IMA Project 2

Nouran Abdalazim

May 27, 2022

0 Code Repository

The code and result files, part of this submission, can be found at

Repo: https://github.com/usi-ima/2022-project-2-nouran-abdalazim.git
Commit: 0d8620825ea375902b848d5fd930cb59be54beb9

1 Data Pre-Processing

I created my github repository using the provided link. I cloned the repository then I used the java classes in the folder (./resources/defects4j-checkout-closure-1f/src/com/google/javascript/jscomp) to extract my feature vectors.
The prepossessing was done in 2 steps. First, the java classes in the previous link were explored and the features were extracted. The feature vectors extracted can be found in .csv file (/2022-project-2-nouran-abdalazim/feature_vectors.csv).
Second, the extracted feature vectors were labelled using the ground truth provided in the folder (./resources/modified_classes).
The resulting csv of extracted, labelled feature vectors can be found in the repository at the following path (/2022-project-2-nouran-abdalazim/new_feature_vectors.csv).

1.1 Feature Vector Extraction

I extracted 281 feature vectors. 12 features from 3 different categories were extracted. From class metrics category 3 features were extracted. From method metrics category 3 features were extracted. Finally, from NLP metrics category 3 features were extracted.
Table 1 show details about the statistics for each extracted feature (mean, standard deviation, minimum and maximum).

1.2 Feature Vector Labelling

After labelling the extracted feature vectors based on the given ground truth, it was found that 76 feature vectors were labelled as buggy (27.1%) and 205 feature vectors were labelled as not buggy (72.9%). This means that the dataset is not balanced.

Table 1: Extracted Features Statistics

Feature Category	Feature	Statistics			
		Mean	Standard Deviation	Minimum	Maximum
Class Metrics	MTH	12.01	20.79	0	209
	FLD	6.76	13.46	0	167
	RFC	107.53	144.58	0	873
	INT	0.67	0.68	0	3
Method Metrics	SZ	26.06	44.90	0	463
	CPX	5.85	8.40	0	96
	EX	0.40	1.02	0	9
	RET	3.77	7.40	0	86
NLP Metrics	BCM	13.83	22.06	1	221
	NML	13.34	4.14	0	25.30
	WRD	324.67	428.98	2	3133
	DCM	16.09	49.26	0.04	475

Kindly note that the next steps of Hypertuning and Evaluation were conducted with the previous, wrong values of NML and DCM features, which were submitted for report feedback. The values of these 2 features were fixed after receiving the feedback and the github repository was updated as well.

2 Classifiers

In order to classify the extracted labelled feature vectors, 5 classifiers were adopted namely: Decision Tree (DT), Naive Byes (NB), Support Vector Machine (SVP), Multi-Layer Perceptron (MLP) and Random Forest (RF). Each classifier passed by a hyperparameter tuning process, where different parameter configurations were assessed in terms of recall, precision and f-measure.
For each classifier, 20 different parameter configurations were generated using *RandomizedSearchCV* python library.
In table 2, the different parameters configurations for each classifier are illsutrated.

Table 2: Hyperparameter Tuning Configurations

Classifier	**Parameters**	**Range**
Decision Tree	max_depth	randint(1, 10)
	max_features	randint(1, 12)
	min_samples_leaf	randint(1, 20)
	criterion	["gini", "entropy"]
Naive Bayes	var_smoothing	logspace(0, -9, num=100)
Support Vector Machine	tol	randint(1,20)
	C	randint(1,20)
Multi-Layer Perceptron	hidden_layer_sizes	[(50,50,50),(100,50,50), (50,100,50), (100,)]
	solver	["sgd", "adam"]
	alpha	[0.0001,0.0002,0.0003,0.0004,0.0005,0.05]
	learning_rate	["constant","adaptive"]
	learning_rate_init	[0.001,0.002,0.003,0.004,0.005,0.006,0.007,0.008,0.009]
	max_iter	[50,100,150,200,250,300,350]
Random Forest	max_depth	randint(1, 10)
	max_features	randint(1, 12)
	min_samples_leaf	randint(1, 20)
	criterion	["gini", "entropy"]
	n_estimators	randint(50,200)

2.1 Decision Tree (DT)

For the Decision Tree (DT) classifier, 4 parameters were assessed based on recall, precision and F1. Details about the 4 parameters can be found in Table 2. The driving factor behind the choice of the these parameters was to try to avoid the overfitting problem. For instance increasing the tree depth, number of features taken into consideration for splitting the tree and the minimum data samples in tree leaves can prevent the model generalization and leads to overfitting to the training dataset. Each parameters configuration was assessed in term of recall, precision and F1.
The results for each parameter configuration are illustrated in Table 3. Parameters configurations are sorted based on their F1 value descending. The highlighted parameter configuration was adopted in the further experiments.

Table 3: Decision Tree Hyperparameter Tuning Results

No.	max_depth	max_features	min_samples_leaf	criterion	recall	precision	f1
1	**9**	**4**	**5**	**entropy**	**0.58**	**0.54**	**0.56**
2	2	6	2	entropy	0.58	0.54	0.56
3	5	1	6	gini	0.50	0.54	0.52
4	9	2	16	gini	0.50	0.54	0.52
5	3	3	10	gini	0.50	0.54	0.52
6	3	3	2	entropy	0.63	0.38	0.48
7	4	11	15	gini	0.39	0.54	0.45
8	8	6	15	entropy	0.40	0.46	0.43
9	7	4	17	entropy	0.50	0.31	0.38
10	2	2	18	gini	0.50	0.31	0.38
11	9	1	7	entropy	0.50	0.31	0.38
12	9	5	12	gini	0.36	0.38	0.37
13	5	10	9	gini	0.36	0.31	0.33
14	2	4	10	gini	0.33	0.31	0.32
15	4	1	7	entropy	0.50	0.23	0.32
16	5	2	15	entropy	0.43	0.23	0.3
17	9	9	17	entropy	0.33	0.23	0.27
18	1	2	9	gini	0	0	0
19	1	3	7	entropy	0	0	0
20	1	5	17	entropy	0	0	0

2.2 Naive Bayes (NB)

For the Naive Bayes (NB)) classifier, 1 parameter was assessed based on recall, precision and F1. Details about the parameter can be found in Table 2.
The driving factor behind choosing the *var_smoothing* parameter was to try to avoid the overfitting problem. Since *var_smoothing* adds a user-defined value to the distribution's variance (whose default value is derived from the training data set). This essentially widens the curve and provides a chance for more samples that are further away from the distribution mean.
In table 4, the results for each parameter configuration are illustrated, configuration are sorted based on their F1 value descending. The highlighted parameter configuration was used in the next experiments.

Table 4: Naive Bayes Hyperparameter Tuning Results

No.	var_smoothing	recall	precision	f1
1	**3.51e-9**	**0.64**	**0.53**	**0.58**
2	3.51e-7	0.64	0.53	0.58
3	2.85e-7	0.64	0.53	0.58
4	3.51e-8	0.64	0.53	0.58
5	1.52e-8	0.64	0.53	0.58
6	0.01	0.73	0.47	0.57
7	0.01	0.73	0.47	0.57
8	0.00	0.67	0.47	0.55
9	0.00	0.67	0.47	0.55
10	0.00	0.67	0.47	0.55
11	1.00e-5	0.62	0.47	0.53
12	3.51e-5	0.62	0.47	0.53
13	8.11e-7	0.62	0.47	0.53
14	4.33e-5	0.62	0.47	0.53
15	5.34e-5	0.62	0.47	0.53
16	0.01	0.7	0.41	0.52
17	0.03	0.67	0.35	0.46
18	0.08	0.625	0.29	0.40
19	0.43	0.6	0.18	0.27
20	0.66	0.5	0.12	0.19

2.3 Support Vector Machine (SVP)

For the Support Vector Machine (SVP) classifier, 2 parameters were assessed based on recall, precision and F1. Details about the parameters can be found in Table 2.
The driving factor behind the choice of the 2 parameters was to avoid the overfitting problem, the first parameter is C parameter, it is the penalty parameter which controls the trade-off between minimizing the training error and maximizing the classification margin. In other words, C is degree of correct classification that the algorithm has to meet. The second parameter is *tol* parameter, it is the tolerance for stopping criterion. The parameters configurations were assessed in term of recall, precision and F1.
In table 5, the results for each parameter configuration is illustrated, parameters are sorted based on their F1 value descending. The highlighted parameter configuration was used in the next experiments.

Table 5: Support Vector Machine Hyperparameter Tuning Results

No.	tol	C	recall	precision	f1
1	**1.20e+1**	**12.00**	**0.41**	**0.54**	**0.47**
2	6.00e+0	11.00	0.75	0.23	0.35
3	1.10e+1	8.00	0.33	0.31	0.32
4	1.40e+1	12.00	0.24	0.46	0.32
5	1.50e+1	1.00	1.00	0.15	0.27
6	5.00	1.00	0.67	0.15	0.25
7	2.00	8.00	0.50	0.15	0.24
8	19.00	2.00	0.50	0.08	0.13
9	4.00	1.00	0.50	0.08	0.13
10	6.00	17.00	0.50	0.08	0.13
11	6.00e+0	16.00	0.50	0.08	0.13
12	1.70e+1	11.00	0.50	0.08	0.13
13	1.00e+0	5.00	0.00	0.00	0.00
14	1.80e+1	12.00	0.00	0.00	0.00
15	3.00e+0	11.00	0.00	0.00	0.00
16	12.00	11	0.00	0.00	0.00
17	8.00	13.00	0.00	0.00	0.00
18	8.00	16	0.00	0.00	0
19	1.00	2	0.00	0.00	0
20	15.00	19	0.00	0.00	0

2.4 Multi-Layer Perceptron (MLP)

For the Multi-Layer Perceptron (MLP) classifier, 6 parameters configurations were assessed based on recall, precision and F1. Details about the parameter can be found in Table 2.
The driving factor behind choosing these parameters was to try to avoid the overfitting problem. Since the number of hidden layers, learning rate, initial learning rate and maximum number of iterations have a influence on the classifier overfitting over the training dataset.
In table 6, the results for each parameter configuration are illustrated, configuration are sorted based on their F1 value descending. The highlighted parameter configuration was used in the next experiments.

Table 6: Multi-Layer Perceptron Hyperparameter Tuning Results

No.	hidden_layer_sizes	solver	alpha	learning_rate	learning_rate_init	max_iter	recall	precision	f1
1	**(50, 50, 50)**	**sgd**	**0.00**	**adaptive**	**0.01**	**250.00**	**0.63**	**0.8**	**0.71**
2	(50, 50, 50)	sgd	0.00	adaptive	0.01	200.00	0.52	0.8	0.63
3	(100,)	sgd	0.00	adaptive	0.01	300.00	0.55	0.73	0.63
4	(50, 100, 50)	sgd	0.00	adaptive	0.01	200.00	0.50	0.8	0.62
5	(100,)	sgd	0.00	constant	0.01	300.00	0.73	0.53	0.62
6	(100, 50, 50)	sgd	0.00	constant	0.01	350.00	0.48	0.8	0.6
7	(100,)	sgd	0.00	adaptive	0.01	300.00	0.56	0.6	0.58
8	(50, 50, 50)	sgd	0.05	adaptive	0.01	250.00	0.44	0.8	0.57
9	(50, 100, 50)	adam	0.00	constant	0.00	50.00	0.40	0.93	0.56
10	(100,)	adam	0.00	constant	0.00	50.00	0.48	0.67	0.56
11	(100, 50, 50)	adam	0.00	constant	0.01	250.00	0.35	0.93	0.51
12	(100,)	adam	0.00	adaptive	0.01	200.00	0.36	0.8	0.5
13	(50, 50, 50)	adam	0.00	constant	0.01	150.00	0.34	0.93	0.5
14	(50, 100, 50)	adam	0.00	constant	0.01	250.00	0.34	0.87	0.49
15	(100,)	adam	0.00	constant	0.00	350.00	0.35	0.8	0.49
16	(50, 50, 50)	sgd	0.00	constant	0.01	100.00	0.83	0.33	0.48
17	(100,)	adam	0.00	constant	0.01	250.00	0.32	0.8	0.46
18	(50, 50, 50)	sgd	0.00	constant	0.004	250	0.71	0.33	0.45
19	(50, 100, 50)	sgd	0.00	constant	0.004	150	1	0.13	0.24
20	(100, 50, 50)	sgd	0.00	adaptive	0.002	350	0.67	0.13	0.22

2.5 Random Forest (RF)

For the Random Forest (RF) classifier, the same 4 parameters previously used for the Decision Tree classifier were adopted in addition too the number of trees. In total 5 parameters were assessed based on recall, precision and F1. Details about the 5 parameters can be found in Table 2.
The driving factor behind the choice of the these parameters was to try to avoid the overfitting problem. Each parameters configuration was assessed in term of recall, precision and F1. For instance increasing of estimators (trees) can hinder the model generalization and leads to model overfitting to the training dataset. The results for each parameter configuration are illustrated in Table 7. Parameters configurations are sorted based on their F1 value descending. The highlighted parameter configuration was adopted in the further experiments.

Table 7: Random Forest Hyperparameter Tuning Results

No.	max_depth	max_features	min_samples_leaf	criterion	n_estimators	recall	precision	f1
1	**9.00e+0**	**4.00**	**14.00**	**gini**	**118.00**	**0.64**	**0.60**	**0.62**
2	2.00e+0	10.00	12.00	gini	152.00	0.56	0.67	0.61
3	6.00e+0	8.00	6.00	gini	77.00	0.60	0.60	0.60
4	9.00e+0	4.00	18.00	entropy	173.00	0.67	0.53	0.59
5	3.00e+0	5.00	9.00	entropy	59.00	0.67	0.53	0.59
6	5.00	3.00	14.00	entropy	84.00	0.67	0.53	0.59
7	8.00	3.00	13.00	gini	197.00	0.67	0.53	0.59
8	1.00	7.00	19.00	gini	150.00	0.67	0.53	0.59
9	5.00	5.00	17.00	gini	182.00	0.67	0.53	0.59
10	4.00	6.00	6.00	entropy	112.00	0.67	0.53	0.59
11	5.00e+0	7.00	19.00	gini	124.00	0.56	0.60	0.58
12	9.00e+0	11.00	13.00	entropy	66.00	0.62	0.53	0.57
13	8.00e+0	6.00	3.00	gini	146.00	0.62	0.53	0.57
14	2.00e+0	7.00	15.00	gini	126.00	0.62	0.53	0.57
15	8.00e+0	11.00	13.00	gini	67.00	0.57	0.53	0.55
16	4.00	2	4.00	gini	173.00	0.64	0.47	0.54
17	8.00	11.00	11.00	entropy	55.00	0.53	0.53	0.53
18	2.00	9	18.00	entropy	129	0.5	0.53	0.52
19	8.00	8	15.00	gini	52	0.5	0.53	0.52
20	2.00	2	6.00	entropy	108	0.6	0.40	0.48

3 Evaluation

After the hyperparameter tuning phase, the optimal parameters for each classifier were picked up for further 5 fold cross validation experiments. 20 cross validation iterations were conducted resulting 100 evaluation observation of each classifier per each evaluation metric (recall, precision and F1).

3.1 Output Distributions

The output distribution of the 5 previously mentioned classifiers along with 2 biased classifier was explored.
The first biased classifier is the frequent biased classifier (BCF) that returns the most representative class label in the dataset. The second biased classifier is the constant biased classifier (BCC) that returns a constant value regardless the input feature vector. The driving factor behind using 2 biased classifiers was that the dataset was unbalanced as mention in Section 1.2 where most of the feature vectors were labeled as not buggy (class label : 0). Given this, the results of recall and precision of BCF was always zero because there were no true positives (TP) predictions. Accordingly BCC was adopted in the experimental evaluation as well. BCC with constant class label value of 1 (buggy) regardless of the input.
In Figures 1, 2 and 3, the distribution of recall, precision and F1 for the 7 classifiers are visualized as a boxplot diagram. Moreover, in Table 8, statistical details for each classifier along with each evaluation metric are presented.

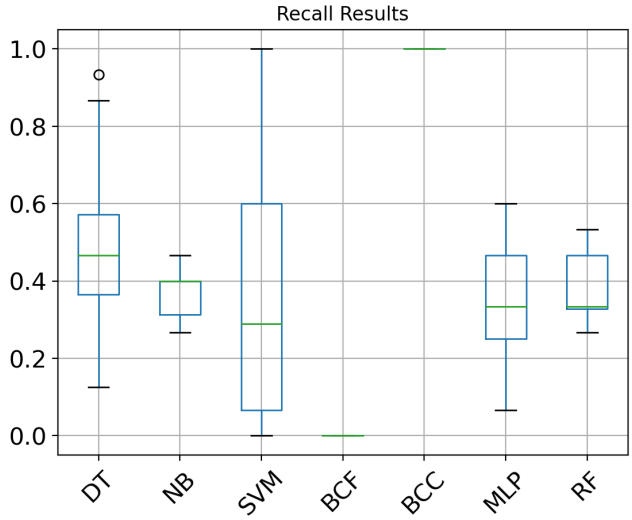

Figure 1: Recall Results Distribution

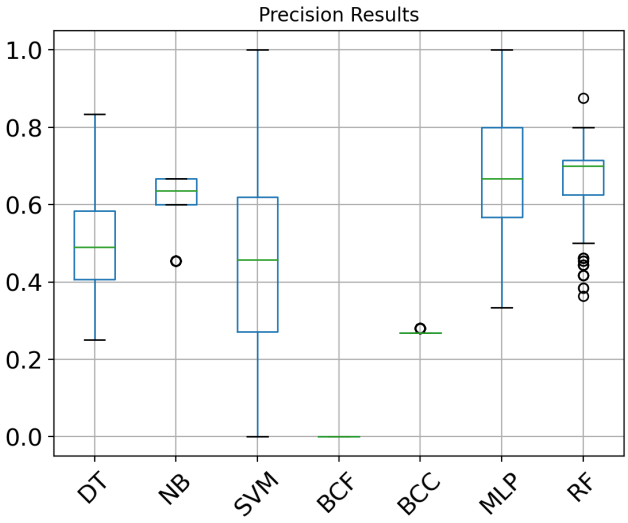

Figure 2: Precision Results Distribution

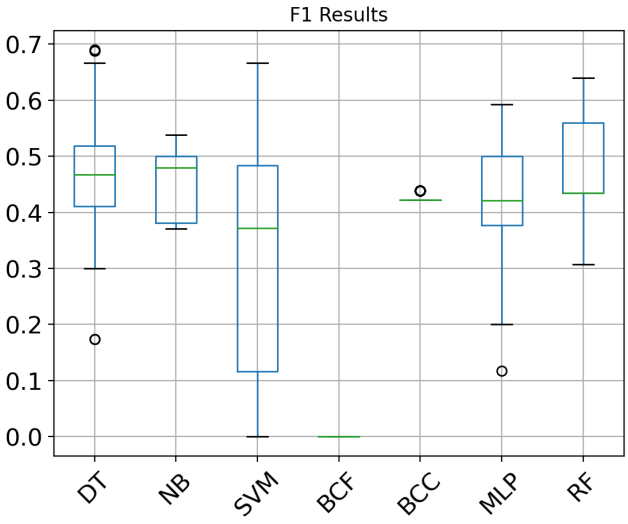

Figure 3: F1 Results Distribution

3.2 Comparison and Significance

After the visualization of the distribution of the each evaluation metric, every combination of 2 classifiers is compared for every evaluation metric used. Wilcoxon test was used for this analysis, it is the non parametric version of t-test. It tests the null hypothesis (H0) that two populations have the same continuous

Table 8: Evaluation Metrics Output Statistics

Classifier	Evaluation Metric	Statistics			
		Mean	Standard Deviation	Minimum	Maximum
Decision Tree	Recall	0.48	0.17	0.125	0.93
	Precision	0.5	0.13	0.25	0.93
	F1	0.47	0.1	0.17	0.69
Naive Bayes	Recall	0.37	0.07	0.27	0.47
	Precision	0.61	0.08	0.45	0.66
	F1	0.45	0.07	0.37	0.54
Support Vector Machine	Recall	0.36	0.32	0	1
	Precision	0.43	0.28	0	1
	F1	0.33	0.22	0	0.67
Multi-Layer Perceptron	Recall	0.34	0.11	0.07	0.6
	Precision	0.68	0.20	0.33	1
	F1	0.42	0.09	0.12	0.59
Random Forest	Recall	0.38	0.07	0.27	0.53
	Precision	0.65	0.12	0.36	0.88
	F1	0.48	0.08	0.31	0.64
Biased Classifier (Frequent)	Recall	0	0	0	0
	Precision	0	0	0	0
	F1	0	0	0	0
Biased Classifier (Constant)	Recall	1	0	1	1
	Precision	0.27	0.01	0.27	0.28
	F1	0.43	0.01	0.42	0.44

distribution. Accordingly, Wilcoxon test analyze the following hypothesis:

1. H0: The median difference is zero versus
2. H1: The median difference is positive α=0.05

Based on the P-value computed, the test can decide to accept or reject which hypothesis.

1. In case of P-value $< \alpha$ (0.05) $\Rightarrow$ Reject the null hypothesis (H0)
2. In case of P-value $> \alpha$ (0.05) $\Rightarrow$ Accept the null hypothesis (H0)

P-value results of Wilcoxon test are each pair of classifier and for each evaluation metric illustrated in Table 9.

Table 9: P-value Results from Wilcoxon Text

Classifier 1	Classifier 2	P-value (Recall)	P-value (Precision)	P-value (F1)
Decision Tree	Naive Bayes	0	0	0.4012
	Support Vector Machine	0.002	0.0458	0
	Biased Classifier (Frequent)	0	0	0
	Biased Classifier (Constant)	0	0	0
	Multli-Layer Perceptron	0	0	0.0015
	Random Forest	0	0	0.527
Naive Bayes	Decision Tree	0	0	0.4012
	Support Vector Machine	0.5606	0	0
	Biased Classifier (Frequent)	0	0	0
	Biased Classifier (Constant)	0	0	0
	Multli-Layer Perceptron	0.1923	0.0009	0.0062
	Random Forest	0.4435	0.0028	0.1383
Support Vector Machine	Decision Tree	0.002	0.0458	0
	Naive Bayes	0.5606	0	0
	Biased Classifier (Frequent)	0	0	0
	Biased Classifier (Constant)	0	0	0.0002
	Multli-Layer Perceptron	0.6654	0	0.0006
	Random Forest	0.4724	0	0
Biased Classifier (Frequent)	Decision Tree	0	0	0
	Naive Bayes	0	0	0
	Support Vector Machine	0	0	0
	Biased Classifier (Constant)	0	0	0
	Multli-Layer Perceptron	0	0	0
	Random Forest	0	0	0
Biased Classifier (Constant)	Decision Tree	0	0	0
	Naive Bayes	0	0	0
	Support Vector Machine	0	0	0.0002
	Biased Classifier (Frequent)	0	0	0
	Multli-Layer Perceptron	0	0	0.6399
	Random Forest	0	0	0
Multli-Layer Perceptron	Decision Tree	0	0	0.0015
	Naive Bayes	0.1923	0.0009	0.0062
	Support Vector Machine	0.6654	0	0.0006
	Biased Classifier (Frequent)	0	0	0
	Biased Classifier (Constant)	0	0	0.6399
	Random Forest	0.0052	0.822	0
Random Forest	Decision Tree	0	0	0.527
	Naive Bayes	0.4435	0.0028	0.1383
	Support Vector Machine	0.4724	0	0
	Biased Classifier (Frequent)	0	0	0
	Biased Classifier (Constant)	0	0	0
	Multli-Layer Perceptron	0.0052	0.822	0

3.2.1 F1 Values

Using Tables 8 and 9, the conclusions about the mean of F1 values presented in Table 10 can be derived.

Table 10: Comparison Between Each Pair of Classifiers based in F1 Mean Using Wilcoxon Test

Classifier 1	F1	Classifier 2	F1	P-value	Decision for H0	Conclusion
DT	0.47	NB	0.45	0.4012	Accept H0	Both classifiers have the same performance
		SVP	0.33	0	Reject H0	DT is better than SVP
		BCF	0	0	Reject H0	DT is better than BCF
		BCC	0.43	0	Reject H0	DT is better than BCC
		MLP	0.42	0.0015	Reject H0	DT is better than MLP
		RF	0.48	0.527	Accept H0	Both classifiers have the same performance
NB	0.45	DT	0.47	0.4012	Accept H0	Both classifiers have the same performance
		SVP	0.33	0	Reject H0	NB is better than SVP
		BCF	0	0	Reject H0	NB is better than BCF
		BCC	0.43	0	Reject H0	NB is better than BCC
		MLP	0.42	0.0062	Reject H0	NB is better than MLP
		RF	0.48	0.1383	Accept H0	Both classifiers have the same performance
SVP	0.33	DT	0.47	0	Reject H0	DT is better than SVP
		NB	0.45	0	Reject H0	NB is better than SVP
		BCF	0	0	Reject H0	SVP is better than BCF
		BCC	0.43	0.0002	Reject H0	BCC is better than SVP
		MLP	0.42	0.0006	Reject H0	MLP is better than SVP
		RF	0.48	0	Reject H0	RF is better than SVP
BCF	0	DT	0.47	0	Reject H0	DT is better than BCF
		NB	0.45	0	Reject H0	NB is better than BCF
		SVP	0.33	0	Reject H0	SVP is better than BCF
		BCC	0.43	0	Reject H0	BCC is better than BCF
		MLP	0.42	0	Reject H0	MLP is better than BCF
		RF	0.48	0	Reject H0	RF is better than BCF
BCC	0.43	DT	0.47	0	Reject H0	DT is better than BCC
		NB	0.45	0	Reject H0	NB is better than BCC
		SVP	0.33	0.0002	Reject H0	BCC is better than SVP
		BCF	0	0	Reject H0	BCC is better than BCF
		MLP	0.42	0.6399	Accept H0	Both classifiers have the same performance
		RF	0.48	0	Reject H0	RF is better than BCC
MLP	0.42	DT	0.47	0.0015	Reject H0	DT is better than MLP
		NB	0.45	0.0062	Reject H0	NB is better than MLP
		SVP	0.33	0.0006	Reject H0	MLP is better than SVP
		BCF	0	0	Reject H0	MLP is better than BCF
		BCC	0.43	0.6399	Accept H0	Both classifiers have the same performance
		RF	0.48	0	Reject H0	RF is better than MLP
RF	0.48	DT	0.47	0.527	Accept H0	Both classifiers have the same performance
		NB	0.45	0.1383	Accept H0	Both classifiers have the same performance
		SVP	0.33	0	Reject H0	RF is better than SVP
		BCF	0	0	Reject H0	RF is better than BCF
		BCC	0.43	0	Reject H0	RF is better than BCC
		MLP	0.42	0	Reject H0	RF is better than MLP

3.2.2 Precision

Using Tables 8 and 9, the conclusions about the mean of precision values presented in Table 11 can be derived.

Table 11: Comparison Between Each Pair of Classifiers based in Precision Mean Using Wilcoxon Test

Classifier 1	Precision	Classifier 2	Precision	P-value	Decision for H0	Conclusion
DT	0.5	NB	0.61	0	Reject H0	NB is better than DT
		SVP	0.43	0.0458	Reject H0	DT is better than SVP
		BCF	0	0	Reject H0	DT is better than BCF
		BCC	0.27	0	Reject H0	DT is better than BCC
		MLP	0.68	0	Reject H0	MLP is better than DT
		RF	0.65	0	Reject H0	RF is better than DT
NB	0.61	DT	0.5	0	Reject H0	NB is better than DT
		SVP	0.43	0	Reject H0	NB is better than SVP
		BCF	0	0	Reject H0	NB is better than BCF
		BCC	0.27	0	Reject H0	NB is better than BCC
		MLP	0.68	0.0009	Reject H0	MLP is better than NB
		RF	0.65	0.0028	Reject H0	RF is better than NB
SVP	0.43	DT	0.5	0.0458	Reject H0	DT is better than SVP
		NB	0.61	0	Reject H0	NB is better than SVP
		BCF	0	0	Reject H0	SVP is better than BCF
		BCC	0.27	0	Reject H0	SVP is better than BCC
		MLP	0.68	0	Reject H0	MLP is better than SVP
		RF	0.65	0	Reject H0	RF is better than SVP
BCF	0	DT	0.5	0	Reject H0	DT is better than BCF
		NB	0.61	0	Reject H0	NB is better than BCF
		SVP	0.43	0	Reject H0	SVP is better than BCF
		BCC	0.27	0	Reject H0	BCC is better than BCF
		MLP	0.68	0	Reject H0	MLP is better than BCF
		RF	0.65	0	Reject H0	RF is better than BCF
BCC	0.27	DT	0.5	0	Reject H0	DT is better than BCC
		NB	0.61	0	Reject H0	NB is better than BCC
		SVP	0.43	0	Reject H0	SVP is better than BCC
		BCF	0	0	Reject H0	BCC is better than BCF
		MLP	0.68	0	Reject H0	MLP is better than BCC
		RF	0.65	0	Reject H0	RF is better than BCC
MLP	0.68	DT	0.5	0	Reject H0	MLP is better than DT
		NB	0.61	0.0009	Reject H0	MLP is better than NB
		SVP	0.43	0	Reject H0	MLP is better than SVP
		BCF	0	0	Reject H0	MLP is better than BCF
		BCC	0.27	0	Reject H0	MLP is better than BCC
		RF	0.65	0.822	Accept H0	Both classifiers have the same performance
RF	0.65	DT	0.5	0	Reject H0	RF is better than DT
		NB	0.61	0.0028	Reject H0	RF is better than NB
		SVP	0.43	0	Reject H0	RF is better than SVP
		BCF	0	0	Reject H0	RF is better than BCF
		BCC	0.27	0	Reject H0	RF is better than BCC
		MLP	0.68	0.822	Accept H0	Both classifiers have the same performance

3.2.3 Recall

Using Tables 8 and 9, the conclusions about the mean of recall values presented in Table 12 can be derived.

Table 12: Comparison Between Each Pair of Classifiers based in Recall Mean Using Wilcoxon Test

Classifier 1	Recall	Classifier 2	Recall	P-value	Decision for H0	Conclusion
DT	0.48	NB	0.37	0	Reject H0	DT is better than NB
		SVP	0.36	0.002	Reject H0	DT is better than SVP
		BCF	0	0	Reject H0	DT is better than BCF
		BCC	1	0	Reject H0	BCC is better than DT
		MLP	0.34	0	Reject H0	DT is better than MLP
		RF	0.38	0	Reject H0	DT is better than RF
NB	0.37	DT	0.48	0	Reject H0	DT is better than NB
		SVP	0.36	0.5606	Accept H0	Both classifiers have the same performance
		BCF	0	0	Reject H0	NB is better than BCF
		BCC	1	0	Reject H0	BCC is better than NB
		MLP	0.34	0.1923	Accept H0	Both classifiers have the same performance
		RF	0.38	0.4435	Accept H0	Both classifiers have the same performance
SVP	0.36	DT	0.48	0.002	Reject H0	DT is better than SVP
		NB	0.37	0.5606	Accept H0	Both classifiers have the same performance
		BCF	0	0	Reject H0	SVP is better than BCF
		BCC	1	0	Reject H0	BCC is better than SVP
		MLP	0.34	0.6654	Accept H0	Both classifiers have the same performance
		RF	0.38	0.4724	Accept H0	Both classifiers have the same performance
BCF	0	DT	0.48	0	Reject H0	DT is better than BCF
		NB	0.37	0	Reject H0	NB is better than BCF
		SVP	0.36	0	Reject H0	SVP is better than BCF
		BCC	1	0	Reject H0	BCC is better than BCF
		MLP	0.34	0	Reject H0	MLP is better than BCF
		RF	0.38	0	Reject H0	RF is better than BCF
BCC	1	DT	0.48	0	Reject H0	BCC is better than DT
		NB	0.37	0	Reject H0	BCC is better than NB
		SVP	0.36	0	Reject H0	BCC is better than SVP
		BCF	0	0	Reject H0	BCC is better than BCF
		MLP	0.34	0	Reject H0	BCC is better than MLP
		RF	0.38	0	Reject H0	BCC is better than RF
MLP	0.34	DT	0.48	0	Reject H0	DT is better than MLP
		NB	0.37	0.1923	Accept H0	Both classifiers have the same performance
		SVP	0.36	0.6654	Accept H0	Both classifiers have the same performance
		BCF	0	0	Reject H0	MLP is better than BCF
		BCC	1	0	Reject H0	BCC is better than MLP
		RF	0.38	0.0052	Reject H0	RF is better than MLP
RF	0.38	DT	0.48	0	Reject H0	DT is better than RF
		NB	0.37	0.4435	Accept H0	Both classifiers have the same performance
		SVP	0.36	0.4724	Accept H0	Both classifiers have the same performance
		BCF	0	0	Reject H0	RF is better than BCF
		BCC	1	0	Reject H0	BCC is better than RF
		MLP	0.34	0.0052	Reject H0	RF is better than MLP

3.3 Practical Usefulness

The obtained classifiers can be used beneficially in realistic bug prediction scenario. They can be deployed in the software development life cycle. For instance, Decision Tree classifier surpassed the other classifiers in terms of recall, while Multi-layer Perceptron classifier surpassed the other classifiers in terms of precision, finally Random Forest classifier surpassed the other classifiers in terms of F1. This appealing results encourage the practical usefulness of the classifiers as it can help software developers to assess the quality of their code during the development and maintenance phases.
This assessment can help in preventing in future bugs and give a clear image to software developers about the quality of their repositories.