**创新实践实验报告**

## 数据分析的背景

大多数在线判题系统或网站的评判系统都使用自动评分方式，通常依靠测试结果来验证所提交源代码的正确性和性能是否达到要求。但是，传统的动态测试需要花费大量时间，并且即使对于那些小规模的程序，性能问题的发现通常也是在问题暴露之后。我们提出了一种预测方法，与传统的动态测试相比可以在代码运行前有效预测代码的性能缺陷。

## 问题描述

我们的工作目标主要是创建一个预测，以确定给定的源代码在运行之前是否可能存在性能缺陷。这里，具有性能缺陷的源代码是指程序不能在问题陈述中指出的有限时间内执行，通常称为超出时间限制（TLE）。即学习一个函数，它可以在运行之前将源代码副本映射到相应的性能预测分类。将代码作为输入，并返回1表示该测试用例包含性能缺陷（TLE），0表示无性能缺陷（TLE）。我们通过学习训练数据集中提供的大量示例来近似该分类函数。也就是说，我们构建预测函数，其中，表示某个代码存在超时问题，表示某个代码可以在限定时间内正确通过所有测试用例。

## 预测模型整体流程

我们提出的方法的整体工作流程如图3.1所示。我们的方法首先从训练和测试集的源代码中获取令牌作为输入，并从中生成特征，然后用于构建和评估模型以预测性能缺陷。

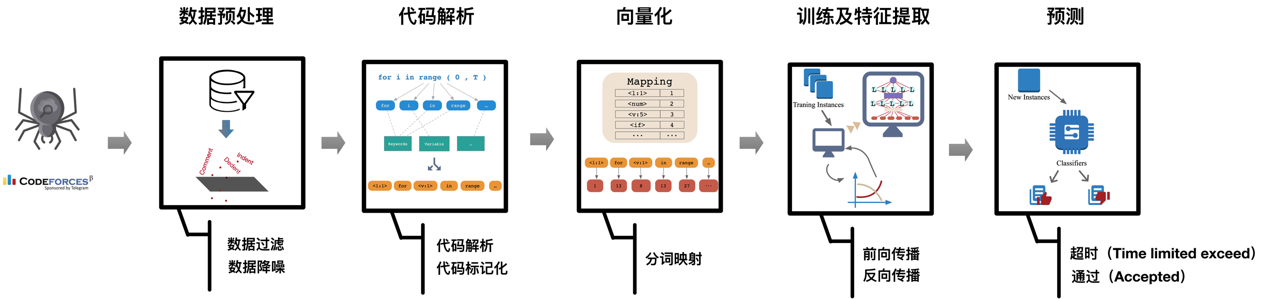


图3.1 流程

具体来说，我们的方法包括三个主要步骤：1）数据预处理和代码解析，用于从源代码中提取代码标记，2）标记向量化，用于将标记序列转换为整数向量，训练和特征提取用于通过学习方法从整数向量生成特征，3）构建预测模型。

### 数据预处理及代码解析与标记化

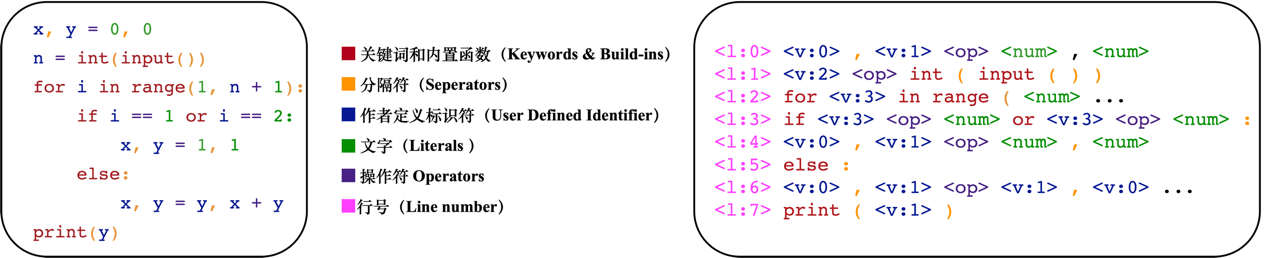


图3.2 代码解析与标记化

separate = [**','**,**'.'**,**':'**,**'('**,**')'**,**'['**,**']'**]  
operator = [**'='**,**'=='**,**'+'**,**'-'**,**'/'**,**'\*'**,**'%'**,**'^'**,**'<'**,**'<='**,**'>'**,**'>='**,**'+='**,**'-='**]  
a=[**'a'**,**'b'**,**'c'**,**'d'**,**'e'**,**'f'**,**'g'**,**'h'**,**'i'**,**'j'**,**'k'**,**'l'**,**'m'**,**'n'**,**'o'**,**'p'**,**'q'**,**'r'**,**'s'**,**'t'**,**'u'**,**'v'**,**'w'**,**'x'**,**'y'**,**'z'**,**'str'**]  
data=[]

**for** sourceIndex,row **in** source.iterrows():  
 mid = []  
 sourceCode = row[**'code'**]  
 sourceverdict = row[**'verdict'**]  
 **for** i **in** sourceverdict:  
 mid.append(i)  
 target.append(**''**.join(mid))  
 str=**''  
 for** toknum,tokval,start,end,\_ **in** tokenize.tokenize(BytesIO(sourceCode.encode(**'utf-8'**)).readline):  
 *# print(count)* print(**"TokenType"**,token.tok\_name[toknum],**'\tToken'**,tokval,**'\tPosition'**,start,end)  
 s=**''**.join(**'%s'** %id **for** id **in** start)  
 e=**''**.join(**'%s'** %id **for** id **in** end)  
 **if** token.tok\_name[toknum]==**'NUMBER'** :  
 str=str+**" "**+**'NUMBER'  
 if** tokval **in** a:  
 str=str+**' '**+**'define'  
 if** tokval **in** operator:  
 str = str + **" "** + **'operator'  
 else**:  
 str = str + **" TokenType "** + token.tok\_name[toknum]+**' Token '**+tokval  
 data.append(str)

### 标记向量化与训练与特征提取

**from** sklearn.feature\_extraction.text **import** TfidfVectorizer  
**from** sklearn.naive\_bayes **import** MultinomialNB  
**from** sklearn.model\_selection **import** train\_test\_split  
*# 测试集与训练集分类*  
x\_train,x\_test,y\_train,y\_test = train\_test\_split(data[0:20000],target[0:20000],test\_size=0.2,random\_state=0)

**from** sklearn.linear\_model **import** LogisticRegression  
**from** sklearn.ensemble **import** RandomForestClassifier  
**from** sklearn.neighbors **import** KNeighborsClassifier  
*# 特征工具*transfer = TfidfVectorizer()  
x\_train = transfer.fit\_transform(x\_train)  
x\_test = transfer.transform(x\_test)

### 预测

*# 朴素贝叶斯算法预估器  
# estimator = MultinomialNB()  
# estimator = LogisticRegression()*estimator = RandomForestClassifier()  
  
estimator.fit(x\_train,y\_train)  
*# 模型评价*y\_predict = estimator.predict(x\_test)  
print(**'y\_predict:'**,y\_predict)  
print(**'预测值是否正确:'**,y\_test==y\_predict)  
  
score = estimator.score(x\_test,y\_test)  
print(**'准确率：'**,score)

## 实验

### 实验平台

### PyCharm

### 评价指标

在实验过程中，为了评估预测结果， 采用准确率estimator.score()，即预测结果正确的百分比。

### 数据集描述

Codeforce网站上提交的代码，每条数据是一次代码的提交。

包含：

PairIndex:代码对信息

Problemlndex:问题编号

contestID:问题集编号

Runtime:运行时间

Verdict：代码状态（ok，TIME\_LIMITED\_EXCEEDED）

submitID：代码提交编号

Code：源代码

### 实验结果

在用pandas读取数据后，建立TF-IDF模型标记向量化和训练数据集。得到模型后用朴素贝叶斯算法预估器进行预测，或许是训练集不够多，最终得到0.63的准确率。

本实验数据集代码较多，这里只截取部分分类结果。

