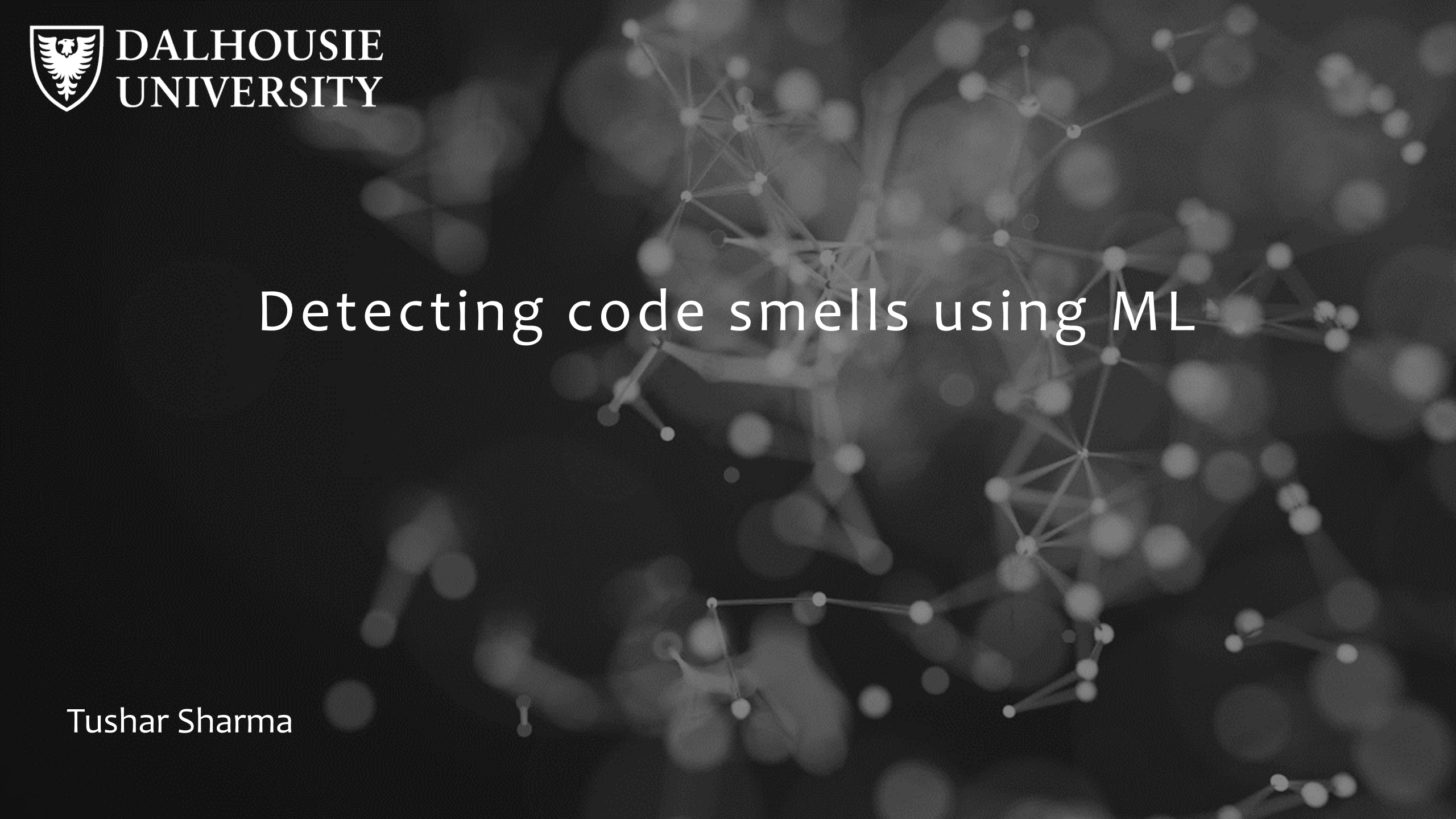

DALHOUSIE
UNIVERSITY
Detecting code smells using ML
Tushar Sharma

Dr. Tushar Sharma

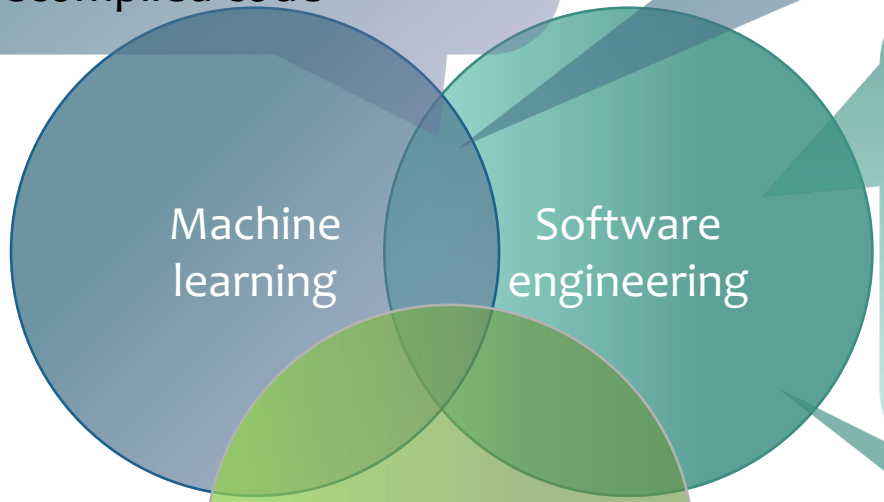
tushar@dal.ca

SMART lab,
Dalhousie University

https://web.cs.dal.ca/~tushar/smart/

- Binary symbol reconstruction
- Program comprehension for decompiled binaries
- Vulnerability analysis for decompiled code

- Machine learning for software engineering
- Software engineering for machine learning



- Source code analysis
- Software quality
- Code smell detection and refactoring
- Developers' productivity
- Program comprehension

Tools and platforms

- Sustainable machine learning
- Energy hotspots and refactorings
- Energy efficient code representation

Sponsors and collaborators

NSERC CRSNG

ACENET
accelerate discovery

Lab2 Market

UCL

CANADA FIRST RESEARCH EXCELLENCE FUND

aws

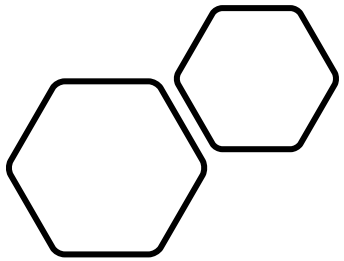
Google Cloud

MITACS

ARCURVE

TIRUPATI

Part 1



Introduction to code smells


CODE SMELLS
CODE SMELLS EVERYWHERE
memegenerator.net


Smells

What is a code smell?

… certain structures in the code that suggest (sometimes they scream for) the possibility of refactoring.

- Kent Beck

20 Definitions of smells: http://www.tusharma.in/smells/smellDefs.html


Bad smells
Fault patterns
Design anomalies
Anti-patterns
Architecture flaws
Design flaws
Problem patterns
Code smells
Structural problems
Design inconsistencies
Architectural bad smells
Violation of design rules
Design principle violations
Design critiques
Architectural mismatch
Malignant patterns
Structural flaws
Aging symptoms
Design smells
Spoiled patterns
Architecturally-relevant code smells
Violation of design heuristics
Architectural anti-patterns
Design problem symptoms
Structural weaknesses
Structural anomalies
Architectural smells
Design disharmonies
Modularity smells
Danger signals
Quality defects
Design defects

Smells' characteristics

- **Indicator** (of a deeper design problem)
- **Poor solution** (a suboptimal or poor solution)
- **Violates best practices** (of the domain)
- **Impacts quality** (make it difficult for a software system to evolve and maintain)
- **Recurrence**

Types of smells

- Code smells
  - Implementation smells
  - Architecture smells
  - Design smells
  - Test smells
  - Performance smells
- Configuration smells
- Database smells
- Models smells
- Usability smells
- Web smells
- …

http://www.tusharma.in/smells/

A Taxonomy of Software Smells

Home

Definitions of a smell

Tools to detect smells

Tushar's Blog

- Code Smells
  - Architecture Smells (38)
  - Design Smells (61)
  - Energy Smells (8)
  - Implementation Smells (13)
  - Performance Smells (14)
  - Test Smells (15)
- Configuration Smells
  - Configuration Smells (Design) (11)
  - Configuration Smells (Implementation) (13)
- Database Smells (13)
- Grammar Smells
  - Navigation Smells (10)
  - Organisation Smells (25)
  - Structure Smells (21)
- Presentation Smells (12)
- Spreadsheet Smells (9)

Total documented smells: 263

Granularity of code smells

Implementation smells

- Scope is limited to a method or a code block

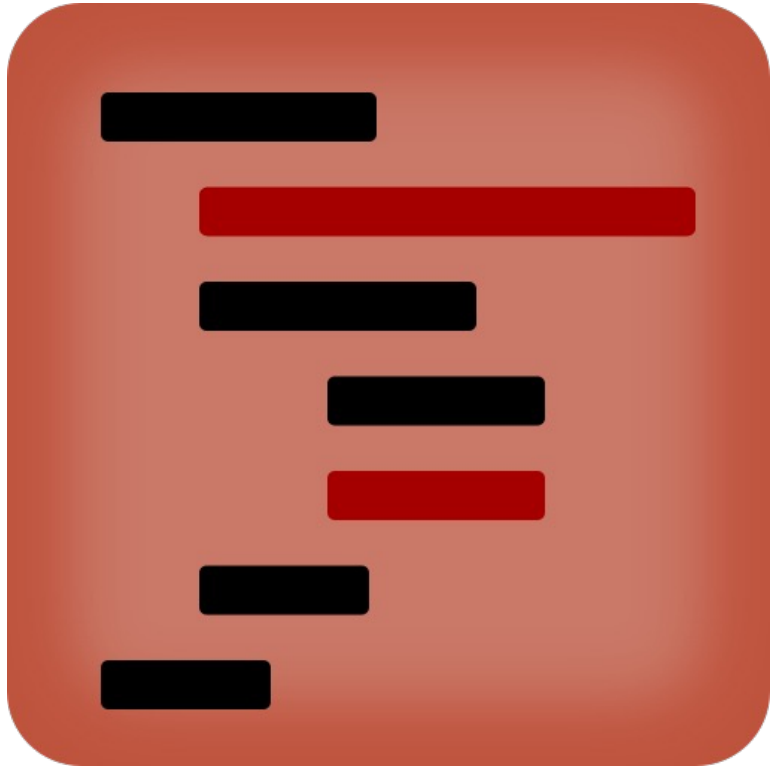
Design smells

- Attributes of a class or a set of classes; scope is classes (types)

Architecture smells

- Scope is at the component level (package/assembly)

Implementation smells



Duplicate Code
Long Parameter List
Complex Conditional
Complex Method
Magic Number
Empty Catch Block
Long Method
Missing Default
Long Identifier
Long Statement

```
private void method_name(parameters) throws Exception {
   try {
      //some initialization statements

      if (condition) {
      //stmts
      }
      for(String tableHeader : pollutantwisePollutantEntriesForReport.keySet()) {
         //stmts
         for(String rowHeader : headers) {
            //stmts
            if (condition){
               //stmts
            }
         }
         for (loop_operands) {
            //stmts
         }
         //stmts
      }
      //stmt
   } catch(Exception e) {
      System.err.println(e);
   }
}
```

```
private void method_name(parameters) throws Exception {
   try {
      //some initialization statements

      if (condition) {
      //stmts
      }
      for(String tableHeader : pollutantwisePollutantEntriesForReport.keySet()) {
         //stmts
         for(String rowHeader : headers) {
            //stmts
            if (condition){
               //stmts
            }
         }
         for (loop_operands) {
            //stmts
         }
         //stmts
      }
      //stmt
   } catch(Exception e) {
      System.err.println(e);
   }
}
```

```
public Boolean drawImage(Image image,
        int x1Dest,
        int y1Dest,
        int x2Dest,
        int y2Dest,
        int x1Source,
        int y1Source,
        int x2Source,
        int y2Source,
        Color color,
        ImageObserver obs)
```

What's that smell?

```
public Boolean drawImage(Image image,
        int x1Dest,
        int y1Dest,
        int x2Dest,
        int y2Dest,
        int x1Source,
        int y1Source,
        int x2Source,
        int y2Source,
        Color color,
        ImageObserver obs)
```

```
if (sum > 13.05689)
        {
        //do something
        }
```

What's that smell?

```
if (sum > 13.05689)
        {
        //do something
        }
```

```
//some other code
try {
    output.writeObject(quarks);
}
catch (IOException ex) {
}
```

Can there be multiple smells in a method?

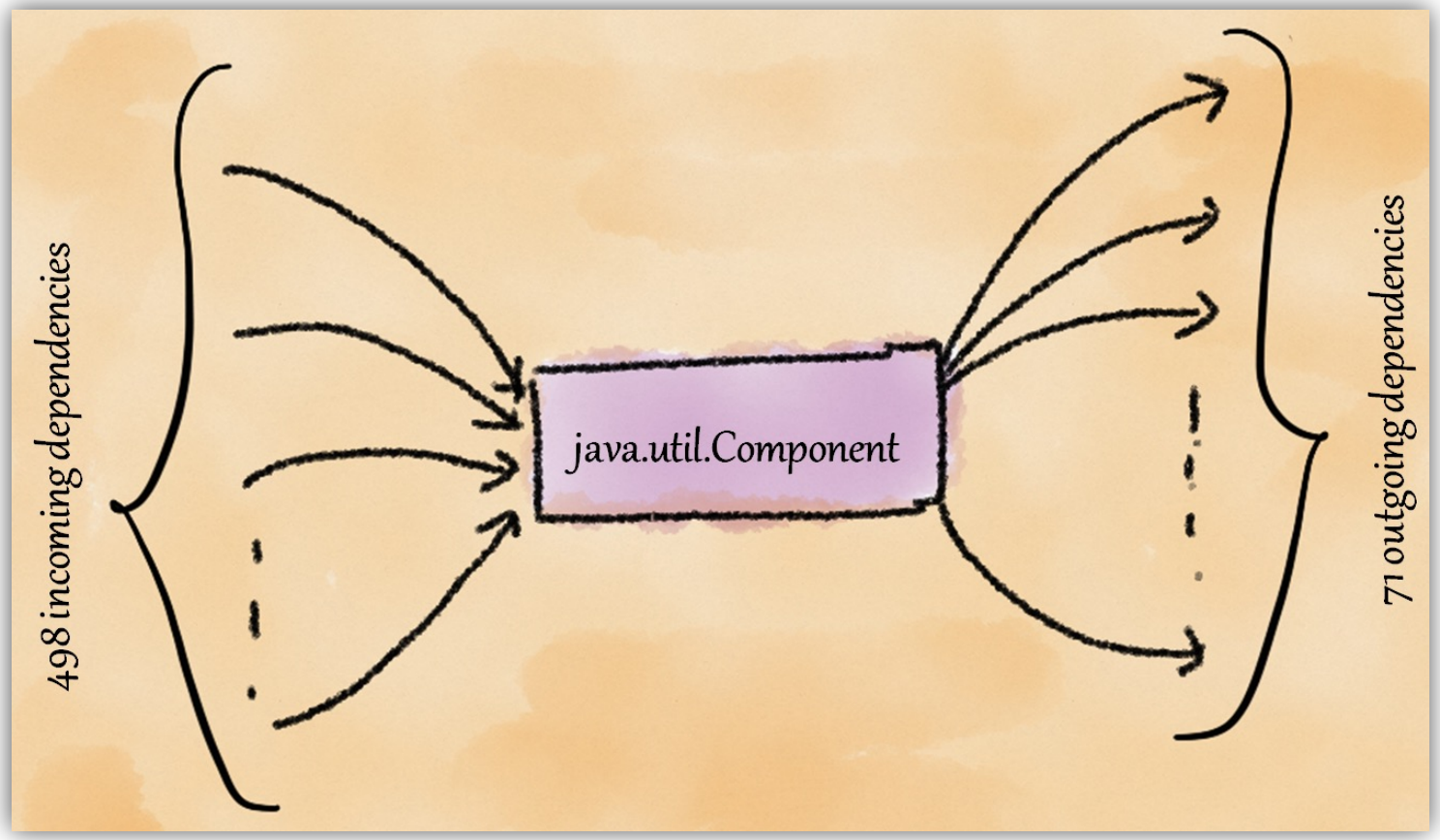
Implementation smells

- Complex Conditional: a complex conditional statement
- Complex Method: a method with high cyclomatic complexity
- Duplicate Code: a code clone within a method
- Empty Catch Block: a catch block of an exception is empty
- Long Identifier: an identifier with excessive length
- Long Method: a method is excessively long
- Long Parameter List: a method has long parameter list
- Long Statement: an excessive long statement
- Magic Number: an unexplained number is used in an expression
- Missing Default: a switch statement does not contain a default case
- Virtual Method Call from Constructor: a constructor calls a virtual method

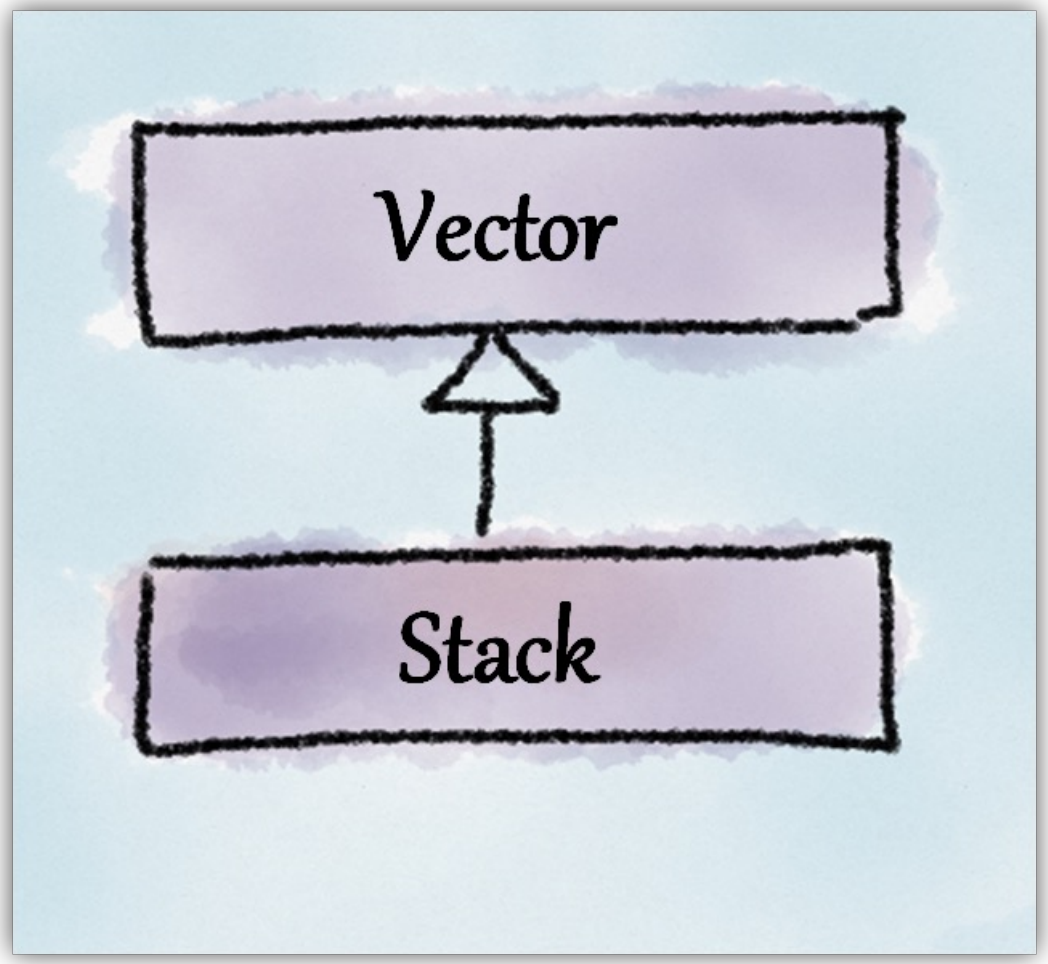
Design smells

Hub-like Modularization

This smell arises when an abstraction has dependencies (both incoming and outgoing) with a large number of other abstractions.



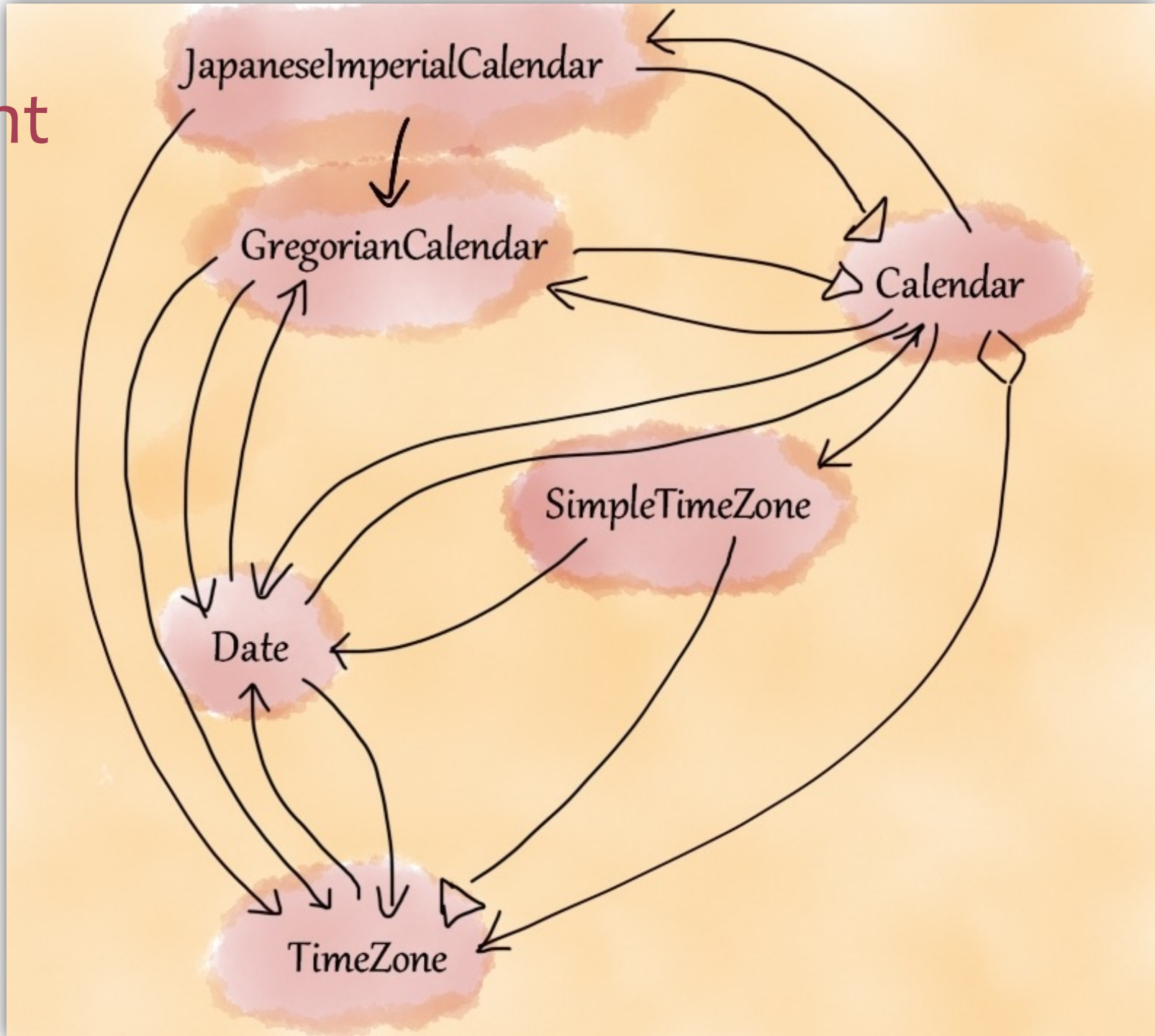
Broken Hierarchy

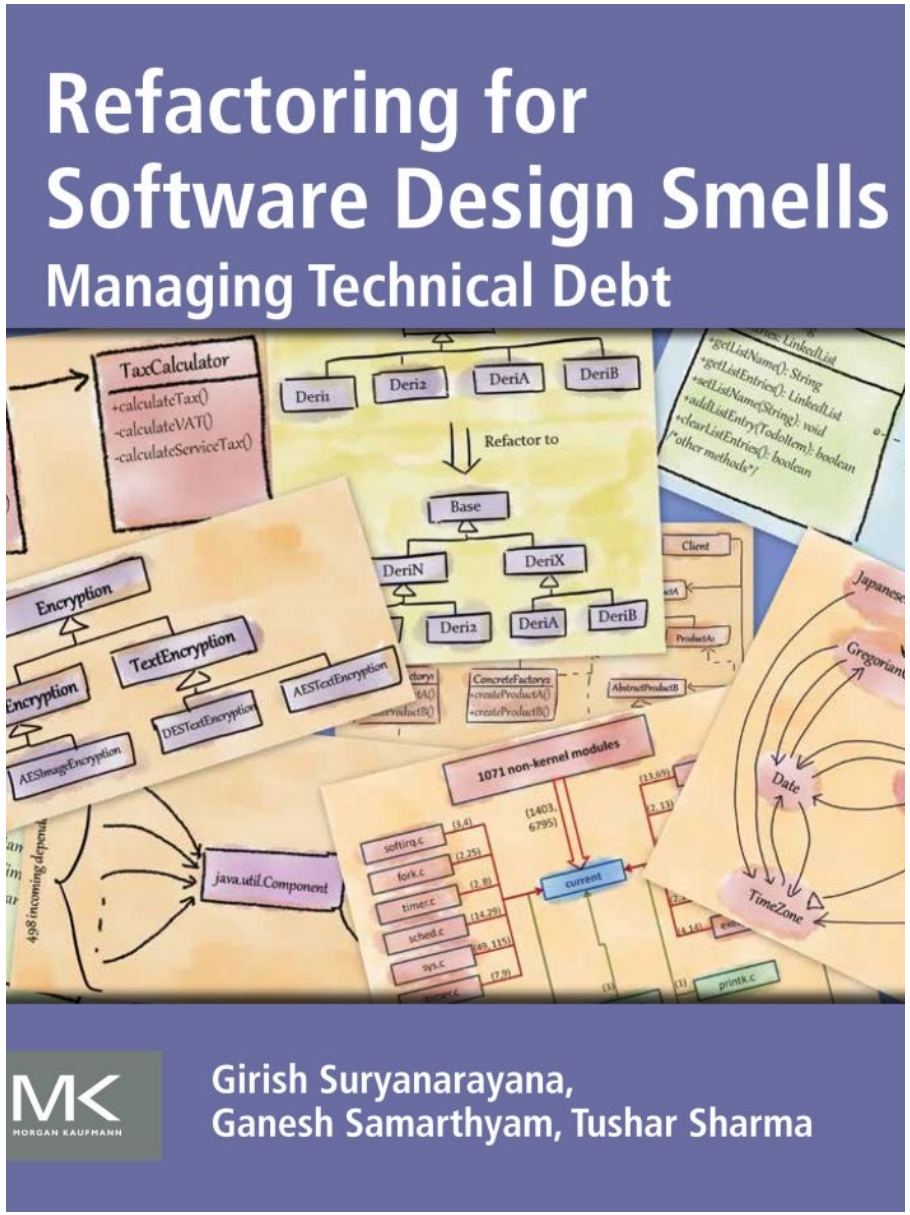


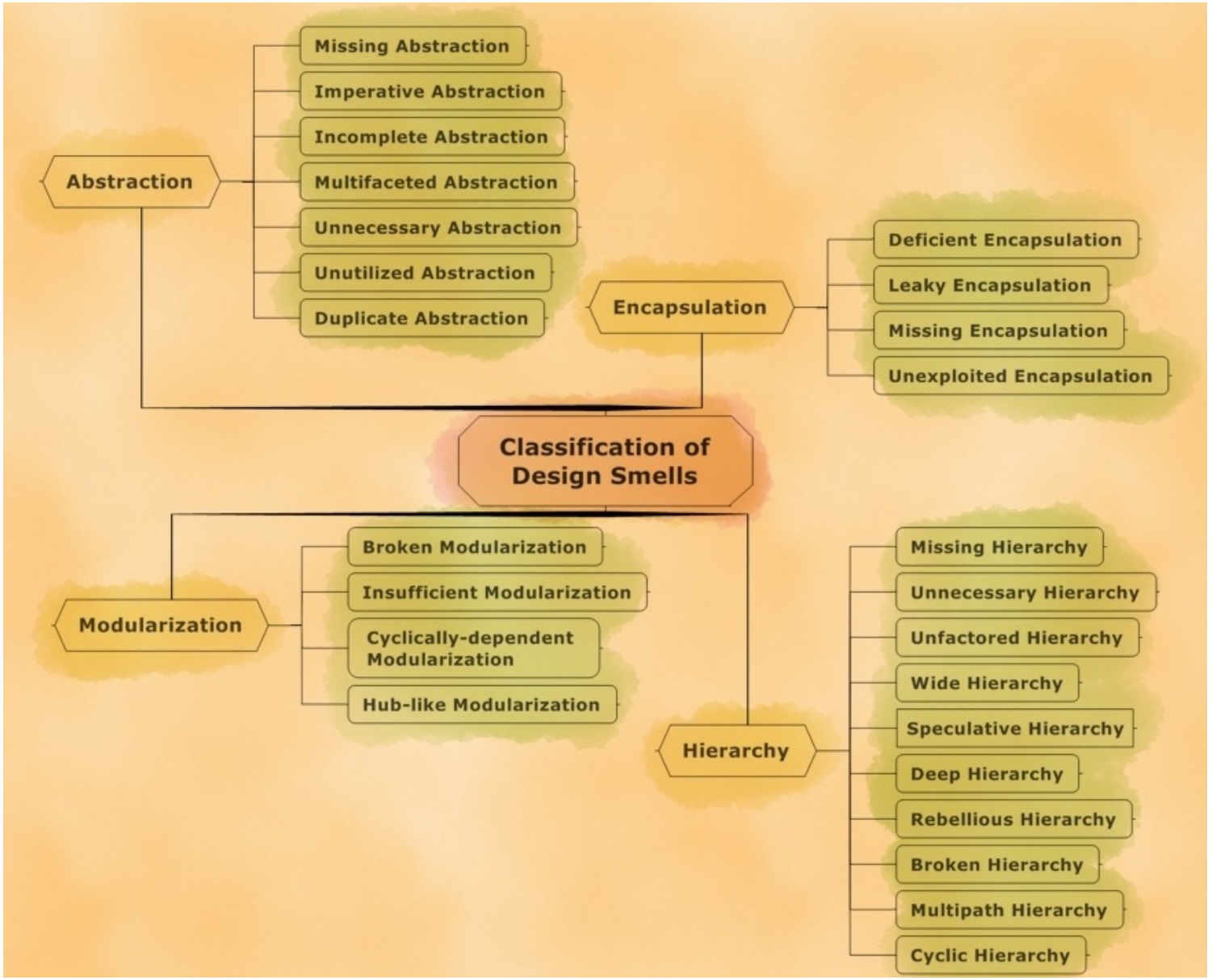
This smell arises when a supertype and its subtype conceptually do not share an “IS-A” relationship resulting in broken substitutability.

Cyclically-dependent Modularization

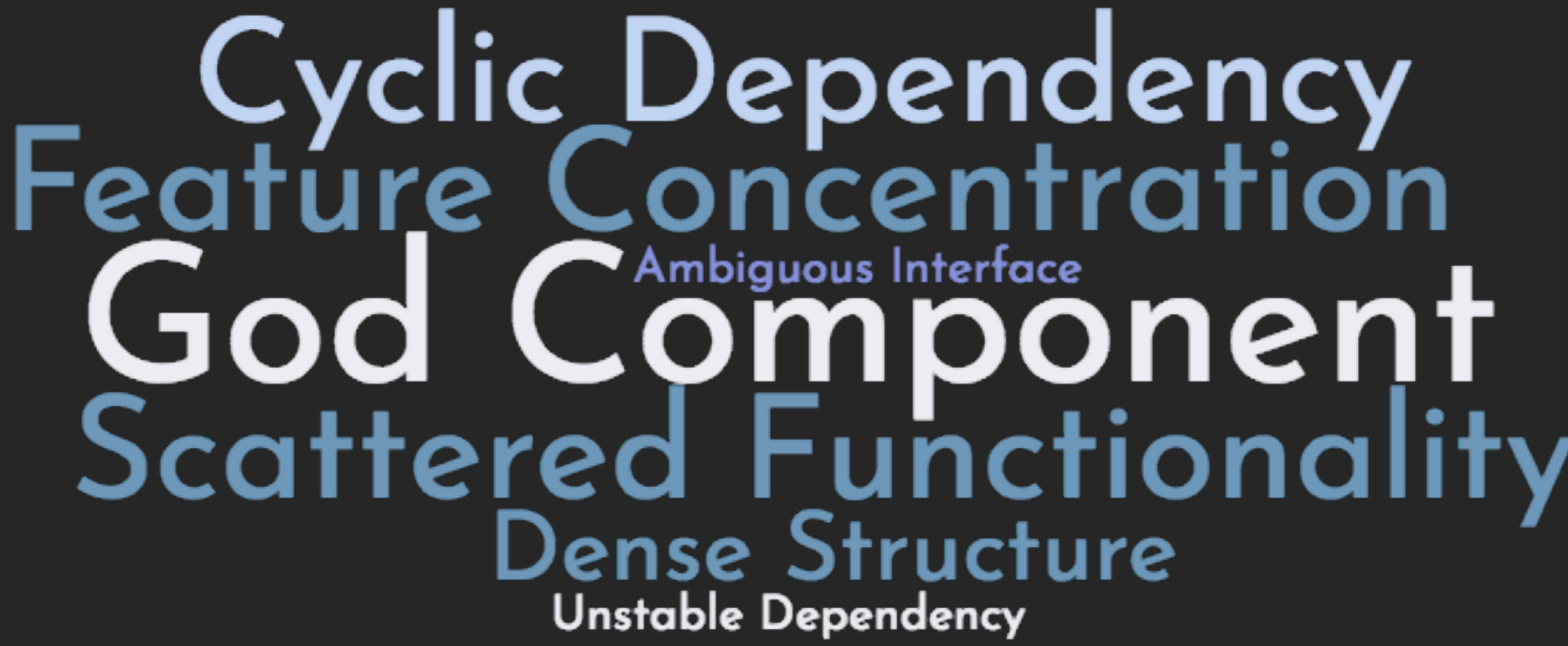
This smell arises when two or more abstractions depend on each other directly or indirectly.

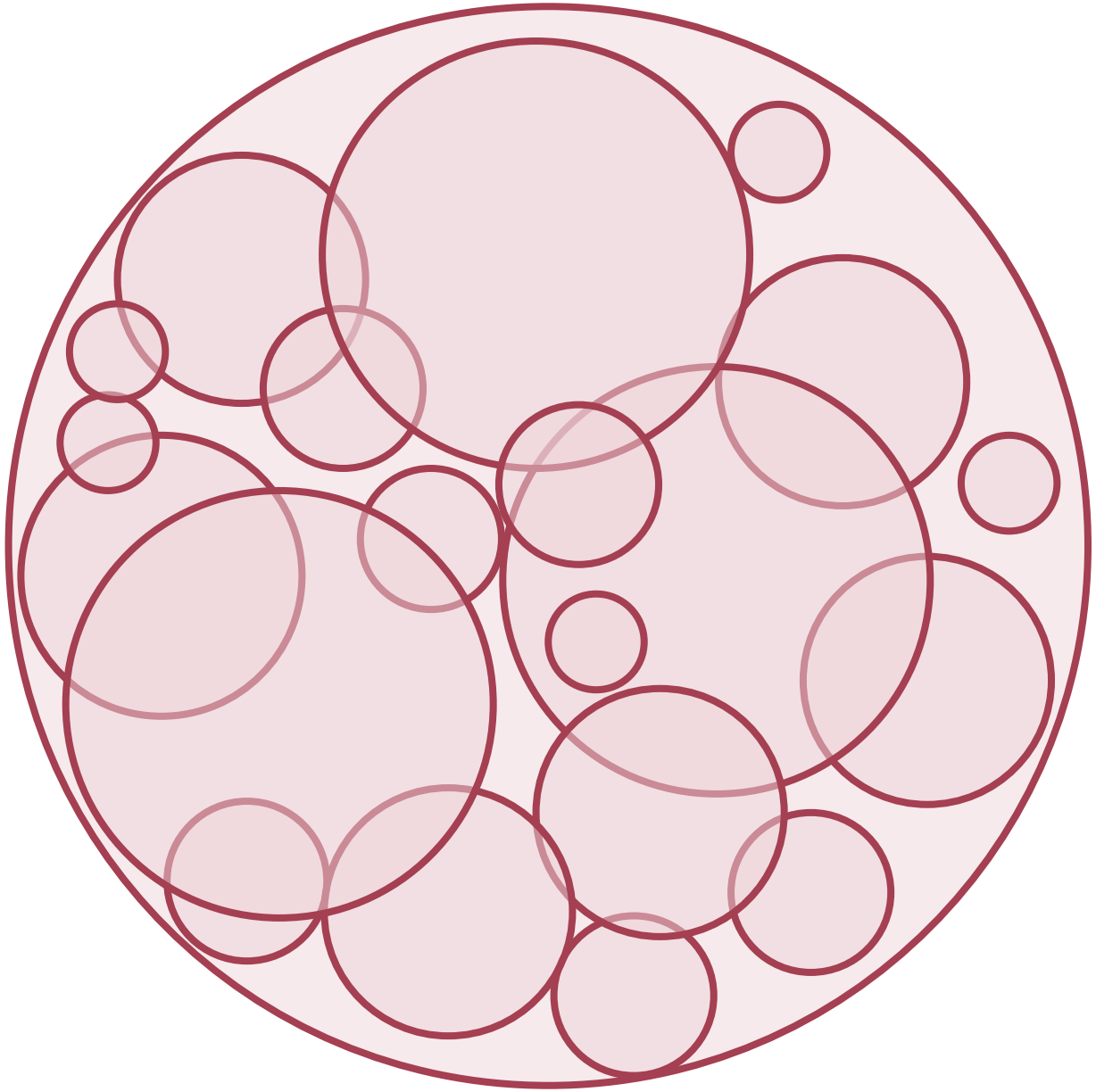



Refactoring for
Software Design Smells
Managing Technical Debt
TaxCalculator
+calculateTax()
-calculateVAT()
-calculateServiceTax()
Refactor to
Base
Encryption
TextEncryption
AESTextEncryption
DESTextEncryption
1071 non-kernel modules
java.util.Component
Date
TimeZone
MORGAN KAUFMANN
Girish Suryanarayana,
Ganesh Samarthyam, Tushar Sharma

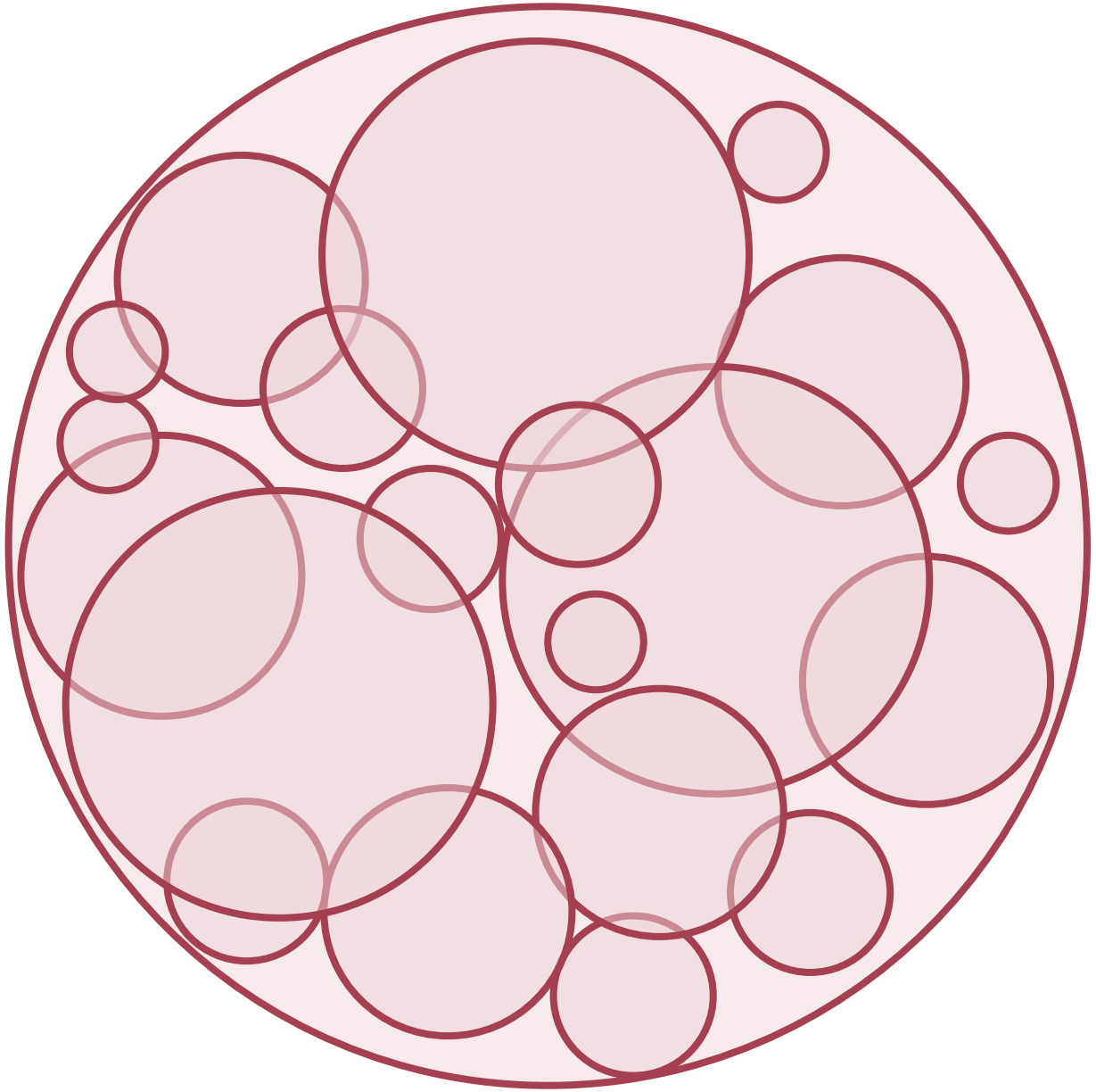

Classification of Design Smells
Abstraction
Missing Abstraction
Imperative Abstraction
Incomplete Abstraction
Multifaceted Abstraction
Unnecessary Abstraction
Unutilized Abstraction
Duplicate Abstraction
Encapsulation
Deficient Encapsulation
Leaky Encapsulation
Missing Encapsulation
Unexploited Encapsulation
Modularization
Broken Modularization
Insufficient Modularization
Cyclically-dependent Modularization
Hub-like Modularization
Hierarchy
Missing Hierarchy
Unnecessary Hierarchy
Unfactored Hierarchy
Wide Hierarchy
Speculative Hierarchy
Deep Hierarchy
Rebellious Hierarchy
Broken Hierarchy
Multipath Hierarchy
Cyclic Hierarchy

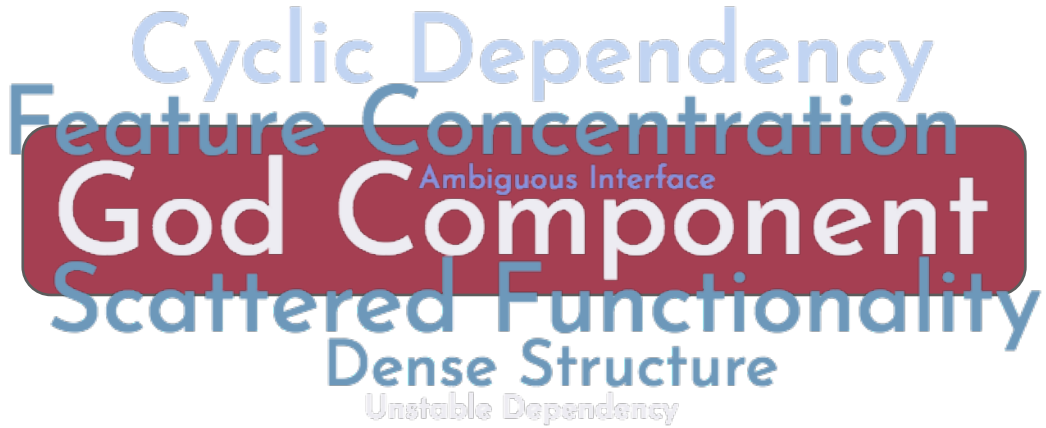
Architecture smells


Cyclic Dependency
Feature Concentration
Ambiguous Interface
God Component
Scattered Functionality
Dense Structure
Unstable Dependency

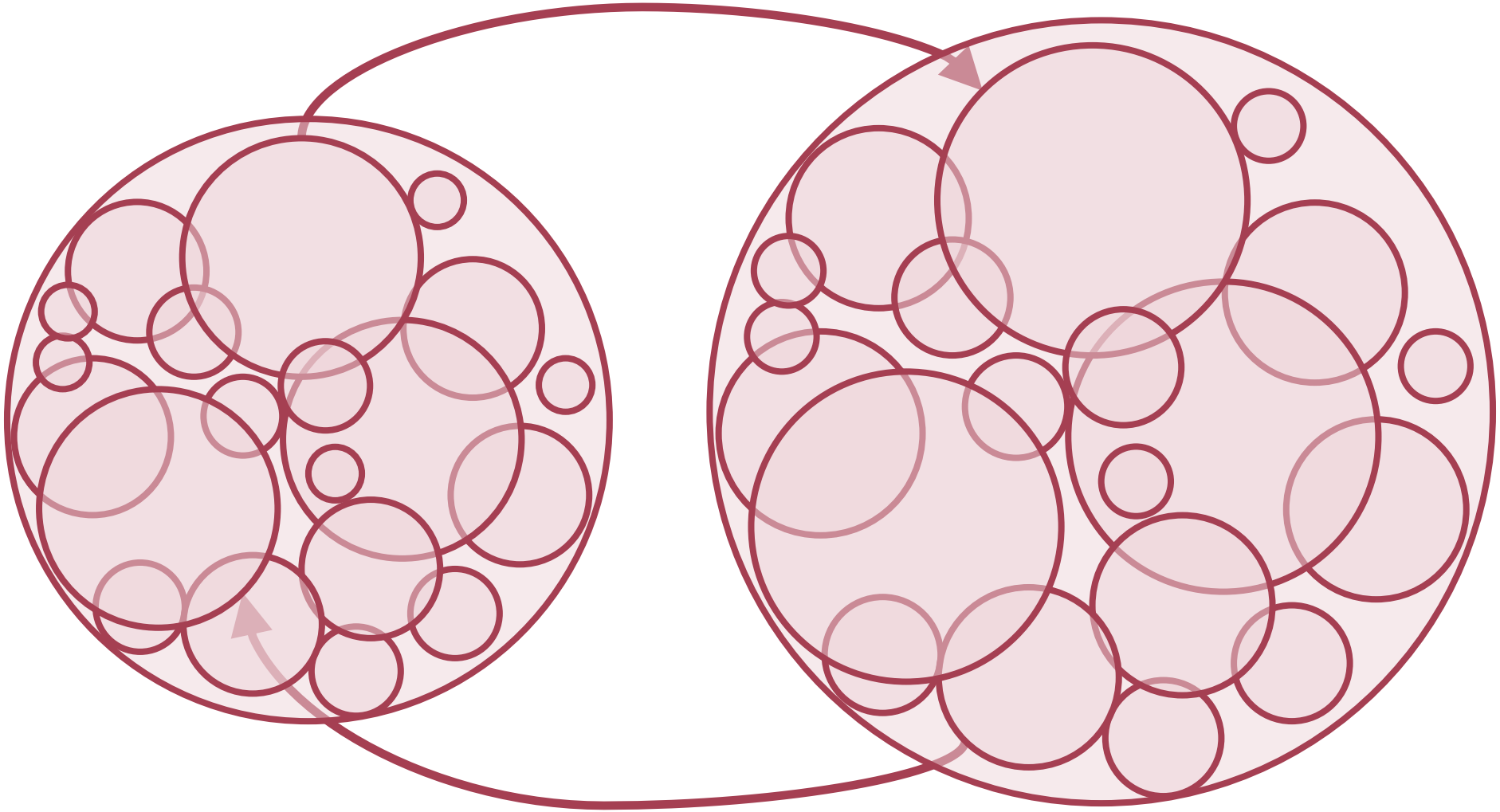


What's that smell?

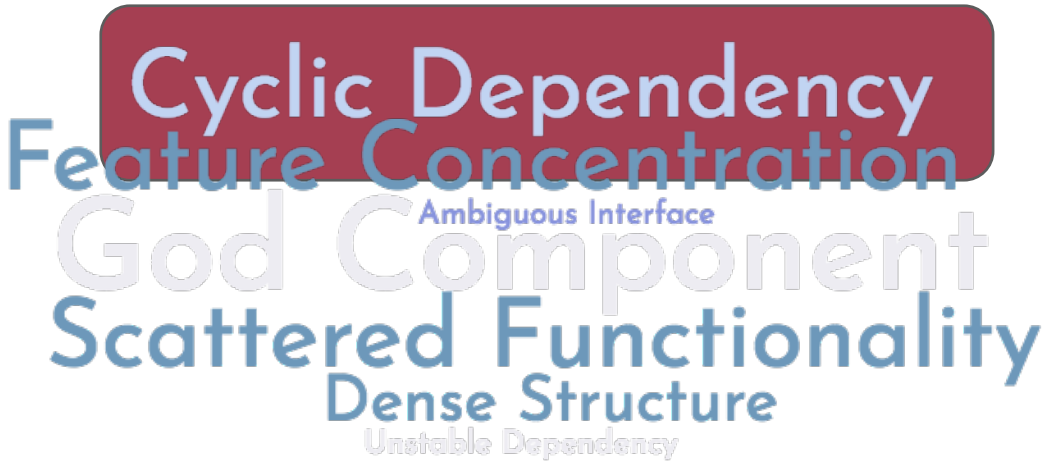




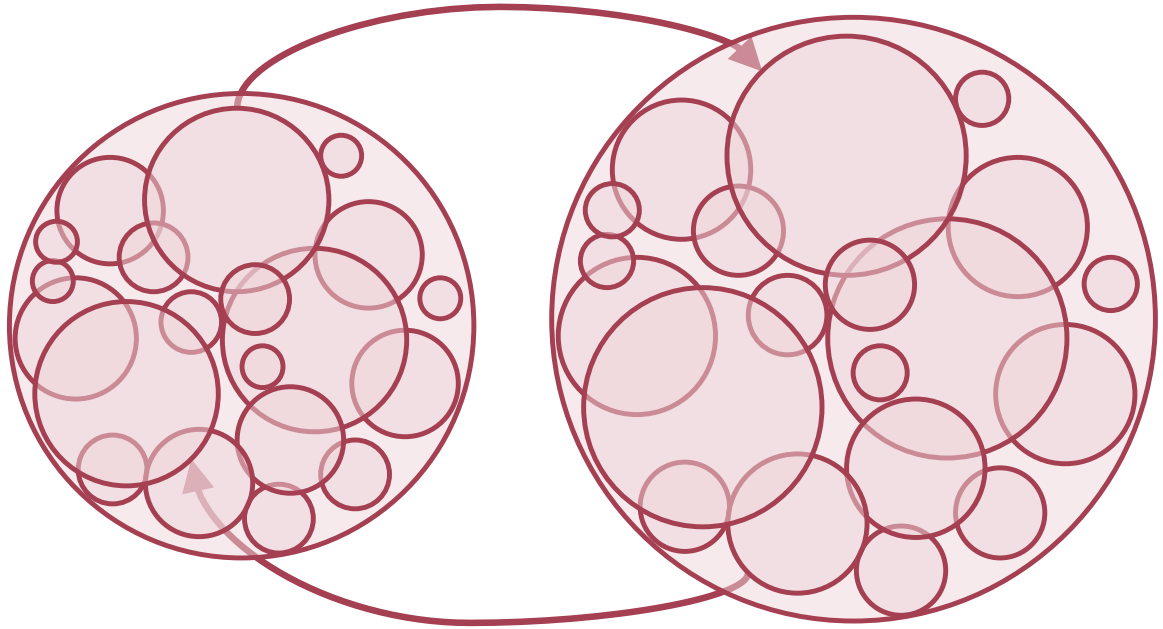
when a component is excessively large either in the terms of LOC or number of classes.



What's that smell?



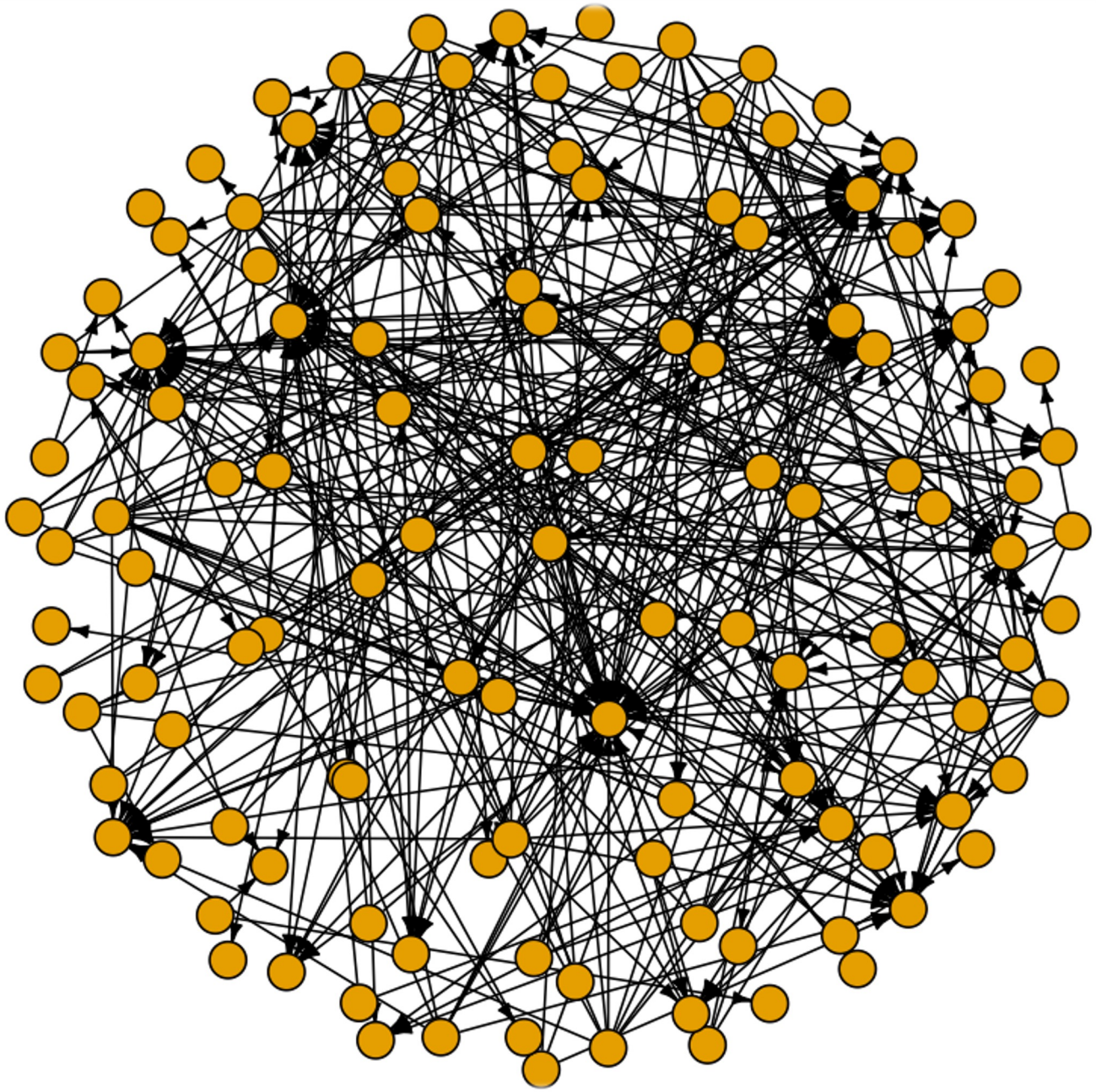
When two or more architecture components depend on each other directly or indirectly



Dense structure

when components have excessive and dense dependencies without any particular structure.

$$\text{Average degree} = \frac{2 * |E|}{|V|}$$



Architecture smells

Cyclic Dependency
This smell arises when two or more architecture components depend on each other directly or indirectly.

Unstable Dependency
This smell arises when a component depends on other components that are less stable than itself.

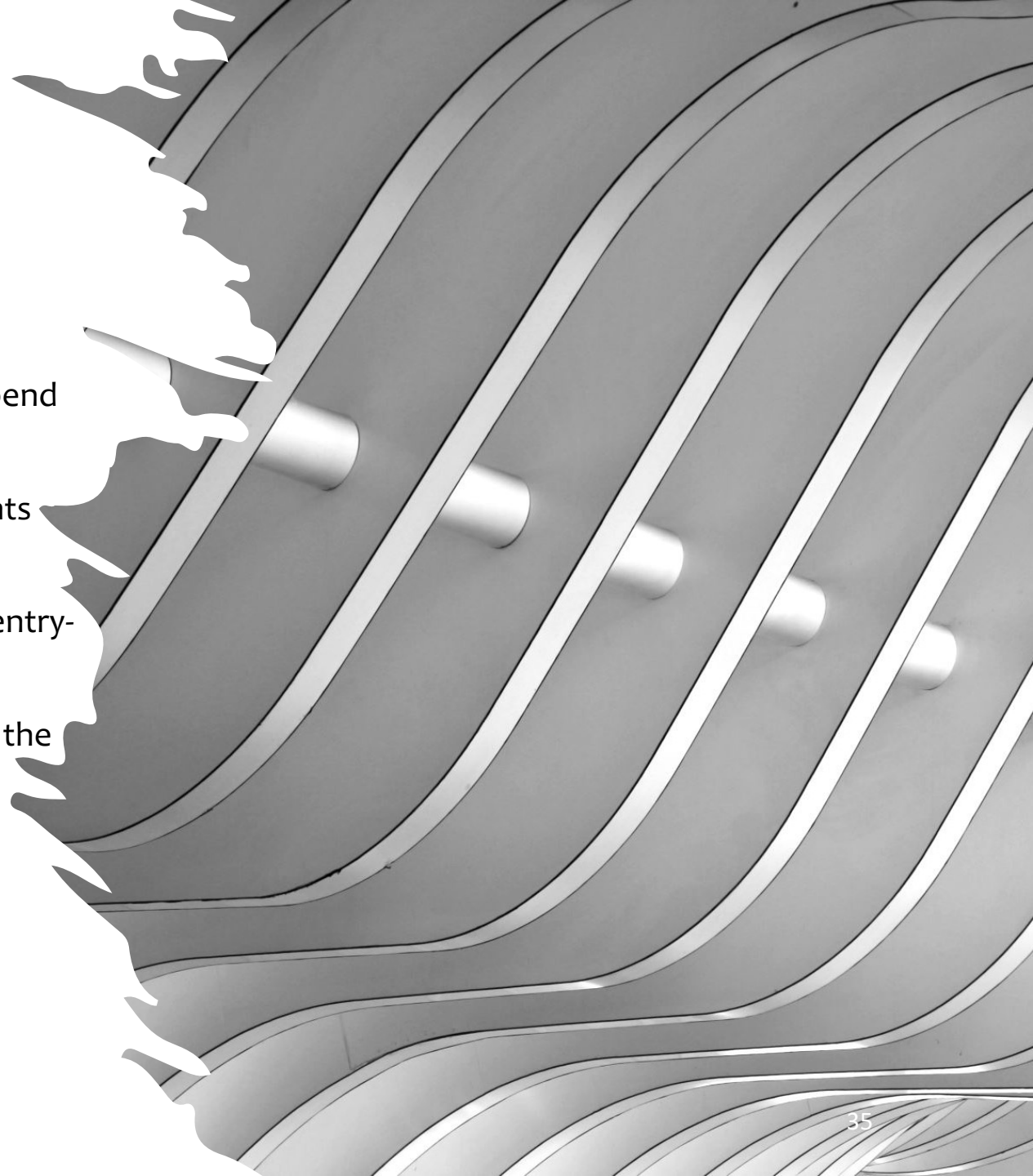
Ambiguous Interface
This smell arises when a component offers only a single, general entry-point into the component.

God Component
This smell occurs when a component is excessively large either in the terms of LOC or number of classes.

Feature Concentration
This smell occurs when a component realizes more than one architectural concern/feature.

Scattered Functionality
This smell arises when multiple components are responsible for realizing the same high-level concern.

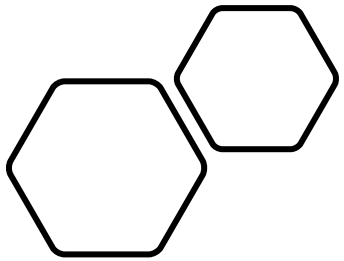
Dense Structure
This smell arises when components have excessive and dense dependencies without any particular structure.



Ponder

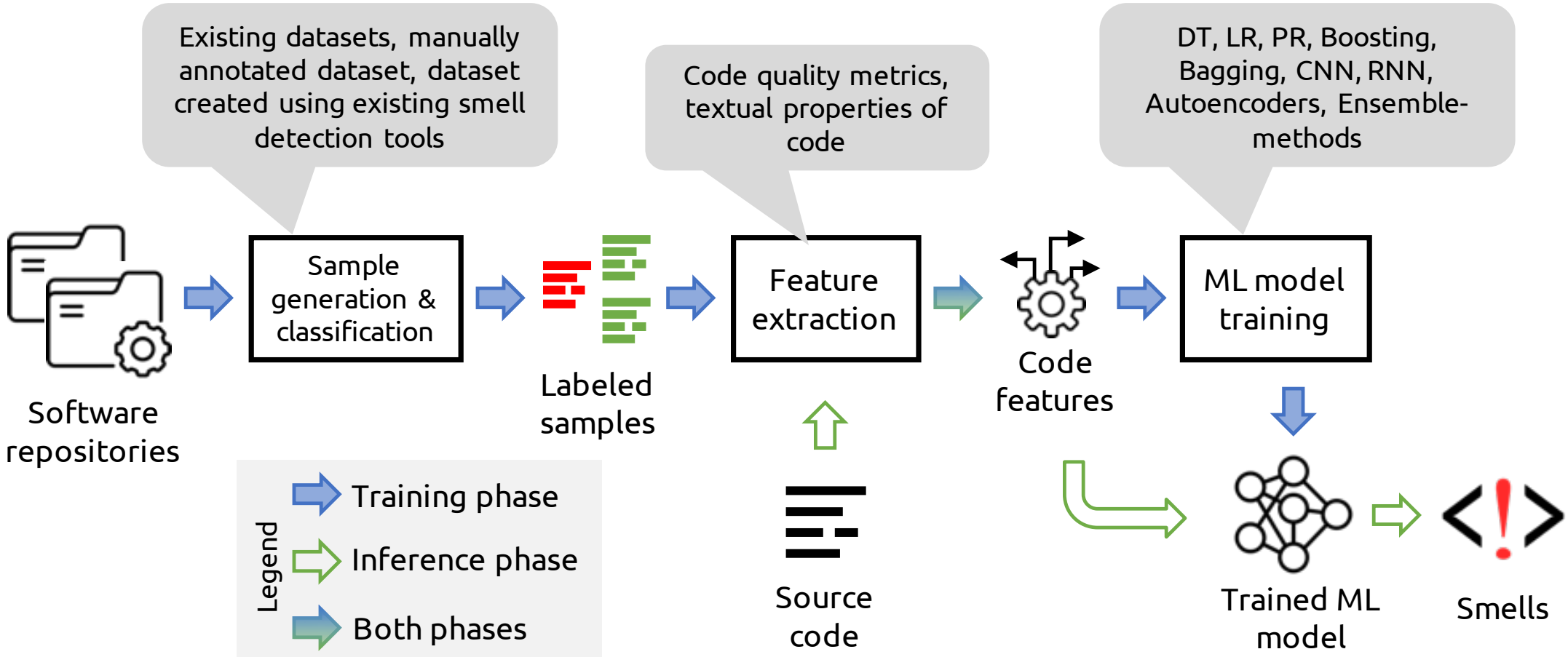
Are smells and bugs same?

Part 2



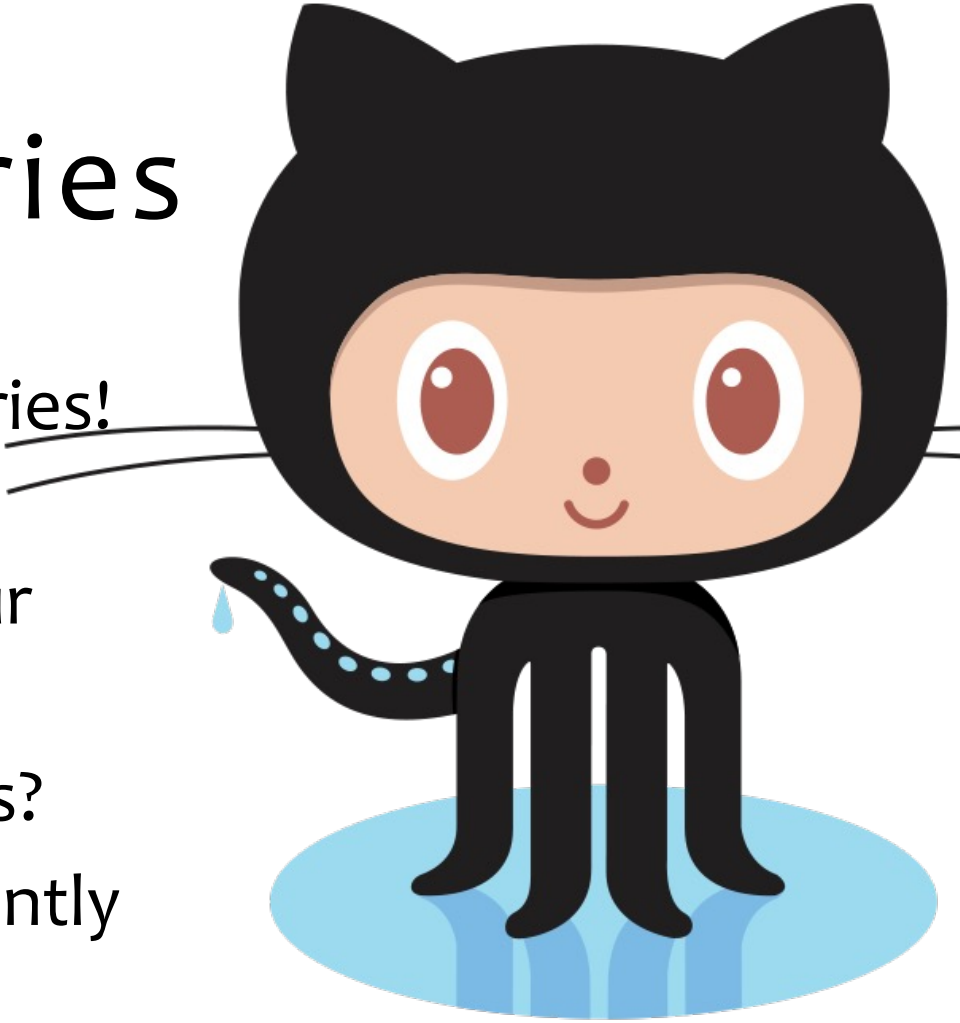
Code smell detection using traditional ML

Overview


Existing datasets, manually annotated dataset, dataset created using existing smell detection tools
Code quality metrics, textual properties of code
DT, LR, PR, Boosting, Bagging, CNN, RNN, Autoencoders, Ensemble-methods
Software repositories
Sample generation & classification
Labeled samples
Feature extraction
Code features
ML model training
Source code
Trained ML model
Smells
Legend
Training phase
Inference phase
Both phases

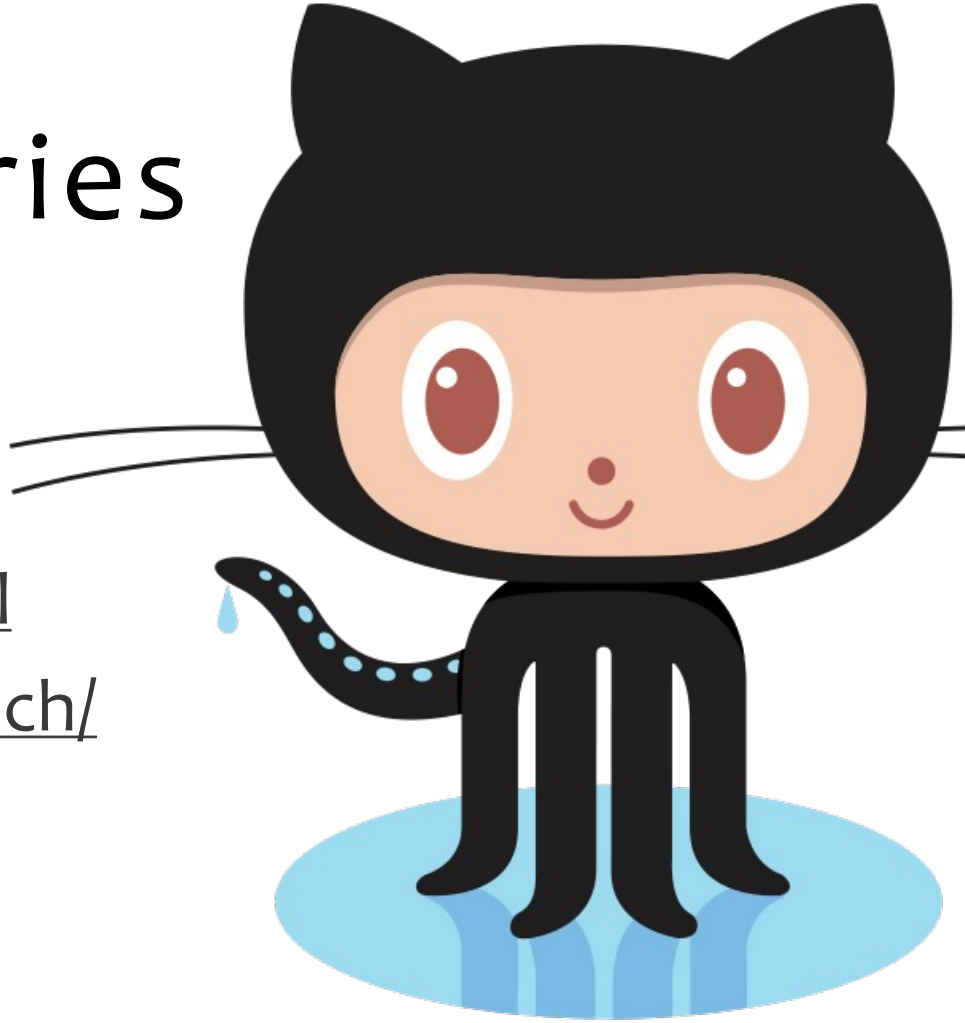
Identify GitHub repositories

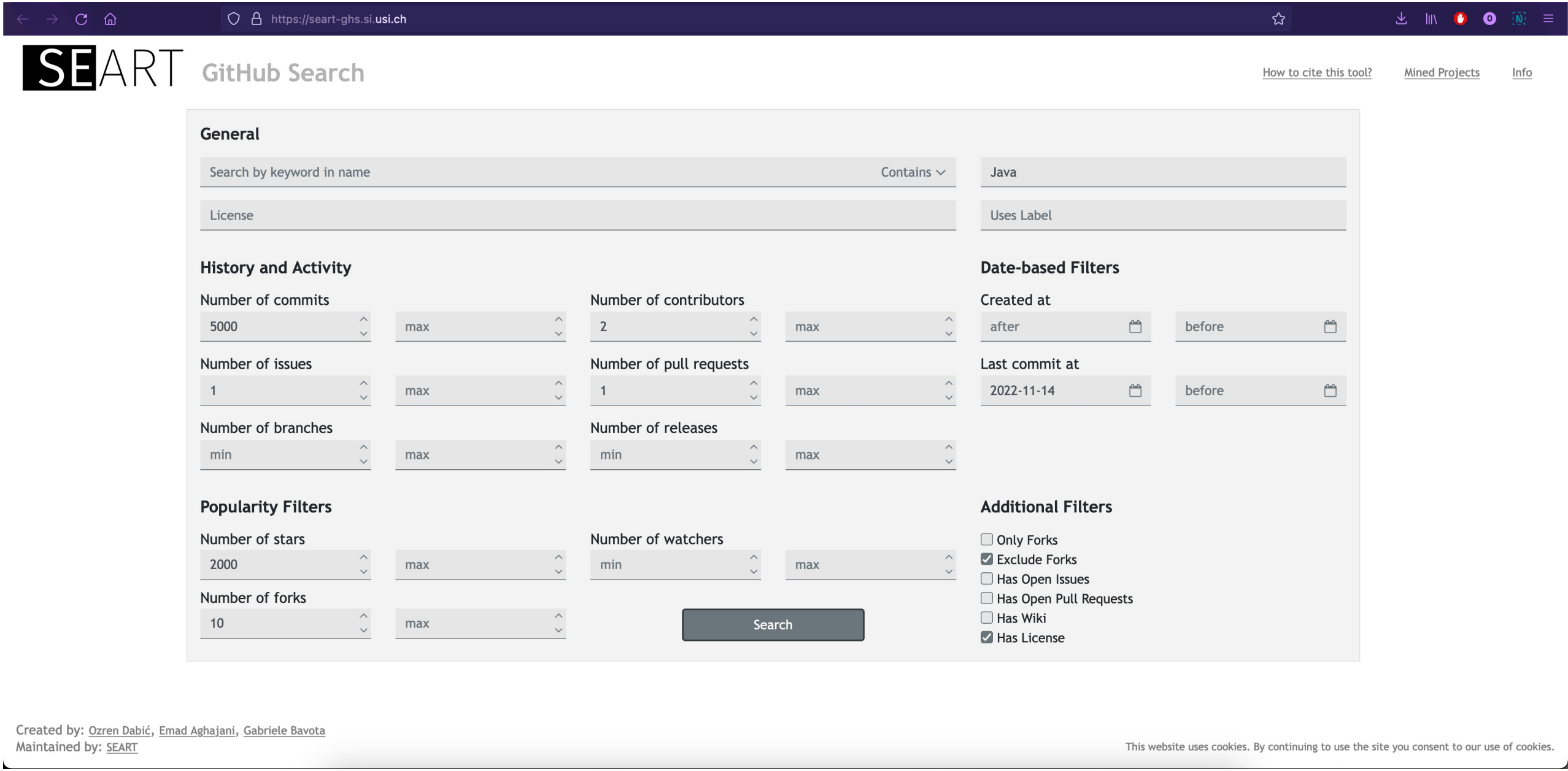
- GitHub has more than 200 million repositories! (June 2022)
- You don't require the entire GitHub for your experiment
- How can you select a subset of repositories?
- A random selection would contain significantly poor-quality code (think about toy, small, obsolete, and example projects)
- Therefore, you need a **concrete criteria** to select GitHub repositories.



Identify GitHub repositories

- GHTorrent (legacy)
- RepoRepears - https://reporeapers.github.io/results/1.html
- GitHub search tool - https://seart-ghs.si.usi.ch/
- GitHub GraphQL API - https://docs.github.com/en/graphql



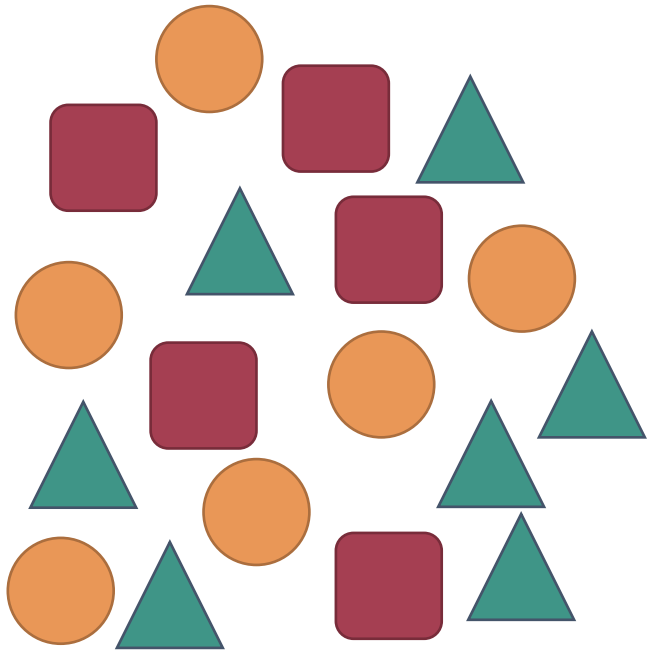

https://seart-ghs.si.usi.ch
SEART
GitHub Search
How to cite this tool?
Mined Projects
Info
General
Search by keyword in name
Contains
Java
License
Uses Label
History and Activity
Number of commits
5000
max
Number of contributors
2
max
Number of issues
1
max
Number of pull requests
1
max
Number of branches
min
max
Number of releases
min
max
Date-based Filters
Created at
after
before
Last commit at
2022-11-14
before
Popularity Filters
Number of stars
2000
max
Number of watchers
min
max
Number of forks
10
max
Search
Additional Filters
Only Forks
Exclude Forks
Has Open Issues
Has Open Pull Requests
Has Wiki
Has License
Created by: Ozren Dabić, Emad Aghajani, Gabriele Bavota
Maintained by: SEART
This website uses cookies. By continuing to use the site you consent to our use of cookies.

Download GitHub repositories

- "git clone" command?
  - Downloading many repositories using plain "git clone" command may lead you ban your IP address for a day or two.
  - One option is to add time delays
- Use a git library for python
- You may also use curl to download repositories with API token

jupyter repo_download.ipynb

Creating training dataset



- Ideally, the training dataset comes from manual annotations.
- However, it is often laborious and costly

- Use existing tool, or better, a set of tools
- Use existing tool(s) to generate initial set and then verify it manually

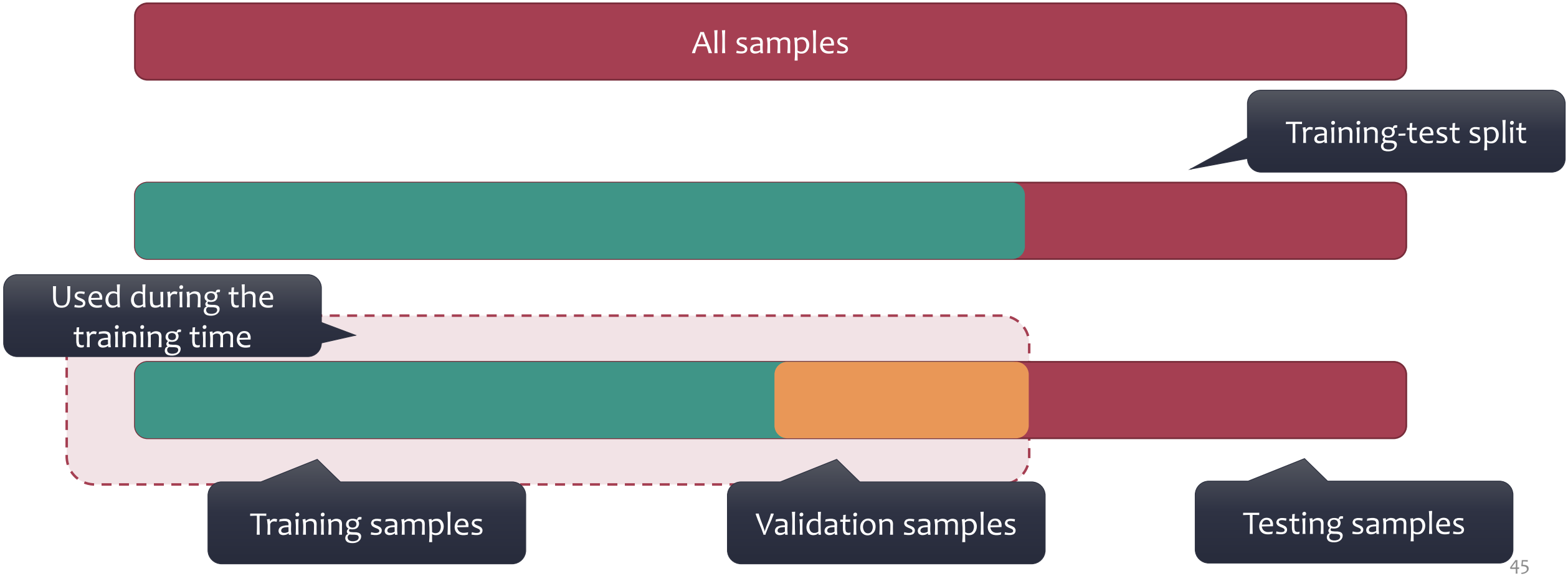
jupyter analyze_code.ipynb

jupyter create_dataset_ML.ipynb

Training ML models

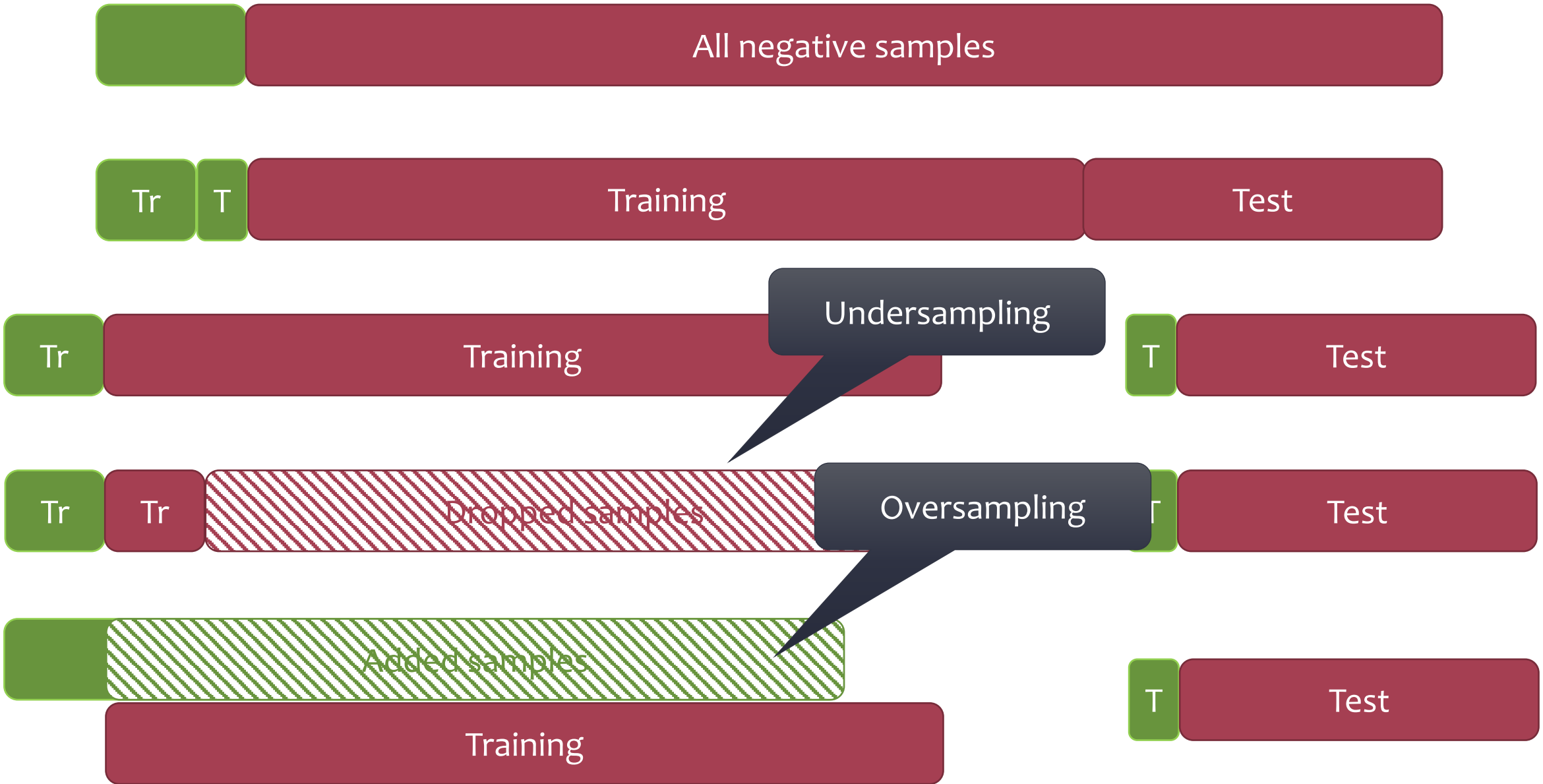
- Balance vs imbalanced dataset
- Splitting dataset into training, validation, and test sets
- ML models
- Metrics

Training, validation, and testing set



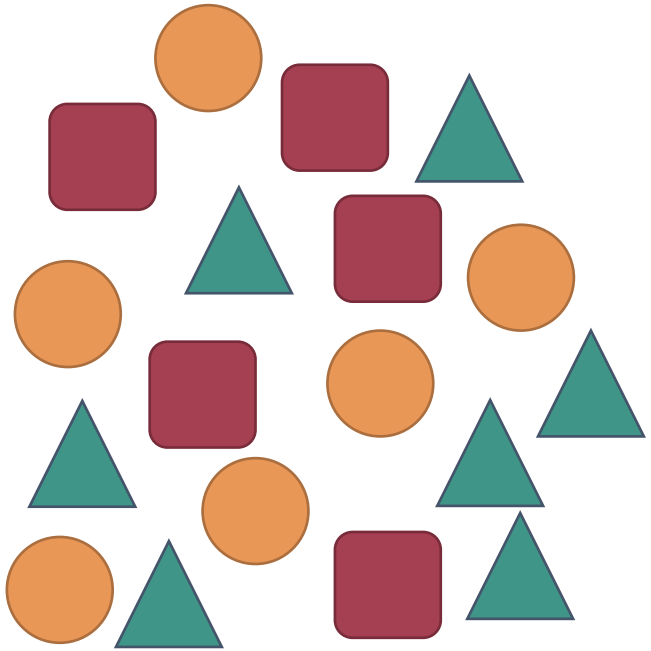
Balanced dataset

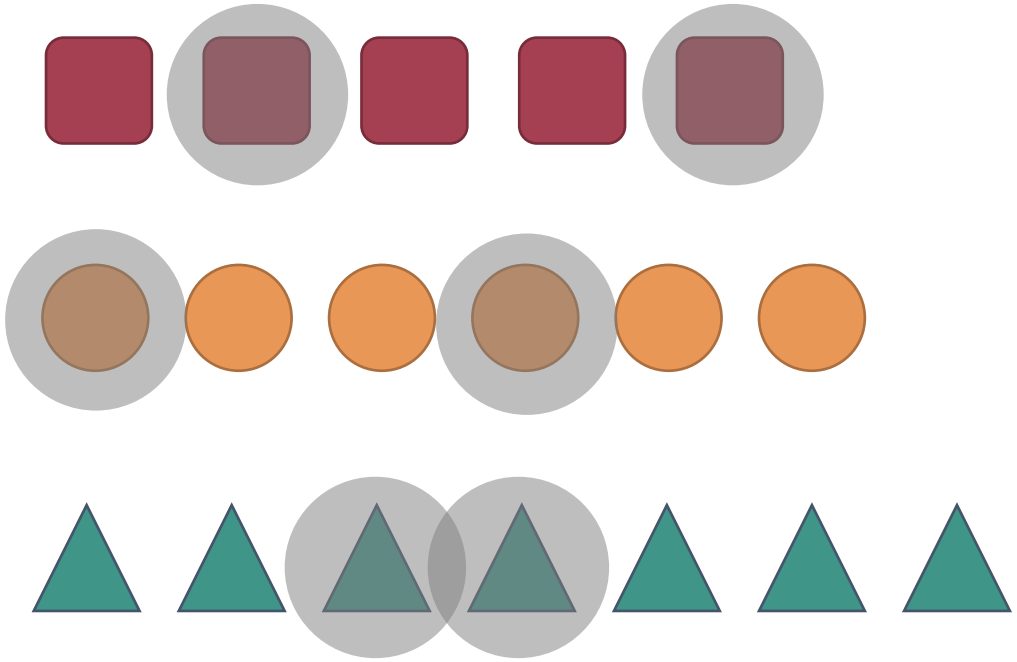
Unbalanced dataset

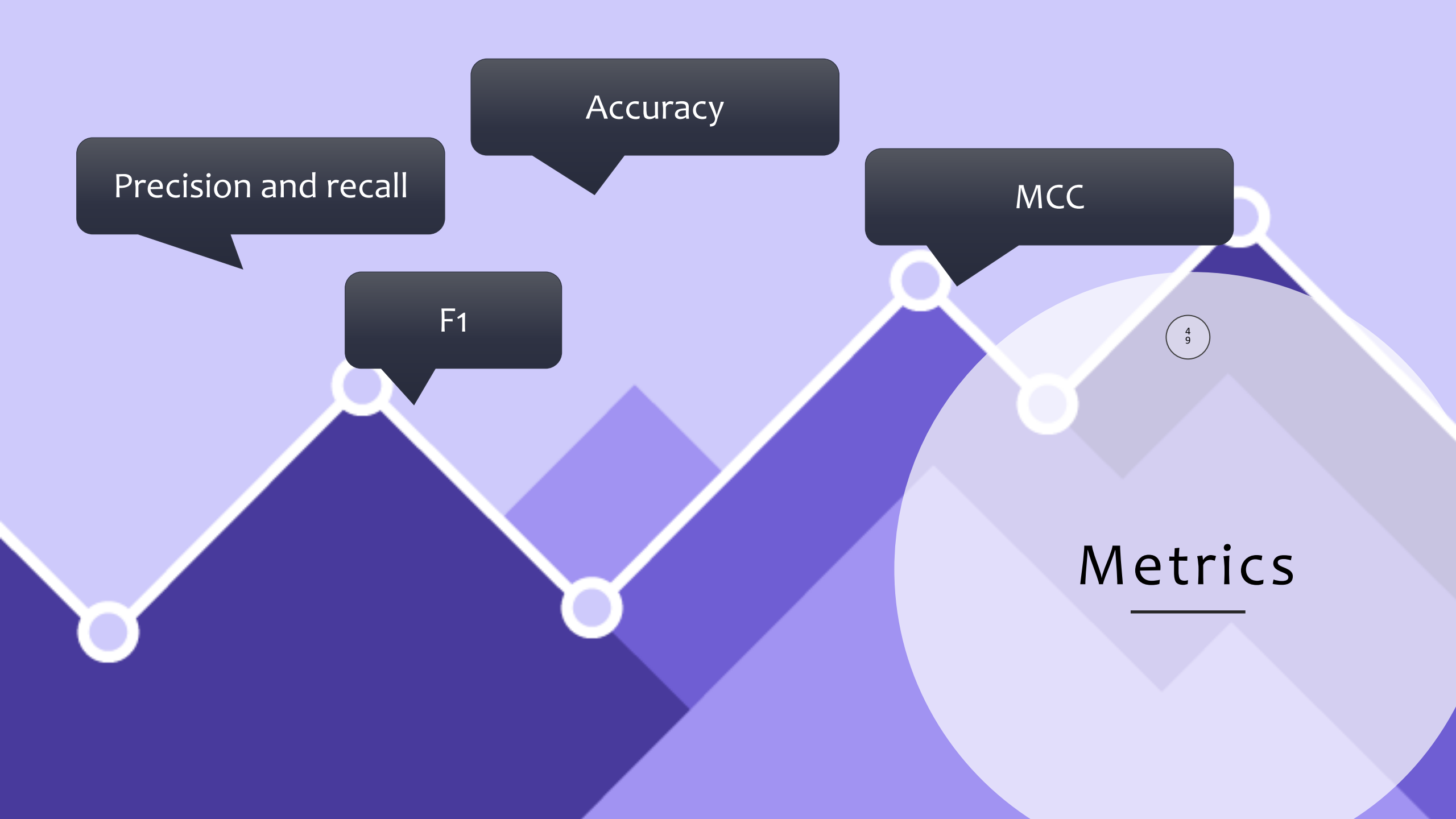


Stratified sampling

Stratified sampling is a method of obtaining a representative sample from a population that one has divided into relatively similar subpopulations (strata).



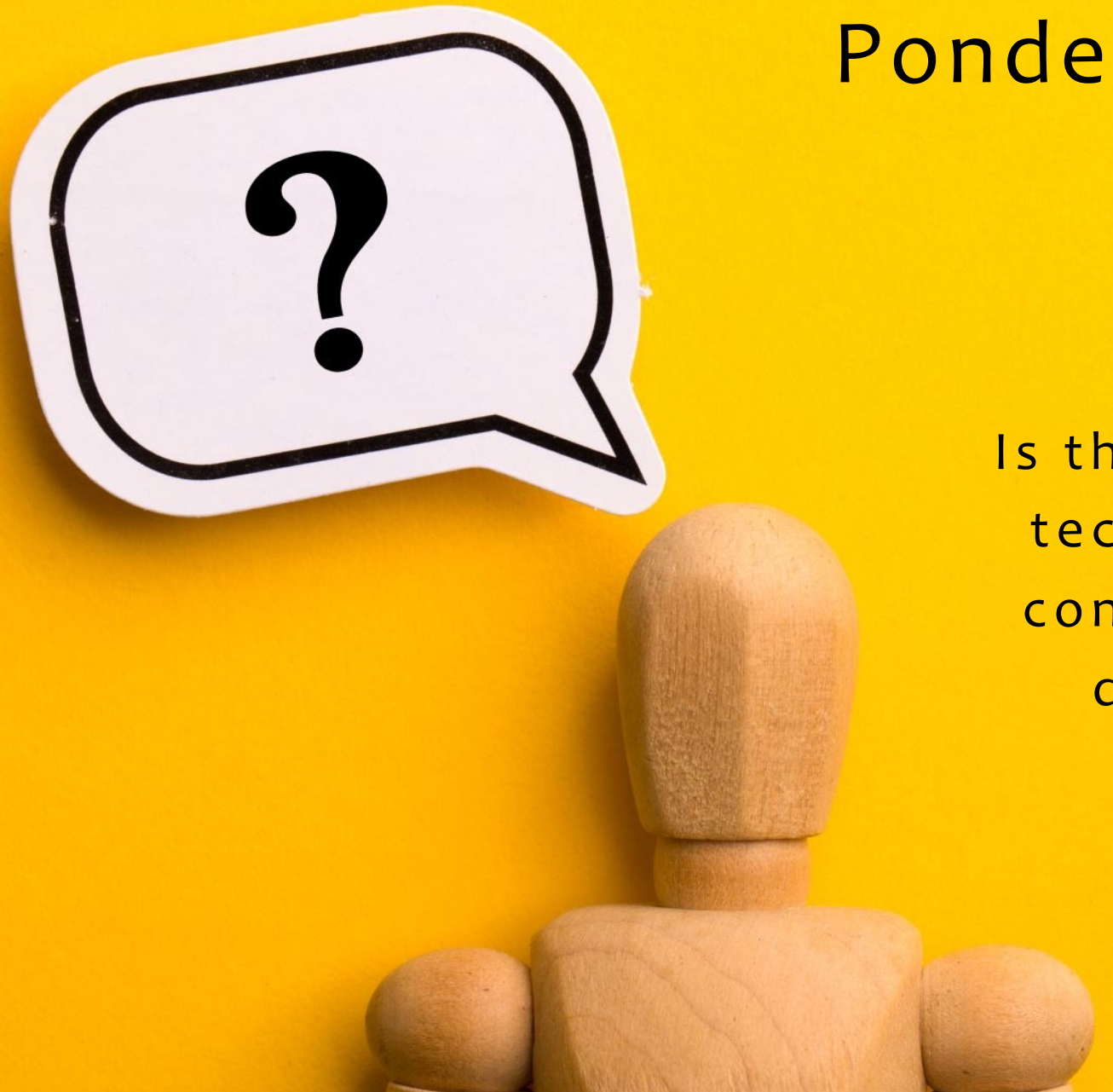



Accuracy
Precision and recall
MCC
F1
49
Metrics

Training ML models

- Balance vs imbalanced dataset
- Splitting dataset into training, validation, and test sets
- ML models
- Metrics

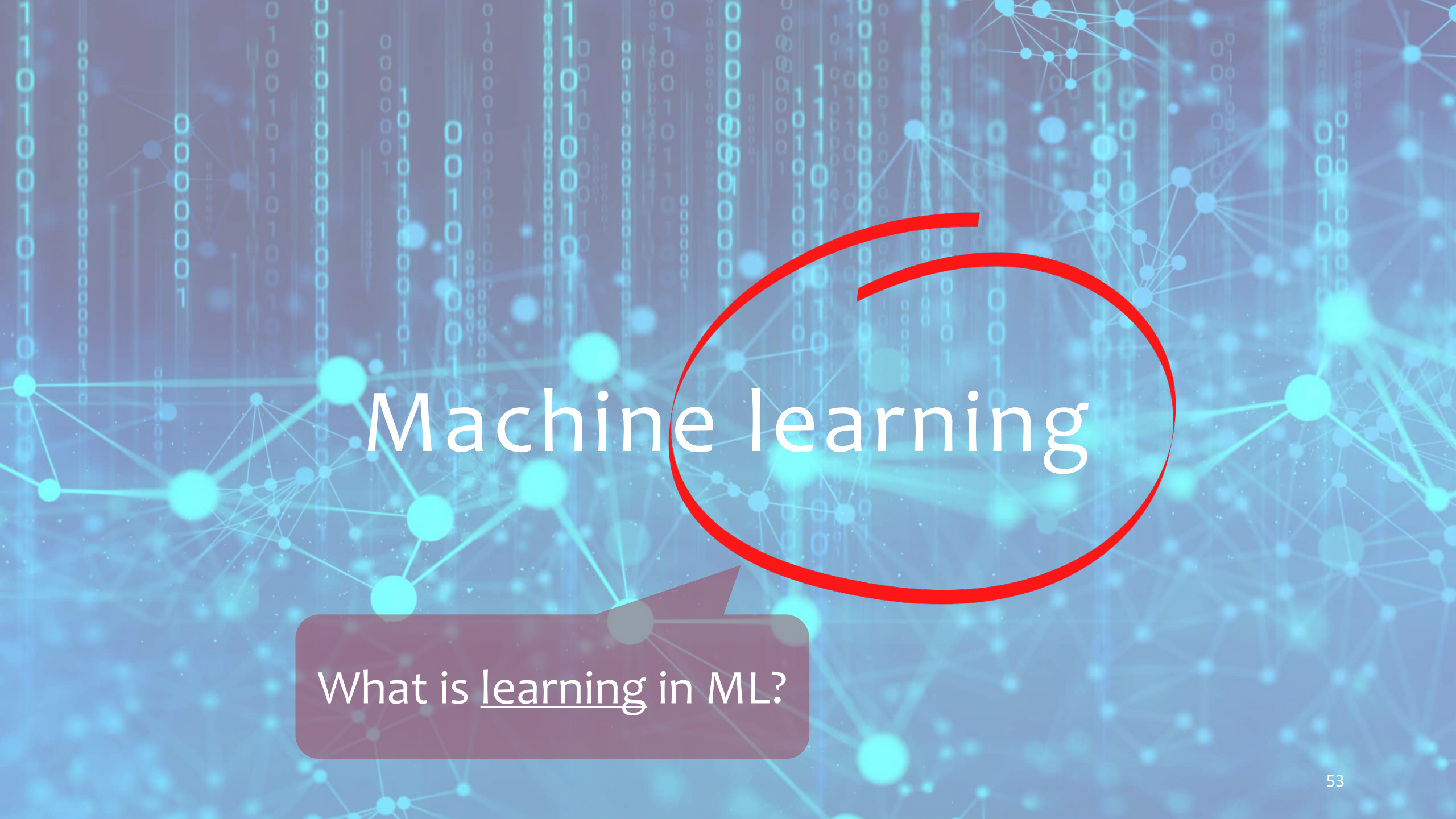
Ponder



Is there any differences in ML techniques that we need to consider to detect smells at different granularities?

Code repository

https://github.com/tushartushar/ML4SECourse


Machine learning
What is learning in ML?

Learning in ML?

A program learns from experience **E** with respect to some task **T** and performance measure **P**, if its performance at task **T**, as measured by **P**, improves with experience **E**.

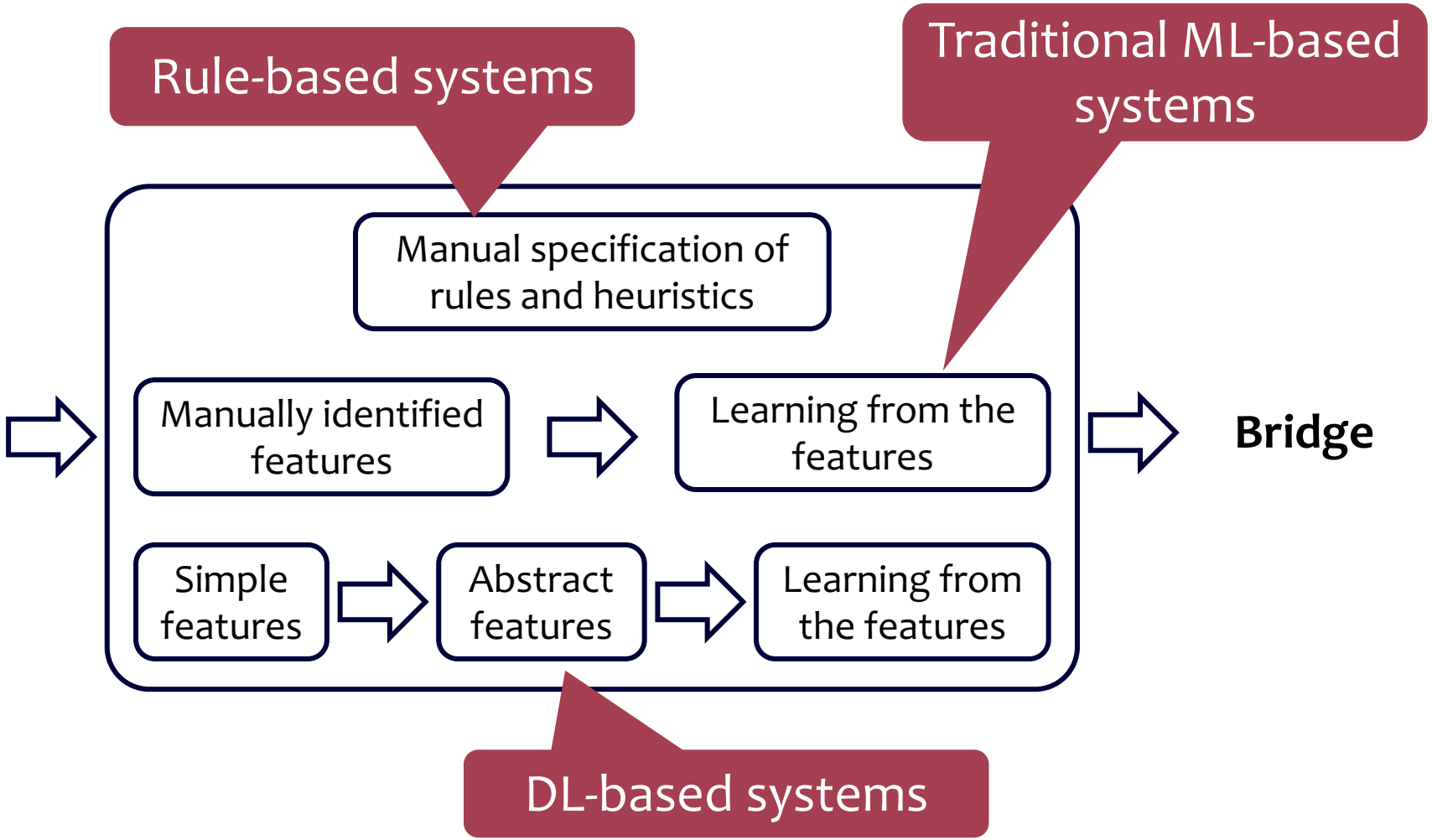
A program does better at a task, every time you run it by learning from its experience.

DL – why and what

Identify the structure

Do some processing

Bridge


Rule-based systems
Traditional ML-based systems
Manual specification of rules and heuristics
Manually identified features
Learning from the features
Simple features
Abstract features
Learning from the features
Bridge
DL-based systems

DL – Why and what

- Feature engineering for every new problem is no longer required … well, explicit specification of features
- End-to-end learning
- Encoding prior knowledge enables better performance (or faster training) via architecture (e.g. convolutions, recurrence)
- Recent advancements enabled DL:
  - Greater computational power (GPUs, TPUs)
  - Data availability

Types of neural network

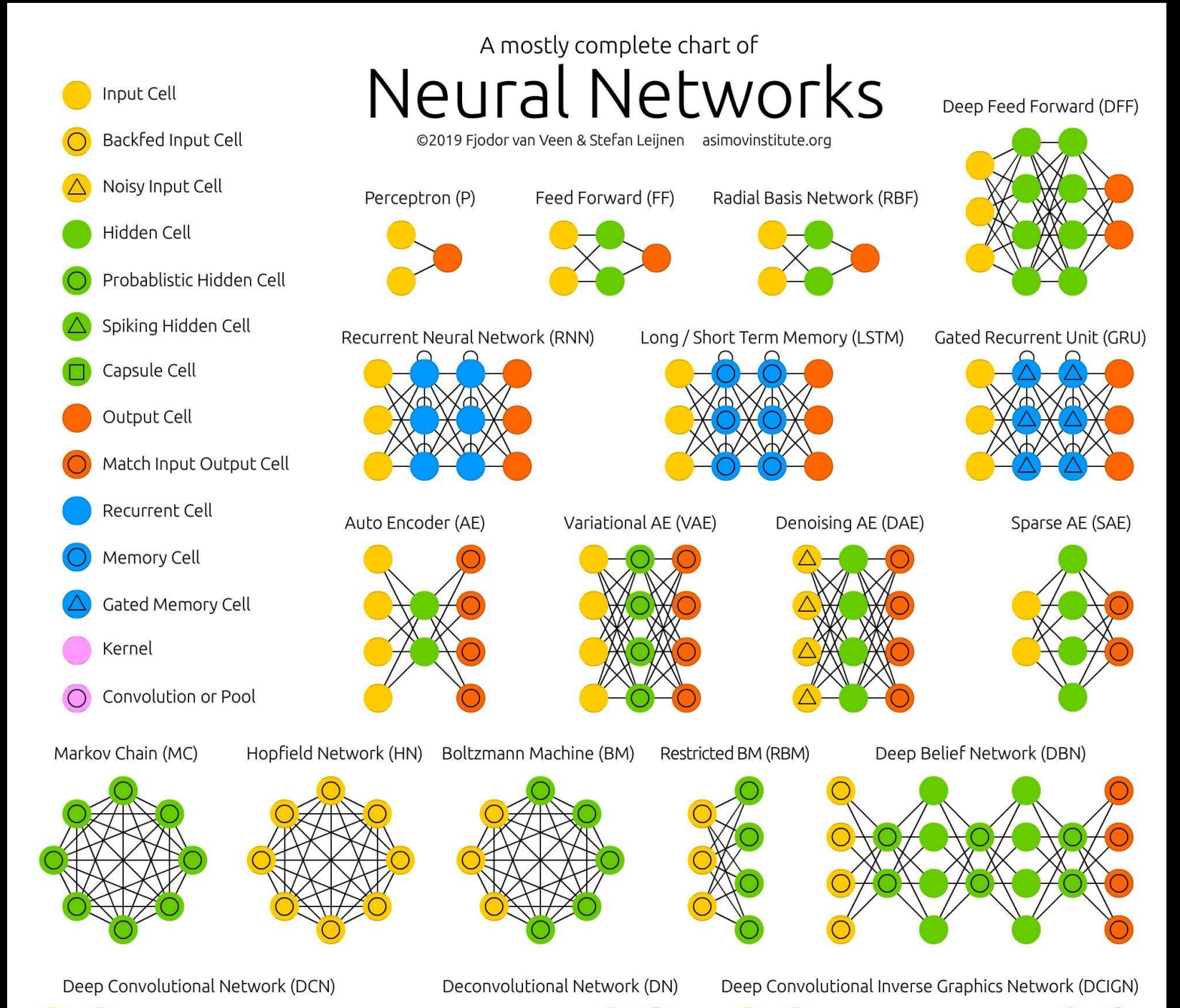


Image credit: https://ai.stackexchange.com/questions/15594/what-are-all-the-different-kinds-of-neural-networks-used-for

Convolution neural network

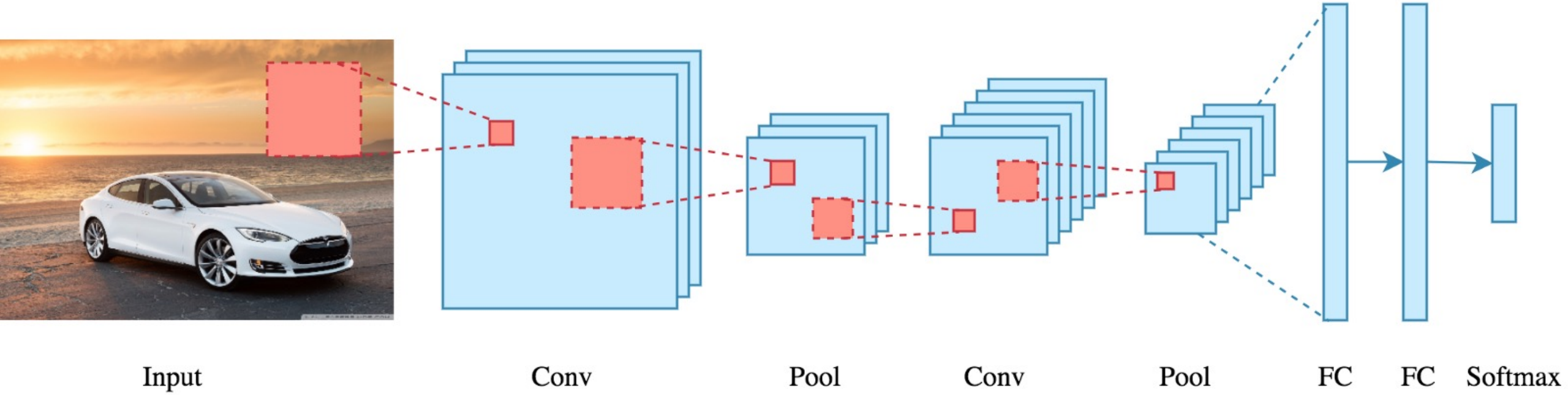
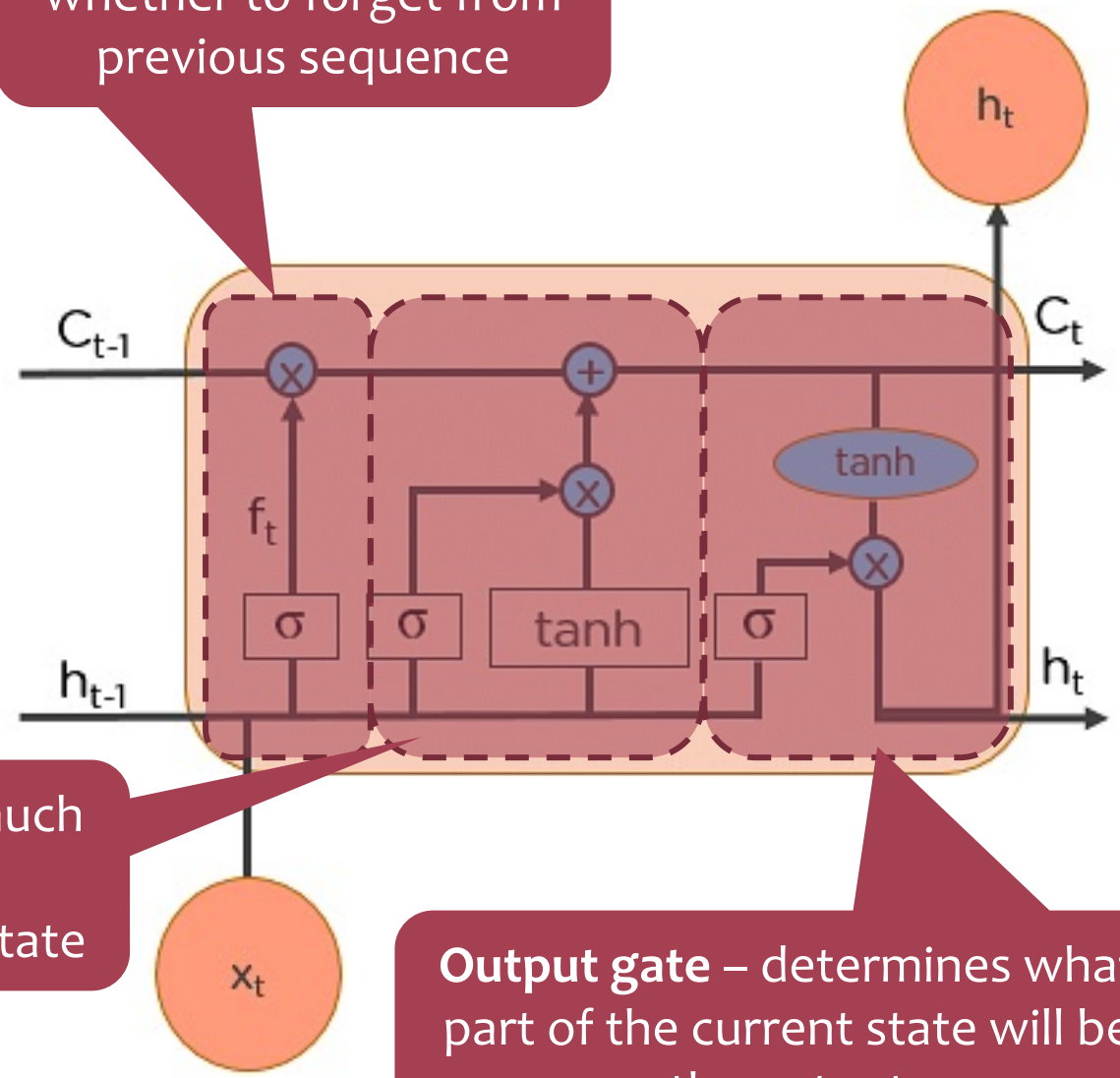
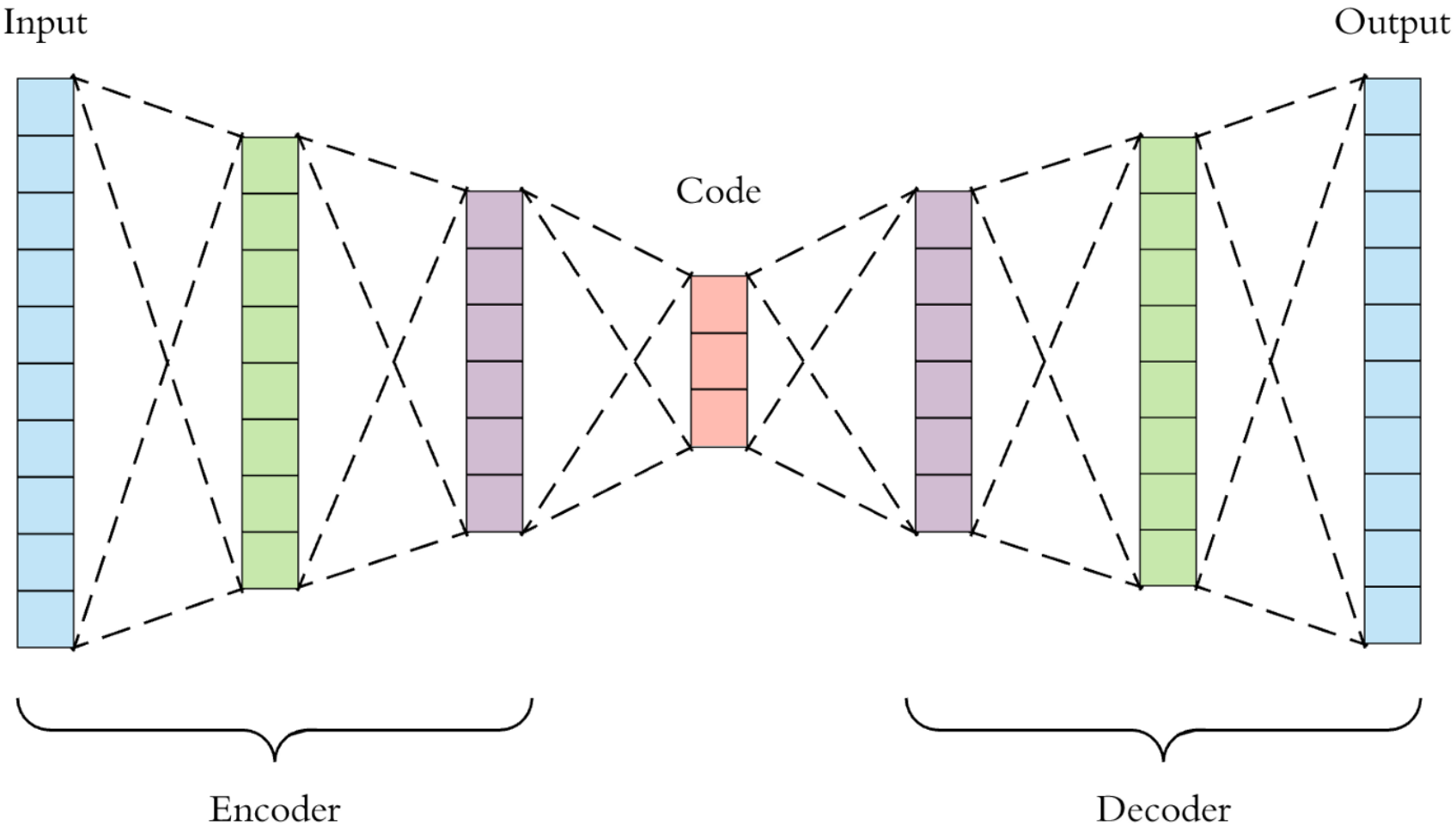


Figure credit: https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/

Long Short-Term Memory

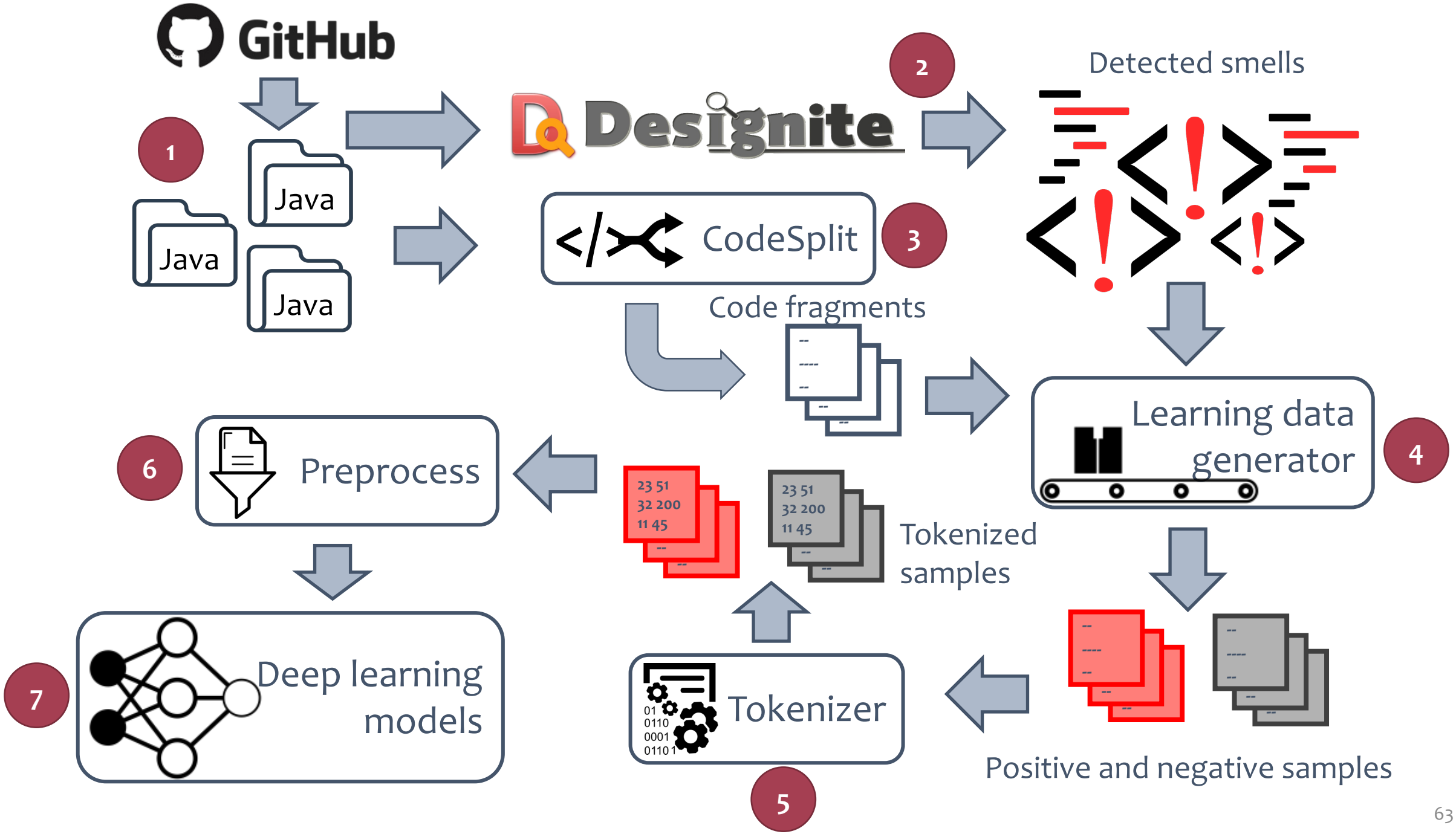




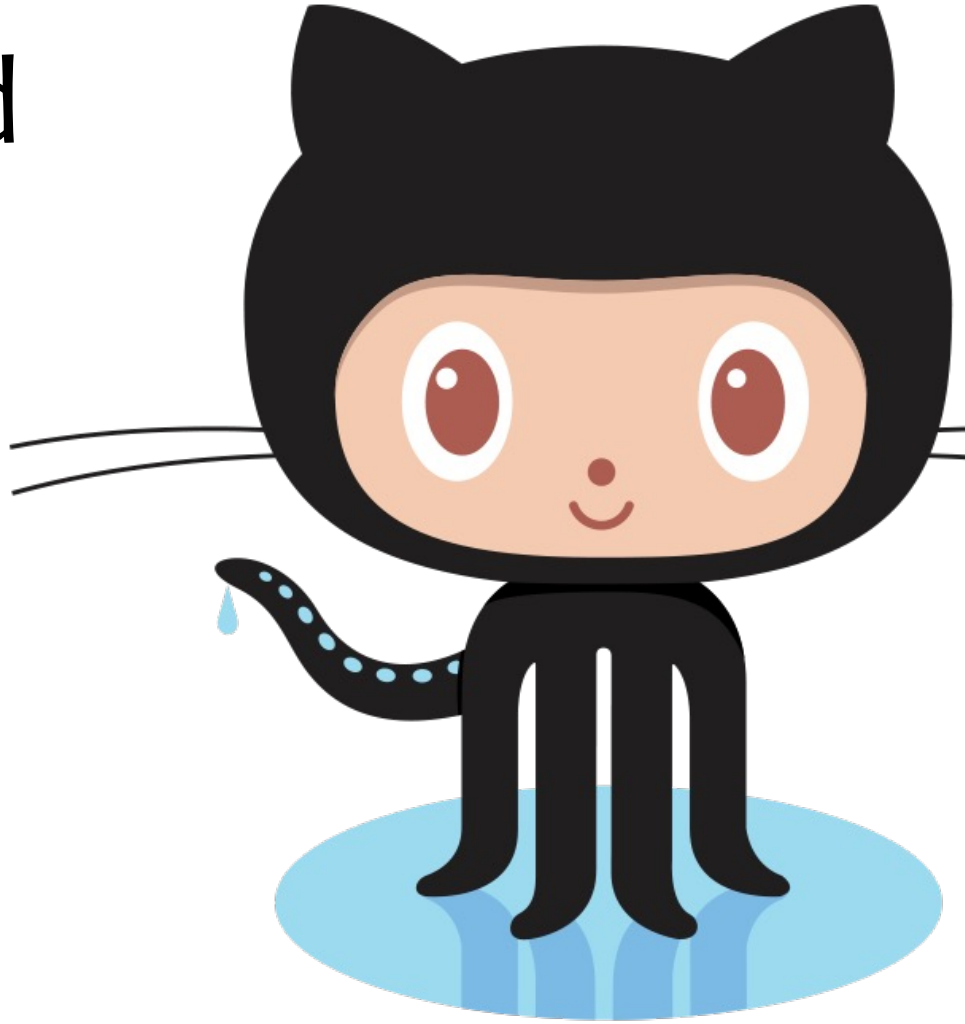
Autoencoder

- Input is same as output
- Input is summarized into a lower-dimensional representation also referred to as latent-space representation.

Part 3: Detect code smells using DL


GitHub
1
Java
Java
Java
Designite
2
Detected smells
CodeSplit
3
Code fragments
Learning data generator
4
Positive and negative samples
Tokenizer
5
Tokenized samples
23 51
32 200
11 45
Preprocess
6
Deep learning models
7

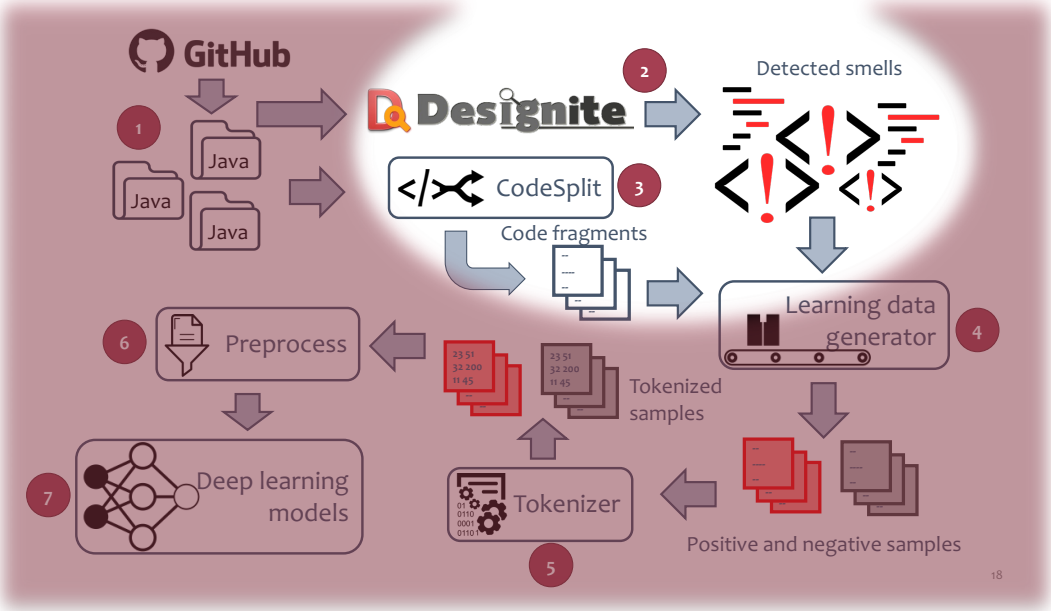
Identify and download GitHub repositories



I assume by now you know

repo_download.ipynb

Analyze code



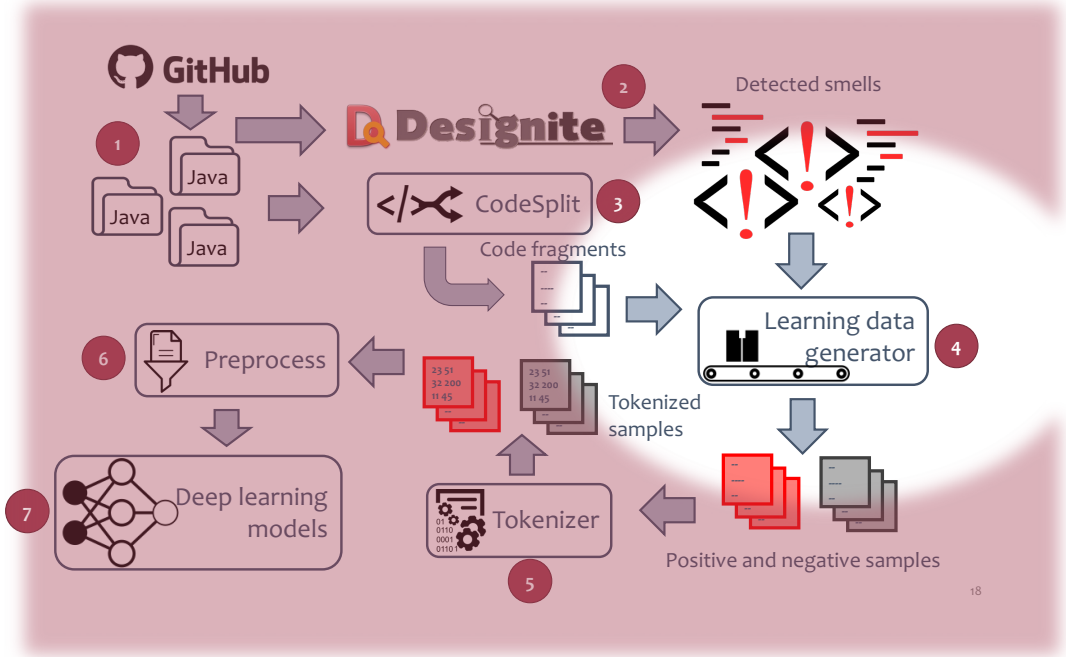
Analyze all downloaded repositories

- DesigniteJava
  - to identify smells
- CodeSplitJava
  - to split individual classes (or methods) into separate files

jupyter analyze_code.ipynb

jupyter analyze_code_codesplit.ipynb

Creating training dataset

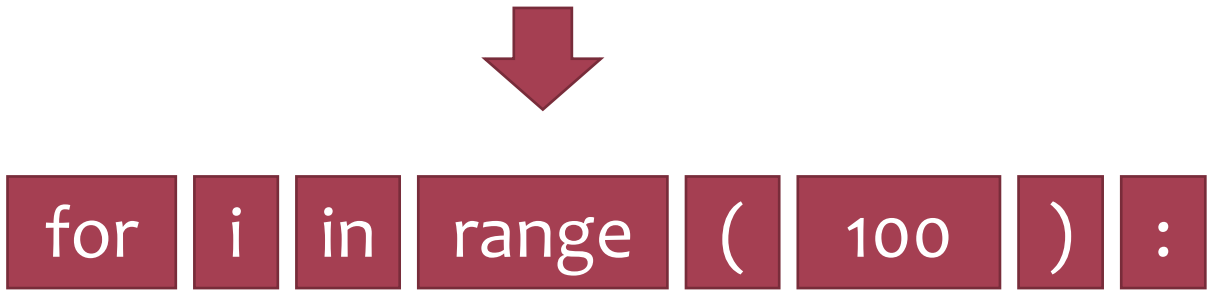


Process the smell data and segregate code snippets generated by CodeSplitJava tool into smelly and non-smelly sets

jupyter create_dataset_DL.pynb

Tokenization

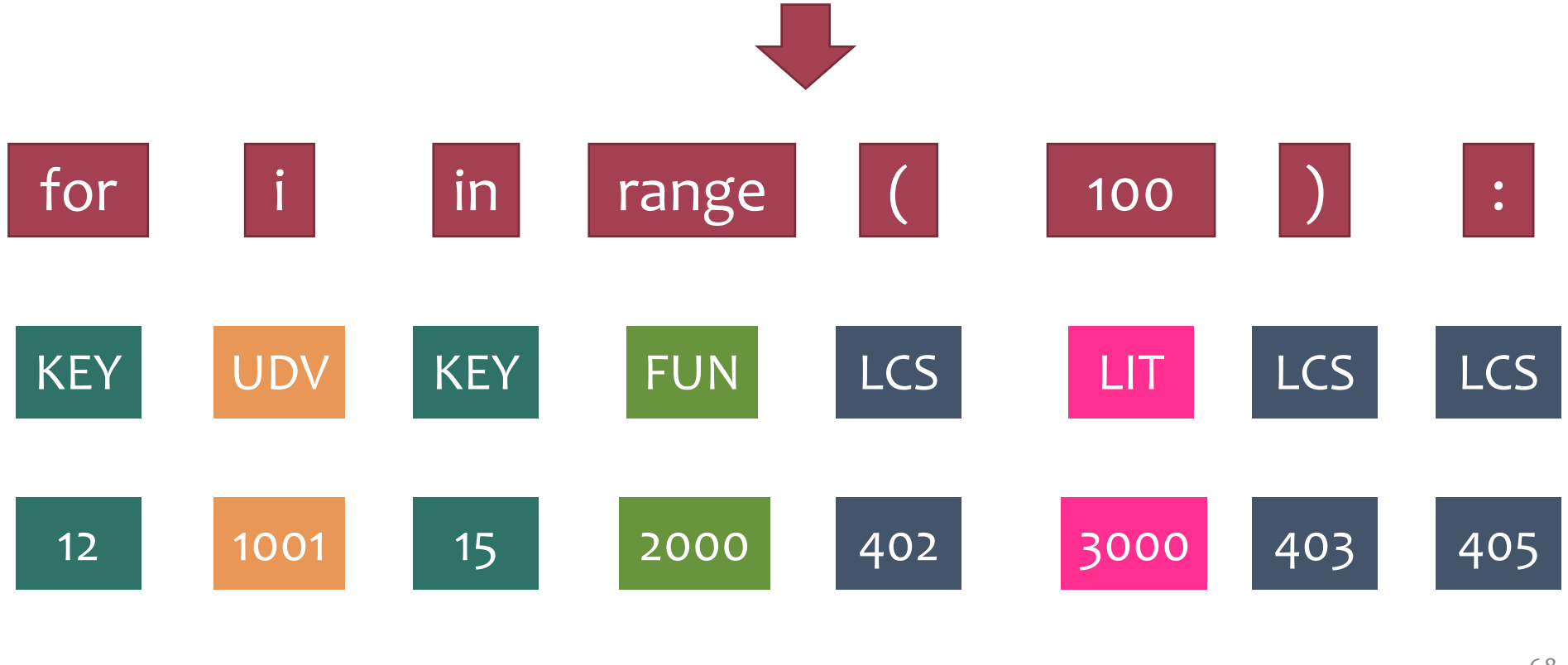
for i in range(100):



Tokenization is a process to split a sequence (of words, for instance) into a set of unit of interest (or tokens).

Tokenization

for i in range(100):

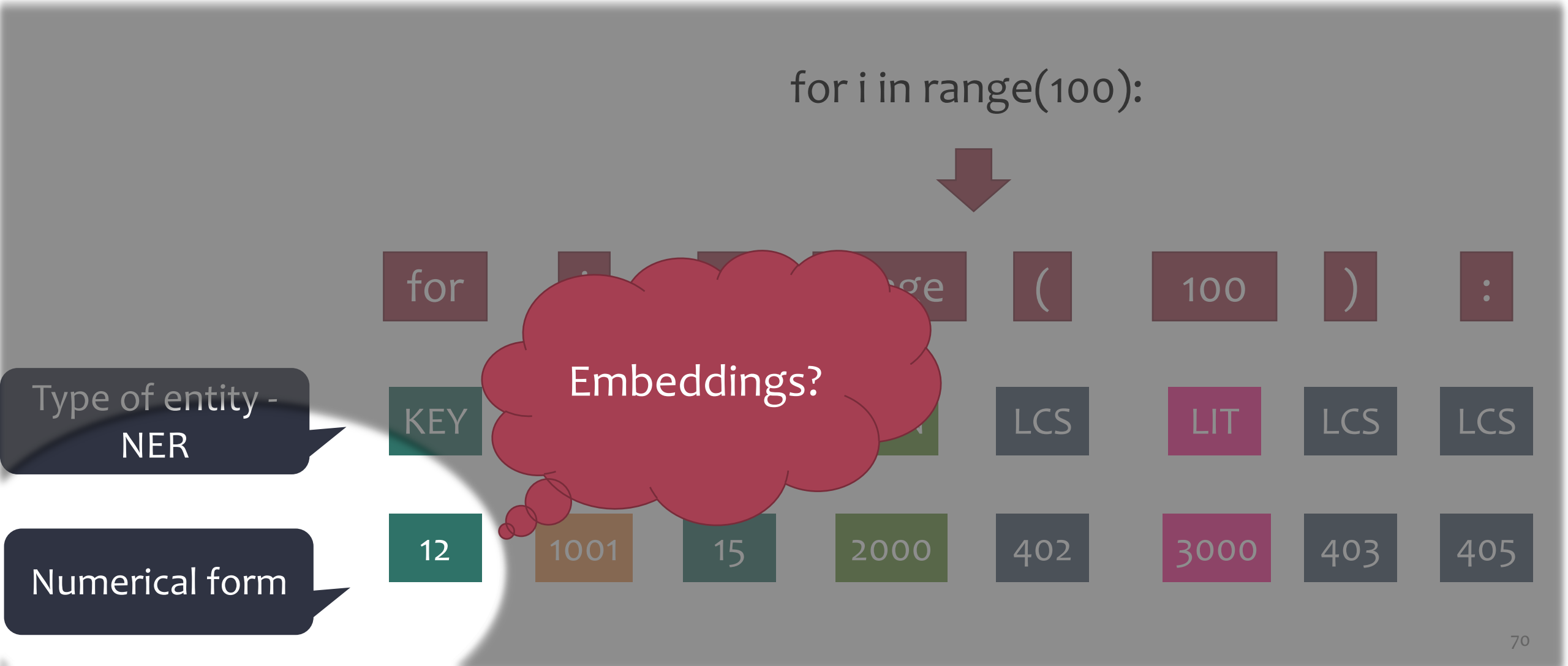


Tokenize

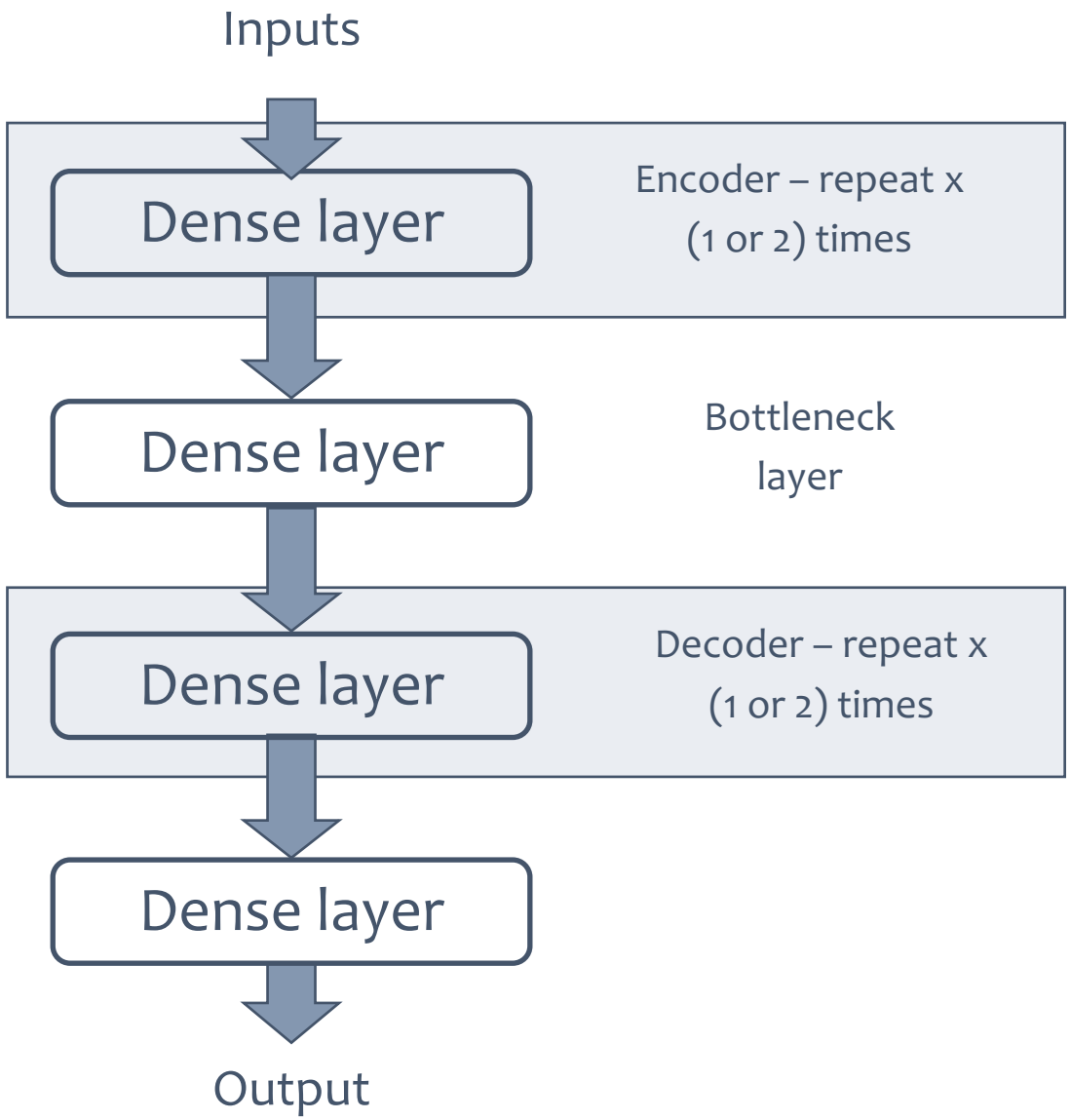
```
public void InternalCallback(object state)
{
    Callback(State);
    try
    {
        timer.Change(Period, TimeSpan.Zero);
    }
    catch (ObjectDisposedException)
    { }
}
```

123 40 2003 41 2002 59 474 2005 2006 2007 2008 123 46 40 44 46 41 2004 59 125 40 2009 329 41 123 125 125

Tokenization



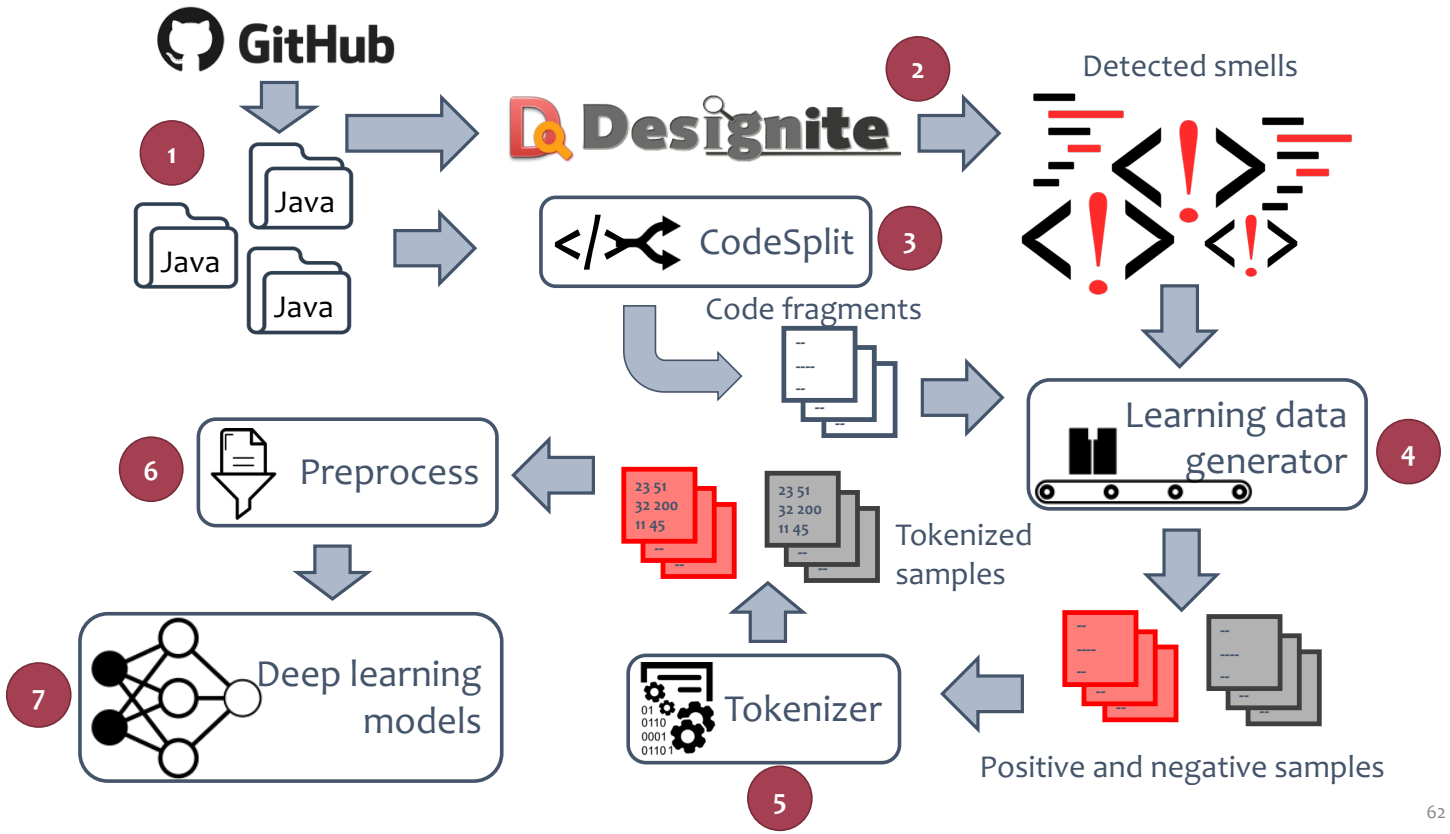
Train Autoencoder



jupyter Output Train_autoencoder.ipynb

Discussion and take away

- Various moving parts (data collection, model design, data preparation, inference)
- Understanding the tradeoffs (compute time, performance, time/effort to clean dataset, …)
- Role of code representation and feature engineering
- Role of LLMs for code



62

References

- A survey on software smells. https://www.sciencedirect.com/science/article/pii/S0164121217303114
- A taxonomy of software smells https://www.tusharma.in/smells/
- Code smell detection by deep direct-learning and transfer-learning https://www.sciencedirect.com/science/article/abs/pii/S0164121221000339
- A Survey on Machine Learning Techniques for Source Code Analysis https://arxiv.org/abs/2110.09610

THANK YOU

Dr. Tushar Sharma

tushar@dal.ca