Benchmarking Machine Learning Tools for Software Engineering Prediction Problem

Frangos Alexandros, Hadjivasiliou Adamos, Koupanou Karolina, Papavasiliou Andreas and Yunie Lou

I. Introduction

NOWADAYS many software developers and researchers have leveraged software analytics to enhance their decision-making process. Software analytics is analysis on software data to support practitioners so that they could obtain insightful and useful information to make better technical and business decisions[1]. When applying in the practice, a popular approach is to use machine learning techniques.[2] Therefore, it has become an increasingly important part of this field to research and analyse how to apply machine learning techniques better.

Our project aims to analyse and compare machine learning tools for defect prediction and effort estimation. We run multiple experiments on Scikit-learn and Caret using various datasets to evaluate their performance. We also test and compare different versions of the Caret library on the same dataset. The ML algorithms used in our experiments contain CART (Classification And Regression Tree), KNN (K-Nearest Neighbours), LR (Logistic Regression) and SVM (Support Vector Machine). The performance measures are MCC, Dis2Heaven, Precision, Recall, f-Measure and AUC for defect prediction and MAE, MedAE and stdefv for effort estimation.

Our results show that the Caret library has better performance than Sklearn both in defect prediction and effort estimation, regardless of whether hyper-parameter tuning is used. For effort estimation, the difference between Caret versions has little effects on its performance. The results are consistent across all the versions. But for defect prediction, the results are different in the early versions.

The remainder of this report is organised as follows. The next section provides the related work. Section 3 describes the research hypothesis. A detailed experiment design including methodology, dataset description, metrics and benchmarks is given in Section 4. Then, we analyse our results in Section 5 and discuss the limitations in Section 6. Section 7 has the conclusions and future work.

II. Related Work

Before beginning our research and our experiment, we searched for papers researching a similar topic to ours. We analysed and identified the strengths and weaknesses of the work of Roy, A. et al. for the paper "Performance Comparison of Machine Learning Platforms" [3], which had the most similar research hypothesis as our project.

Since the paper was published in 2019, it compared machine learning platforms that are widely used nowadays such as Python, R, Apache Spark ML and Microsoft Azure ML. They tested exclusively classification algorithms and they evaluated different metrics including accuracy, area under the curve (AUC) and F-score. After they picked the best method for each platform according to the accuracy they proceeded to further investigation. The fact that they have used data sets that are publicly available and widely used enable future researchers to compare their measurements or even verify the accuracy of the results of the research paper. Last but not least, they tested both default and tuned parameters in order to come up with fascinating conclusions.

However, the validity of the conclusions and the fair comparison of the results from the paper are questioned due to the lack of consistency and the process chosen in the experiments carried out. First of all, they do not specify the version of all the tools used so we cannot be aware of the changes that are launched from the time that they carried out the experiment. Also, the Naïve Bayes technique was replaced for some platforms by Bayesian Nets technique that might result in bias in the evaluation. Moreover, the hardware used while testing the different techniques through different platforms was not the same, which may affect the outcome of the study. In addition, due to the fact that the time needed for some platforms to complete certain algorithms was high, they had to alter the process manually to get a result.

Overall, even though there were some experimental techniques that were well-structured and can be adapted in future research, there is no doubt that there were several parts that made the research biased and its validity not guaranteed. It inspired us to proceed to our project and to produce a well-designed and organised

experimental study to ensure a reliable outcome.

III. Research Hypothesis

Our research aims to investigate the possible differences between platforms and techniques of machine learning (ML) classification and regression algorithms according to their effectiveness and their accuracy. After completing our literature review in the first term we concluded that the most relevant and useful platforms currently are Python and R. The tools used are scikit-learn (sklearn) and caret for Python and R respectively. A defect prediction algorithm was used for classification algorithm testing whereas an effort estimation algorithm was used for regression algorithm testing. For both types of algorithms, we used four techniques: Support Vector Machine (SVM), Logistic and Linear Regression (LR), Classification and Regression Trees (CART) and K-Nearest Neighbors (KNN). We then completed the research questions that are mentioned below. The results would be compared by using Precision, Recall, F-measure (F1 score), Area Under the Curve (AUC), Matthews Correlation Coefficient (MCC) and Distance to heaven (Dis2heaven) matrix. For all the experiments we used laptops with the same hardware specifications (MacBook Pro, 2.9 GHz Quad-Core Intel Core i7 processor, 16 GB 2133 MHz LPDDR3 memory).

1) Research Question 1

The first research question was whether the outcome from sklearn and caret algorithms with the default settings and hyper-parameters and for the same datasets are different. This question was important to be answered before investigating further the tools. Based on the answer, we formed the rest of our research questions and the different tests we planned to carry out.

2) Research Question 2

Since caret includes hyper-parameter tuning by default, the second research question aims to show how the results obtained by Sklearn differ from those obtained by Caret when no tuning is done in the latter. By applying this change in our tests, we would discover whether tuning affects the results in terms of accuracy and if so, by how much.

3) Research Question 3

The third research question investigates how the two tools compare in terms of evaluation when the same hyper-parameters are set for both algorithms. By answering this question, we eliminate any differences in the results due to the different default settings of the tools and we can make a fairer comparison. This task was divided in two subsections to cover all the possibilities that cause the methods to output non-identical results. Research Question 3.1 was how the performance of the models vary between the two tools when we manually set the default caret parameters for both algorithms (sklearn and caret). Whereas Research Question 3.2 investigates how the results vary in terms of performance when the default sklearn parameters are used for both libraries.

4) Research Question 4

The fourth and last research question verifies how results compare for different versions of caret with the latest R version. The caret versions chosen to be tested were 6.0-84, 6.0-80, 6.0-76, 6.0-68, 6.0-47, 6.0-29, 5.16-04, 5.15-023, 4.88, 4.37, 4.11, 3.16. By answering this research question, we could also observe the potential improvements of each version for the library in terms of model performance.

IV. Experiment Design

A. Methodology

To answer the research questions described above, we have carried out an empirical study. We perform the experiments on different datasets to compare the performance of the Sklearn and Caret libraries. We test on the 'lucene-2.0', 'lucene-2.2', 'lucene-2.4' datasets for defect prediction and the 'KitchDev' and 'KitchPerf' datasets for effort estimation. These datasets are all publicly available and widely used.The study is divided into three parts.

The first one corresponds to Research Question 1. We run the Sklearn and Caret models as they are. We use the default settings without any adjustments. In this case, Caret uses hyper-parameter tuning by default.

For Research Question 2, we turn off the hyper-parameter tuning for Caret and compare the results with those of models run in Sklearn. The outcome would tell the effects of tuning on the accuracy.

We then conduct another experiment to answer Research Question 3. We would like to see how the model performance vary when the two libraries use the same settings. In this part, We first use the default values of all parameters for all techniques in the Caret library to run both the Caret and Sklearn models, and then use the default values in the Sklearn library to run both the Sklearn and Caret models. Finally, we compare the results of running the two libraries with the same parameter values.

In addition, we test different versions of the Caret library on the same dataset. This is used to see the effect of versions on the model performance. For defect prediction, we test the following versions on the 'lucene-2.4' dataset: caret_6.0-84, caret_6.0-80, caret_6.0-76, caret_6.0-68, caret_6.0-47, caret_6.0-29,

caret_5.16-04, caret_5.15-023, caret_4.88, caret_4.37, caret_4.11, caret_3.16. For effort estimation, we test the same versions on the 'KitchDev' dataset. This experiment is for Research Question 4.

B. Dataset Description / Data Collection:

Our goal was to use widely used and publicly available datasets so that future researchers will be able to compare measurements and verify the accuracy of the results of our paper. We were not very concerned however with finding a particular dataset as we were not interested in the optimization of the actual accuracy of the models and platforms used. Our main priority was to find reliable and widely used publicly available datasets. We used three different datasets for Defect Prediction and two for Effort Estimation. In the table below the datasets used for both are summarised.

Defect Prediction	Rows	Features
Lucene-2.4.csv	341	12
Lucene-2.2.csv	248	12
Lucene-2.0.csv	196	12

Effort Estimation	Rows	Features
KitchDevData.csv	30	14
KitchPerfData.csv	41	14

C. Metrics

1) Defect Prediction

After creating the models based on each algorithm provided, we need to evaluate the performance of these models. The performance of the model can be thought as the ability to correctly classify unseen data. In order to test how the models in Defect Prediction are performing we will use Classification Metrics as our problem requires classification to be solved as mentioned before. In several problems testing a model against a single metric may not give the correct evaluation of the model. It is suggested that a subset of metrics is used in order to have an accurate determination of the performance of our models. As a team we chose to use the following metrics, MCC, Dis2heaven, Precision, Recall, Fmeasure and AUC. A detailed explanation on all of them is provided as well as the reason we chose to use each one of these metrics. [4][5]

- Precision

In cases where our class distributions are not balanced, one class is more frequent than the rest, we need a class specific performance metric if we want an accurate evaluation of our model. One such example is precision which can be used to find the percentage at which defective class, of our model, is correctly predicted based on the number of predictions made. The formula to calculate the precision is as follows:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

In the formula above True Positive stands for the number of correctly predicted as positive class or class one. False Positive stands for the number of incorrectly predicted as class one. In a simpler manner one can say that precision is the number of correctly predicted divided by the total number of predictions made for the positive class.[6]

- Recall

Similar to the precision metric, recall is a very important metric which can be defined as the percentage of the correctly predicted elements of the positive class over the total elements of this class. The formula for recall is the following:

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

The only difference it has to precision is the use of False Negative instead of False Positive. False Negative stands for the incorrectly predicted negative class, therefore True Positive + False Negative as a sum stands for the total true cases that exist.[6]

F-measure (F1 score) Since in most cases recall and precision are really important and help in understanding how our model is performing, combining them in a single metric lead to various other metrics. One example of them is f-measure which is the harmonic mean of precision and recall [6]:

$$F{-}measure = 2{*}Precision{*}Recall/(Precision{+}Recall)$$

- AUC

This metric widely known as Area Under the Curve is another performance measurement for classification which, using the ROC curve, provides the probability with which the model is capable of distinguishing each class. ROC is a probability curve and AUC is used to represent the degree of separability. The higher the AUC value, the better the performance of the model is.[7]

- Matthews Correlation Coefficient (MCC)

In our case of the Defect Prediction we have two possible classes, defect or non-defect. Hence as this is a binary classification we can successfully apply

the MCC metric to our problem and provide ourselves with more accurate interpretation of the performance of our model. Precision, recall and f-measure are all metrics that are asymmetric hence the need of a more sophisticated metric exists. MCC is not affected if one class is disproportionately under or over represented than another one.

To calculate the MCC you just have to calculate the correlation coefficient between the true and predicted values. The higher this correlation is the better the predictions our model makes. This is similar to the phi-coefficient ϕ although renamed to MCC for classifiers.

$$MCC = \frac{TP*TN-FP*FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

By using this metric, we add to our performance evaluation a new form of evaluation which is perfectly symmetric.[8]

- Dis2heaven (Distance to heaven)

[9]

$$dis2heaven = \frac{\sqrt{(1-recall)^2 + (falsealarm)^2}}{2}$$

2) Effort Estimation

As in defect prediction, after developing the models for the algorithms used we used a number of evaluation metrics to measure the quality of our machine learning models. It is of high importance to use multiple metrics to evaluate a model, thus for effort estimation we used three which are: Mean Absolute Error (MAE), Median Absolute Error (MedAE) and Standard Deviation (stDev). Below an explanation of the metrics is outlined including the reason we have chosen to use each one of them.

- Mean Absolute Error (MAE)

The Mean Absolute Error uses residuals which are the difference between the actual value and the model's estimate. By calculating the absolute value of the residual of every data point and then taking the average of all these residuals you compute the Mean Absolute Error. [10] Mathematically MAE can be explained as:

$$MAE = 1/n * \sigma|y - y^1|$$

Where n = total number of data points, y = actual output value, yÎ The MAE metric is very intuitive and we chose it to get an indication if our model is great at prediction. A small MAE suggests a great predictive model whereas a large MAE suggests that the model may have trouble in certain areas.

- Median Absolute Error (MedAE)

The Median Absolute Error computes the weighted median absolute error of the model's data. It is calculated by taking the median of each absolute difference between the actual output value and the predicted output value.[11] Mathematically this is explained below.

$$MedAE = median(|y1 - y^1|, .., |y * n - y^n|$$

Where n = total number of data points, y = actual output value, y^n = predicted output value.

We have chosen MedAE as a metric as it is less affected by outliers making it a robust statistic.

- Standard Deviation (stDev)

Standard deviation is a measure of dispersion and an indicator of how spread our data values are. A low standard deviation signifies that values tend to be close to the mean whereas a high standard deviation indicates that the values are spread out over a wider range. Using this metric to calculate the variability of a population is an important test of a machine learning model's accuracy against real world data.[12] Mathematically this is shown in equation (1).

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{n}}$$

Where n = total number of data points, y_i = each value from the population, $\bar{y}$ = mean.

D. Benchmarks

- Classification and Regression Tree (CART):

The regression and classification tree is a machine learning method to build predictive models from specific datasets. The data is split into numerous blocks recursively and the predictive model is fit on each of such partitions of the prediction model, where each partition represents the data as a graphical decision tree.

The main difference between Classification Trees and Regression Trees is that the Classification Decision Trees are built with unordered values with dependent variables whereas Regression Decision Trees take ordered values with continuous values.[13]

- k-Nearest Neighbours (k-NN):

The k-nearest neighbors (KNN) algorithm is a supervised machine learning algorithm which is used for both classification and regression. In both situations, the input consists of the k closest training examples in the feature space and the output depends on whether k-nearest neighbors is used for classification or regression.

The key difference between k-NN for Regression and k-NN for Classification is that k-NN Regression attempts to predict the value of the output variable by using a local average

whereas k-NN Classification predicts the class to which the output variable belongs by computing the local probability. [14]

- Linear Regression (LR):

Linear Regression is a supervised machine learning algorithm, which performs the task to predict a dependent variable based on a given independent variable. [15] A Linear Regression model aims to model the relationship between two variables by drawing a line of best fit on the observed data of the form $y = a + bX$,

Where X is the explanatory variable and Y is the dependent variable.

- Support Vector Machine (SVM):

Support Vector Machines are supervised algorithms that can be used for both Classification and Regression. Given labeled training data the algorithm attempts to output an optimal hyperplane which categorizes new examples. The main advantages of using SVM are that it works well on smaller and cleaner datasets and it can be more efficient as it uses a subset of training points. [16]

V. Analysis of Results

Note: Raw results for each question and be found in the appendix.

1) Research question 1

For the first research question, which was to see if the same algorithms output different results when run on sklearn and caret with all default settings, we saw large variances between the two languages. Both for effort estimation and defect prediction, caret was more accurate in general than sklearn, which is to be expected since caret utilizes tuning with hyper-parameters by default. Hence caret produced different results also with each run of the algorithms. This was the main motivation for our second research question, since theoretically a deterministic approach is better for our project than a non-deterministic one. Additionally, caret on average still produced more accurate results on most metrics, on all databases. For defect prediction sklearn was on all accounts less accurate, but for effort estimation, while caret's results had smaller mean and median absolute error, the standard deviation was slightly higher. That is consistent with a non-deterministic approach's behaviour.

2) Research question 2

These results show a similar comparison, between sklearn algorithms with still default parameters, but now caret algorithms used with no tuning. The results were close to question 1, we again found large variances between the two platforms. The relationships between the results of the metric were the same, with the only exception that now caret had less standard deviation for effort estimation, result of the no tuning approach.

3) Research Question 3

The third question is how the two tools compare in terms of evaluation when the same hyper-parameters are set for both algorithms. We also found significant differences, although smaller. More importantly, for all models, it is impossible to set all the parameters the same for both platforms. This is why the packages offer the option to set some common parameters, but also some others that are unique to each platform. Both are not performing the training of the algorithm themselves; they instead call packages that is specific for each model. Inspecting these packages showed even more underlying mutually exclusive parameters. Thus, research question 3 was rather impossible to answer conclusively, since it's impossible to conduct exactly what the question requires, which is to set the same parameters, essentially trying to arrive at the same exact initial state of the algorithms.

4) Research Question 4

The last research question was designed to see then if there are differences even between the same platform's versions. For defect prediction, we found out that for the last releases, since caret 6.0-76 the results are exactly the same for all models and for all metrics. But going further back, their results start to differ. For CART, SVM and KNN, from caret 6.0-47 the results start to differ, changing every 2-3 versions. For LR the changes start from caret 6.0-68. This is important for the replicability of previous studies, since we also experienced a lot of issues with other packages not supporting the early versions of caret, which had its initial release in 2007. For effort estimation, interestingly results were consistent across all versions.

VI. Discussion & Limitations

We have discovered that sklearn and caret, while powerful tools, have a high level of abstraction that prevents the user from having full control over it. This is not a problem for most day to day tasks, but it severely impacts the reproducibility of relevant studies. Furthermore, since results were different in all metrics, a study that chose one platform for purely practical/unrelated reasons might have derived at different conclusion if it chose another.

While our results were conclusive, we still cannot prove that this is a phenomenon that applies to platforms other than the two chosen. It might be the case that one of the two is the outlier, and all other platforms or languages are consistent with the results of the other. Further testing is needed.

Finally, another note is that because machine learning is a relatively recent and fast-growing field, the development of platforms and the overall usage percentage of each one changes rapidly. So regular testing might be needed to keep up with the trends, and also to keep up with their latest version updates.

VII. Conclusion and Future Work

To sum up the comparison of Python and R as machine learning platforms and of the libraries used for each one answered all our initial four research questions. With the results provided for each question we can clearly support our arguments mentioned and explained above. As the machine learning industry is still developing thoroughly a wider approach on more machine learning platforms will allow for this experiment to further identify patterns between the different platforms used. A lot of new platforms are used everyday in

the industry although there is still a little research done around them. With the addition of further platforms, we are hoping to be able to identify whether a platform tested already is an outlier or not.

REFERENCES

[1] L. Guerrouj, O. Baysal, D. Lo, and F. Khomh, "Software analytics," *Proceedings of the 38th International Conference on Software Engineering Companion - ICSE 16*, 2016.

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

[3] A. Roy, S. Qureshi, K. Pande, D. Nair, K. Gairola, P. Jain, S. Singh, K. Sharma, A. Jagadale, Y.-Y. Lin *et al.*, "Performance comparison of machine learning platforms," *INFORMS Journal on Computing*, vol. 31, no. 2, pp. 207–225, 2019.

[4] A. Mishra, "Metrics to evaluate your machine learning algorithm," Nov 2018. [Online]. Available: https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234

[5] S. Minaee, "20 popular machine learning metrics. part 1: Classification regression evaluation metrics," Oct 2019. [Online]. Available: https://towardsdatascience.com/20-popular-machine-learning-metrics-part-1-classification-regression-evaluation-metrics-1ca3e282a2ce

[6] "Evaluation metrics for machine learning - accuracy, precision, recall, and f1 defined." [Online]. Available: https://pathmind.com/wiki/accuracy-precision-recall-f1

[7] S. Narkhede, "Understanding auc - roc curve," May 2019. [Online]. Available: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

[8] B. Shmueli, "Matthews correlation coefficient is the best classification metric you've never heard of," Dec 2019. [Online]. Available: https://towardsdatascience.com/the-best-classification-metric-youve-never-heard-of-the-matthews-correlation-coefficient-3bf50a2f3e9a

[9] W. Fu, T. Menzies, D. Chen, and A. Agrawal, "Building better quality predictors using "-dominance"," 2018.

[10] "Evaluation metrics," May 2019. [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/evaluation-metrics

[11] R. Bonnin, "Machine learning for developers." [Online]. Available: https://www.oreilly.com/library/view/machine-learning-for/9781786469878/9f44e711-deb6-42de-abbd-524832ad32cc.xhtml

[12] "Standard deviation," May 2019. [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/standard-deviation

[13] G. Pulipaka, "An essential guide to classification and regression trees in r language," Jun 2016. [Online]. Available: https://medium.com/gppulipaka/an-essential-guide-to-classification-and-regression-trees-in-r-language-4ced657d176b

[14] O. Harrison, "Machine learning basics with the k-nearest neighbors algorithm," Jul 2019. [Online]. Available: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

[15] "Linear regression." [Online]. Available: https://ml-cheatsheet.readthedocs.io/en/latest/linearregression.html

[16] S. Patel, "Chapter 2 : Svm (support vector machine) - theory," May 2017. [Online]. Available: https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72

VIII. Appendix

Lucene-2.0	MCC		Dis2Heaven		Precision		Recall		F-Measure		AUC	
	sklearn	caret	sklearn	caret	sklearn	caret	sklearn	caret	sklearn	caret	sklearn	caret
CART	0.24	0.34	0.38	0.48	0.59	0.83	0.63	0.32	0.61	0.46	0.62	0.72
KNN	0.11	0.23	0.45	0.41	0.53	0.62	0.47	0.48	0.50	0.54	0.55	0.62
LR	0.22	0.40	0.64	0.33	1.00	0.74	0.09	0.56	0.16	0.64	0.54	0.71
SVM	-0.17	0.39	0.71	0.32	0.00	0.71	0.00	0.59	0.00	0.65	0.47	0.70
AVERAGE	0.10	0.34	0.55	0.39	0.53	0.72	0.30	0.49	0.32	0.57	0.55	0.69

Lucene-2.2	MCC		Dis2Heaven		Precision		Recall		F-Measure		AUC	
	sklearn	caret	sklearn	caret	sklearn	caret	sklearn	caret	sklearn	caret	sklearn	caret
CART	0.26	0.31	0.38	0.33	0.69	0.86	0.69	0.41	0.69	0.77	0.63	0.71
KNN	0.25	0.33	0.39	0.56	0.68	0.42	0.72	0.44	0.05	0.53	0.62	0.61
LR	0.06	0.76	0.69	0.81	0.80	0.87	0.03	0.60	0.05	0.66	0.51	0.40
SVM	0.08	0.44	0.64	0.72	0.59	0.64	0.94	0.61	0.73	0.70	0.52	0.76
AVERAGE	0.16	0.46	0.52	0.61	0.69	0.70	0.60	0.52	0.38	0.67	0.57	0.62

Lucene-2.4	MCC		Dis2Heaven		Precision		Recall		F-Measure		AUC	
	sklearn	caret	sklearn	caret	sklearn	caret	sklearn	caret	sklearn	caret	sklearn	caret
CART	0.28	0.20	0.37	0.40	0.71	0.68	0.70	0.66	0.71	0.67	0.64	0.60
KNN	0.21	0.22	0.41	0.41	0.68	0.68	0.70	0.74	0.69	0.71	0.60	0.61
LR	0.19	0.36	0.65	0.35	1.00	0.73	0.08	0.79	0.15	0.76	0.54	0.68
SVM	0.28	0.41	0.58	0.32	0.64	0.74	0.98	0.81	0.77	0.78	0.58	0.71
AVERAGE	0.24	0.30	0.50	0.37	0.76	0.71	0.62	0.75	0.58	0.73	0.59	0.65

TABLE I: Research Question 1: Defect prediction

;

KitchDev	MAE		MedAE		StDev	
	sklearn	caret	sklearn	caret	sklearn	caret
CART	2074.20	1544.89	921.40	338.30	2916.35	2652.01
KNN	1192.97	1251.32	322.00	392.60	1713.83	1653.37
LR	1119.41	1285.50	730.00	763.58	3484.08	3176.82
SVM	1897.62	1743.32	756.00	506.75	21.06	691.53
AVERAGE	1571.05	1456.26	682.35	500.31	2033.83	2043.43

KitchPerf	MAE		MedAE		StDev	
	sklearn	caret	sklearn	caret	sklearn	caret
CART	447.57	607.01	691.85	447.57	1666.44	1583.79
KNN	268.60	359.15	268.50	268.60	1558.81	1560.14
LR	148.51	350.48	194.50	148.51	1915.20	1695.14
SVM	267.78	651.21	805.50	267.78	36.09	1069.04
AVERAGE	283.12	491.96	490.09	283.12	1294.14	1477.03

TABLE II: Research Question 1: Effort estimation

Lucene-2.0	MCC		Dis2Heaven		Precision		Recall		F-Measure		AUC	
	sklearn	caret	sklearn	caret	sklearn	caret	sklearn	caret	sklearn	caret	sklearn	caret
CART	0.24	0.36	0.38	0.32	0.59	0.67	0.63	0.64	0.61	0.65	0.62	0.68
KNN	0.11	0.33	0.45	0.34	0.53	0.65	0.47	0.63	0.50	0.64	0.55	0.66
LR	0.22	0.36	0.64	0.32	1.00	0.67	0.09	0.64	0.16	0.65	0.54	0.68
SVM	0.17	0.36	0.71	0.32	0.00	0.67	0.00	0.64	0.00	0.65	0.47	0.68
AVERAGE	0.18	0.35	0.55	0.33	0.53	0.66	0.30	0.63	0.32	0.65	0.55	0.68
Lucene-2.2	MCC		Dis2Heaven		Precision		Recall		F-Measure		AUC	
	sklearn	caret	sklearn	caret	sklearn	caret	sklearn	caret	sklearn	caret	sklearn	caret
CART	0.26	0.27	0.38	0.40	0.69	0.67	0.69	0.78	0.69	0.72	0.63	0.64
KNN	0.25	0.26	0.39	0.40	0.68	0.67	0.72	0.76	0.05	0.72	0.62	0.64
LR	0.06	0.27	0.69	0.40	0.80	0.67	0.03	0.78	0.05	0.72	0.51	0.64
SVM	0.08	0.27	0.64	0.40	0.59	0.67	0.94	0.78	0.73	0.72	0.52	0.64
AVERAGE	0.16	0.26	0.52	0.40	0.69	0.67	0.60	0.77	0.38	0.72	0.57	0.64
Lucene-2.4	MCC		Dis2Heaven		Precision		Recall		F-Measure		AUC	
	sklearn	caret	sklearn	caret	sklearn	caret	sklearn	caret	sklearn	caret	sklearn	caret
CART	0.28	0.20	0.37	0.40	0.71	0.68	0.70	0.66	0.71	0.67	0.64	0.60
KNN	0.21	0.20	0.41	0.41	0.68	0.67	0.70	0.70	0.69	0.69	0.60	0.60
LR	0.19	0.36	0.65	0.35	1.00	0.73	0.08	0.79	0.15	0.76	0.54	0.68
SVM	0.28	0.41	0.58	0.32	0.64	0.74	0.98	0.81	0.77	0.78	0.58	0.71
AVERAGE	0.24	0.29	0.50	0.37	0.76	0.71	0.62	0.74	0.58	0.72	0.59	0.65

TABLE III: Research Question 2: Defect Prediction

KitchDev	MAE		MedAE		StDev	
	sklearn	caret	sklearn	caret	sklearn	caret
CART	2074.20	2371.95	921.40	1597.50	2916.35	131.16
KNN	1192.97	1192.77	322.00	321.40	1713.83	1713.87
LR	1119.41	1241.22	730.00	455.49	3484.08	2235.47
SVM	1897.62	1778.56	756.00	668.07	21.06	414.29
AVERAGE	1571.05	1646.13	682.35	760.62	2033.83	1123.70
KitchPerf	MAE		MedAE		StDev	
	sklearn	caret	sklearn	caret	sklearn	caret
CART	447.57	305.12	691.85	542.75	1666.44	2103.32
KNN	268.60	412.93	268.50	245.34	1558.81	932.88
LR	148.51	278.01	194.50	173.80	1915.20	1572.77
SVM	267.78	183.33	805.50	654.57	36.09	295.27
AVERAGE	283.12	294.85	490.09	404.11	1294.14	1226.06

TABLE IV: Research Question 2: Effort Estimation.

		mcc	dis2heaven	precision	recall	f-measure	AUC
SVM	R	0.16	0.51	0.63	0.83	0.71	0.59
	Python	0.00	0.71	0.58	1.00	0.74	0.50
LR	R	0.13	0.48	0.62	0.75	0.68	0.57
	Python	0.06	0.69	0.80	0.03	0.05	0.51
CART	R	0.21	0.40	0.68	0.64	0.66	0.60
	Python	0.31	0.37	0.70	0.77	0.73	0.65
KNN	R	0.27	0.40	0.74	0.71	0.73	0.59
	Python	0.26	0.38	0.68	0.72	0.70	0.63

TABLE V: Research Question 3: According to R

		mcc	dis2heaven	precision	recall	f-measure	AUC
SVM	R	0.15	0.65	0.64	0.97	0.72	0.70
	Python	0.00	0.64	0.44	0.96	0.65	0.58
LR	R	0.14	0.52	0.46	0.73	0.60	0.72
	Python	0.07	0.8	0.92	0.13	0.05	0.63
CART	R	0.27	0.44	0.54	0.55	0.50	0.64
	Python	0.24	0.27	0.68	0.63	0.89	0.53
KNN	R	0.30	0.47	0.47	0.65	0.80	0.46
	Python	0.29	0.46	0.66	0.61	0.83	0.61

TABLE VI: Research Question 3: According to sklearn

	caret_4.88	caret_5.16-04	caret_6.0-29	caret_6.0-47	caret_6.0-68	caret_6.0-76	caret_6.0-80	caret_6.0-84		
CART	MCC	0.224	0.164	0.164	0.179		0.179	0.179	0.199	0.199
	Dis2Heaven	0.400	0.419	0.419	0.413		0.413	0.413	0.404	0.404
	Precision	0.684	0.668	0.668	0.674		0.674	0.674	0.680	0.680
	Recall	0.704	0.626	0.626	0.640		0.640	0.640	0.660	0.660
	f-Measure	0.694	0.646	0.646	0.657		0.657	0.657	0.670	0.670
	AUC	0.613	0.581	0.581	0.588		0.588	0.588	0.599	0.599
KNN	MCC	0.252	0.237	0.237	0.249		0.203	0.203	0.203	0.203
	Dis2Heaven	0.398	0.410	0.410	0.404		0.413	0.413	0.413	0.413
	Precision	0.688	0.680	0.680	0.684		0.675	0.675	0.675	0.675
	Recall	0.749	0.754	0.754	0.759		0.704	0.704	0.704	0.704
	f-Measure	0.717	0.715	0.715	0.720		0.689	0.689	0.689	0.689
	AUC	0.630	0.623	0.623	0.629		0.603	0.603	0.603	0.603
LR	MCC	0.365	0.365	0.365	0.365		0.365	0.358	0.358	0.358
	Dis2Heaven	0.342	0.342	0.342	0.342		0.342	0.347	0.347	0.347
	Precision	0.729	0.729	0.729	0.729		0.729	0.725	0.725	0.725
	Recall	0.793	0.793	0.793	0.793		0.793	0.793	0.793	0.793
	f-Measure	0.759	0.759	0.759	0.759		0.759	0.758	0.758	0.758
	AUC	0.688	0.688	0.688	0.688		0.688	0.685	0.685	0.685
SVM	MCC	0.416	0.416	0.404	0.404		0.409	0.409	0.409	0.409
	Dis2Heaven	0.318	0.318	0.321	0.321		0.323	0.323	0.323	0.323
	Precision	0.747	0.747	0.744	0.744		0.743	0.743	0.743	0.743
	Recall	0.813	0.813	0.803	0.803		0.813	0.813	0.813	0.813
	f-Measure	0.778	0.778	0.773	0.773		0.776	0.776	0.776	0.776
	AUC	0.714	0.714	0.707	0.707		0.711	0.711	0.711	0.711

TABLE VII: Research Question 4: Defect Prediction

	caret_4.88	caret_5.16-04	caret_6.0-29	caret_6.0-47	caret_6.0-68	caret_6.0-76	caret_6.0-80	caret_6.0-84	
CART	MAE	1545	1545	1545	1545	1545	1545	1545	1545
	MedAE	338	338	338	338	338	338	338	338
	stDev	2652	2652	2652	2652	2652	2652	2652	2652
KNN	MAE	1188	1188	1188	1188	1188	1188	1188	1188
	MedAE	321	321	321	321	321	321	321	321
	stDev	1711	1711	1711	1711	1711	1711	1711	1711
LR	MAE	1141	1141	1141	1141	1141	1257	1257	1257
	MedAE	764	764	764	764	764	764	764	764
	stDev	3460	3460	3460	3460	3460	3247	3247	3247
SVM	MAE	1791	1791	1791	1791	1791	1791	1791	1791
	MedAE	567	567	567	567	567	567	567	567
	stDev	987	987	987	987	987	987	987	987

TABLE VIII: Research Question 4: Effort Estimation