# Appendix A

# List of Publications

## A.1 Journal Papers

1. **Rakesh Kumar** and Amrita Chaturvedi. "Software Bug Prediction Using Reward-Based Weighted Majority Voting Ensemble Technique." **IEEE Transactions on Reliability**, vol. 73.1, pp. 726-740
   doi: 10.1109/TR.2023.3295598. (2023). (Published) **(SCI/SCIE, IF: 5.9)**

2. **Rakesh Kumar**, Amrita Chaturvedi, and Lakshmanan Kailasam. "An unsupervised software fault prediction approach using threshold derivation." **IEEE Transactions on Reliability**, vol. 71.2, pp. 911-932.
   (doi: 10.1109/TR.2022.3151125).(2022) (Published) **(SCI/SCIE, IF: 5.9)**

3. **Rakesh Kumar** and Amrita Chaturvedi: An Unsupervised Software Bug Count Prediction Model Based on Selected Software Metrics, **Expert Systems with Applications**, Elsevier (Under Review), **(SCI/SCIE, IF: 8.5)**

4. **Rakesh Kumar** and Amrita Chaturvedi: An Unsupervised Software Defect Prediction System in Functional Paradigm, **Expert Systems with Applications**, Elsevier, (Under Review) **(SCI/SCIE, IF: 8.5)**

## A.2 Conference Papers

1. **Rakesh Kumar** and Amrita Chaturvedi. "Software Fault Prediction Using Data Mining Techniques on Software Metrics." (Proceedings of International Conference on Machine Learning and Big Data Analytics (**ICMLBDA**) 2021), (**IIT Patna**). **Springer** International Publishing, 2022. https://doi.org/10.1007/978-3-030-82469-3_27

2. **Rakesh Kumar** and Amrita Chaturvedi. "A Framework for Software Defect Prediction Using Optimal Hyper-Parameters of Deep Neural Network." In: The 29$^{th}$ International Conference on Neural Information Processing (**29$^{th}$ ICONIP 2022**), **IIT Indore**, India, **Springer** International Publishing, 2022.

3. **Rakesh Kumar** and Amrita Chaturvedi. "Software Bug Count Prediction Model Based on Software Source Code," In: 14th International Conference on Computing Communication and Networking Technologies (**14$^{th}$ ICCCNT**), **IIT Delhi**, India, 2023, pp. 1-6, published in **IEEE** Digital Library Xplore®, doi: 10.1109/ICCCNT56998.2023.10307457.

# Appendix B

# Sample Datasets and Other Results

The datasets used in the chapters are from open-source projects and may not be generalized well to proprietary or industrial projects with different characteristics. Therefore, we conducted an experiment using eight additional datasets: Eclipse [226, 243, 244], which has three versions (2.0, 2.1, 3.0), and Android [183, 245], which has five versions (2013_1, 2013_2 ,2013_3, 2014_1, 2014_2). The Eclipse and Android projects were chosen due to their alignment with industrial/commercial systems in terms of size and complexity, following industry-standard development processes. Each Eclipse dataset[1] release contains file name, pre-release and post-release bugs,

[1]https://www.st.cs.uni-saarland.de/softevo/

TABLE B.1: Sample bug dataset (First 20 modules of Tomcat project)

| name | version | name | wmc | dit | noc | cbo | rfc | lcom | ca | ce | npm | lcom3 | loc | dam | moa | mfa | cam | ic | cbm | amc | max_cc | avg_cc | bug |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Tomcat | 6.0.389418 | org.apache.coyote.http11.filters.VoidOutputFilter | 8.00 | 1.00 | 0.00 | 6.00 | 14.00 | 26.00 | 2.00 | 0.00 | 7.00 | 1.00 | 39.00 | 1.00 | 2.00 | 0.00 | 0.39 | 0.00 | 0.00 | 3.50 | 1.00 | 0.75 | 0 |
| Tomcat | 6.0.389418 | org.apache.el.parser.AstGreaterThan | 2.00 | 4.00 | 0.00 | 4.00 | 5.00 | 1.00 | 1.00 | 0.00 | 2.00 | 2.00 | 36.00 | 0.00 | 0.00 | 0.98 | 0.67 | 1.00 | 1.00 | 17.00 | 1.00 | 0.50 | 0 |
| Tomcat | 6.0.389418 | org.apache.coyote.Request | 56.00 | 1.00 | 0.00 | 49.00 | 89.00 | 1310.00 | 40.00 | 0.00 | 56.00 | 0.96 | 583.00 | 1.00 | 24.00 | 0.00 | 0.14 | 0.00 | 0.00 | 8.82 | 5.00 | 1.20 | 0 |
| Tomcat | 6.0.389418 | javax.el.MethodNotFoundException | 4.00 | 5.00 | 1.00 | 0.00 | 8.00 | 6.00 | 0.00 | 0.00 | 4.00 | 2.00 | 20.00 | 0.00 | 0.00 | 1.00 | 0.67 | 0.00 | 0.00 | 4.00 | 0.00 | 0.00 | 0 |
| Tomcat | 6.0.389418 | org.apache.naming.EjbRef | 3.00 | 2.00 | 0.00 | 3.00 | 8.00 | 3.00 | 3.00 | 0.00 | 3.00 | 1.50 | 73.00 | 0.00 | 0.00 | 0.94 | 0.83 | 0.00 | 0.00 | 22.00 | 3.00 | 1.00 | 0 |
| Tomcat | 6.0.389418 | org.apache.jasper.compiler.Compiler | 16.00 | 1.00 | 2.00 | 38.00 | 91.00 | 58.00 | 23.00 | 0.00 | 13.00 | 0.78 | 717.00 | 0.89 | 7.00 | 0.00 | 0.28 | 0.00 | 0.00 | 43.25 | 15.00 | 1.94 | 1 |
| Tomcat | 6.0.389418 | javax.servlet.ServletContextAttributeListener | 3.00 | 1.00 | 0.00 | 0.00 | 3.00 | 3.00 | 0.00 | 0.00 | 3.00 | 2.00 | 3.00 | 0.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 0.00 | 1.00 | 1.00 | 0 |
| Tomcat | 6.0.389418 | org.apache.el.MethodExpressionLiteral | 10.00 | 3.00 | 0.00 | 3.00 | 24.00 | 3.00 | 1.00 | 0.00 | 10.00 | 0.48 | 114.00 | 1.00 | 0.00 | 0.43 | 0.21 | 1.00 | 1.00 | 10.10 | 3.00 | 1.00 | 0 |
| Tomcat | 6.0.389418 | org.apache.naming.ResourceEnvRef | 3.00 | 2.00 | 0.00 | 2.00 | 7.00 | 3.00 | 2.00 | 0.00 | 3.00 | 1.50 | 30.00 | 0.00 | 0.00 | 0.94 | 0.83 | 0.00 | 0.00 | 8.67 | 3.00 | 1.00 | 0 |
| Tomcat | 6.0.389418 | org.apache.catalina.mbeans.StandardServerMBean | 3.00 | 2.00 | 0.00 | 5.00 | 8.00 | 3.00 | 0.00 | 0.00 | 2.00 | 1.00 | 27.00 | 1.00 | 0.00 | 0.98 | 1.00 | 0.00 | 0.00 | 7.67 | 1.00 | 0.33 | 0 |
| Tomcat | 6.0.389418 | org.apache.naming.NamingService | 11.00 | 2.00 | 0.00 | 1.00 | 25.00 | 35.00 | 0.00 | 0.00 | 11.00 | 0.65 | 283.00 | 1.00 | 0.00 | 0.41 | 0.32 | 0.00 | 0.00 | 24.36 | 2.00 | 1.18 | 0 |
| Tomcat | 6.0.389418 | org.apache.catalina.startup.WebRuleSet | 4.00 | 2.00 | 0.00 | 12.00 | 22.00 | 2.00 | 1.00 | 0.00 | 4.00 | 0.58 | 1700.00 | 1.00 | 3.00 | 0.50 | 0.50 | 1.00 | 1.00 | 423.00 | 1.00 | 0.50 | 2 |
| Tomcat | 6.0.389418 | org.apache.naming.NameNotFoundException | 6.00 | 5.00 | 0.00 | 1.00 | 7.00 | 15.00 | 1.00 | 0.00 | 6.00 | 2.00 | 14.00 | 0.00 | 0.00 | 0.86 | 0.46 | 1.00 | 1.00 | 1.33 | 1.00 | 0.83 | 0 |
| Tomcat | 6.0.389418 | javax.servlet.jsp.tagext.JspTag | 0.00 | 1.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 2.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0 |
| Tomcat | 6.0.389418 | org.apache.tomcat.util.net.jsse.JSSE13SocketFactory | 6.00 | 3.00 | 0.00 | 8.00 | 33.00 | 3.00 | 1.00 | 0.00 | 1.00 | 0.40 | 177.00 | 1.00 | 0.00 | 0.83 | 0.40 | 1.00 | 6.00 | 28.33 | 1.00 | 0.83 | 0 |
| Tomcat | 6.0.389418 | org.apache.catalina.realm.MemoryRuleSet | 3.00 | 2.00 | 0.00 | 6.00 | 9.00 | 1.00 | 2.00 | 0.00 | 3.00 | 0.00 | 35.00 | 1.00 | 0.00 | 0.67 | 0.56 | 1.00 | 1.00 | 10.33 | 1.00 | 0.33 | 0 |
| Tomcat | 6.0.389418 | org.apache.tomcat.jni.Pool | 14.00 | 1.00 | 0.00 | 3.00 | 15.00 | 91.00 | 3.00 | 0.00 | 14.00 | 2.00 | 17.00 | 0.00 | 0.00 | 0.00 | 0.29 | 0.00 | 0.00 | 0.21 | 1.00 | 0.93 | 0 |
| Tomcat | 6.0.389418 | javax.servlet.jsp.JspEngineInfo | 2.00 | 1.00 | 1.00 | 0.00 | 3.00 | 1.00 | 0.00 | 0.00 | 2.00 | 2.00 | 5.00 | 0.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 1.50 | 1.00 | 0.50 | 0 |
| Tomcat | 6.0.389418 | org.apache.jk.common.HandlerRequest | 22.00 | 2.00 | 0.00 | 20.00 | 145.00 | 127.00 | 0.00 | 0.00 | 15.00 | 0.87 | 1226.00 | 0.86 | 1.00 | 0.55 | 0.31 | 2.00 | 2.00 | 54.09 | 10.00 | 2.32 | 0 |

TABLE B.2: Eclipse and Android bug datasets descriptions

| Group | Datasets | #Metrics | #Module | #Bug | Bug% |
|---|---|---|---|---|---|
| Eclipse | Eclipse-2.0 | 199 | 6729 | 975 | 14.49 |
| | Eclipse-2.1 | 199 | 7888 | 854 | 10.83 |
| | Eclipse-3.0 | 199 | 10593 | 1568 | 14.80 |
| Android | A_2013_1 | 107 | 73 | 35 | 47.95 |
| | A_2013_2 | 107 | 98 | 22 | 22.45 |
| | A_2013_3 | 107 | 109 | 2 | 1.83 |
| | A_2014_1 | 107 | 116 | 8 | 6.90 |
| | A_2014_2 | 107 | 119 | 2 | 1.68 |

complexity metrics, and abstract syntax tree metrics, including number of nodes, types, and frequency. Android datasets[2] is extracted from Android-Universal-Image-Loader software projects[3]. Eclipse and Android datasets descriptions in terms of number of metrics, modules, bugs, and bug% is given in Table B.2. The experimental results of the SBP/SBCV prediction models presented in Chapters 3, 4, and 5 are shown in Tables B.3, B.4, and B.5, respectively.

[2]https://www.inf.u-szeged.hu/ ferenc/papers/GitHubBugDataSet/GitHubBugDataSet.zip
[3]https://github.com/nostra13/Android-Universal-Image-Loader

TABLE B.3: Performance of SBP techniques (Chapter 3) over Eclipse and Android datasets in terms of accuracy, FM, MCC

| Groups | Projects | Accuracy (%) | | | | | | F-measure | | | | | | MCC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | NB | SVM | KNN | RF | C50 | SMV | WMV | NB | SVM | KNN | RF | C50 | SMV | WMV | NB | SVM | KNN | RF | C50 | SMV | WMV |
| ECLIPSE | Eclipse-2.0 | 95.96 | 95.70 | 79.44 | 95.75 | 95.98 | 96.78 | 97.78 | 0.99 | 0.98 | 0.85 | 0.98 | 0.98 | 0.98 | 0.98 | 0.88 | 0.87 | 0.58 | 0.87 | 0.87 | 0.87 | 0.90 |
| | Eclipse-2.1 | 96.46 | 94.89 | 78.15 | 96.67 | 95.99 | 96.67 | 97.67 | 0.98 | 0.97 | 0.84 | 0.97 | 0.96 | 0.97 | 0.97 | 0.87 | 0.86 | 0.56 | 0.86 | 0.86 | 0.86 | 0.89 |
| | Eclipse-3.0 | 95.90 | 95.63 | 78.76 | 95.68 | 95.99 | 96.85 | 97.85 | 0.98 | 0.98 | 0.84 | 0.97 | 0.98 | 0.98 | 0.99 | 0.86 | 0.87 | 0.57 | 0.88 | 0.88 | 0.88 | 0.90 |
| | AVG | 96.11 | 95.41 | 78.78 | 96.04 | 95.99 | 96.77 | **97.77** | **0.98** | 0.97 | 0.84 | 0.97 | 0.97 | **0.98** | **0.98** | 0.87 | 0.87 | 0.57 | 0.87 | 0.87 | 0.87 | **0.90** |
| Android | A_2013_1 | 86.34 | 76.46 | 67.39 | 77.89 | 66.83 | 86.30 | 87.93 | 0.89 | 0.78 | 0.62 | 0.82 | 0.73 | 0.88 | 0.90 | 0.73 | 0.59 | 0.47 | 0.65 | 0.51 | 0.67 | 0.74 |
| | A_2013_2 | 95.78 | 79.22 | 63.65 | 95.07 | 89.04 | 95.92 | 95.92 | 0.95 | 0.79 | 0.71 | 0.95 | 0.89 | 0.96 | 0.96 | 0.92 | 0.67 | 0.43 | 0.92 | 0.80 | 0.88 | 0.92 |
| | A_2013_3 | 99.55 | 99.64 | 76.19 | 99.90 | 96.89 | 99.91 | 99.99 | 0.99 | 1.00 | 0.65 | 1.00 | 0.97 | 1.00 | 1.00 | 1.00 | 1.00 | 0.65 | 1.00 | 0.98 | 1.00 | 1.00 |
| | A_2014_1 | 97.39 | 88.21 | 77.08 | 98.98 | 98.09 | 99.14 | 99.14 | 0.97 | 0.88 | 0.67 | 0.99 | 0.98 | 0.99 | 0.99 | 0.98 | 0.78 | 0.61 | 1.00 | 0.98 | 0.95 | 0.98 |
| | A_2014_2 | 99.99 | 99.17 | 88.42 | 98.26 | 97.30 | 99.16 | 99.16 | 0.99 | 0.99 | 0.88 | 0.98 | 0.97 | 0.99 | 0.99 | 0.98 | 0.98 | 0.80 | 0.98 | 0.98 | 0.98 | 0.98 |
| | AVG | 95.81 | 88.54 | 74.54 | 94.04 | 89.63 | 96.10 | **96.43** | 0.96 | 0.89 | 0.70 | 0.95 | 0.91 | 0.96 | **0.97** | 0.92 | 0.80 | 0.59 | 0.91 | 0.85 | 0.90 | **0.93** |

TABLE B.4: Performance of different SBP techniques (Chapter 4) over Eclipse datasets (larger datasets)

| Projects Name | Accuracy % | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Supervised Methods | | | | | Unsupervised Methods | | | | Threshold Based Methods | | | | |
| | NB | SVM | KNN | RF | C50 | KMS | NGC | MBC | HC | CLAMI | CLAMI+ | ACL | TCL | TCLP |
| Eclipse-2.0 | 85.05 | 85.43 | 85.09 | 88.33 | 87.84 | 86.65 | 86.75 | 76.48 | 85.92 | 50.36 | 52.40 | 61.49 | 84.80 | 86.23 |
| Eclipse-2.1 | 87.11 | 87.30 | 86.20 | 90.00 | 88.56 | 88.17 | 87.26 | 73.73 | 86.38 | 42.77 | 44.98 | 54.04 | 84.74 | 87.75 |
| Eclipse-3.0 | 84.18 | 85.40 | 85.87 | 86.18 | 85.50 | 85.70 | 82.30 | 74.73 | 85.30 | 54.72 | 57.54 | 62.31 | 83.17 | 86.65 |
| AVG | 85.45 | 86.04 | 85.72 | **88.17** | 87.30 | 86.84 | 85.44 | 74.98 | 85.86 | 49.28 | 51.64 | 59.28 | 84.23 | 86.88 |

| | F-measure | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Supervised Methods | | | | | Unsupervised Methods | | | | Threshold Based Methods | | | | |
| | NB | SVM | KNN | RF | C50 | KMS | NGC | MBC | HC | CLAMI | CLAMI+ | ACL | TCL | TCLP |
| Eclipse-2.0 | 0.91 | 0.91 | 0.90 | 0.92 | 0.92 | 0.32 | 0.39 | 0.78 | 0.80 | 0.60 | 0.67 | 0.72 | 0.91 | 0.92 |
| Eclipse-2.1 | 0.91 | 0.93 | 0.92 | 0.92 | 0.92 | 0.28 | 0.43 | 0.83 | 0.84 | 0.54 | 0.73 | 0.74 | 0.91 | 0.92 |
| Eclipse-3.0 | 0.90 | 0.90 | 0.91 | 0.91 | 0.91 | 0.26 | 0.40 | 0.76 | 0.82 | 0.65 | 0.65 | 0.73 | 0.90 | 0.91 |
| AVG | 0.91 | 0.91 | 0.91 | **0.92** | **0.92** | 0.29 | 0.41 | 0.79 | 0.82 | 0.60 | 0.69 | 0.73 | 0.91 | **0.92** |

| | MCC | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Supervised Methods | | | | | Unsupervised Methods | | | | Threshold Based Methods | | | | |
| | NB | SVM | KNN | RF | C50 | KMS | NGC | MBC | HC | CLAMI | CLAMI+ | ACL | TCL | TCLP |
| Eclipse-2.0 | 0.32 | 0.32 | 0.29 | 0.42 | 0.38 | 0.31 | 0.30 | 0.26 | 0.27 | 0.26 | 0.27 | 0.31 | 0.33 | 0.34 |
| Eclipse-2.1 | 0.25 | 0.26 | 0.30 | 0.29 | 0.33 | 0.25 | 0.24 | 0.24 | 0.24 | 0.15 | 0.19 | 0.24 | 0.24 | 0.26 |
| Eclipse-3.0 | 0.27 | 0.28 | 0.28 | 0.27 | 0.27 | 0.25 | 0.28 | 0.26 | 0.26 | 0.25 | 0.27 | 0.25 | 0.28 | 0.30 |
| AVG | 0.28 | 0.29 | 0.29 | **0.33** | **0.33** | 0.27 | 0.27 | 0.26 | 0.26 | 0.22 | 0.25 | 0.27 | 0.29 | 0.29 |

TABLE B.5: Results of MTB/MTBP and 8 ML models (Chapter 5) over Eclipse and Android DSs

| Groups | | Mean Absolute Error (MAE) | | | | | | | | | Mean Relative Error (MRE) | | | | | | | | | Prediction at level l=0.3 error (Pred(l)_Error) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Datasets | MTB | MTBP | LM | MLP | GA | DTR | SVR | KNN | BRR | NBR | MTB | MTBP | LM | MLP | GA | DTR | SVR | KNN | BRR | NBR | MTB | MTBP | LM | MLP | GA | DTR | SVR | KNN | BRR | NBR |
| ECLIPSE | Eclipse-2.0 | 0.19 | 0.16 | 0.27 | 0.24 | 0.26 | 0.27 | 0.20 | 0.28 | 0.29 | 0.28 | 0.10 | 0.06 | 0.18 | 0.11 | 0.17 | 0.18 | 0.09 | 0.18 | 0.19 | 0.18 | 0.13 | 0.10 | 0.20 | 0.10 | 0.27 | 0.27 | 0.10 | 0.24 | 0.18 | 0.20 |
| | Eclipse-2.1 | 0.36 | 0.32 | 0.43 | 0.43 | 0.46 | 0.41 | 0.34 | 0.42 | 0.45 | 0.43 | 0.21 | 0.14 | 0.31 | 0.41 | 0.33 | 0.29 | 0.16 | 0.21 | 0.31 | 0.31 | 0.31 | 0.26 | 0.45 | 0.66 | 0.26 | 0.29 | 0.21 | 0.26 | 0.47 | 0.45 |
| | Eclipse-3.0 | 0.21 | 0.11 | 0.19 | 0.18 | 0.18 | 0.18 | 0.14 | 0.18 | 0.18 | 0.19 | 0.16 | 0.05 | 0.13 | 0.26 | 0.12 | 0.13 | 0.07 | 0.11 | 0.12 | 0.13 | 0.19 | 0.09 | 0.13 | 0.08 | 0.17 | 0.09 | 0.08 | 0.13 | 0.13 | 0.13 |
| | A_2013_1 | 0.53 | 0.44 | 0.61 | 0.63 | 0.69 | 0.62 | 0.55 | 0.63 | 0.66 | 0.60 | 0.33 | 0.18 | 0.38 | 0.24 | 0.40 | 0.43 | 0.21 | 0.36 | 0.42 | 0.38 | 0.40 | 0.29 | 0.51 | 0.38 | 0.96 | 0.82 | 0.27 | 0.56 | 0.86 | 0.51 |
| | A_2013_2 | 0.36 | 0.24 | 0.38 | 0.34 | 0.35 | 0.34 | 0.31 | 0.32 | 0.34 | 0.31 | 0.29 | 0.14 | 0.22 | 0.21 | 0.19 | 0.14 | 0.11 | 0.21 | 0.22 | 0.22 | 0.32 | 0.21 | 0.35 | 0.22 | 0.12 | 0.09 | 0.13 | 0.23 | 0.33 | 0.35 |
| | A_2013_3 | 0.36 | 0.08 | 0.06 | 0.06 | 0.04 | 0.04 | 0.03 | 0.04 | 0.04 | 0.05 | 0.35 | 0.07 | 0.03 | 0.03 | 0.03 | 0.03 | 0.02 | 0.02 | 0.03 | 0.03 | 0.36 | 0.08 | 0.02 | 0.02 | 0.02 | 0.02 | 0.02 | 0.02 | 0.02 | 0.02 |
| | A_2014_1 | 0.42 | 0.20 | 0.14 | 0.08 | 0.15 | 0.14 | 0.10 | 0.13 | 0.13 | 0.13 | 0.40 | 0.17 | 0.09 | 0.03 | 0.09 | 0.09 | 0.05 | 0.09 | 0.09 | 0.09 | 0.37 | 0.19 | 0.06 | 0.06 | 0.06 | 0.06 | 0.06 | 0.09 | 0.06 | 0.06 |
| | A_2014_2 | 0.33 | 0.11 | 0.06 | 0.04 | 0.03 | 0.02 | 0.03 | 0.03 | 0.03 | 0.05 | 0.32 | 0.11 | 0.04 | 0.01 | 0.02 | 0.02 | 0.02 | 0.02 | 0.03 | 0.04 | 0.29 | 0.10 | 0.02 | 0.02 | 0.02 | 0.02 | 0.02 | 0.02 | 0.02 | 0.02 |
| | AVG | 0.34 | **0.21** | 0.27 | 0.25 | 0.27 | 0.25 | **0.21** | 0.25 | 0.27 | 0.26 | 0.27 | 0.11 | 0.17 | 0.16 | 0.17 | 0.16 | **0.09** | 0.15 | 0.18 | 0.17 | 0.35 | 0.18 | 0.19 | 0.14 | 0.24 | 0.20 | **0.10** | 0.18 | 0.26 | 0.19 |

TABLE B.6: Effect size using CohenD between WMV and others models (Chapter 3)

| WMV | NB | SVM | KNN | RF | C50 | SMV |
|---|---|---|---|---|---|---|
| Accuracy | 0.75 | 1.60 | 2.15 | 0.42 | 0.66 | 0.14 |
| FM | 0.82 | 1.67 | 1.90 | 0.47 | 0.67 | 0.15 |
| MCC | 0.07 | 0.30 | 0.57 | 0.05 | 0.07 | 0.16 |

TABLE B.7: Effect size using CohenD between TCLP and others models (Chapter 4)

| TCLP | NB | SVM | KNN | RF | C50 | KMS | NGC | MBC | HC | CLA | CLAMI | CLAMI+ | ACL | TCL |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Accuracy | 0.49 | 0.54 | 0.48 | 0.58 | 0.56 | 0.02 | 0.00 | 0.98 | 0.04 | 1.98 | 1.92 | 1.95 | 1.82 | 0.02 |
| FM | 0.28 | 0.25 | 0.12 | 0.27 | 0.24 | 0.92 | 1.31 | 1.15 | 0.11 | 2.04 | 1.93 | 1.89 | 1.76 | 0.01 |
| MCC | 0.81 | 0.22 | 0.14 | 0.65 | 0.70 | 0.68 | 0.48 | 0.51 | 0.95 | 0.23 | 0.15 | 0.06 | 0.05 | 0.04 |

TABLE B.8: Effect size using CohenD between UMV and others models (Chapter 6)

| UMV | CLAMI | CLAMI+ | ACL | TCLP | SQRT | CBRT | KMS | NGC | MBC | HC | NB | SVM | KNN | RF | C50 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Accuracy | 2.06 | 4.82 | 0.64 | 0.71 | 0.81 | 0.88 | 0.83 | 0.64 | 1.82 | 1.67 | 0.16 | 0.93 | 0.32 | 2.30 | 1.51 |
| FM | 2.12 | 4.80 | 0.83 | 0.94 | 1.01 | 1.01 | 2.38 | 0.79 | 1.64 | 1.70 | 0.31 | 0.94 | 0.19 | 1.84 | 1.00 |
| MCC | 1.93 | 3.67 | 0.56 | 0.25 | 0.40 | 0.23 | 0.69 | 0.45 | 0.94 | 1.72 | 1.13 | 1.47 | 0.78 | 2.48 | 2.10 |

TABLE B.9: Cross-version (Training on previous version and testing on current version) performance of different SBP models (Chapter 3)

| Datasets | | Accuracy (%) | | | | | | | F-measure | | | | | | | MCC | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Training | Testing | NB | SVM | KNN | RF | C50 | SMV5 | WMV5 | NB | SVM | KNN | RF | C50 | SMV5 | WMV5 | NB | SVM | KNN | RF | C50 | SMV5 | WMV5 |
| Ant-1.6 | Ant-1.7 | 79.87 | 78.12 | 80.13 | 77.72 | 79.33 | 79.60 | 81.17 | 0.87 | 0.86 | 0.88 | 0.85 | 0.87 | 0.87 | 0.89 | 0.43 | 0.35 | 0.39 | 0.42 | 0.42 | 0.39 | 0.39 |
| Camel-1.4 | Camel-1,6 | 77.93 | 78.03 | 79.90 | 80.62 | 80.93 | 79.59 | 80.53 | 0.87 | 0.87 | 0.89 | 0.89 | 0.89 | 0.88 | 0.89 | 0.16 | 0.12 | 0.15 | 0.20 | 0.20 | 0.17 | 0.21 |
| Forrest-0.7 | Forrest-0.8 | 59.38 | 65.63 | 93.75 | 81.25 | 68.75 | 68.75 | 73.10 | 0.72 | 0.78 | 0.97 | 0.89 | 0.80 | 0.80 | 0.94 | 0.27 | 0.31 | 0.00 | 0.45 | 0.45 | 0.33 | 0.41 |
| Ivy-1.4 | Ivy-2.0 | 89.49 | 87.78 | 88.92 | 89.20 | 87.50 | 88.92 | 92.93 | 0.94 | 0.93 | 0.94 | 0.94 | 0.93 | 0.94 | 0.94 | 0.39 | 0.25 | 0.26 | 0.39 | 0.39 | 0.28 | 0.39 |
| Jedit-4.2 | Jedit-4.3 | 91.67 | 91.46 | 93.50 | 90.65 | 91.67 | 92.68 | 90.19 | 0.96 | 0.96 | 0.97 | 0.95 | 0.96 | 0.96 | 0.90 | 0.16 | 0.16 | 0.20 | 0.15 | 0.15 | 0.38 | 0.22 |
| Log4j-1.1 | Log4j-1.2 | 32.68 | 38.05 | 25.37 | 29.76 | 25.37 | 29.76 | 32.66 | 0.18 | 0.17 | 0.16 | 0.17 | 0.15 | 0.17 | 0.20 | 0.13 | 0.09 | 0.09 | 0.12 | 0.12 | 0.12 | 0.12 |
| Lucene-2.2 | Lucene-2.4 | 54.71 | 59.12 | 47.94 | 54.41 | 53.53 | 53.53 | 58.26 | 0.45 | 0.52 | 0.27 | 0.44 | 0.40 | 0.61 | 0.61 | 0.25 | 0.33 | 0.17 | 0.24 | 0.24 | 0.24 | 0.32 |
| Poi-2.5 | Poi-3.0 | 58.14 | 51.36 | 42.53 | 53.62 | 42.76 | 53.85 | 59.39 | 0.54 | 0.44 | 0.21 | 0.45 | 0.23 | 0.60 | 0.61 | 0.34 | 0.20 | 0.13 | 0.29 | 0.29 | 0.29 | 0.29 |
| Prop-5 | Prop-6 | 79.24 | 79.39 | 89.09 | 81.97 | 83.64 | 82.58 | 92.24 | 0.88 | 0.88 | 0.94 | 0.90 | 0.91 | 0.90 | 0.93 | 0.18 | 0.08 | 0.22 | 0.17 | 0.17 | 0.15 | 0.20 |
| Synapse-1.1 | Synapse-1.2 | 71.48 | 65.23 | 71.88 | 72.27 | 70.31 | 71.48 | 73.42 | 0.79 | 0.73 | 0.80 | 0.80 | 0.79 | 0.79 | 0.72 | 0.34 | 0.24 | 0.33 | 0.35 | 0.35 | 0.34 | 0.36 |
| Velocity-1.5 | Velocity-1.6 | 67.69 | 65.94 | 65.50 | 67.25 | 66.81 | 67.69 | 69.39 | 0.37 | 0.30 | 0.22 | 0.29 | 0.22 | 0.79 | 0.78 | 0.20 | 0.14 | 0.10 | 0.17 | 0.17 | 0.18 | 0.19 |
| Xalan-2.6 | Xalan-2.7 | 21.67 | 19.14 | 18.37 | 26.07 | 25.52 | 20.24 | 28.66 | 0.03 | 0.03 | 0.03 | 0.03 | 0.03 | 0.03 | 0.03 | 0.06 | 0.05 | 0.05 | 0.06 | 0.06 | 0.05 | 0.06 |
| Xerces-1.3 | Xerces-1.4 | 36.73 | 33.67 | 30.61 | 38.27 | 34.18 | 34.01 | 39.72 | 0.44 | 0.43 | 0.41 | 0.45 | 0.44 | 0.45 | 0.93 | 0.19 | 0.14 | 0.06 | 0.21 | 0.21 | 0.15 | 0.22 |
| | Average | 63.13 | 62.53 | 63.65 | 64.85 | 62.33 | 63.28 | **67.05** | 0.62 | 0.61 | 0.59 | 0.62 | 0.59 | 0.68 | **0.72** | 0.24 | 0.19 | 0.16 | 0.25 | 0.25 | 0.24 | **0.26** |

## B.1 Distinguishing properties of functional paradigms

Functional programming is a paradigm that significantly emphasizes the utilization of pure functions and immutable data to craft robust and predictable software. This paradigm is distinguished by several key properties [246, 247].

- First-Class and Higher-Order Functions: Functions in functional programming are treated as first-class citizens. This means they can be assigned to variables, passed as arguments, and returned by other functions, enhancing flexibility and power through the use of higher-order functions. These can accept other functions as inputs or return them as outputs, facilitating complex functional constructs.

- Pure Functions: Functions within this paradigm are inherently pure, avoiding any side effects such as modifying global or static variables. They consistently produce the same output for the same set of inputs, thereby enhancing the reliability and predictability of code behavior.

- Immutability: In functional programming, data is immutable, meaning once it is created, it cannot be altered. This approach avoids the complications associated with shared mutable state and enhances code safety and predictability.

- Referential Transparency: An expression exhibits referential transparency if it can be substituted with its value without altering the behavior of the program. This is enabled by the use of pure functions and immutable data, allowing compilers to optimize more easily and enhance predictability.

- Recursion: Given the immutable nature of data, functional programming often relies on recursion to handle loops or repetitive operations. Compilers for functional languages typically optimize tail recursion to improve the performance of these recursive functions.

- Function Composition: This paradigm often employs the composition of simpler functions to build more complex functionalities. Function composition

fosters modularity and reusability, making the code simpler to maintain and understand.

- Lazy Evaluation: Some functional programming languages implement lazy evaluation, delaying the computation of expressions until their values are needed. This technique can significantly boost performance, particularly when dealing with potentially infinite data structures or operations.

- Type Systems: Many functional languages are equipped with robust, static type systems that include features like type inference. These systems help catch errors at compile time, enhancing code safety and performance while minimizing the burden of excessive type annotations.

# Bibliography


[1] L. Qiao, X. Li, Q. Umer, and P. Guo, "Deep learning based software defect prediction," *Neurocomputing*, 2020, v. 385, pp. 100–110.

[2] D. Bowes, T. Hall, and J. Petrić, "Software defect prediction: do different classifiers find the same defects?" *Software Quality Journal*, 2018, v. 26, n. 2, pp. 525–552.

[3] V. A. Prakash, D. Ashoka, and V. Aradya, "Application of data mining techniques for defect detection and classification," in *Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2014*. Springer, 2015, pp. 387–395.

[4] S. Planning, "The economic impacts of inadequate infrastructure for software testing," *National Institute of Standards and Technology*, 2002, p. 1.

[5] H. Li, G. Gao, R. Chen, X. Ge, S. Guo, and L.-Y. Hao, "The influence ranking for testers in bug tracking systems," *International Journal of Software Engineering and Knowledge Engineering*, 2019, v. 29, n. 01, pp. 93–113.

[6] "Iso/iec/ieee international standard - systems and software engineering – vocabulary," *ISO/IEC/IEEE 24765:2010(E)*, 2010, pp. 1–418.

[7] N. Fention and M. Neill, "A critique of software defect predction models," *IEEE Transactions on software engineering*, 1999, v. 25, n. 5, pp. 675–689.

[8] J. Pachouly, S. Ahirrao, K. Kotecha, G. Selvachandran, and A. Abraham, "A systematic literature review on software defect prediction using artificial intelligence: Datasets, data validation methods, approaches, and tools," *Engineering Applications of Artificial Intelligence*, 2022, v. 111, p. 104773.

[9] G. Tassey, "The economic impacts of inadequate infrastructure for software testing. national institute of standards and technology," *RTI Project*, 2002, v. 7007, n. 11, pp. 1–309.

[10] S. S. Rathore and S. Kumar, "A study on software fault prediction techniques," *Artificial Intelligence Review*, 2019, v. 51, n. 2, pp. 255–327.

[11] I. Sommerville, *Software Engineering, 9/E.* Pearson Education India, 2011.

[12] S. S. Rathore and S. Kumar, "An empirical study of some software fault prediction techniques for the number of faults prediction," *Soft Computing*, 2017, v. 21, n. 24, pp. 7417–7434.

[13] M. K. Thota, F. H. Shajin, P. Rajesh *et al.*, "Survey on software defect prediction techniques," *International Journal of Applied Science and Engineering*, 2020, v. 17, n. 4, pp. 331–344.

[14] L. Kumar, S. Rath, and A. Sureka, "An Empirical Analysis on Effective Fault Prediction Model Developed Using Ensemble Methods," *Proceedings - International Computer Software and Applications Conference*, 2017, v. 1, pp. 244–249.

[15] S. Herbold, "On the costs and profit of software defect prediction," *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 2019, v. X, n. X, pp. 1–16.

[16] C. Ni, X. Xia, D. Lo, X. Chen, and Q. Gu, "Revisiting Supervised and Unsupervised Methods for Effort-Aware Cross-Project Defect Prediction," *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 2019.

[17] J. Ren and F. Liu, "A novel approach for software defect prediction based on the power law function," *Applied Sciences (Switzerland)*, 2020, v. 10, n. 5.

[18] D. A. Tamburri, F. Palomba, R. Kazman, and S. Member, "Success and Failure in Software Engineering : A Followup Systematic Literature Review," *IEEE Transactions on Engineering Management*, 2020, pp. 1–13. [Online]. Available: available:https://figshare.com/s/e6f0968e55c2cd02438

[19] T. Lee, J. Nam, D. Han, S. Kim, and H. Peter In, "Micro Interaction Metrics for Software Defect Prediction," in *ESEC/FSE'11*, v. 42, n. 11. Seged, Hungary: ACM, 2011, pp. 311–321.

[20] N. Nagappan, T. Ball, and A. Zeller, "Mining metrics to predict component failures," *Proceedings - International Conference on Software Engineering*, 2006, v. 2006, n. ICSE'06, pp. 452–461.

[21] T. J. Ostrand, E. J. Weyuker, and R. M. Bell, "Predicting the location and number of faults in large software systems," *IEEE Transactions on Software Engineering*, 2005, v. 31, n. 4, pp. 340–355.

[22] P. Tomaszewski, H. Grahn, and L. Lundberg, "A method for an accurate early prediction of faults in modified classes," *IEEE International Conference on Software Maintenance, ICSM*, 2006, n. 06, pp. 487–495.

[23] J. Nam, "Survey on software defect prediction," *Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Tech. Rep*, 2014.

[24] T. Menzies, A. Dekhtyar, J. Distefano, and J. Greenwald, "Problems with precision: A response to" comments on'data mining static code attributes to learn defect predictors'"," *IEEE Transactions on Software Engineering*, 2007, v. 33, n. 9, pp. 637–640.

[25] J. Nam, S. J. Pan, and S. Kim, "Transfer Defect Learning," in *Proceedings - International Conference on Software Engineering*. San Franscisco, CA, USA: IEEE, 2013, pp. 382–391.

[26] S. Shivaji, E. James Whitehead, R. Akella, and S. Kim, "Reducing features to improve code change-based bug prediction," *IEEE Transactions on Software Engineering*, 2013, v. 39, n. 4, pp. 552–569.

[27] B. Turhan, T. Menzies, A. B. Bener, and J. Di, "On the relative value of cross-company and within-company data for defect prediction," *Empirical Software Engineering*, 2009, v. 540, n. 14, pp. 1–34.

[28] M. D'Ambros, M. Lanza, and R. Robbes, "Evaluating defect prediction approaches: A benchmark and an extensive comparison," *Empirical Software Engineering*, 2012, v. 17, n. 4-5, pp. 531–577.

[29] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A systematic literature review on fault prediction performance in software engineering," *IEEE Transactions on Software Engineering*, 2012, v. 38, n. 6, pp. 1276–1304.

[30] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking classification models for software defect prediction: A proposed framework and novel findings," *IEEE Transactions on Software Engineering*, 2008, v. 34, n. 4, pp. 485–496.

[31] Y. Ma, G. Luo, X. Zeng, and A. Chen, "Transfer learning for cross-company software defect prediction," *Information and Software Technology*, 2012, v. 54, n. 3, pp. 248–256. [Online]. Available: http://dx.doi.org/10.1016/j.infsof.2011.09.007

[32] B. Ma, K. Dejaeger, J. Vanthienen, and B. Baesens, "Software defect prediction based on association rule classification," *Available at SSRN 1785381*, 2011.

[33] S. K. Pandey, R. B. Mishra, and A. K. Tripathi, "Machine learning based methods for software fault prediction: A survey," *Expert Systems with Applications*, 2021, v. 172, p. 114595.

[34] N. Li, M. Shepperd, and Y. Guo, "A systematic review of unsupervised learning techniques for software defect prediction," *Information and Software Technology*, 2020, v. 122, p. 106287.

[35] Ö. F. Arar and K. Ayan, "A feature dependent naive bayes approach and its application to the software defect prediction problem," *Applied Soft Computing*, 2017, v. 59, pp. 197–209.

[36] K. Wang, L. Liu, C. Yuan, and Z. Wang, "Software defect prediction model based on lasso–svm," *Neural Computing and Applications*, 2021, v. 33, n. 14, pp. 8249–8259.

[37] T. Zhou, X. Sun, X. Xia, B. Li, and X. Chen, "Improving defect prediction with deep forest," *Information and Software Technology*, 2019, v. 114, pp. 204–216.

[38] A. Iqbal and S. Aftab, "A classification framework for software defect prediction using multi-filter feature selection technique and mlp." *International Journal of Modern Education & Computer Science*, 2020, v. 12, n. 1.

[39] X. Chen, Y. Zhao, Q. Wang, and Z. Yuan, "Multi: Multi-objective effort-aware just-in-time software defect prediction," *Information and Software Technology*, 2018, v. 93, pp. 1–13.

[40] M. J. Siers and M. Z. Islam, "Software defect prediction using a cost sensitive decision forest and voting, and a potential solution to the class imbalance problem," *Information Systems*, 2015, v. 51, pp. 62–71.

[41] S. S. Rathore and S. Kumar, "Predicting number of faults in software system using genetic programming," *Procedia Computer Science*, 2015, v. 62, pp. 303–311.

[42] S. Chatterjee and B. Maji, "A new fuzzy rule based algorithm for estimating software faults in early phase of development," *Soft Computing*, 2016, v. 20, n. 10, pp. 4023–4035.

[43] G. Czibula, Z. Marian, and I. G. Czibula, "Software defect prediction using relational association rule mining," *Information Sciences*, 2014, v. 264, pp. 260–278.

[44] X.-Y. Jing, S. Ying, Z.-W. Zhang, S.-S. Wu, and J. Liu, "Dictionary learning based software defect prediction," in *Proceedings of the 36th international conference on software engineering*, 2014, pp. 414–423.

[45] H. Tong, B. Liu, and S. Wang, "Software defect prediction using stacked denoising autoencoders and two-stage ensemble learning," *Information and Software Technology*, 2018, v. 96, pp. 94–111.

[46] Z.-W. Zhang, X.-Y. Jing, and T.-J. Wang, "Label propagation based semi-supervised learning for software defect prediction," *Automated Software Engineering*, 2017, v. 24, n. 1, pp. 47–69.

[47] R. Kumar, A. Chaturvedi, and L. Kailasam, "An unsupervised software fault prediction approach using threshold derivation," *IEEE Transactions on Reliability*, 2022, v. 71, n. 2, pp. 911–932.

[48] J. Nam and S. Kim, "Clami: Defect prediction on unlabeled datasets (t)," in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2015, pp. 452–463.

[49] S. K. Pandey, R. B. Mishra, and A. K. Tripathi, "Bpdet: An effective software bug prediction model using deep representation and ensemble learning techniques," *Expert Systems with Applications*, 2020, v. 144, p. 113085.

[50] Q. Song, Y. Guo, and M. Shepperd, "A comprehensive investigation of the role of imbalanced learning for software defect prediction," *IEEE Transactions on Software Engineering*, 2018, v. 45, n. 12, pp. 1253–1269.

[51] C. Tantithamthavorn, A. E. Hassan, and K. Matsumoto, "The impact of class rebalancing techniques on the performance and interpretation of defect prediction models," *IEEE Transactions on Software Engineering*, 2018, v. 46, n. 11, pp. 1200–1219.

[52] S. S. Rathore, S. S. Chouhan, D. K. Jain, and A. G. Vachhani, "Generative oversampling methods for handling imbalanced data in software fault prediction," *IEEE Transactions on Reliability*, 2022, v. 71, n. 2, pp. 747–762.

[53] Y. Kamei, A. Monden, S. Matsumoto, T. Kakimoto, and K.-i. Matsumoto, "The effects of over and under sampling on fault-prone module detection," in *First international symposium on empirical software engineering and measurement (ESEM 2007)*. IEEE, 2007, pp. 196–204.

[54] G. Menardi and N. Torelli, "Training and assessing classification rules with imbalanced data," *Data mining and knowledge discovery*, 2014, v. 28, n. 1, pp. 92–122.

[55] L. Kumar, S. Misra, and S. K. Rath, "An empirical analysis of the effectiveness of software metrics and fault prediction model for identifying faulty classes," *Computer standards & interfaces*, 2017, v. 53, pp. 1–32.

[56] F. Matloob, T. M. Ghazal, N. Taleb, S. Aftab, M. Ahmad, M. A. Khan, S. Abbas, and T. R. Soomro, "Software defect prediction using ensemble learning: A systematic literature review," *IEEE Access*, 2021.

[57] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, "Cross-project Defect Prediction," in *ESEC/FSE'09*. Amsterdam, The Netherlands: ACM, 2009, pp. 91–100.

[58] G. Canfora, A. De Lucia, M. Di Penta, R. Oliveto, A. Panichella, and S. Panichella, "Multi-objective cross-project defect prediction," in *Proceedings - IEEE 6th International Conference on Software Testing, Verification and Validation, ICST 2013*, 2013, pp. 252–261.

[59] T. Fukushima, Y. Kamei, S. McIntosh, K. Yamashita, and N. Ubayashi, "An empirical study of just-in-time defect prediction using cross-project models," in *11th Working Conference on Mining Software Repositories, MSR 2014 - Proceedings*, 2014, pp. 172–181.

[60] S. Watanabe, H. Kaiya, and K. Kaijiri, "Adapting a fault prediction model to allow inter language reuse," in *PROMISE'08*. Leipzig, Germany: ACM, 2008, pp. 19–24.

[61] F. Zhang, A. Mockus, I. Keivanloo, and Y. Zou, "Towards building a universal defect prediction model," *Empirical Software Engineering*, 2016, v. 21, n. 5, pp. 2107–2145.

[62] J. Nam and S. Kim, "Heterogeneous Defect Prediction," *IEEE Transactions on Software Engineering*, 2015, v. 44, n. 9, pp. 874–896.

[63] B. Turhan, T. Menzies, A. B. Bener, and J. Di Stefano, "On the relative value of cross-company and within-company data for defect prediction," *Empirical Software Engineering*, 2009, v. 14, n. 5, pp. 540–578.

[64] B. Turhan, "On the dataset shift problem in software engineering prediction models," *Empirical Software Engineering*, 2012, v. 17, n. 1-2, pp. 62–74.

[65] F. Zhang, Q. Zheng, Y. Zou, and A. E. Hassan, "Cross-project defect prediction using a connectivity-based unsupervised classifier," in *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*. IEEE, 2016, pp. 309–320.

[66] G. Abaei, A. Selamat, and H. Fujita, "An empirical study based on semi-supervised hybrid self-organizing map for software fault prediction," *Knowledge-Based Systems*, 2015, v. 74, pp. 28–39.

[67] S. Singh and R. Singla, "Classification of defective modules using object-oriented metrics," *International Journal of Intelligent Systems Technologies and Applications*, 2017, v. 16, n. 1, pp. 1–13.

[68] R. Sasidharan and P. Sriram, "Hyper-quadtree-based k-means algorithm for software fault prediction," in *Computational Intelligence, Cyber Security and Computational Models*. Springer, 2014, pp. 107–118.

[69] Q. Huang, X. Xia, and D. Lo, "Supervised vs unsupervised models: A holistic look at effort-aware just-in-time defect prediction," in *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2017, pp. 159–170.

[70] R. Kumar and A. Chaturvedi, "Software bug prediction using reward-based weighted majority voting ensemble technique," *IEEE Transactions on Reliability*, 2024, v. 73, n. 1, pp. 726–740.

[71] X. Yu, J. Keung, Y. Xiao, S. Feng, F. Li, and H. Dai, "Predicting the precise number of software defects: Are we there yet?" *Information and Software Technology*, 2022, v. 146, p. 106847.

[72] R. Kumar and A. Chaturvedi, "Software bug count prediction model based on software source code," in *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 2023, pp. 1–6.

[73] Q. Huang, C. Ni, X. Chen, Q. Gu, and K. Cao, "Multi-project regression based approach for software defect number prediction." in *SEKE*, 2019, pp. 425–546.

[74] W. Zhang, Y. Du, T. Yoshida, Q. Wang, and X. Li, "Samen-svr: using sample entropy and support vector regression for bug number prediction," *IET Software*, 2018, v. 12, n. 3, pp. 183–189.

[75] S. S. Rathore and S. Kumar, "A decision tree regression based approach for the number of software faults prediction," *ACM SIGSOFT Software Engineering Notes*, 2016, v. 41, n. 1, pp. 1–6.

[76] M. Chen and Y. Ma, "An empirical study on predicting defect numbers." in *SEKE*, v. 15, 2015, pp. 397–402.

[77] S. S. Rathore and S. Kumar, "Towards an ensemble based system for predicting the number of software faults," *Expert Systems with Applications*, 2017, v. 82, pp. 357–382.

[78] M. Wu, S. Ye, C. Li, Z. Ma, and Z. Fu, "Revisting the impact of regression models for predicting the number of defects." in *SEKE*, 2018, pp. 427–426.

[79] V. Prokopev, "Lambda calculus and functional programming: Using haskell," *theseus.fi*, 2017.

[80] J. Kunasaikaran and A. Iqbal, "A brief overview of functional programming languages," *electronic Journal of Computer Science and Information Technology*, 2016, v. 6, n. 1.

[81] A. Khanfor and Y. Yang, "An overview of practical impacts of functional programming," in *2017 24th Asia-Pacific Software Engineering Conference Workshops (APSECW)*. IEEE, 2017, pp. 50–54.

[82] Z. Hu, J. Hughes, and M. Wang, "How functional programming mattered," *National Science Review*, 2015, v. 2, n. 3, pp. 349–370.

[83] P. Jernlund and M. Stenberg, "Functional and imperative object-oriented programming in theory and practice: A study of online discussions in the programming community," 2019.

[84] M. Sherriff, L. Williams, and M. Vouk, "Using in-process metrics to predict defect density in haskell programs," in *Fast Abstract, International Symposium on Software Reliability Engineering, St. Malo, France*, 2004.

[85] M. Sherriff, N. Nagappan, L. Williams, and M. Vouk, "Early estimation of defect density using an in-process Haskell metrics model," *Proceedings of the 1st International Workshop on Advances in Model-Based Testing, A-MOST '05*, 2005, pp. 2–7.

[86] Z. Xu, L. Li, M. Yan, J. Liu, X. Luo, J. Grundy, Y. Zhang, and X. Zhang, "A comprehensive comparative study of clustering-based unsupervised defect prediction models," *Journal of Systems and Software*, 2021, v. 172, p. 110862.

[87] S. Zhong, T. M. Khoshgoftaar, and N. Seliya, "Unsupervised Learning for Expert-Based Software Quality Estimation," in *Eighth IEEE International Symposium on High Assurance Systems Engineering*. IEEE, 2004.

[88] S. K. Pandey and A. K. Tripathi, "Bcv-predictor: A bug count vector predictor of a successive version of the software system," *Knowledge-Based Systems*, 2020, v. 197, p. 105924.

[89] S. S. Rathore and S. Kumar, "Ensemble methods for the prediction of number of faults: a study on eclipse project," in *2016 11th International Conference on Industrial and Information Systems (ICIIS)*. IEEE, 2016, pp. 540–545.

[90] R. Kumar and A. Chaturvedi, "A framework for software defect prediction using optimal hyper-parameters of deep neural network," in *29th International Conference on Neural Information Processing (ICONIP)*. Springer, 2022, pp. 163–174.

[91] K. Gao, T. M. Khoshgoftaar, H. Wang, and N. Seliya, "Choosing software metrics for defect prediction: an investigation on feature selection techniques," *Software: Practice and Experience*, 2011, v. 41, n. 5, pp. 579–606.

[92] L. Kumar, S. K. Sripada, A. Sureka, and S. K. Rath, "Effective fault prediction model developed using least square support vector machine (lssvm)," *Journal of Systems and Software*, 2018, v. 137, pp. 686–712.

[93] A. Tosun, A. B. Bener, and S. Akbarinasaji, "A systematic literature review on the applications of bayesian networks to predict software quality," *Software Quality Journal*, 2017, v. 25, n. 1, pp. 273–305.

[94] S. Chatterjee and B. Maji, "A bayesian belief network based model for predicting software faults in early phase of software development process," *Applied Intelligence*, 2018, v. 48, n. 8, pp. 2214–2228.

[95] L. Madeyski and M. Jureczko, "Which process metrics can significantly improve defect prediction models? an empirical study," *Software Quality Journal*, 2015, v. 23, n. 3, pp. 393–422.

[96] T. Menzies, Z. Milton, B. Turhan, B. Cukic, Y. Jiang, and A. Bener, "Defect prediction from static code features: current results, limitations, new approaches," *Automated Software Engineering*, 2010, v. 17, n. 4, pp. 375–407.

[97] H. Zhang, A. Nelson, and T. Menzies, "On the value of learning from defect dense components for software defect prediction," in *Proceedings of the 6th International Conference on Predictive Models in Software Engineering*, 2010, pp. 1–9.

[98] T. Menzies, B. Turhan, A. Bener, G. Gay, B. Cukic, and Y. Jiang, "Implications of ceiling effects in defect predictors," in *Proceedings of the 4th international workshop on Predictor models in software engineering*, 2008, pp. 47–54.

[99] K. Stapor, "Evaluating and comparing classifiers: Review, some recommendations and limitations," in *International Conference on Computer Recognition Systems*. Springer, 2017, pp. 12–21.

[100] C. De Stefano, F. Fontanella, G. Folino, and A. S. Di Freca, "A bayesian approach for combining ensembles of gp classifiers," in *International Workshop on Multiple Classifier Systems*. Springer, 2011, pp. 26–35.

[101] S. Moustafa, M. Y. ElNainay, N. El Makky, and M. S. Abougabal, "Software bug prediction using weighted majority voting techniques," *Alexandria engineering journal*, 2018, v. 57, n. 4, pp. 2763–2774.

[102] C. Manjula and L. Florence, "Deep neural network based hybrid approach for software defect prediction using software metrics," *Cluster Computing*, 2019, v. 22, n. 4, pp. 9847–9863.

[103] H. I. Aljamaan and M. O. Elish, "An empirical study of bagging and boosting ensembles for identifying faulty classes in object-oriented software," in *2009 IEEE Symposium on Computational Intelligence and Data Mining*. IEEE, 2009, pp. 187–194.

[104] T. Wang, W. Li, H. Shi, and Z. Liu, "Software defect prediction based on classifiers ensemble," *Journal of Information & Computational Science*, 2011, v. 8, n. 16, pp. 4241–4254.

[105] I. H. Laradji, M. Alshayeb, and L. Ghouti, "Software defect prediction using ensemble learning on selected features," *Information and Software Technology*, 2015, v. 58, pp. 388–402.

[106] S. S. Rathore and S. Kumar, "An empirical study of ensemble techniques for software fault prediction," *Applied Intelligence*, 2021, v. 51, n. 6, pp. 3615–3644.

[107] V. R. Basili, L. C. Briand, W. L. Melo, and I. C. Society, "A Validation of Object-Oriented Design Metrics as Quality Indicators," *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 1996, v. 22, n. 10, pp. 751–761.

[108] P. Singh, N. R. Pal, S. Verma, and O. P. Vyas, "Fuzzy rule-based approach for software fault prediction," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2016, v. 47, n. 5, pp. 826–837.

[109] N. Ohlsson and H. Alberg, "Predicting Fault-Prone Software Modules in Telephone Switches," *IEEE Transactions on Software Engineering*, 1996, v. 22, n. 12, pp. 886–894.

[110] Y. Li, W. E. Wong, S.-Y. Lee, and F. Wotawa, "Using tri-relation networks for effective software fault-proneness prediction," *IEEE Access*, 2019, v. 7, pp. 63 066–63 080.

[111] W. E. Wong, J. R. Horgan, M. Syring, W. Zage, and D. Zage, "Applying design metrics to predict fault-proneness: a case study on a large-scale software system," *Software: Practice and Experience*, 2000, v. 30, n. 14, pp. 1587–1608.

[112] F. Rahman and P. Devanbu, "How, and Why, Process Metrics Are Better," in *ICSE 2003, San Franscisco, CA, USA, IEEE*. IEEE, 2013, pp. 432–441.

[113] T. Zimmermann and N. Nagappan, "Predicting defects using network analysis on dependency graphs," in *Proceedings - International Conference on Software Engineering*, 2008, pp. 531–540.

[114] A. Panichella, R. Oliveto, and A. De Lucia, "Cross-project defect prediction models: L'Union fait la force," in *2014 Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering, CSMR-WCRE 2014 - Proceedings*. Antwerp, Belgium: IEEE, 2014, pp. 164–173.

[115] C. Catal, U. Sevim, and B. Diri, "Clustering and metrics thresholds based software fault prediction of unlabeled program modules," *ITNG 2009 - 6th International Conference on Information Technology: New Generations*, 2009, pp. 199–204.

[116] J. Nam and S. Kim, "CLAMI: Defect prediction on unlabeled datasets," *Proceedings - 2015 30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015*, 2015, pp. 452–463.

[117] M. Yan, X. Zhang, C. Liu, L. Xu, M. Yang, and D. Yang, "Automated change-prone class prediction on unlabeled dataset using unsupervised method," *Information and Software Technology*, 2017, v. 92, pp. 1–16.

[118] S. K. Pandey and A. K. Tripathi, "Dnnattention: A deep neural network and attention based architecture for cross project defect number prediction," *Knowledge-Based Systems*, 2021, v. 233, p. 107541.

[119] S. Yang, X. Gou, M. Yang, Q. Shao, C. Bian, M. Jiang, and Y. Qiao, "Software bug number prediction based on complex network theory and panel data model," *IEEE Transactions on Reliability*, 2022, v. 71, n. 1, pp. 162–177.

[120] J. Wang and H. Zhang, "Predicting defect numbers based on defect state transition models," in *Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*. IEEE, 2012, pp. 191–200.

[121] W. E. Wong, V. Debroy, A. Surampudi, H. Kim, and M. F. Siok, "Recent catastrophic accidents: Investigating how software was responsible," in *2010 Fourth International Conference on Secure Software Integration and Reliability Improvement*. IEEE, 2010, pp. 14–22.

[122] J. Nam and S. Kim, "Heterogeneous defect prediction," in *Proceedings of the 2015 10th joint meeting on foundations of software engineering*, 2015, pp. 508–519.

[123] R. S. Wahono, "A systematic literature review of software defect prediction," *Journal of software engineering*, 2015, v. 1, n. 1, pp. 1–16.

[124] X. Chen, D. Zhang, Y. Zhao, Z. Cui, and C. Ni, "Software defect number prediction: Unsupervised vs supervised methods," *Information and Software Technology*, 2019, v. 106, pp. 161–181.

[125] D. Abajirov, "Does functional programming improve software quality?: an empirical analysis of open source projects on github," *https://elib.uni-stuttgart.de/*, 2021.

[126] I. Gondra, "Applying machine learning to software fault-proneness prediction," *Journal of Systems and Software*, 2008, v. 81, n. 2, pp. 186–195.

[127] L. Kumar, S. K. Sripada, A. Sureka, and S. K. Rath, "Effective fault prediction model developed using Least Square Support Vector Machine (LSSVM)," *Journal of Systems and Software*, 2018, v. 137, pp. 686–712. [Online]. Available: https://doi.org/10.1016/j.jss.2017.04.016

[128] R. Kumar and D. Gupta, "Software bug prediction system using neural network," *European Journal of Advances in Engineering and Technology*, 2016, v. 3, n. 7, pp. 78–84.

[129] K. Tameswar, G. Suddul, and K. Dookhitram, "Enhancing deep learning capabilities with genetic algorithm for detecting software defects," in *Progress in Advanced Computing and Intelligent Engineering*. Springer, 2021, pp. 211–220.

[130] C. Manjula and L. Florence, "Deep neural network based hybrid approach for software defect prediction using software metrics," *Cluster Computing*, 2019, v. 22, n. 4, pp. 9847–9863.

[131] M. Hossain, "Challenges of software quality assurance and testing," *International Journal of Software Engineering and Computer Systems*, 2018, v. 4, n. 1, pp. 133–144.

[132] C. Ryder and S. Thompson, "Software Metrics : Measuring Haskell," in *Trends in Functional Progamming*, 2005, ch. 1, pp. 31–46.

[133] N. Lunardon, G. Menardi, and N. Torelli, "Rose: A package for binary imbalanced learning." *R journal*, 2014, v. 6, n. 1.

[134] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, 2002, v. 16, pp. 321–357.

[135] S. Omri and C. Sinz, "Deep learning for software defect prediction: A survey," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, 2020, pp. 209–214.

[136] H. Ji, S. Huang, Y. Wu, Z. Hui, and C. Zheng, "A new weighted naive bayes method based on information diffusion for software defect prediction," *Software Quality Journal*, 2019, v. 27, n. 3, pp. 923–968.

[137] Z. He, F. Shu, Y. Yang, M. Li, and Q. Wang, "An investigation on the feasibility of cross-project defect prediction," *Automated Software Engineering*, 2012, v. 19, pp. 167–199.

[138] K. Herzig, S. Just, A. Rau, and A. Zeller, "Predicting defects using change genealogies," in *2013 IEEE 24th International Symposium on Software Reliability Engineering, ISSRE 2013*. IEEE, 2013, pp. 118–127.

[139] Y. Suresh, L. Kumar, and S. K. Rath, "Statistical and machine learning methods for software fault prediction using ck metric suite: a comparative analysis," *International Scholarly Research Notices*, 2014, v. 2014.

[140] J. Yang and H. Qian, "Defect prediction on unlabeled datasets by using unsupervised clustering," in *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, 2016, pp. 465–472.

[141] A. Boucher and M. Badri, "Software metrics thresholds calculation techniques to predict fault-proneness: An empirical comparison," *Information and Software Technology*, 2018, v. 96, pp. 38–67.

[142] H. Turabieh, M. Mafarja, and X. Li, "Iterated feature selection algorithms with layered recurrent neural network for software fault prediction," *Expert systems with applications*, 2019, v. 122, pp. 27–42.

[143] A. O. Balogun, F. B. Lafenwa-Balogun, H. A. Mojeed, V. E. Adeyemo, O. N. Akande, A. G. Akintola, A. O. Bajeh, and F. E. Usman-Hamza, "Smote-based homogeneous ensemble methods for software defect prediction," in *International Conference on Computational Science and Its Applications*. Springer, 2020, pp. 615–631.

[144] H. Aljamaan and A. Alazba, "Software defect prediction using tree-based ensembles," in *Proceedings of the 16th ACM International Conference on Predictive Models and Data Analytics in Software Engineering*, 2020, pp. 1–10.

[145] V. K. Kulamala, L. Kumar, and D. P. Mohapatra, "Software fault prediction using lssvm with different kernel functions," *Arabian Journal for Science and Engineering*, 2021, v. 46, n. 9, pp. 8655–8664.

[146] H. Wei, C. Hu, S. Chen, Y. Xue, and Q. Zhang, "Establishing a software defect prediction model via effective dimension reduction," *Information Sciences*, 2019, v. 477, pp. 399–409.

[147] R. Li, L. Zhou, S. Zhang, H. Liu, X. Huang, and Z. Sun, "Software defect prediction based on ensemble learning," in *Proceedings of the 2019 2nd International conference on data science and information technology*, 2019, pp. 1–6.

[148] S. Mehta and K. S. Patnaik, "Improved prediction of software defects using ensemble machine learning techniques," *Neural Computing and Applications*, 2021, pp. 1–12.

[149] F. Yucalar, A. Ozcift, E. Borandag, and D. Kilinc, "Multiple-classifiers in software quality engineering: Combining predictors to improve software fault prediction ability," *Engineering Science and Technology, an International Journal*, 2020, v. 23, n. 4, pp. 938–950.

[150] X. Wan, Z. Zheng, and Y. Liu, "Spe2: Self-paced ensemble of ensembles for software defect prediction," *IEEE Transactions on Reliability*, 2022, v. 71, n. 2, pp. 865–879.

[151] A. Dogan and D. Birant, "A weighted majority voting ensemble approach for classification," in *2019 4th International Conference on Computer Science and Engineering (UBMK)*. IEEE, 2019, pp. 1–6.

[152] L. Yu, "Using negative binomial regression analysis to predict software faults: A study of apache ant," *IJ Information Technology and Computer Science*, 2012, v. 8, pp. 63–70.

[153] K. Gao and T. M. Khoshgoftaar, "A comprehensive empirical study of count models for software fault prediction," *IEEE Transactions on Reliability*, 2007, v. 56, n. 2, pp. 223–236.

[154] W. Afzal, R. Torkar, and R. Feldt, "Prediction of fault count data using genetic programming," in *2008 IEEE International Multitopic Conference*. IEEE, 2008, pp. 349–356.

[155] S. S. Rathore and S. Kuamar, "Comparative analysis of neural network and genetic programming for number of software faults prediction," in *2015 National Conference on Recent Advances in Electronics & Computer Engineering (RAECE)*. IEEE, 2015, pp. 328–332.

[156] T. J. Ostrand, E. J. Weyuker, and R. M. Bell, "Predicting the location and number of faults in large software systems," *IEEE Transactions on Software Engineering*, 2005, v. 31, n. 4, pp. 340–355.

[157] M. Nevendra and P. Singh, "Software bug count prediction via adaboost. r-et," in *2019 IEEE 9th International Conference on Advanced Computing (IACC)*. IEEE, 2019, pp. 7–12.

[158] S. Zhong, T. M. Khoshgoftaar, and N. Seliya, "Analyzing Software Measurement Data with Clustering Techniques," *IEEE INTELLIGENT SYSTEMS*, 2004, pp. 20–27.

[159] N. Seliya and T. M. Khoshgoftaar, "Software quality analysis of unlabeled program modules with semisupervised clustering," *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, 2007, v. 37, n. 2, pp. 201–211.

[160] G. Abaei, A. Selamat, and H. Fujita, "An empirical study based on semi-supervised hybrid self-organizing map for software fault prediction," *Knowledge-Based Systems*, 2015, v. 74, pp. 28–39.

[161] M. Yan, X. Zhang, C. Liu, L. Xu, M. Yang, and D. Yang, "Automated change-prone class prediction on unlabeled dataset using unsupervised method," *Information and Software Technology*, 2017, v. 92, pp. 1–16.

[162] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," *IEEE Transactions on software engineering*, 1994, v. 20, n. 6, pp. 476–493.

[163] T. J. McCabe, "A complexity measure," *IEEE Transactions on software Engineering*, 1976, n. 4, pp. 308–320.

[164] M. H. Halstead, *Elements of Software Science (Operating and programming systems series)*. Elsevier Science Inc., 1977.

[165] S. Kim, E. J. Whitehead, and Y. Zhang, "Classifying software changes: Clean or buggy?" *IEEE Transactions on Software Engineering*, 2008, v. 34, n. 2, pp. 181–196.

[166] T. Lee, J. Nam, D. Han, S. Kim, and H. P. In, "Developer micro interaction metrics for software defect prediction," *IEEE Transactions on Software Engineering*, 2016, v. 42, n. 11, pp. 1015–1035.

[167] T. Hall, M. Zhang, D. Bowes, and Y. Sun, "Some code smells have a significant but small effect on faults," *ACM Trans. Softw. Eng. Methodol.*, sep 2014, v. 23, n. 4. [Online]. Available: https://doi.org/10.1145/2629648

[168] M. S. Biçer and B. Diri, "Predicting defect prone modules in web applications," in *International Conference on Information and Software Technologies*. Springer, 2015, pp. 577–591.

[169] W. Ma, L. Chen, Y. Yang, Y. Zhou, and B. Xu, "Empirical analysis of network measures for effort-aware fault-proneness prediction," *Information and Software Technology*, 2016, v. 69, pp. 50–70.

[170] Y. Zhao, Y. Yang, H. Lu, J. Liu, H. Leung, Y. Wu, Y. Zhou, and B. Xu, "Understanding the value of considering client usage context in package cohesion for fault-proneness prediction," *Automated Software Engineering*, 2017, v. 24, n. 2, pp. 393–453.

[171] C. Ryder, "SOFTWARE MEASUREMENT FOR FUNCTIONAL PROGRAMMING," Ph.D. dissertation, The University of Kent Canterbury, 2004.

[172] V. Pankratius, F. Schmidt, and G. Garretón, "Combining functional and imperative programming for multicore software: An empirical study evaluating scala and java," in *2012 34th International Conference on Software Engineering (ICSE)*. IEEE, 2012, pp. 123–133.

[173] D. Le, M. A. Alipour, R. Gopinath, and A. Groce, "Mutation testing of functional programming languages," *Oregon State University, Tech. Rep*, 2014.

[174] D. Manjhi and A. Chaturvedi, "Reuse estimate and interval prediction using moga-nn and rbf-nn in the functional paradigm," *Science of Computer Programming*, 2021, v. 208, p. 102643.

[175] M. Jureczko and D. Spinellis, "Using object-oriented design metrics to predict software defects," *Models and methods of system dependability. Oficyna Wydawnicza Politechniki Wrocławskiej*, 2010, pp. 69–81.

[176] M. Jureczko and L. Madeyski, "Towards identifying software project clusters with regard to defect prediction," in *Proceedings of the 6th international conference on predictive models in software engineering*, 2010, pp. 1–10.

[177] M. D'Ambros, M. Lanza, and R. Robbes, "Evaluating defect prediction approaches: a benchmark and an extensive comparison," *Empirical Software Engineering*, 2012, v. 17, n. 4, pp. 531–577.

[178] M. Shepperd, Q. Song, Z. Sun, and C. Mair, "Data quality: Some comments on the NASA software defect datasets," *IEEE Transactions on Software Engineering*, 2013, v. 39, n. 9, pp. 1208–1215.

[179] R. Wu, H. Zhang, S. Kim, and S. C. Cheung, "ReLink: Recovering links between bugs and changes," *SIGSOFT/FSE 2011 - Proceedings of the 19th ACM SIGSOFT Symposium on Foundations of Software Engineering*, 2011, pp. 15–25.

[180] F. Peters and T. Menzies, "Privacy and utility for defect prediction: Experiments with MORPH," *Proceedings - International Conference on Software Engineering*, 2012, pp. 189–199.

[181] Z. Li, X.-Y. Jing, X. Zhu, H. Zhang, B. Xu, and S. Ying, "Heterogeneous defect prediction with two-stage ensemble learning," *Automated Software Engineering*, 2019, v. 26, n. 3, pp. 599–651.

[182] F. Peters and T. Menzies, "Privacy and utility for defect prediction: Experiments with morph," in *2012 34th International conference on software engineering (ICSE)*. IEEE, 2012, pp. 189–199.

[183] R. Kumar and A. Chaturvedi, "Software fault prediction using data mining techniques on software metrics," in *Proceedings of International Conference on Machine Learning and Big Data Analytics (ICMLBDA) 2021*. Springer, 2022, pp. 304–313.

[184] J. Sayyad Shirabad and T. Menzies, "The PROMISE Repository of Software Engineering Databases." School of Information Technology and Engineering, University of Ottawa, Canada, 2005. [Online]. Available: http://promise.site.uottawa.ca/SERepository

[185] T. Menzies and B. Caglayan, "The promise repository of empirical software engineering data," 2012.

[186] N. Chinchor and B. M. Sundheim, "Muc-5 evaluation metrics," in *Fifth Message Understanding Conference (MUC-5): Proceedings of a Conference Held in Baltimore, Maryland, August 25-27, 1993*, 1993.

[187] M. Bekkar, H. K. Djemaa, and T. A. Alitouche, "Evaluation Measures for Models Assessment over Imbalanced Data Sets," *Journal of Information Engineering and Applications*, 2013, v. 3, n. 10, pp. 27–38. [Online]. Available: http://www.iiste.org/Journals/index.php/JIEA/article/view/7633

[188] S. Boughorbel, F. Jarray, and M. El-Anbari, "Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric," *PLoS ONE*, 2017, v. 12, n. 6, pp. 1–17.

[189] S. D. Conte, H. E. Dunsmore, and Y. Shen, *Software engineering metrics and models*. Benjamin-Cummings Publishing Co., Inc., 1986.

[190] S. G. MacDonell, "Establishing relationships between specification size and software process effort in case environments," *Information and Software Technology*, 1997, v. 39, n. 1, pp. 35–45.

[191] F. Wilcoxon, "Individual Comparisons by Ranking Methods," *International Biometric Society*, 1945, v. 1, n. 3, pp. 80–83.

[192] P. Nemenyi, "Distribution-free multiple comparisons," Nemenyi1963, Princeton University, 1963. [Online]. Available: https://www.worldcat.org/title/distribution-free-multiple-comparisons/oclc/39810544

[193] G. Madjarov, D. Kocev, D. Gjorgjevikj, and S. Džeroski, "An extensive experimental comparison of methods for multi-label learning," *Pattern Recognition*, 2012, v. 45, n. 9, pp. 3084–3104.

[194] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, 2006, v. 7, pp. 1–30.

[195] E. S. Pearson, K. Pearson, and H. Hartley, *Biometrika Tables for Statisticians: 3d Ed.* Cambridge University Press, 1966.

[196] P. Cohen, S. G. West, and L. S. Aiken, *Applied multiple regression/correlation analysis for the behavioral sciences.* Psychology press, 2014.

[197] J. Cohen, P. Cohen, S. G. West, and L. S. Aiken, *Applied multiple regression/correlation analysis for the behavioral sciences.* Routledge, 2013.

[198] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, 1992, v. 46, n. 3, pp. 175–185.

[199] I. Rish *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, v. 3, n. 22, 2001, pp. 41–46.

[200] W. S. Noble, "What is a support vector machine?" *Nature biotechnology*, 2006, v. 24, n. 12, pp. 1565–1567.

[201] L. Breiman, "Random forests," *Machine learning*, 2001, v. 45, n. 1, pp. 5–32.

[202] J. R. Quinlan, "C4.5: Program for machine learning," *C4.5*, 1993. [Online]. Available: https://www.rulequest.com/see5-unix.html

[203] M. Kuhn, "Building predictive models in r using the caret package," *Journal of statistical software*, 2008, v. 28, pp. 1–26.

[204] P. Bühlmann, "Bagging, Boosting and Ensemble Learning," *Handbook of Computational Statistics*, 2012, n. 1, pp. 1–38. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-21551-3{_}33

[205] M. Kuhn, "Building predictive models in r using the caret package," *Journal of Statistical Software*, 2008, v. 28, n. 5, p. 1–26. [Online]. Available: https://www.jstatsoft.org/index.php/jss/article/view/v028i05

[206] H. Barkmann, R. Lincke, W. Lowe, and S. T. Group, "Quantitative Evaluation of Software Quality Metrics in Open-Source Projects," *Advanced Information Networking and Applications*, 2009, pp. 1067–1072.

[207] H. M. Olague, L. H. Etzkorn, S. Member, S. Gholston, and S. Quattlebaum, "Empirical Validation of Three Software Metrics Suites to Predict Fault-Proneness of Object-Oriented Classes Developed Using Highly Iterative or Agile Software Development Processes," *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 2007, v. 33, n. 6, pp. 402–419.

[208] T. Gyimothy, R. Ferenc, and I. Siket, "Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction," *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 2005, v. 31, n. 10, pp. 897–910.

[209] J. W. Osborne, "Notes on the use of data transformations," *Practical Assessment, Research & Evaluation*, 2002, v. 8(6), n. 1989, pp. 1–8. [Online]. Available: http://pareonline.net/getvn.asp?v=8{&}n=6

[210] F. Zhang, I. Keivanloo, and Y. Zou, "Data transformation in cross-project defect prediction," *Empirical Software Engineering*, 2017, v. 22, n. 6, pp. 3186–3218.

[211] T. Zimmermann, N. Nagappan, and A. Zeller, "Predicting Bugs from History," in *Software Evolution*. Canada: Springer, 2008, ch. 4, pp. 69–88.

[212] M. Jureczko and L. Madeyski, "Towards identifying software project clusters with regard to defect prediction," *Proceedings of the 6th International Conference on Predictive Models in Software Engineering*, 2010, pp. 1–10.

[213] K. E. Emam, S. Benlarbi, N. Goel, W. Melo, H. Lounis, and S. N. Rai, "The Optimal Class Size for Object-Oriented Software Khaled," *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 2002, v. 28, n. 5, pp. 494–509.

[214] W. Liu, S. Liu, Q. Gu, J. Chen, X. Chen, and D. Chen, "Empirical studies of a two-stage data preprocessing approach for software fault prediction," *IEEE Transactions on Reliability*, 2015, v. 65, n. 1, pp. 38–53.

[215] T. M. Khoshgoftaar and K. Gao, "Feature selection with imbalanced data for software defect prediction," *8th International Conference on Machine Learning and Applications, ICMLA 2009*, 2009, pp. 235–240.

[216] H. Wang, T. M. Khoshgoftaar, and N. Seliya, "How many software metrics should be selected for defect prediction?" in *Proceedings of the 24th International Florida Artificial Intelligence Research Society, FLAIRS - 24*, n. Mi. AAAi, 2011, pp. 69–74.

[217] C. L. Chang, "Finding Prototypes for Nearest Neighbor Classifiers," *IEEE Transactions on Computers*, 1974, v. C-23, n. 11, pp. 1179–1184.

[218] E. Kocaguneli, T. Menzies, A. B. Bener, and J. W. Keung, "Exploiting the essential assumptions of analogy-based effort estimation," *IEEE Transactions on Software Engineering*, 2012, v. 38, n. 2, pp. 425–438.

[219] Y. F. Li, M. Xie, and T. N. Goh, "A study of project selection and feature weighting for analogy based software cost estimation," *Journal of Systems and Software*, 2009, v. 82, n. 2, pp. 241–252. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2008.06.001

[220] K. Gao, T. M. Khoshgoftaar, H. Wang, and N. Seliya, "Choosing software metrics for defect prediction: an investigation on feature selection techniques Kehan," *Software - Practice and Experience*, 2009, v. 39, n. 7, pp. 701–736.

[221] L. Breiman, "Random forests," *Machine learning*, 2001, v. 45, n. 1, pp. 5–32.

[222] K. E. Bennin, J. Keung, P. Phannachitta, A. Monden, and S. Mensah, "Mahakil: Diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction," *IEEE Transactions on Software Engineering*, 2017, v. 44, n. 6, pp. 534–550.

[223] M. Laib and M. Kanevski, "Unsupervised feature selection based on space filling concept," *arXiv preprint arXiv:1706.08894*, 2017.

[224] F. Rahman and P. Devanbu, "How, and why, process metrics are better," in *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, 2013, pp. 432–441.

[225] T. Menzies, A. Dekhtyar, J. Distefano, and J. Greenwald, "Problems with precision: A response to" comments on'data mining static code attributes to learn defect predictors'"," *IEEE Transactions on Software Engineering*, 2007, v. 33, n. 9, pp. 637–640.

[226] T. Zimmermann, N. Nagappan, and A. Zeller, "Predicting bugs from history," in *Software evolution*. Springer, 2008, pp. 69–88.

[227] K. El Emam, S. Benlarbi, N. Goel, W. Melo, H. Lounis, and S. N. Rai, "The optimal class size for object-oriented software," *IEEE Transactions on software engineering*, 2002, v. 28, n. 5, pp. 494–509.

[228] J. Fox, *Applied regression analysis, linear models, and related methods.* sage publications, Inc, 1997.

[229] V. R. Basili and B. T. Perricone, "Software errors and complexity: an empirical investigation0," *Communications of the ACM*, 1984, v. 27, n. 1, pp. 42–52.

[230] T. Rowland, "Church-turing thesis," *Hg. v. Eric W. Weisstein. Mathworld–A Wolfram Web Resource. Online verfügbar unter http://mathworld. wolfram. com/Church-TuringThesis. html, zuletzt geprüft am*, 2015, v. 14.

[231] J. C. Murphy, B. Shivkumar, A. Pritchard, G. Iraci, D. Kumar, S. H. Kim, and L. Ziarek, "A survey of real-time capabilities in functional languages and compilers," *Concurrency and Computation: Practice and Experience*, 2019, v. 31, n. 4, p. e4902.

[232] A. Khanfor and Y. Yang, "An overview of practical impacts of functional programming," in *2017 24th Asia-Pacific Software Engineering Conference Workshops (APSECW)*, 2017, pp. 50–54.

[233] H. Li and S. Thompson, "Tool support for refactoring functional programs," *Proceedings of the 2nd Workshop on Refactoring Tools, WRT '08, in conjunction with the Conference on Object Oriented Programming Systems Languages and Applications, OOPSLA 2008*, 2008, pp. 27–38.

[234] G. Lacerda, F. Petrillo, M. Pimenta, and Y. G. Guéhéneuc, "Code smells and refactoring: A tertiary systematic review of challenges and observations," *Journal of Systems and Software*, 2020, v. 167, p. 110610.

[235] D. Manjhi and A. Chaturvedi, "Software Component Reusability Classification in Functional Paradigm," *Proceedings of 2019 3rd IEEE International Conference on Electrical, Computer and Communication Technologies, ICECCT 2019*, 2019, pp. 1–7.

[236] E. R. Gansner, "Gvpr 1 - graph pattern scanning and processing language," *https://www.graphviz.org/pdf/gvpr.1.pdf*, 2012, pp. 1–14.

[237] J. Nam, W. Fu, S. Kim, T. Menzies, and L. Tan, "Heterogeneous Defect Prediction," *IEEE Transactions on Software Engineering*, 2018, v. 44, n. 9, pp. 874–896.

[238] R. Shatnawi, "Comparison of threshold identification techniques for object-oriented software metrics," *IET Software*, 2020, v. 14, n. 6, pp. 727–738.

[239] C. Shi, B. Wei, S. Wei, W. Wang, H. Liu, and J. Liu, "A quantitative discriminant method of elbow point for the optimal number of clusters in clustering algorithm," *EURASIP Journal on Wireless Communications and Networking*, 2021, v. 2021, n. 1, pp. 1–16.

[240] K. P. Sinaga and M.-S. Yang, "Unsupervised k-means clustering algorithm," *IEEE Access*, 2020, v. 8, pp. 80 716–80 727.

[241] B. Angelin and A. Geetha, "Outlier detection using clustering techniques–k-means and k-median," in *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*. IEEE, 2020, pp. 373–378.

[242] K. Sari, S. Efendi, and S. Nasution, "Combining the active learning algorithm based on the silhouette coefficient with pckmeans algorithm," in *2020 3rd International Conference on Mechanical, Electronics, Computer, and Industrial Technology (MECnIT)*. IEEE, 2020, pp. 232–237.

[243] Z. Tóth, P. Gyimesi, and R. Ferenc, "A public bug database of github projects and its application in bug prediction," in *Computational Science and Its Applications–ICCSA 2016: 16th International Conference, Beijing, China, July 4-7, 2016, Proceedings, Part IV 16*. Springer, 2016, pp. 625–638.

[244] R. Ferenc, Z. Tóth, G. Ladányi, I. Siket, and T. Gyimóthy, "A public unified bug dataset for java and its assessment regarding metrics and bug prediction," *Software Quality Journal*, 2020, v. 28, pp. 1447–1506.

[245] M. Canaparo and E. Ronchieri, "Data mining techniques for software quality prediction in open source software-an initial assessment," in *EPJ Web of Conferences*, v. 214. EDP Sciences, 2019, p. 05007.

[246] M. Lipovaca, *Learn you a haskell for great good!: a beginner's guide*. no starch press, 2011.

[247] P. Hudak and J. H. Fasel, "A gentle introduction to haskell," *ACM Sigplan Notices*, 1992, v. 27, n. 5, pp. 1–52.