sqrt{x_2^2+y_2^2}}$$

或者

$$\cos(\theta) = \frac{a\cdot b}{|a|\,|b|}$$

- 皮尔逊相关系数

$$\rho_{XY} = \frac{Cov(X,Y)}{\sqrt{D(X)}\sqrt{D(Y)}} = \frac{E((X-EX)(Y-EY))}{\sqrt{D(X)}\sqrt{D(Y)}}$$

一、聚类分析概述

根据聚类算法的原理不同，需要对训练数据集特征进行如下处理：

变量标准化

- 标准化处理：把变量转换为均值为0方差为1
- 归一化处理：把变量转换为最小值为0，最大值为1
- 如果变量是类别型，例如性别，我们要把它用于距离计算，可以使用独热编码的方式先把变量转换为数值，然后再用上面的方面进行去量纲化处理。

二、K均值聚类算法原理

K均值算法

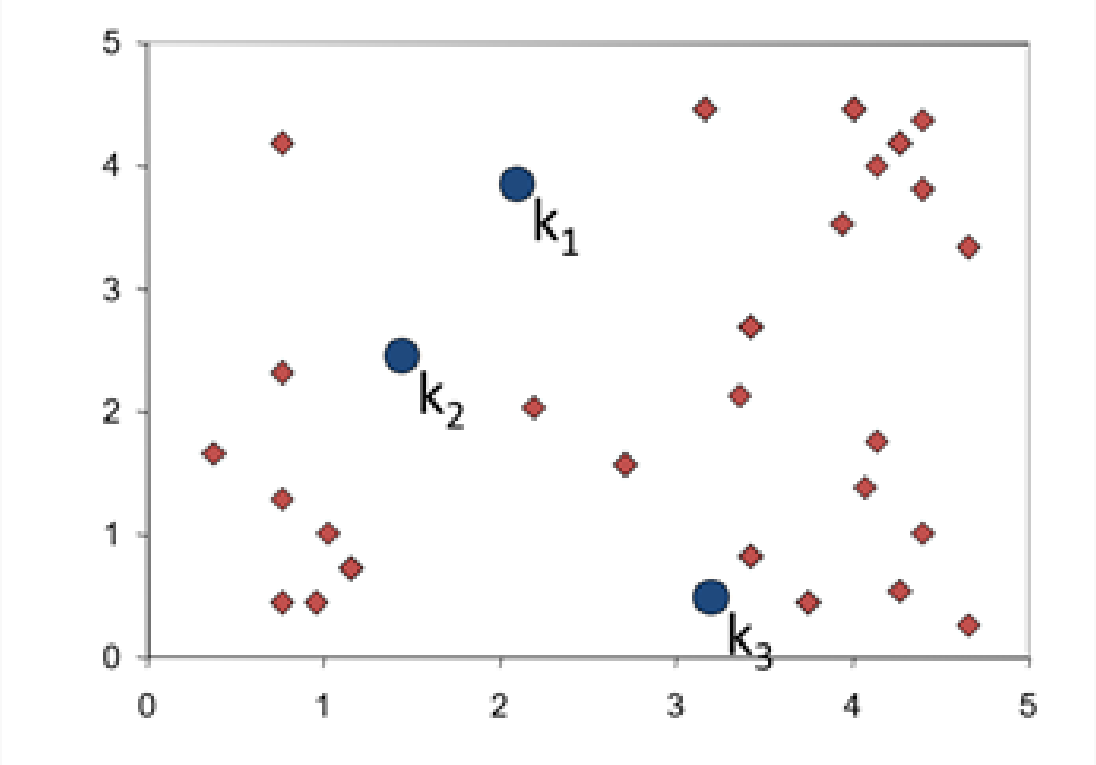
K均值算法，又叫K-means，是基于划分（Partition）的聚类算法中的典型代表，也是最常用的一种聚类算法，它的算法核心思想是这样的：

- ✓ **Step1**：随机选择K个数据点作为“种子”
- ✓ **Step2**：根据数据点与“种子”的距离大小进行类分配
- ✓ **Step3**：更新类中心点的位置，以新的类中心点作为“种子”
- ✓ **Step4**：按照新的“种子”对数据归属的类进行重新分配
- ✓ **Step5**：更新类中心点（step3、step4），不断迭代，直到类中心点变化很小

二、K均值聚类算法原理

K均值算法

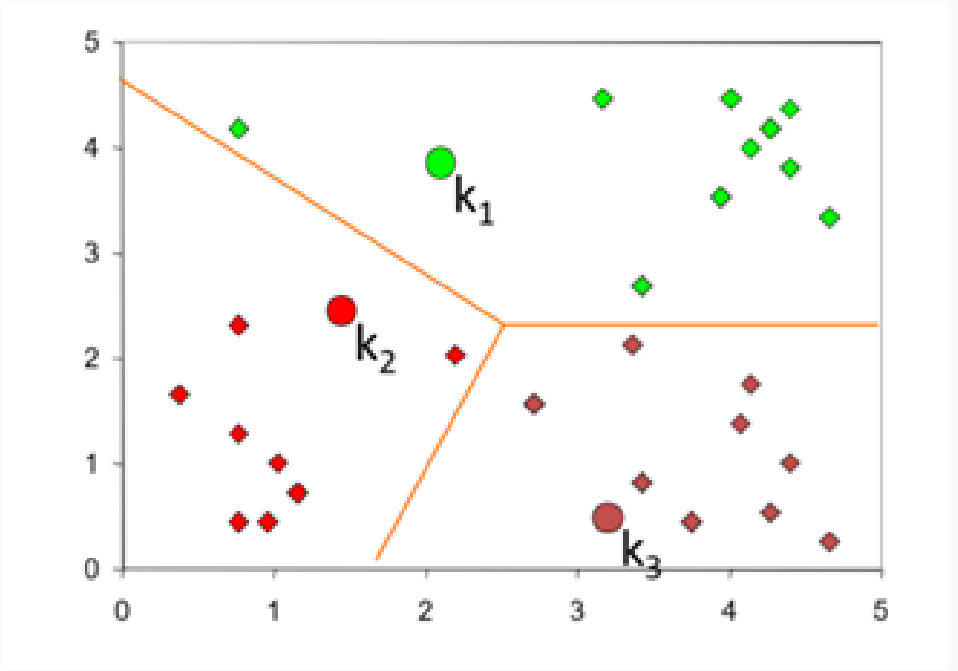
Step1：随机选择K个数据点作为“种子”



二、K均值聚类算法原理

Step2：根据数据点与“种子”的距离大小进行类分配

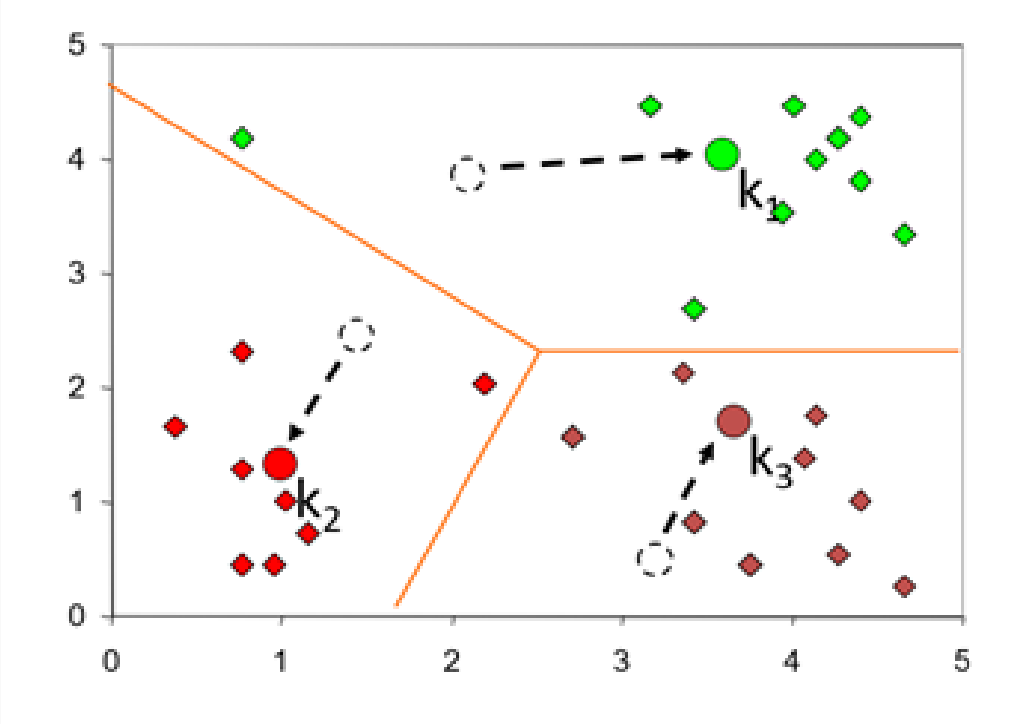
K均值算法



二、K均值聚类算法原理

Step3：更新类中心点的位置，以新的类中心点作为“种子”

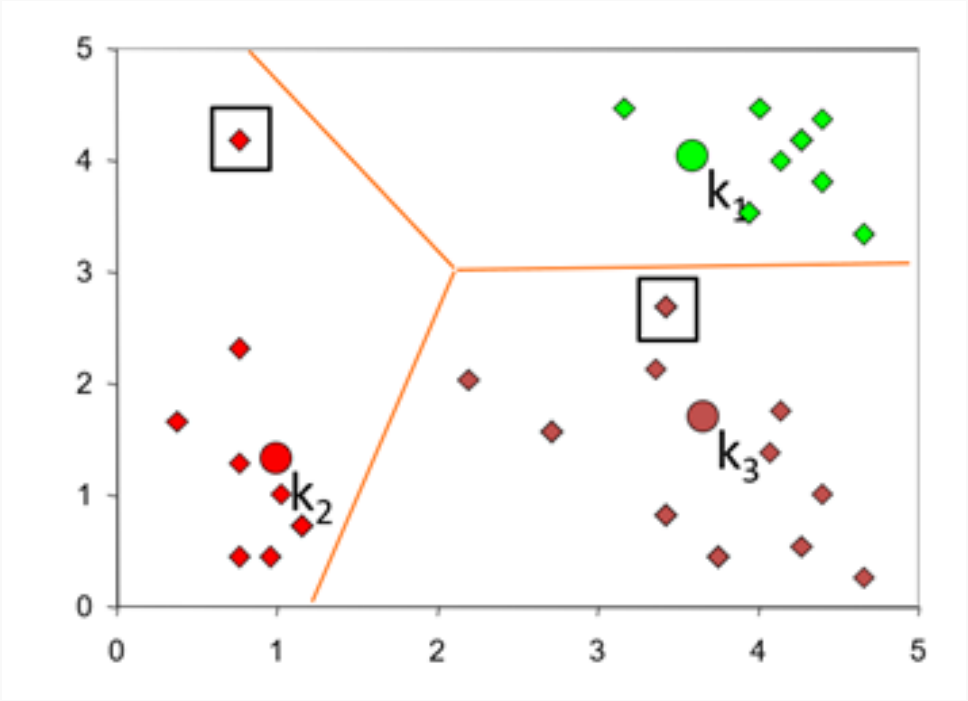
K均值算法



二、K均值聚类算法原理

Step4：按照新的“种子”对数据归属的类进行重新分配

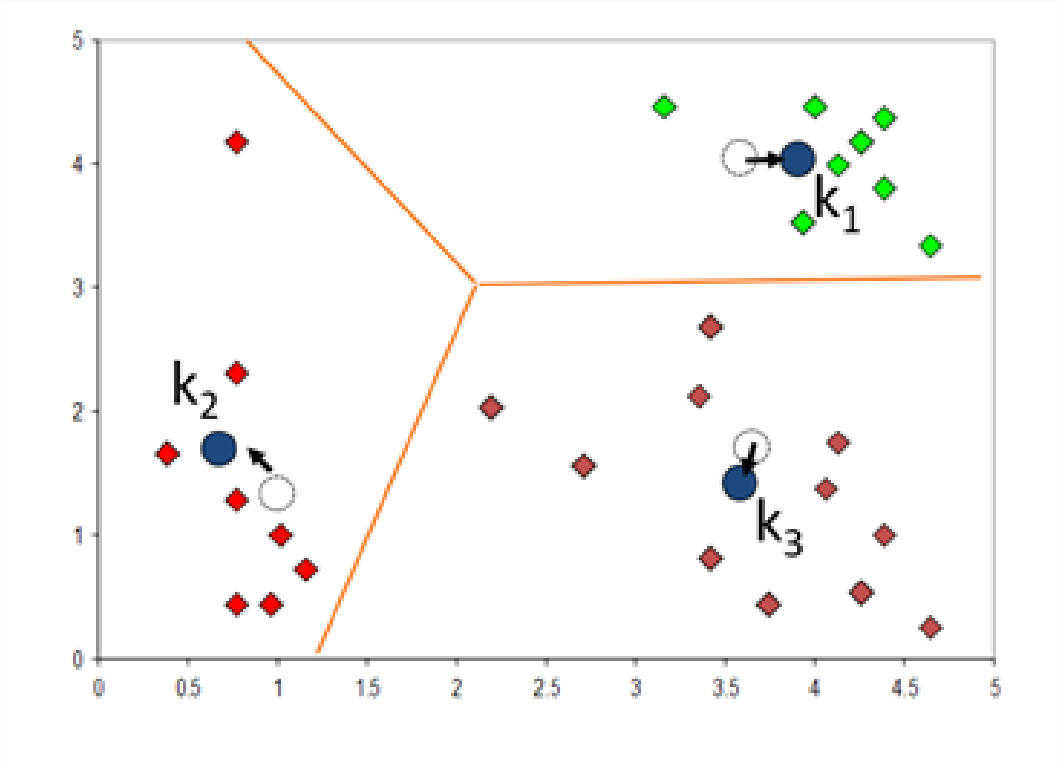
K均值算法



二、K均值聚类算法原理

Step5：更新类中心点（->step3 ->step4），不断迭代

K均值算法



二、K均值聚类算法原理

K均值算法优缺点

优点：

- 算法原理简单，处理速度较快
- 当数据点是密集的，且类与类之间区别明显时，效果较好

缺点：

- K均值算法中的K值选定比较难
- 对孤立点比较敏感
- 结果不稳定，初始值（种子）的选定对结果有一定的影响

Thank you~