# README

Mayu

**Part 1(Use machineLearning)**

**brief introduction**

A Method based on Oversampling and Ensemble Learning for Predicting the Number of Software Defects

**Abstract**

Predicting the number of the defects in software modules can help researchers make good use of the testing resource by paying more attention on the modules with more defects. Focusing on the issue that software defect dataset is imbalanced, a method SMOTERE based on oversampling and ensemble learning is proposed in this paper. First, get n balance datasets by oversampling original software defect dataset n times. Then train the n balance datasets using regression algorithms and get n weak prediction models. Finally, combine these models to be a strong prediction model and predict new software modules using the strong model. The result shows that SMOTERE improves the accuracy of prediction obviously.

**项目详细介绍**

**运行环境介绍**

本项目是在anaconda3的win10电脑上开发出来的，若想看到本项目的具体效果，请安装相对应的环境，采用的编辑器是目前交互性极好的jupyter，若想打开正确浏览里面的代码和注释，强烈建议使用jupyter，其他的编辑器打开可能会出现问题。

**本项目中使用的技术和库**

使用了python的一些常用库，包括主要用于科学计算的numpy和pandas,通过 IPython.display 和 matplotlib.pyplot，进行数据的可视化。根据我们样本的特点我们采用了sklearn中的BayesianRidge线性模型、DecisionTreeRegressor决策树回归模型和NearestNeighbors临近算法来训练我们的model。

首先是通过python实现对选定的datasets的遍历读入，之后定义了一个

**def seperateData(modules, Type):**  （数据集分析函数）

+ 输入数据集以及Type，

+ Type == -1时，返回数据集中的（1）缺陷模块数据集（2）缺陷模块的特征数据集（3）缺陷模块的bug数据集

+ Type == 0时，返回数据集中的（1）模块特征数据集（2）模块bug数据集

+ Type == 1时，返回数据集中的（1）正常模块数据集（2）正常模块的特征数据集

根据我们的datasets的数据，由于数据种类过多。导致单个种类的数据在总体数据中所占的比例较小，所以我们决定使用smote算法，来进行数据的采集效果会更加的好。

其中：

* Input:训练模块数据集，k邻近的值，从k邻近中任取n个值
* Output：经过SMOTE过采样后的新数据集，dataframe格式，21维，包含20维的特征值和1维的bug数量

## 决策树函数

def decisiontree(trainChar,trainBug, testChar):

* Input: 训练数据集的自变量，训练数据集的因变量，测试数据集的自变量
* Output: 测试数据集因变量，即预测出的bug数量，narray格式

## FPA计算函数

def FPA(testBug, testPre):

全称是（Function Point Analysis）.功能点分析方法是最重要也是最有效的软件测量规模方法。可以度量软件的规模是否合理

* Input: 测试数据集的 实际 bug， 测试数据集的 预测 bug
* Output: FPA值

## 十折交叉验证法函数

为了保证数据的正确性，我采用了十折交叉验证法函数。是常用的测试方法。将数据集分成十份，轮流将其中9份作为训练数据，1份作为测试数据，进行试验。 每次试验都会得出相应的正确率（或差错率）。10次的结果的正确率（或差错率）的平均值作为对算法精度的估计，一般还需要进行多次10折交叉验证（例如10次10折交叉验证），再求其均值，作为对算法准确性的估计。

其中包含了一个bagging算法

## bagging算法

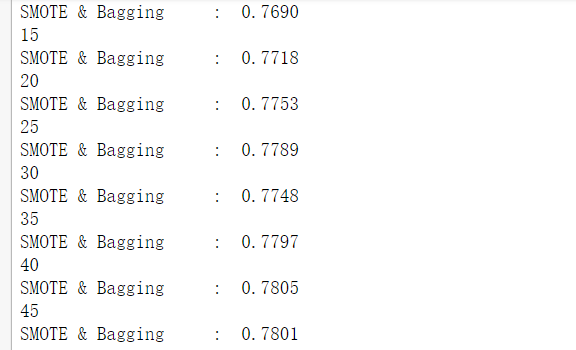
bagging（bootstrap aggregating）是一种用来提高学习算法准确度的方法，这种方法通过构造一个预测函数系列，然后以一定的方式将它们组合成一个预测函数。

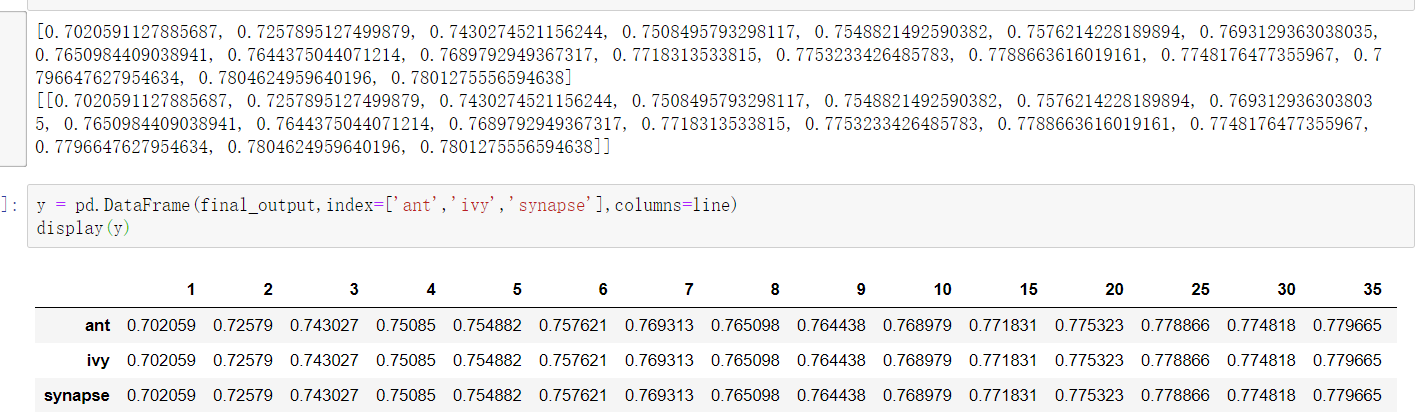
* Input: 总数据集
* Output: FPA

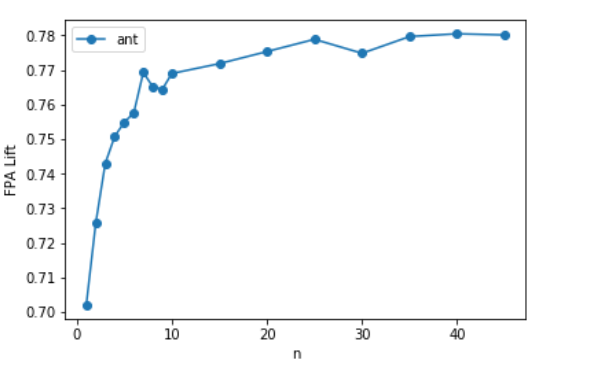
之后我分别定义了四种不同的预测算法，来进行横向比较，分别是：

1.def Smote\_Bagging(dataset,n): （及使用了smote又使用了bagging ）

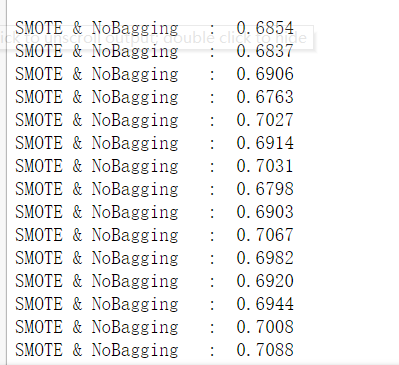
执行后的训练效果（都是在ant dataset上的结果）：

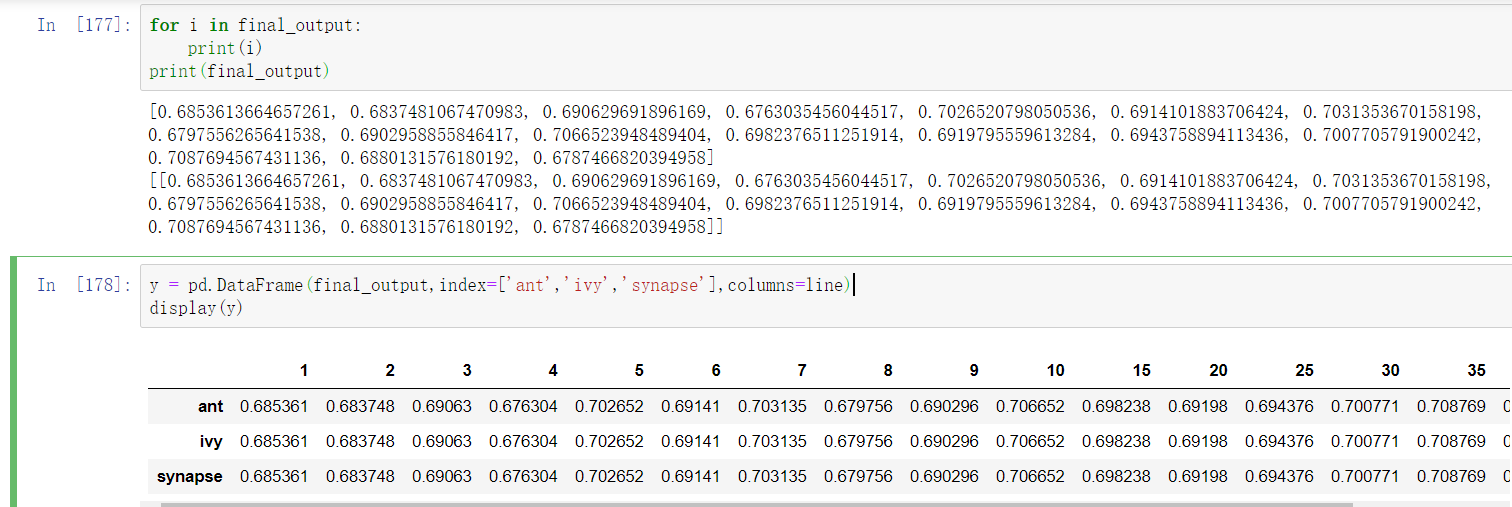


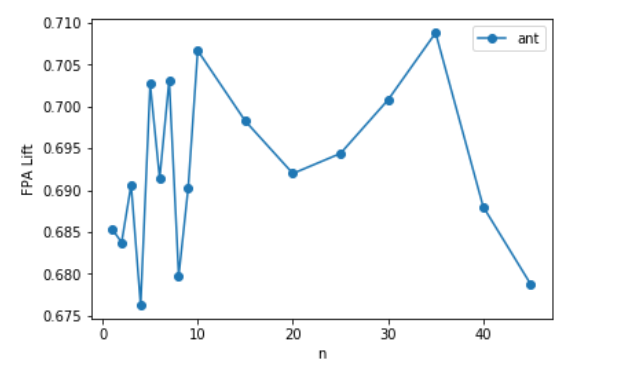




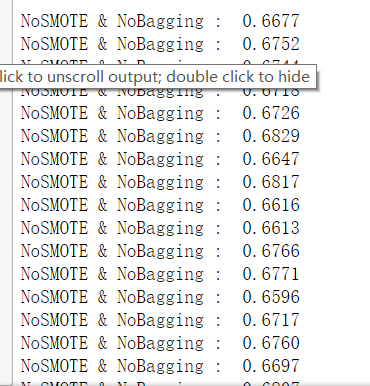
2.def Smote\_NoBagging(dataset): （使用了smote但是没有bagging）

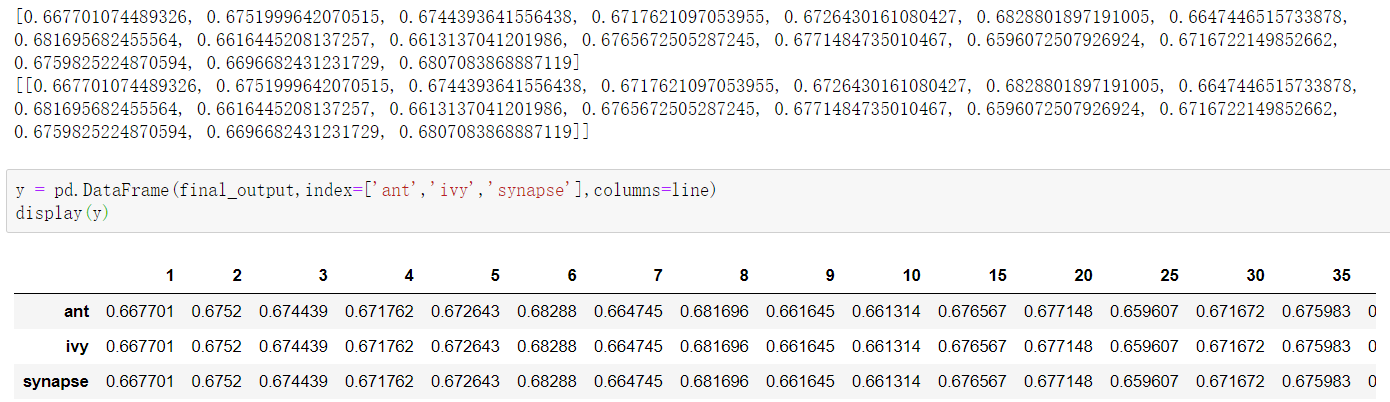


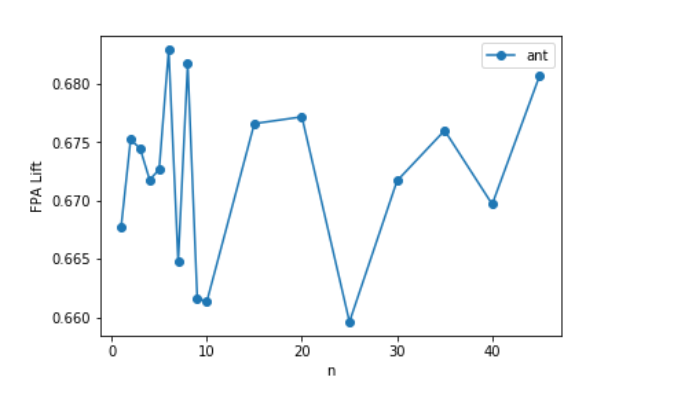




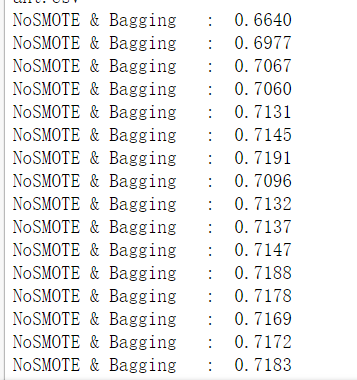
3.def NoSmote\_NoBagging(dataset,n): （没有smote同样没有bagging）

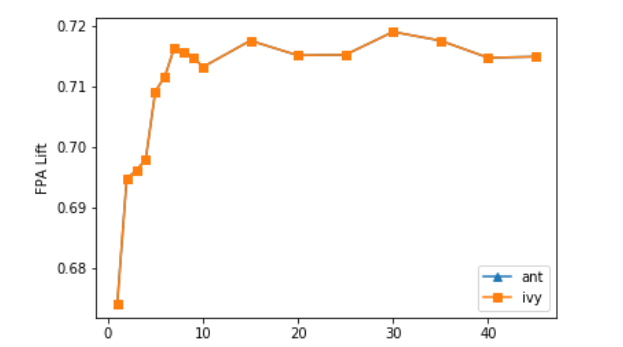
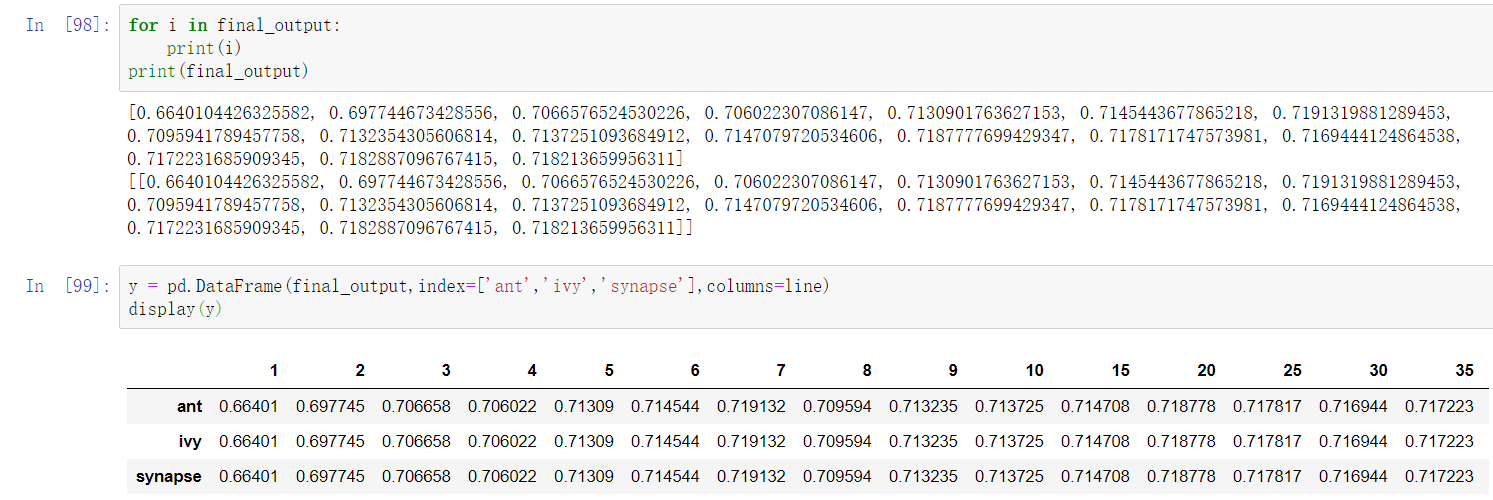






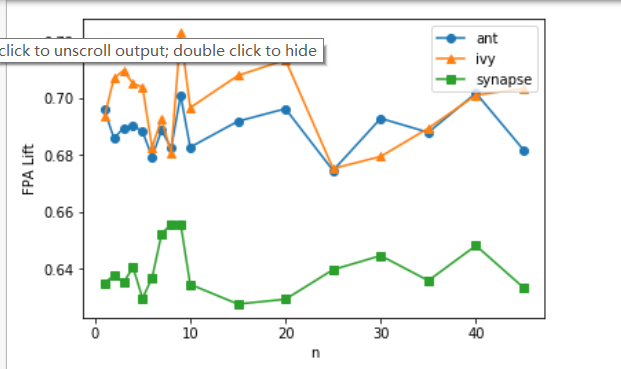
4.def NoSmote\_Bagging(dataset,n): （没有smote但是bagging）





从上面四组数据中可以很清楚的得出结论，对于数据集运用smote和bagging算法优化的数据集，表现会更加好。模型最后可以收敛，而且正确率也是最高的达到了78%。

**纵向比较**



从上图也可以看出:

将ant, ivy, synapse三种数据集，同时运行，因为担心训练时间太久，我就去了比计较少的训练次数和步长。可以实现用不同的算法在不同的数据集上面，运行，从而发现，每种算法的优势,从而根据不同特点的datasets找到合适的软件缺陷预测算法，提高准确率。

**Part 2(use DeepLearning)**

**ABSTRACT**

Software Defect Prediction is an important aspect in order to ensure software quality. Deep Learning techniques can also be used for the same. In this paper, w​e propose to extract a set of expressive features from an initial set of basic change measures using Artificial Neural Network (ANN), and then train a classifier based on the extracted features using Decision tree and compare it to three other methods wherein features are extracted from a set of initial change measures using dimensionality reduction techniques that include Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA) and Kernel PCA. We use five open source datasets from NASA Promise Data Repository to perform this comparative study. For evaluation, three widely used metrics: Accuracy, F1 scores and Areas under Receiver Operating Characteristic curve are used. It is found that ​Artificial Neural Network outperformed all the other ​dimensionality reduction techniques. Kernel PCA performed best amongst the dimensionality reduction techniques.

**项目简介**

**运行环境介绍**

第二部分，采用的是和第一部分完全不同的DeepLearning 中常常用于软件缺陷测试的LDA,CPA和KCPA算法，同样也是在win10，anaconda3下面中默认的SpyderIDE中开发的，主要是考虑到开发和调试的便利性，而且Spyder对于数据的可视化做的更加优秀，虽然体量较大，但是能够接受。

**项目中使用的库和技术**

包括常用于科学计算的，numpy和pandas更多的是sklearn下的封装好的preprocessing（用于数据的预处理）；PCA（主成分分析）算法、LDC（Linear Discriminant Analysis）算法、KPCA（核主成分分析）算法，决策树分类算法（DecisionTreeClassifier），使用ROC curve曲线图来可视化算法的精确度。

1.使用pd.read\_csv读入cm1.csv文件，该文件是NASA的中关于软件缺陷预测的中的datasets中的一个。

2.使用labelencoder对文件进行，预处理的编码过程。

3.接着使用train\_test\_split0函数来将原本是一个整体的文件按照25%的比例合理的分为测试集和训练集。

4.之后使用StandardScaler函数对数据进行进一步的规范化处理。

5.然后就是程序的核心，本部分一共提供了三种不同的算法来对，软件缺陷进行预测，比较出那种算法的优势和不足，并得出混淆矩阵。

6.利用sklearn.metrics中的roc\_auc\_score方法求出roc curve中的AUC

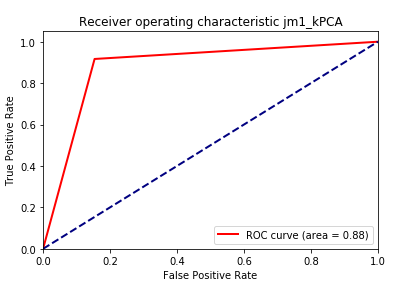
7.之后就是数据的可视化，先利用sklearn中的预处理方法如label\_binarize、

OneHotEncoder，对生成的数据进行处理。

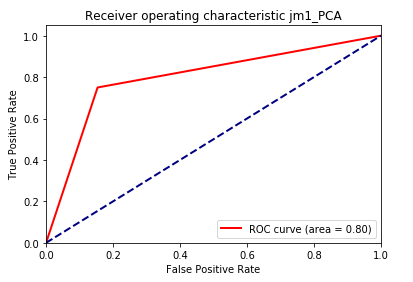
8.同样使用python常用的matplotlib.pyplot可视化库，来可视化roc曲线，和对应的数据。

对比三种算法运行的roc曲线效果图：

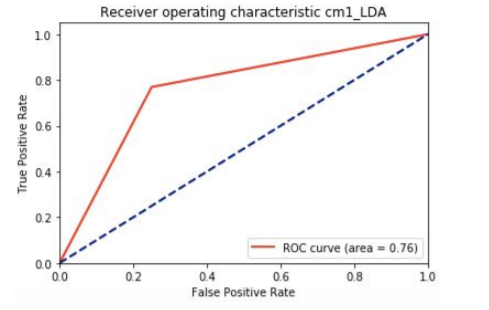
KCPA算法：



CPA算法：



LDA算法：



综上可以看出，在cm1.csv的数据集上面，KCPA算法的表现最好，LDA算法准确度最低。