# Evaluating the Adaptive Selection of Classifiers for Cross-Project Bug Prediction


Dario Di Nucci

Fabio Palomba

Andrea De Lucia



RAISE'18

6th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering


Developers usually introduce bugs

# Software failures can compromise business

## RBS could take until weekend to make 600,000 missing payments after glitch

**Delay in payments – including tax credits and disability living allowance – is latest technology failure to affect UK bank's customers**


▲ Royal Bank of Scotland was fined £56m in 2012 after an IT meltdown left more than 6 million customers unable to use their accounts. Photograph: Yui Mok/PA

https://www.theguardian.com/business/2015/jun/17/rbs-fails-to-make-600000-payments-customers-it-technology-failure-bank

# Software failures can compromise safety

## US aviation authority: Boeing 787 bug could cause 'loss of control'

**More trouble for Dreamliner as Federal Aviation Administration warns glitch in control unit causes generators to shut down if left powered on for 248 days**


▲ The Boeing 787 has four generator-control units that, if powered on at the same, could fail simultaneously and cause a complete electrical shutdown. Photograph: Elaine Thompson/AP

https://www.theguardian.com/business/2015/may/01/us-aviation-authority-boeing-787-dreamliner-bug-could-cause-loss-of-control

Large software systems contain thousands of bugs!

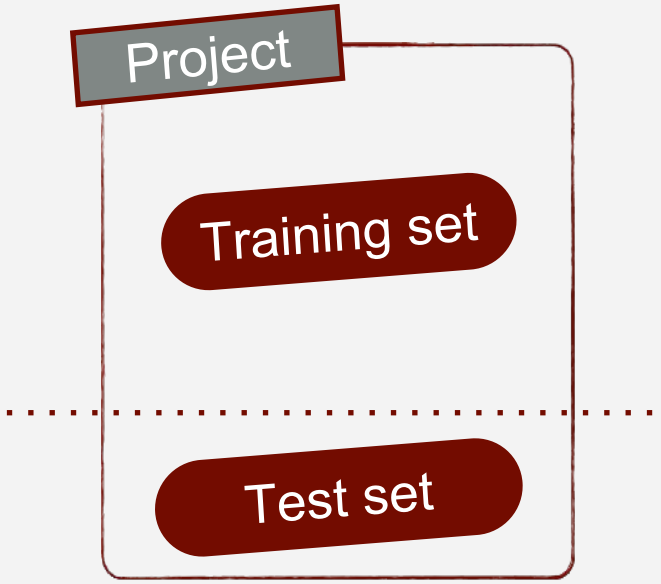
but resources are limited...

Thus, we should predict which components are more prone to be buggy!

# Machine Learning features for Bug Prediction

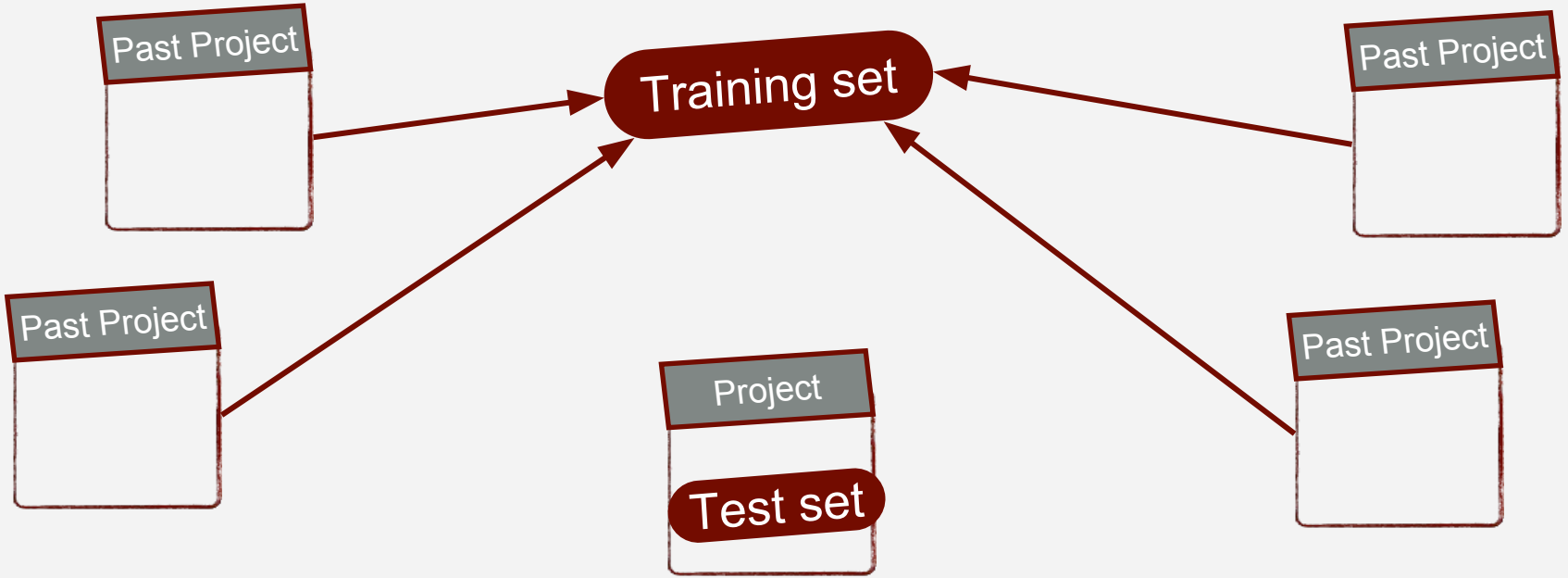
| | |
|---|---|
| Structural Metrics | CK metrics, LOC, etc... |
| Process Metrics | # Changes, # Developers, Ownership, etc... |
| Developers Metrics | Scattering, Micro-Interactions, etc... |

# Training a Machine Learner for Bug Prediction



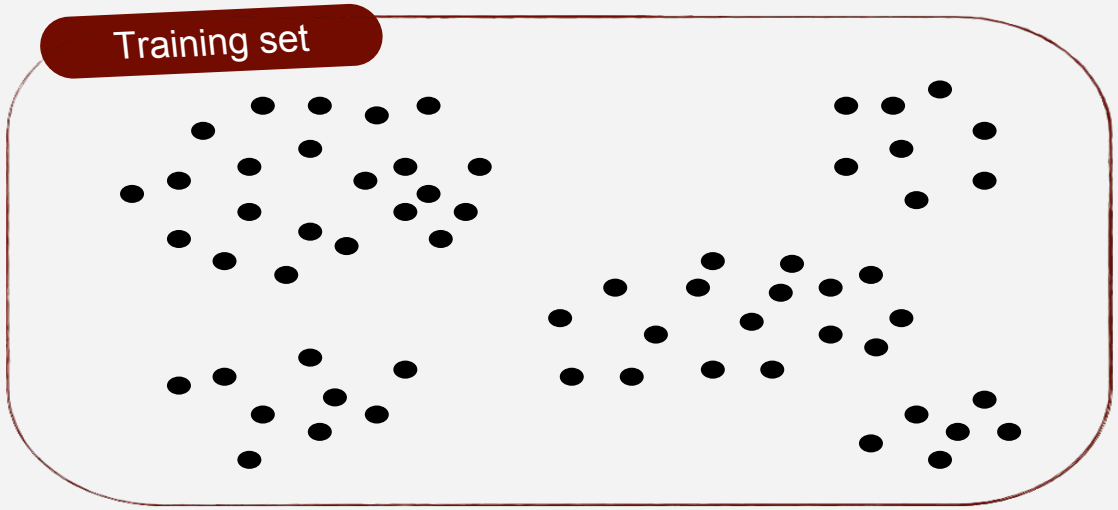

# Within-Project

# Training a Machine Learner for Bug Prediction




# Cross-Project

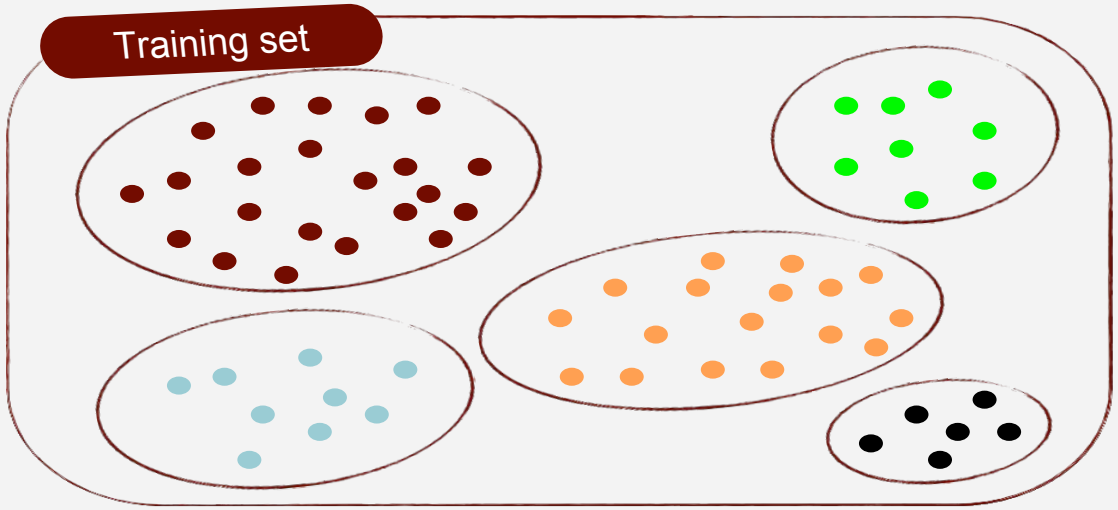
# Training a Machine Learner for Bug Prediction




T. Menzies, A. Butcher, D. Cok, A. Marcus, L. Layman, F. Shull, B. Turhan, and T. Zimmermann
"Local versus global lessons for defect prediction and effort estimation"
*IEEE Transactions on Software Engineering*

# Local-Project

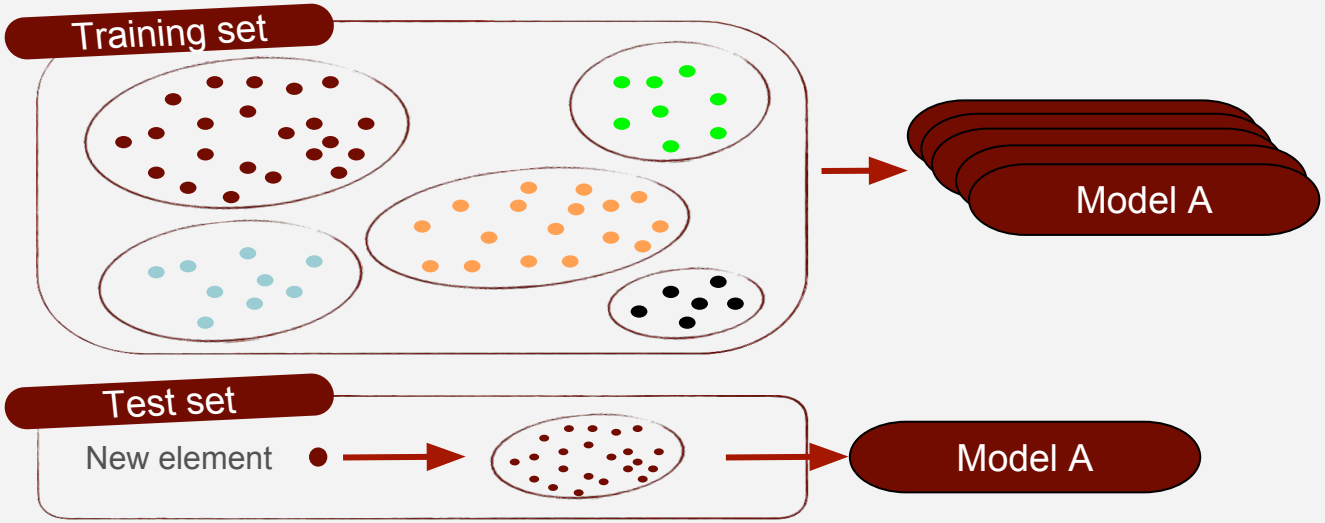
# Training a Machine Learner for Bug Prediction




T. Menzies, A. Butcher, D. Cok, A. Marcus, L. Layman, F. Shull, B. Turhan, and T. Zimmermann
"Local versus global lessons for defect prediction and effort estimation"
*IEEE Transactions on Software Engineering*

# Local-Project

# Training a Machine Learner for Bug Prediction




T. Menzies, A. Butcher, D. Cok, A. Marcus, L. Layman, F. Shull, B. Turhan, T. Zimmermann
"Local versus global lessons for defect prediction and effort estimation"
*IEEE Transactions on software engineering*

# Local-Project

# Selecting a Machine Learner for Bug Prediction

# Selecting a Machine Learner for Bug Prediction

## Revisiting the Impact of Classification Techniques on the Performance of Defect Prediction Models

Baljinder Ghotra, Shane McIntosh, Ahmed E. Hassan
Software Analysis and Intelligence Lab (SAIL)
School of Computing, Queen's University, Canada
{ghotra, mcintosh, ahmed}@cs.queensu.ca

***Abstract*—Defect prediction models help software quality assurance teams to effectively allocate their limited resources to the most defect-prone software modules. A variety of classification techniques have been used to build defect prediction models ranging from simple (e.g., logistic regression) to advanced techniques (e.g., Multivariate Adaptive Regression Splines (MARS)). Surprisingly, recent research on the NASA dataset suggests that the performance of a defect prediction model is not significantly impacted by the classification technique that is used to train it. However, the dataset that is used in the prior study is both: (a) noisy, i.e., contains erroneous entries and (b) biased, i.e., only contains software developed in one setting. Hence, we set out to replicate this prior study in two experimental settings. First, we apply the replicated procedure to the same (known-to-be noisy) NASA dataset, where we derive similar results to the prior study, i.e., the impact that classification techniques have appear to be minimal. Next, we apply the replicated procedure to two new datasets: (a) the cleaned version of the NASA dataset and (b) the PROMISE dataset, which contains open source software developed in a variety of settings (e.g., Apache, GNU). The results in these new datasets show a clear, statistically distinct separation of groups of techniques, i.e., the choice of classification technique has an impact on the performance of defect prediction models. Indeed, contrary to earlier research, our results suggest that some classification techniques tend to produce defect prediction models that outperform others.**

### I. Introduction

A disproportionate amount of the cost of developing software is spent on maintenance [18]. Fixing defects is a central software maintenance activity. However, before defects can be fixed, they must be detected. Software Quality Assurance (SQA) teams are dedicated to this task of defect detection.

Defect prediction models can be used to assist SQA teams with defect detection. Broadly speaking, defect prediction models are trained using software metrics (e.g., size and complexity metrics) to predict whether software modules will be defective or not in the future. Knowing which software modules are likely to be defect-prone before a system has been deployed allows practitioners to more efficiently and effectively allocate SQA effort.

To perform defect prediction studies [21, 58], researchers have explored the use of various classification techniques to train defect prediction models [12, 34]. For example, in early studies, researchers used simple techniques like logistic regression [2, 7, 47] and linear regression [24, 65] to train defect prediction models. In more recent work, researchers have used more advanced techniques like Multivariate Adaptive Regression Splines (MARS) [5, 27], Personalized Change Classification (PCC) [27], and Logistic Model Trees (LMT) [51, 53]. Ensemble methods that combine different machine learning techniques have also been explored [14, 61]. Moreover, researchers have started to develop context-sensitive techniques that are aware of the peculiarities of software defects [8, 35].

Despite recent advances in machine learning, studies suggest that the classification technique used to train defect prediction models has little impact on its performance [34, 38, 56]. For example, Lessmann *et al.* [34] conducted a study comparing the performance of 22 different classification techniques on the NASA corpus. Their results show that the performance of 17 of the 22 classification techniques are statistically indistinguishable from each other.

However, the NASA corpus that was used in the prior work is noisy and biased. Indeed, Shepperd *et al.* [57] found that the original NASA corpus contains several erroneous and implausible entries. This noise in the studied corpus may have led prior studies to draw incorrect conclusions. Furthermore, the studied corpus only contains proprietary software developed within one organization. Software developed in a different setting (e.g., open source) may lead to different conclusions.

Therefore, we set out to revisit the findings of Lessmann *et al.* [34] both in the original, known-to-be noisy setting (Section IV), and in two additional settings (Section V). We find that we can indeed replicate the initial results of Lessmann *et al.* [34] when the procedure is reapplied to the known-to-be noisy NASA corpus. On the other hand, the results of our experiment in two additional settings seem to contradict Lessmann *et al.*'s early results. We structure our extended replication by addressing the following two research questions:

**RQ1 Do the defect prediction models of different classification techniques still perform similarly when trained using the cleaned NASA corpus?**
No, we find a clear separation of classification techniques when models are trained using the cleaned NASA corpus [57]. The Scott-Knott statistical test [26] groups the classification techniques into four statistically distinct ranks. Our results show that models trained using LMT and statistical techniques like Simple Logistic tend to outperform models trained using clustering, rule-based, Support Vector Machines (SVM), nearest neighbour, and neural network techniques.

The choice of the machine learner strongly impacts performance!

Performance can increase or decrease up to 30% depending on the used classifier.

# Selecting a Machine Learner for Bug Prediction

Cross-Project Defect Prediction Models:
L'Union Fait la Force

Annibale Panichella[1], Rocco Oliveto[2], Andrea De Lucia[1]
[1]Department of Management & Information Technology, University of Salerno, Fisciano (SA), Italy
[2]Department of Bioscience and Territory, University of Molise, Pesche (IS), Italy
apanichella@unisa.it, rocco.oliveto@unimol.it, adelucia@unisa.it

***Abstract*—Existing defect prediction models use product or process metrics and machine learning methods to identify defect-prone source code entities. Different classifiers (*e.g.*, linear regression, logistic regression, or classification trees) have been investigated in the last decade. The results achieved so far are sometimes contrasting and do not show a clear winner. In this paper we present an empirical study aiming at statistically analyzing the equivalence of different defect predictors. We also propose a combined approach, coined as CODEP (COmbined DEfect Predictor), that employs the classification provided by different machine learning techniques to improve the detection of defect-prone entities. The study was conducted on 10 open source software systems and in the context of cross-project defect prediction, that represents one of the main challenges in the defect prediction field. The statistical analysis of the results indicates that the investigated classifiers are not equivalent and they can complement each other. This is also confirmed by the superior prediction accuracy achieved by CODEP when compared to stand-alone defect predictors.**

## I. Introduction

During software development and maintenance, software engineers need to manage time and resources. Improper management of these factors could lead to project failure. Given its importance, in recent years a lot of effort has been devoted to provide software engineers with tools supporting management activities. Defect prediction is one of these tools. Knowing the parts of a software system that are more defect-prone may aid in scheduling testing or refactoring activities. In fact, it is reasonable to allocate more resources on more critical and defect-prone source code entities.

Existing prediction models use different information (*i.e.*, source code metrics, process metrics, or past defects) and machine learning methods to identify defect-prone source code entities. A plethora of classifiers (*e.g.*, linear regression, logistic regression, or classification trees) have been proposed and empirically investigated in the last decade [1], [2], [3]. Generally, the evaluation of these classifiers is based only on metrics—such as precision and recall—able to capture the prediction accuracy of the experimented approaches. The results achieved are sometimes contrasting and generally do not allow to identify a method that sensibly overcomes the others (see *e.g.*, [1]). Indeed, looking at the prediction accuracy (captured by metrics), the impression is that all the classifiers exploited so far are able to capture the same information when used to detect software entities having a high likelihood to exhibit faults. Thus, selecting a specific classifier does not seem to play an important role [4], [1].

However, metrics do not tell the whole story. They only provide an overview of the prediction accuracy without emphasizing peculiarities (if any) of each investigated method. What is missing is a thorough analysis of the predictions of each approach aiming at verifying whether or not different classifiers are able to identify different defect-prone entities. In other words, while different classifiers achieve the same accuracy, the following question remains still unanswered:

*Do different classifiers identify the same defect-prone entities or different entities with the same defect-proneness?*

In this paper we try to bridge this gap. We conjecture that the investigated classifiers are not equivalent even if they provide the same prediction accuracy. In other words, we believe that different classifiers could be able to identify different defect-prone entities. This calls for a combined approach that employs different (and complementary) classifiers aiming at improving the prediction accuracy of stand-alone approaches.

To verify our conjecture we conducted an empirical study aiming at statistically analyzing the equivalence of different classifiers. The study is based on Principal Component Analysis (PCA) and overlap metrics. With such analyses we are able to identify classifiers that complement each others, *i.e.*, they identify different sets of defect-prone entities. The experimentation was conducted on 10 open-source software systems from the Promise repository. In the context of our study we investigated six different machine learning techniques used for defect prediction in many previous works [5], [6], [7], [8], [9], [2]. These techniques belong to four different categories: regression functions, neural networks, decision trees, and rules models. All these approaches were experimented in the context of cross-project defect prediction, that represents one of the main challenges in the defect prediction field [10].

The achieved results confirmed our conjecture. *The investigated classifiers are not equivalent despite the similar overall prediction accuracy. This means that they are able to identify different sets of defect-prone entities.* This is confirmed by our statistical analysis but especially by the superior performances of a novel combined approach, coined as CODEP (**CO**mbined **DE**fect **P**redictor), that uses machine learning techniques to combine different and complementary classifiers.

In summary, the contributions of this paper are:

Classifiers are highly complementary!

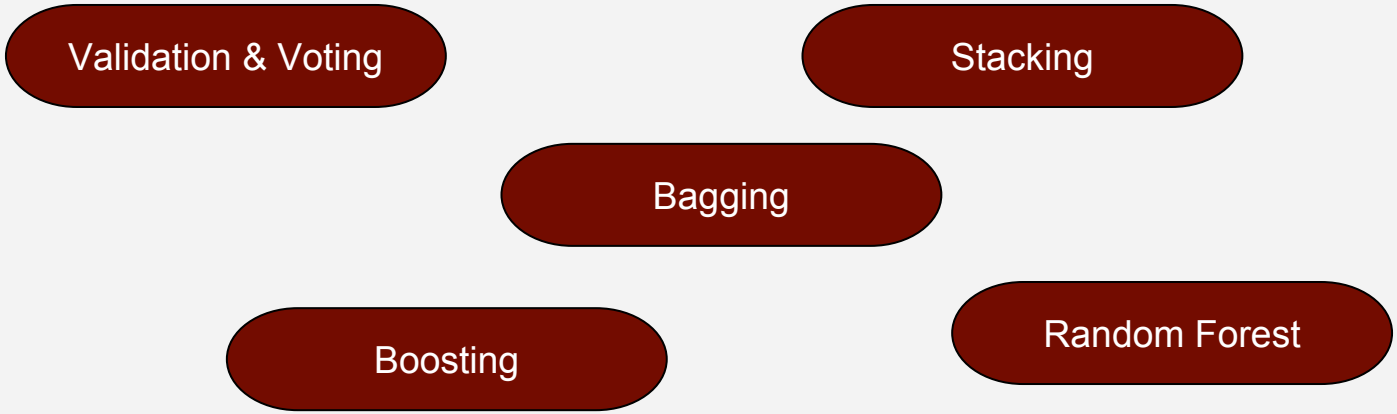
Different classifiers capture different sets of buggy components!

# Ensemble of Classifiers

**Ensemble of classifiers.** Learning algorithms that construct a set of classifiers and then classify new data points by taking a weighted vote of their predictions.

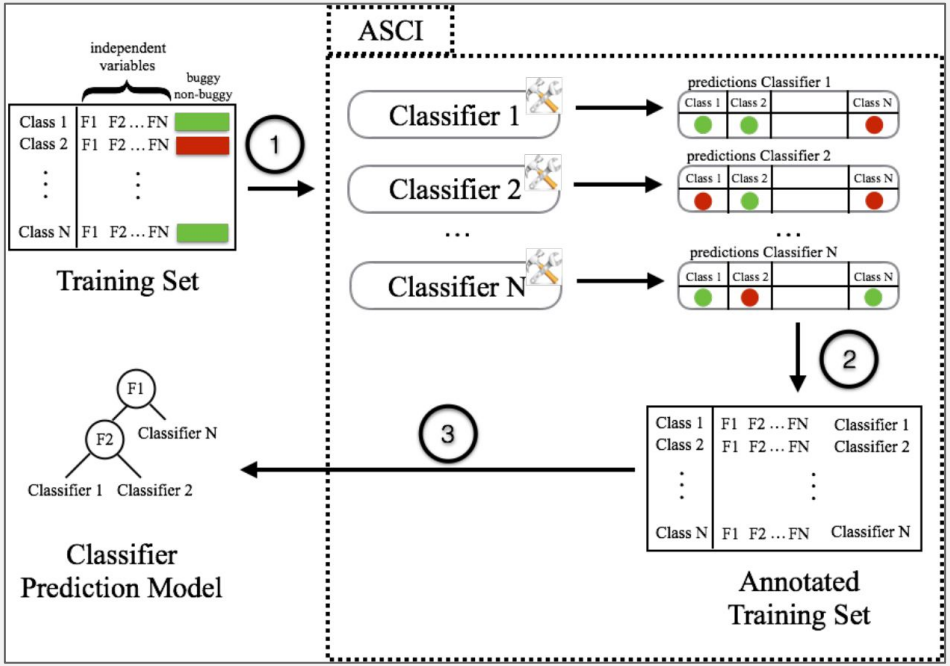
***Thomas G. Dietterich***

# Ensemble of Classifiers




# Which Technique?

# ASCI: **A**daptive **S**election of **C**lass**I**fiers

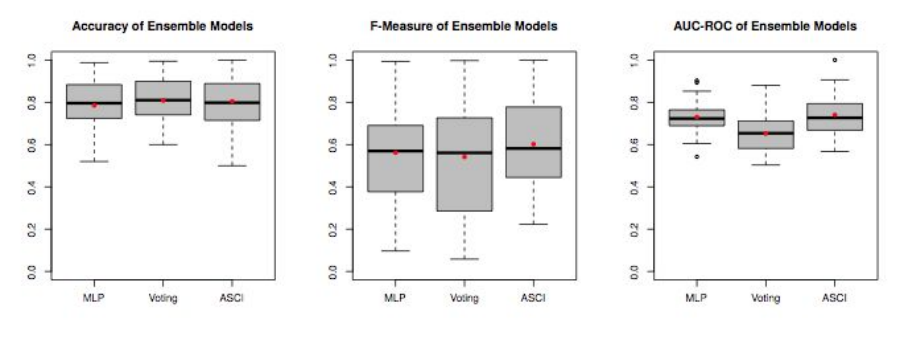



D. Di Nucci, F. Palomba, R. Oliveto, and A. De Lucia
"Dynamic selection of classifiers in bug prediction: An adaptive method"
*IEEE Transactions on Emerging Topics in Computational Intelligence*

# ASCI: Adaptive Selection of Classlfiers


30 Software Systems

Within-Project




D. Di Nucci, F. Palomba, R. Oliveto, and A. De Lucia
"Dynamic selection of classifiers in bug prediction: An adaptive method
*IEEE Transactions on Emerging Topics in Computational Intelligence*

# Evaluating the Adaptive Selection of Classifiers for Cross-Project Bug Prediction

# Experimented Systems

10 Software Systems

i) Less than 50% of Buggy Classes

ii) Randomly Selected


C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto
"Automated parameter optimization of classification techniques for defect prediction models,"
*International Conference on Software Engineering 2016*

*How does **ASCI** work in the context of **cross-project** bug prediction when compared to existing ensemble techniques?*

# RQ1

*To what extent can **local learning** improve the performance of **ASCI**?*

# Pre-processing steps

Data Cleaning

M. Shepperd, Q. Song, Z. Sun, and C. Mair
“Data quality: Some comments on the NASA software defect datasets”
*IEEE Transactions on Software Engineering*

# Pre-processing steps

Data Cleaning

Data Normalization

J. Nam, S. J. Pan, and S. Kim
“Transfer defect learning”
*International Conference on Software Engineering 2013*

# Pre-processing steps

Data Cleaning

Data Normalization

Feature Selection
Correlation-based Feature Selection

T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell
“A systematic literature review on fault prediction performance in software engineering”
*IEEE Transactions on Software Engineering*

# Pre-processing steps

Data Cleaning

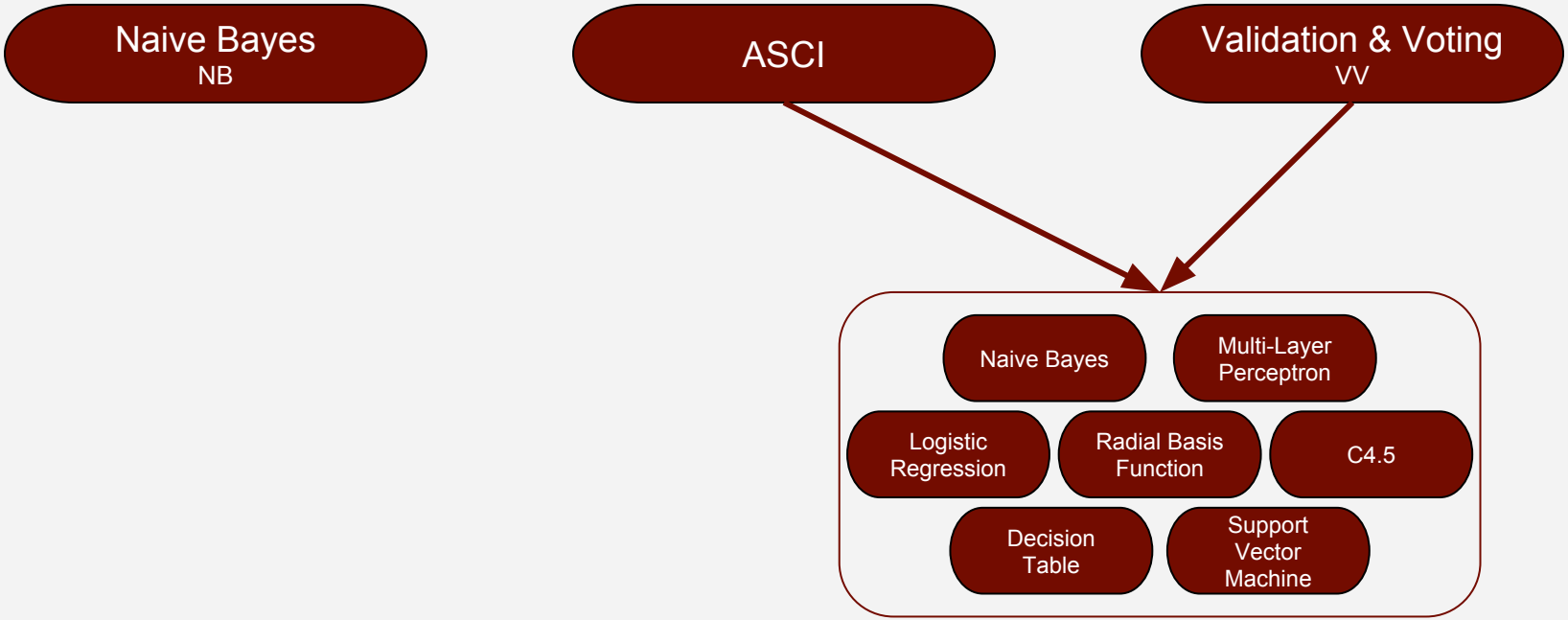
Data Normalization

Feature Selection
Correlation-based Feature Selection

Data Balancing
Synthetic Minority Over-sampling TEchnique

K. E. Bennin, J. Keung, P. Phannachitta, A. Monden, and S. Mensah
“Mahakil: Diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction”
*IEEE Transactions on Software Engineering*

# Experimented Models



# Validation Metrics

Accuracy

| | |
|---|---|
| TP | FP |
| FN | TN |

# Validation Metrics

Accuracy

F-Measure

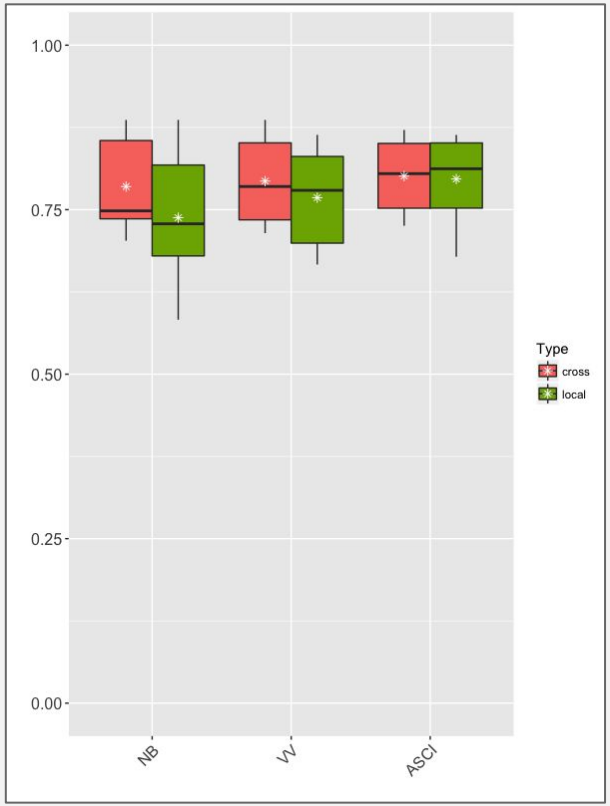
| | |
|---|---|
| TP | FP |
| FN | TN |

# Validation Metrics

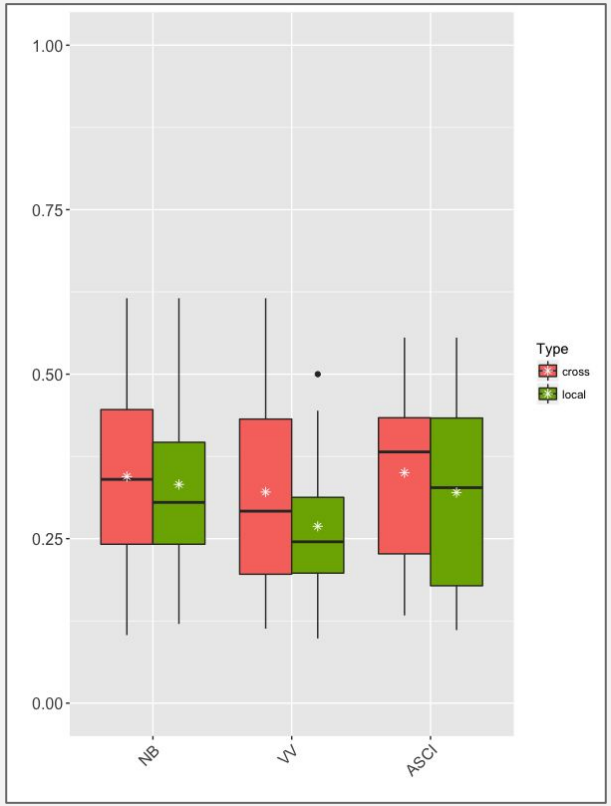
- Accuracy
- F-Measure
- Area Under Curve

| TP | FP |
| --- | --- |
| FN | TN |

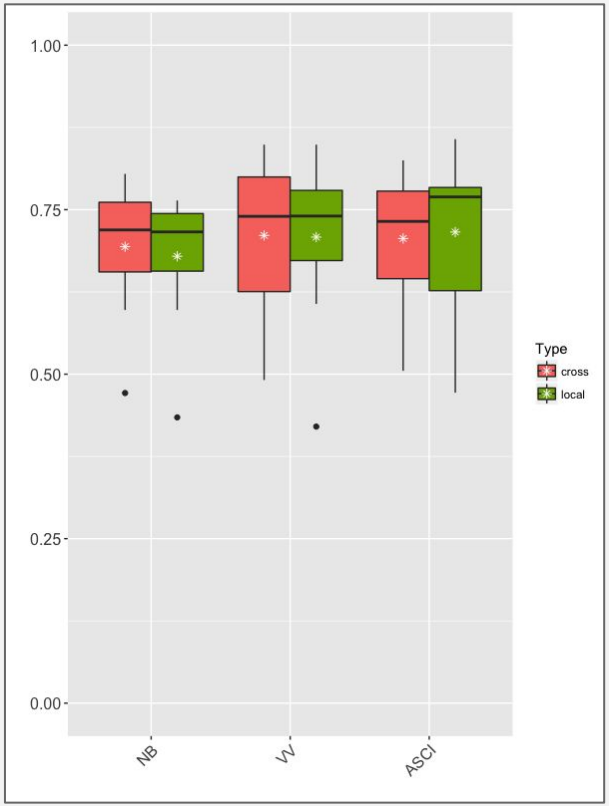
# Results




Accuracy

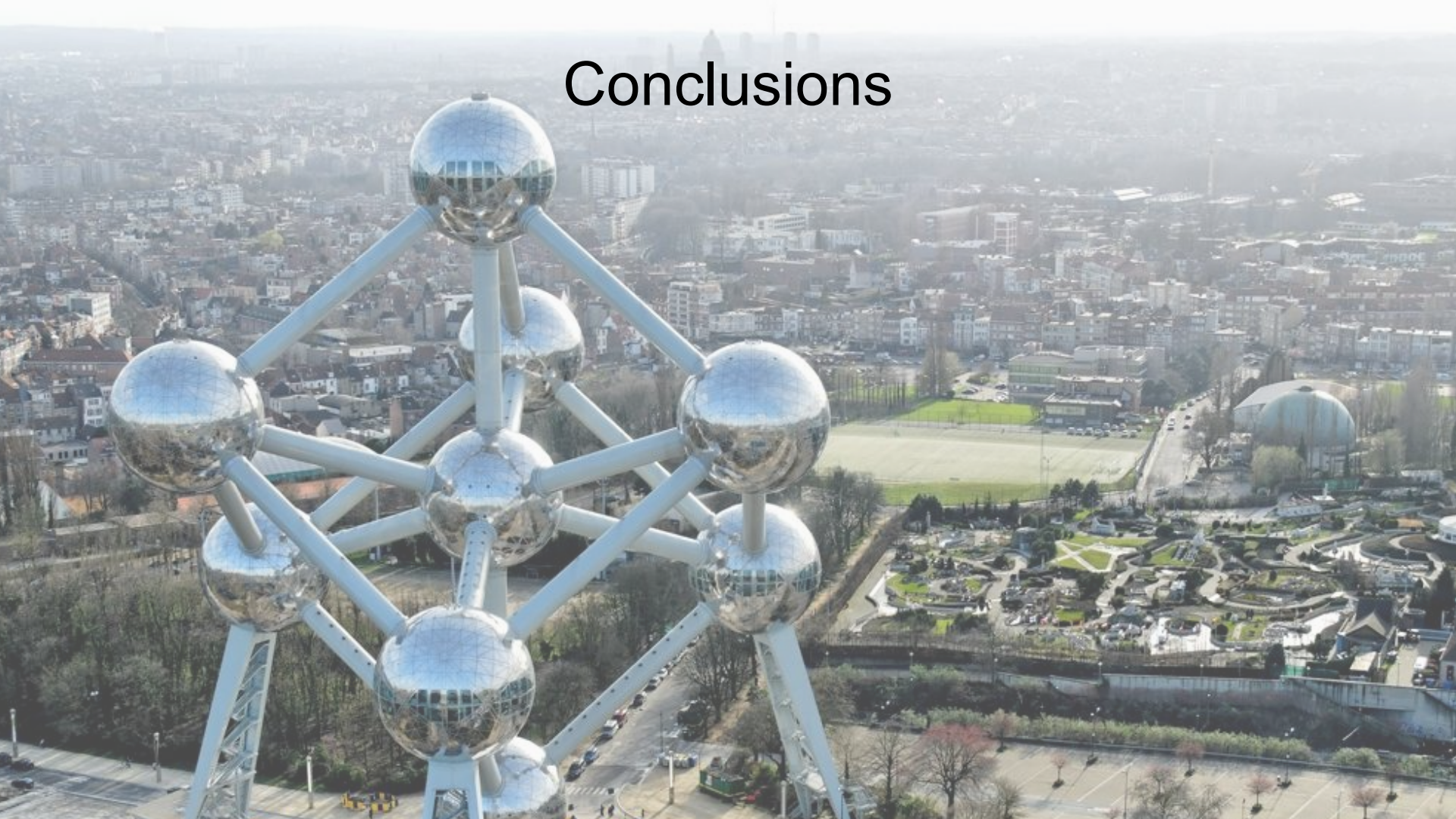



F-Measure

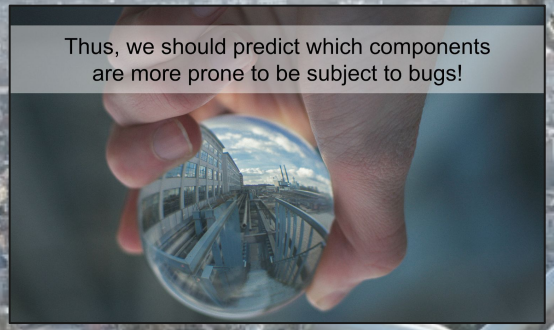



AUC-ROC

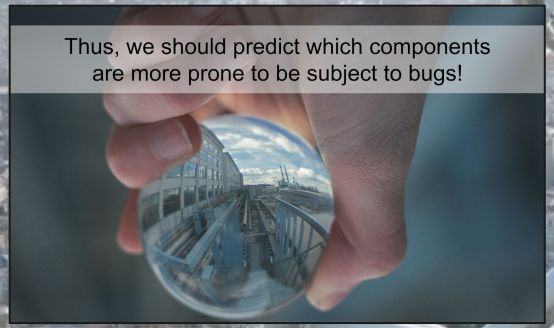
# Conclusions

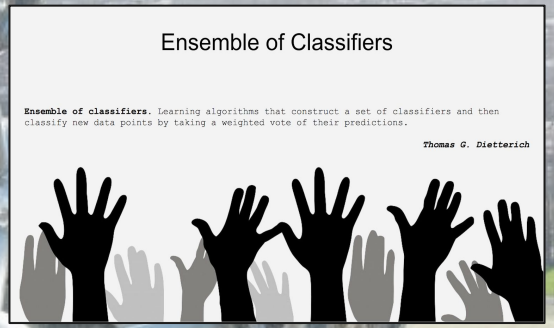


# Conclusions

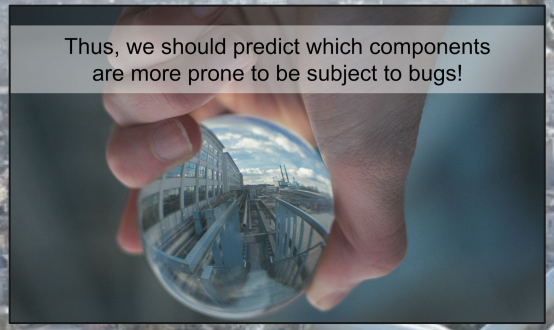


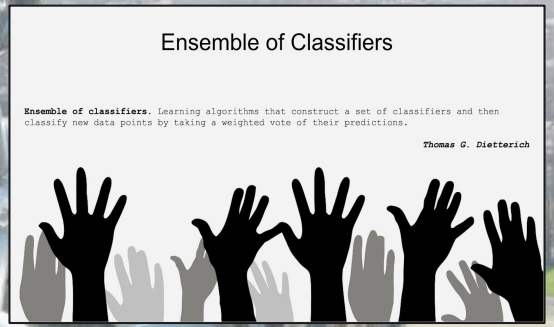
# Conclusions

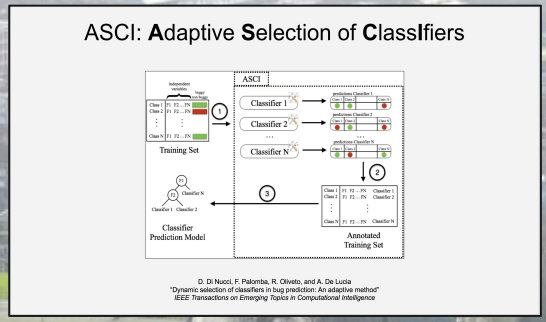


# Conclusions

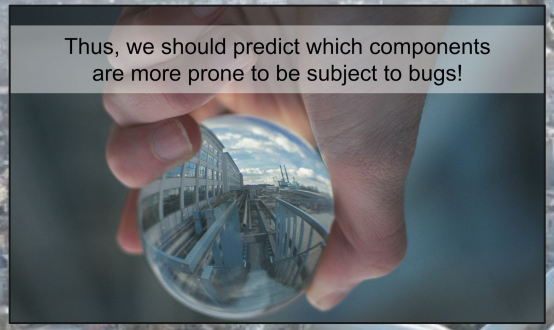


# Conclusions

Thus, we should predict which components are more prone to be subject to bugs!

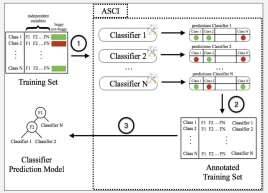


## Ensemble of Classifiers

**Ensemble of classifiers.** Learning algorithms that construct a set of classifiers and then classify new data points by taking a weighted vote of their predictions.

*Thomas G. Dietterich*

## ASCI: **A**daptive **S**election of **C**lass**I**fiers




D. Di Nucci, F. Palomba, R. Oliveto, and A. De Lucia
"Dynamic selection of classifiers in bug prediction: An adaptive method"
*IEEE Transactions on Emerging Topics in Computational Intelligence*

## Experimented Systems

10 Software Systems

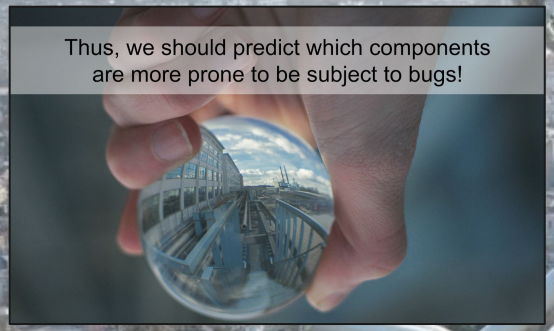
i) Randomly Selected

ii) Less than 50% of Buggy Classes

C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto
"Automated parameter optimization of classification techniques for defect prediction models,"
*International Conference on Software Engineering 2016*

# Conclusions

Thus, we should predict which components are more prone to be subject to bugs!

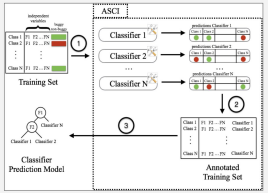


## Ensemble of Classifiers

**Ensemble of classifiers.** Learning algorithms that construct a set of classifiers and then classify new data points by taking a weighted vote of their predictions.

*Thomas G. Dietterich*

## ASCI: **A**daptive **S**election of **C**lass**I**fiers




D. Di Nucci, F. Palomba, R. Oliveto, and A. De Lucia
"Dynamic selection of classifiers in bug prediction: An adaptive method"
*IEEE Transactions on Emerging Topics in Computational Intelligence*

## Experimented Systems

10 Software Systems

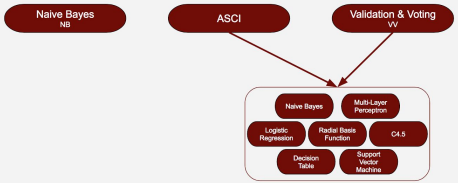
i) Randomly Selected

ii) Less than 50% of Buggy Classes

THE APACHE SOFTWARE FOUNDATION

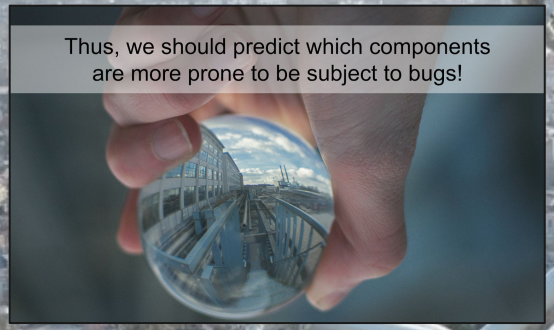
C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto
"Automated parameter optimization of classification techniques for defect prediction models,"
*International Conference on Software Engineering 2016*

## Experimented Models



# Conclusions

Thus, we should predict which components are more prone to be subject to bugs!

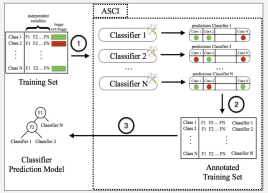


## Ensemble of Classifiers

**Ensemble of classifiers.** Learning algorithms that construct a set of classifiers and then classify new data points by taking a weighted vote of their predictions.

*Thomas G. Dietterich*

## ASCI: **A**daptive **S**election of **C**lass**I**fiers




D. Di Nucci, F. Palomba, R. Oliveto, and A. De Lucia
"Dynamic selection of classifiers in bug prediction: An adaptive method"
*IEEE Transactions on Emerging Topics in Computational Intelligence*

## Experimented Systems

10 Software Systems

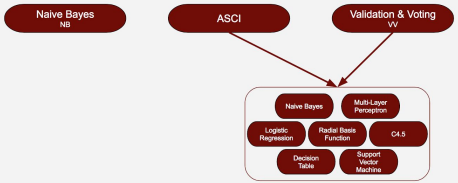
i) Randomly Selected

ii) Less than 50% of Buggy Classes

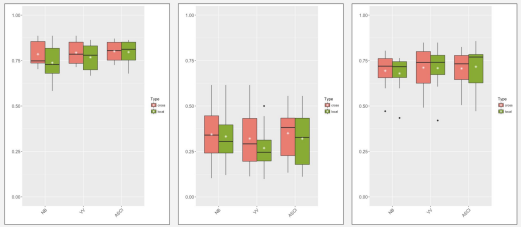
THE APACHE SOFTWARE FOUNDATION

C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto
"Automated parameter optimization of classification techniques for defect prediction models,"
*International Conference on Software Engineering 2016*

## Experimented Models




## Results



Accuracy F-Measure AUC-ROC