Appendix for "Unsupervised Software Defect Prediction through Multi-view Clustering"

1 Related Work of Unsupervised Defect Prediction

Unsupervised defect prediction (UDP) focuses on the task of identifying potential defect-prone modules on unlabeled datasets without relying on labeled data [1] [2] [3]. To facilitate a holistic understanding of this topic, Table 1 shows a detailed summary of the related work, highlighting the main methods and techniques in the field of UDP.

TABLE 1 A summary of related work to unsupervised defect prediction

Type	Study	Method	Technique	Year
Clustering-based unsupervised defect prediction (CUDP)	Xu et al. [2]	empirical study	comparison of 40 clustering models	2021
	Yang et al. [3]	empirical study	comparison of 22 clustering models	2024
	Zhong et al. [4]	expert method	k-means, neural-gas	2004
	Catal et al. [5]	threshold method	k-means, metric thresholds	2009
	Nam and Kim [6]	CLA, CLAMI	threshold clustering, metric and instance selection	2015
	Yang and Qian [7]	ACL	threshold clustering	2016
	Yang et al. [8]	CEL	threshold clustering ensemble	2018
	Kumar et al. [9]	TCL, TCLP	threshold clustering, metric and instance selection	2022
	Zhang et al. [10]	SC	spectral clustering	2016
	Marjuni et al. [11]	MAD	spectral clustering	2019
	Yang et al. [12]	AP	affinity propagation	2008
	Bishnu and Bhattacherjee [13]	QDK	quad tree-based k-means	2012
Ranking-based unsupervised defect prediction (RUDP)	Yang et al. [14]	-	sorting the reciprocal value of each metric in descending	2016
	Liu et al. [15]	CCUM	sorting the reciprocal value of code churn in descending	2017
	Zhou et al. [16]	ManualDown MuanualUp	sorting the value of LOC in descending or ascending	2018

2 Software Projects and Metrics

In this paper, we conduct defect prediction on 28 releases across 8 open-source software projects from the Apache community [17]. Each project consists of 65 software metrics along 3 dimensions, i.e., 54 code metrics, 5 process metrics, and 6 ownership metrics. Among them, code metrics (e.g., LOC and cyclomatic complexity) describe the relationship between code properties and software quality, process metrics (e.g., the number of added lines and deleted lines) describe the relationship between development process and software quality, and ownership metrics (e.g., the number of own commits and own lines) describe the relationship between the ownership of modules and software quality. We refer each metric dimension as one view for multi-view clustering and there are 3 views in total. Tables 2 and 3 separately show the statistical summary of the studied software projects and metrics.

TABLE 2 An overview of the used software projects

Project	#Metrics	#Files	#Defective files	%Defective files
activemq-5.0.0	65	1884	293	15.55%
activemq-5.1.0	65	1970	154	7.82%
activemq-5.2.0	65	2040	219	10.74%
activemq-5.3.0	65	2367	258	10.90%
activemq-5.8.0	65	3420	206	6.02%
derby-10.2.1.6	65	1963	661	33.67%
derby-10.3.1.4	65	2206	669	30.33%
derby-10.5.1.1	65	2705	383	14.16%
groovy-1_5_7	65	757	26	3.43%
groovy-1_6_BETA_1	65	821	70	8.53%
groovy-1_6_BETA_2	65	884	76	8.60%
hbase-0.94.0	65	1059	218	20.59%
hbase-0.95.0	65	1669	383	22.95%
hbase-0.95.2	65	1834	483	26.34%
hive-0.10.0	65	1560	176	11.28%
hive-0.12.0	65	2662	213	8.00%
hive-0.9.0	65	1416	283	19.99%
jruby-1.1	65	731	87	11.90%
jruby-1.4.0	65	978	180	18.40%
jruby-1.5.0	65	1131	82	7.25%
jruby-1.7.0.	65	1614	87	5.39%
lucene-2.3.0	65	805	196	24.35%
lucene-2.9.0	65	1368	273	19.96%
lucene-3.0.0	65	1337	155	11.59%

TABLE 3 An overview of the used software metrics

Metric type	Metric name	Description
Code metrics (54)	CountDeclMethodPrivate	the number of local (not inherited) private methods
	AvgLineCode	the average number of lines containing source code for all nested functions or methods
	CountLine	the number of physical lines
	MaxCyclomatic	the maximum cyclomatic complexity of all nested functions or methods
	CountDeclMethodDefault	the number of local default methods
	AvgEssential	the average essential complexity for all nested functions or methods
	CountDeclClassVariable	the number of class variables
	SumCyclomaticStrict	the sum of strict cyclomatic complexity of all nested functions or methods
	AvgCyclomatic	the average cyclomatic complexity for all nested functions or methods
	AvgLine	the average number of lines for all nested functions or methods
	CountDeclClassMethod	the number of class methods
	AvgLineComment	the average number of lines containing comment for all nested functions or methods
	AvgCyclomaticModified	the modified McCabe Cyclomatic complexity
	CountDeclFunction	the number of functions
	CountLineComment	the number of lines containing comment
	CountDeclClass	the number of classes
	CountDeclMethod	the number of local (not inherited) methods
	SumCyclomaticModified	the sum of modified cyclomatic complexity of all nested functions or methods
	CountLineCodeDecl	the number of lines containing declarative source code
	CountDeclMethodProtected	the number of local protected methods
	CountDeclInstanceVariable	the number of instance variables
	MaxCyclomaticStrict	the maximum strict cyclomatic complexity of nested functions or methods

	CountDeclMethodPublic	the number of local (not inherited) public methods
	CountLineCodeExe	the number of lines containing executable source code
	SumCyclomatic	the sum of cyclomatic complexity of all nested functions or methods
	SumEssential	the sum of essential complexity of all nested functions or methods
	CountStmtDecl	the number of declarative statements
	CountLineCode	the number of lines containing source code
	CountStmtExe	the number of executable statements
	RatioCommentToCode	the ratio of comment lines to code lines
	CountLineBlank	the number of blank lines
	CountStmt	the number of statements
	MaxCyclomaticModified	the maximum modified cyclomatic complexity of nested functions or methods
	CountSemicolon	the number of semicolons
	AvgLineBlank	the average number of blank lines for all nested functions or methods
	CountDeclInstanceMethod	the number of instance methods
	AvgCyclomaticStrict	the average strict cyclomatic complexity for all nested functions or methods
	PercentLackOfCohesion	the 100% minus the average cohesion for package entities
	MaxInheritanceTree	the maximum depth of class in inheritance tree
	CountClassDerived	the number of immediate subclasses
	CountClassCoupled	the number of other classes to which a class is coupled
	CountClassBase	the number of immediate base classes
	CountInput (max, min, mean)	the number of calling subprograms plus global variables read
	CountOutput (max, min, mean)	the number of called subprograms plus global variables set
	CountPath (max, min, mean)	the number of unique paths trhough a body of code, not counting abnormal exits or gotos
	MaxNesting (max, min, mean)	the maximum nesting level of control constructs
Process metrics (5)	COMM	the number of Git commits
	ADEV	the number of active developers
	DDEV	the number of distinct developers
	Added_lines	the normalized number of lines added to the module
	Del_lines	the number normalized of lines deleted to the module
Ownership metrics (6)	OWN_LINE	the proportion of lines of code written by the developer who has the highest contribution of lines of code on the module
	OWN_COMMIT	the proportion of code changes (i.e., Git commits) made by the developer who has the highest contribution of code changes on the module
	MINOR_COMMIT	the number of unique developers who have contributed less than 5% of the total code changes (i.e., Git commits) on the module
	MINOR_LINE	the number of unique developers who have contributed less than 5% of the total lines of code on the module
	MAJOR_COMMIT	the number of unique developers who have contributed more than 5% of the total code changes (i.e., Git commits) on the module
	MAJOR_LINE	the number of unique developers who have contributed more than 5% of the total lines of code on the module

3 Metrics After Correlation and Redundancy Analysis

TABLE 4 An overview of the selected metrics used by AutoSpearman

Dataset	Metrics
activemq-5.0.0	AvgCyclomaticModified, AvgEssential, AvgLineBlank, AvgLineComment, CountClassBase, CountClassCoupled, CountClassDerived, CountDeclClass, CountDeclClassMethod, CountDeclClassVariable, CountDeclInstanceVariable, CountDeclMethodDefault, CountDeclMethodPrivat
activemq-5.1.0	AvgCyclomaticModified, AvgEssential, AvgLineBlank, AvgLineComment, CountClassBase, CountClassCoupled, CountClassDerived, CountDeclClass, CountDeclClassMethod, CountDeclClassVariable, CountDeclInstanceVariable, CountDeclMethodDefault, CountDeclMethodPrivate, CountDeclMethodProtected, CountDeclMethodPublic, CountInput_Mean, CountInput_Min, CountOutput_Min, MaxInheritanceTree, MaxNesting_Min, PercentLackOfCohesion, RatioCommentToCode, Added_lines, MAJOR_LINE, MINOR_COMMIT, MINOR_LINE, OWN_COMMIT
activemq-5.2.0	AvgCyclomaticModified, AvgEssential, AvgLineBlank, AvgLineComment, CountClassBase, CountClassCoupled, CountClassDerived, CountDeclClass, CountDeclClassMethod, CountDeclClassVariable, CountDeclInstanceVariable, CountDeclMethodDefault, CountDeclMethodPrivate, CountDeclMethodProtected, CountInput_Mean, CountInput_Min, CountLineComment, CountOutput_Min, MaxInheritanceTree, MaxNesting_Min, PercentLackOfCohesion, RatioCommentToCode, Added_lines, MAJOR_LINE, MINOR_COMMIT, MINOR_LINE, OWN_COMMIT
activemq-5.3.0	AvgCyclomaticModified, AvgEssential, AvgLineBlank, CountClassBase, CountClassCoupled, CountClassDerived, CountDeclClass, CountDeclClassMethod, CountDeclClassVariable, CountDeclInstanceVariable, CountDeclMethodDefault, CountDeclMethodPrivate, CountDeclMethodProtected, CountDeclMethodPublic, CountInput_Mean, CountInput_Min, CountLineComment, CountOutput_Mean, CountOutput_Min, MaxInheritanceTree, MaxNesting_Min, PercentLackOfCohesion, RatioCommentToCode, Added_lines, MAJOR_LINE, MINOR_COMMIT, MINOR_LINE, OWN_COMMIT
activemq-5.8.0	AvgCyclomaticModified, AvgEssential, AvgLineBlank, CountClassBase, CountClassCoupled, CountClassDerived, CountDeclClass, CountDeclClassMethod, CountDeclInstanceVariable, CountDeclMethodDefault, CountDeclMethodPrivate, CountDeclMethodProtected, CountDeclMethodPublic, CountInput_Mean, CountInput_Min, CountLineComment, CountOutput_Mean, CountOutput_Min, MaxInheritanceTree, MaxNesting_Min, PercentLackOfCohesion, RatioCommentToCode, Added_lines, DDEV, Del_lines, MAJOR_LINE, MINOR_COMMIT, MINOR_LINE, OWN_COMMIT
derby-10.2.1.6	AvgEssential, AvgLineBlank, AvgLineComment, CountClassBase, CountClassCoupled, CountClassDerived, CountDeclClass, CountDeclClassMethod, CountDeclClassVariable, CountDeclInstanceVariable, CountDeclMethodDefault, CountDeclMethodPrivate, CountDeclMethodProtected, CountDeclMethodPublic, CountInput_Mean, CountInput_Min, CountLineComment, CountOutput_Min, MaxInheritanceTree, MaxNesting_Mean, MaxNesting_Min, PercentLackOfCohesion, RatioCommentToCode, Added_lines, MAJOR_LINE, MINOR_COMMIT, OWN_COMMIT, OWN_LINE
derby-10.3.1.4	AvgEssential, AvgLineBlank, AvgLineComment, CountClassBase, CountClassCoupled, CountClassDerived, CountDeclClass, CountDeclClassMethod, CountDeclClassVariable, CountDeclInstanceVariable, CountDeclMethodDefault, CountDeclMethodPrivate, CountDeclMethodProtected, CountDeclMethodPublic, CountInput_Mean, CountInput_Min, CountLineComment, CountOutput_Min, CountPath_Min, MaxInheritanceTree, MaxNesting_Mean, PercentLackOfCohesion, RatioCommentToCode, Added_lines, MAJOR_COMMIT, MAJOR_LINE, MINOR_COMMIT, OWN_COMMIT, OWN_LINE

derby-10.5.1.1	AvgEssential, AvgLineBlank, AvgLineComment, CountClassBase, CountClassCoupled, CountClassDerived, CountDeclClass, CountDeclClassMethod, CountDeclClassVariable, CountDeclInstanceVariable, CountDeclMethodDefault, CountDeclMethodPrivate, CountDeclMethodProtected, CountDeclMethodPublic, CountInput_Mean, CountInput_Min, CountLineComment, CountOutput_Min, MaxInheritanceTree, MaxNesting_Mean, MaxNesting_Min, PercentLackOfCohesion, RatioCommentToCode, Added_lines, Del_lines, MAJOR_COMMIT, MAJOR_LINE, MINOR_COMMIT, MINOR_LINE
groovy-1_5_7	AvgCyclomaticStrict, AvgEssential, AvgLineBlank, AvgLineComment, CountClassBase, CountClassCoupled, CountClassDerived, CountDeclClass, CountDeclClassMethod, CountDeclClassVariable, CountDeclInstanceMethod, CountDeclInstanceVariable, CountDeclMethodDefault, CountDeclMethodPrivate, CountDeclMethodProtected, CountInput_Mean, CountInput_Min, CountOutput_Min, MaxInheritanceTree, MaxNesting_Min, RatioCommentToCode, Added_lines, MAJOR_LINE, MINOR_COMMIT, MINOR_LINE, OWN_COMMIT
groovy-1_6_beta1	AvgCyclomaticStrict, AvgEssential, AvgLineBlank, AvgLineComment, CountClassBase, CountClassCoupled, CountClassDerived, CountDeclClass, CountDeclClassVariable, CountDeclInstanceMethod, CountDeclInstanceVariable, CountDeclMethodDefault, CountDeclMethodPrivate, CountDeclMethodProtected, CountInput_Mean, CountInput_Min, CountLineComment, CountOutput_Min, MaxInheritanceTree, MaxNesting_Min, PercentLackOfCohesion, RatioCommentToCode, Added_lines, MAJOR_LINE, MINOR_COMMIT, MINOR_LINE, OWN_COMMIT
groovy-1_6_beata2	AvgCyclomaticStrict, AvgEssential, AvgLineBlank, AvgLineComment, CountClassBase, CountClassCoupled, CountClassDerived, CountDeclClass, CountDeclClassVariable, CountDeclInstanceMethod, CountDeclInstanceVariable, CountDeclMethodDefault, CountDeclMethodPrivate, CountDeclMethodProtected, CountInput_Mean, CountInput_Min, CountLineComment, CountOutput_Min, MaxInheritanceTree, MaxNesting_Min, PercentLackOfCohesion, RatioCommentToCode, Added_lines, MAJOR_COMMIT, MAJOR_LINE, MINOR_COMMIT, MINOR_LINE
hbase-0.94.0	AvgEssential, AvgLineBlank, AvgLineComment, CountClassBase, CountClassCoupled, CountClassDerived, CountDeclClassMethod, CountDeclClassVariable, CountDeclMethodDefault, CountDeclMethodProtected, CountInput_Mean, CountInput_Min, CountOutput_Mean, CountOutput_Min, CountPath_Min, MaxInheritanceTree, MaxNesting_Mean, MaxNesting_Min, PercentLackOfCohesion, RatioCommentToCode, Del_lines, MAJOR_COMMIT, MAJOR_LINE, MINOR_COMMIT, MINOR_LINE, OWN_COMMIT
hbase-0.95.0	AvgEssential, AvgLineBlank, AvgLineComment, CountClassBase, CountClassCoupled, CountClassDerived, CountDeclMethodDefault, CountDeclMethodProtected, CountInput_Mean, CountInput_Min, CountLineComment, CountOutput_Min, CountPath_Min, MaxInheritanceTree, MaxNesting_Mean, MaxNesting_Min, PercentLackOfCohesion, RatioCommentToCode, Added_lines, Del_lines, MAJOR_COMMIT, MAJOR_LINE, MINOR_COMMIT, MINOR_LINE, OWN_COMMIT
hbase-0.95.2	AvgEssential, AvgLineBlank, AvgLineComment, CountClassBase, CountClassCoupled, CountClassDerived, CountDeclClass, CountDeclMethodDefault, CountDeclMethodProtected, CountInput_Mean, CountInput_Min, CountLineComment, CountOutput_Min, CountPath_Min, MaxInheritanceTree, MaxNesting_Mean, MaxNesting_Min, PercentLackOfCohesion, RatioCommentToCode, Del_lines, MAJOR_COMMIT, MAJOR_LINE, MINOR_COMMIT, MINOR_LINE, OWN_COMMIT
hive-0.10.0	AvgEssential, AvgLineBlank, AvgLineComment, CountClassBase, CountClassCoupled, CountClassDerived, CountDeclInstanceVariable, CountDeclMethodPrivate, CountDeclMethodProtected, CountInput_Mean, CountInput_Min, CountLineComment, CountOutput_Min, CountPath_Min, MaxInheritanceTree, MaxNesting_Mean, MaxNesting_Min, PercentLackOfCohesion, RatioCommentToCode, Added_lines, MAJOR_LINE, MINOR_COMMIT, MINOR_LINE, OWN_COMMIT
hive-0.12.0	AvgEssential, AvgLineBlank, AvgLineComment, CountClassBase, CountClassCoupled, CountClassDerived, CountDeclClass, CountDeclInstanceVariable, CountDeclMethodDefault, CountDeclMethodPrivate, CountDeclMethodProtected, CountInput_Mean, CountInput_Min, CountLineComment, CountOutput_Min, CountPath_Min, MaxInheritanceTree, MaxNesting_Mean, MaxNesting_Min, PercentLackOfCohesion, RatioCommentToCode, Added_lines, MAJOR_COMMIT, MAJOR_LINE, MINOR_COMMIT, MINOR_LINE, OWN_COMMIT
hive-0.9.0	AvgEssential, AvgLineBlank, AvgLineComment, CountClassBase, CountClassCoupled, CountClassDerived, CountDeclClass, CountDeclClassMethod, CountDeclInstanceVariable, CountDeclMethodPrivate, CountInput_Mean, CountInput_Min, CountLineComment, CountOutput_Min, CountPath_Min, MaxInheritanceTree, MaxNesting_Mean, MaxNesting_Min, PercentLackOfCohesion, RatioCommentToCode, Added_lines, MAJOR_LINE, MINOR_COMMIT, MINOR_LINE, OWN_COMMIT
jruby-1.1	AvgEssential, AvgLineBlank, AvgLineComment, CountClassBase, CountClassCoupled, CountClassDerived, CountDeclClass, CountDeclClassMethod, CountDeclClassVariable, CountDeclInstanceMethod, CountDeclInstanceVariable, CountDeclMethodDefault, CountDeclMethodPrivate, CountDeclMethodProtected, CountInput_Mean, CountInput_Min, CountLineComment, CountOutput_Min, CountPath_Min, MaxInheritanceTree, MaxNesting_Mean, MaxNesting_Min, PercentLackOfCohesion, RatioCommentToCode, Added_lines, MAJOR_COMMIT, MAJOR_LINE, MINOR_COMMIT, MINOR_LINE, OWN_COMMIT
jruby-1.4.0	AvgEssential, AvgLineBlank, AvgLineComment, CountClassBase, CountClassCoupled, CountClassDerived, CountDeclClass, CountDeclClassMethod, CountDeclClassVariable, CountDeclInstanceMethod, CountDeclInstanceVariable, CountDeclMethodDefault, CountDeclMethodPrivate, CountDeclMethodProtected, CountInput_Mean, CountInput_Min, CountLineComment, CountOutput_Mean, CountOutput_Min, CountPath_Min, MaxInheritanceTree, MaxNesting_Mean, MaxNesting_Min, PercentLackOfCohesion, RatioCommentToCode, Added_lines, MAJOR_LINE, MINOR_COMMIT, MINOR_LINE, OWN_COMMIT
jruby-1.5.0	AvgEssential, AvgLineBlank, AvgLineComment, CountClassBase, CountClassCoupled, CountClassDerived, CountDeclClass, CountDeclClassMethod, CountDeclClassVariable, CountDeclInstanceMethod, CountDeclInstanceVariable, CountDeclMethodDefault, CountDeclMethodPrivate, CountDeclMethodProtected, CountInput_Mean, CountInput_Min, CountLineComment, CountOutput_Min, CountPath_Min, MaxInheritanceTree, MaxNesting_Mean, MaxNesting_Min, PercentLackOfCohesion, RatioCommentToCode, Added_lines, MAJOR_LINE, MINOR_COMMIT, MINOR_LINE, OWN_COMMIT
jruby-1.7.0.	AvgEssential, AvgLineBlank, AvgLineComment, CountClassBase, CountClassCoupled, CountClassDerived, CountDeclClass, CountDeclClassMethod, CountDeclClassVariable, CountDeclInstanceVariable, CountDeclMethodDefault, CountDeclMethodPrivate, CountDeclMethodProtected, CountInput_Mean, CountInput_Min, CountLineComment, CountOutput_Mean, CountOutput_Min, CountPath_Min, MaxInheritanceTree, MaxNesting_Mean, MaxNesting_Min, PercentLackOfCohesion, RatioCommentToCode, Added_lines, Del_lines, MAJOR_LINE, MINOR_COMMIT, MINOR_LINE, OWN_COMMIT
lucene-2.3.0	AvgEssential, AvgLineBlank, AvgLineComment, CountClassBase, CountClassCoupled, CountClassDerived, CountDeclClass, CountDeclClassMethod, CountDeclClassVariable, CountDeclInstanceVariable, CountDeclMethodDefault, CountDeclMethodPrivate, CountDeclMethodProtected, CountDeclMethodPublic, CountInput_Mean, CountInput_Min, CountOutput_Min, CountPath_Min, MaxInheritanceTree, MaxNesting_Mean, PercentLackOfCohesion, RatioCommentToCode, Added_lines, MAJOR_LINE, MINOR_COMMIT, MINOR_LINE, OWN_COMMIT
lucene-2.9.0	AvgEssential, AvgLineBlank, AvgLineComment, CountClassBase, CountClassCoupled, CountClassDerived, CountDeclClass, CountDeclClassMethod, CountDeclClassVariable, CountDeclInstanceVariable, CountDeclMethodDefault, CountDeclMethodPrivate, CountDeclMethodProtected, CountDeclMethodPublic, CountInput_Mean, CountInput_Min, CountOutput_Mean, CountOutput_Min, CountPath_Min, MaxInheritanceTree, MaxNesting_Mean, PercentLackOfCohesion, RatioCommentToCode, Added_lines, Del_lines, MAJOR_COMMIT, MAJOR_LINE, MINOR_COMMIT, MINOR_LINE, OWN_COMMIT
lucene-3.0.0	AvgEssential, AvgLineBlank, AvgLineComment, CountClassBase, CountClassCoupled, CountClassDerived, CountDeclClass, CountDeclClassMethod, CountDeclClassVariable, CountDeclInstanceVariable, CountDeclMethodDefault, CountDeclMethodPrivate, CountDeclMethodProtected, CountDeclMethodPublic, CountInput_Mean, CountInput_Min, CountLineComment, CountOutput_Mean, CountOutput_Min, MaxInheritanceTree, MaxNesting_Mean, MaxNesting_Min, PercentLackOfCohesion, RatioCommentToCode, Added_lines, Del_lines, MAJOR_COMMIT, MAJOR_LINE, MINOR_COMMIT, MINOR_LINE, OWN_COMMIT
activemq-5.0.0	AvgEssential, AvgLineBlank, CountClassBase, CountClassCoupled, CountClassDerived, CountDeclClass, CountDeclClassMethod, CountDeclClassVariable, CountDeclInstanceVariable, CountDeclMethodDefault, CountDeclMethodPrivate, CountDeclMethodProtected, CountDeclMethodPublic, CountInput_Mean, CountInput_Min, CountLineComment, CountOutput_Mean, CountOutput_Min, MaxInheritanceTree, MaxNesting_Mean, MaxNesting_Min, PercentLackOfCohesion, RatioCommentToCode, Added_lines, Del_lines, MAJOR_LINE, MINOR_COMMIT, MINOR_LINE, OWN_COMMIT
activemq-5.1.0	AvgCyclomaticModified, AvgEssential, AvgLineBlank, AvgLineComment, CountClassBase, CountClassCoupled, CountClassDerived, CountDeclClass, CountDeclClassMethod, CountDeclInstanceVariable, CountDeclMethodDefault, CountDeclMethodPrivate, CountDeclMethodProtected, CountInput_Mean, CountInput_Min, CountLineComment, CountOutput_Min, MaxInheritanceTree, MaxNesting_Min, PercentLackOfCohesion, RatioCommentToCode, Added_lines, Del_lines, MAJOR_LINE, MINOR_COMMIT, OWN_COMMIT, OWN_LINE
activemq-5.2.0	AvgCyclomaticModified, AvgEssential, AvgLineBlank, AvgLineComment, CountClassBase, CountClassCoupled, CountClassDerived, CountDeclClass, CountDeclClassMethod, CountDeclInstanceVariable, CountDeclMethodDefault, CountDeclMethodPrivate, CountDeclMethodProtected, CountInput_Mean, CountInput_Min, CountLineComment, CountOutput_Mean, CountOutput_Min, MaxInheritanceTree, MaxNesting_Min, PercentLackOfCohesion, RatioCommentToCode, Added_lines, Del_lines, MAJOR_LINE, MINOR_COMMIT, MINOR_LINE, OWN_COMMIT
activemq-5.3.0	AvgEssential, AvgLineBlank, AvgLineComment, CountClassBase, CountClassCoupled, CountClassDerived, CountDeclClass, CountDeclClassMethod, CountDeclInstanceVariable, CountDeclMethodDefault, CountDeclMethodPrivate, CountDeclMethodProtected, CountDeclMethodPublic, CountInput_Mean, CountInput_Min, CountOutput_Min, CountPath_Min, MaxInheritanceTree, MaxNesting_Mean, MaxNesting_Min, PercentLackOfCohesion, RatioCommentToCode, Added_lines, MAJOR_COMMIT, MAJOR_LINE, MINOR_COMMIT, MINOR_LINE

Prior work [18] points out that the correlated or redundant software metrics impact the performance and interpretation of defect models. Therefore, we conduct correlation and redundancy analyses prior to building our MUSDP models. Similar to the previous defect prediction studies [19] [20] [21], we use the *AutoSpearman* algorithm

[22] to remove the correlated and redundant metrics for data pre-processing. The rest of software metrics after performing the *AutoSpearman* algorithm are shown in Table 4. From the table, we can observe that more than half of the software metrics have been removed for each project. Subsequently, we use these metrics to preform unsupervised defect prediction experiments for MUSDP.

4 Results

4.1 Comparison of Unsupervised and Supervised Defect Prediction Methods

TABLE 5 Comparison results of each method using the NPSKESD test in terms of *Pd*

Dataset	CLA	CLAMI	SC	Kmeoids	MD	MU	TCL	TCLP	DNN	CNN	GRU	MVKNN	EASC	MUSDP_v1	MUSDP_v2
activemq-5.0.0	0.792	0.660	0.811	0.798	0.813	0.187	0.447	0.485	0.575	0.582	0.560	0.426	0.565	0.186	0.817
activemq-5.1.0	0.830	0.685	0.839	0.842	0.894	0.107	0.443	0.490	0.264	0.282	0.300	0.097	0.491	0.764	0.764
activemq-5.2.0	0.750	0.550	0.819	0.761	0.824	0.174	0.391	0.436	0.450	0.453	0.442	0.353	0.450	0.782	0.782
activemq-5.3.0	0.747	0.584	0.786	0.767	0.821	0.180	0.382	0.432	0.315	0.318	0.323	0.134	0.428	0.740	0.740
activemq-5.8.0	0.816	0.708	0.856	0.810	0.896	0.107	0.444	0.487	0.260	0.268	0.267	0.054	0.586	0.848	0.848
derby-10.2.1.6	0.707	0.460	0.675	0.692	0.742	0.255	0.266	0.306	0.665	0.663	0.634	0.609	0.651	0.245	0.755
derby-10.3.1.4	0.686	0.463	0.638	0.736	0.742	0.258	0.294	0.322	0.613	0.600	0.579	0.446	0.614	0.708	0.708
derby-10.5.1.1	0.771	0.605	0.741	0.786	0.830	0.170	0.373	0.451	0.408	0.411	0.421	0.149	0.679	0.609	0.609
groovy-1_5_7	0.800	0.613	0.833	0.857	0.806	0.185	0.696	0.696	0.333	0.304	0.375	0.000	0.750	0.818	0.818
groovy-1_6beta1	0.800	0.500	0.637	0.707	0.800	0.200	0.440	0.435	0.459	0.500	0.453	0.067	0.500	0.867	0.867
groovy-1_6beta2	0.746	0.435	0.692	0.722	0.771	0.229	0.414	0.406	0.429	0.481	0.483	0.042	0.465	0.787	0.787
hbase-0.94.0	0.755	0.477	0.774	0.822	0.808	0.191	0.299	0.325	0.474	0.489	0.509	0.325	0.761	0.753	0.753
hbase-0.95.0	0.659	0.278	0.648	0.662	0.697	0.305	0.218	0.252	0.511	0.516	0.476	0.233	0.628	0.523	0.519
hbase-0.95.2	0.691	0.309	0.665	0.602	0.730	0.270	0.224	0.259	0.450	0.454	0.432	0.314	0.641	0.532	0.532
hive-0.10.0	0.834	0.533	0.842	0.780	0.887	0.110	0.444	0.507	0.425	0.469	0.465	0.205	0.696	0.856	0.856
hive-0.12.0	0.715	0.379	0.738	0.719	0.739	0.262	0.229	0.253	0.329	0.360	0.335	0.023	0.548	0.812	0.812
hive-0.9.0	0.728	0.540	0.714	0.685	0.811	0.189	0.415	0.472	0.579	0.573	0.574	0.219	0.593	0.638	0.638
jruby-1.1	0.879	0.695	0.820	0.937	0.921	0.077	0.651	0.641	0.530	0.553	0.523	0.410	0.787	0.846	0.846
jruby-1.4.0	0.815	0.558	0.718	0.798	0.872	0.131	0.465	0.507	0.475	0.492	0.471	0.301	0.633	0.770	0.770
jruby-1.5.0	0.934	0.688	0.862	0.931	0.954	0.043	0.643	0.656	0.356	0.386	0.429	0.172	0.844	0.906	0.906
jruby-1.7.0	0.902	0.693	0.864	0.882	0.943	0.057	0.571	0.615	0.258	0.265	0.333	0.092	0.875	0.875	0.875
lucene-2.3.0	0.725	0.452	0.654	0.753	0.745	0.250	0.204	0.228	0.743	0.740	0.729	0.600	0.524	0.912	0.910
lucene-2.9.0	0.695	0.479	0.613	0.678	0.772	0.226	0.239	0.277	0.457	0.453	0.479	0.143	0.542	0.761	0.761
lucene-3.0.0	0.737	0.492	0.733	0.725	0.828	0.170	0.236	0.270	0.393	0.435	0.452	0.066	0.655	0.848	0.848
lucene-3.1	0.750	0.649	0.707	0.713	0.829	0.171	0.364	0.410	0.115	0.116	0.162	0.000	0.625	0.692	0.692
wicket-1.3.0beta2	0.880	0.703	0.806	0.842	0.912	0.088	0.478	0.496	0.260	0.279	0.283	0.070	0.571	0.769	0.769
wicket-1.3.0beta1	0.918	0.747	0.862	0.886	0.925	0.075	0.554	0.552	0.373	0.379	0.418	0.147	0.625	0.811	0.822
wicket-1.5.3	0.872	0.758	0.861	0.687	0.933	0.067	0.513	0.520	0.107	0.136	0.154	0.000	0.475	0.750	0.750
Median	0.773	0.562	0.764	0.765	0.828	0.171	0.402	0.444	0.420	0.433	0.434	0.161	0.595	0.770	0.781

(1) MD → ManualDown, and MU → ManualUp.

(2) The gray cells indicate the results that are better than others with statistical significance using the NPSKESD test in each row. If multiple treatments tie for the "best", then there will be multiple gray cells in that row (the same below).

TABLE 6 Comparison results of each method using the NPSKESD test in terms of *Pf*

Dataset	CLA	CLAMI	SC	Kmeodids	MD	MU	TCL	TCLP	DNN	CNN	GRU	MVKNN	EASC	MUSDP_v1	MUSDP_v2
activemq-5.0.0	0.327	0.222	0.292	0.385	0.443	0.557	0.069	0.091	0.058	0.069	0.073	0.030	0.110	0.800	0.198
activemq-5.1.0	0.341	0.251	0.343	0.415	0.466	0.533	0.079	0.114	0.031	0.043	0.046	0.004	0.096	0.270	0.270
activemq-5.2.0	0.327	0.239	0.336	0.401	0.461	0.539	0.065	0.103	0.042	0.055	0.061	0.012	0.078	0.167	0.167
activemq-5.3.0	0.322	0.228	0.330	0.420	0.460	0.539	0.071	0.107	0.055	0.062	0.070	0.005	0.085	0.222	0.222
activemq-5.8.0	0.343	0.220	0.346	0.410	0.475	0.525	0.081	0.100	0.030	0.032	0.038	0.001	0.147	0.304	0.304
derby-10.2.1.6	0.288	0.131	0.234	0.405	0.377	0.623	0.039	0.069	0.134	0.158	0.128	0.083	0.254	0.856	0.144
derby-10.3.1.4	0.309	0.149	0.261	0.442	0.395	0.604	0.056	0.080	0.147	0.148	0.167	0.078	0.246	0.243	0.243
derby-10.5.1.1	0.341	0.185	0.297	0.415	0.446	0.553	0.061	0.100	0.077	0.083	0.094	0.007	0.244	0.224	0.224
groovy-1_5_7	0.368	0.092	0.323	0.448	0.488	0.510	0.101	0.122	0.012	0.024	0.042	0.000	0.150	0.295	0.295
groovy-1_6beta1	0.380	0.210	0.302	0.436	0.471	0.527	0.094	0.117	0.029	0.042	0.068	0.000	0.163	0.299	0.299
groovy-1_6beta2	0.369	0.193	0.312	0.474	0.473	0.526	0.098	0.124	0.029	0.040	0.060	0.000	0.155	0.267	0.267
hbase-0.94.0	0.363	0.177	0.349	0.465	0.419	0.578	0.069	0.091	0.094	0.117	0.134	0.049	0.345	0.248	0.248
hbase-0.95.0	0.373	0.110	0.369	0.454	0.441	0.558	0.068	0.096	0.101	0.115	0.120	0.030	0.319	0.209	0.209
hbase-0.95.2	0.346	0.107	0.330	0.494	0.415	0.583	0.060	0.083	0.155	0.169	0.148	0.053	0.291	0.190	0.190
hive-0.10.0	0.364	0.116	0.314	0.350	0.449	0.550	0.085	0.107	0.046	0.064	0.104	0.010	0.179	0.288	0.288
hive-0.12.0	0.407	0.178	0.388	0.456	0.479	0.521	0.091	0.138	0.035	0.040	0.051	0.000	0.275	0.361	0.361
hive-0.9.0	0.328	0.079	0.293	0.386	0.420	0.578	0.040	0.063	0.081	0.088	0.121	0.017	0.138	0.283	0.283
jruby-1.1	0.367	0.127	0.265	0.476	0.444	0.554	0.059	0.081	0.042	0.063	0.076	0.013	0.200	0.280	0.280
jruby-1.4.0	0.320	0.131	0.213	0.395	0.417	0.583	0.055	0.073	0.082	0.108	0.112	0.025	0.149	0.312	0.312
jruby-1.5.0	0.375	0.161	0.268	0.421	0.464	0.535	0.088	0.114	0.026	0.036	0.048	0.005	0.193	0.369	0.369
jruby-1.7.0	0.396	0.169	0.280	0.436	0.474	0.525	0.091	0.119	0.025	0.029	0.055	0.002	0.379	0.387	0.387
lucene-2.3.0	0.358	0.204	0.276	0.377	0.420	0.578	0.077	0.090	0.071	0.076	0.095	0.050	0.215	0.291	0.291
lucene-2.9.0	0.385	0.202	0.280	0.451	0.431	0.568	0.092	0.105	0.099	0.112	0.152	0.022	0.236	0.331	0.331
lucene-3.0.0	0.396	0.229	0.324	0.477	0.458	0.541	0.106	0.122	0.047	0.058	0.095	0.002	0.301	0.330	0.330
lucene-3.1	0.401	0.264	0.328	0.498	0.487	0.513	0.102	0.133	0.015	0.017	0.049	0.000	0.256	0.342	0.342
wicket-1.3.0beta2	0.382	0.227	0.287	0.434	0.466	0.532	0.090	0.110	0.038	0.047	0.054	0.003	0.115	0.266	0.266
wicket-1.3.0beta1	0.397	0.230	0.298	0.454	0.473	0.526	0.092	0.111	0.027	0.028	0.040	0.003	0.124	0.281	0.280
wicket-1.5.3	0.393	0.256	0.319	0.481	0.482	0.518	0.103	0.127	0.016	0.021	0.027	0.000	0.089	0.323	0.323
Median	0.362	0.192	0.306	0.435	0.458	0.542	0.080	0.103	0.048	0.058	0.075	0.007	0.169	0.290	0.280

Tables 5-10 show the median values of each method in terms of *Pd*, *Pf*, *F-measure*, *MCC*, *F-measure@20%*, and *IFA*. These results are subjected to the NPSKESD test.

In the context of unsupervised setting, ManualDown exhibits the best performance among all the unsupervised methods across 28 datasets in terms of *Pd*, closely followed by our proposed approaches, particularly for the MUSDP_v2. As for *Pf* in Table 6, TCL stands out as the method with the best performance among them, closely followed by TCLP and CLAMI. According to Menzies et al.'s study [23], they reported average defect prediction results with a *Pd* of 71% and a *Pf* of 25%, and they argued these prediction results are useful in practice. Comparing this with MUSDP_v1 and MUSDP_v2, the overall median results across the 28 datasets are 0.770 and 0.781 for *Pd*, and 0.290 and 0.280 for *Pf*, respectively. These findings are very closely aligned with Menzies et al.'s study, highlighting the effectiveness of our proposed approaches compared to the unsupervised CLA, CLAMI, SC, Kmedoids, ManualDown, ManualUp, TCL, and TCLP methods.

TABLE 7 Comparison results of each method using the NPSKESD test in terms of *F-measure*

Dataset	CLA	CLAMI	SC	Kmeadoids	MD	MU	TCL	TCLP	DNN	CNN	GRU	MVKNN	EASC	MUSDP_v1	MUSDP_v2
activemq-5.0.0	0.442	0.450	0.474	0.408	0.382	0.089	0.484	0.482	0.601	0.596	0.564	0.532	0.518	0.067	0.562
activemq-5.1.0	0.284	0.281	0.286	0.252	0.242	0.029	0.371	0.349	0.316	0.314	0.327	0.169	0.376	0.307	0.307
activemq-5.2.0	0.336	0.302	0.357	0.297	0.292	0.061	0.406	0.380	0.496	0.472	0.446	0.481	0.432	0.492	0.492
activemq-5.3.0	0.339	0.318	0.349	0.287	0.292	0.065	0.389	0.369	0.357	0.340	0.338	0.230	0.404	0.416	0.416
activemq-5.8.0	0.230	0.265	0.236	0.191	0.192	0.023	0.325	0.321	0.309	0.307	0.289	0.100	0.305	0.256	0.256
derby-10.2.1.6	0.619	0.533	0.628	0.561	0.598	0.206	0.397	0.423	0.689	0.674	0.670	0.685	0.591	0.167	0.738
derby-10.3.1.4	0.571	0.516	0.571	0.551	0.560	0.194	0.416	0.429	0.626	0.609	0.586	0.549	0.548	0.628	0.628
derby-10.5.1.1	0.397	0.440	0.414	0.370	0.362	0.075	0.424	0.432	0.430	0.422	0.414	0.252	0.412	0.401	0.401
groovy-1_5_7	0.134	0.165	0.154	0.117	0.106	0.027	0.311	0.280	0.375	0.300	0.294	0.000	0.265	0.168	0.168
groovy-1_6beta1	0.268	0.273	0.264	0.212	0.234	0.059	0.361	0.326	0.511	0.505	0.411	0.121	0.308	0.340	0.340
groovy-1_6beta2	0.258	0.224	0.282	0.216	0.232	0.069	0.338	0.297	0.491	0.504	0.452	0.075	0.297	0.340	0.340
hbase-0.94.0	0.476	0.459	0.503	0.445	0.473	0.112	0.379	0.385	0.508	0.503	0.493	0.423	0.490	0.553	0.553
hbase-0.95.0	0.453	0.339	0.443	0.410	0.439	0.188	0.298	0.319	0.545	0.536	0.503	0.345	0.466	0.469	0.469
hbase-0.95.2	0.522	0.380	0.517	0.406	0.508	0.185	0.324	0.348	0.484	0.468	0.466	0.428	0.522	0.517	0.517
hive-0.10.0	0.358	0.405	0.397	0.356	0.333	0.040	0.421	0.434	0.465	0.465	0.416	0.308	0.456	0.414	0.414
hive-0.12.0	0.223	0.215	0.238	0.219	0.207	0.073	0.202	0.183	0.381	0.395	0.332	0.045	0.233	0.271	0.271
hive-0.9.0	0.480	0.551	0.496	0.440	0.469	0.109	0.529	0.552	0.607	0.585	0.544	0.341	0.548	0.455	0.455
jruby-1.1	0.381	0.519	0.430	0.339	0.348	0.029	0.621	0.571	0.566	0.538	0.484	0.538	0.491	0.418	0.418
jruby-1.4.0	0.506	0.520	0.544	0.441	0.468	0.071	0.545	0.555	0.515	0.503	0.480	0.429	0.551	0.483	0.483
jruby-1.5.0	0.277	0.359	0.327	0.252	0.243	0.009	0.463	0.425	0.422	0.408	0.414	0.265	0.397	0.275	0.275
jruby-1.7.0	0.202	0.295	0.255	0.183	0.183	0.012	0.367	0.332	0.305	0.295	0.295	0.162	0.204	0.200	0.200
lucene-2.3.0	0.511	0.439	0.522	0.521	0.489	0.165	0.284	0.298	0.757	0.742	0.716	0.679	0.473	0.643	0.641
lucene-2.9.0	0.433	0.418	0.449	0.405	0.445	0.130	0.302	0.326	0.489	0.475	0.443	0.236	0.437	0.493	0.493
lucene-3.0.0	0.309	0.308	0.346	0.262	0.308	0.064	0.222	0.243	0.447	0.462	0.397	0.123	0.324	0.384	0.384
lucene-3.1	0.125	0.156	0.143	0.099	0.119	0.025	0.188	0.176	0.151	0.147	0.122	0.000	0.158	0.134	0.134
wicket-1.3.0beta2	0.264	0.309	0.295	0.214	0.238	0.022	0.367	0.343	0.295	0.293	0.278	0.124	0.372	0.301	0.301
wicket-1.3.0beta1	0.230	0.277	0.265	0.184	0.195	0.017	0.366	0.332	0.405	0.409	0.403	0.247	0.349	0.262	0.265
wicket-1.5.3	0.158	0.196	0.182	0.102	0.138	0.011	0.256	0.228	0.144	0.164	0.170	0.000	0.262	0.155	0.155
Median	0.334	0.337	0.360	0.284	0.304	0.063	0.369	0.357	0.459	0.452	0.424	0.263	0.410	0.368	0.400

In the context of supervised setting, the proposed approaches outperform the supervised DNN, CNN, GRU, MVKNN, and EASC methods with statistical significance, exhibiting higher *Pd* values. However, when considering the *Pf*, MVKNN is statistically significantly better than other methods across 28 datasets. Both MUSDP_v1 and MUSDP_v2 achieve lower scores compared to the supervised methods, suggesting they seem to scarify *Pf* to obtain higher Recall. Nonetheless, the proposed approaches still maintain acceptable *Pf* values at 0.290 and 0.280, respectively.

In summary, MUSDP_v2 and MUSDP_v1 have demonstrated statistical superiority over the competing supervised and unsupervised methods regarding *Pd*. However, in the case of the *Pf* metric, they demonstrate lower performance compared to most of these methods with statistical significance. When considering *Pd* and *Pf* simultaneously, like *G-mean* (in the paper), the proposed approaches show superior performance with statistical significance compared to the competing supervised and unsupervised defect prediction methods, with MUSDP_v2 emerging as the top performer among them. Nonetheless, our proposed approaches exhibit great potential, particularly in scenarios where historical

defect data for projects are unavailable or limited. They can serve as valuable tools for effectively identifying potential software defects when traditional supervised defect prediction methods may not be feasible in the case of insufficient historical defect data.

TABLE 8 Comparison results of each method using the NPSKESD test in terms of *MCC*

Dataset	CLA	CLAMI	SC	Kmedoids	MD	MU	TCL	TCLP	DNN	CNN	GRU	MVKNN	EASC	MUSDP_v1	MUSDP_v2
activemq-5.0.0	0.343	0.329	0.383	0.312	0.269	-0.268	0.404	0.394	0.537	0.528	0.488	0.489	0.426	-0.485	0.489
activemq-5.1.0	0.271	0.246	0.274	0.234	0.229	-0.228	0.310	0.288	0.284	0.268	0.275	0.228	0.320	0.284	0.284
activemq-5.2.0	0.271	0.211	0.311	0.230	0.224	-0.223	0.339	0.297	0.456	0.417	0.385	0.498	0.358	0.444	0.444
activemq-5.3.0	0.273	0.228	0.292	0.211	0.226	-0.225	0.321	0.286	0.292	0.275	0.261	0.293	0.326	0.356	0.356
activemq-5.8.0	0.232	0.256	0.250	0.200	0.200	-0.199	0.280	0.281	0.280	0.268	0.250	0.201	0.274	0.273	0.273
derby-10.2.1.6	0.399	0.363	0.425	0.288	0.347	-0.347	0.338	0.316	0.542	0.514	0.525	0.564	0.383	-0.603	0.603
derby-10.3.1.4	0.349	0.342	0.361	0.298	0.316	-0.316	0.330	0.311	0.472	0.449	0.412	0.436	0.369	0.446	0.446
derby-10.5.1.1	0.304	0.341	0.319	0.268	0.265	-0.265	0.351	0.338	0.344	0.330	0.321	0.306	0.333	0.293	0.293
groovy-1_5_7	0.165	0.202	0.198	0.144	0.118	-0.118	0.336	0.309	0.395	0.293	0.284	0.000	0.309	0.207	0.207
groovy-1_6beta1	0.234	0.199	0.203	0.141	0.183	-0.183	0.299	0.254	0.475	0.469	0.356	0.209	0.235	0.331	0.331
groovy-1_6beta2	0.212	0.168	0.227	0.145	0.173	-0.171	0.277	0.223	0.469	0.459	0.402	0.184	0.225	0.311	0.311
hbase-0.94.0	0.315	0.316	0.358	0.290	0.312	-0.312	0.287	0.271	0.410	0.378	0.361	0.362	0.341	0.424	0.424
hbase-0.95.0	0.245	0.196	0.232	0.170	0.214	-0.214	0.205	0.196	0.428	0.412	0.370	0.317	0.265	0.296	0.296
hbase-0.95.2	0.308	0.234	0.299	0.090	0.280	-0.278	0.242	0.235	0.314	0.290	0.302	0.355	0.316	0.338	0.338
hive-0.10.0	0.306	0.328	0.347	0.301	0.279	-0.279	0.344	0.352	0.417	0.404	0.346	0.337	0.391	0.379	0.379
hive-0.12.0	0.166	0.133	0.195	0.172	0.143	-0.142	0.124	0.091	0.340	0.352	0.281	0.113	0.164	0.247	0.247
hive-0.9.0	0.325	0.459	0.346	0.267	0.313	-0.312	0.471	0.471	0.515	0.485	0.438	0.355	0.432	0.290	0.290
jruby-1.1	0.334	0.458	0.381	0.287	0.302	-0.305	0.567	0.506	0.517	0.476	0.421	0.533	0.434	0.378	0.378
jruby-1.4.0	0.389	0.407	0.432	0.316	0.354	-0.353	0.477	0.469	0.419	0.390	0.366	0.409	0.437	0.363	0.363
jruby-1.5.0	0.294	0.336	0.336	0.255	0.255	-0.255	0.434	0.396	0.394	0.375	0.375	0.317	0.396	0.286	0.286
jruby-1.7.0	0.230	0.293	0.280	0.205	0.212	-0.212	0.346	0.315	0.275	0.264	0.264	0.244	0.226	0.222	0.222
lucene-2.3.0	0.313	0.247	0.335	0.329	0.276	-0.279	0.177	0.179	0.684	0.665	0.627	0.602	0.291	0.529	0.529
lucene-2.9.0	0.252	0.251	0.280	0.206	0.272	-0.273	0.184	0.199	0.371	0.356	0.295	0.244	0.264	0.345	0.345
lucene-3.0.0	0.216	0.194	0.269	0.159	0.235	-0.236	0.120	0.134	0.393	0.397	0.318	0.188	0.232	0.335	0.335
lucene-3.1	0.136	0.169	0.154	0.084	0.131	-0.131	0.162	0.156	0.134	0.138	0.085	0.000	0.163	0.137	0.137
wicket-1.3.0beta2	0.261	0.286	0.291	0.211	0.232	-0.232	0.316	0.294	0.251	0.252	0.225	0.191	0.333	0.284	0.284
wicket-1.3.0beta1	0.256	0.279	0.283	0.208	0.215	-0.214	0.333	0.305	0.379	0.381	0.368	0.320	0.328	0.275	0.280
wicket-1.5.3	0.194	0.224	0.224	0.080	0.178	-0.178	0.242	0.221	0.133	0.147	0.141	0.000	0.247	0.176	0.176
Median	0.272	0.265	0.295	0.213	0.240	-0.239	0.311	0.287	0.388	0.371	0.336	0.306	0.319	0.304	0.320

From tables 7 and 8, in terms of *F-measure* and *MCC*, MUSDP demonstrates superior prediction performance compared to the unsupervised CLA, CLAMI, SC, Kmedoids, ManualDown, ManualUp, and TCLP, except for TCL exhibits comparable performance. Among them, the MUSDP_v2 approach stands out as the method with the best results among them. When considering the supervised DNN, CNN, GRU, MVKNN, and EASC methods, MUSDP_v2 and MUSDP_v1 demonstrate lower scores compared to these methods, except for the comparison with MVKNN. These findings highlight the efficacy of MUSDP as an effective alternative for defect prediction, especially in the scenarios where labeled training data is limited or unavailable.

TABLE 9 Comparison results of each method using the NPSKESD test in terms of *F-measure@20%*

Dataset	CLA	CLAMI	SC	Kmedoids	MD	MU	TCL	TCLP	DNN	CNN	GRU	MVKNN	EASC	MUSDP_v1	MUSDP_v2
activemq-5.0.0	0.200	0.208	0.244	0.174	0.239	0.098	0.315	0.284	0.513	0.505	0.473	0.476	0.318	0.053	0.366
activemq-5.1.0	0.162	0.145	0.163	0.121	0.253	0.041	0.259	0.188	0.289	0.293	0.300	0.188	0.240	0.184	0.184
activemq-5.2.0	0.181	0.167	0.209	0.142	0.167	0.074	0.320	0.232	0.471	0.447	0.423	0.417	0.327	0.376	0.376
activemq-5.3.0	0.192	0.175	0.204	0.138	0.232	0.076	0.251	0.189	0.329	0.309	0.312	0.201	0.241	0.277	0.277
activemq-5.8.0	0.113	0.154	0.125	0.081	0.261	0.028	0.216	0.195	0.257	0.255	0.255	0.176	0.170	0.142	0.142
derby-10.2.1.6	0.394	0.316	0.416	0.344	0.071	0.286	0.279	0.261	0.549	0.537	0.542	0.563	0.371	0.152	0.587
derby-10.3.1.4	0.357	0.299	0.350	0.352	0.092	0.267	0.273	0.250	0.492	0.485	0.452	0.443	0.352	0.427	0.427
derby-10.5.1.1	0.213	0.251	0.223	0.189	0.134	0.111	0.294	0.258	0.356	0.353	0.344	0.279	0.268	0.272	0.272
groovy-1_5_7	0.028	0.082	0.051	0.031	0.308	0.020	0.179	0.129	0.308	0.261	0.250	0.069	0.146	0.056	0.056
groovy-1_6beta1	0.168	0.157	0.157	0.130	0.127	0.097	0.255	0.233	0.488	0.485	0.384	0.123	0.205	0.266	0.266
groovy-1_6beta2	0.164	0.116	0.174	0.126	0.121	0.106	0.243	0.188	0.467	0.480	0.437	0.129	0.190	0.273	0.273
hbase-0.94.0	0.313	0.329	0.362	0.299	0.050	0.168	0.299	0.275	0.469	0.462	0.434	0.399	0.336	0.426	0.426
hbase-0.95.0	0.366	0.288	0.376	0.334	0.000	0.288	0.279	0.283	0.533	0.531	0.496	0.411	0.388	0.429	0.429
hbase-0.95.2	0.413	0.310	0.432	0.341	0.011	0.307	0.275	0.281	0.452	0.440	0.431	0.414	0.426	0.453	0.453
hive-0.10.0	0.252	0.317	0.285	0.267	0.000	0.097	0.361	0.352	0.414	0.436	0.375	0.313	0.366	0.335	0.335
hive-0.12.0	0.197	0.196	0.219	0.196	0.000	0.105	0.185	0.148	0.199	0.214	0.199	0.088	0.207	0.239	0.239
hive-0.9.0	0.284	0.416	0.305	0.266	0.020	0.176	0.427	0.407	0.517	0.509	0.458	0.325	0.360	0.314	0.314
jruby-1.1	0.151	0.233	0.169	0.138	0.135	0.074	0.407	0.336	0.436	0.414	0.374	0.465	0.208	0.180	0.180
jruby-1.4.0	0.298	0.331	0.334	0.258	0.085	0.158	0.374	0.371	0.393	0.389	0.367	0.345	0.359	0.307	0.307
jruby-1.5.0	0.121	0.167	0.143	0.111	0.136	0.047	0.292	0.219	0.383	0.372	0.370	0.208	0.211	0.110	0.110
jruby-1.7.0	0.088	0.167	0.135	0.086	0.188	0.030	0.242	0.216	0.275	0.262	0.261	0.077	0.103	0.101	0.101
lucene-2.3.0	0.360	0.313	0.375	0.393	0.073	0.226	0.234	0.240	0.697	0.682	0.648	0.639	0.346	0.501	0.500
lucene-2.9.0	0.281	0.290	0.312	0.271	0.081	0.185	0.218	0.219	0.437	0.433	0.396	0.293	0.309	0.362	0.362
lucene-3.0.0	0.224	0.254	0.283	0.192	0.063	0.105	0.188	0.205	0.415	0.442	0.380	0.284	0.253	0.323	0.323
lucene-3.1	0.081	0.103	0.102	0.062	0.143	0.029	0.146	0.124	0.083	0.083	0.092	0.048	0.104	0.085	0.085
wicket-1.3.0beta2	0.126	0.192	0.167	0.091	0.203	0.035	0.263	0.219	0.271	0.277	0.255	0.140	0.257	0.176	0.176
wicket-1.3.0beta1	0.106	0.152	0.149	0.081	0.275	0.032	0.256	0.208	0.369	0.380	0.384	0.159	0.231	0.138	0.148
wicket-1.5.3	0.073	0.121	0.107	0.043	0.172	0.022	0.188	0.151	0.096	0.105	0.112	0.096	0.201	0.082	0.082
Median	0.195	0.217	0.218	0.166	0.118	0.094	0.267	0.235	0.407	0.407	0.372	0.268	0.268	0.252	0.280

As indicated in Table 9, with respect to the *F-measure@20%*, the proposed approaches exhibit superior performance compared to the unsupervised methods CLA, CLAMI, SC, Kmedoids, ManualDown, ManualUP, TCL, and TCLP. However, when compared to supervised models like DNN, CNN, and GRU, the proposed approaches yield relatively lower values, with the exception of MVKNN and EASC, which achieved better or comparable results. Given that supervised methods rely on labeled training data, the performance of the proposed approaches for UDP remains effective and valuable, making them a powerful solution.

From Table 10, it is evident that MUSDP yields poor *IFA* results compared with other methods. The simple ManualDown method emerges as the top performer among all compared methods. This can be attributed to its reliance on module size, which assumes that modules with larger LOC are more prone to containing defects. Considering that the predicted defective models are firstly ranked in descending order according to the predicted density values, modules with the larger LOC are typically more likely to be defective. On the contrary, the ManualUp method performs the worst in

terms of *IFA* among all the comparison methods. Besides, supervised methods exhibit superior *IFA* values compared to unsupervised methods. Hence, future research should prioritize enhancing the *IFA* value of unsupervised methods.

TABLE 10 Comparison results of each method using the NPSKESD test in terms of *IFA*

Dataset	CLA	CLAMI	SC	Kmeoids	MD	MU	TCL	TCLP	DNN	CNN	GRU	MVKNN	EASC	MUSDP_v1	MUSDP_v2
activemq-5.0.0	2.5	5	6	8	1	15	1	5	1	2	1	2	3	18	2
activemq-5.1.0	9	23.5	21	35	1	153	3	7.5	2	3	3	2	2	18	18
activemq-5.2.0	3	6	4	13	2	63	1	3	1	1.5	3	1	1	1	1
activemq-5.3.0	3	5	5	16.5	1	96.5	4	4	7	8	7	2	3	10	10
activemq-5.8.0	5	8	9	28.5	1	82.5	6	5	5	4.5	4	1	2	7.5	7.5
derby-10.2.1.6	1	2	1	2	1	2	1	2	2	4	2	2	4	2	1
derby-10.3.1.4	1	2	2	3	1	58	1	2	3	3	3	2	2	2	2
derby-10.5.1.1	2	3	5	5	1	67.5	2	4	3	4	3	1	6.5	47	47
groovy-1_5_7	49.5	14.5	27	39	1	72	9.5	11.5	2.5	5	8	20.5	5	33	33
groovy-1_6beta1	5	9	6	17	1	48	3	4.5	1.5	2	3	1	5	5	5
groovy-1_6beta2	6.5	10	10.5	21	1	51	5	6	1	2	1.5	1	9	7	7
hbase-0.94.0	2	2	5.5	9.5	2	11	2	2	3	3	3.5	2	3	4	4
hbase-0.95.0	2	2	9.5	7	2	14	3	2	1	2	2	1	2	2	2
hbase-0.95.2	4	2	11	12	2	28	1	1	5	6	4	2	2	5	5
hive-0.10.0	11.5	4	13	18	3	44.5	1	3	2	2.5	3	2	3	11	11
hive-0.12.0	4	4	7	12.5	4	2	2.5	3	1	1	1	1	6	6	6
hive-0.9.0	3	3	11	15	1	9	1	2	1	1	2	1	3	22	22
jruby-1.1	5	5	5	14	2	70	2	5	3	5	7	1	6	4.5	4.5
jruby-1.4.0	4	4	3	12	1	27	2	2	2	4	4	2	6	15.5	15.5
jruby-1.5.0	23	16.5	23	49	1	99	7	11.5	3	3	5	2	6	44.5	44.5
jruby-1.7.0	46	10.5	22	58.5	1	202.5	10	14.5	4	5	7	1	47.5	43	43
lucene-2.3.0	2	2	2	7	1	30	2	2	2	2	2	2	1	4	4
lucene-2.9.0	5	2	2	9	1	49	2	3	3	4	6.5	2	2	6	6
lucene-3.0.0	4.5	3	5.5	23	4	107	2	4	1	2	6	1	5	7	7
lucene-3.1	46	9	16	67	1.5	173	12	11	7	7	22	23.5	12	48.5	48.5
wicket-1.3.0beta2	4	11	6.5	37	1	210	3	4	4	3	6	2	1.5	9	9
wicket-1.3.0beta1	5	9.5	4	34	1	158	3	4	4	4	6	1	4	17.5	13
wicket-1.5.3	29	11	17	97	1	109.5	3.5	9	6	7	9.5	11	5	46.5	46.5
Median	5	5	8	17	1	56	2	4	2	3	4	2	4	9	8

4.2 Comparison of MUSDP and Its Variants

Tables 11 and 14 present the median results of MUSDP and its variants for each dataset based on 100 repetitions in terms of *G-mean*, *AUC*, P_{opt}, and *Recall@20%*. These results are subjected to the NPSKESD test for statistical analysis. Based on this, we can observe that:

In terms of the traditional non-effort-aware evaluation measures *G-mean* and *AUC*, MUSDP_v1 with CP and CPO demonstrate superior performance with statistical significance compared to the other variants, highlighting their effectiveness in unsupervised defect prediction. This suggests that relying solely on code metrics and process metrics

yields better results than using each metric individually. Our findings align with previous studies [24] [25], which have pointed out that combining code metrics and process metrics can improve the performance of defect prediction, indicating their complementarity. Furthermore, while the addition of ownership metrics does not lead to significant improvement, it still helps maintain performance. Hence, the combination of these three types of metrics proves valuable in identifying potential software defects without the need for labeled training data. Similarly, MUSDP_v2 exhibits the same trend as MUSDP_v1 in this regard.

TABLE 11 Comparison results of MUSDP against its variants with the NPSKESD test in terms of *G-mean*

Dataset	MUSDP_v1							MUSDP_v2						
	C	P	O	CP	CO	PO	CPO	C	P	O	CP	CO	PO	CPO
activemq-5.0.0	0.697	0.801	0.772	0.791	0.775	0.180	0.193	0.697	0.801	0.772	0.791	0.775	0.814	0.809
activemq-5.1.0	0.692	0.554	0.685	0.743	0.731	0.678	0.744	0.692	0.544	0.685	0.743	0.731	0.678	0.744
activemq-5.2.0	0.669	0.567	0.764	0.749	0.778	0.765	0.806	0.669	0.413	0.764	0.749	0.778	0.765	0.806
activemq-5.3.0	0.710	0.235	0.749	0.738	0.739	0.662	0.758	0.710	0.284	0.749	0.738	0.739	0.662	0.758
activemq-5.8.0	0.719	0.747	0.696	0.772	0.745	0.764	0.767	0.719	0.747	0.696	0.772	0.745	0.764	0.767
derby-10.2.1.6	0.664	0.456	0.212	0.712	0.226	0.198	0.189	0.664	0.490	0.778	0.712	0.772	0.794	0.803
derby-10.3.1.4	0.655	0.422	0.647	0.678	0.725	0.646	0.735	0.655	0.431	0.288	0.678	0.725	0.289	0.735
derby-10.5.1.1	0.688	0.459	0.650	0.717	0.728	0.522	0.683	0.688	0.459	0.650	0.717	0.728	0.522	0.683
groovy-1_5_7	0.737	0.314	0.748	0.771	0.720	0.774	0.767	0.737	0.264	0.748	0.771	0.720	0.774	0.767
groovy-1_6beta1	0.682	0.347	0.714	0.746	0.721	0.756	0.778	0.682	0.287	0.714	0.746	0.717	0.756	0.778
groovy-1_6beta2	0.642	0.341	0.399	0.722	0.672	0.768	0.760	0.641	0.452	0.592	0.722	0.672	0.768	0.760
hbase-0.94.0	0.670	0.382	0.689	0.646	0.757	0.722	0.752	0.666	0.382	0.286	0.646	0.757	0.722	0.752
hbase-0.95.0	0.581	0.530	0.549	0.582	0.669	0.649	0.642	0.416	0.530	0.449	0.582	0.669	0.649	0.642
hbase-0.95.2	0.571	0.373	0.517	0.481	0.683	0.653	0.656	0.541	0.373	0.381	0.481	0.683	0.653	0.656
hive-0.10.0	0.689	0.455	0.746	0.747	0.764	0.767	0.783	0.689	0.424	0.746	0.747	0.764	0.767	0.783
hive-0.12.0	0.585	0.403	0.672	0.705	0.679	0.674	0.717	0.575	0.350	0.088	0.705	0.268	0.088	0.717
hive-0.9.0	0.584	0.473	0.655	0.696	0.691	0.617	0.673	0.581	0.463	0.655	0.696	0.691	0.617	0.673
jruby-1.1	0.721	0.283	0.666	0.801	0.726	0.766	0.778	0.721	0.169	0.000	0.801	0.698	0.766	0.778
jruby-1.4.0	0.729	0.275	0.664	0.719	0.739	0.674	0.729	0.729	0.000	0.325	0.719	0.739	0.672	0.729
jruby-1.5.0	0.745	0.383	0.687	0.782	0.730	0.689	0.756	0.745	0.351	0.128	0.782	0.730	0.130	0.756
jruby-1.7.0	0.691	0.627	0.639	0.770	0.679	0.681	0.731	0.691	0.627	0.253	0.770	0.197	0.681	0.731
lucene-2.3.0	0.625	0.325	0.788	0.688	0.776	0.810	0.802	0.625	0.490	0.783	0.688	0.740	0.810	0.802
lucene-2.9.0	0.643	0.658	0.532	0.698	0.688	0.684	0.709	0.642	0.658	0.214	0.698	0.686	0.684	0.709
lucene-3.0.0	0.667	0.673	0.717	0.717	0.723	0.731	0.754	0.665	0.673	0.717	0.717	0.723	0.731	0.754
lucene-3.1	0.658	0.603	0.585	0.694	0.663	0.592	0.671	0.658	0.603	0.371	0.694	0.663	0.389	0.671
wicket-1.3.0beta2	0.733	0.741	0.489	0.758	0.726	0.726	0.751	0.733	0.741	0.472	0.758	0.726	0.726	0.751
wicket-1.3.0beta1	0.716	0.786	0.315	0.781	0.234	0.712	0.766	0.716	0.786	0.666	0.781	0.712	0.746	0.771
wicket-1.5.3	0.700	0.300	0.595	0.734	0.708	0.645	0.712	0.700	0.185	0.401	0.734	0.708	0.645	0.712
Median	0.681	0.543	0.664	0.728	0.718	0.684	0.737	0.681	0.526	0.632	0.728	0.721	0.702	0.747

In terms of the effort-aware evaluation measures P_{opt} and *Recall@20%*, MUSDP_v1 with O achieves better P_{opt} performance with statistical significance compared to the other variants, while MUSDP_v1 with PO exhibits superior *Recall@20%* values. Even so, MUSDP_v1 with CPO still yield satisfactory results with respect to P_{opt} and *Recall@20%*. When considering MUSDP_v2, PO and CPO typically show better performance in comparison with the other variants

with statistical significance in terms of P_{opt} and *Recall@20%*. This indicates that only the use of process metrics and ownership metrics tends to produce higher results regarding the effort-aware performance measures. On this basis, the addition of code metrics is unlikely to result in a significant change to performance of MUSDP.

The above analysis shows that the effectiveness of MUSDP on defect data varies with the combination of software metrics and performance measures used. We conclude that employing all types of metrics for defect data does not necessarily improve prediction performance for certain measures. It is worth noting that some software metrics may have stronger correlations with defects than others. This highlights the importance of carefully selecting and combining the appropriate types of metrics based on the specific characteristics of the data and the nature of the software project.

TABLE 12 Comparison results of MUSDP against its variants with the NPSKESD test in terms of *AUC*

Dataset	MUSDP_v1							MUSDP_v2						
	C	P	O	CP	CO	PO	CPO	C	P	O	CP	CO	PO	CPO
activemq-5.0.0	0.700	0.801	0.774	0.793	0.779	0.186	0.194	0.700	0.801	0.774	0.793	0.779	0.814	0.809
activemq-5.1.0	0.700	0.607	0.687	0.744	0.737	0.688	0.744	0.700	0.603	0.687	0.744	0.737	0.688	0.744
activemq-5.2.0	0.677	0.600	0.764	0.750	0.782	0.782	0.806	0.677	0.539	0.764	0.750	0.782	0.782	0.806
activemq-5.3.0	0.715	0.500	0.750	0.739	0.746	0.688	0.758	0.715	0.500	0.750	0.739	0.746	0.688	0.758
activemq-5.8.0	0.731	0.748	0.701	0.777	0.751	0.765	0.771	0.731	0.748	0.701	0.777	0.751	0.765	0.771
derby-10.2.1.6	0.664	0.525	0.220	0.728	0.227	0.205	0.195	0.664	0.589	0.780	0.728	0.773	0.795	0.805
derby-10.3.1.4	0.656	0.570	0.668	0.691	0.726	0.668	0.736	0.656	0.567	0.332	0.691	0.726	0.332	0.736
derby-10.5.1.1	0.696	0.562	0.657	0.717	0.730	0.584	0.688	0.696	0.562	0.657	0.717	0.730	0.584	0.688
groovy-1_5_7	0.745	0.537	0.754	0.772	0.730	0.776	0.771	0.745	0.500	0.754	0.772	0.730	0.776	0.771
groovy-1_6beta1	0.688	0.526	0.719	0.748	0.734	0.763	0.783	0.688	0.500	0.719	0.748	0.732	0.763	0.783
groovy-1_6beta2	0.644	0.520	0.402	0.724	0.674	0.769	0.761	0.643	0.563	0.594	0.724	0.674	0.769	0.761
hbase-0.94.0	0.671	0.544	0.700	0.666	0.762	0.723	0.752	0.668	0.544	0.313	0.666	0.762	0.723	0.752
hbase-0.95.0	0.585	0.602	0.575	0.624	0.674	0.665	0.657	0.431	0.602	0.455	0.624	0.672	0.665	0.657
hbase-0.95.2	0.574	0.528	0.561	0.586	0.687	0.664	0.672	0.555	0.533	0.437	0.586	0.687	0.664	0.672
hive-0.10.0	0.695	0.509	0.757	0.747	0.775	0.773	0.787	0.695	0.500	0.757	0.747	0.775	0.773	0.787
hive-0.12.0	0.589	0.507	0.722	0.707	0.699	0.723	0.722	0.583	0.500	0.278	0.707	0.320	0.277	0.722
hive-0.9.0	0.593	0.562	0.655	0.700	0.692	0.634	0.675	0.591	0.552	0.655	0.700	0.692	0.634	0.675
jruby-1.1	0.729	0.512	0.718	0.801	0.751	0.767	0.781	0.729	0.500	0.282	0.801	0.716	0.767	0.781
jruby-1.4.0	0.731	0.500	0.676	0.727	0.746	0.676	0.729	0.731	0.500	0.347	0.727	0.745	0.674	0.729
jruby-1.5.0	0.754	0.522	0.733	0.782	0.748	0.733	0.771	0.754	0.500	0.276	0.782	0.747	0.277	0.771
jruby-1.7.0	0.701	0.654	0.675	0.770	0.716	0.692	0.743	0.701	0.654	0.325	0.770	0.292	0.692	0.743
lucene-2.3.0	0.626	0.507	0.802	0.693	0.790	0.816	0.810	0.626	0.569	0.798	0.693	0.747	0.816	0.809
lucene-2.9.0	0.646	0.662	0.620	0.699	0.694	0.684	0.711	0.644	0.662	0.380	0.699	0.691	0.684	0.711
lucene-3.0.0	0.676	0.676	0.735	0.717	0.736	0.733	0.760	0.671	0.676	0.734	0.717	0.736	0.733	0.760
lucene-3.1	0.664	0.629	0.603	0.694	0.664	0.600	0.672	0.664	0.629	0.397	0.694	0.664	0.404	0.672
wicket-1.3.0beta2	0.743	0.742	0.491	0.758	0.728	0.726	0.751	0.743	0.742	0.475	0.758	0.728	0.726	0.751
wicket-1.3.0beta1	0.727	0.786	0.329	0.782	0.276	0.712	0.768	0.727	0.786	0.671	0.782	0.726	0.749	0.772
wicket-1.5.3	0.716	0.500	0.604	0.738	0.715	0.650	0.714	0.716	0.500	0.412	0.738	0.715	0.650	0.714
Median	0.686	0.599	0.681	0.730	0.727	0.703	0.741	0.686	0.595	0.636	0.730	0.728	0.706	0.750

TABLE 13 Comparison results of MUSDP against its variants with the NPSKESD test in terms of P_{opt}

Dataset	MUSDP_v1							MUSDP_v2						
	C	P	O	CP	CO	PO	CPO	C	P	O	CP	CO	PO	CPO
activemq-5.0.0	0.534	0.727	0.704	0.682	0.674	0.344	0.397	0.534	0.727	0.704	0.682	0.674	0.753	0.721
activemq-5.1.0	0.530	0.572	0.590	0.594	0.604	0.625	0.631	0.530	0.576	0.590	0.594	0.604	0.625	0.631
activemq-5.2.0	0.543	0.622	0.731	0.627	0.699	0.774	0.732	0.550	0.606	0.731	0.627	0.699	0.774	0.732
activemq-5.3.0	0.571	0.521	0.672	0.603	0.642	0.638	0.651	0.571	0.501	0.672	0.603	0.642	0.638	0.651
activemq-5.8.0	0.572	0.602	0.631	0.622	0.613	0.639	0.619	0.572	0.602	0.631	0.622	0.613	0.639	0.619
derby-10.2.1.6	0.535	0.634	0.424	0.563	0.565	0.400	0.503	0.535	0.598	0.755	0.563	0.679	0.779	0.726
derby-10.3.1.4	0.515	0.475	0.707	0.486	0.614	0.707	0.609	0.515	0.466	0.477	0.486	0.614	0.477	0.609
derby-10.5.1.1	0.574	0.523	0.598	0.566	0.603	0.529	0.574	0.573	0.523	0.591	0.566	0.603	0.529	0.574
groovy-1_5_7	0.496	0.497	0.596	0.534	0.528	0.638	0.570	0.496	0.484	0.596	0.534	0.528	0.638	0.570
groovy-1_6beta1	0.573	0.577	0.729	0.587	0.687	0.764	0.739	0.573	0.434	0.729	0.587	0.672	0.764	0.739
groovy-1_6beta2	0.509	0.595	0.433	0.592	0.550	0.767	0.688	0.509	0.568	0.664	0.592	0.550	0.767	0.688
hbase-0.94.0	0.623	0.454	0.722	0.486	0.728	0.648	0.671	0.623	0.439	0.460	0.486	0.727	0.648	0.671
hbase-0.95.0	0.620	0.418	0.674	0.448	0.610	0.563	0.539	0.587	0.418	0.566	0.448	0.607	0.563	0.539
hbase-0.95.2	0.585	0.464	0.691	0.338	0.590	0.572	0.521	0.598	0.435	0.508	0.338	0.590	0.541	0.521
hive-0.10.0	0.655	0.582	0.804	0.659	0.803	0.804	0.785	0.655	0.484	0.804	0.659	0.803	0.804	0.785
hive-0.12.0	0.566	0.712	0.937	0.682	0.816	0.938	0.774	0.567	0.604	0.401	0.682	0.596	0.397	0.774
hive-0.9.0	0.487	0.473	0.594	0.553	0.626	0.539	0.579	0.486	0.429	0.594	0.553	0.626	0.539	0.579
jruby-1.1	0.562	0.588	0.727	0.595	0.668	0.697	0.617	0.562	0.500	0.432	0.595	0.571	0.650	0.617
jruby-1.4.0	0.627	0.592	0.655	0.541	0.708	0.624	0.642	0.627	0.564	0.561	0.541	0.702	0.615	0.642
jruby-1.5.0	0.584	0.574	0.690	0.572	0.611	0.697	0.632	0.584	0.546	0.467	0.572	0.606	0.481	0.632
jruby-1.7.0	0.555	0.519	0.640	0.592	0.650	0.619	0.632	0.555	0.519	0.421	0.592	0.452	0.619	0.632
lucene-2.3.0	0.499	0.570	0.867	0.580	0.832	0.870	0.821	0.499	0.551	0.857	0.580	0.744	0.870	0.820
lucene-2.9.0	0.548	0.599	0.732	0.616	0.654	0.649	0.668	0.548	0.599	0.447	0.616	0.648	0.649	0.668
lucene-3.0.0	0.627	0.626	0.770	0.657	0.736	0.707	0.738	0.609	0.626	0.770	0.657	0.736	0.707	0.738
lucene-3.1	0.532	0.544	0.607	0.559	0.543	0.594	0.559	0.532	0.544	0.385	0.559	0.543	0.404	0.559
wicket-1.3.0beta2	0.596	0.565	0.520	0.586	0.572	0.561	0.578	0.596	0.565	0.524	0.586	0.572	0.561	0.578
wicket-1.3.0beta1	0.556	0.623	0.466	0.607	0.519	0.558	0.593	0.556	0.623	0.612	0.607	0.589	0.606	0.608
wicket-1.5.3	0.567	0.578	0.570	0.593	0.571	0.609	0.584	0.567	0.591	0.511	0.593	0.571	0.609	0.584
Median	0.561	0.573	0.671	0.587	0.629	0.640	0.622	0.560	0.548	0.582	0.587	0.622	0.626	0.638

TABLE 14 Comparison results of MUSDP against its variants with the NPSKESD test in terms of *Recall@20%*

Dataset	MUSDP_v1							MUSDP_v2						
	C	P	O	CP	CO	PO	CPO	C	P	O	CP	CO	PO	CPO
activemq-5.0.0	0.214	0.434	0.369	0.362	0.343	0.121	0.123	0.214	0.434	0.369	0.362	0.343	0.472	0.409
activemq-5.1.0	0.235	0.306	0.281	0.301	0.305	0.354	0.336	0.235	0.306	0.281	0.301	0.305	0.354	0.336
activemq-5.2.0	0.232	0.291	0.442	0.329	0.406	0.578	0.472	0.231	0.289	0.442	0.329	0.406	0.578	0.472
activemq-5.3.0	0.296	0.237	0.389	0.318	0.359	0.398	0.381	0.296	0.197	0.389	0.318	0.359	0.398	0.381
activemq-5.8.0	0.260	0.309	0.303	0.309	0.301	0.359	0.302	0.260	0.309	0.303	0.309	0.301	0.359	0.302
derby-10.2.1.6	0.327	0.320	0.203	0.359	0.239	0.186	0.215	0.327	0.284	0.523	0.359	0.458	0.543	0.508
derby-10.3.1.4	0.314	0.239	0.462	0.308	0.395	0.461	0.395	0.314	0.215	0.173	0.308	0.395	0.175	0.395

derby-10.5.1.1	0.301	0.243	0.360	0.312	0.338	0.294	0.342	0.301	0.243	0.357	0.312	0.338	0.294	0.342
groovy-1_5_7	0.125	0.222	0.240	0.182	0.160	0.310	0.222	0.125	0.182	0.240	0.182	0.160	0.310	0.222
groovy-1_6beta1	0.349	0.405	0.540	0.406	0.455	0.565	0.535	0.349	0.180	0.540	0.406	0.440	0.565	0.535
groovy-1_6beta2	0.273	0.403	0.160	0.364	0.308	0.591	0.500	0.273	0.308	0.411	0.364	0.308	0.591	0.500
hbase-0.94.0	0.400	0.230	0.471	0.314	0.487	0.434	0.456	0.400	0.221	0.195	0.314	0.482	0.434	0.456
hbase-0.95.0	0.502	0.291	0.552	0.340	0.498	0.462	0.442	0.476	0.291	0.402	0.340	0.493	0.462	0.437
hbase-0.95.2	0.449	0.265	0.536	0.211	0.481	0.459	0.422	0.443	0.213	0.323	0.211	0.481	0.425	0.422
hive-0.10.0	0.425	0.344	0.580	0.459	0.550	0.593	0.562	0.425	0.271	0.580	0.459	0.550	0.593	0.562
hive-0.12.0	0.393	0.491	0.731	0.558	0.611	0.736	0.616	0.392	0.266	0.024	0.558	0.211	0.023	0.616
hive-0.9.0	0.310	0.284	0.385	0.358	0.374	0.367	0.373	0.305	0.242	0.385	0.358	0.374	0.367	0.373
jruby-1.1	0.217	0.294	0.355	0.278	0.308	0.352	0.278	0.217	0.264	0.200	0.278	0.269	0.292	0.278
jruby-1.4.0	0.376	0.366	0.417	0.333	0.440	0.395	0.407	0.375	0.297	0.299	0.333	0.430	0.386	0.407
jruby-1.5.0	0.232	0.276	0.347	0.274	0.268	0.353	0.286	0.232	0.215	0.140	0.274	0.259	0.143	0.286
jruby-1.7.0	0.250	0.282	0.300	0.333	0.313	0.365	0.344	0.250	0.282	0.127	0.333	0.176	0.365	0.344
lucene-2.3.0	0.290	0.329	0.609	0.360	0.566	0.622	0.561	0.290	0.295	0.590	0.360	0.490	0.622	0.559
lucene-2.9.0	0.323	0.418	0.439	0.415	0.409	0.458	0.433	0.323	0.418	0.068	0.415	0.401	0.458	0.433
lucene-3.0.0	0.385	0.476	0.543	0.472	0.509	0.533	0.533	0.380	0.476	0.541	0.472	0.509	0.533	0.533
lucene-3.1	0.293	0.323	0.326	0.310	0.316	0.326	0.307	0.293	0.323	0.177	0.310	0.316	0.218	0.307
wicket-1.3.0beta2	0.259	0.305	0.196	0.314	0.255	0.296	0.302	0.259	0.305	0.206	0.314	0.255	0.296	0.302
wicket-1.3.0beta1	0.212	0.324	0.110	0.304	0.129	0.212	0.272	0.212	0.324	0.300	0.304	0.270	0.300	0.296
wicket-1.5.3	0.216	0.253	0.238	0.274	0.235	0.290	0.266	0.216	0.266	0.174	0.274	0.235	0.290	0.266
Median	0.296	0.310	0.383	0.331	0.367	0.403	0.382	0.296	0.286	0.300	0.331	0.358	0.397	0.398

4.3 Running Time

To examine the efficiency of MUSDP and the competing methods, we record the time cost for each method on each dataset with 100 repetitions and across all 28 datasets. Our experiments were conducted on a 64-bit Ubuntu 22.04 computer system equipped with an Intel ® Xeon (R) CPU E3-1271 @ 3.60GHz and 32G RAM, utilizing Python for the computation. Since ManualDown and ManualUp are single metric methods, they execute very quickly, and therefore, we did not include them in the comparison.

Table 15 shows the mean (± deviation) running time across all 28 datasets, measured in seconds. From the table, we can observe that the running time of MUSDP is approximately 1.15 seconds, indicating high computational efficiency. The time cost of MUSDP is only slightly higher than that of unsurprised methods (CLA, CLAMI, Kmedoids, SC, and TCL), while significantly lower than supervised methods (DNN, CNN, GRU, and MVKNN). Hence, the execution of MUSDP does not consume much time. This analysis underscores the time efficiency of MUSDP for unsupervised defect prediction in practical applications.

TABLE 15 The average running time (mean ± deviation) for each method with 100 repetition across all 28 datasets

Method	CLA	CLAMI	SC	Kmedoid	TCL	TCLP	DNN	CNN	GRU	MVKNN	EASC	MUSDP _v1	MUSDP _v2
Time (second)	0.001±0.001	0.002±0.001	0.050±0.049	0.010±0.006	0.816±0.333	1.823±0.713	94.439±5.581	76.625±2.262	65.338±2.560	14.625±8.819	1.992±0.495	1.157±1.155	1.156±1.152

Reference

[1] N. Li, M. J. Shepperd, and G. Yuchen, "A systematic review of unsupervised learning techniques for software defect prediction," Information and Software Technology, vol. 122, p. 106287, 2020.

[2] 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, vol. 172, no. 3, p. 110862, 2021.

[3] P. Yang, L. Zhu, Y. Zhang, C. Ma, L. Liu, X. Yu, and W. Hu, "On the relative value of clustering techniques for unsupervised effort-aware defect prediction," Expert Systems with Applications, vol. 245, p. 123041, 2024.

[4] S. Zhong, T. M. Khoshgoftaar, and N. Seliya, "Unsupervised learning for expert-based software quality estimation," in HASE'04. IEEE, 2004, pp. 149–155.

[5] C. Catal, U. Sevim, and B. Diri, "Clustering and metrics thresholds based software fault prediction of unlabeled program modules," in ITNG'09. IEEE, 2009, pp. 199–204.

[6] J. Nam and S. Kim, "Clami: Defect prediction on unlabeled datasets," in ASE'15, 2015, pp. 1–12.

[7] J. Yang and H. Qian, "Defect prediction on unlabeled datasets by using unsupervised clustering," in 2016 IEEE HPCC/SmartCity/DSS. IEEE, 2017, pp. 465–472.

[8] Y. Yang, J. Yang, and H. Qian, "Defect prediction by using cluster ensembles," in ICACI'18. IEEE, 2018, pp. 631–636.

[9] R. Kumar, A. Chaturvedi, and L. Kailasam, "An unsupervised software fault prediction approach using threshold derivation," IEEE Transactions on Reliability, vol. 71, no. 2, pp. 911–932, 2022.

[10] F. Zhang, Q. Zheng, Y. Zou, and A. E. Hassan, "Cross-project defect prediction using a connectivity-based unsupervised classifier," in ICSE'16, 2016, pp. 309–320.

[11] A. Marjuni, T. B. Adji, and R. Ferdiana, "Unsupervised software defect prediction using median absolute deviation threshold based spectral classifier on signed laplacian matrix," Journal of Big Data, vol. 6, no. 1, pp. 1–20, 2019.

[12] B. Yang, Q. Yin, S. Xu, and P. Guo, "Software quality prediction using affinity propagation algorithm," in IJCNN'08. IEEE, 2008, pp. 1891–1896.

[13] P. S. Bishnu and V. Bhattacherjee, "Software fault prediction using quad tree-based k-means clustering algorithm," IEEE Transactions on Knowledge and Data Engineering, vol. 24, no. 6, pp. 1146–1150, 2012.

[14] Y. Yang, Y. Zhou, J. Liu, Y. Zhao, H. Lu, L. Xu, B. Xu, and H. Leung, "Effort-aware just-in-time defect prediction: Simple unsupervised models could be better than supervised models," in FSE'16, 2016, pp. 157–168.

[15] J. Liu, Y. Zhou, Y. Yang, H. Lu, and B. Xu, "Code churn: A neglected metric in effort-aware just-in-time defect prediction," in ESEM'17. IEEE, 2017, pp. 11–19.

[16] Y. Zhou, Y. Yang, H. Lu, L. Chen, Y. Li, Y. Zhao, J. Qian, and B. Xu, "How far we have progressed in the journey? an examination of crossproject defect prediction," ACM Transactions on Software Engineering and Methodology, vol. 27, no. 1, pp. 1–51, 2018.

[17] S. Yatish, J. Jiarpakdee, P. Thongtanunam, and C. Tantithamthavorn, "Mining software defects: Should we consider affected releases?" in ICSE'19, 2019, pp. 654–665.

[18] J. Jiarpakdee, C. Tantithamthavorn, and A. Hassan, "The impact of correlated metrics on the interpretation of defect models," IEEE Transactions on Software Engineering, vol. 47, no. 2, pp. 320–331, 2021.

[19] J. Jiarpakdee, C. Tantithamthavorn, H. K. Dam, and J. Grundy, "An empirical study of model-agnostic techniques for defect prediction models," IEEE Transactions on Software Engineering, vol. 48, no. 1, pp. 166–185, 2022.

[20] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, "The impact of automated parameter optimization on defect prediction models," IEEE Transactions on Software Engineering, vol. 45, no. 7, pp. 683–711, 2019.

[21] 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, vol. 46, no. 11, pp. 1200–1219, 2020.

[22] J. Jiarpakdee, C. Tantithamthavorn, and C. Treude, "Autospearman: Automatically mitigating correlated software metrics for interpreting defect models," in ICSME'18. IEEE, 2018, pp. 92–103.

[23] T. Menzies, J. Greenwald, and A. Frank, "Data mining static code attributes to learn defect predictors," IEEE Transactions on Software

Engineering, vol. 33, no. 1, pp. 2–13, 2007.

[24] L. Madeyski and M. Jureczko, “Which process metrics can significantly improve defect prediction models? an empirical study,” Software Quality Journal, vol. 23, pp. 393–422, 2015.

[25] Q. Yu, S. Jiang, J. Qian, L. Bo, L. Jiang, and G. Zhang, “Process metrics for software defect prediction in object-oriented programs,” IET Software, vol. 14, no. 3, pp. 283–292, 2020.