Repository Risk Analysis

Introduction

This Project will help project manager to track risk associated with a file in a project using Software Engineering Metrics. Since output is csv file changes in values can be easily spotted by finding difference between two commits.

Use Case

This tool will come handy when a project manager wants to check what is the quality of code. Sometimes developers tends to use more lines of code which are not useful at all, or they use to much of space or comments to increase LOC whiich can be monitored easily using this tool. This tool uses some common metrics like Halstead's Software Metrics, Cyclomatic complexity (McCabe), Maintainability Index, Number of operators, etc. An examinar can see this values in csv file which is output of this tool and mark files which can have potential risk.

Usage

Open `Code_Bug_Prediction.ipynb` in Google Colab. In cell where `!git clone` is written change the link with your repository path. And Run All Cells Output file will be in your Files section.

Features

- Support multiple programming languages
- Easy to run python notebook file
- CSV output for easy visualization
- Made to work on Java

Packages Required

Seperate Installation is not required. Notebook has installing command writen.

> priv-kweihmann/multimetric

Output Column Description

item	description	range	recommendation
comment_ratio	Comment to Code percentage	0..100	> 30.0
cyclomatic_complexity	Cyclomatic complexity according to McCabe	0..(inf)	< 10
fanout_external	Number imports from out of tree modules	0..(inf)	
fanout_internal	Number imports from same source tree modules	0..(inf)	
halstead_bugprop	Number of delivered bugs according to Halstead	0..(inf)	< 0.05
halstead_difficulty	Difficulty according to Halstead	0..(inf)	
halstead_effort	Effort according to Halstead	0..(inf)	
halstead_timerequired	Time required to program according to Halstead	0..(inf)	
halstead_volume	Volume according to Halstead	0..(inf)	
lang	list of identified programming languages	list	
loc	Lines of code	1..(inf)	
maintainability_index	Maintainability index	0..100	> 80.0
operands_sum	Number of used operands	1..(inf)	
operands_uniq	Number of unique used operands	1..(inf)	

item	description	range	recommendation
operators_sum	Number of used operators	1..(inf)	
operators_uniq	Number of unique used operators	1..(inf)	
pylint	General quality score according to pylint	0..100	> 80.0
tiobe_compiler	Compiler warnings score according to TIOBE	0..100	> 90.0
tiobe_complexity	Complexity according to TIOBE	0..100	> 80.0
tiobe_coverage	Coverage according to TIOBE	0..100	> 80.0
tiobe_duplication	Code duplications score according to TIOBE	0..100	> 80.0
tiobe_fanout	Fan-Out score according to TIOBE	0..100	> 80.0
tiobe_functional	Functional defect score according to TIOBE	0..100	> 90.0
tiobe_security	Security score according to TIOBE	0..100	> 90.0
tiobe_standard	Language standard score according to TIOBE	0..100	> 80.0
tiobe	General quality score according to TIOBE	0..100	> 80.0

Task List

- [x] Java Support
- [x] Multi Language Support
- [x] Formatted Output
- [ ] Module Level Analysis
- [ ] Rule Base Output
- [ ] Change of values on Git Commit

FlowChart

```
 step1=>start: Start
step2=>operation: clone repository
step3=>operation: install multimetric
step4=>operation: get metric for all files in folder
step5=>operation: output json file
step6=>operation: format json file
step7=>operation: export csv from json
step8=>end: csv file
```

Developer

- Sarthak Pan (TCS Intern)

Authorization

Tata Consultancy Services