开发环境：

操作系统:Ubuntu16.04

编译环境：Pycharm

开发语言：python3.6

主要依赖包：Anaconda3.6

1. 操作系统Ubuntu16.04的安装

我这里是安装了win10和Ubuntu双系统，win10的安装就不说了，太简单。对于Ubuntu的安装也不难，首先分盘，我给Ubuntu分了500个G的磁盘空间，然后是系统U盘安装盘制作，使用的是Ultralso工具，接着是UEFI设置，最后启动时进入U盘引导界面，一步步设置，直至最后安装完成。

1. Pycharm的安装

现在Pycharm官网上下载Pycharm的安装包，选择Linux版本的，这里有两个版本，一个是Communication，另一个是Profession，即专业版和社区版，专业版功能更多，收费，社区版功能少一点，免费开源。

链接：<https://www.jetbrains.com/pycharm/download/#section=linux>

下载之后，解压，进入文件夹bin目录，在终端执行：sh pycharm.sh命令，即可打开pycharm软件

1. Anaconda3.6的安装

安装Anaconda主要解决各种安装包的依赖。

官方下载地址：<https://www.anaconda.com/download/>

下载对对应的Linux版本的：Anaconda3-4.2.0-Linux-x86\_64.sh

然后在终端执行：bash Anaconda3-4.2.0-Linux-x86\_64.sh 命令，按照提示操作就可以完成安装。安装好Anaconda之后，里面自带python，无需在另外安装python。

第一阶段程序：

分为三个包:Main,Utils,Model.

Main:测试程序用的，Utils是程序运行时所需要的各种工具方法，Model是主模型

第一部分主模型Model:

依赖包：

import numpy as np  
from sklearn.ensemble.base import BaseEnsemble  
from sklearn.base import ClassifierMixin, RegressorMixin, is\_classifier, is\_regressor  
from sklearn.externals import six  
from sklearn.externals.six.moves import zip  
from sklearn.externals.six.moves import xrange as range  
from sklearn.ensemble.forest import BaseForest  
from sklearn.tree.tree import BaseDecisionTree  
from sklearn.tree.\_tree import DTYPE  
from sklearn.metrics import accuracy\_score, r2\_score  
from Utils.validation import check\_array, check\_X\_y, check\_random\_state  
from Utils.validation import has\_fit\_parameter, check\_is\_fitted  
from sklearn.utils.extmath import stable\_cumsum

模型的基类：

class BaseWeightBoosting(six.with\_metaclass(ABCMeta, BaseEnsemble)):

模型继承类：

class BoostingAttackTreeClassifier(BaseWeightBoosting, ClassifierMixin):

模型目前的主要方法：

初始化方法：

def \_\_init\_\_(self,  
 base\_estimator,  
 n\_estimators=50,  
 learning\_rate=1.,  
 # algorithm='SAMME.R',  
 algorithm='SAMME',  
 random\_state=None):  
 super(BoostingAttackTreeClassifier, self).\_\_init\_\_(  
 base\_estimator=base\_estimator,  
 n\_estimators=n\_estimators,  
 learning\_rate=learning\_rate,  
 random\_state=random\_state)  
  
 self.algorithm=algorithm  
 self.base\_estimator=base\_estimator

训练方法：

def fit(self, X, y, sample\_weight=None):

提升方法：

def \_boost(self, iboost, X, y, sample\_weight, random\_state):

def \_boost\_real(self, iboost, X, y, sample\_weight, random\_state):

def \_boost\_discrete(self, iboost, X, y, sample\_weight, random\_state):

预测方法

def predict(self,X):

决策方法

def decision\_function(self, X):

第二部分：工具包Utils

依赖包：

import warnings  
import numbers  
  
import numpy as np  
import scipy.sparse as sp  
from scipy import \_\_version\_\_ as scipy\_version  
from distutils.version import LooseVersion  
  
from numpy.core.numeric import ComplexWarning  
  
from sklearn.externals import six  
from sklearn.utils.fixes import signature  
from sklearn import get\_config as \_get\_config  
from sklearn.exceptions import NonBLASDotWarning  
from sklearn.exceptions import NotFittedError  
from sklearn.exceptions import DataConversionWarning

工具方法：

def \_assert\_all\_finite(X, allow\_nan=False):

def as\_float\_array(X, copy=True, force\_all\_finite=True):

def \_is\_arraylike(x):

def \_num\_samples(x):

def \_ensure\_sparse\_format(spmatrix, accept\_sparse, dtype, copy,  
 force\_all\_finite, accept\_large\_sparse):

def check\_array(array, accept\_sparse=False, accept\_large\_sparse=True,  
 ensure\_min\_features=1, warn\_on\_dtype=False, estimator=None):

def \_check\_large\_sparse(X, accept\_large\_sparse=False):

def check\_X\_y(X, y, accept\_sparse=False, accept\_large\_sparse=True)

def check\_is\_fitted(estimator, attributes, msg=None, all\_or\_any=all):

def check\_non\_negative(X, whom):

…………..