IMA Project 2

Fabian Gobet

June 17, 2024

0 Code Repository

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

Repo:
https://github.com/infoMA2023/project-02-bug-prediction-FabianGobet
Commit: ff91781

1 Data Pre-Processing

After having cloned the git repository, all work was based on the .java files found in /resources/defects4j-checkout-closure-1f/src/com/google/javascript/jscomp, whereas the true label for buggy java classes can be found in resources/modified_classes.

A Jupiter notebook named all.ipynb, found in the sketches folder, contains the agglomerate code of all sections in a tested manner for implementation purposes. Furthermore, all generated files can be found in the generated folder.

1.1 Feature Vector Extraction

The code to extract the feature vectors of all classes consists in a walk through all .java files of the former mentioned working directory. For each .java file, only outer classes are considered, e.g. classes within classes are not accounted for.

For each of the metrics referenced in the slide there exists a function that specifically generated such values. Some of these functions return more than one metric because the search procedure is very similar for metrics within the same function.

For each of the classes, after having generated the feature vectors, all this data is saved into a CSV file under the name of class_metrics.csv. One of the fields is the relative path to the class .java file itself, which has proven itself in the past to be useful to have. This parameter column is appropriately handled (discarded) in future actions.

After execution and generation of the CSV file, I ended up with 291 classes of which I took the liberty of picking the 8 first and their respective features,

with some rounded to 2 decimal cases for visualization purposes, giving origin to Figure 1.

Class Name	Mth	Fld	Rfc	Int	Ex	Ret	Bcm	Wrd	Nml	Sz	Cpx	Dcm
PeepholeSimplifyRegExp	1	0	29	0	0	2	1	15	15.0	18	4	0.833
MinimizeExitPoints	6	1	91	0	0	9	5	391	13.833	113	28	3.46
RenameLabels	1	3	51	0	0	4	8	412	10.2	51	9	8.078
JsMessageExtractor	2	3	19	0	1	3	5	198	14.25	22	4	9.0
Tracer	25	16	177	2	1	34	70	2381	13.714	276	58	8.627
VerboseMessageFormatter	3	0	12	0	0	3	1	20	10.0	7	1	2.857
DiagnosticType	9	5	15	0	0	9	11	195	6.778	13	0	15.0
StrictModeCheck	9	13	107	0	0	3	11	158	11.75	107	36	1.477

Figure 1: Data correlation matrix

1.2 Feature Vector Labelling

In order to know all buggy classes, similarly like before, a walk through all files in the modified_classes folder is done, extracting the classes name from each .src file. It worth noting that each of these files may contain more than one class.

After extracting all buggy class names into a list, to the former feature vectors CSV file a new column is added and the respective classes are marked with a value 1, whilst the remaining stay at 0.

A total of 76 classes are labeled buggy, while the remaining 215 are not.

2 Classifiers

For all classifiers a Grid Search strategy was employed to find the optimal hyper parameter configurations. The scoring strategy employed was precision, recall and f1 with a fitting to the latter. All these experiments can be consulted in the prior mentioned all.ipynb file.

2.1 Decision Tree (DT)

Optimal found parameters for Decision tree were the gini criterion, witha max depth of 60, 8 minimum number of samples per leaf, 2 minimum number of samples to split with a random splitter strategy.
This resulted in a precision of 0.44, a recall of 0.5 and f1 score of 0.47.

2.2 Naive Bayes (NB)

For the Naive Bayes classifierno hyperparameters were needed to be declared. The results showed a precision of 0.8, with a recall of 0.33 and f1 score of 0.47

2.3 Support Vector Machine (SVM)

For the support vector machine, the best hyperparamaters found were a regularization parameters of 10, with a gamma value of 0.0001 using the RBF kernel. The resulting score were 0.46 for precision, 0.37 for recall and 0.41 for f1.

2.4 Multi-Layer Perceptron (MLP)

For the Multi Layer Perceptron, the optimal found values were 3 hidden layers of sizes 12, 50 and 25, with relu as activation function, an alpha value of 0.05, constant learning rate with adam as the solver optimizer. This resulted in 0.5 precision, 0.12 recall and an f1 score of 0.2.

2.5 Random Forest (RF)

For ther random forest the best found criterion was gini, with a max depth of 8, 10 estimators and limitation of square root for the max nuber of features. This resulted in 0.41 precision, 0.31 recall and an f1 score of 0.35.

3 Evaluation

3.1 Output Distributions

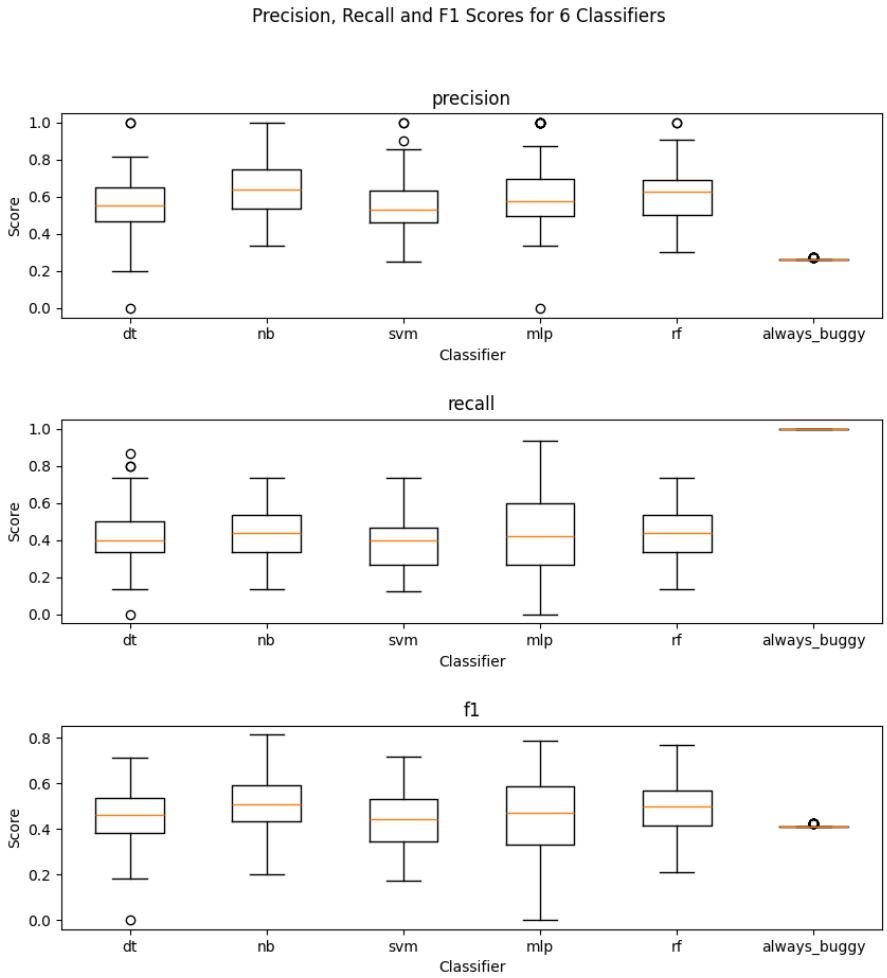

Figure 2: Decision Tree vs. AlwaysBuggy

	metric	Decision Tree	Naive Bayes	SVM	MLP	Random Forest	Always Buggy
0	f1 mean	0.460854	0.512582	0.436686	0.451467	0.49421	0.4141
1	f1 std	0.123348	0.115593	0.118886	0.161733	0.121723	0.006283
2	precision mean	0.554474	0.643369	0.551581	0.599379	0.603918	0.261134
3	precision std	0.151186	0.143019	0.15168	0.179501	0.139632	0.005026
4	recall mean	0.413875	0.435625	0.384625	0.419125	0.434792	1.0
5	recall std	0.147362	0.115565	0.138379	0.213742	0.133223	0.0

Figure 3: Mean and standard deviation for all metrics in all classifiers

3.2 Comparison and Significance

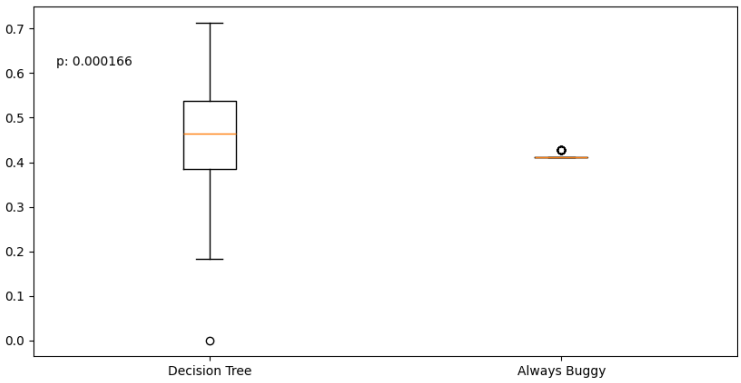

Figure 4: Decision Tree vs. Always Buggy

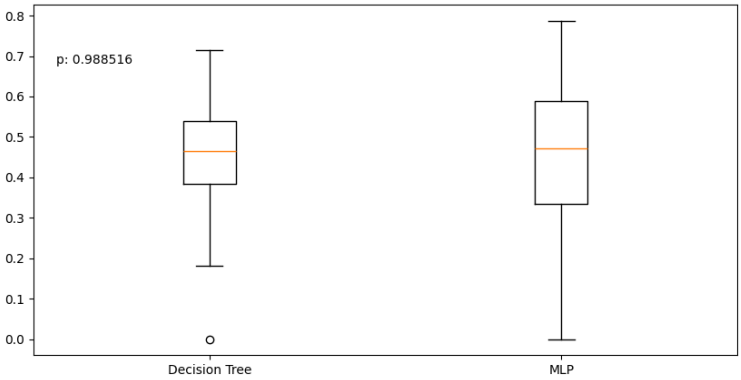

Figure 5: Decision Tree vs. MLP

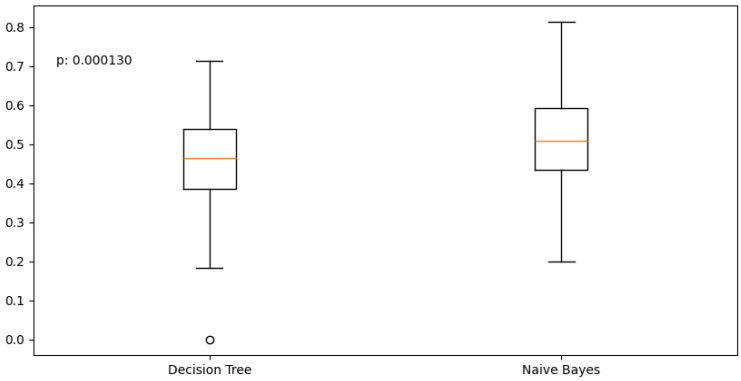

Figure 6: Decision Tree vs. Naive Bayes

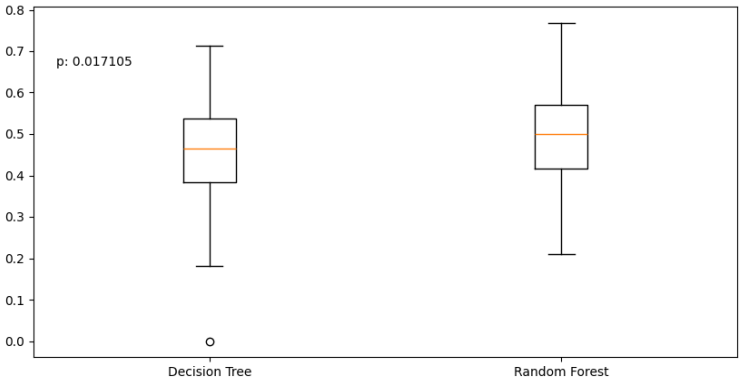

Figure 7: Decision Tree vs. Rnadom Forest

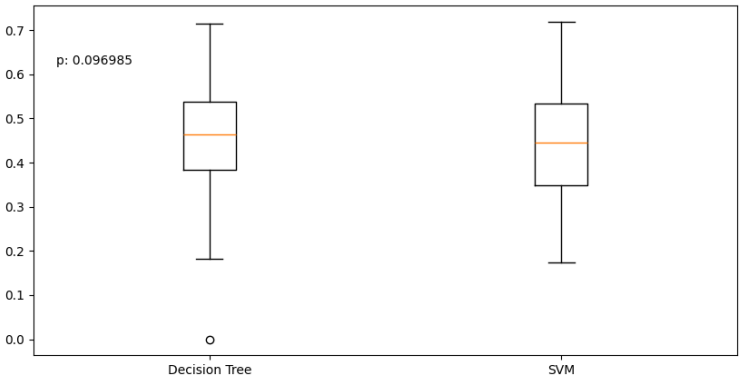

Figure 8: Decision Tree vs. SVMl

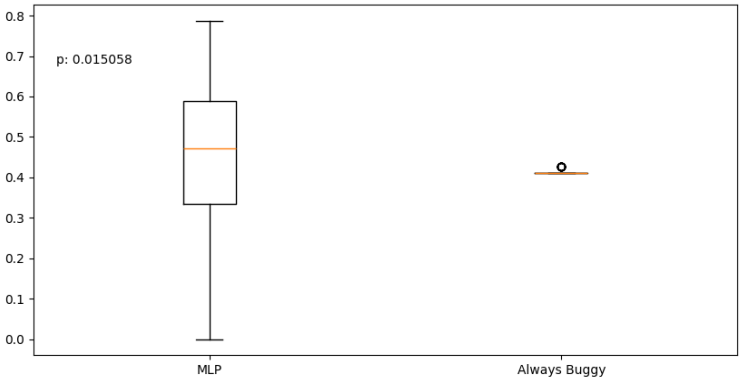

Figure 9: MLP vs. Always Buggy

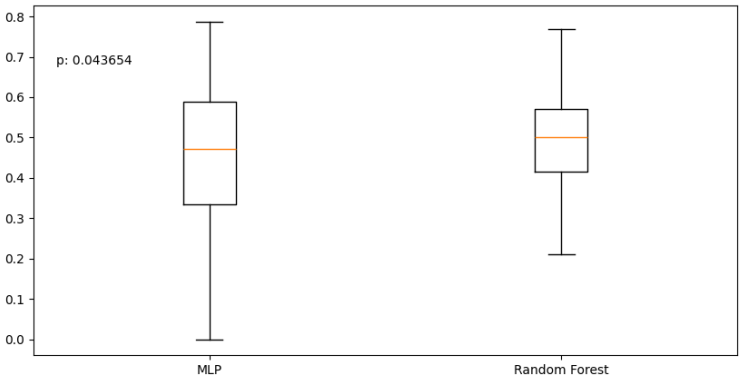

Figure 10: MLP vs. Random Forest

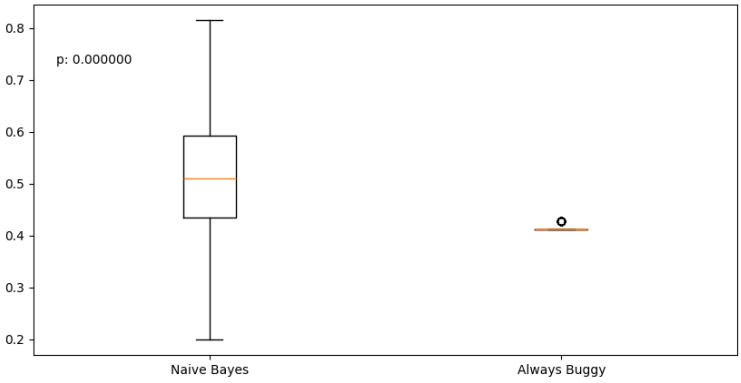

Figure 11: Naive Bayes vs. Always Buggy

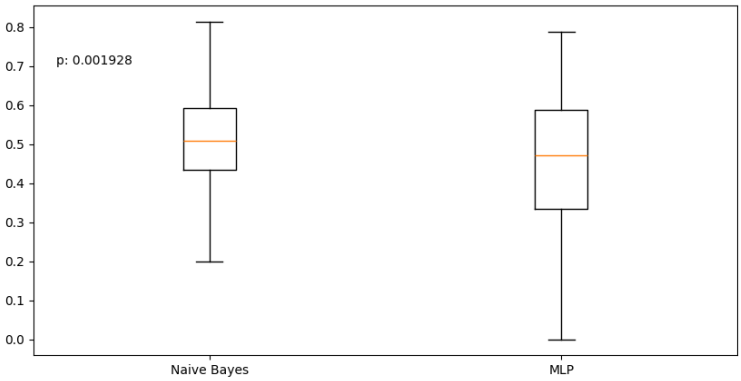

Figure 12: Naive Bayes vs. MLP

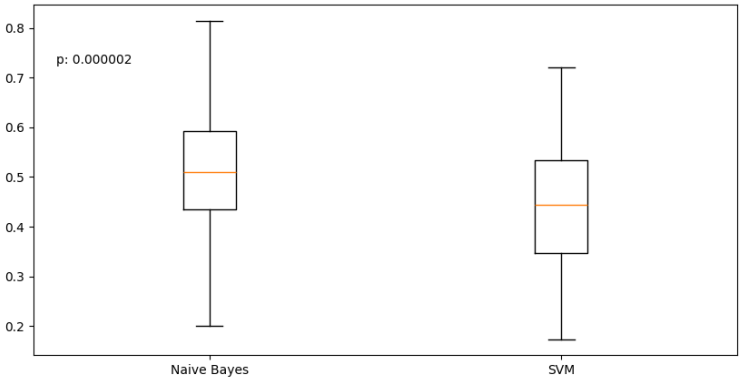

Figure 13: Naive Bayes vs. SVM

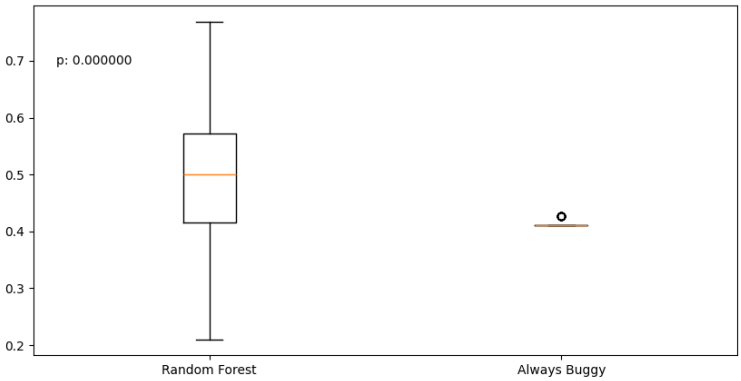

Figure 14: Random Forest vs. Always Buggy

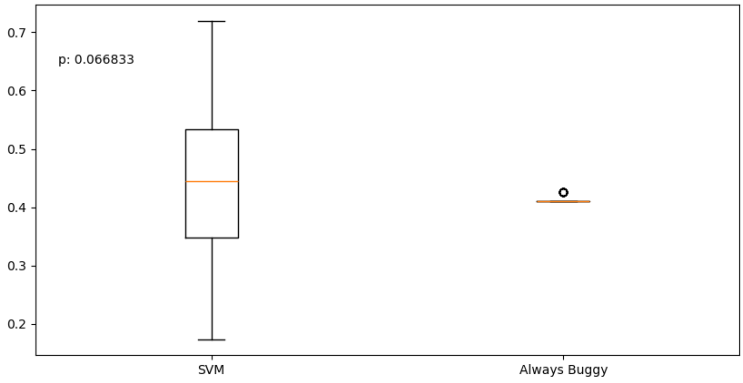

Figure 15: SVM vs. Always Buggy

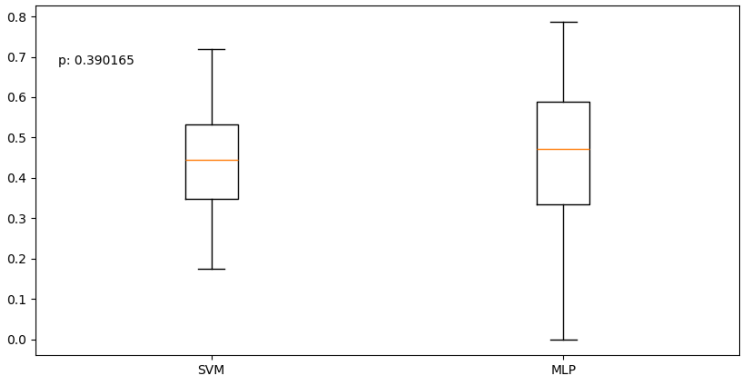

Figure 16: SVM vs. MLP

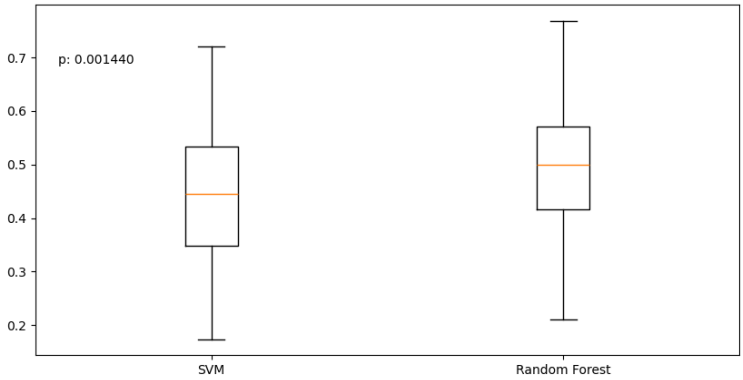

Figure 17: SVM vs. Random Forest

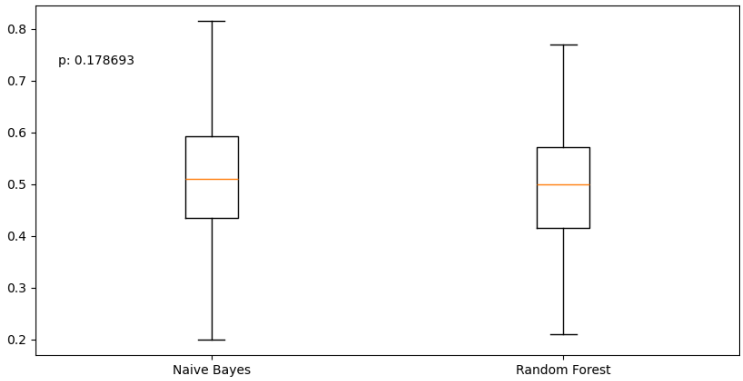

Figure 18: Naive Bayes vs. Random Forest

	method	Naive Bayes	SVM	MLP	Random Forest	Always Buggy
0	Decision Tree	0.025004	0.000118	0.944062	0.093184	0.000002
1	Naive Bayes	NaN	0.0	0.041169	0.292752	0.0
2	SVM	NaN	NaN	0.000607	0.000001	0.272696
3	MLP	NaN	NaN	NaN	0.214035	0.000578
4	Random Forest	NaN	NaN	NaN	NaN	0.0
5	Always Buggy	NaN	NaN	NaN	NaN	NaN

Figure 19: p-values pairs

3.2.1 F1 Values

For the F1 score, which balances precision and recall, Naive Bayes has the highest mean F1 score at 0.512582. Always Buggy, despite its perfect recall, has the lowest F1 score at 0.4141 due to its poor precision. Random Forest, Decision Tree, MLP, and SVM have mean F1 scores of 0.49421, 0.460854, 0.451467, and 0.436686, respectively. The standard deviation for the F1 score is highest for MLP (0.161733) and lowest for Always Buggy (0.006283). Statistically, Naive Bayes significantly outperforms Decision Tree (p=0.025004), SVM (p=0.0), and MLP (p=0.041169) in terms of the F1 score. Random Forest also significantly outperforms SVM (p=0.000001).

3.2.2 Precision

For precision, the Naive Bayes classifier has the highest mean precision at 0.643369, indicating it makes fewer false positive errors compared to other classifiers. The lowest mean precision is observed with the Always Buggy classifier at 0.261134. The Random Forest and MLP classifiers follow with mean precisions of 0.603918 and 0.599379, respectively, while the Decision Tree and SVM classifiers have slightly lower precisions at 0.554474 and 0.551581. The precision standard deviation is highest for MLP (0.179501) and lowest for Always Buggy (0.005026), indicating consistent yet poor precision for Always Buggy. Statistically, the precision of Naive Bayes is significantly better than that of Decision Tree (p=0.025004) and SVM (p=0.0).

3.2.3 Recall

For recall, the Always Buggy classifier achieves the highest mean recall at 1.0. This is expected as it marks every instance as positive, capturing all actual positives but at the cost of many false positives. SVM has the lowest mean recall at 0.384625, meaning it misses more actual positive instances. Naive Bayes, Random Forest, MLP, and Decision Tree have recall means of 0.435625, 0.434792, 0.419125, and 0.413875, respectively. The standard deviation for recall

is highest for MLP (0.213742) and lowest for Always Buggy (0.0). Statistically, Naive Bayes significantly outperforms Decision Tree in recall (p=0.025004). The perfect recall of Always Buggy is due to its simplistic strategy and is not a desirable trait in balanced classification.

3.3 Practical Usefulness

In conclusion, Naive Bayes performs best in terms of precision, indicating its strength in reducing false positives. Always Buggy achieves perfect recall, but this is due to its simplistic strategy that leads to poor precision. The F1 score analysis shows that Naive Bayes is the most effective overall, balancing precision and recall effectively. The p-values further confirm that Naive Bayes generally outperforms other classifiers, particularly Decision Tree and SVM, across these metrics, indicating its robustness in this dataset.