PHP opcode opcode是php语言里供zend引擎执行的一种中间代码，类似java中的字节码、或者python中的字节码对象pycodeobject.

PHP VLD扩展：

-dvld.active=1 是 VLD 的基础参数，表示激活 VLD 模式。

-dvld.execute 是否执行这段PHP脚本，默认值为1，表示执行。可以使用-dvld.execute=0，表示只显示中间代码，不执行生成的中间代码。

通过指定stderr=subprocess.STDOUT，将子程序的标准错误输出重定向到了标准输出，以使我们可以直接从标准输出中同时获取标准输出和标准错误的信息。

正则匹配：

tokens = re.findall(r'\s(\b[A-Z\_]+\b)\s', output)

+ 号代表前面的字符必须至少出现一次（1次或多次）。

\babc\b匹配的是"abc"

\sabc\s匹配的不是"abc", 前后还带空格" abc "

\b只是匹配字符串开头结尾及空格回车等的位置, 不会匹配空格符本身

subprocess模块：

subprocess模块subprocess通过子进程来执行外部指令，并通过input/output/error管道，获取子进程的执行的返回信息

subprocess.check\_output()：执行命令，并返回执行状态，，如果当返回值为0时，直接返回输出结果，如果返回值不为0，直接抛出异常。需要说明的是，

os.walk() 方法用于通过在目录树中游走输出在目录中的文件名

返回的是一个三元组(root,dirs,files)。

root 所指的是当前正在遍历的这个文件夹的本身的地址

dirs 是一个 list ，内容是该文件夹中所有的目录的名字(不包括子目录)

files 同样是 list , 内容是该文件夹中所有的文件(不包括子目录)

os.path.join()函数用于路径拼接文件路径。

sklearn是python的重要机器学习库，其中封装了大量的机器学习算法，如：分类、回归、降维以及聚类；还包含了监督学习、非监督学习、数据变换三大模块。sklearn拥有完善的文档，使得它具有了上手容易的优势；并它内置了大量的数据集，节省了获取和整理数据集的时间。

from sklearn.feature\_extraction.text import CountVectorizer, TfidfTransformer

词袋法：

不考虑词语出现的顺序，每个出现过的词汇单独作为一列特征

这些不重复的特征词汇集合为词表

每一个文本都可以在很长的词表上统计出一个很多列的特征向量

如果每个文本都出现的词汇，一般被标记为 停用词 不计入特征向量

CountVectorizer

CountVectorizer是属于常见的特征数值计算类，是一个文本特征提取方法。对于每一个训练文本，它只考虑每种词汇在该训练文本中出现的频率。

（作用是把一系列文档的集合转化成数值矩阵。）

CountVectorizer会将文本中的词语转换为词频矩阵，它通过fit\_transform函数计算各个词语出现的次数。

cv=CountVectorizer(ngram\_range=(3, 3), decode\_error="ignore", token\_pattern=r'\b\w+\b')

ngram\_range=(3, 3)，词组切分的长度范围，

decode\_error="ignore"，默认为strict，遇到不能解码的字符将报UnicodeDecodeError错误，设为ignore将会忽略解码错误

token\_pattern=r'\b\w+\b'， 过滤规则，表示token的正则表达式，

X = cv.fit\_transform(X).toarray()

cv.fit\_transform(X) 对数据进行某种统一处理（比如标准化~N(0,1)，将数据缩放(映射)到某个固定区间，归一化，正则化等）

fit\_transform(partData)对部分数据先拟合fit，找到该part的整体指标，如均值、方差、最大值最小值等等（根据具体转换的目的），然后对该partData进行转换transform，从而实现数据的标准化、归一化等等。

.toarray()转数组

transformer = TfidfTransformer(smooth\_idf=False)

TfidfTransformer 的作用是把数值矩阵规范化为 tf 或 tf-idf 。

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.4, random\_state=0)

train\_test\_split的作用是“随机”分配训练集和测试集。这里的随机不是每次都随机，在参数确定的时候，每次随机的结果都是相同的。有时，为了增加训练结果的有效性，我们会用到交叉验证（cross validations）。

train\_test\_split函数用于将矩阵随机划分为训练子集和测试子集，并返回划分好的训练集测试集样本和训练集测试集标签。

test\_size：如果是浮点数，在0-1之间，表示样本占比；如果是整数的话就是样本的数量

random\_state：是随机数的种子。

随机数种子：其实就是该组随机数的编号，在需要重复试验的时候，保证得到一组一样的随机数。比如你每次都填1，其他参数一样的情况下你得到的随机数组是一样的。但填0或不填，每次都会不一样。

随机数的产生取决于种子，随机数和种子之间的关系遵从以下两个规则：

种子不同，产生不同的随机数；种子相同，即使实例不同也产生相同的随机数。

GaussianNB ：Scikit-learn 对朴素贝叶斯算法的实现。朴素贝叶斯算法是常用的监督型算法。

TfidfTransformer和TfidfVectorizer

具体来说，TfidfTransformer的作用已经不是在特征提取了，而是特征加权。 而文本特征权重的计算方法有许多，scikit-learn貌似也只提供了TF-TDF这一种。

关于TfidfTransformer给出的例程尤其简单，这里就不进行讲述。

而TfidfVectorizer则是CountVectorizer和TfidfTransformer的结合版本。

而TfidfVectorizer则是把两者的功能合在一起，连参数也都是两者的参数合在一起，所以可以方便的直接使用TfidfVectorizer。但是如果想在CountVectorizer来提取特征后想处理特征，比如降维之类的，这样直接使用TfidfVectorizer就不行了。

from sklearn.externals import joblib

在训练模型后将模型保存的方法，以免下次重复训练。

使用pickle模块或者sklearn内部的joblib

一、使用pickle模块

from sklearn import svm

from sklearn import datasets

clf=svm.SVC()

iris=datasets.load\_iris()

X,y=iris.data,iris.target

clf.fit(X,y)

import pickle

s=pickle.dumps(clf)

f=open('svm.txt','w')

f.write(s)

f.close()

f2=open('svm.txt','r')

s2=f2.read()

clf2=pickle.loads(s2)

clf2.score(X,y)

二、使用joblib

joblib更适合大数据量的模型，且只能往硬盘存储，不能往字符串存储

from sklearn.externals import joblib

joblib.dump(clf,'filename.pkl')

clf=joblib.load('filename.pkl')