ICSEA 2023

SOURCE CODE ANALYSIS OF GITHUB PROJECTS FROM E-COMMERCE AND GAME DOMAINS

Doğa Babacan, Tugkan Tuglular

Presenter: Tugkan Tuglular

tugkantuglar@iyte.edu.tr

Izmir Institute of Technology

Assoc. Prof. Tugkan TUGLULAR, Ph.D.
Izmir Institute of Technology, Turkey

Tugkan Tuglular received the B.S., M.S., and Ph.D. degrees in Computer Engineering from Ege University, Turkey, in 1993, 1995, and 1999.

He worked as a research associate at Purdue University from 1996 to 1998.

He has been with Izmir Institute of Technology since 2000.

After becoming an Assistant Professor at Izmir Institute of Technology, he worked as Chief Information Officer in the university from 2003-2007.

In addition to his academic duties, he acted as IT advisor to the Rector between 2010-2014.

In 2018, he became an Associate Professor in the Department of Computer Engineering of the same university.

He has more than 75 publications and an active record of duties with international and national conferences.

His current research interests include model-based testing and software quality with machine learning support.

Objective

- Similarities and differences between repositories depending on their source code analysis result attributes
- Each topic separately:
  - E-commerce
  - Game
- Public repositories from GitHub

What is used?

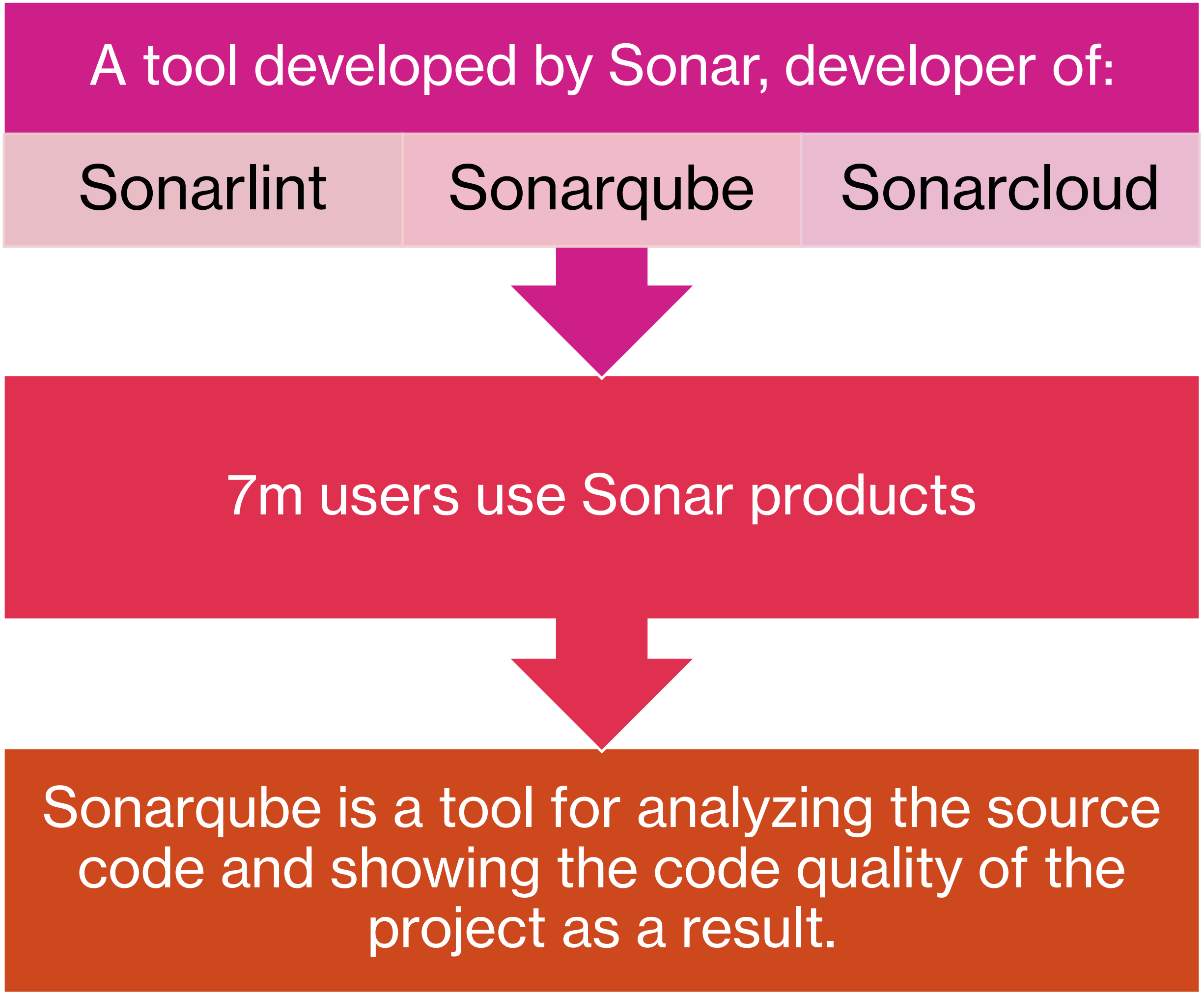
GitHub utilized

Python language is used to:

- Reach repositories’ informations
- Clone repositories
- Upload to Sonarqube for inspection

Sonarqube is the main tool of this project

WHAT IS SONARQUBE?



A tool developed by Sonar, developer of:

Sonarlint | Sonarqube | Sonarcloud

7m users use Sonar products

Sonarqube is a tool for analyzing the source code and showing the code quality of the project as a result.

Difficulties with Sonarqube

- Many version problems
- Only Maven utilized programs work without any setting
- Sonarqube actually must be set for each project separately
- Each test directory is different for different projects so user must show each directory to Sonarqube. So we skipped tests.
- For this reason managed to do assessment only on 25 projects:
    - 10 from game category
    - 15 from e-commerce category
- These are the only ones that can be downloaded from Github then uploaded to Sonarqube automatically from the first 100 projects.

How to Get Repositories from GitHub

`«https://api.github.com/search/repositories?q=e-commerceis:featured+language:java&sort=stars&order=desc&per_page=100&page=1»`

- The line is used to for a get request to retrieve the first hundred repositories in GitHub.
- With the help of GitHub package and a token retrieved from GitHub, able to send multiple requests in small periods.
- The directory is named specificly in a pattern to carry over the star count to the tables, we also use the directory name for the project key in Sonarqube.
- `«3091E-c-o-Mshopizer»` is an example directory name where 3091 is the stargazers value, 'E-c-o-M' shows that it is an e-commerce project, 'shopizer' is the name of the project. By separating these values, it is easy to add the stargazers value to our dataframe.

How to Get Repositories from GitHub cont.

- We imported the os library in Python and used `«os.mkdir(path)»` to create the path for cloning the repository where path is combination of parent directory which is for example `«C:/Programming/ProjectRepo/»` and the directory which is `«3091E-c-o-Mshopizer»`
- If 'pom' file exists (Maven specific file), with utilizing Python again we write to console the following line; `«git clone {repo_url} {directory_name}»`
  - `«repo_url»` is the cloning link of the chosen GitHub repository, for instance `«https://github.com/shopizer-ecommerce/shopizer.git»`
  - `«directory_name»` is still the same which is `«3091E-c-o-Mshopizer»`
- By entering this line we start cloning the repository

How to Upload Repositories to Sonarqube

- We connect to Sonarqube by using the installing and importing the `«SonarQubeClient»` we pass the arguments, username, password and url of the Sonarqube which is in our case is `«http://localhost:9000/»`
- We create the sonarqube project for each repository by `«sonar.projects.create_project()»` where name and project parameters are both the directory `(3091E-c-o-Mshopizer)`
- Then we upload the repository to Sonarqube by a console command entered by using os.system with command such as `«mvn clean verify sonar:sonar -D maven.test.skip=true -D sonar.projectKey={projectKey} -D sonar.host.url=http://localhost:9000 -D sonar.login=sqa_***********»`

How to Get Source Code Analysis Results and Utilize

- To recieve the source code analysis results from the Sonarqube, Sonarqube api package in Python is utilized. With the right token, it is easy to send requests to get each attribute
- Then we place them in a dictionary and convert them to dataframe and save as csv to work on it on the Jupyter Notebook.
- In Jupyter Notebook we normalized the data, created a correlation matrix to see the realtionship between our features.
- Finally we created elbow graphs and cluster graphs between the attributes we have selected such as comment lines vs. code smells, bugs vs. classes

Attributes

- **Bug:** A coding mistake that can cause problems during runtime of the software
- **Vulnerability:** A certain point in the code which is open to attack.
- **Code Smell:** An problem which causes the code to be less understandable and difficult to maintain.
- **Violations:** Any form of issue is also called violations. Prefixes change depending on the importance of the violation, it can be blocker, critical, major, minor and info.
    - **Blocker:** It has a high chance to affect the behavior of the software in production. Sonarqubes suggestion is to fix the violation immediately.
    - **Critical:** It has a low chance to impact the behavior of the software in production or there is a security flaw that needs attention. Suggestion of Sonarqube is to review the code as soon as possible.
    - **Major:** A flow in quality where these issues can cause huge reductions in productivity of the development phase of software. These can be unused parameters, duplicated blocks etc..
    - **Minor:** A flow in quality where these issues cause small reductions in productivity of the development phase of software. Too long lines, number of ‘switch’ cases lower than 3 are some examples of minor violations.

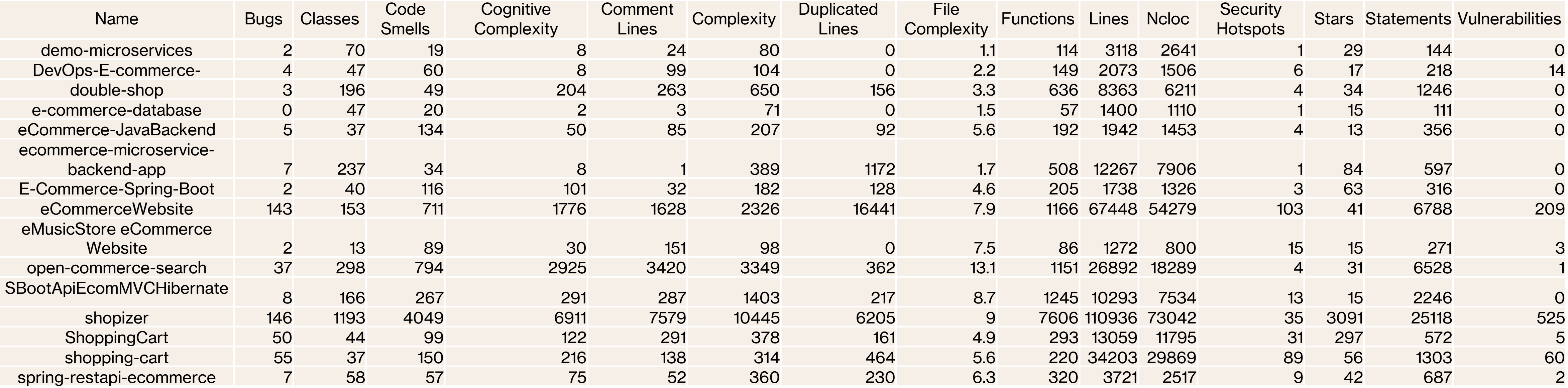
Attributes cont.

- **Security Hotspots:** A piece of code which is security sensitive, however it is not as important as vulnerability, these hotspots may not have an impact on the whole software unlike the vulnerability.
- **Lines:** Number of physical lines.
- **Lines of Code:** The number of physical lines that contain at least one character, however this character will not be counted if it is a whitespace, tabular space or part of a comment.
- **Functions:** Number of functions.
- **Statements:** Number of statements.
- **Complexity:** Complexity (cyclomatic complexity) is a type of metric where the number of paths in a code is calculated and minimum value of function is 1 . When the control flow of a piece of code diverges, the complexity increases. This calculation may differ depending on the language being used. (For java keywords incrementing the complexity are: if, for, while, case, catch, throw, &&, ||, ?)

- **Cognitive Complexity:** Cognitive complexity is a more detailed way of inspecting the complexity of a code. It is not a quantitative way of measuring as it is in cyclomatic complexity, it also counts in the degree of interconnectedness and abstraction or indirection in a piece of code. Cognitive complexity looks for how understandable the code is and how much it is easy to maintain.

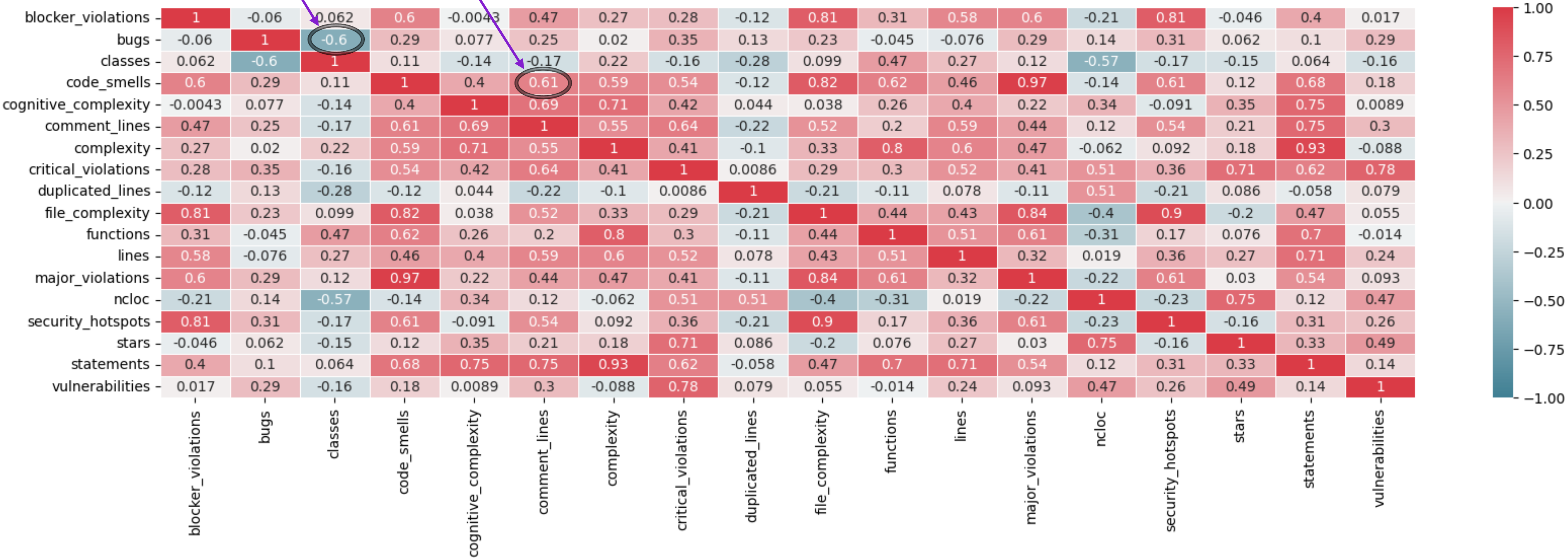
SOURCE CODE ANALYSIS RESULTS OF REPOSITORIES WITH TOPIC ‘E-COMMERCE’

The Original Data Received From Sonarqube



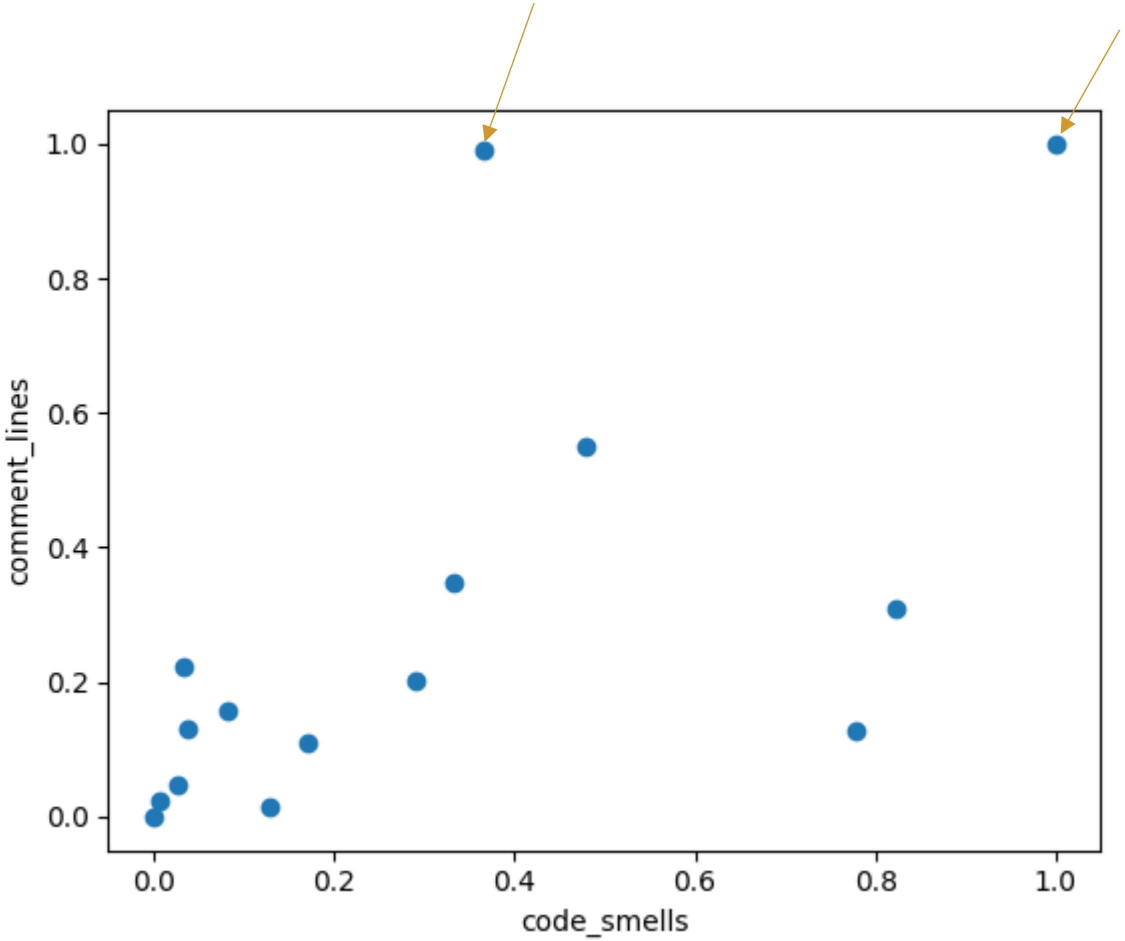
Name	Bugs	Classes	Code Smells	Cognitive Complexity	Comment Lines	Complexity	Duplicated Lines	File Complexity	Functions	Lines	Ncloc	Security Hotspots	Stars	Statements	Vulnerabilities
demo-microservices	2	70	19	8	24	80	0	1.1	114	3118	2641	1	29	144	0
DevOps-E-commerce-	4	47	60	8	99	104	0	2.2	149	2073	1506	6	17	218	14
double-shop	3	196	49	204	263	650	156	3.3	636	8363	6211	4	34	1246	0
e-commerce-database	0	47	20	2	3	71	0	1.5	57	1400	1110	1	15	111	0
eCommerce-JavaBackend	5	37	134	50	85	207	92	5.6	192	1942	1453	4	13	356	0
ecommerce-microservice-backend-app	7	237	34	8	1	389	1172	1.7	508	12267	7906	1	84	597	0
E-Commerce-Spring-Boot	2	40	116	101	32	182	128	4.6	205	1738	1326	3	63	316	0
eCommerceWebsite	143	153	711	1776	1628	2326	16441	7.9	1166	67448	54279	103	41	6788	209
eMusicStore eCommerce Website	2	13	89	30	151	98	0	7.5	86	1272	800	15	15	271	3
open-commerce-search	37	298	794	2925	3420	3349	362	13.1	1151	26892	18289	4	31	6528	1
SBootApiEcomMVCHibernate	8	166	267	291	287	1403	217	8.7	1245	10293	7534	13	15	2246	0
shopizer	146	1193	4049	6911	7579	10445	6205	9	7606	110936	73042	35	3091	25118	525
ShoppingCart	50	44	99	122	291	378	161	4.9	293	13059	11795	31	297	572	5
shopping-cart	55	37	150	216	138	314	464	5.6	220	34203	29869	89	56	1303	60
spring-restapi-ecommerce	7	58	57	75	52	360	230	6.3	320	3721	2517	9	42	687	2

Correlation Analysis of Attributes of E-Commerce Repositories



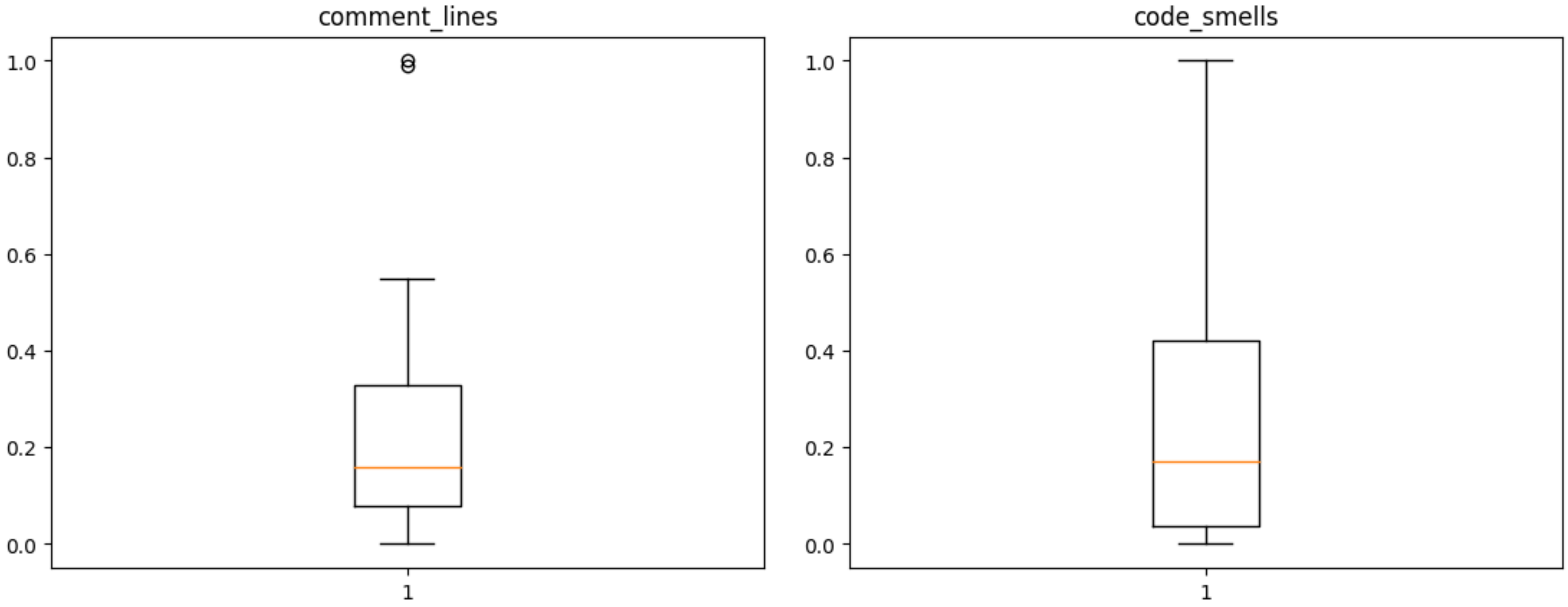
	blocker_violations	bugs	classes	code_smells	cognitive_complexity	comment_lines	complexity	critical_violations	duplicated_lines	file_complexity	functions	lines	major_violations	ncloc	security_hotspots	stars	statements	vulnerabilities
blocker_violations	1	-0.06	0.062	0.6	-0.0043	0.47	0.27	0.28	-0.12	0.81	0.31	0.58	0.6	-0.21	0.81	-0.046	0.4	0.017
bugs	-0.06	1	-0.6	0.29	0.077	0.25	0.02	0.35	0.13	0.23	-0.045	-0.076	0.29	0.14	0.31	0.062	0.1	0.29
classes	0.062	-0.6	1	0.11	-0.14	-0.17	0.22	-0.16	-0.28	0.099	0.47	0.27	0.12	-0.57	-0.17	-0.15	0.064	-0.16
code_smells	0.6	0.29	0.11	1	0.4	0.61	0.59	0.54	-0.12	0.82	0.62	0.46	0.97	-0.14	0.61	0.12	0.68	0.18
cognitive_complexity	-0.0043	0.077	-0.14	0.4	1	0.69	0.71	0.42	0.044	0.038	0.26	0.4	0.22	0.34	-0.091	0.35	0.75	0.0089
comment_lines	0.47	0.25	-0.17	0.61	0.69	1	0.55	0.64	-0.22	0.52	0.2	0.59	0.44	0.12	0.54	0.21	0.75	0.3
complexity	0.27	0.02	0.22	0.59	0.71	0.55	1	0.41	-0.1	0.33	0.8	0.6	0.47	-0.062	0.092	0.18	0.93	-0.088
critical_violations	0.28	0.35	-0.16	0.54	0.42	0.64	0.41	1	0.0086	0.29	0.3	0.52	0.41	0.51	0.36	0.71	0.62	0.78
duplicated_lines	-0.12	0.13	-0.28	-0.12	0.044	-0.22	-0.1	0.0086	1	-0.21	-0.11	0.078	-0.11	0.51	-0.21	0.086	-0.058	0.079
file_complexity	0.81	0.23	0.099	0.82	0.038	0.52	0.33	0.29	-0.21	1	0.44	0.43	0.84	-0.4	0.9	-0.2	0.47	0.055
functions	0.31	-0.045	0.47	0.62	0.26	0.2	0.8	0.3	-0.11	0.44	1	0.51	0.61	-0.31	0.17	0.076	0.7	-0.014
lines	0.58	-0.076	0.27	0.46	0.4	0.59	0.6	0.52	0.078	0.43	0.51	1	0.32	0.019	0.36	0.27	0.71	0.24
major_violations	0.6	0.29	0.12	0.97	0.22	0.44	0.47	0.41	-0.11	0.84	0.61	0.32	1	-0.22	0.61	0.03	0.54	0.093
ncloc	-0.21	0.14	-0.57	-0.14	0.34	0.12	-0.062	0.51	0.51	-0.4	-0.31	0.019	-0.22	1	-0.23	0.75	0.12	0.47
security_hotspots	0.81	0.31	-0.17	0.61	-0.091	0.54	0.092	0.36	-0.21	0.9	0.17	0.36	0.61	-0.23	1	-0.16	0.31	0.26
stars	-0.046	0.062	-0.15	0.12	0.35	0.21	0.18	0.71	0.086	-0.2	0.076	0.27	0.03	0.75	-0.16	1	0.33	0.49
statements	0.4	0.1	0.064	0.68	0.75	0.75	0.93	0.62	-0.058	0.47	0.7	0.71	0.54	0.12	0.31	0.33	1	0.14
vulnerabilities	0.017	0.29	-0.16	0.18	0.0089	0.3	-0.088	0.78	0.079	0.055	-0.014	0.24	0.093	0.47	0.26	0.49	0.14	1

Comment Lines vs Code Smells Graphs of E-Commerce Repositories

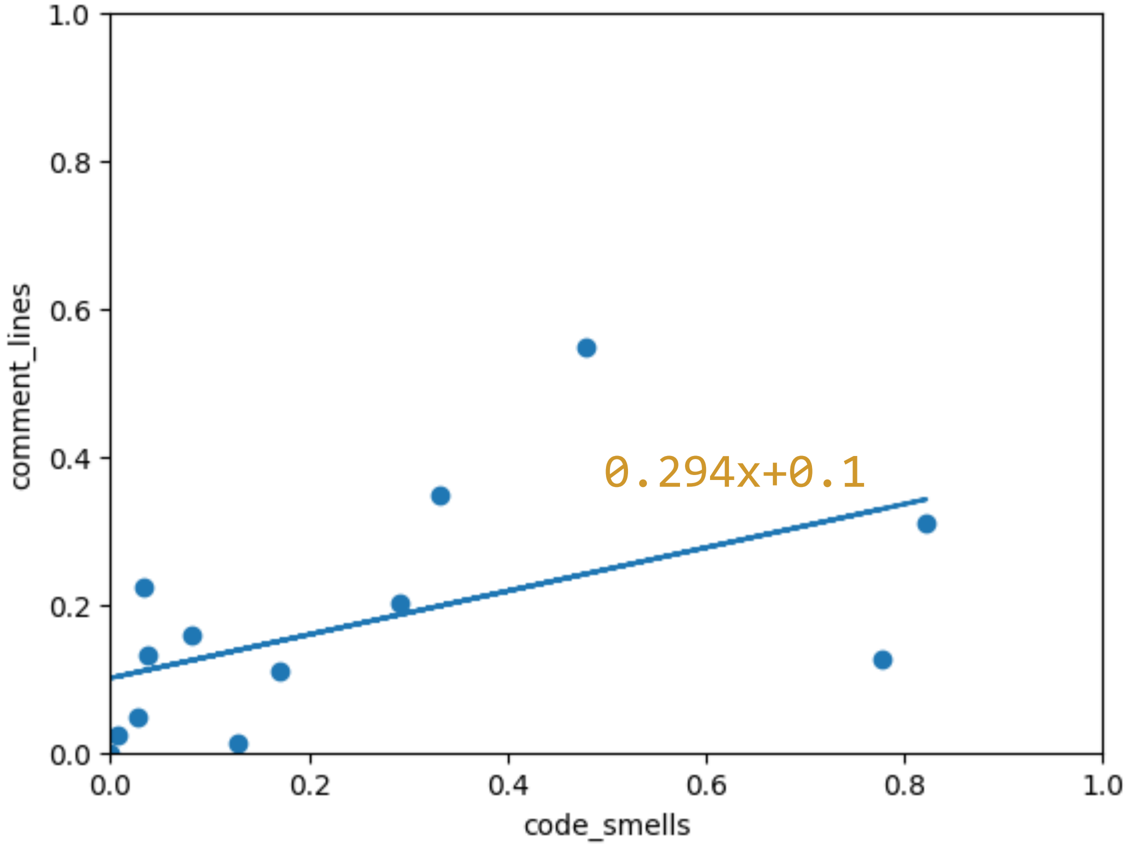


There seems to be two outliers

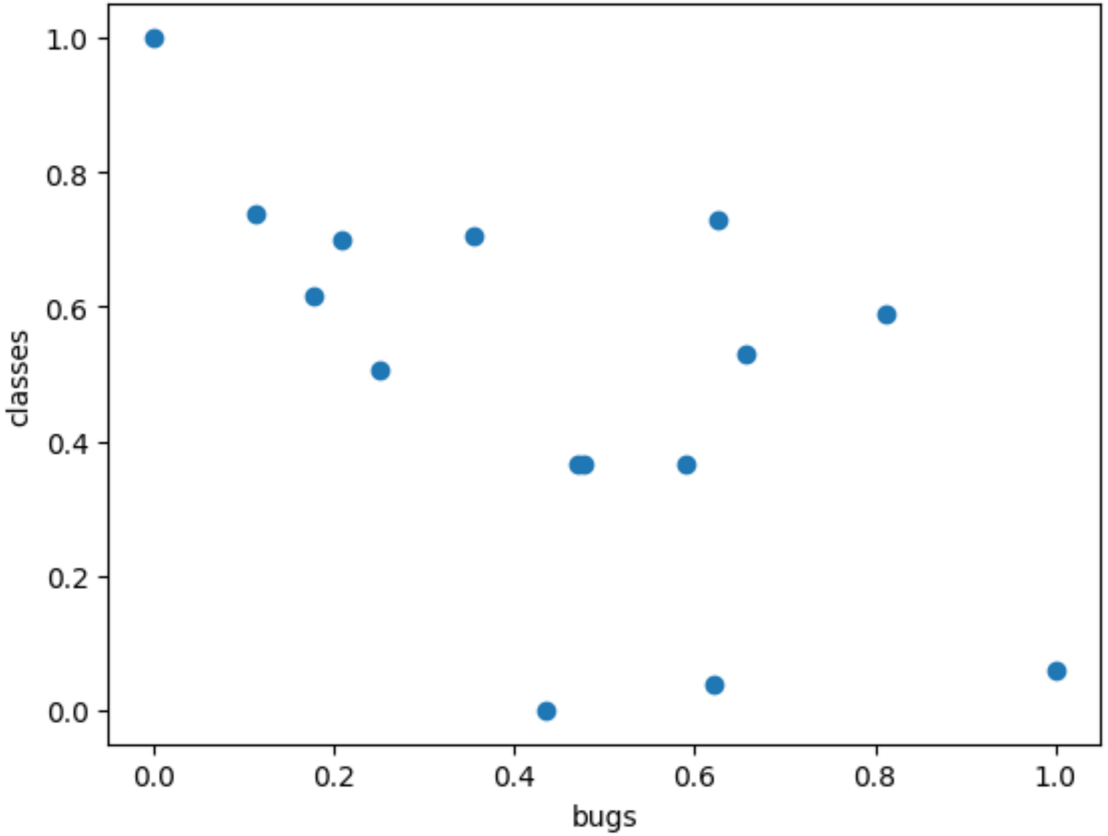
Box Plots of Comment Lines and Code Smells



Linear Regression of Comment Lines vs Code Smells After Removing the Outliers

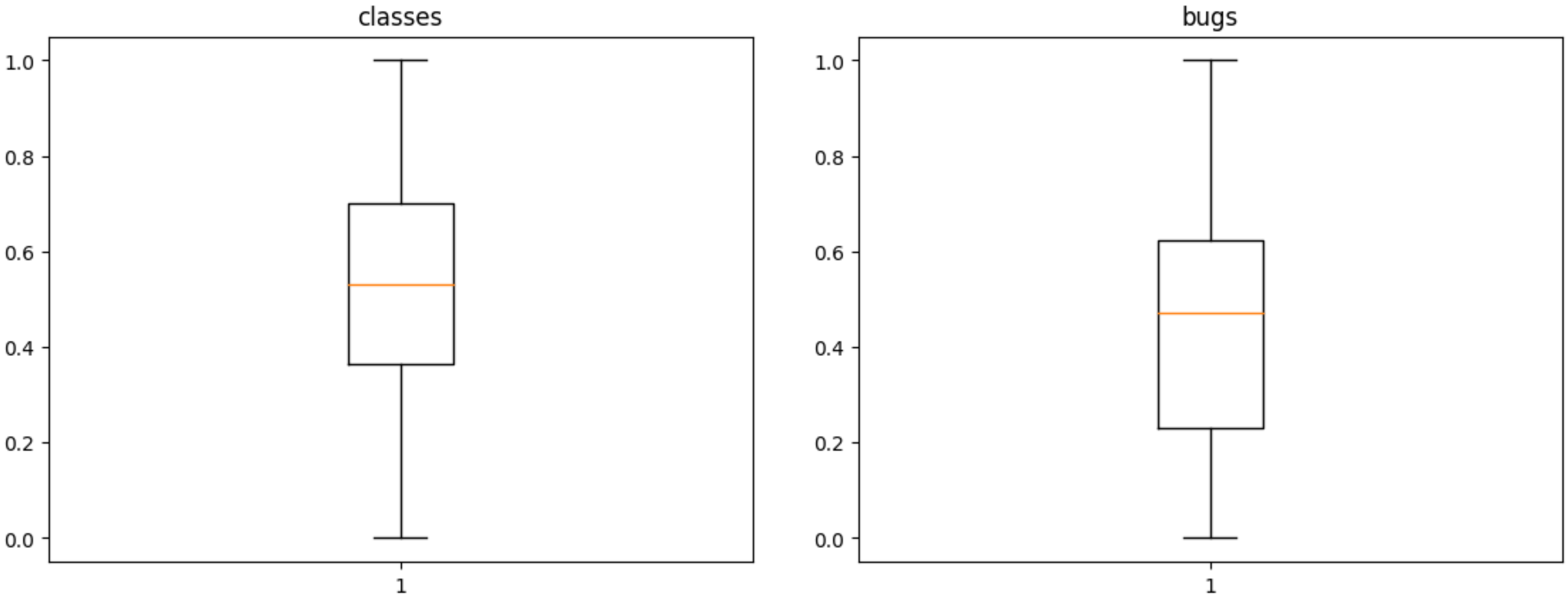


Number of Bugs vs Number of Classes Graphs of E-Commerce Repositories

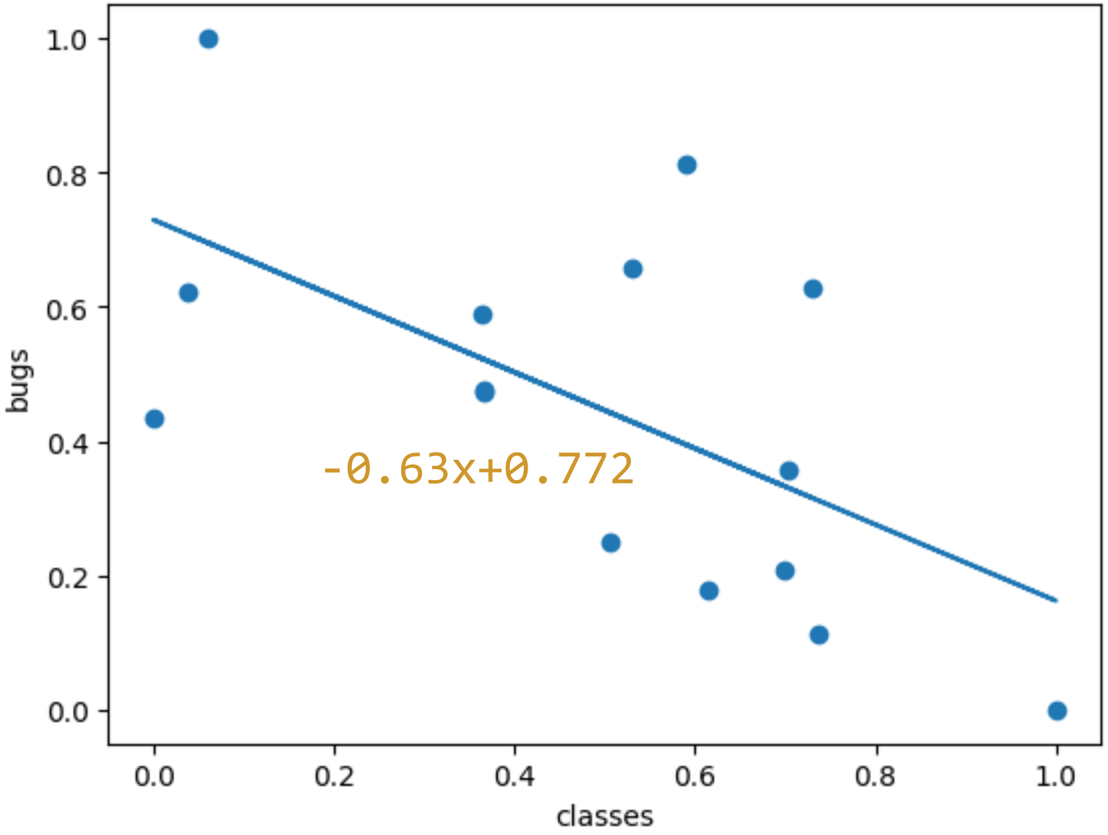


There is almost no outliers to remove

Box Plots of Number of Classes and Bugs



Linear Regression of Number of Bugs vs Number of Classes After Removing the Outliers

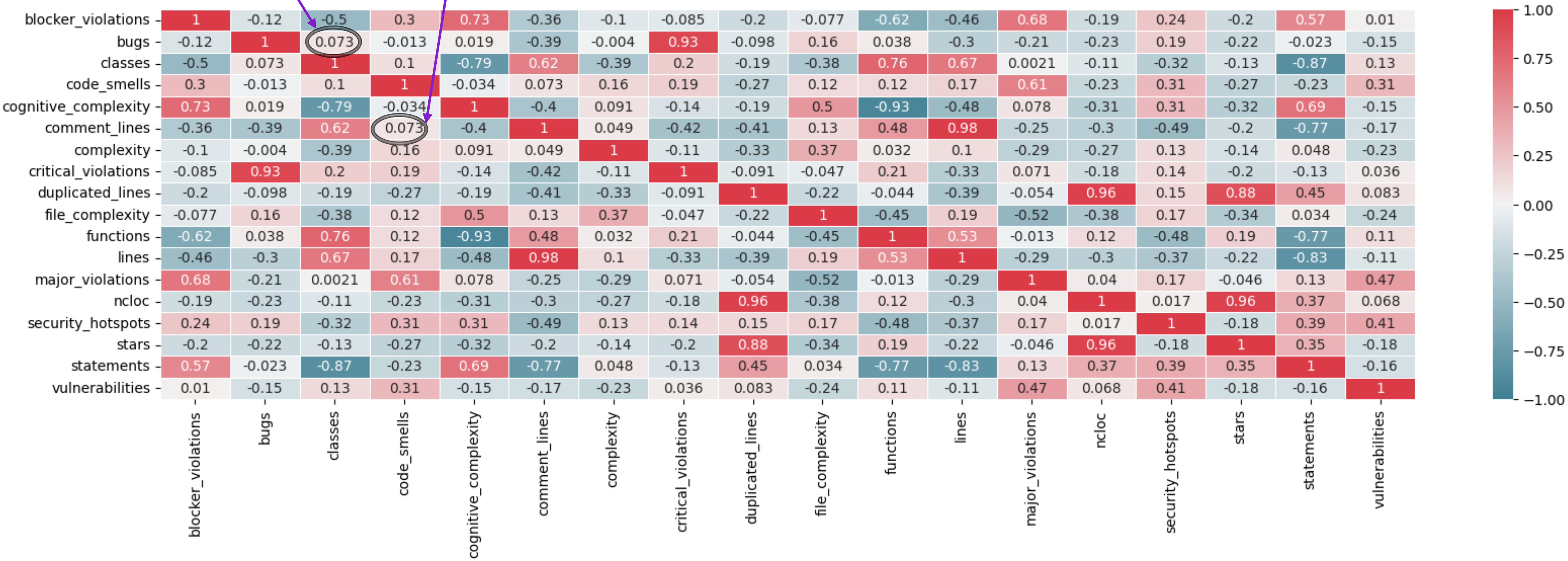


SOURCE CODE ANALYSIS RESULTS OF REPOSITORIES WITH TOPIC ‘GAME’

The Original Data Received From Sonarqube

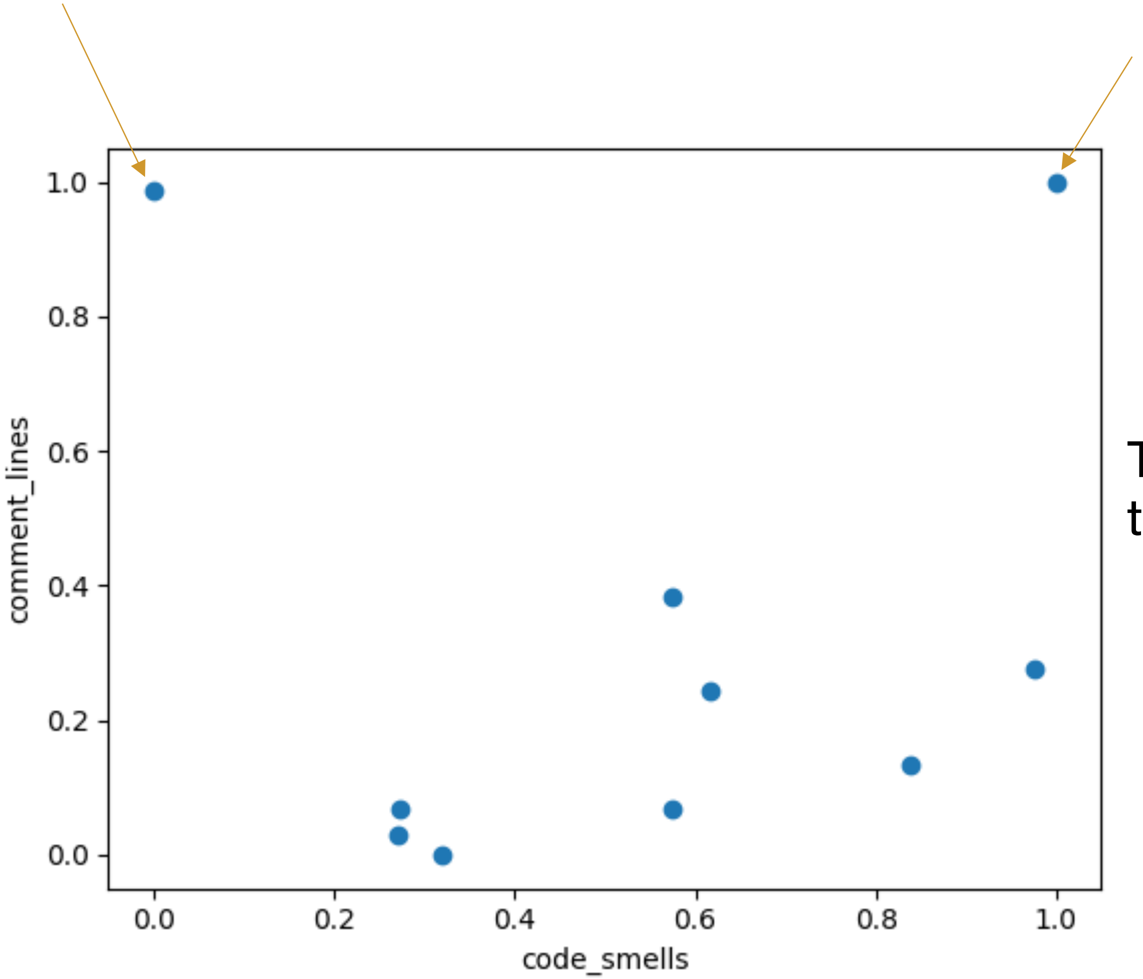
Name	Bugs	Classes	Code Smells	Cognitive Complexity	Comment Lines	Complexity	Duplicated Lines	File Complexity	Functions	Lines	Ncloc	Security Hotspots	Stars	Statements
AsciiTerminal	16	11	61	404	62	352	340	50.3	93	2247	1803	3	24	928
BatBat-Game	21	45	116	633	276	810	198	18.8	298	5271	3906	12	15	1999
gameserver	39	475	2126	2387	2749	4739	6956	11.7	3225	39201	26089	71	17	10047
GameShardingDb	21	69	403	730	771	748	50	11.5	379	6366	4286	7	43	1974
jcards	0	10	49	44	286	106	0	11.8	66	1250	509	1	31	163
lwjglbook-leg	197	1748	3446	9166	5755	20724	130553	14.4	13388	155460	116949	203	564	59028
OpenFighting	35	22	60	81	74	199	110	9	147	1559	1049	2	21	384
playn	64	398	1649	2478	6800	6168	2431	24.1	4412	46763	28753	10	239	11889
SypherEngine	3	67	233	346	622	731	164	9	462	5984	3814	4	43	1514
WraithEngine	1	101	19	307	2283	711	0	8.2	577	9081	4110	0	50	1289

Correlation Analysis of Attributes of Game Repositories



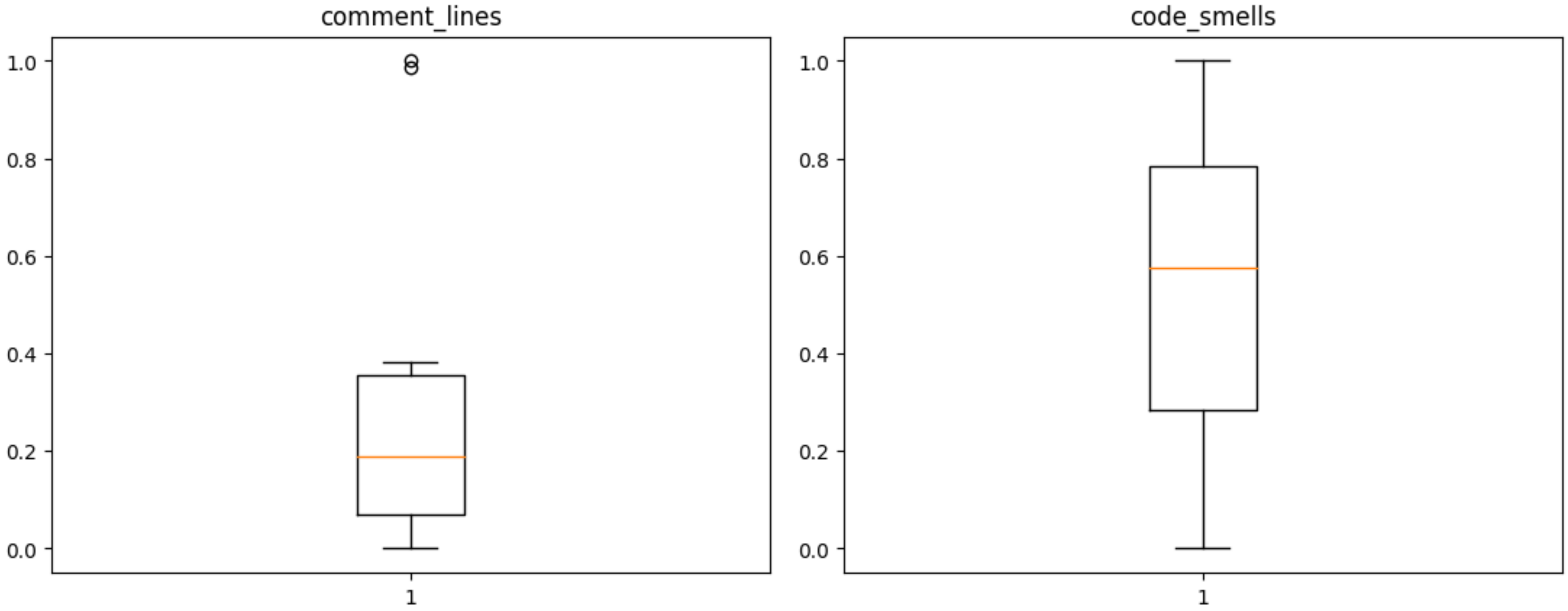
	blocker_violations	bugs	classes	code_smells	cognitive_complexity	comment_lines	complexity	critical_violations	duplicated_lines	file_complexity	functions	lines	major_violations	ncloc	security_hotspots	stars	statements	vulnerabilities
blocker_violations	1	-0.12	-0.5	0.3	0.73	-0.36	-0.1	-0.085	-0.2	-0.077	-0.62	-0.46	0.68	-0.19	0.24	-0.2	0.57	0.01
bugs	-0.12	1	0.073	-0.013	0.019	-0.39	-0.004	0.93	-0.098	0.16	0.038	-0.3	-0.21	-0.23	0.19	-0.22	-0.023	-0.15
classes	-0.5	0.073	1	0.1	-0.79	0.62	-0.39	0.2	-0.19	-0.38	0.76	0.67	0.0021	-0.11	-0.32	-0.13	-0.87	0.13
code_smells	0.3	-0.013	0.1	1	-0.034	0.073	0.16	0.19	-0.27	0.12	0.12	0.17	0.61	-0.23	0.31	-0.27	-0.23	0.31
cognitive_complexity	0.73	0.019	-0.79	-0.034	1	-0.4	0.091	-0.14	-0.19	0.5	-0.93	-0.48	0.078	-0.31	0.31	-0.32	0.69	-0.15
comment_lines	-0.36	-0.39	0.62	0.073	-0.4	1	0.049	-0.42	-0.41	0.13	0.48	0.98	-0.25	-0.3	-0.49	-0.2	-0.77	-0.17
complexity	-0.1	-0.004	-0.39	0.16	0.091	0.049	1	-0.11	-0.33	0.37	0.032	0.1	-0.29	-0.27	0.13	-0.14	0.048	-0.23
critical_violations	-0.085	0.93	0.2	0.19	-0.14	-0.42	-0.11	1	-0.091	-0.047	0.21	-0.33	0.071	-0.18	0.14	-0.2	-0.13	0.036
duplicated_lines	-0.2	-0.098	-0.19	-0.27	-0.19	-0.41	-0.33	-0.091	1	-0.22	-0.044	-0.39	-0.054	0.96	0.15	0.88	0.45	0.083
file_complexity	-0.077	0.16	-0.38	0.12	0.5	0.13	0.37	-0.047	-0.22	1	-0.45	0.19	-0.52	-0.38	0.17	-0.34	0.034	-0.24
functions	-0.62	0.038	0.76	0.12	-0.93	0.48	0.032	0.21	-0.044	-0.45	1	0.53	-0.013	0.12	-0.48	0.19	-0.77	0.11
lines	-0.46	-0.3	0.67	0.17	-0.48	0.98	0.1	-0.33	-0.39	0.19	0.53	1	-0.29	-0.3	-0.37	-0.22	-0.83	-0.11
major_violations	0.68	-0.21	0.0021	0.61	0.078	-0.25	-0.29	0.071	-0.054	-0.52	-0.013	-0.29	1	0.04	0.17	-0.046	0.13	0.47
ncloc	-0.19	-0.23	-0.11	-0.23	-0.31	-0.3	-0.27	-0.18	0.96	-0.38	0.12	-0.3	0.04	1	0.017	0.96	0.37	0.068
security_hotspots	0.24	0.19	-0.32	0.31	0.31	-0.49	0.13	0.14	0.15	0.17	-0.48	-0.37	0.17	0.017	1	-0.18	0.39	0.41
stars	-0.2	-0.22	-0.13	-0.27	-0.32	-0.2	-0.14	-0.2	0.88	-0.34	0.19	-0.22	-0.046	0.96	-0.18	1	0.35	-0.18
statements	0.57	-0.023	-0.87	-0.23	0.69	-0.77	0.048	-0.13	0.45	0.034	-0.77	-0.83	0.13	0.37	0.39	0.35	1	-0.16
vulnerabilities	0.01	-0.15	0.13	0.31	-0.15	-0.17	-0.23	0.036	0.083	-0.24	0.11	-0.11	0.47	0.068	0.41	-0.18	-0.16	1

Comment Lines vs Code Smells Graphs of Game Repositories

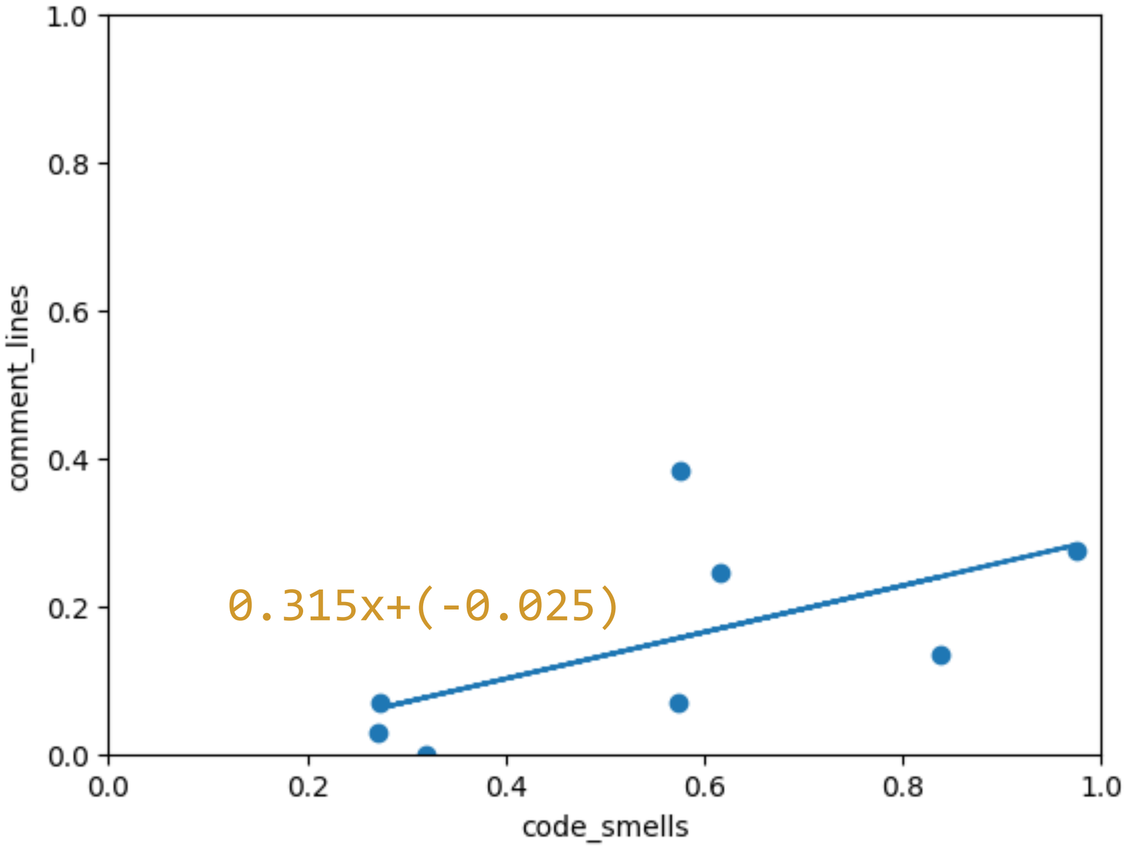


There are some outliers to remove

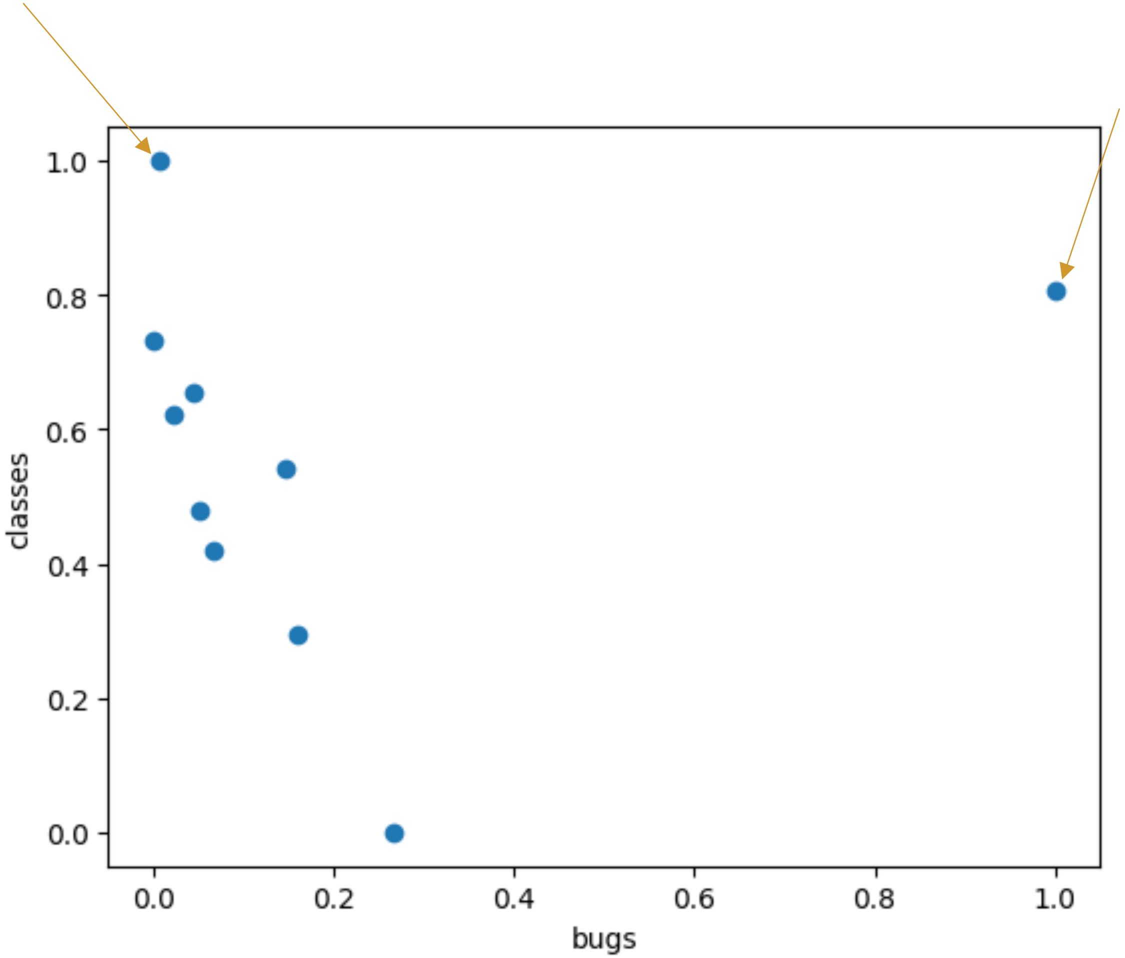
Box Plots of Code Smells and Comment Lines



Linear Regression of Comment Lines vs Code Smells After Removing the Outliers

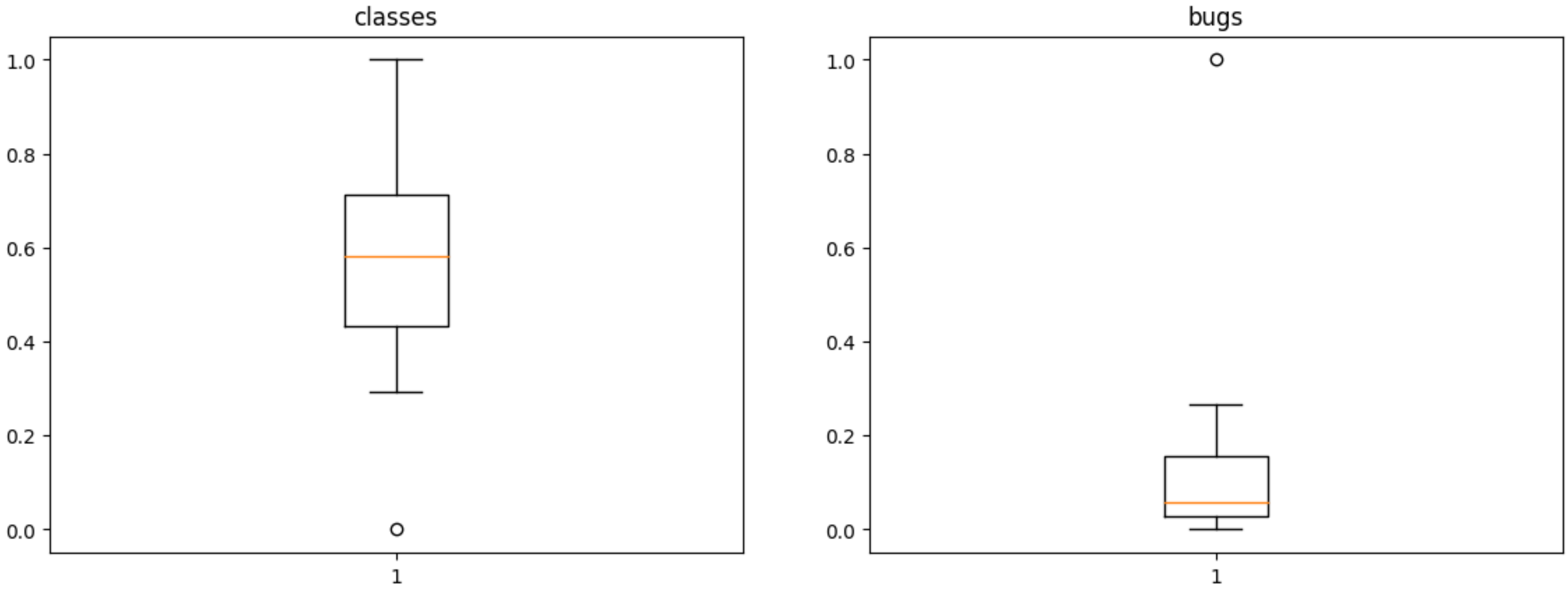


Number of Bugs vs Number of Classes Graphs of Game Repositories

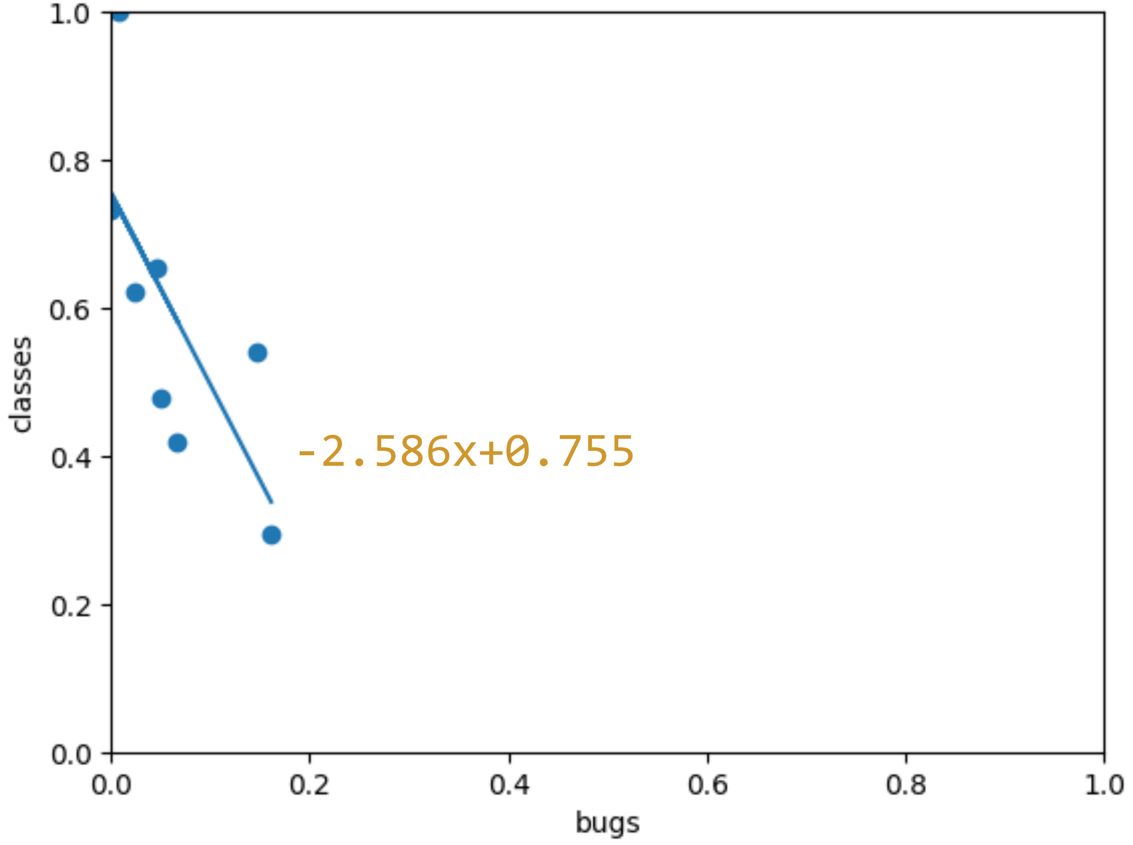


There are some outliers to remove

Box Plots of Number of Classes and Bugs



Linear Regression of Number of Bugs vs Number of Classes After Removing the Outliers



Conclusion

- When correlation diagrams compared, it seems that game topic is much more different than e-commerce topic and e-commerce is behaves as expected unlike game topic.
- This may be interpreted as e-commerce field developers follow certain paths, patterns and rules that are familiar to the software developers in general unlike the game field.
- The link will navigate to the GitHub page of the project where all the codes that have been used can be found:

  `«https://github.com/dogababacan/RepositoryInspectionWithSonarqube.git»`