Machine Learning for Detecting Code Smells

Marcello M. Bonsangue

JCSSE 2022, Bangkok, 23 June 2022

Universiteit Leiden
The Netherlands

We are SCIENCE SINCE 1815

Discover the world at Leiden University

Universiteit Leiden

liacs Leiden Institute of Advanced Computer Science

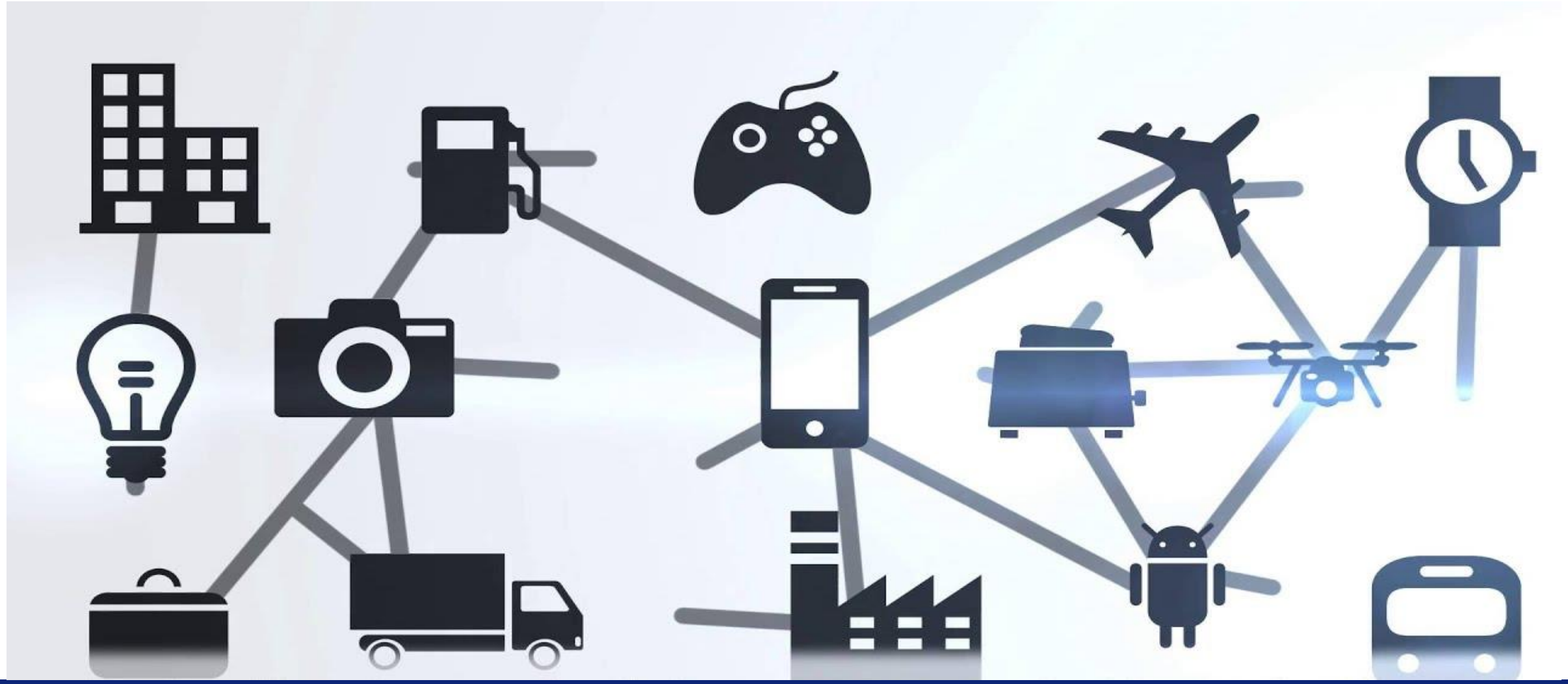
Jan van Rijn
M. Bonsangue

มหาวิทยาลัยขอนแก่น
KHON KAEN UNIVERSITY

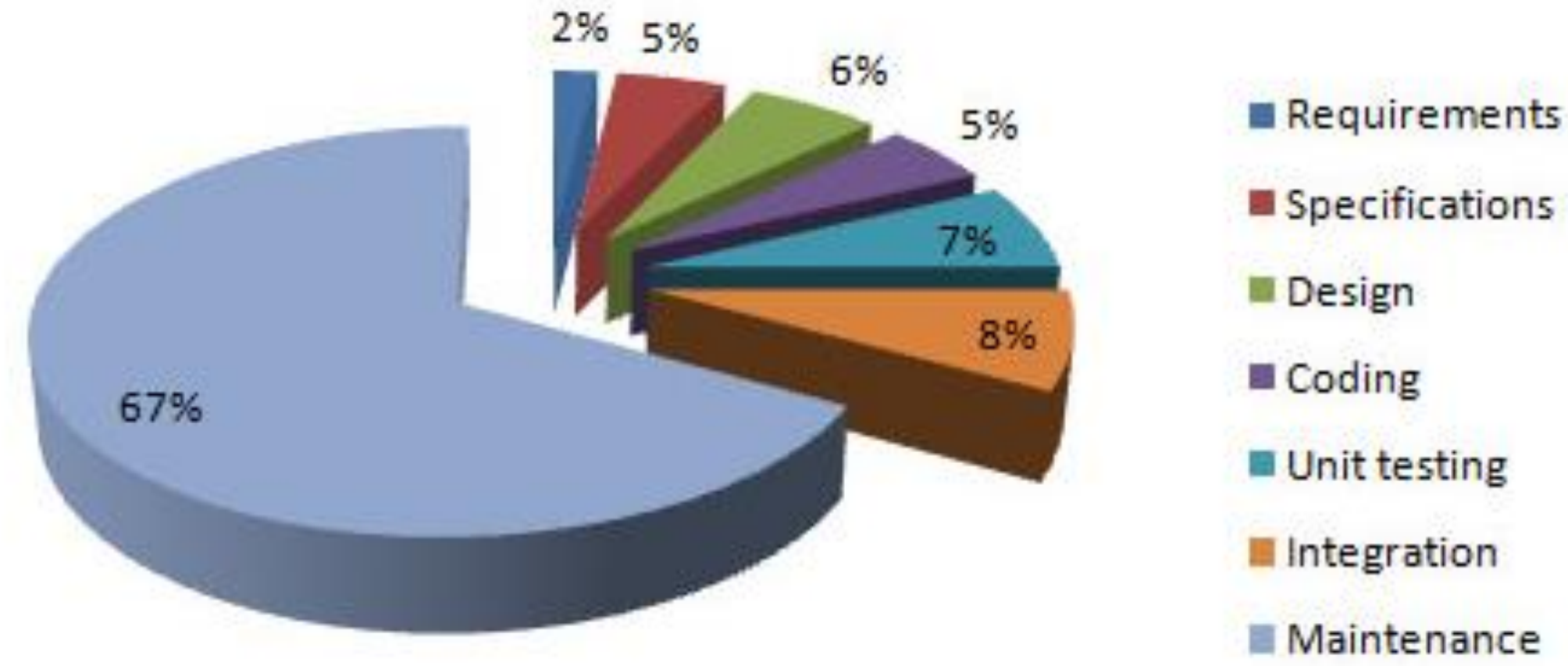
Chitsutha Soomlek

Software is everywhere today

It leads every our step. It is a part of everything we do



Software Life-Cycle Costs

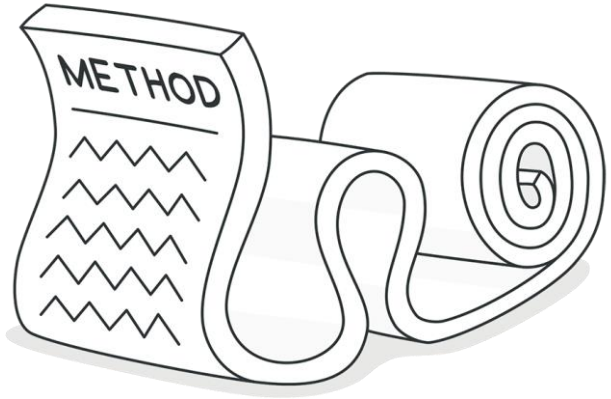


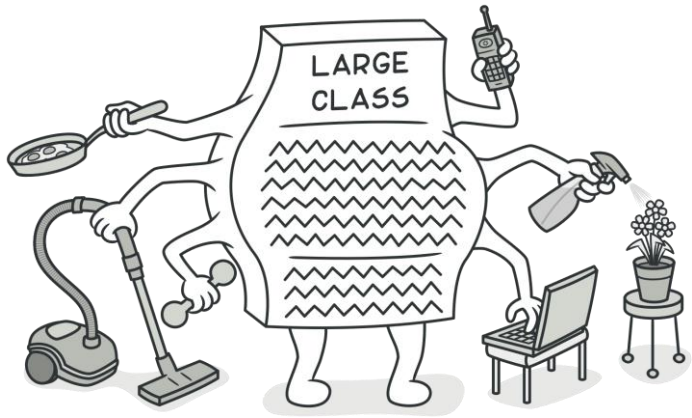
Source : Digital Aggregates

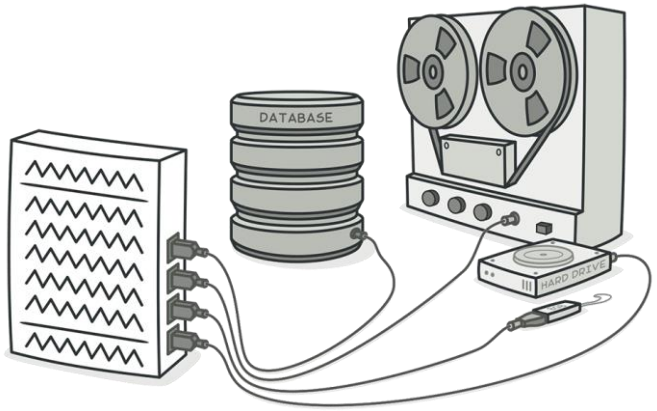
What is software maintenance?

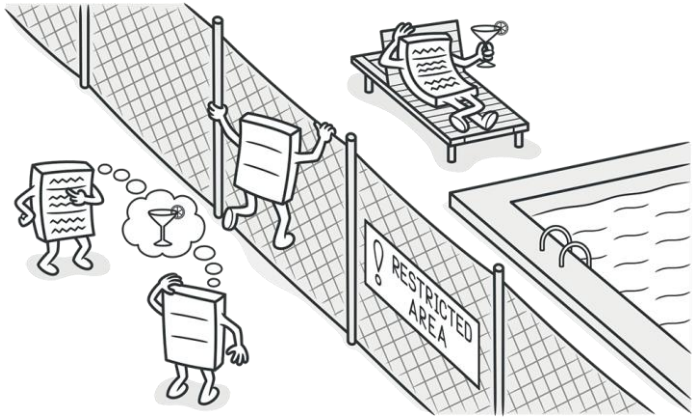
- repairing design and implementation faults
- adapting software to a different environment (hardware, OS)
- adding or modifying functionalities

Code Smell: violating design principles









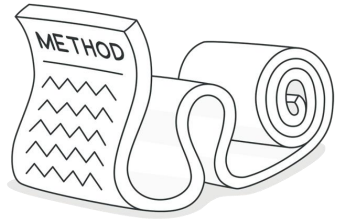
Figures from https://refactoring.guru/

What exactly is a Code Smell?

"A code smell is a surface indication that usually corresponds to a deeper problem in the system… they are often an indicator of a problem rather than the problem themselves"

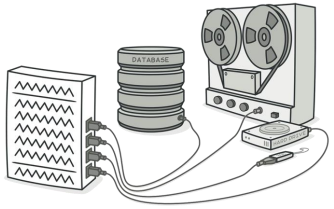
Kent Beck and Martin Fowler (1999

Code Smell: few examples



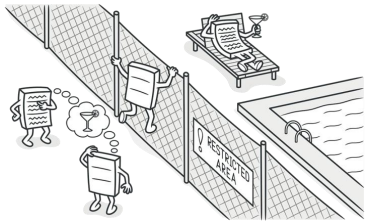
Long method

A method with too many LOC



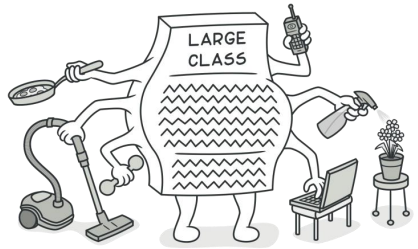
Data class

a class the contain almost only getters and setters



Feature Envy

A method that accesses the data of another object more than its own data



Blob

A class that tries to do too many things

Figures from https://refactoring.guru/

Refactoring can remove code smells

- Refactoring = *changing the structure of a software system without changing its observable behavior*

- Use refactoring to remove code smells:
  a) How to detect smells, either about code or design?
  b) After detecting these smells, which refactoring should be applied?
  c) What are the steps to apply these refactoring?
  d) What are the gains when applying these refactoring?

We need automated support

Why is code smell difficult to identify?

- There is no formal definition or standard
- Developers have different perceptions of code smells
- Commonly used approaches for code smells detection rely on a fixed set of metrics and corresponding threshold values
- There are reliability issues of the threshold values
- Programming languages have been evolving.

Existing methods to identify smell

- Metric-based = use fixed set of metrics and corresponding threshold
- Values/symptom-based = use rules to detect smells by symptoms
- Visualization-based = use graphical representations to fine smells
- Probabilistic/search-based = learn standard, search deviation

- Machine learning = use data from existing software projects

All lack of standardization/formalization of the smell definition

Tools, tools, tools

Bad Smell	Recall				Precision			
	inFusion	JDeodorant	PMD	JSpIRIT	inFusion	JDeodorant	PMD	JSpIRIT
Large Class	14%	14%	14%	14%	100%	9%	100%	50%
Long Method	33%	33%	50%	67%	100%	17%	100%	80%

Tools are often compared against human perception...
...but comparing to 2 or 20 software systems
is not very statistically relevant

Machine learning to identify smell

Can we mimic a developer's perception of a code smell?

How does machine learning perform when comparing to existing tools?

The MLCQ Dataset

Madeyski, L., Lewowski, T.: *MLCQ: industry-relevant code smell data set*. In: Proceedings of the Evaluation and Assessment in Software Engineering, pp. 342–347 (2020).

Code smells	No. of reviews	Positive			Negative	No. of code sample with multiple reviews
		Critical	Major	Minor	None	
Blob	4076	129	316	539	3092	1741
Data class	4078	146	401	510	3021	2522
Feature Envy	3337	24	142	288	2883	1030
Long method	3362	78	274	454	2556	1060

MLCQ alone is not enough

- To use machine learning for detecting code smells, we need to extract features metrics from each code
  - Line of codes of methods and classes
  - Number of method parameters
  - Number of attributes
  - … (20+ in total)

- How? MLCQ does not have this data…

- Extraction partly by PMD, but only present is smell is found

- Use of a different statistical and static analysis tools (like SciTools).

Rules vs. Data

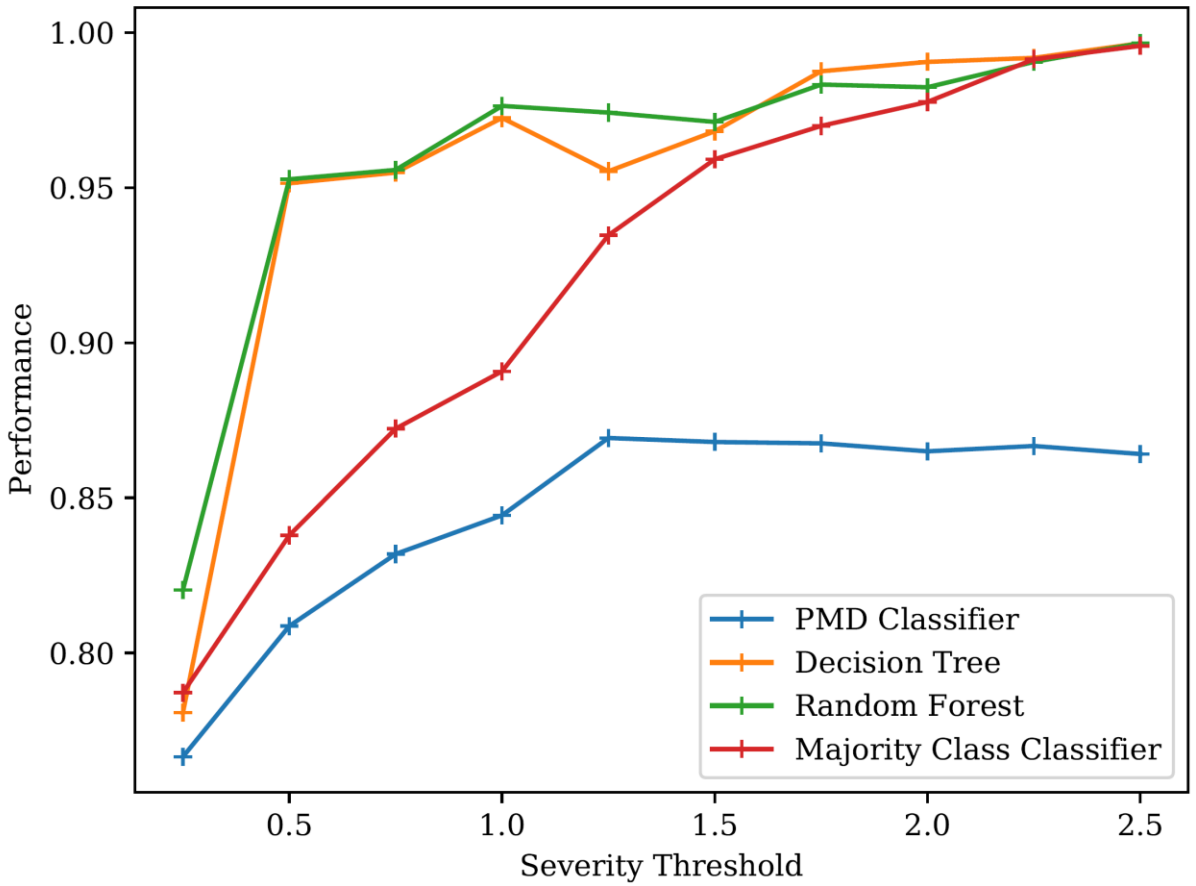
PMD version 6.29.0 and our customized version

- Blob
  - Weighted method count (WMC)
  - Access to foreign data (AFTD)
  - Tight class cohesion (TCC)
- Data class
  - Weight of class (WOC),
  - Number of public attributes (NOPA)
  - Number of accessor methods (NOAM)
  - Weighted method count (WMC)

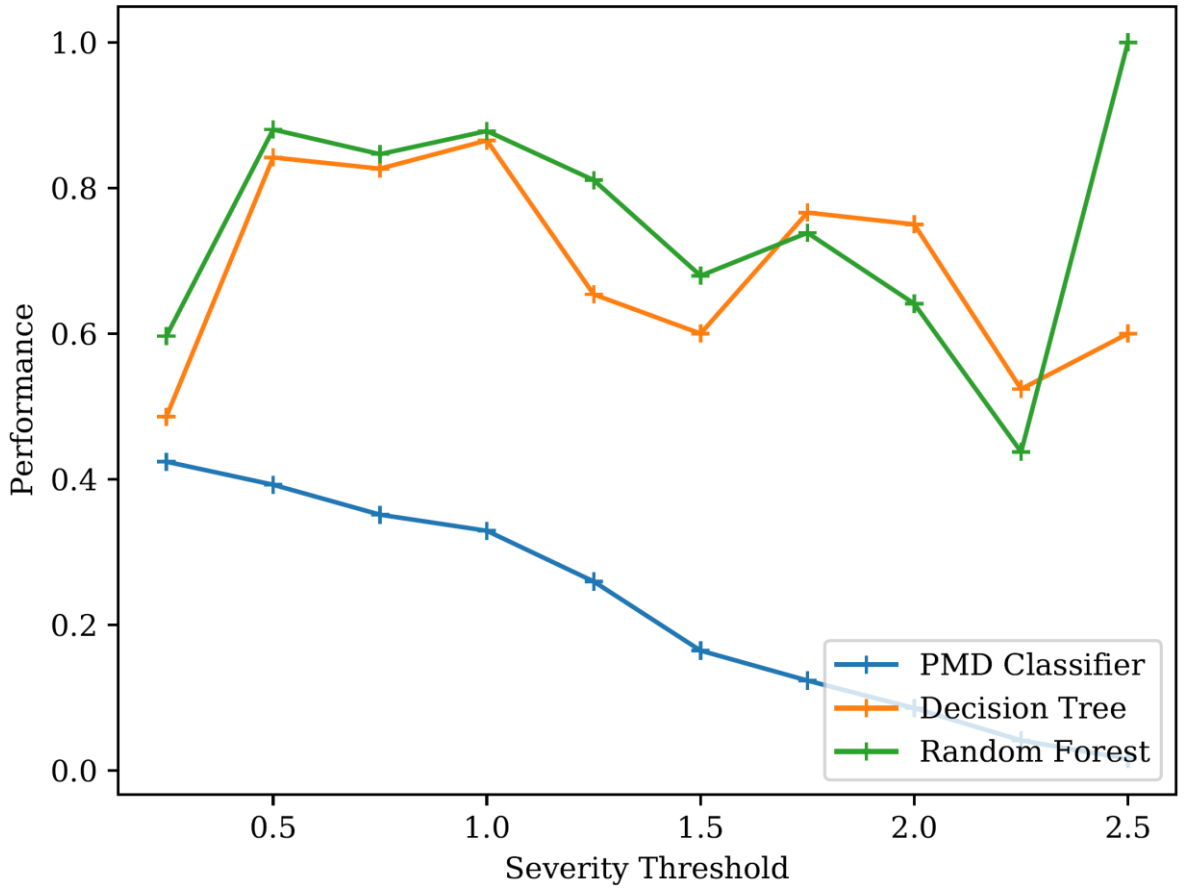
Our machine learning prototype

- MLCQ averaged over multiple reviewers and enriched with metric information
- Three different binary classifiers
  - Random forest classifier
  - Decision tree classifier
  - Majority class classifier
- Variable threshold value to get best result

Accuracy and Precision of Predicting Blob

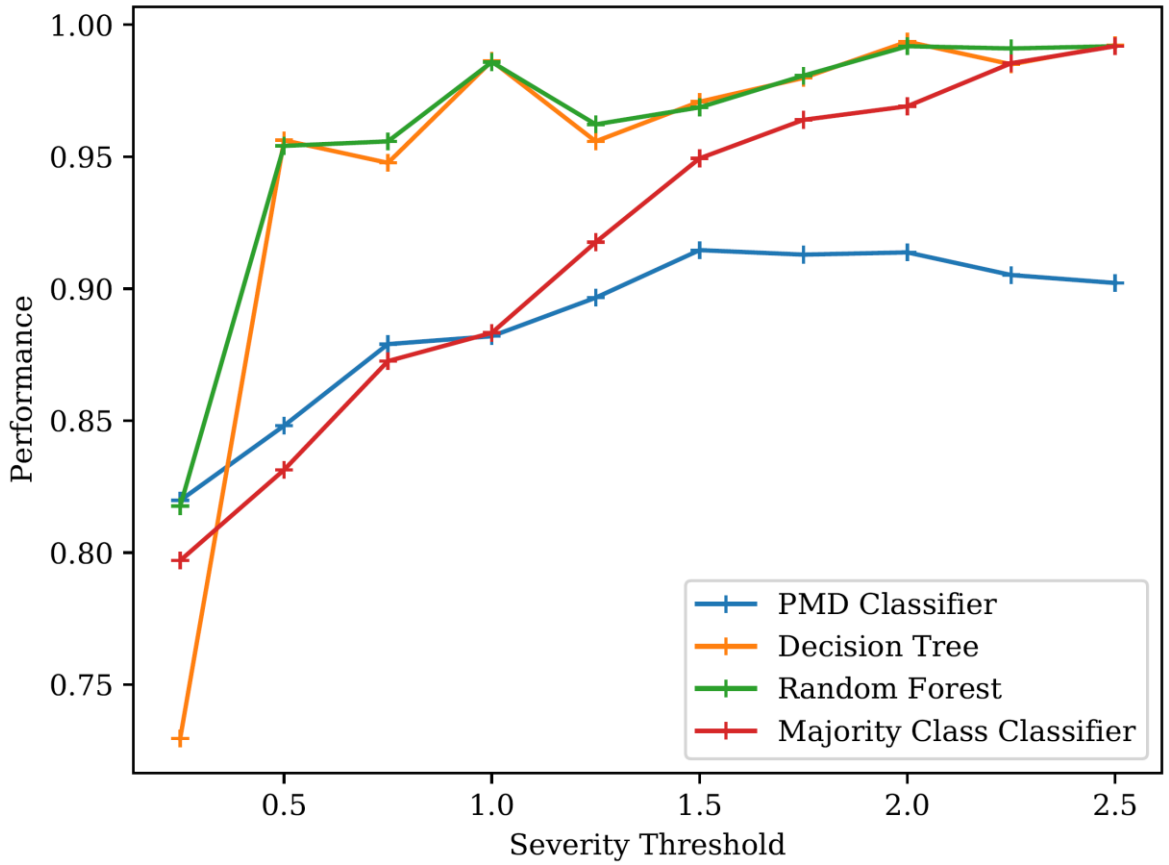


Accuracy

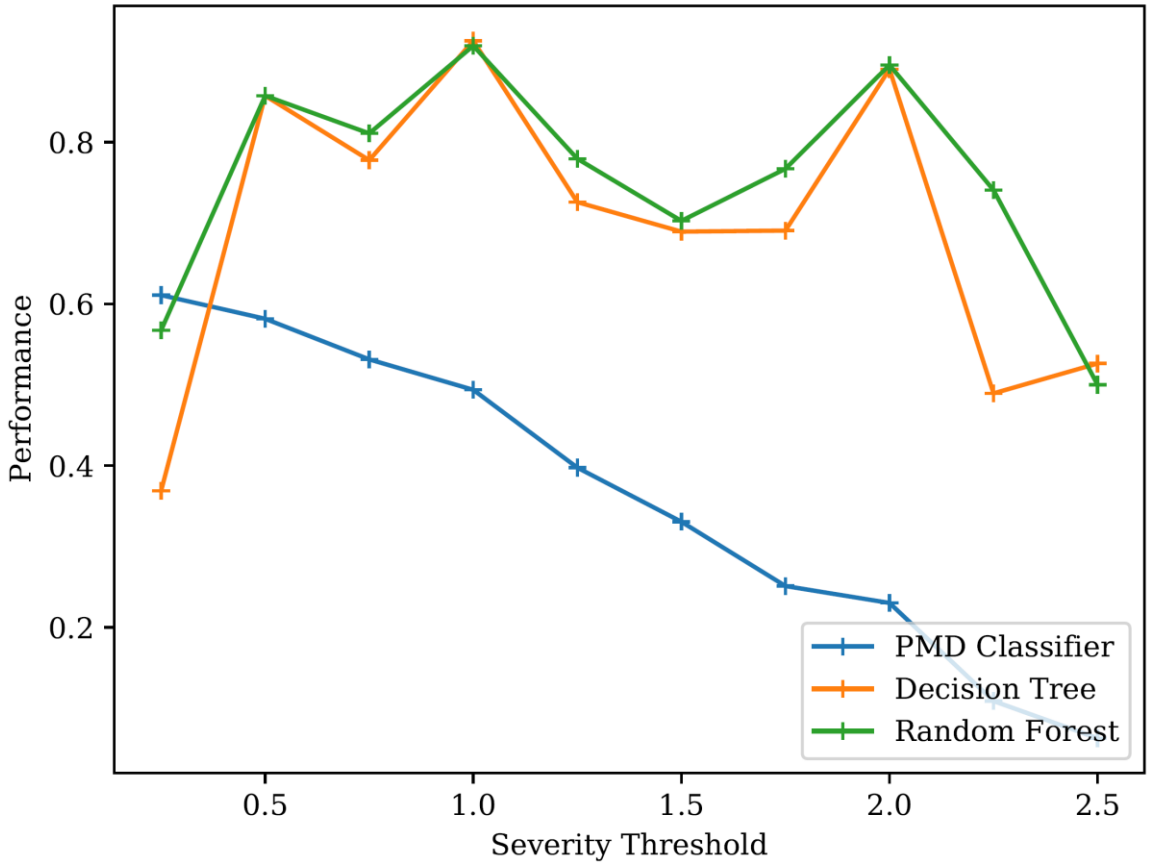


Precision

Accuracy and Precision of Predicting Data Class



Accuracy



Precision

Conclusions

- *Can we mimic a developer's perception of a code smell?*

  Yes – but needs a large dataset of industry projects reviewed by developers, cleaned, and prepared the data to be used in the training process

- *How does machine learning perform compared to other tools?*

  Machine learning classifiers outperform the static metric-based code smell detection tool on all settings.

- The extended dataset is publicly available on OpenML. (https://www.openml.org/d/43078 and https://www.openml.org/d/43079)

Challenges

- More code smells: duplicate code, lazy class,…
- Combining different approaches
- Many programming languages
- More data for validation
- Perception evolution

Observation

- Self-supervised & Unsupervised
- Code smell severity
- Interpretability
- Transfer learning

Questions?

You can always send an e-mail to:
m.m.bonsangue@liacs.leidenuniv.nl

Universiteit
Leiden
The Netherlands


liacs.leidenuniv.nl


Coming soon …


莱顿大学科学学院


liacsCS


@ul_liacs


groups/2084197


User: liacsmedialab


liacs