机器学习简单流程：

1. 使用大量和任务相关的数据集来训练模型
2. 通过模型在数据集上的误差不断迭代训练模型，得到对数据集拟合合理的模型
3. 将训练好调整好的模型应用到真实的场景中

我们把模型在真实环境中的误差叫做泛化误差，最终的目的是希望训练好的模型泛化误差越低越好

我们希望通过某个信号来了解模型的泛化误差，这样就可以指导我们得到泛化能力更强的模型：

1. 使用泛化误差本身。如果泛化误差小还可以接受，但通常情况下没有那么幸运，泛化误差可能很大，这个时候你肯定会将部署的模型撤回重新训练，可能需要部署和训练之间往复很多次，这种方式虽然能够很好的指导我们的模型，但成本和效率非常差
2. 使用模型在数据集上训练的拟合程度来作为评估模型的信号，但是往往我们获得的数据集并不是完全的干净以及有代表性，通常我们获取到的数据集可能很少，数据的代表性不够，包含太多的噪声或者一些无关特征污染，我们获取到的数据集或多或少都会有这些问题，那么模型对训练数据集的拟合程度不能指导泛化误差，也就是说训练的时候拟合的好并不代表模型的泛化误差就小。

我们既不能通过直接将泛化误差作为了解模型泛化能力的信号，因为在部署环境和训练模型之间往复代价很高，也不能使用模型对训练数据集的拟合程度来作为了解模型泛化能力的信号，因为我们获得的数据往往不干净。

更好的方式就是将数据分割成两部分：训练集和测试集。我们可以使用训练集的数据来训练模型，然后用测试集上的误差作为最终模型在对应现实场景中的泛化误差。有了测试集，我们想要验证模型的最终效果，只需将训练好的模型在测试集上计算误差，即可认为此误差为泛化误差的近似，只需要让训练好的模型在测试集上的误差最小即可。

1. 通常将数据集的80%作为训练集，20%作为测试集
2. 通常需要在开始构建模型之前把数据集进行划分，防止数据窥探偏误，也就是说我们避免了解太多关于测试集中的样本特点，防止我们认为的挑选有助于测试集数据的模型
3. 通常我们在构建模型的时候需要将数据进行处理，包括一些数据的清洗，数据的特征缩放（标准化或者归一化），此时我们只需要在训练集上进行这些操作，然后将其在训练集上得到的参数应用到测试集中，也就是说，在工作流程中，不能使用在测试集数据上计算的得到任何结果。比如：我们得到的属性中可能有缺失值，因为在这些操作之前，我们已经把数据集分成了训练集和测试集，通常的做法是通过计算属性值的中位数来填充缺失值，此时计算属性的中位数是通过训练集上的数据进行计算的。如果想要测试模型的测试误差来近似泛化误差的时候，可能此时的测试集也会有一些缺失值，此时对应属性的缺失值是通过训练集计算的中位数进行填充的
4. 由于测试集作为对泛化误差的近视，所以训练好模型，最后在测试集上近似估计模型的泛化能力。此时假设有两个不同的机器学习模型，可以通过训练两个模型，然后对比他们在测试数据上的泛化误差，选择泛化能力强的模型

验证集

我们将数据集划分为训练集和测试集，我们让模型在训练集上进行训练，然后在测试集上来近似模型的泛化能力。如果想要挑选不同的模型的话，可以让两个模型分别在训练集上训练，然后将两个训练好的模型分别在训练集上进行测试，由于我们把测试集上的误差近似为泛化误差，所以我们自然可以选择在测试集上误差小的模型作为最终要选择的泛化能力强的模型。

但我们要做的不仅是不同的模型与模型之间的对比，很多时候我们需要对模型本身进行选择，假如我们有两个模型，线性模型和神经网络模型，神经网络的泛化能力要比线性模型要强，但是神经网络中还有很多需要人工选择的参数，比如神经网络的层数和每层神经网络的神经元个数以及正则化的一些参数等，我们将这些参数称为超参数。这些参数不同选择对模型最终的效果也很重要，我们在开发模型时需要调节这些超参数。

我们需要调节这些超参数来使得模型泛化能力最强。我们使用测试集来作为泛化误差，而我们最终的目的就是选择泛化能力强的模型，那么我们可以直接通过模型在测试集上的误差来调节这些参数不就可以了。可能模型在测试集上的误差为0，但是拿着这样的模型去部署到真实场景中使用的话，效果可能非常差。

这一现象叫做信息泄露。我们使用测试集作为泛化误差的近似，所以不到最后是不能将测试集的信息泄露出去的，好比考试一样，我们平时做的题相当于训练集，测试集相当于最终的考试，我们通过最终的考试来检验我们最终的学习能力，将测试集信息泄露出去，相当于学生提前知道了考试题目，那最后再考这些提前知道的考试题目当然代表不了什么。所以，我们在学习的时候，老师会准备一些小测试来帮助我们查缺补漏，这些小测试也就是要说的验证集。我们通过验证集来作为调整模型的依据，这样不至于将测试集中的信息泄露。

在训练集上训练模型，在验证集上评估模型，一旦找到最佳的参数，就在测试集上最后测试一次，测试集上的误差作为泛化误差的近似。

当数据量不是很大的时候(万级别以下)的时候将训练集、验证集以及测试集划分为6:2:2

若是数据很大，可以将比例调整为98:1:1

当可用数据很少的情况下可以使用留出法，K折交叉验证等

分类任务的常用评价指标

1. 准确率Accuracy

分类问题中最基本的评价指标，正确分类样本所占的比例，即正确分类的测试样本/全部测试样本

优点：计算简单，易于理解，既可以用于二分类也可以用于多分类

缺点：当数据不均衡时，无法很好的衡量分类器的好坏

1. 精确率Precision

只能用于二分类（多分类时，只能两两计算，然后求平均）

1. 召回率Recall

只能用于二分类（多分类时，只能两两计算，然后求平均）

1. Recall和Precision的应用场景

对于地震的预测，我们希望的是召回率非常高，也就是说每次地震我们都希望预测出来，这时候可以牺牲精确率。情愿发出1000次警报，把10次地震都预测正确了，也不要预测100次对了8次漏了两次。

嫌疑人定罪基于不错怪一个好人的原则，对于嫌疑人的定罪我们希望是非常准确的，即使有时放过了一些罪犯（recall低），但也是值得的。

1. F1值（F1-score）

我们期望当时是精确率和召回率都高，但现实往往不能兼得，所以搞了结合它两的东西，这个值越大越好

1. P-R曲线

召回率为横轴，精确率为纵轴

现实任务中的P-R曲线是非单调、不平滑的，在很多局部有上下波动

1. ROC曲线

接受者操作特征Receiver Operating Characteristic

在众多的机器学习模型中，很多模型输出的是预测概率，而使用精确率、召回率这类指标进行模型评估时，还需要对预测概率设分类阈值，比如预测概率大于阈值为正例，反之为负例。这使得模型多了一个超参数，并且这超参数会影响模型的泛化能力。ROC曲线不需要这样一个设定阈值。

ROC曲线纵坐标是真正率，横坐标是假正率

真正率True Positive Rate：recall

假正率False Positive Rate：分母是所有的负类样本，预测时，包括预测成负的+预测成正的，分子就是预测是把本身是负类的样本预测成了正类

ROC的计算方法同时考虑了学习器对与正例和负例的分类能力，在样本不平衡的情况下，依然能对分类器作出合理的评价。ROC对样本类别是否均衡并不敏感，这也是不均衡样本常用ROC评价学习器性能的一个原因。

例如在癌症预测的场景中，假设没有患癌症的样本为正例，患癌症样本为负例，负例占比很少(大概0.1%),如果使用准确率评估，把所有的样本预测为正例便可以获得99.9%的准确率，但如果使用ROC，把所有样本预测为正例，TPR=1，FPR=1，这种情况下学习器的ROC值等于0.5，成功规避了样本不均衡带来的问题。

ROC对样本类别是否均衡并不敏感原因：计算TPR和FPR是计算条件概率，以真实Y=0或者Y=1为条件，所有跟Y的分布无关

ROC曲线和PR曲线有些类似：ROC曲线越靠近左上角性能越好，左上角坐标为(0,1),即FPR=0，TPR=1，根据FPR和TPR公式可知，此时FN=0，FP=0，模型对所有样本分类正确，绘制ROC曲线，首先对所有样本按预测概率排序，预测概率指的是分类器一般不会直接给出类别，而是给出属于某个类的概率，然后需要手动设定阈值来确定大于多少概率属于哪一类，在这里的意思就是不预先设定阈值，而是拿每一个样本的概率都当作一边阈值来求FPR和TPR，以每个样本的预测概率为阈值，计算相应的FPR和TPR，然后线段连接

当数据量少时，绘制的ROC曲线不平滑，当数据量大时，绘制的ROC曲线会趋于平滑

ROC曲线的作用：

能很容易的查出任意阈值对学习器的泛化性能影响

有助于选择最佳的阈值，ROC曲线越靠近左上角，模型的查全率就越高，最靠近左上角的ROC曲线上的点是分类错误最少的最好的阈值，其假正例和假反例总数最少

可以对不同的学习器比较性能，将各个学习器的ROC曲线绘制到同一坐标中，直观地鉴别优劣，靠近左上角的ROC曲所代表的学习器准确性最高

ROC的优点：

对数据平不平衡不敏感，可用来衡量不平衡数据

该方法简单、直观、通过图示可观察分析方法的准确性，并可用肉眼作出判断。ROC曲线将真正例率和假正例率以图示方法结合在一起，可准确反映出某种学习器真正例率和假正例率的关系，是检测准确性的综合代表

1. AUC（Area under curve）

只能用于二分类模型的评价

AUC就是ROC曲线下的面积，衡量学习器优劣的一种性能指标

AUC和logloss比accuracy更常用，因为很多机器学习的模型对分类问题的预测结果都是概率，如果要计算accuracy，需要先把概率转化成类别,这就需要手动设置一个阈值，如果对一个样本的预测概率高于这个预测，就把这个样本放进一个类别里面，低于这个阈值，放进另一个类别里面。所以这个阈值很大程度上影响了accuracy的计算。使用AUC或者logloss可用避免把预测概率转换成类别

1. 宏平均Macro-averaging和微平均Micro-averaging

用途：用于多个类别的分类

宏平均：把每个类别都当作正类计算precision,recall,f1然后求平均

微平均：把每一个类别都当作正类，微平均不再计算每一个类别的precision,recall,f1,而是要计算出需要计算precision,recall,f1的值，即TP，FP，TN，FN，然后加起来求平均，然后用这平均值来计算precision,recall,f1

1. 混淆矩阵

在每个类别下，模型预测错误的结果数量，以及错误预测的类别和正确预测的数量都在一个矩阵下面显示出来

多分类评价指标

1. 多分类转化为2VS2问题来评价

准确率：与二分类相同，预测正确的样本占总样本的比例。

精确率： ‘macro’， 对于每个标签，分别计算Precision，然后取不加权平均

查全率： ‘macro’，对于每个标签，分别计算Recall，然后取不加权平均

F1-Score：‘macro’， 对于每个标签，分别计算发，然后取不加权平均

‘micro’, 将n个二分类评价的TP,FP,FN对应相加，计算P和R，然后求得F1

一般macro-f1和micro-f1都高的分类器性能好

1. 直接定义的多分类指标
   1. Kappa系数

在统计学中评估一致性的一种方法，取值范围是[-1,1],实际应用中，一般是[0,1]

P0表示总体分类精度

Pe表示SUM（第i类真实样本数\*第i类预测出来的样本数）/样本总数平方

from sklearn.metrics import cohen\_kappa\_score

Kappa = cohen\_kappa\_score(y\_true,y\_pred,label=None)

* 1. 海明距离

衡量预测标签与真实标签的距离，取值在0-1之间，距离为0说明预测结果与真实结果完全相同，距离为1说明模型与我们想要的结果完全背道而驰

from sklearn.metrics import hamming\_loss

ham\_distance = hamming\_loss(y\_true,y\_pred)

* 1. 杰卡德相似系数

它与海明距离的不同之处在于分母。当预测结果与实际情况完全相符时，系数为1；当预测结果与实际情况完全不符时，系数为0；当预测结果是实际情况的真子集或真超集时，距离介于0到1之间。

我们可以通过对所有样本的预测情况求平均得到算法在测试集上的总体表现情况。

from sklearn.metrics import jaccard\_similarity\_score

jaccrd\_score = jaccrd\_similarity\_score(y\_true,y\_pred,normalize = default)

#normalize默认为true，这是计算的是多个类别的相似系数的平均值，normalize = false时分别计算各个类别的相似系数

* 1. 铰链损失

铰链损失（Hinge loss）一般用来使“边缘最大化”（maximal margin）。损失取值在0~1之间，当取值为0，表示多分类模型分类完全准确，取值为1表明完全不起作用。

from sklearn.metrics import hinge\_loss

hinger = hinger\_loss(y\_true,y\_pred)

选择传统机器学习还是深度学习的标准是什么？

于数据挖掘和处理类的问题，使用一般的机器学习方法，需要提前做大量的特征工程工作，而且特征工程的好坏会在很大程度上决定最后效果的优劣，也就是说数据和特征决定了机器学习的上限，而模型和算法只是逼近这个上限而已

使用深度学习的话，特征工程就没那么重要了，特征只需要做些预处理就可以，因为它可以自动完成传统机器学习算法中需要特征工程才能实现的任务，特别是在图像和声音数据的处理中更是如此，但模型结构会比较复杂，训练较为麻烦，虽然深度学习让我们可以省去特征工程这一较为繁琐的过程，但也让我们失去了对特征的认识，如特征的重要性等

如何选择或衡量这两种方法：

第一是看数据量，比如训练数据量达到百万以上，深度学习的方法会比较有优势，如果样本集不是大样本，那么特征工程加传统的机器学习方法使用起来泛化能力会更好

第二看是否需要对结果有较强的解释性和可调节性，解释性是说我们能够了解到产出该输出结果的原因，这样我们就能够知道特征的重要程度，并在出错时能够对错误原因进行分析。可调节性是指在出错或有特征的增删时，能够方便的对原模型进行修正以满足新的要求。在这一方面，一般的机器学习方法有一定的优势。

各自的优势领域：

深度学习：图像处理、自然语言处理等

机器学习：金融风控、量化分析、推荐系统、广告预测等因为需要较好的可解释性

深拷贝和浅拷贝的区别

1. 特征分解

矩阵A的特征向量是指一个经过与矩阵A变换后方向保持不变的向量，而特征值为在这个变化中特征向量的比例因子

对于一个3×3维的矩阵A，我们可以将矩阵A与其特征向量x的变换理解为将矩阵A与另一个矩阵x的乘积

令

如果一个n×n维矩阵的n个特征向量均是线性无关的，则这个矩阵能够被对角化

一个n×n维的矩阵可以由三个独立的矩阵构成，即一个由特征向量组成的n×n维的矩阵X和矩阵X的逆，以及一个由特征值组成的n×n维的对角矩阵Λ。而这个过程也被称为矩阵的特征分解。

1. 奇异值分解

特征分解适用于n×n维的方形矩阵，而由于m×n维的矩形矩阵在变换过程中会改变矩阵原本的维数，从而对于矩形矩阵并没有对其特征值进行过定义。因此对于一个m×n维的矩形矩阵，我们能够使用下面的方法对其进行特征分解——即奇异值分解。

其中，矩阵U和V是正交矩阵，Σ表示一个包含有奇异值的对角阵

1. 主成分分析

PCA在机器学习中是一种常用的无监督学习算法，它通过构建一种被称为主成分的变量，并将所用到的所有向量映射到由主成分变量构建的空间上去，从而能够减少数据的维度。

给定一个集合：

在的非空子集定义一个集函数

是中所有元素的中心，即

表示集合中元素的个数

对于集合的一个给定划分，可以得到目标函数为

其中是所在类的中心

在K-均值聚类算法中，任选k个数据点作为初始聚类中心，计算每个数据到k个中心的距离，并将数据划分到最近距离所在的类，这样就得到了集合的一种划分方式，重新计算每一类的中心位置，这样就得到了目标函数的一个取值，即

是元素所在类的中心，即中的某一个

计算每个数据到这k个中心的距离，并将数据划分到最近距离所在的类，计算

由于是k个中心的一个，而是k个中心中距离最近的一个，所以

计算新划分得到的每一类中心位置，从而得到目标函数的一个新值，目标函数值不会增加，当与完全重合时，目标函数值不变。

只需证明

是新划分的每一类的中心位置，由于左右两边对集合的划分方式是一样的，只是左边是每类的中心，而右边不是。证明转化为：

设是欧式空间的n个向量，则取到最小值当且仅当x是这n个向量的中心位置即

该目标函数严格凸的，所以是目标函数的唯一最小值点，说明只要上一步骤中有某个中心位置发生了变化，那么目标函数值严格较小。也即经过一次迭代后，目标函数不增。

1. Best Subset

选择线性回归变量子集的直接方法是尝试所有可能的组合，并选择一个最小化某些标准的组合。对于每一个,其中p是可用特征的总数，它选择大小的子集k给出最小的均方误差。由于平方和必然随着k减小,对于预测模型，测试数据上的均方误差是常见的选择。

1. Ridge Regression
2. Lasso

缩减方法lasso对回归系数的约束条件：

在足够小的时候，一些系数会缩减到0。L1正则化的约束条件求解，可以使用二次规划算法或局部二次近似算法求解。由于需要模拟参数求解过程，以下采用局部二次近似算法进行模拟。

1. Principal Components Regression
2. Partial Least Squares