# Learning Utility-changes for Rule-based Adaptation of Dynamic Architectures

CHRISTIAN M. ADRIANO[†], Hasso Plattner Institute for Digital Engineering, Germany

SONA GHAHREMANI, Hasso Plattner Institute for Digital Engineering, Germany

HOLGER GIESE, Hasso Plattner Institute for Digital Engineering, Germany

## 1 INTRODUCTION

This report details the technical steps taken to train different machine learning models and compares decisions made with these models. We report on the intermediary results for building, validating, and evaluating these models.

*Summary of the final results.* Here we report on the final results of model training and validation 1

## 2 PREREQUISITES

Fig. 1 depicts an example of an instantiation of the mrubis runtime model in form of an object diagram. In case of *rule-based adaptation* a rule *r* is triggered when the pre-condition of the rule (i.e the left hand side) is matched. A match *m* for rule *r* is a fragment of the runtime model that satisfies the pre-condition of the rule. Fig. 2 depicts the left hand side of a rule and the gray part of Fig. 1 presents an example for a match of this left hand side of the rules. The change of the state of a component to REMOVED is considered a failure. Such a situation is therefore considered as a match (i.e. trigger) for the repair rules.

## 3 METHODOLOGY

Figure. 3 represents the proposed methodology together with the activities required to evaluate it.

*Evaluation setup.* we exported the Prediction Model to an XML exchange format, The prediction model is developed in R and exported to an XML format, pmml (prediction modeling markup language). Each prediction model has a pmml file that can later be instantiated and called from a Java code. This architecture is presented in Figure. 3.

---

[1] This report provides additional supporting material ( code and results), for the corresponding paper submitted to ICAC2018.
[*]This is the corresponding author

Authors' addresses: Christian M. Adriano, Hasso Plattner Institute for Digital Engineering, Prof.-Dr.-Helmert-Straße 2-3, Potsdam, D-14482, Germany, christian.adriano@hpi.de; Sona Ghahremani, Hasso Plattner Institute for Digital Engineering, Prof.-Dr.-Helmert-Straße 2-3, Potsdam, D-14482, Germany, sona.ghahremani@hpi.de; Holger Giese, Hasso Plattner Institute for Digital Engineering, Prof.-Dr.-Helmert-Straße 2-3, Potsdam, D-14482, Germany, holger.giese@hpi.de.
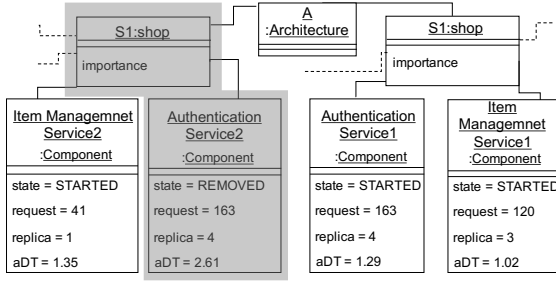
Fig. 1. Exemplary object diagram of a runtime model
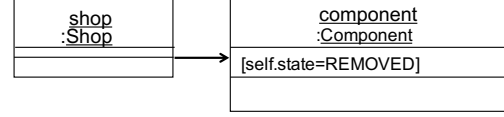


Fig. 2. Example of a match for a repair rule

## 4   APPLICATION

*4.0.1   Training.* The data used for training is publicly available [1]

[1]. $https://github.com/christianadriano/ML_SelfHealingUtility/tree/master/data/$

Table 1. Final Prediction Models

| Method | Number of Trees | Specific hyper-parameters |
| --- | --- | --- |
| XGB | 500 | objective="reg:linear" base_score=0.5 early_stopping_rounds=500 metrics="rmse" |
| GBM | 15000 | distribution="gaussian" interaction.depth=10 n.minobsinnode=5 shrinkage=1 bag.fraction=1 |
| RF | 100 | node.size=5 metrics="rmse" |



Fig. 3. Methodology and its evaluation activities

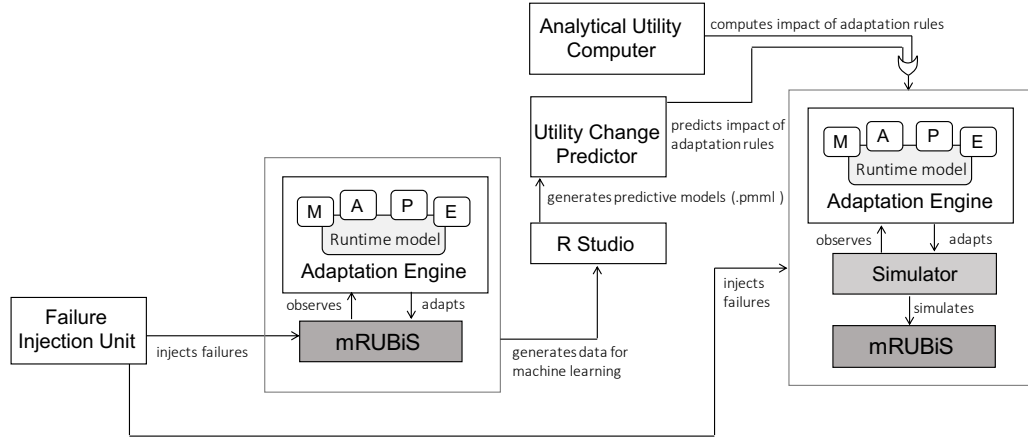| Complexity | Dataset_size | Split | Train_RMSE_MEAN | Train_RMSE_STD | Test_RMSE_MEAN | Test_RMSE_STD | RMSE | R_Squared | MAPD_% |
|---|---|---|---|---|---|---|---|---|---|
| Discontinuous | 1k | 90_10 | 0.003 | 0.000 | 26.774 | 16.521 | 3.350 | 1.000 | 0.643 |
| Discontinuous | 1k | 80_20 | 0.002 | 0.000 | 26.879 | 11.300 | 22.850 | 0.990 | 1.968 |
| Discontinuous | 1k | 70_30 | 0.002 | 0.000 | 26.091 | 14.796 | 32.464 | 0.980 | 2.657 |
| Discontinuous | 3k | 90_10 | 0.003 | 0.000 | 18.549 | 6.138 | 6.823 | 0.999 | 0.197 |
| Discontinuous | 3k | 80_20 | 0.003 | 0.000 | 18.275 | 5.494 | 11.043 | 0.998 | 0.479 |
| Discontinuous | 3k | 70_30 | 0.018 | 0.003 | 16.528 | 7.459 | 13.210 | 0.997 | 1.256 |
| Discontinuous | 9k | 90_10 | 0.008 | 0.001 | 21.963 | 9.060 | 4.923 | 1.000 | 0.127 |
| Discontinuous | 9k | 80_20 | 0.005 | 0.000 | 21.739 | 11.884 | 5.393 | 1.000 | 0.185 |
| Discontinuous | 9k | 70_30 | 0.069 | 0.007 | 20.633 | 9.277 | 21.539 | 0.994 | 0.912 |
| Linear | 1k | 90_10 | 0.001 | 0.000 | 0.237 | 0.220 | 0.048 | 1.000 | 0.022 |
| Linear | 1k | 80_20 | 0.001 | 0.000 | 0.294 | 0.294 | 0.106 | 1.000 | 0.048 |
| Linear | 1k | 70_30 | 0.002 | 0.000 | 0.277 | 0.254 | 0.068 | 1.000 | 0.062 |
| Linear | 3k | 90_10 | 0.002 | 0.000 | 0.584 | 0.556 | 0.066 | 1.000 | 0.013 |
| Linear | 3k | 80_20 | 0.003 | 0.001 | 0.553 | 0.572 | 0.237 | 1.000 | 0.044 |
| Linear | 3k | 70_30 | 0.002 | 0.000 | 0.726 | 0.606 | 0.342 | 1.000 | 0.066 |
| Linear | 9k | 90_10 | 0.002 | 0.000 | 0.200 | 0.190 | 0.202 | 1.000 | 0.013 |
| Linear | 9k | 80_20 | 0.002 | 0.000 | 0.137 | 0.099 | 0.300 | 1.000 | 0.022 |
| Linear | 9k | 70_30 | 0.002 | 0.000 | 0.206 | 0.200 | 0.326 | 1.000 | 0.018 |
| Saturating | 1k | 90_10 | 0.003 | 0.001 | 9.902 | 4.861 | 1.800 | 1.000 | 0.180 |
| Saturating | 1k | 80_20 | 0.019 | 0.002 | 11.844 | 6.543 | 3.023 | 0.999 | 0.479 |
| Saturating | 1k | 70_30 | 0.002 | 0.000 | 13.409 | 6.000 | 4.439 | 0.999 | 0.788 |
| Saturating | 3k | 90_10 | 0.004 | 0.000 | 8.169 | 3.216 | 3.174 | 1.000 | 0.152 |
| Saturating | 3k | 80_20 | 0.141 | 0.011 | 7.500 | 2.315 | 3.880 | 0.999 | 0.419 |
| Saturating | 3k | 70_30 | 0.003 | 0.000 | 8.263 | 1.962 | 5.353 | 0.999 | 0.627 |
| Saturating | 9k | 90_10 | 0.008 | 0.001 | 9.501 | 3.616 | 3.299 | 1.000 | 0.113 |
| Saturating | 9k | 80_20 | 0.035 | 0.003 | 9.124 | 1.958 | 4.690 | 0.999 | 0.248 |
| Saturating | 9k | 70_30 | 0.006 | 0.000 | 10.182 | 2.974 | 6.286 | 0.999 | 0.473 |

Fig. 4. Extreme Boost Trees trained with 500 trees over different datasets and data splits

| Utility Type | Model | Dataset Size | RMSE | R_Squared | MADP% | Elapsed Time | Number of Trees |
|---|---|---|---|---|---|---|---|
| Linear10K | XGB | 10K | 0.57 | 1.00 | 0.12 | 19.61 | 448 |
| Discontinuous10K | XGB | 10K | 24.84 | 0.99 | 3.09 | 22.98 | 448 |
| Saturating10K | XGB | 10K | 9.28 | 1.00 | 1.32 | 24.80 | 448 |
| Combined10K | XGB | 10K | 161.24 | 1.00 | 1.95 | 25.51 | 448 |

Fig. 5. Extreme Boost Trees trained with 500 trees over different datasets and data splits

*XGB model:* We first investigated how XGB performed with different dataset sizes under different splits of data between training and validation 4. Since the results of 70/30 split was better, we adopted this split for the other two method types (GBM and Random Forest). The source code that generated the XGB results is available.[2] We also trained XGB with other dataset sizes, 10K and 100k. However, the values for MADP saturated after 10K dataset size (Figure.5). Source code is available[3].

*Random Forest - RF:.* We investigate if the performance of the model would improve with a higher number of trees and sizes of datasets (Figure.6).

*Random Forest - RF-Heavy.* We trained a forest with 200 trees to investigate if we could improve MADP without impacting much the execution time. We were not successful with that (Figure.7).

---

[2] $https: //github.com/christianadriano/ML_S elf Healing U tility/blob/master/models/mainControl_X GB.R$
[3] $https: //github.com/christianadriano/ML_S elf Healing U tility/blob/master/models/mainControl_X GB.R$

| Utility Type | Model | Dataset Size | RMSE | R_Squared | MADP% | Elapsed Time | Number of Trees |
|---|---|---|---|---|---|---|---|
| Linear | Random Forest | 1k | 0.28 | 1.00 | **0.25** | 13.08 | **100** |
| Linear | Random Forest | 3K | 0.62 | 1.00 | **0.26** | 61.35 | **100** |
| Linear | Random Forest | 9K | 0.76 | 1.00 | **0.05** | 224.22 | **100** |
| Discontinuous | Random Forest | 1K | 24.80 | 0.98 | **6.43** | 49.61 | **100** |
| Discontinuous | Random Forest | 3K | 27.11 | 0.99 | **4.20** | 268.01 | **100** |
| Discontinuous | Random Forest | 9K | 25.06 | 0.99 | **1.55** | 1485.87 | **100** |
| Saturating | Random Forest | 1K | 8.17 | 1.00 | **2.13** | 17.92 | **100** |
| Saturating | Random Forest | 3K | 10.11 | 0.99 | **1.80** | 145.30 | **100** |
| Saturating | Random Forest | 9K | 11.83 | 1.00 | **1.54** | 857.70 | **100** |
| Combined | Random Forest | 1K | 375.88 | 0.97 | **8.52** | 45.43 | **100** |
| Combined | Random Forest | 3K | 416.38 | 0.98 | **7.24** | 181.80 | **100** |
| Combined | Random Forest | 9K | 429.05 | 0.98 | **4.11** | 968.25 | **100** |

Fig. 6.  Random Forests trained with 100 trees across different dataset sizes
.

| Utility Type | Model | Dataset Size | RMSE | R_Squared | MADP% | Elapsed Time | Number of Trees |
|---|---|---|---|---|---|---|---|
| Linear | Random Forest | 9K | 0.74 | 1.00 | **0.05** | 335.53 | **200** |
| Discontinuous | Random Forest | 9K | 25.55 | 0.99 | **1.54** | 1427.55 | **200** |
| Saturating | Random Forest | 9K | 11.71 | 1.00 | **1.53** | 1384.28 | **200** |
| Combined | Random Forest | 9K | 441.13 | 0.98 | **4.09** | 1704.05 | **200** |

Fig. 7.  Random Forests trained with 200 trees for 9K dataset size

| Utility Type | Model | Dataset Size | RMSE | R_Squared | MADP% | Elapsed Time | Number of Trees |
|---|---|---|---|---|---|---|---|
| Linear | GBM | 1K | 0.27 | 1.00 | **0.37** | 43.33 | **9920** |
| Linear | GBM | 3K | 1.03 | 1.00 | **1.37** | 114.37 | **726** |
| Linear | GBM | 9K | 0.83 | 1.00 | **0.58** | 331.79 | **1342** |
| Discontinuous | GBM | 1K | 13.38 | 0.99 | **13.19** | 42.31 | **9999** |
| Discontinuous | GBM | 3K | 14.04 | 1.00 | **4.71** | 107.81 | **10000** |
| Discontinuous | GBM | 9K | 13.32 | 1.00 | **2.94** | 309.84 | **6158** |
| Saturating | GBM | 1K | 4.41 | 1.00 | **1.52** | 42.08 | **9773** |
| Saturating | GBM | 3K | 8.02 | 1.00 | **1.49** | 107.42 | **10000** |
| Saturating | GBM | 9K | 7.15 | 1.00 | **1.15** | 311.08 | **9999** |
| Combined | GBM | 1K | 262.94 | 0.99 | **5.63** | 51.75 | **9996** |
| Combined | GBM | 3K | 288.42 | 0.99 | **4.87** | 112.5 | **9979** |
| Combined | GBM | 9K | 257.20 | 0.99 | **3.73** | 316.17 | **10000** |

Fig. 8.  GBM across different dataset sizes

***Tuning GBM model***: GBM require the largest number of trees compare to the other models. For this reason it also took the longer to be trained as the elapsed time column shows. We trained with 10K trees (Figure.8) and 15K trees (Figure.9). Source code is available[4].

---

[4] $https: //github.com/christianadriano/ML_SelfHealingUtility/blob/master/models/mainControl_GBM.R$

| Utility_Type | Model | Dataset_Size | RMSE | R_Squared | MAPD% | Elapsed_Time | Number_of_Trees |
|---|---|---|---|---|---|---|---|
| Linear | GBM | 1K | 0.47 | 1.00 | **0.82** | 69.31 | **171** |
| Linear | GBM | 3K | 0.87 | 1.00 | **0.91** | 179.7 | **317** |
| Linear | GBM | 9K | 0.64 | 1.00 | **0.23** | 526.21 | **714** |
| Discontinuous | GBM | 1K | 32.31 | 0.98 | **42.69** | 83.97 | **128** |
| Discontinuous | GBM | 3K | 19.33 | 0.99 | **12.61** | 221.71 | **578** |
| Discontinuous | GBM | 9K | 18.52 | 1.00 | **7.78** | 752.14 | **480** |
| Saturating | GBM | 1K | 2.93 | 1.00 | **0.78** | 84.45 | **14973** |
| Saturating | GBM | 3K | 5.41 | 1.00 | **0.74** | 234.61 | **3445** |
| Saturating | GBM | 9K | 5.04 | 1.00 | **0.37** | 676.8 | **14832** |
| Combined | GBM | 1K | 211.20 | 0.99 | **2.72** | 94.17 | **14812** |
| Combined | GBM | 3K | 247.39 | 0.99 | **3.27** | 203.19 | **1060** |
| Combined | GBM | 9K | 198.15 | 1.00 | **1.24** | 524.32 | **14954** |

Fig. 9. GBM across different dataset sizes

| Complexity | Dataset_size | Split | Train_RMSE_MEAN | Train_RMSE_STD | Test_RMSE_MEAN | Test_RMSE_STD | RMSE | R_Squared | MAPD_% |
|---|---|---|---|---|---|---|---|---|---|
| Discontinuous | 1k | 90_10 | 0.003 | 0.000 | 26.774 | 16.521 | 3.350 | 1.000 | 0.643 |
| Discontinuous | 1k | 80_20 | 0.002 | 0.000 | 26.879 | 11.300 | 22.850 | 0.990 | 1.968 |
| Discontinuous | 1k | 70_30 | 0.002 | 0.000 | 26.091 | 14.796 | 32.464 | 0.980 | 2.657 |
| Discontinuous | 3k | 90_10 | 0.003 | 0.000 | 18.549 | 6.138 | 6.823 | 0.999 | 0.197 |
| Discontinuous | 3k | 80_20 | 0.003 | 0.000 | 18.275 | 5.494 | 11.043 | 0.998 | 0.479 |
| Discontinuous | 3k | 70_30 | 0.018 | 0.003 | 16.528 | 7.459 | 13.210 | 0.997 | 1.256 |
| Discontinuous | 9k | 90_10 | 0.008 | 0.001 | 21.963 | 9.060 | 4.923 | 1.000 | 0.127 |
| Discontinuous | 9k | 80_20 | 0.005 | 0.000 | 21.739 | 11.884 | 5.393 | 1.000 | 0.185 |
| Discontinuous | 9k | 70_30 | 0.069 | 0.007 | 20.633 | 9.277 | 21.539 | 0.994 | 0.912 |
| Linear | 1k | 90_10 | 0.001 | 0.000 | 0.237 | 0.220 | 0.048 | 1.000 | 0.022 |
| Linear | 1k | 80_20 | 0.001 | 0.000 | 0.294 | 0.294 | 0.106 | 1.000 | 0.048 |
| Linear | 1k | 70_30 | 0.002 | 0.000 | 0.277 | 0.254 | 0.068 | 1.000 | 0.062 |
| Linear | 3k | 90_10 | 0.002 | 0.000 | 0.584 | 0.556 | 0.066 | 1.000 | 0.013 |
| Linear | 3k | 80_20 | 0.003 | 0.001 | 0.553 | 0.572 | 0.237 | 1.000 | 0.044 |
| Linear | 3k | 70_30 | 0.002 | 0.000 | 0.726 | 0.606 | 0.342 | 1.000 | 0.066 |
| Linear | 9k | 90_10 | 0.002 | 0.000 | 0.200 | 0.190 | 0.202 | 1.000 | 0.013 |
| Linear | 9k | 80_20 | 0.002 | 0.000 | 0.137 | 0.099 | 0.300 | 1.000 | 0.022 |
| Linear | 9k | 70_30 | 0.002 | 0.000 | 0.206 | 0.200 | 0.326 | 1.000 | 0.018 |
| Saturating | 1k | 90_10 | 0.003 | 0.001 | 9.902 | 4.861 | 1.800 | 1.000 | 0.180 |
| Saturating | 1k | 80_20 | 0.019 | 0.002 | 11.844 | 6.543 | 3.023 | 0.999 | 0.479 |
| Saturating | 1k | 70_30 | 0.002 | 0.000 | 13.409 | 6.000 | 4.439 | 0.999 | 0.788 |
| Saturating | 3k | 90_10 | 0.004 | 0.000 | 8.169 | 3.216 | 3.174 | 1.000 | 0.152 |
| Saturating | 3k | 80_20 | 0.141 | 0.011 | 7.500 | 2.315 | 3.880 | 0.999 | 0.419 |
| Saturating | 3k | 70_30 | 0.003 | 0.000 | 8.263 | 1.962 | 5.353 | 0.999 | 0.627 |
| Saturating | 9k | 90_10 | 0.008 | 0.001 | 9.501 | 3.616 | 3.299 | 1.000 | 0.113 |
| Saturating | 9k | 80_20 | 0.035 | 0.003 | 9.124 | 1.958 | 4.690 | 0.999 | 0.248 |
| Saturating | 9k | 70_30 | 0.006 | 0.000 | 10.182 | 2.974 | 6.286 | 0.999 | 0.473 |

Fig. 10. XGB across different dataset sizes

## 5 EVALUATION

We evluted the prediction models under for different dataset sizes. Results can be compared in Figure. 11

(a) Linear variant



(b) Discontinuous variant



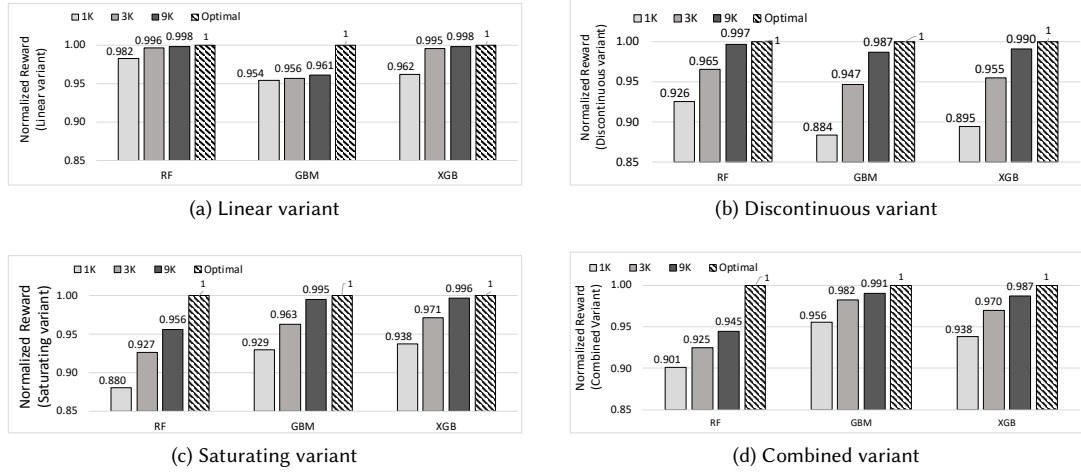(c) Saturating variant



(d) Combined variant

Fig. 11. Normalized reward of variants across learning methods and dataset sizes