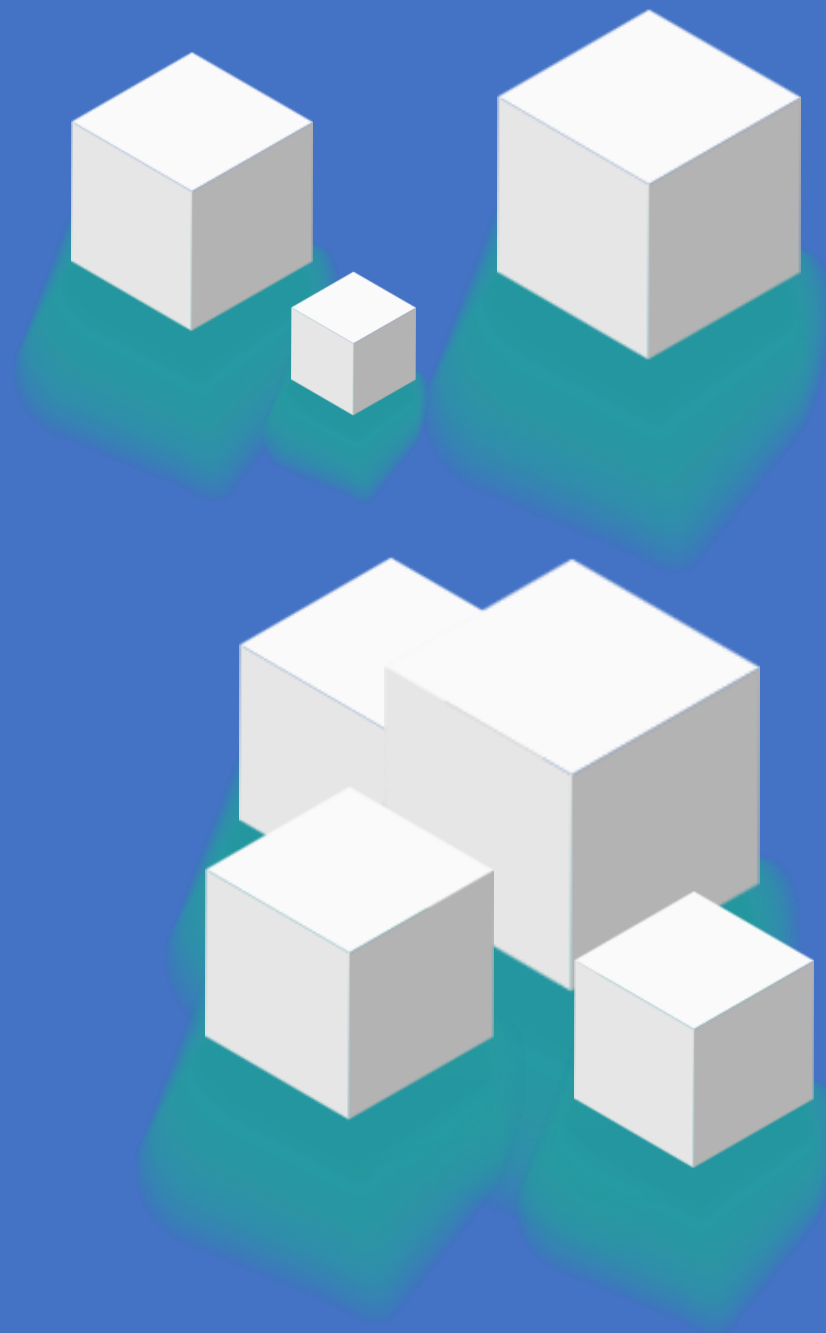
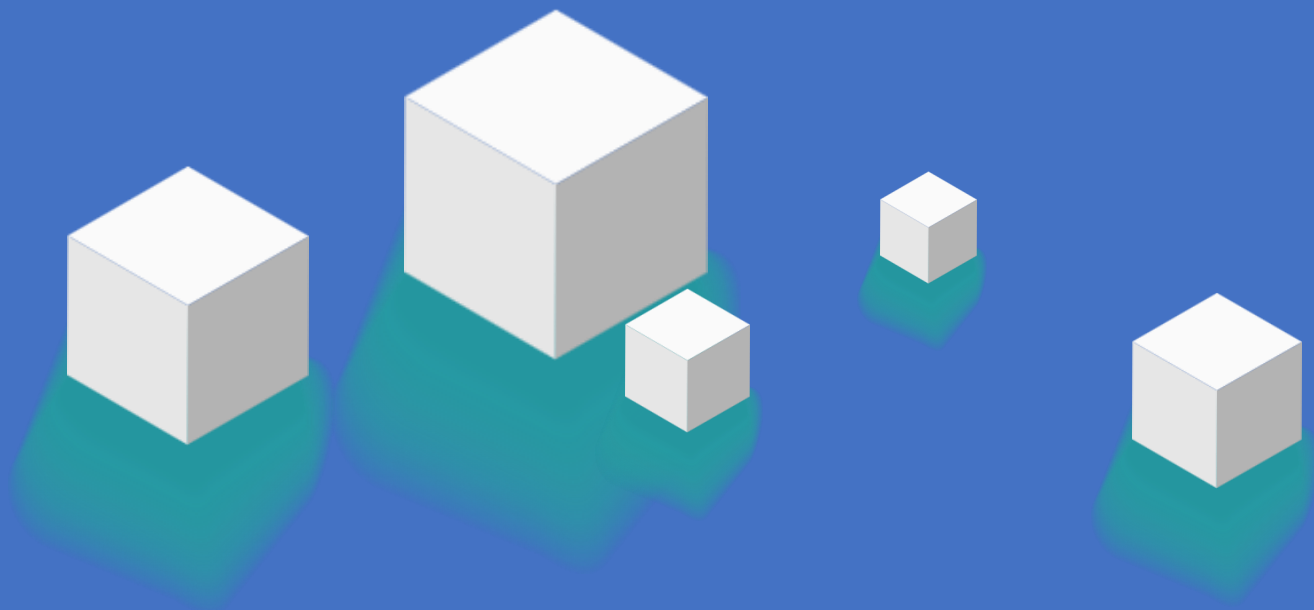


分析式AI基础



>> 今天的学习目标

分析式AI基础

- 分析式AI与生成式AI
- 十大经典机器学习算法
- 常用分类算法

贝叶斯分类器

决策树与随机森林

SVM支持向量机

逻辑回归

项目实战

- CASE: 二手车价格预测

分类与回归的关系

数据探索

特征选择

模型训练与预测

分析式AI与生成式AI

Thinking: 分析式AI与生成式AI的区别是什么？

	分析式AI (Analytical AI)	生成式AI (Generative AI)
目标	分析和解释数据，提供洞察、预测或 决策支持 。	生成新的内容 ，这些内容在形式和风格上与训练数据相似
用途	数据分析、预测建模、风险评估、分类任务等。	文本生成、图像创作、音频生成、内容创作等。
输出	报告、图表、预测结果、决策建议等。	新的文本、图像、音频、视频等内容。
技术原理	基于统计分析、机器学习算法 （如回归分析、决策树、神经网络等）。	基于生成式大模型 ，如DeepSeek, Qwen, ChatGPT (Transformer架构)。
数据需求	需要高质量、标注清晰的数据 ，以学习数据中的规律。	需要大量未标注数据 ，以学习数据分布。
应用场景	金融风险评估、医疗诊断、市场预测、客户行为分析等。	广告设计、内容创作、游戏开发、艺术创作等。
风险挑战	模型可能过度拟合训练数据 ，导致在新数据上表现不佳；数据偏差可能导致错误决策。	生成内容可能包含错误 、误导性信息或不符合伦理的内容需防止内容被滥用。

机器学习的模型

——10大经典模型

- 分类算法：C4.5, 朴素贝叶斯 (Naive Bayes) , SVM, KNN, Adaboost, CART
- 聚类算法：K-Means, EM
- 关联分析：Apriori
- 连接分析：PageRank

机器学习算法工具包

算法	工具
决策树	<code>from sklearn.tree import DecisionTreeClassifier</code>
朴素贝叶斯	<code>from sklearn.naive_bayes import MultinomialNB</code>
SVM	<code>from sklearn.svm import SVC</code>
KNN	<code>from sklearn.neighbors import KNeighborsClassifier</code>
Adaboost	<code>from sklearn.ensemble import AdaBoostClassifier</code>
K-Means	<code>from sklearn.cluster import KMeans</code>
EM	<code>from sklearn.mixture import GMM</code>
Apriori	<code>from efficient_apriori import apriori</code>
PageRank	<code>import networkx as nx</code>

贝叶斯原理

贝叶斯原理：

- 贝叶斯为了解决一个叫“逆向概率”问题写了一篇文章，尝试解答在没有太多可靠证据的情况下，怎样做出更符合数学逻辑的推测
- “逆向概率”是相对“正向概率”而言
- 正向概率，比较容易理解，比如我们已经知道袋子里面有N个球，不是黑球就是白球，其中M个是黑球，那么把手伸进去摸一个球，就能知道摸出黑球的概率是多少 => 这种情况往往是上帝视角，即了解了事情的全貌再做判断
- 逆向概率，贝叶斯则从实际场景出发，提了一个问题：如果我们事先不知道袋子里面黑球和白球的比例，而是通过我们摸出来的球的颜色，能判断出袋子里面黑白球的比例么？


$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

贝叶斯原理

贝叶斯原理：

- 影响了接下来近200年的统计学理论
- 建立在主观判断的基础上：在我们不了解所有客观事实的情况下，同样可以先估计一个值
- 先验概率，通过经验来判断事情发生的概率
- 后验概率，就是发生结果之后，推测原因的概率
- 条件概率， $P(A|B)$ 表示在事件 B 已经发生的前提下，事件 A 发生的概率
- 似然函数，把概率模型的训练过程理解为求参数估计的过程

比如，一个硬币在10次抛落中正面均朝上 => 那么这个硬币是均匀的可能性是多少？这里硬币均匀就是个参数，似然函数就是用来衡量这个模型的参数

经过推导，贝叶斯推理告诉我们后验概率是与先验概率和似然函数成正比得，即

$$P(A|B) = P(A) \cdot \frac{P(B|A)}{P(B)}$$

The diagram illustrates the components of Bayes' theorem. Arrows point from the terms in the equation to their descriptions:

- $P(A|B)$ points to: 后验概率，可能性函数，分类概率
- $P(A)$ points to: 先验概率，类别概率
- $\frac{P(B|A)}{P(B)}$ points to: 似然函数，条件概率
- $P(B)$ points to: 证据，归一化常数（计算可省略）

贝叶斯原理

Thinking: 假设有一种病叫做“贝叶死”，它的发病率是万分之一，即10000 人中会有1个人得病。现有一种测试可以检验一个人是否得病的准确率是99.9%，误报率（假阳）是0.1%

那么，如果一个人被查出来患有“叶贝死”，实际上患有的可能性有多大？

（查出患有“贝叶死”的准确率是99.9%，是不是实际上患“贝叶死”的概率也是99.9%？）

在10000个人中，还存在0.1%的误查的情况，也就是10个人没有患病但是被诊断成阳性。当然10000个人中，也确实存在一个患有贝叶死的人，他有99.9%的概率被检查出来

可以粗算下，患病的这个人实际上是这11个人里面的一员，即实际患病比例是 $1/11 \approx 9\%$

贝叶斯原理就是求解后验概率，假设：**A** 表示事件“测出为阳性”，**B1** 表示“患有贝叶死”，**B2** 表示“没有患贝叶死”

患有贝叶死的情况下，测出为阳性的概率为 $P(A|B1)=99.9\%$

没有患贝叶死，但测出为阳性的概率为 $P(A|B2)=0.1\%$

患有贝叶死的概率为 $P(B1)=0.01\%$

没有患贝叶死的概率 $P(B2)=99.99\%$

那么我们检测出来为阳性，而且是贝叶死的概率 $P(B1, A) = P(B1)*P(A|B1)=0.01\%*99.9\%=0.00999\% \approx 0.01\%$

$P(B1,A)$ 代表的是联合概率，同样我们可以求得

$P(B2,A)=P(B2)*P(A|B2)=99.99\%*0.1\%=0.09999\% \approx 0.1\%$

贝叶斯原理

检查为阳性的情况下，患有贝叶死的概率，即 $P(B_1|A)$

$$P(B_1 | A) = \frac{0.01\%}{0.01\% + 0.1\%} \approx 9\%$$

检查出是阳性的条件下，但没有患有贝叶死的概率为

$$P(B_2 | A) = \frac{0.1\%}{0.01\% + 0.1\%} \approx 90.9\%$$

0.01%+0.1%均出现在了 $P(B_1|A)$ 和 $P(B_2|A)$ 的计算中作为分母，称之为论据因子，也相当于一个权值因子

我们可以总结出贝叶斯公式

$$P(B_i | A) = \frac{P(B_i)P(A | B_i)}{P(B_1)P(A | B_1) + P(B_2)P(A | B_2)}$$

从而我们可以得到通用的贝叶斯公式

$$P(B_i | A) = \frac{P(B_i)P(A | B_i)}{\sum_{i=1}^n P(B_i)P(A | B_i)}$$

朴素贝叶斯

朴素贝叶斯：

- 一种简单但极为强大的预测建模算法
- 假设每个输入变量是独立的。这是一个强硬的假设，实际情况并不一定，但是这项技术对于绝大部分的复杂问题仍然非常有效

朴素贝叶斯模型由两种类型的概率组成：

- 每个类别的概率 $P(C_j)$
- 每个属性的条件概率 $P(A_i|C_j)$

什么是类别概率：

- 假设我有7个棋子，其中3个是白色的，4个是黑色的

那么棋子是白色的概率就是 $3/7$ ，黑色的概率就是 $4/7 \Rightarrow$ 这个是类别概率

什么是条件概率：

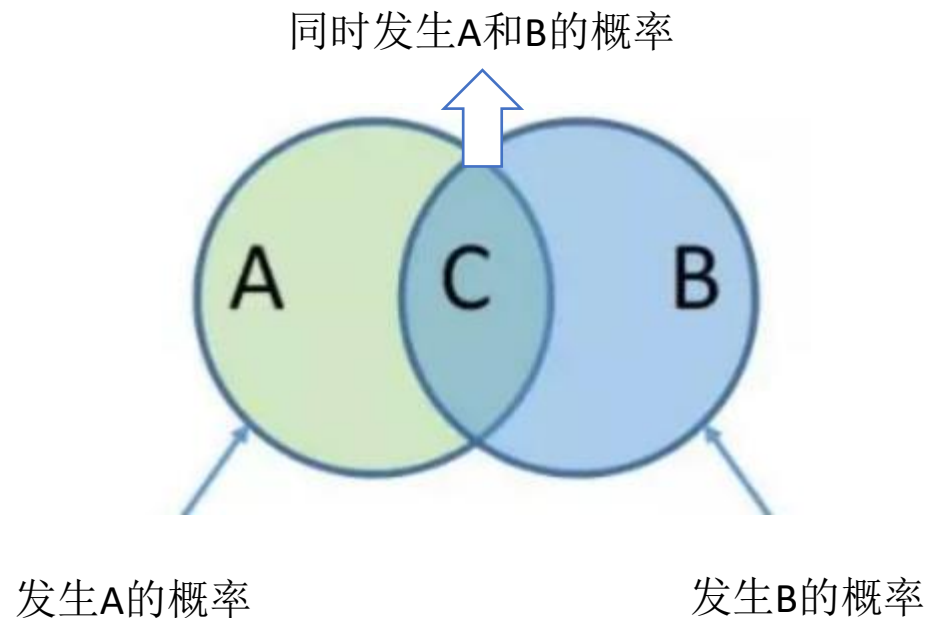
- 假设我把这7个棋子放到了两个盒子里，其中盒子A里面有2个白棋，2个黑棋；盒子B里面有1个白棋，2个黑棋

那么在盒子A中抓到白棋的概率就是 $1/2$ ，抓到黑棋的概率也是 $1/2$ ，这个就是条件概率，也就是在某个条件（比如在盒子A中）下的概率

朴素贝叶斯

训练朴素贝叶斯模型的过程：

- 在朴素贝叶斯中，我们要统计的是属性的条件概率，也就是假设取出来的是白色的棋子，那么它属于盒子A的概率是 $\frac{2}{3}$
- Step1: 先给出训练数据，以及这些数据对应的分类
- Step2, 计算类别概率和条件概率
- Step3, 使用概率模型（基于贝叶斯原理）对新数据进行预测



朴素贝叶斯分类器

朴素贝叶斯分类（离散值）：

- 如何根据身高，体重，鞋码，判断是否为男女，比如一个新的数据：身高“高”、体重“中”，鞋码“中” => 男 or 女？
- 求在A1、A2、A3属性下，Cj的概率

$$P(C_j | A_1A_2A_3) = \frac{P(A_1A_2A_3 | C_j)P(C_j)}{P(A_1A_2A_3)}$$

因为一共有2个类别，所以我们只需要求得P(C1|A1A2A3)和P(C2|A1A2A3)的概率即可，然后比较下哪个分类的可能性大，就是哪个分类结果

分别求下这些条件下的概率：

$$P(A1|C1)=1/2, P(A2|C1)=1/2, P(A3|C1)=1/4$$

$$P(A1|C2)=0, P(A2|C2)=1/2, P(A3|C2)=1/2$$

编号	身高	体重	鞋码	性别
1	高	重	大	男
2	高	重	大	男
3	中	中	大	男
4	中	中	中	男
5	矮	轻	小	女
6	矮	轻	小	女
7	矮	中	中	女
8	中	中	中	女

所以 $P(A1A2A3|C1)=1/16$, $P(A1A2A3|C2)=0$

因为 $P(A1A2A3|C1)P(C1)>P(A1A2A3|C2)P(C2)$ ，所以应该是C1类别，即男性

朴素贝叶斯分类器

朴素贝叶斯分类（连续值）：

- 身高180、体重120，鞋码41，请问该人是男是女呢
- 可以假设男性和女性的身高、体重、鞋码都是正态分布，通过样本计算出均值和方差，也就是得到正态分布的密度函数。有了密度函数，就可以把值代入，算出某一点的值

比如，男性的身高是均值179.5、标准差为3.697的正态分布。所以男性的身高为180的概率为0.1069

同理可以计算得出男性体重为120的概率为0.000382324，男性鞋码为41号的概率为0.120304111

$$P(A1A2A3|C1)=P(A1|C1)P(A2|C1)P(A3|C1)=0.1069*0.000382324*0.120304111=4.9169e-6$$

编号	身高（CM）	体重（斤）	鞋码（欧码）	性别
1	183	164	45	男
2	182	170	44	男
3	178	160	43	男
4	175	140	40	男
5	160	88	35	女
6	165	100	37	女
7	163	110	38	女
8	168	120	39	女

$$P(A1A2A3|C2)=P(A1|C2)P(A2|C2)P(A3|C2)=0.00000147489*0.015354144*0.120306074=2.7244e-9$$

很明显这组数据分类为男的概率大于分类为女的概率

朴素贝叶斯分类器

Excel中的NORMDIST函数:

- NORMDIST(x,mean,standard_dev,cumulative)

x: 正态分布中, 需要计算的数值;

Mean: 正态分布的平均值;

Standard_dev: 正态分布的标准差;

Cumulative: 取值为逻辑值, 即False或True, 决定了函数的形式

当为True时, 函数结果为累积分布

当为False时, 函数结果为概率密度

NORMDIST(180,179.5,3.697,0)=0.1069

Python中的概率密度:

- stats.norm.pdf(x, mu, sigma)

返回参数为 μ 和 σ 的正态分布密度函数在x处的值

```
from scipy import stats
```

```
mu = 179.5
```

```
sigma = 3.697
```

```
x = 180
```

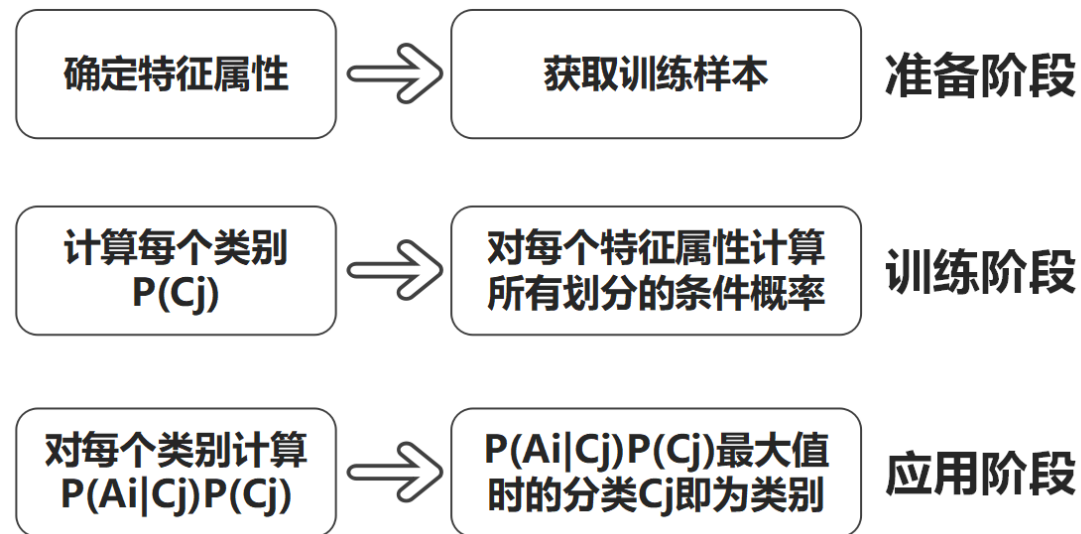
```
prob = stats.norm.pdf(x, mu, sigma)
```

```
print(prob)
```

朴素贝叶斯分类器

朴素贝叶斯分类：

- 常用于文本分类，文本过滤、情感预测、推荐系统等，尤其是对于英文等语言来说，分类效果很好
- 准备阶段，需要确定特征属性，属性值以及label => 训练集
- 训练阶段，输入是特征属性和训练样本，输出是分类器，主要工作是计算每个类别在训练样本中的出现频率及每个特征属性划分对每个类别的条件概率
- 应用阶段，使用分类器对新数据进行分类



朴素贝叶斯分类器

sklearn工具使用:

- `from sklearn.naive_bayes import GaussianNB`

高斯朴素贝叶斯: 特征变量是连续变量, 符合高斯分布, 比如说人的身高, 物体的长度

`GaussianNB(priors=None)` #模型创建

`priors`, 先验概率大小, 如果没有给定, 模型则根据样本数据自己计算 (利用极大似然法)

查看模型对象的属性:

`class_prior_`: 每个样本的概率

`class_count_`: 每个类别的样本数量

`theta_`: 每个类别中每个特征的均值

`sigma_`: 每个类别中每个特征的方差

- `from sklearn.naive_bayes import MultinomialNB`

多项式朴素贝叶斯: 特征变量是离散变量, 符合多项分布, 在文档分类中特征变量体现在一个单词出现的次数, 或者是单词的TF-IDF值等

`MultinomialNB(alpha=1.0, fit_prior=True, class_prior=None)`

`alpha`: 先验平滑因子, 默认等于1, 当等于1时表示拉普拉斯平滑

`fit_prior`: 是否去学习类的先验概率, 默认是True

`class_prior`: 各个类别的先验概率, 如果没有指定, 则模型会根据数据自动学习, 每个类别的先验概率相同, 即类别数N分之一

模型对象的属性:

`class_count_`: 训练样本中各类别对应的样本数

`feature_count_`: 每个类别中各个特征出现的次数

朴素贝叶斯分类器

sklearn工具使用:

- `from sklearn.naive_bayes import BernoulliNB`

伯努利朴素贝叶斯: 特征变量是布尔变量, 符合0/1分布, 在文档分类中特征是单词是否出现

`BernoulliNB(alpha=1.0, fit_prior=True, class_prior=None)`

`alpha`: 平滑因子, 与多项式中的`alpha`一致

`fit_prior`: 是否去学习类的先验概率, 默认是`True`

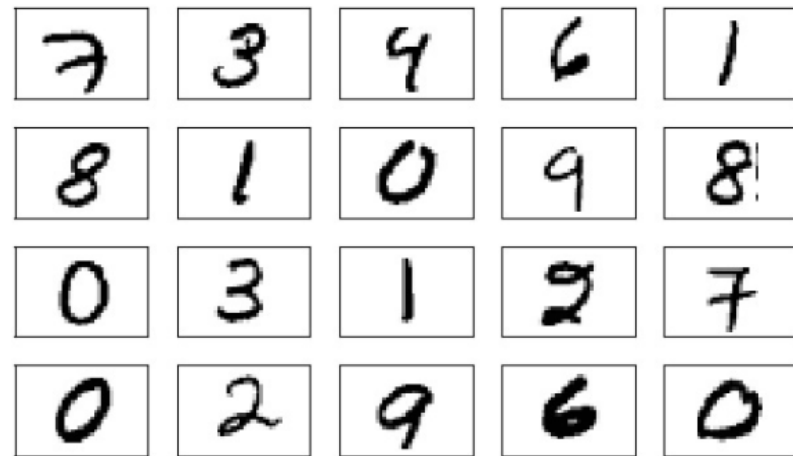
`class_prior`: 各个类别的先验概率, 如果没有指定, 则模型会根据数据自动学习, 每个类别的先验概率相同, 即类别数`N`分之一

模型对象的属性:

`class_count_`: 训练样本中各类别对应的样本数

`feature_count_`: 每个类别中各个特征出现的次数

- To Do 使用朴素贝叶斯进行mnist手写数字分类



决策树与随机森林

决策树：

- 决策树基本上就是把我们的经验总结出来
- 常见的决策树算法有C4.5、ID3和CART
- Thinking：如何构造一个判断是否去打篮球的决策树

将哪个属性（天气、温度、湿度、刮风）作为根节点是个关键问题

天气	温度	湿度	刮风	是否打球
晴天	高	中	否	否
晴天	高	中	是	否
阴天	高	高	否	是
小雨	高	高	否	是
小雨	低	高	否	否
晴天	中	中	是	是
阴天	中	高	是	否

决策树与随机森林

信息、熵以及信息增益：

- 引用香农的话来说，信息是用来消除随机不确定性的东西
- 对于机器学习中的决策树而言，如果带分类的事物集合可以划分为多个类别当中，则某个类（ x_i ）的信息可以定义为

$$I(X = x_i) = -\log_2 p(x_i)$$

随机变量的信息

当 x_i 发生时的概率

- 熵是约翰.冯.诺依曼建议使用的命名，熵=信息的期望值

$$H(X) = \sum_{i=1}^n p(x_i) I(x_i) = -\sum_{i=1}^n p(x_i) \log p(x_i)$$

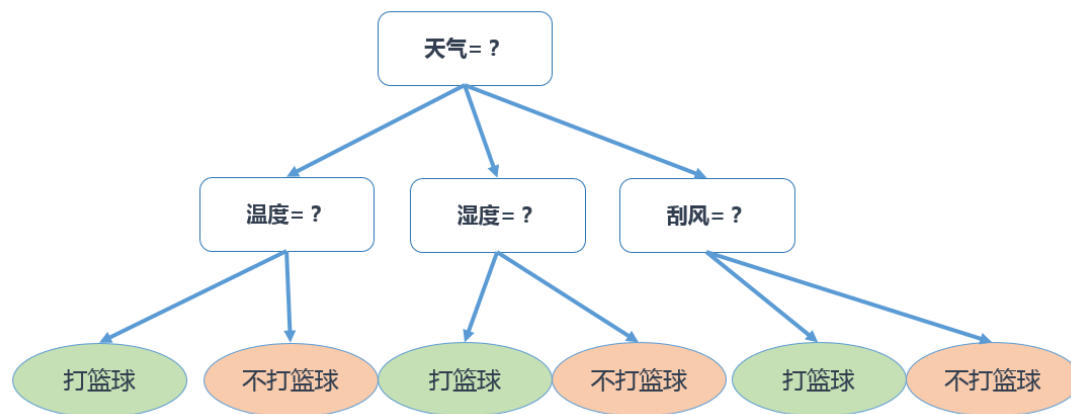
熵用来度量不确定性的，当熵越大， $X=x_i$ 的不确定性越大

对于机器学习中的分类问题，熵越大 => 这个类别的不确定性大

- 信息增益在决策树算法中是用来选择特征的指标，信息增益越大，则这个特征的选择性越好

信息增益 $g(D, A) = H(D) - H(D | A)$

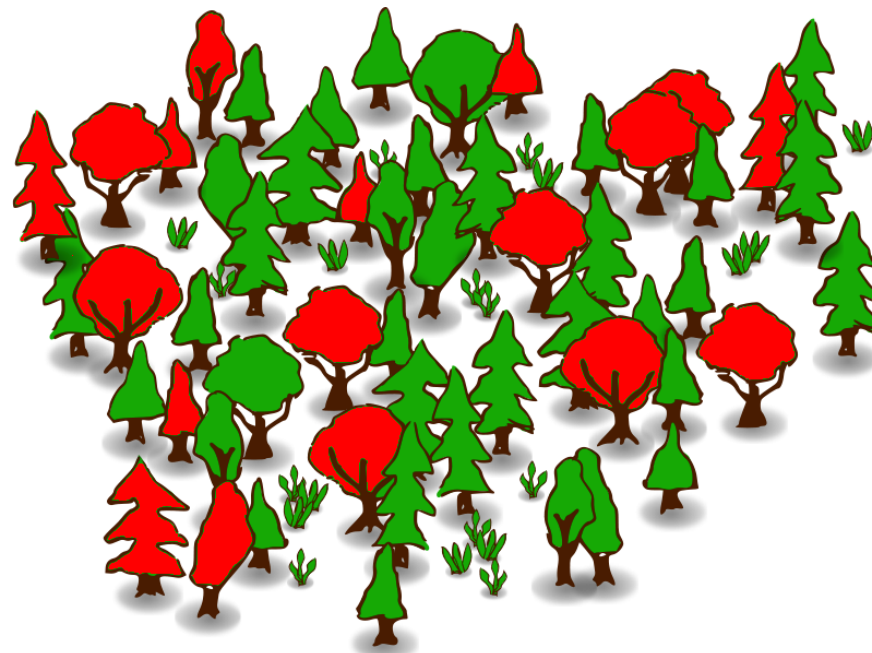
原有树的熵 $H(D)$ 增加了一个分裂节点，使得熵变成了 $H(D | A)$



决策树与随机森林

随机森林的生成：

- 森林中的每棵树都是独立的
- bagging思想，将若干个弱分类器的分类结果进行投票选择，从而组成一个强分类器
- bagging不用单棵决策树来做预测，增加了预测准确率，即不容易过拟合，减少了预测方差



少数服从多数，获得票数最多的类别
就是森林的分类结果

决策树与随机森林

随机森林的生成：

- 如果训练集大小为 N ，对于每棵树而言，随机且有放回地从训练集中的抽取 N 个训练样本（采样方式称为bootstrap sample），作为该树的训练集

每棵树的训练集都是不同的，而且里面包含重复的训练样本

- 如果每个样本的特征维度为 M ，指定一个常数 $m \ll M$ ，随机地从 M 个特征中选取 m 个特征子集，每次树进行分裂时，从这 m 个特征中选择最优的
- 每棵树都尽最大程度的生长，没有剪枝过程

随机森林分类效果（错误率）：

- 森林中任意两棵树的相关性：相关性越大，错误率越大
- 森林中每棵树的分类能力：每棵树的分类能力越强，整个森林的错误率越低

特征选择个数 m （随机森林的参数）：

- m 越大，树的相关性和分类能力会相应提升
- 如何选择最优的 m 是关键问题

决策树与随机森林

决策树工具:

- `from sklearn.tree import DecisionTreeClassifier`
- `DecisionTreeClassifier(criterion='entropy')`

`criterion`, `gini`或者`entropy`, 默认是基尼系数, 后者是信息熵

`max_depth`, 决策树最大深度, 默认为`None`

`min_samples_split`, 内部节点再划分所需最小样本数

`max_leaf_nodes`, 最大叶子节点数

`class_weight`, 类别权重

随机森林工具:

- `from sklearn.ensemble import RandomForestClassifier`
- `RandomForestClassifier(n_estimators=10, criterion='gini')`

`n_estimators`, 森林中决策树的个数, 默认为10

`max_features`, 寻找最佳分割时需要考虑的特征数目, 默认为`auto`, 即`max_features=sqrt(n_features)`

`criterion`, `gini`或者`entropy`, 默认是基尼系数, 后者是信息熵

`max_depth`, 决策树最大深度, 默认为`None`

`min_samples_split`, 内部节点再划分所需最小样本数

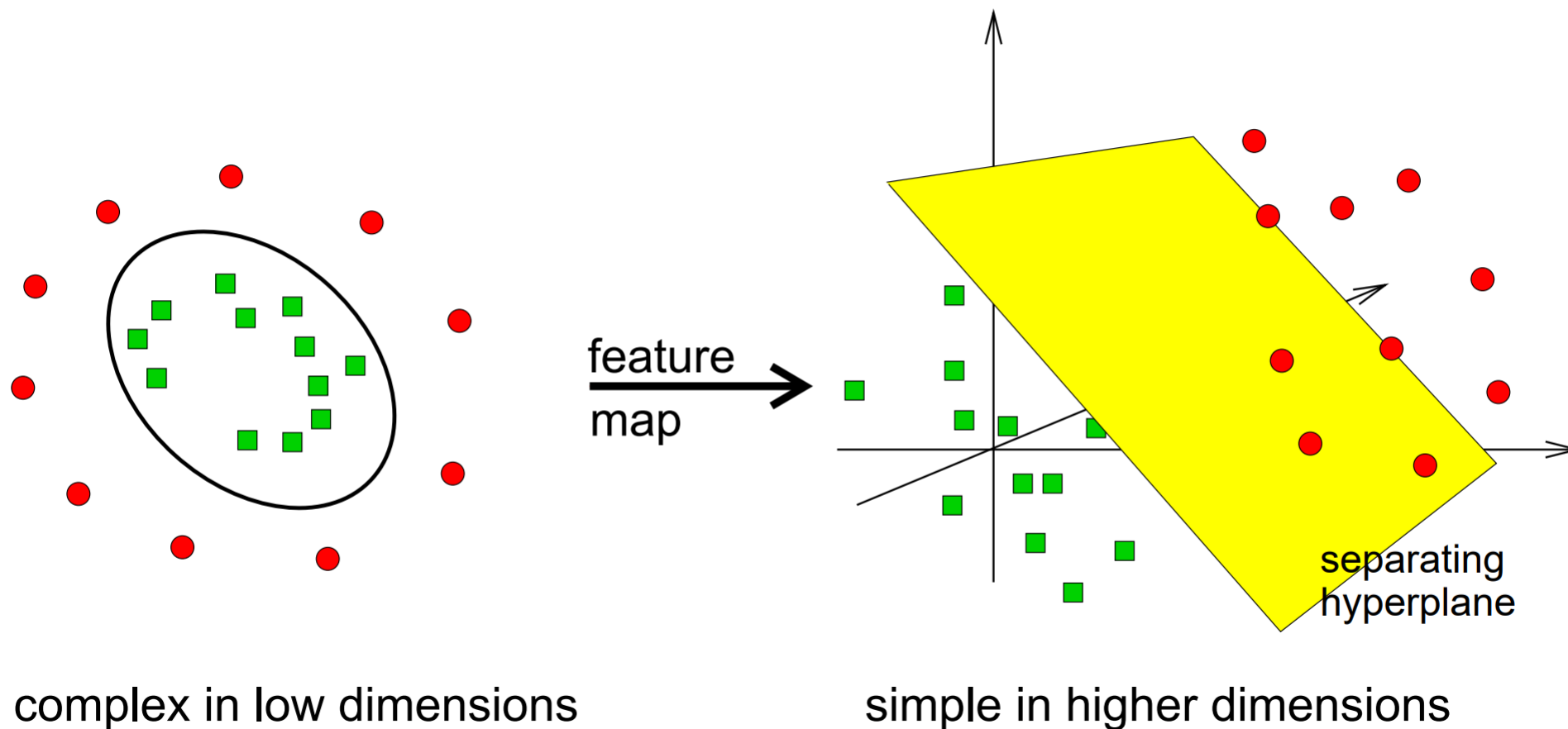
`max_leaf_nodes`, 最大叶子节点数

`class_weight`, 类别权重

SVM工具

SVM思想：一些线性不可分的问题可能是非线性可分的，也就是在高维空间中存在分离超平面（separating hyperplane）

使用非线性函数从原始的特征空间映射至更高维的空间，转化为线性可分问题



SVM工具

SVM工具:

- sklearn中支持向量分类主要有三种方法: SVC、NuSVC、LinearSVC
- `sklearn.svm.SVC(C=1.0, kernel='rbf', degree=3, gamma='auto', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', random_state=None)`
- `sklearn.svm.NuSVC(nu=0.5, kernel='rbf', degree=3, gamma='auto', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', random_state=None)`
- `sklearn.svm.LinearSVC(penalty='l2', loss='squared_hinge', dual=True, tol=0.0001, C=1.0, multi_class='ovr', fit_intercept=True, intercept_scaling=1, class_weight=None, verbose=0, random_state=None, max_iter=1000)`

常用参数:

- C, 惩罚系数, 类似于LR中的正则化系数, C越大惩罚越大
- nu, 代表训练集训练的误差率的上限 (用于NuSVC)
- kernel, 核函数类型, RBF, Linear, Poly, Sigmoid, precomputed, 默认为RBF径向基核 (高斯核函数)
- gamma, 核函数系数, 默认为auto
- degree, 当指定kernel为'poly'时, 表示选择的多项式的最高次数, 默认为三次多项式
- probability, 是否使用概率估计
- shrinking, 是否进行启发式, SVM只用少量训练样本进行计算
- penalty, 正则化参数, L1和L2两种参数可选, 仅LinearSVC有
- loss, 损失函数, 有 'hinge' 和 'squared_hinge' 两种可选, 前者又称L1损失, 后者称为L2损失
- tol: 残差收敛条件, 默认是0.0001, 与LR中的一致

SVM工具

SVM工具:

- SVC, Support Vector Classification, 支持向量机用于分类
 - SVR, Support Vector Regression, 支持向量机用于回归
 - sklearn中支持向量分类主要有三种方法: SVC、NuSVC、LinearSVC
 - 基于libsvm工具包实现, 台湾大学林智仁教授在2001年开发的一个简单易用的SVM工具包
 - SVC, C-Support Vector Classification, 支持向量分类
 - NuSVC, Nu-Support Vector Classification, 核支持向量分类, 和SVC类似, 不同的是可以使用参数来控制支持向量的个数
 - LinearSVC, Linear Support Vector Classification
- libsvm中自带了四种核函数: 线性核、多项式核、RBF以及sigmoid核
- Kernel 核的选择技巧的:
- 如果样本数量 < 特征数:
 - 方法1: 简单的使用线性核就可以, 不用选择非线性核
 - 方法2: 可以先对数据进行降维, 然后使用非线性核
 - 如果样本数量 >= 特征数
 - 可以使用非线性核, 将样本映射到更高维度, 可以得到比较好的结果

线性支持向量分类, 使用的核函数是linear

逻辑回归

逻辑回归模型

逻辑回归：

- 假设数据服从伯努利分布
- 通过极大化似然函数的方法
- 运用梯度下降来求解参数
- 将数据进行二分类

逻辑回归模型

逻辑回归的假设：

- 任何的模型都有自己的假设，在假设条件下才是适用的
- 假设1：数据服从伯努利分布

典型例子：连续的掷n次硬币（每次实验结果不受其他实验结果的影响，即n次实验是相互独立的）

贝努力分布为离散型概率分布，如果成功，随机变量取值为1；如果失败，随机变量取值为0。成功概率记为p，失败概率为 $q=1-p$

$$f_X(x) = p^x(1-p)^{1-x} = \begin{cases} p & \text{if } x = 1 \\ q & \text{if } x = 0 \end{cases}$$

对应二分类问题，样本为正类的概率p，和样本为负类的概率 $q=1-p$

$$p = h_{\theta}(x; \theta)$$

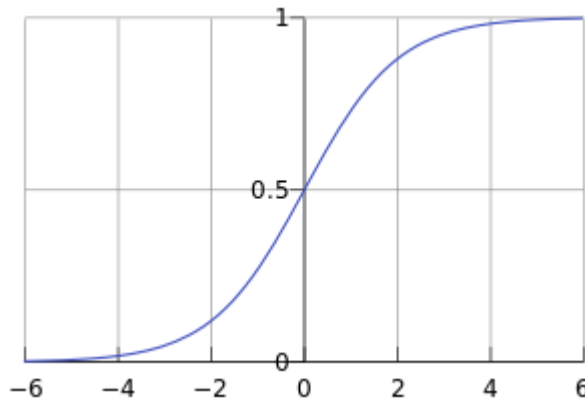
$$q = 1 - h_{\theta}(x; \theta)$$

- 假设2：正类的概率由sigmoid函数计算，即

$$g(z) = \frac{1}{1 + e^{-z}}$$



$$p = \frac{1}{1 + e^{-\theta^T x}}$$

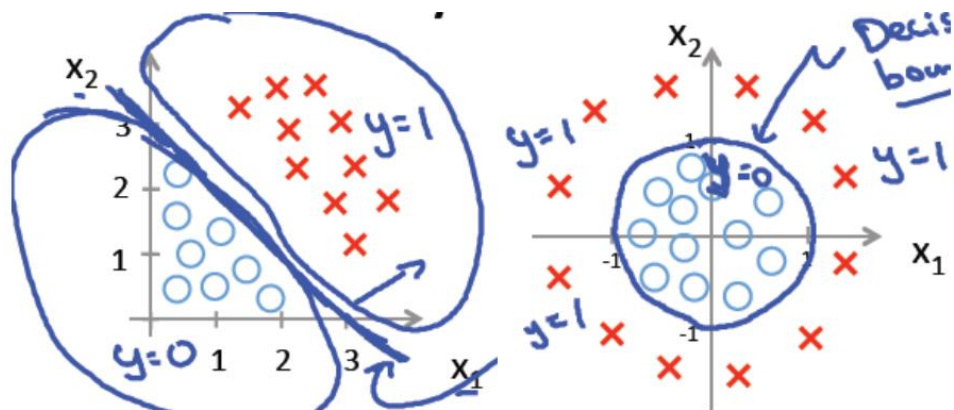


预测样本为正的的概率 $p(y = 1|x; \theta) = h_{\theta}(x; \theta) = \frac{1}{1 + e^{-\theta^T x}}$

预测样本为负的概率 $p(y = 0|x; \theta) = 1 - h_{\theta}(x; \theta) = \frac{1}{1 + e^{\theta^T x}}$

逻辑回归模型

Thinking: 决策边界是线性 or 非线性?



对于线性的决策边界: $\theta_0 + \theta_1 x_1 + \dots + \theta_n x_n = \sum_{i=1}^n \theta_i x_i = \theta^T x$

构造预测函数 $h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$

$h_\theta(x)$ 表示结果为1的概率, 对于输入 x , 分类结果为1和0的概率为

$$\begin{aligned} P(y=1 | x; \theta) &= h_\theta(x) \\ P(y=0 | x; \theta) &= 1 - h_\theta(x) \end{aligned} \quad \text{公式(1)}$$

- 构造损失函数

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

$$J(\theta) = \frac{1}{m} \sum_{i=1}^n \text{Cost}(h_\theta(x^{(i)}), y^{(i)}) = -\frac{1}{m} \sum_{i=1}^n (y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})))$$

Cost函数与J函数是基于最大似然估计推导得到的

将公式1综合起来: $P(y | x; \theta) = (h_\theta(x))^y (1 - h_\theta(x))^{1-y}$

取似然函数为 $L(\theta) = \prod_{i=1}^m P(y^{(i)} | x^{(i)}; \theta) = \prod_{i=1}^m (h_\theta(x^{(i)}))^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{1-y^{(i)}}$

对数似然函数为

$$l(\theta) = \log L(\theta) = \sum_{i=1}^m (y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})))$$

最大似然估计就是求使 $l(\theta)$ 取最大值时的 θ , 这里可以使用梯度上升法求解, 也即对 $J(\theta)$ 梯度下降法求最小值

$$J(\theta) = -\frac{1}{m} l(\theta)$$

逻辑回归模型

似然函数：

- 关于统计模型中的参数的函数，表示模型参数中的似然性
- 给定输出x时，关于参数 θ 的似然函数 $L(\theta|x)$ 等于给定参数 θ 后变量X的概率

$$L(\theta|x) = P(X = x|\theta)$$

逻辑回归：

- 如何进行分类：设定一个阈值，判断正类概率是否大于该阈值，一般阈值是0.5，所以只用判断正类概率是否大于0.5即可
- **Thinking：**为什么会在训练中将高度相关的特征去掉？
 - 1) 可解释性更好
 - 2) 提高训练的速度，特征多了，会增大训练的时间

逻辑回归模型

优点:

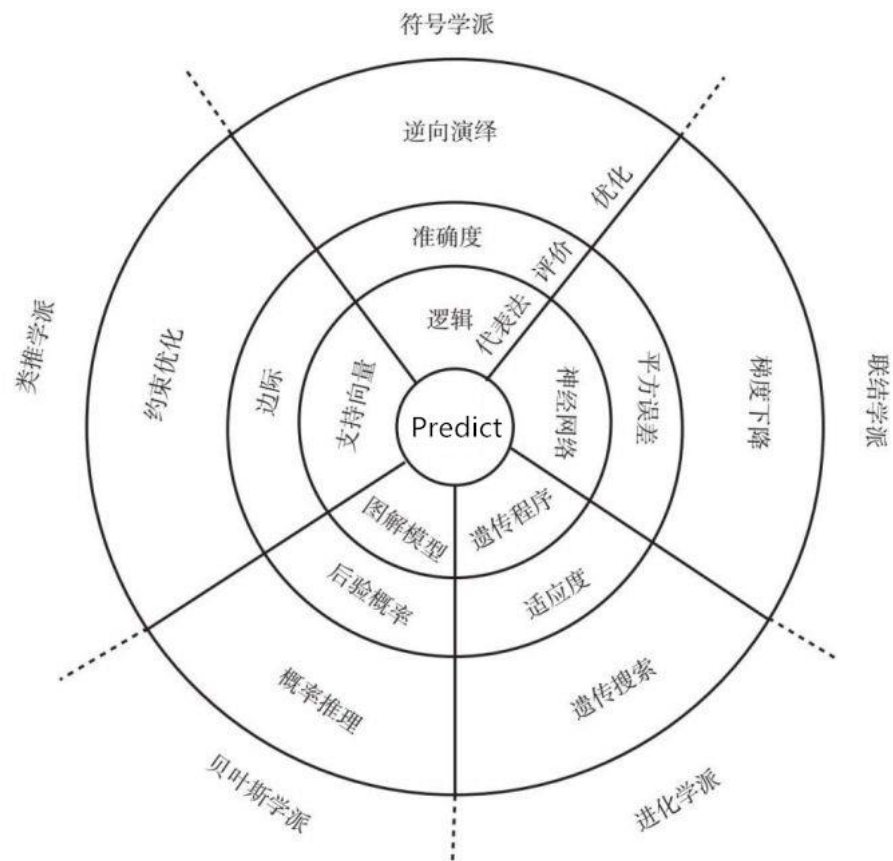
- 形式简单，模型的可解释性非常好
- 根据特征的权重可以得到不同的特征对最后结果的影响（某个特征的权重值高 => 这个特征对结果的影响大）
- 工程上的baseline，如果特征工程做的好，效果不会差
- 训练速度较快，计算量只和特征的数目相关
- 模型资源占用小，只需要存储各个维度的特征值
- 方便输出结果调整。逻辑回归可以很方便的得到最后的分类结果，因为输出的是每个样本的概率分数，我们可以很容易的对这些概率分数进行cut off，也就是划分阈值(大于某个阈值的是一类，小于某个阈值的是一类)

缺点:

- 准确率不是很高，形式简单，很难拟合数据的真实分布
- 很难处理样本不均衡问题
- 很难处理非线性数据，在不引入其他方法的情况下，只能处理线性可分的数据
- 逻辑回归本身无法筛选特征，可以采用gbdt来筛选特征，然后再用逻辑回归

Summary

- 机器学习包括监督学习，无监督学习，半监督学习，强化学习
- 符号学派，认为事情都是有因果的，机器可以自己摸索出规律，典型代表为决策树
- 贝叶斯学派，因果之间不是必然发生，是有一定概率的，即 $P(A|B)$ ，典型代表为朴素贝叶斯
- 类推学派，通过类比可以让我们学习到很多未知的知识，所以我们需要先定义“相似度”，通过相似度进行发现
- 联结学派，模仿人脑神经元的工作原理，所有模式识别和记忆建立在神经元的不同连接方式上，典型代表为神经网络，深度学习
- 进化学派，上帝通过基因选择来适者生存，典型代表为遗传算法



AI大赛：二手车价格预测

Project：二手车交易价格预测

• 数据集：

used_car_train_20200313.csv

used_car_testB_20200313.csv

数据来自某交易平台的二手车交易记录

<https://tianchi.aliyun.com/competition/entrance/231784/introduction>

ToDo：给你一辆车的各个属性（除了price字段），预测它的价格

Field	Description
SaleID	交易ID，唯一编码
name	汽车交易名称，已脱敏
regDate	汽车注册日期，例如20160101，2016年01月01日
model	车型编码，已脱敏
brand	汽车品牌，已脱敏
bodyType	车身类型：豪华轿车：0，微型车：1，厢型车：2，大巴车：3，敞篷车：4，双门汽车：5，商务车：6，搅拌车：7
fuelType	燃油类型：汽油：0，柴油：1，液化石油气：2，天然气：3，混合动力：4，其他：5，电动：6
gearbox	变速箱：手动：0，自动：1
power	发动机功率：范围[0, 600]
kilometer	汽车已行驶公里，单位万km
notRepairedDamage	汽车有尚未修复的损坏：是：0，否：1
regionCode	地区编码，已脱敏
seller	销售方：个体：0，非个体：1
offerType	报价类型：提供：0，请求：1
creatDate	汽车上线时间，即开始售卖时间
price	二手车交易价格（预测目标）
v系列特征	匿名特征，包含v0-14在内15个匿名特征

Project: 二手车交易价格预测

- 评价标准MAE(Mean Absolute Error):

若真实值为 $y = (y_1, y_2, \dots, y_n)$, 模型的预测值为 $\hat{y} = (\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n)$, 那么该模型的MAE计算公式为

$$MAE = \frac{\sum_{i=1}^n |y_i - \hat{y}_i|}{n}.$$

例如, 真实值 $y = (15, 20, 12)$, 预测值 $\hat{y} = (17, 24, 9)$, 那么这个预测结果的MAE为

$$MAE = \frac{|15 - 17| + |20 - 24| + |12 - 9|}{3} = 3.$$

MAE是L1 loss, MAE越小模型越准确

- 提交结果, 与sample_submit.csv中的格式一致
- | SaleID | price |
|--------|-------|
| 150000 | 687 |
| 150001 | 1250 |
| 150002 | 2580 |
| 150003 | 1178 |

分类与回归的区别

分类与回归是监督学习中的两大核心任务，二者既有联系又有区别

- 输出类型

分类：预测离散类别标签（如“男/女”、“是/否”）。

回归：预测连续数值（如房价、资产金额）。

- 目标函数

分类常用交叉熵损失（Cross-Entropy），回归常用均方误差（MSE）。

分类与回归的区别

Thinking: 分类与回归的内在联系是什么？

- 概率视角

分类的输出可视为类别的概率（如逻辑回归），而回归可通过阈值处理转化为分类（如预测概率 >0.5 为一类）。

- 模型共享

某些模型（如朴素贝叶斯、决策树、SVM、逻辑回归、神经网络）通过调整输出层即可切换任务：

分类：Softmax输出离散概率。

回归：线性输出连续值。

- 问题转化

回归问题可离散化为分类（如年龄预测 \rightarrow 年龄段分类）。

某些分类任务（如有序分类）可视为回归的离散特例。

有序分类的类别具有明确的顺序关系（如“差/中/好”、“1~5星评分”），类别标签不仅是符号，还隐含了连续的潜在变量。

分类与回归的区别

Thinking: 分类与回归的本质共性是什么?

均为从输入数据中学习映射关系 $f(X) \rightarrow Y$, 核心目标是最小化预测与真实值的差异。

分类与回归是监督学习的一体两面, 区别在于输出空间的性质, 但二者共享模型框架和优化思想。
实际应用中常相互转化或结合 (如回归后分类)。

Project: 二手车交易价格预测

- Step1, 数据加载

原始数据是用空格分隔

- Step2, 数据探索

数据整体情况

查看缺失值, 缺失值可视化

查看label的分布 (该项目中price为label)

- Step3, 特征选择

- Step4, 模型训练

使用XGBoost, 超参数设置

祖传参数:

```
model = xgb.XGBRegressor(  
    n_estimator=2000, learning_rate=0.1,  
    subsample=0.8, colsample_bytree=0.8,  
    min_child_samples=3, eval_metric='auc',  
    reg_lambda=0.5,  
    max_depth=15, tree_method='gpu_hist' )
```

- Step5, 模型预测

Project: 二手车交易价格预测

Thinking: 有哪些提分（减少MAE）的方式？

1、时间类型数据 (regDate, creatDate)

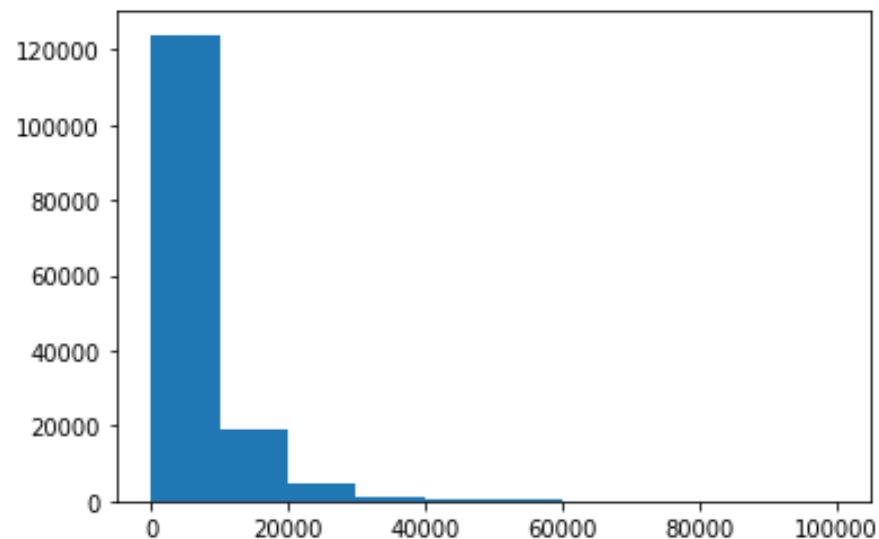
2、模型融合, 使用XGBoost, LightGBM

3、增加新的特征, 对brand进行统计

brand_amount, brand_price_max, brand_price_min,
brand_price_median, brand_price_sum, brand_price_std,
brand_price_mean

单模型XGBoost, MAE=?

XGBoost + LightGBM, MAE=?



预测的price分布情况

Project: 二手车交易价格预测

编写Python代码，打开 `used_car_train_20200313.csv`，查看数据字段及前5行数据

我会帮你编写代码来查看二手车训练数据。首先让我检查一下文件是否存在。

> Listed 5 items in current directory

很好，我看到训练数据文件 `used_car_train_20200313.csv` 已经存在。我将创建一个新的Python文件来查看数据。

```
view_data.py +21 -1 ✓  
# -*- coding: utf-8 -*-  
"""  
查看二手车训练数据的基本信息  
"""  
  
import pandas as pd  
  
# 读取训练数据  
train_data = pd.read_csv('used_car_train_20200313.csv')  
  
# 显示数据基本信息  
print("\n数据基本信息：")  
print(train_data.info())
```

Project: 二手车交易价格预测

@view_data.py 帮我做EDA

我会帮你对二手车数据进行探索性数据分析(EDA)。我将创建一个更详细的EDA分析脚本，包括数据概览、缺失值分析、数值型特征分布、类别型特征分布等。

```
view_data.py +81 -9 ✓

# -*- coding: utf-8 -*-
"""
查看二手车训练数据的基本信息
二手车数据探索性分析(EDA)
"""

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
plt.rcParams['font.sans-serif'] = ['SimHei'] # 用
plt.rcParams['axes.unicode_minus'] = False # 用来
```

我已经创建了一个全面的EDA分析脚本，这个脚本将进行以下分析：

1. 数据概览

- 显示数据基本信息（数据类型、非空值数量等）
- 显示数据统计描述（均值、标准差、分位数等）

2. 缺失值分析

- 统计每个特征的缺失值数量和比例
- 只显示存在缺失值的特征

3. 数值型特征分布分析

- 识别所有数值型特征
- 绘制价格分布图

4. 类别型特征分析

- 识别所有类别型特征
- 绘制Top 10品牌分布图

5. 相关性分析

- 计算数值型特征之间的相关性
- 绘制相关性热力图

Project: 二手车交易价格预测

我需要对 price进行建模，使用 XGBoost

这里需要注意：

对日期变量regDate 和 creatDate进行处理，

目前 regDate和creatDate的格式类似 20040402

===

编写新的Python

我会帮你创建一个新的Python文件，使用XGBoost对二手车价格进行建模，并处理日期特征。

```
train_xgboost.py +160 -1 ✓  
# -*- coding: utf-8 -*-  
"""  
二手车价格预测 - XGBoost模型  
"""  
  
import pandas as pd  
import numpy as np  
from sklearn.model_selection import train_test_sp  
from sklearn.preprocessing import LabelEncoder  
import xgboost as xgb  
from sklearn.metrics import mean_squared_error, r  
import matplotlib.pyplot as plt  
from datetime import datetime  
import seaborn as sns
```

Project: 二手车交易价格预测

- 1、评估中增加 MAE指标的评估
 - 2、使用 测试集 used_car_testB_20200421.csv, 对数据进行预测
- 预测结果写入到 used_car_sample_submit.csv, 按照这个格式, 即SaleID, price
- 将price字段写入。最终生成新的 submit结果文件

我会帮你修改代码, 增加MAE评估指标并添加测试集预测功能:

train_xgboost.py +74 -28 ✓

```
from sklearn.model_selection import train_test_sp
from sklearn.preprocessing import LabelEncoder
import xgboost as xgb
from sklearn.metrics import mean_squared_error, r
from sklearn.metrics import mean_squared_error, r
import matplotlib.pyplot as plt
from datetime import datetime
import seaborn as sns

12 hidden lines
    day = int(date_str[6:8])
    return pd.Series([year, month, day], index=['

def load_and_process_data():
def process_data(data, label_encoders=None, is tr
```

Project: 二手车交易价格预测

查看这些分类特征的 唯一值的个数

我会修改代码来查看每个分类特征的唯一值个数：

```
train_xgboost.py +38 ✓  
  
def analyze_categorical_features(data):  
    """  
    分析分类特征的唯一值个数  
    """  
    categorical_features = ['name', 'model', 'brand', 'type', 'color', 'mileage', 'year', 'price']  
  
    print("\n分类特征分析：")  
    print("-" * 50)  
    for feature in categorical_features:  
        if feature in data.columns:  
            unique_values = data[feature].nunique()  
            print(f"{feature}: {unique_values} 个")  
            if unique_values < 10: # 如果唯一值较少  
                print(f"具体值: {sorted(data[feature].unique())}")
```

Project: 二手车交易价格预测

我在训练 xgboost 的时候，现在验证结果是 rmse，想改成 mae

[960] validation_0-rmse:551.36867 validation_1-rmse:1277.40573

我会修改XGBoost模型的评估指标从RMSE改为MAE。需要修改两个地方：模型参数中的 `eval_metric` 和评估函数中的指标计算。

train_xgboost.py +1

```
if __name__ == "__main__":  
    main()
```

我已经修改了XGBoost模型的评估指标，主要做了以下更改：

1. 在模型参数中，将 `eval_metric` 从默认的RMSE改为'mae'：

python

▷ Apply to predict_catb...

```
params = {  
    ...  
    'eval_metric': 'mae' # 修改为MAE评估指标  
}
```

Project: 二手车交易价格预测

将训练前的数据进行保存，方便后续直接使用python

就是两个文件，即数据预处理的python，和模型训练和预测的python

我会将代码拆分为两个文件：一个用于数据预处理和保存，另一个用于模型训练和预测。

首先，创建数据预处理文件：

```
data_preprocessing.py +158 -1 ✓  
# -*- coding: utf-8 -*-  
"""  
二手车数据预处理  
"""  
  
import pandas as pd  
import numpy as np  
from sklearn.model_selection import train_test_split  
from sklearn.preprocessing import LabelEncoder  
import joblib  
import os  
  
def process_date(date_str):
```

现在，创建模型训练和预测文件：

```
train_xgboost.py +24 -120 ✓  
  
import pandas as pd  
import numpy as np  
from sklearn.model_selection import train_test_split  
from sklearn.preprocessing import LabelEncoder  
import xgboost as xgb  
from sklearn.metrics import mean_squared_error, r2_score  
import matplotlib.pyplot as plt  
from datetime import datetime  
import seaborn as sns  
import joblib  
  
# 设置中文显示
```

我已经将代码拆分为两个文件：

Project: 二手车交易价格预测

我想进一步优化模型，有一些思路

1、当前日期类型数据采用的是 原数据 =>year, month, day
是否增加一个 日期diff，即基于base_date的日期diff

2、使用XGBoost进行训练，目前 learning_rate=0.1 是否要改成
learning_rate = 0.01

另外n_estimators 可以增加，我发现 验证集的mae 和实际提交结果的mae很接近。实际提交结果是548，在验证集的mae是546

===

帮我分析 这些思路是否可行

我来分析一下您提出的优化思路：

1. 关于日期特征的优化：

- 当前的日期处理方式是将日期拆分为年、月、日，这种方式虽然保留了时间的周期性特征，但确实没有很好地体现时间的连续性和相对关系。
- 添加基于base_date的日期差值是一个很好的想法，因为：
 - a. 可以直接反映车辆的"市场年龄"
 - b. 相比分开的年月日，日期差值更能体现时间的连续性
 - c. 可以更好地捕捉时间对价格的线性影响
- 建议实现方式：
 - 选择一个基准日期（如数据集中最早的日期或固定的某个日期）
 - 计算每条记录的日期与基准日期的天数差
 - 保留原有的年月日特征（因为可能存在季节性影响）

2. 关于XGBoost参数调整：

- 当前参数： `learning_rate=0.1, n_estimators=1000`
- 验证集MAE(546)和实际提交结果MAE(548)非常接近，这说明：

Project: 二手车交易价格预测

@train_xgboost.py 参考这个，帮我编写一个 lightgbm训练和预测的代码，新的Python

我来帮你编写一个使用LightGBM的训练和预测代码。
LightGBM通常在处理大规模数据时比XGBoost更快，同时保持着相似的性能。

```
train_lightgbm.py +174 -1 ✓  
# -*- coding: utf-8 -*-  
"""  
二手车价格预测 - LightGBM模型  
"""  
  
import pandas as pd  
import numpy as np  
import lightgbm as lgb  
from sklearn.metrics import mean_squared_error, r  
import matplotlib.pyplot as plt  
import seaborn as sns  
import joblib  
  
# 设置中文显示
```

Project: 二手车交易价格预测

@train_xgboost.py 参考这个，帮我编写一个 catboost训练和预测的代码，新python文件

我来帮你创建一个使用CatBoost进行二手车价格预测的代码。我会参考XGBoost的实现，并针对CatBoost的特点进行优化。

```
train_catboost.py +167 -1 ✓  
# -*- coding: utf-8 -*-  
"""  
二手车价格预测 - CatBoost模型  
"""  
  
import pandas as pd  
import numpy as np  
from catboost import CatBoostRegressor  
from sklearn.metrics import mean_squared_error, r  
import matplotlib.pyplot as plt  
import seaborn as sns  
import joblib  
  
# 设置中文显示
```

Project: 二手车交易价格预测

都有哪些特征工程的思路，可以筛选一些重要的，然后使用CatBoost进行建模和预测，给我新的Python代码

我来为你创建一个结合高效特征工程和CatBoost建模的完整解决方案。我会选择最重要的特征工程思路，并实现完整的代码。

```
...e_engineering_and_catboost.py +426 -1 ✓ ...  〰  〰  
# -*- coding: utf-8 -*-  
"""  
二手车价格预测 - 高级特征工程与CatBoost建模  
"""  
  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
from catboost import CatBoostRegressor, Pool  
from sklearn.metrics import mean_squared_error, m  
from sklearn.model_selection import train_test_sp  
import joblib  
import datetime
```

Project: 二手车交易价格预测

刚才的特征工程，帮我总结到 特征工程.md 中

特征工程.md > # 二手车价格预测特征工程总结

```
1  # 二手车价格预测特征工程总结
2
3  ## 1. 时间特征
4
5  时间特征是二手车价格预测中的关键因素，我们对原始的日期数据进行了以下处理：
6
7  ### 1.1 基础时间特征
8  - **车辆年龄**：计算车辆从注册到创建记录的天数和年数
9  - **注册日期分解**：将注册日期拆分为年、月、日三个独立特征
10 - **创建日期分解**：将创建日期拆分为年、月、日三个独立特征
11
12 ### 1.2 高级时间特征
13 - **车龄分段**：将车龄划分为不同区间（0-1年、1-3年、3-5年、5-10年、10年以上）
14 - **季节特征**：从月份计算季度信息
15 - **是否为新车**：根据车龄判断是否为新车（使用年限<1年）
16 - **每年行驶公里数**：公里数除以使用年限
17
18 ## 2. 车辆特征
19
20 ### 2.1 缺失值处理
21 - 为每个数值型特征创建缺失标记（_missing）
22 - 使用中位数填充缺失值
23
24 ### 2.2 特征交互
25 - **功率与排量比**：衡量车辆性能指标
```



Thank You
Using data to solve problems

