JAVA

|  |  |  |  |
| --- | --- | --- | --- |
| Tool | Accuracy (%) | Average Response Time (in seconds) | Storage requirement (in MBs) |
| BloatLibD |  | 12.0431 | 1.52 |
| DepClean |  | 4.4578 | Not Applicable |
| Jingredients |  |  |  |

For DepClean, the Response Time used to calculate the Average Response Time involves only the Running Time of the actual Depclean dependency analysis on the maven project.

As DepClean relies on analysis of code rather than comparison with a database of libraries , storage requirement is not applicable to it.

5 maven projects have been selected, and the used dependencies in them (whether direct or transitive) have been used as data (in the form of jars) for BloatLibD and Jingredients. There are 38 jar files (taking union of all the dependencies as there were some common dependencies amongst them).

PYTHON

|  |  |  |  |
| --- | --- | --- | --- |
| Tool | Accuracy (%) | Average Response Time (in Milli Seconds) | Storage requirement (in MBs) |
| PyCln | 98.93 | 0.1755 | Not Applicable |
| AutoFlake | 98.93 | 0.2280 | Not Applicable |

Both PyCln and AutoFlake rely on code analysis rather than matching with a white list, hence Storage requirement not applicable on them

3 Projects have been selected from GitHub at random consisting of 48 .py files (including init.py files as well)

Response time for both Python and Java has been measured using pythons time.time() function (diffrence b/w start time and end time taken per file and then average is taken).

To calculate metrics, total number of imports are obtained from all .py files and then based on whether an import is actually used and whether the tool correctly removes/flags the unused ones the following are defined:

AU – Import actually used

AUN – Import actually unused

DU – Import detected as used (better say not flagged as unused) by the tool

DUN – Import detected as unused by the tool

Total imports = 283 (each individual module imported is consider e.g. import a,b,c means 3 imports)

AU/DU ---> a True Positive (detected and is actually used in the file)

AU/DUN ---> a False Negative (actually used but incorrectly flagged as unused)

AUN/DU ---> a False Positive (actually unused but incorrectly missed by the tool)

AUN/DUN ---> a True Negative (actually unused and correctly flagged as unused)

In essence a confusion matrix is formed

For PyCln

|  |  |  |
| --- | --- | --- |
| Tool | DU | DUN |
| AU | 258 | 0 |
| AUN | 3 | 22 |

For AutoFlake

|  |  |  |
| --- | --- | --- |
| Tool | DU | DUN |
| AU | 257 | 0 |
| AUN | 3 | 23 |

Accuracy = TP+TN/ Total Obserations

\*\*\*\* Both produce near identical results, there are differences (example is of the file xsstrike/xsstrike.py ‘s imports)

\*\*\*\* May be possible that both PyCln and Autoflake both can’t detect transitively unused imports, e.g. consider two files A.py and B.py . B imports some modules but does not use at least one of them, then A imports B (as in **from B import \*** ) alongside possibly some of it’s own modules (where there are modules that don’t match with those of B). In this case both tools won’t flag the extra unused libraries residing in B. Alternatively A imports B completely but does not use some imported module(s) within B.

Case in point check XSStrike/core/log.py file’s imports.