# A Replication Study: Just-In-Time Defect Prediction with Ensemble Learning

S.Young, T.Abdou, A.Bener

Data Science Laboratory

Ryerson University

Toronto, Ontario

Sunday, May 27, 2018

## Outline

## Introduction

- Accurate defect prediction at change-level can assist with ensuring software quality during the development process and assist developers in finding and fixing defects in a timely manner.
- This paper reports on the replication and extension of empirical research using a set of guidelines for experimental replications to verify the results of the original study.
- Following on from the original study, we tested another ensemble called Deep Super Learner, and compared the results to the original approach.

## Introduction

- Accurate defect prediction at change-level can assist with ensuring software quality during the development process and assist developers in finding and fixing defects in a timely manner.
- This paper reports on the replication and extension of empirical research using a set of guidelines for experimental replications to verify the results of the original study.
- Following on from the original study, we tested another ensemble called Deep Super Learner, and compared the results to the original approach.

## Introduction

- Accurate defect prediction at change-level can assist with ensuring software quality during the development process and assist developers in finding and fixing defects in a timely manner.
- This paper reports on the replication and extension of empirical research using a set of guidelines for experimental replications to verify the results of the original study.
- Following on from the original study, we tested another ensemble called Deep Super Learner, and compared the results to the original approach.

## *Information About the Original Study*

- The original study proposed a methodology that employs a two-layer ensemble called TLEL
- The performance of TLEL is compared against previously proposed methodologies for just-in-time defect prediction
- One of these methodologies, Deeper, was also proposed by the authors of the original study that is being replicated here

Yang, X., Lo, D., Xia, X. and Sun, J. (2017) TLEL: A Two-layer Ensemble Learning Approach for Just-in-time Defect Prediction, in Information and Software Technology, pp. 206-220.

## Information About the Original Study

- The original study proposed a methodology that employs a two-layer ensemble called TLEL
- The performance of TLEL is compared against previously proposed methodologies for just-in-time defect prediction
- One of these methodologies, Deeper, was also proposed by the authors of the original study that is being replicated here

Yang, X., Lo, D., Xia, X. and Sun, J. (2017) TLEL: A Two-layer Ensemble Learning Approach for Just-in-time Defect Prediction, in Information and Software Technology, pp. 206-220.

## Information About the Original Study

- The original study proposed a methodology that employs a two-layer ensemble called TLEL
- The performance of TLEL is compared against previously proposed methodologies for just-in-time defect prediction
- One of these methodologies, Deeper, was also proposed by the authors of the original study that is being replicated here

`Yang, X., Lo, D., Xia, X. and Sun, J. (2017) TLEL: A Two-layer Ensemble Learning Approach for Just-in-time Defect Prediction, in Information and Software Technology, pp. 206-220.`

## Information About the Replication

- This replication study was performed externally without involvement from the original researchers
- Data from the same six open source projects and the same feature set as the original study was used
- The replication of the experiment, research questions, and design are similar to the original experiment

## *Information About the Replication*

- This replication study was performed externally without involvement from the original researchers
- Data from the same six open source projects and the same feature set as the original study was used
- The replication of the experiment, research questions, and design are similar to the original experiment

## *Information About the Replication*

- This replication study was performed externally without involvement from the original researchers
- Data from the same six open source projects and the same feature set as the original study was used
- The replication of the experiment, research questions, and design are similar to the original experiment

## Deep Super Learner

- Can use any arbitrary set of base learners, essential for performance and a strong generalization ability
- Optimized weights for the classifiers
- An adaptive number of layers depending on the dataset enables the extraction of features from lower to higher level layers to improve performance

Young S., Abdou T., Bener A. (2018) Deep Super Learner: A Deep Ensemble for Classification Problems. In: Bagheri E., Cheung J. (eds) Advances in Artificial Intelligence. pp 84-95 Canadian AI18. Lecture Notes in Computer Science, vol 10832. Springer, Cham

## Research Questions

1. How effective are the classification methodologies
2. How effective are they with different percentages of lines of code inspected
3. What is the benefit of having multiple layers
4. What is the effect of varying the amount of training data on their effectiveness
5. How much time does it take for the methodologies to run
6. What is the effect of varying the parameter settings

- Note RQ2 is omitted due to the datasets in this replication study do not include enough information about the number of lines of code inspected to calculate cost effectiveness

## Datasets

- The used datasets come from six open source projects: Bugzilla, Columba, JDT, Platform, Mozilla, and PostgreSQL
- Combined, they contain data on 137,417 changes.
- Changes are labeled as either buggy or not buggy.
- Classes are imbalanced with the portion of buggy changes ranging from 5% to 36% within the datasets.

Kamei, Y., Shihab, E., Adams, B., Hassan, A. E., Mockus, A., Sinha, A. and Ubayashi, N. (2013) A Large-scale Empirical Study of Just-in-time Quality Assurance, IEEE Transactions on Software Engineering, 39(6), pp. 757773.

## Datasets

- The used datasets come from six open source projects: Bugzilla, Columba, JDT, Platform, Mozilla, and PostgreSQL
- Combined, they contain data on 137,417 changes.
- Changes are labeled as either buggy or not buggy.
- Classes are imbalanced with the portion of buggy changes ranging from 5% to 36% within the datasets.

Kamei, Y., Shihab, E., Adams, B., Hassan, A. E., Mockus, A., Sinha, A. and Ubayashi, N. (2013) A Large-scale Empirical Study of Just-in-time Quality Assurance, IEEE Transactions on Software Engineering, 39(6), pp. 757773.

## Datasets

- The used datasets come from six open source projects: Bugzilla, Columba, JDT, Platform, Mozilla, and PostgreSQL
- Combined, they contain data on 137,417 changes.
- Changes are labeled as either buggy or not buggy.
- Classes are imbalanced with the portion of buggy changes ranging from 5% to 36% within the datasets.

Kamei, Y., Shihab, E., Adams, B., Hassan, A. E., Mockus, A., Sinha, A. and Ubayashi, N. (2013) A Large-scale Empirical Study of Just-in-time Quality Assurance, IEEE Transactions on Software Engineering, 39(6), pp. 757773.

## *Datasets*

- The used datasets come from six open source projects: Bugzilla, Columba, JDT, Platform, Mozilla, and PostgreSQL
- Combined, they contain data on 137,417 changes.
- Changes are labeled as either buggy or not buggy.
- Classes are imbalanced with the portion of buggy changes ranging from 5% to 36% within the datasets.

Kamei, Y., Shihab, E., Adams, B., Hassan, A. E., Mockus, A., Sinha, A. and Ubayashi, N. (2013) A Large-scale Empirical Study of Just-in-time Quality Assurance, IEEE Transactions on Software Engineering, 39(6), pp. 757773.

## Original Study's Proposed Methodology

- The proposed methodology employs a Two-Layer Ensemble Learner (TLEL)
- The inner layer combines decision trees with bagging to build random forests
- The outer layer uses random under-sampling of the majority class to train multiple random forests and stacks them together in an equally weighted manner

## TLEL

```
for iteration in 1 to number of learners do
    Random sub-sample from majority class to balance classes;
    Train random forest classifier;
end
class = 0
for iteration in 1 to number of learners do
    Predict test instance with trained model;
    if predicted class is buggy then
        class ← class +1
    end
end
if class ≥ (number of learners)/2 then
    final predicted class ← buggy
end
```

## *Benchmark Algorithms*

- The performance of TLEL is compared against previously proposed methodologies for just-in-time defect prediction
- One of these methodologies, Deeper, was also proposed by the authors of the original study that is being replicated here.

## Deep Super Learner

- Deep learning uses layers of processing units where output of a layer cascades to be input of next layer
- Deep Super Learner (DSL) uses an optimally weighted average combination ensemble for each layer
- The DSL here uses five base learners: logistic regression, k-nearest neighbors, random forest, extremely randomized trees, and XGBoost
- Weighted to minimize cross entropy

## Deep Super Learner

```
for iteration in 1 to max iterations do
    Split data into k folds each with train and validate sets;
    for each fold in k folds do
        for each learner in ensemble do
            Train learner on train set in fold;
            Get class probabilities from learner on validate set in fold;
            Build predictions matrix of class probabilities;
        end
    end
    Get weights to minimize loss function with predictions and true labels;
    Get average probabilities across learners by multiplying predictions with
      weights;
    Get loss value of loss function with average probabilities and true labels;
    if loss value is less than loss value from previous iteration then
        Append average probabilities to data;
    else
        Save iteration;
        Break for;
    end
end
```

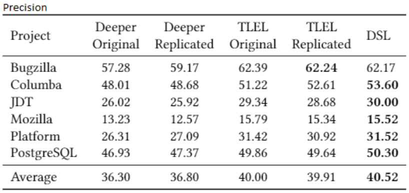
## Performance Measures

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

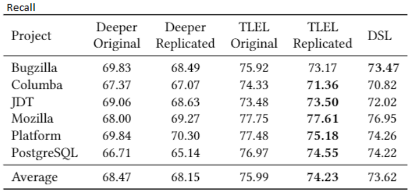
$$F_1\ score = \frac{2 * Recall * Precision}{Recall + Precision}$$

## RQ1 How effective are the methodologies?



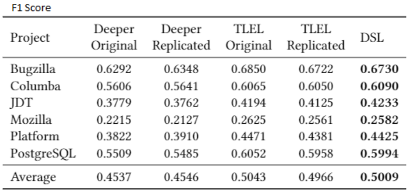
Precision

| Project | Deeper Original | Deeper Replicated | TLEL Original | TLEL Replicated | DSL |
|---|---|---|---|---|---|
| Bugzilla | 57.28 | 59.17 | 62.39 | **62.24** | 62.17 |
| Columba | 48.01 | 48.68 | 51.22 | 52.61 | **53.60** |
| JDT | 26.02 | 25.92 | 29.34 | 28.68 | **30.00** |
| Mozilla | 13.23 | 12.57 | 15.79 | 15.34 | **15.52** |
| Platform | 26.31 | 27.09 | 31.42 | 30.92 | **31.52** |
| PostgreSQL | 46.93 | 47.37 | 49.86 | 49.64 | **50.30** |
| Average | 36.30 | 36.80 | 40.00 | 39.91 | **40.52** |



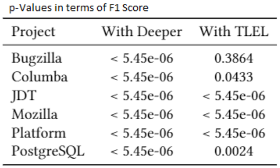
Recall

| Project | Deeper Original | Deeper Replicated | TLEL Original | TLEL Replicated | DSL |
|---|---|---|---|---|---|
| Bugzilla | 69.83 | 68.49 | 75.92 | 73.17 | **73.47** |
| Columba | 67.37 | 67.07 | 74.33 | **71.36** | 70.82 |
| JDT | 69.06 | 68.63 | 73.48 | **73.50** | 72.02 |
| Mozilla | 68.00 | 69.27 | 77.75 | **77.61** | 76.95 |
| Platform | 69.84 | 70.30 | 77.48 | **75.18** | 74.26 |
| PostgreSQL | 66.71 | 65.14 | 76.97 | **74.55** | 74.22 |
| Average | 68.47 | 68.15 | 75.99 | **74.23** | 73.62 |



F1 Score

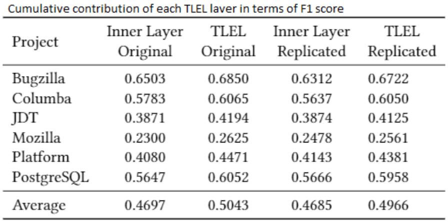
| Project | Deeper Original | Deeper Replicated | TLEL Original | TLEL Replicated | DSL |
|---|---|---|---|---|---|
| Bugzilla | 0.6292 | 0.6348 | 0.6850 | 0.6722 | **0.6730** |
| Columba | 0.5606 | 0.5641 | 0.6065 | 0.6050 | **0.6090** |
| JDT | 0.3779 | 0.3762 | 0.4194 | 0.4125 | **0.4233** |
| Mozilla | 0.2215 | 0.2127 | 0.2625 | 0.2561 | **0.2582** |
| Platform | 0.3822 | 0.3910 | 0.4471 | 0.4381 | **0.4425** |
| PostgreSQL | 0.5509 | 0.5485 | 0.6052 | 0.5958 | **0.5994** |
| Average | 0.4537 | 0.4546 | 0.5043 | 0.4966 | **0.5009** |



p-Values in terms of F1 Score

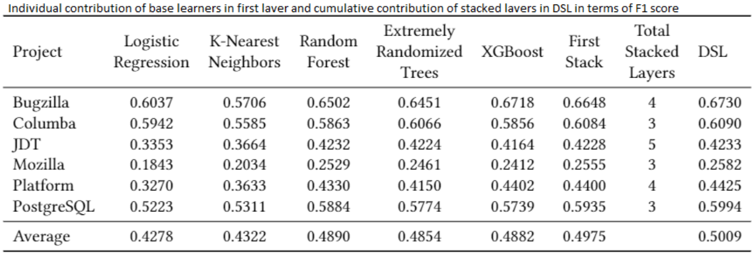
| Project | With Deeper | With TLEL |
|---|---|---|
| Bugzilla | < 5.45e-06 | 0.3864 |
| Columba | < 5.45e-06 | 0.0433 |
| JDT | < 5.45e-06 | < 5.45e-06 |
| Mozilla | < 5.45e-06 | < 5.45e-06 |
| Platform | < 5.45e-06 | < 5.45e-06 |
| PostgreSQL | < 5.45e-06 | 0.0024 |

## RQ3 What is the benefit of having multiple layers?



Cumulative contribution of each TLEL layer in terms of F1 score

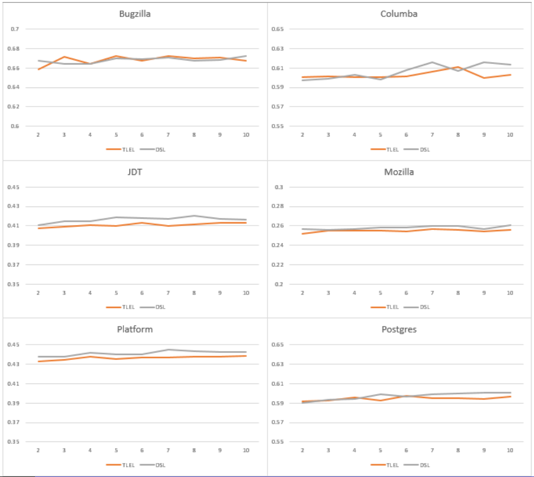
| Project | Inner Layer Original | TLEL Original | Inner Layer Replicated | TLEL Replicated |
|---|---|---|---|---|
| Bugzilla | 0.6503 | 0.6850 | 0.6312 | 0.6722 |
| Columba | 0.5783 | 0.6065 | 0.5637 | 0.6050 |
| JDT | 0.3871 | 0.4194 | 0.3874 | 0.4125 |
| Mozilla | 0.2300 | 0.2625 | 0.2478 | 0.2561 |
| Platform | 0.4080 | 0.4471 | 0.4143 | 0.4381 |
| PostgreSQL | 0.5647 | 0.6052 | 0.5666 | 0.5958 |
| Average | 0.4697 | 0.5043 | 0.4685 | 0.4966 |



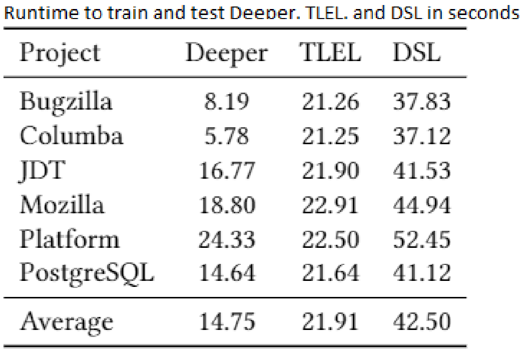
Individual contribution of base learners in first layer and cumulative contribution of stacked layers in DSL in terms of F1 score

| Project | Logistic Regression | K-Nearest Neighbors | Random Forest | Extremely Randomized Trees | XGBoost | First Stack | Total Stacked Layers | DSL |
|---|---|---|---|---|---|---|---|---|
| Bugzilla | 0.6037 | 0.5706 | 0.6502 | 0.6451 | 0.6718 | 0.6648 | 4 | 0.6730 |
| Columba | 0.5942 | 0.5585 | 0.5863 | 0.6066 | 0.5856 | 0.6084 | 3 | 0.6090 |
| JDT | 0.3353 | 0.3664 | 0.4232 | 0.4224 | 0.4164 | 0.4228 | 5 | 0.4233 |
| Mozilla | 0.1843 | 0.2034 | 0.2529 | 0.2461 | 0.2412 | 0.2555 | 3 | 0.2582 |
| Platform | 0.3270 | 0.3633 | 0.4330 | 0.4150 | 0.4402 | 0.4400 | 4 | 0.4425 |
| PostgreSQL | 0.5223 | 0.5311 | 0.5884 | 0.5774 | 0.5739 | 0.5935 | 3 | 0.5994 |
| Average | 0.4278 | 0.4322 | 0.4890 | 0.4854 | 0.4882 | 0.4975 | | 0.5009 |

## RQ4 What is the effect of varying the amount of training data on their effectiveness?



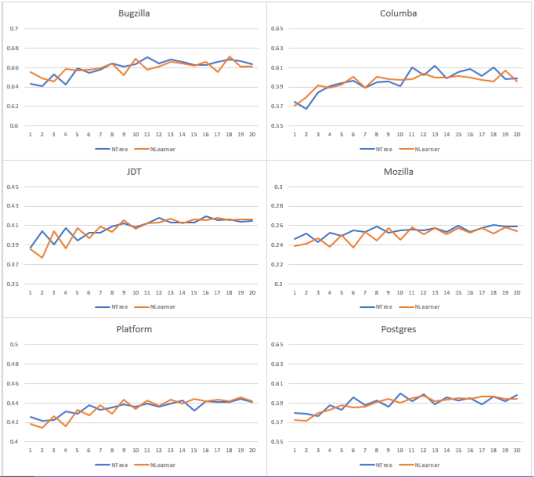
## RQ5 How much time does it take for the methodologies to run?



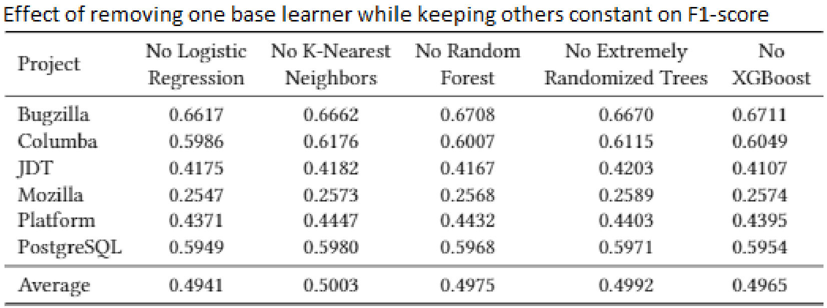
Runtime to train and test Deeper, TLEL, and DSL in seconds

| Project | Deeper | TLEL | DSL |
|---|---|---|---|
| Bugzilla | 8.19 | 21.26 | 37.83 |
| Columba | 5.78 | 21.25 | 37.12 |
| JDT | 16.77 | 21.90 | 41.53 |
| Mozilla | 18.80 | 22.91 | 44.94 |
| Platform | 24.33 | 22.50 | 52.45 |
| PostgreSQL | 14.64 | 21.64 | 41.12 |
| Average | 14.75 | 21.91 | 42.50 |

## RQ6 What is the effect of varying the parameter settings?



## *RQ6 What is the effect of varying the parameter settings? (continued)*



Effect of removing one base learner while keeping others constant on F1-score

| Project | No Logistic Regression | No K-Nearest Neighbors | No Random Forest | No Extremely Randomized Trees | No XGBoost |
|---|---|---|---|---|---|
| Bugzilla | 0.6617 | 0.6662 | 0.6708 | 0.6670 | 0.6711 |
| Columba | 0.5986 | 0.6176 | 0.6007 | 0.6115 | 0.6049 |
| JDT | 0.4175 | 0.4182 | 0.4167 | 0.4203 | 0.4107 |
| Mozilla | 0.2547 | 0.2573 | 0.2568 | 0.2589 | 0.2574 |
| Platform | 0.4371 | 0.4447 | 0.4432 | 0.4403 | 0.4395 |
| PostgreSQL | 0.5949 | 0.5980 | 0.5968 | 0.5971 | 0.5954 |
| Average | 0.4941 | 0.5003 | 0.4975 | 0.4992 | 0.4965 |

## Conclusion

- We have conducted an empirical validation of the hybrid ensemble methodologies to improve performance in defect prediction relative to previously tested methodologies
- We analyze and highlight conclusions about using multiple base learners, optimized weightings, and additional layers with respect to their relationship to classification performance on the tested datasets
- We present a new technique with strong generalization abilities that can be used in future research
- Replication of just-in-time defect prediction approaches has enabled us to confirm the external validity of our new prediction approach, which makes us more confident that Deep Super Learner is competitive with the best alternatives for just-in-time defect prediction.

## Conclusion

- We have conducted an empirical validation of the hybrid ensemble methodologies to improve performance in defect prediction relative to previously tested methodologies
- We analyze and highlight conclusions about using multiple base learners, optimized weightings, and additional layers with respect to their relationship to classification performance on the tested datasets
- We present a new technique with strong generalization abilities that can be used in future research
- Replication of just-in-time defect prediction approaches has enabled us to confirm the external validity of our new prediction approach, which makes us more confident that Deep Super Learner is competitive with the best alternatives for just-in-time defect prediction.

## Conclusion

- We have conducted an empirical validation of the hybrid ensemble methodologies to improve performance in defect prediction relative to previously tested methodologies
- We analyze and highlight conclusions about using multiple base learners, optimized weightings, and additional layers with respect to their relationship to classification performance on the tested datasets
- We present a new technique with strong generalization abilities that can be used in future research
- Replication of just-in-time defect prediction approaches has enabled us to confirm the external validity of our new prediction approach, which makes us more confident that Deep Super Learner is competitive with the best alternatives for just-in-time defect prediction.

## Conclusion

- We have conducted an empirical validation of the hybrid ensemble methodologies to improve performance in defect prediction relative to previously tested methodologies
- We analyze and highlight conclusions about using multiple base learners, optimized weightings, and additional layers with respect to their relationship to classification performance on the tested datasets
- We present a new technique with strong generalization abilities that can be used in future research
- Replication of just-in-time defect prediction approaches has enabled us to confirm the external validity of our new prediction approach, which makes us more confident that Deep Super Learner is competitive with the best alternatives for just-in-time defect prediction.

## Thank you!

*Background:* Just-in-time defect prediction can be used to efficiently allocate resources and manage project schedules in the software testing and debugging process.

*Aim:* To apply a set of guidelines for reporting replication experiments and to analyze our findings for drawing cross study conclusions, including extending the study by applying a novel ensemble technique.

*Method:* We present the original study's proposed approach and the Deep Super Learner ensemble. The datasets and performance measures for evaluation are also described.

*Results:* Experimental results of the replication are consistent with the original study. The Deep Super Learner achieves statistically significantly better results than the original approach on five of the six projects in predicting defects as measured by $F_1$ *score*.

*Conclusion:* Although this study is a replication, we present a new technique with strong generalization abilities that can be used in future research.

Thank you!

Email: steven.young@ryerson.ca
tamer.abdou@ryerson.ca
ayse.bener@ryerson.ca

Please visit www.datasciencelab.ca