# An empirical study on the characteristics of reusable code clones


**Chunli Yu**

**chunli.yu@queensu.ca**

Haoxiang Zhang

haoxianghz@gmail.com

Ying (Jenny) Zou

ying.zou@queensu.ca

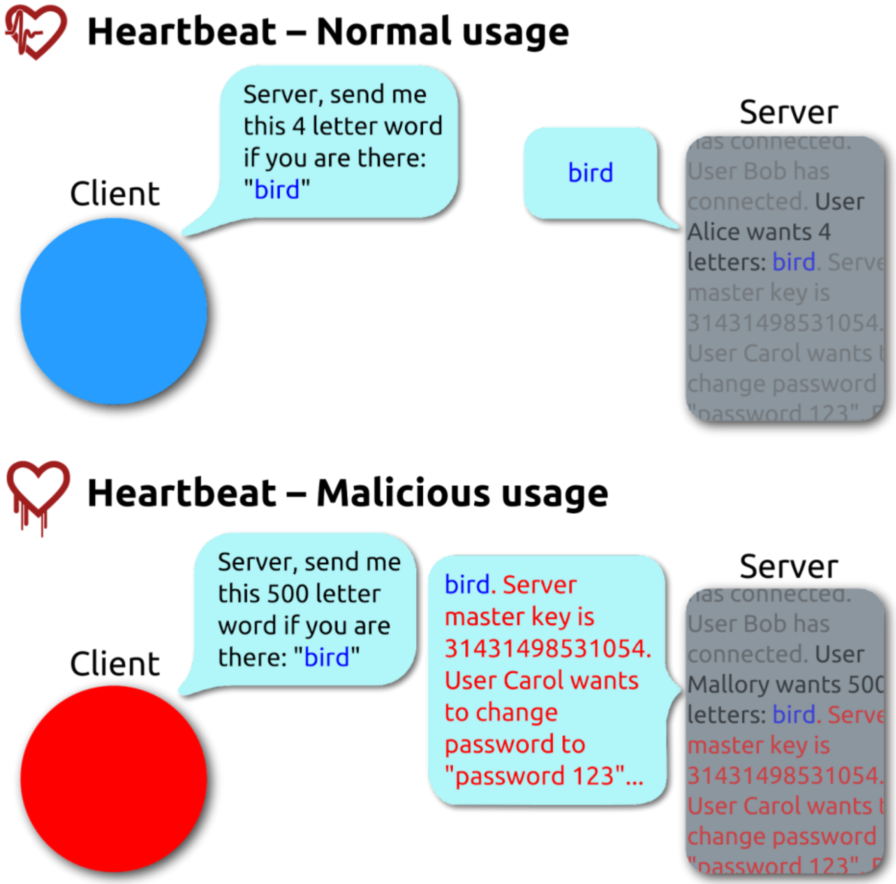
Queen's UNIVERSITY

Software Evolution & Analytics Lab



Consortium for Software Engineering Research (CSER) 2023 Spring Meeting, June 7, 2023

# Inappropriate code cloning presents both financial and reputation loss for the organization




memcpy(bp, pl, payload);


- At least 3,156 projects have similar copies of Heartbleed buggy code.

# Code clones are identical or similar code fragments

### Original source

```
void sumProd(int n) {
    float sum = 0.0;
    float prod = 1.0;
    for (int i = 1; i<=n; i++) {
        sum = sum + i;
        prod = prod * i;
        foo(sum, prod);
    }
}
```

### Clone Type 1

```
void sumProd(int n) {
  float sum = 0.0; //C1
  float prod = 1.0; // C2
  for (int i = 1; i <= n; i++) {
  ____      sum = sum + i;
  ____      prod = prod * i;
  ____      foo(sum, prod);
  }
}
```

- spaces and comments are added.

### Clone Type 2

```
void sumProd(int n) {
  int s = 0; //C1
  int p = 1; // C2
  for (int i = 1; i <= n; i++) {
  ____      s = s + i;
  ____      p = p * i;
  ____      foo(s, p);
  }
}
```

- spaces and comments are added.
- variable names and types are changed

### Clone Type 3

```
void sumProd(int n) {
  int s = 0; //C1
  int p = 1; // C2
  for (int i = 1; i <= n; i++) {
  ____      s = s + i * i;
  ____      foo(s, p);
  }
}
```

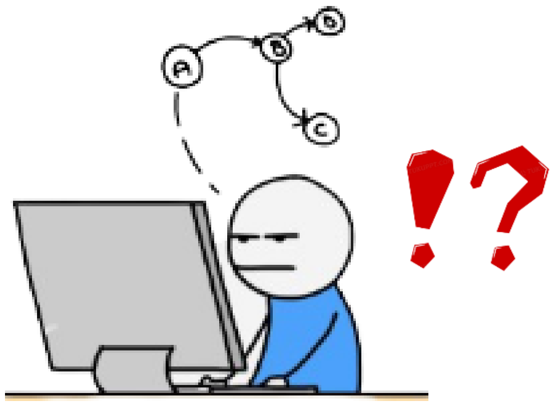
- spaces and comments are added.
- variable names and types are changed
- statements are deleted, modified

- Code clones are created by copy-and-paste activities.

**Often 20% - 30% redundancy**

# Code cloning makes software maintenance difficult


Developers face a large amount of code clones to manage.

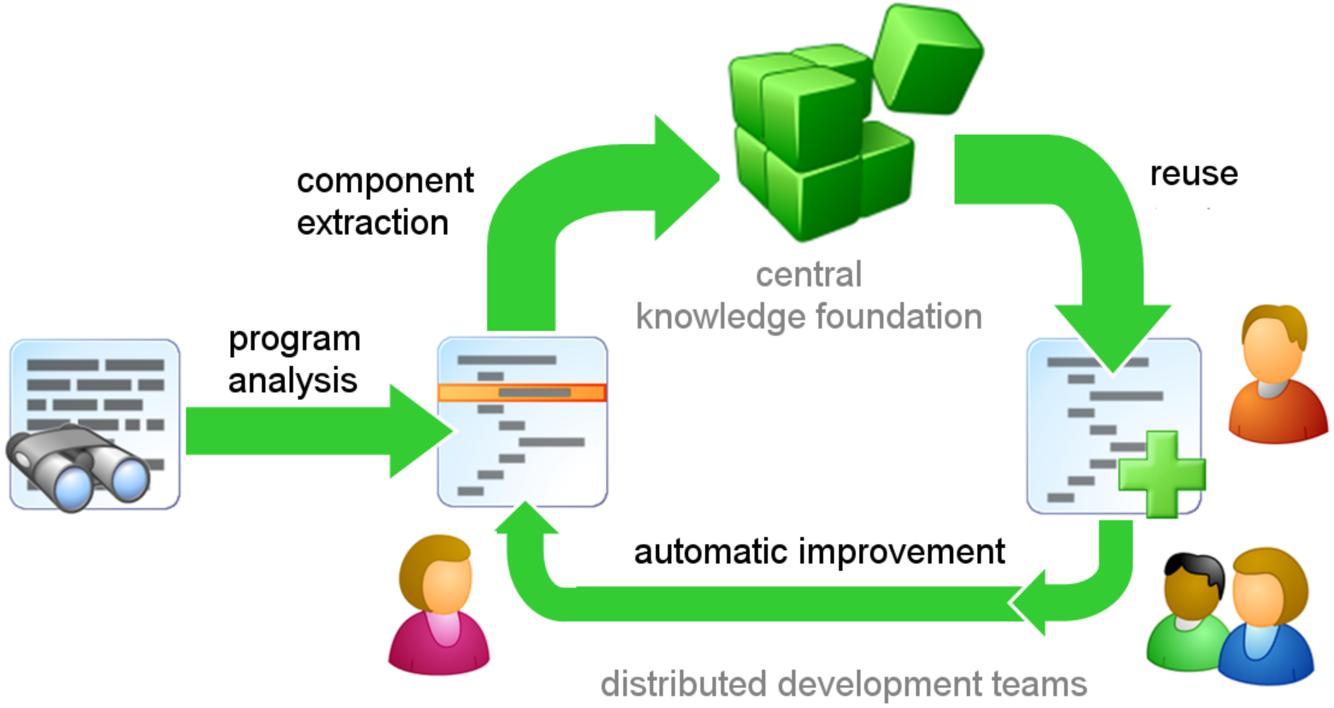


- Most developers are contributing voluntarily
- Focus is towards fixing the issues
- Clone detection tools produce a large amount of clones
- It is not possible to check all clones

# Limitation of existing approaches to improve clone quality

- Several prior studies attempt to identify problematic clones for refactoring.
- The prior studies provide reuse suggestions mainly based on clone prevalence and/or API-usage related to clones.
- However, reusing code clones reliably from the quality perspective remains unstudied.

# Reusing code clones to improve code quality




To assist development and maintenance team in enhancing code quality, we aim to pinpoint high-quality code clones for reliable reuse.

# Research Challenges

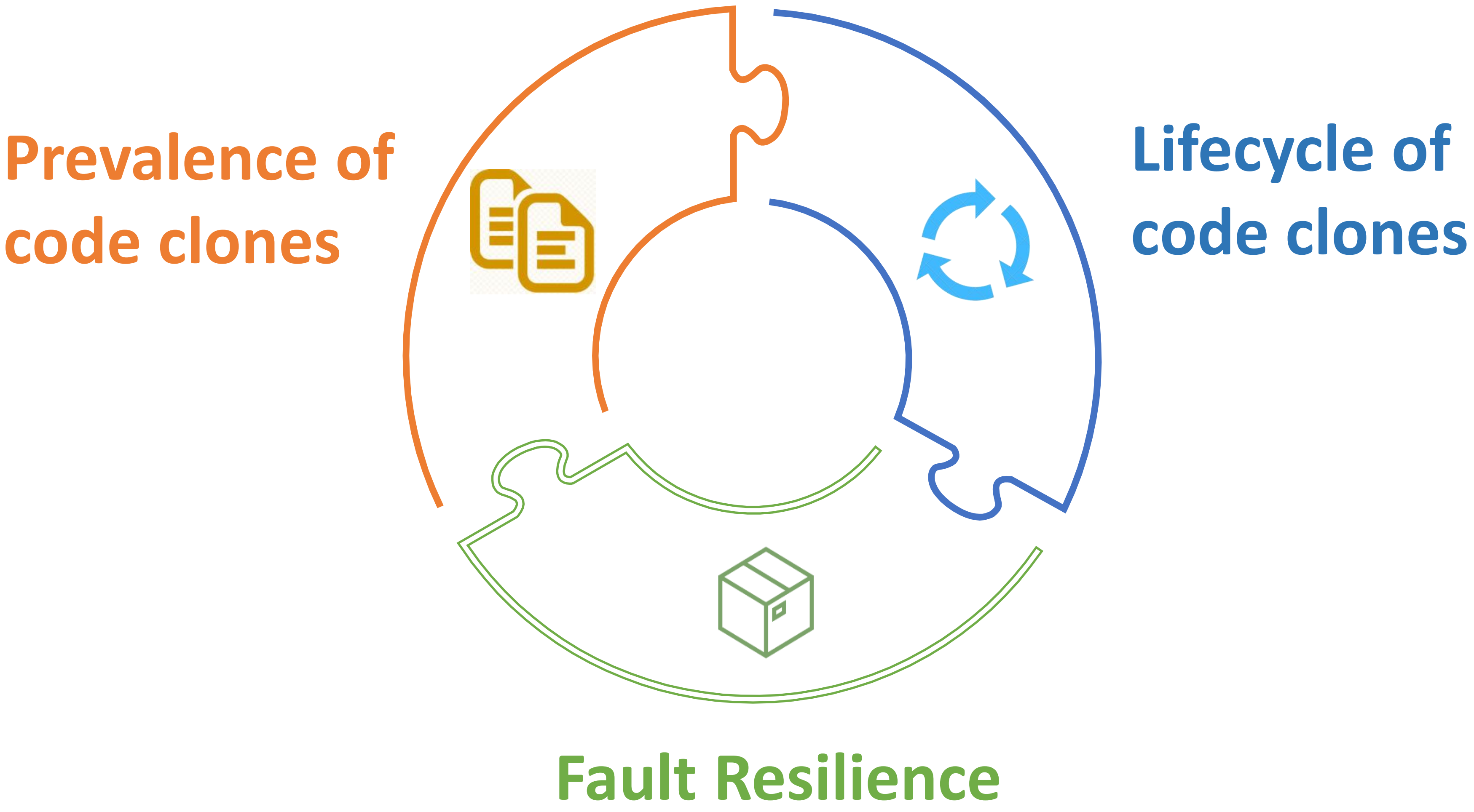
- Difficult to manually examine clones to create a labeled dataset.
- The manually labelled results could be subjective and unreliable.

# Applying classification models to identify reusable code clones

- Automatically classify the clones for more reliable reuse considering both functional requirements and non-functional requirements

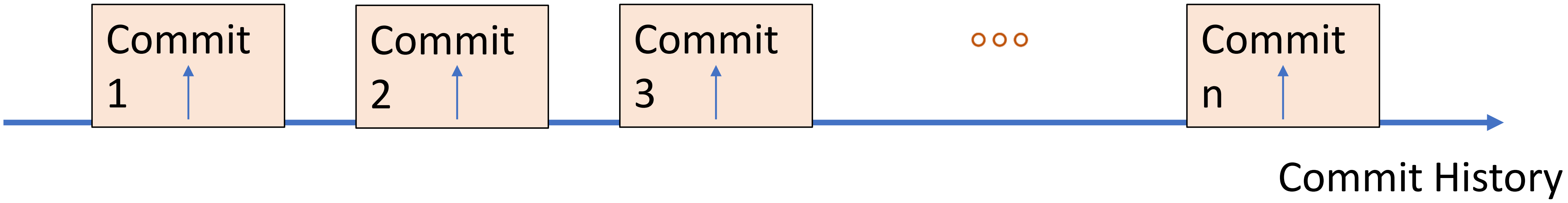
- Prioritize reusable clones for immediate attentions in quality improvement

# Criteria for selecting reusable clones



# Clone longevity

Clones that survive for a longer time period imply higher reusability.



# Clone prevalence

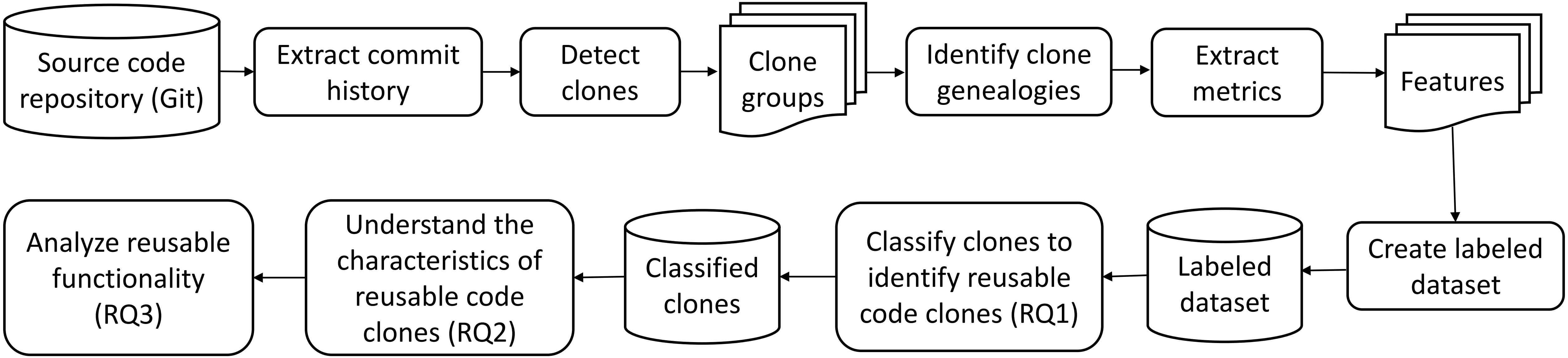
- Refers to the frequency of reusing code fragments.
- Assesses functionality usefulness.
- Calculated by the number of clone siblings.

# Clone fault-resilience

- Reusing bug-prone clones can harm the quality of a project.
- Frequent buggy changes present signs of inferior code duplicates that are to be sifted out.

$$\text{Fault-resilience} = \frac{\#\ Non\text{−}buggy\ commits}{\#\ \text{Buggy}\ commits}$$

# Data collection approach



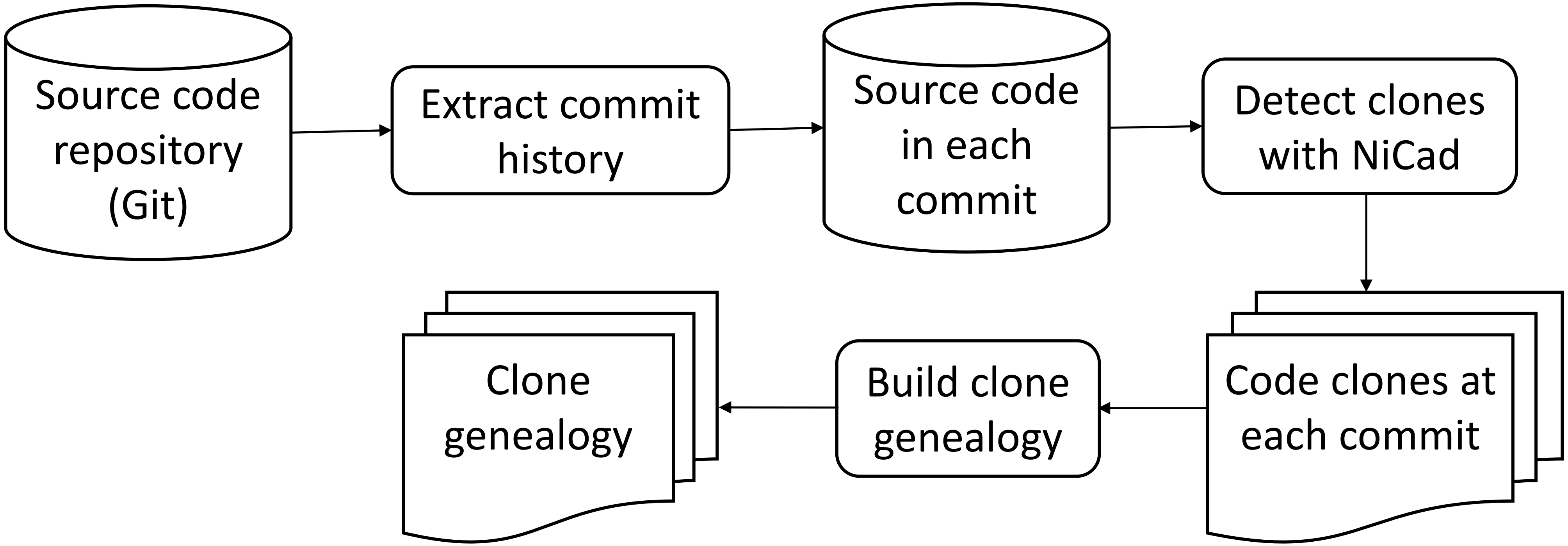
# Criteria for selecting subject systems

- Github Java projects
- Commits > 1,000
- Issues > 1,000
- Pull requests > 1,000
- Source lines of code (SLOC) > 100,000

In total, we have 27 subject systems

# Detecting code clones

- NiCad clone detector is leveraged to detect Type I, Type II, Type III code clones on method level.



# Calculating clone metrics

## Clone Metrics

# clone instances
# of followers
# length of common paths

## Product Metrics

LOC

Complexity

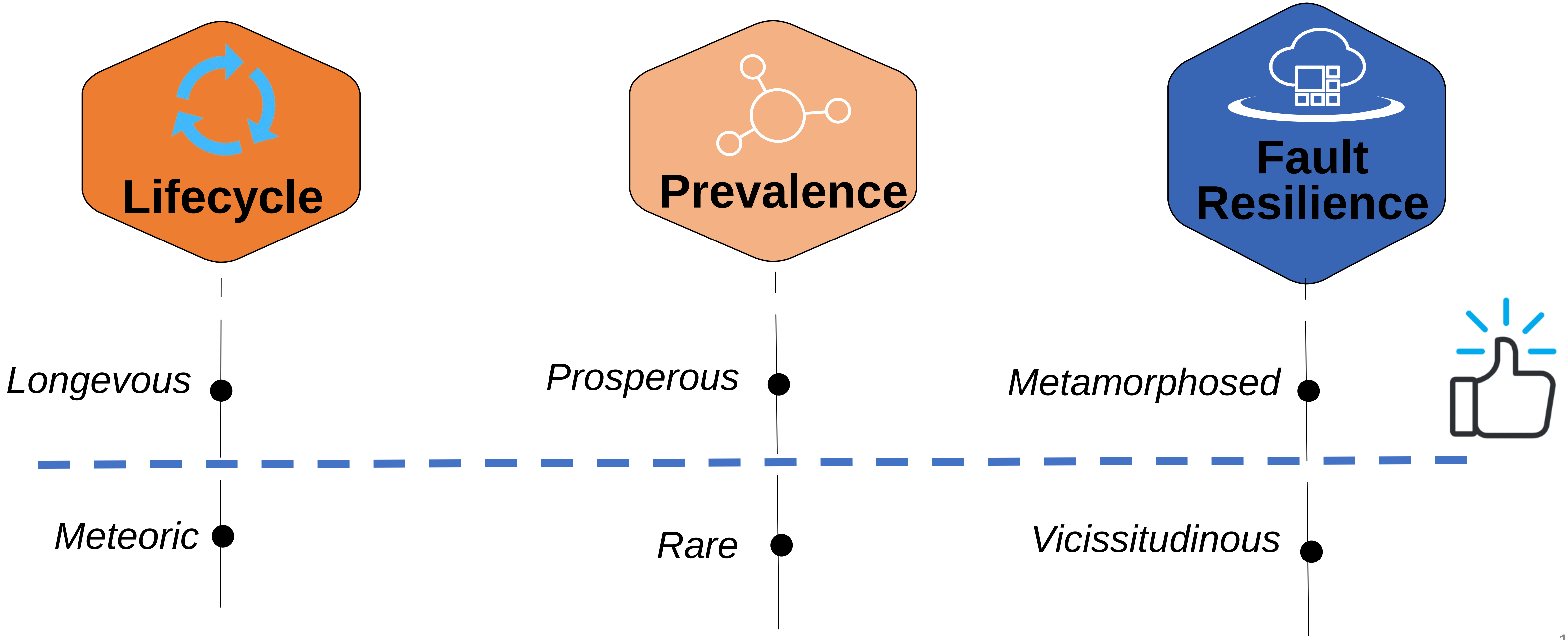
# Fanin
# Fanout
...

## Process Metrics

# contributors

...

# Constructing training data set



# Building classification models

- Tree-based classifiers are leveraged in our case study.
  - Decision Tree
  - Random Forest
  - XGBoost
  - CatBoost
  - LightGBM
  - AdaBoost

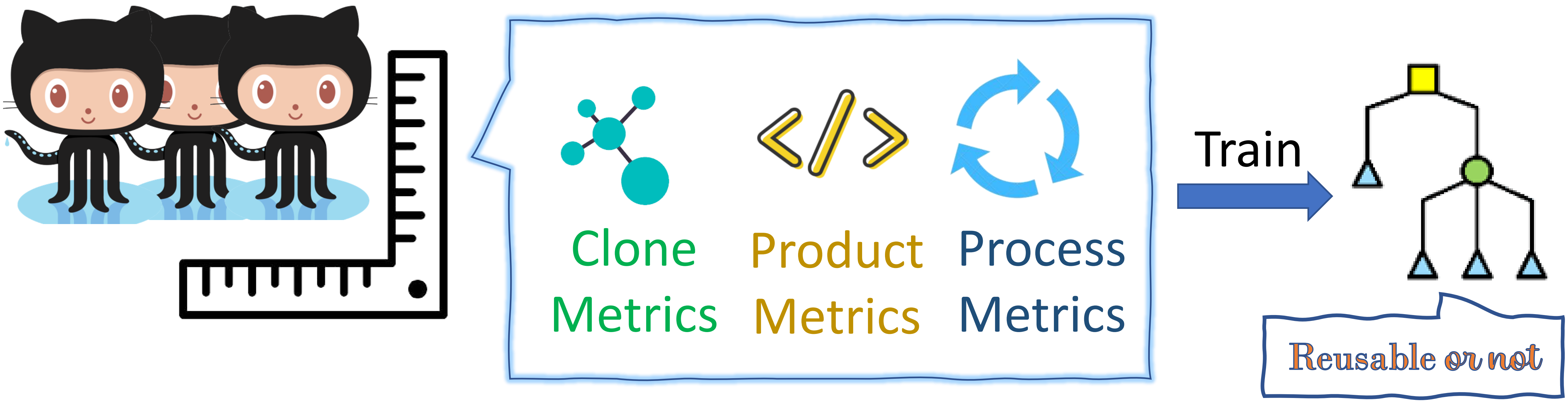
- 10 * 10 cross validation is used to fine-tune our models.

# Research questions (RQ)

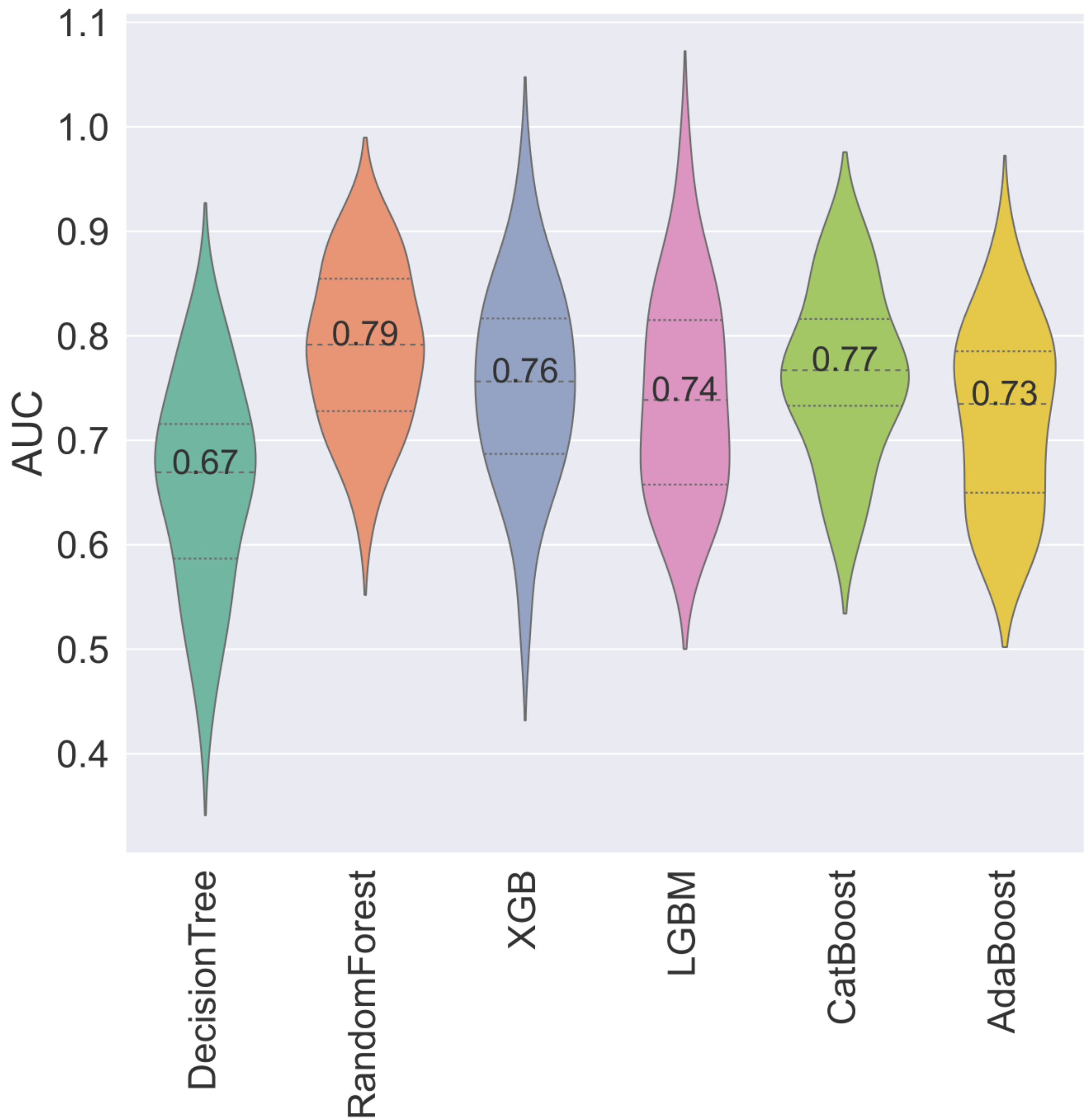
**RQ1**

How well can we classify the reusable code clones?

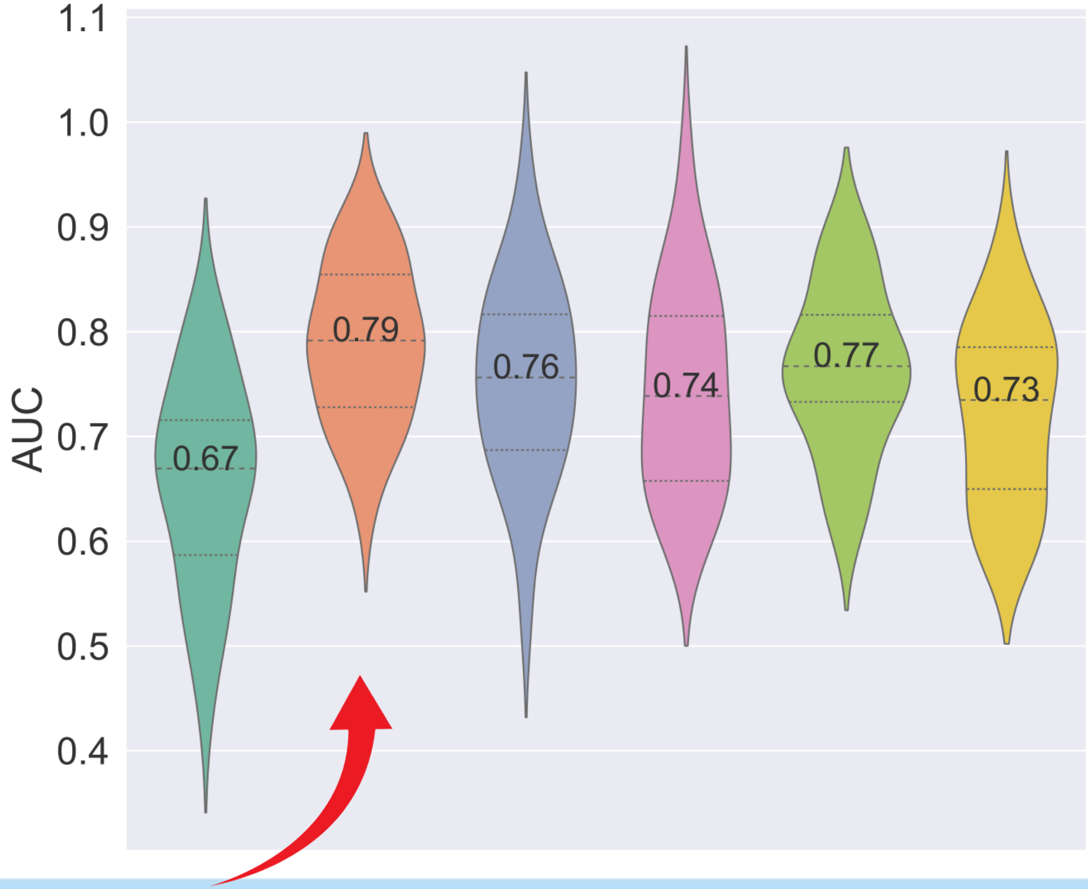
# RQ1.How well can we classify the reusable code clones



# RQ1.The AUC achieved by ML classifiers in classifying the reusable code clones

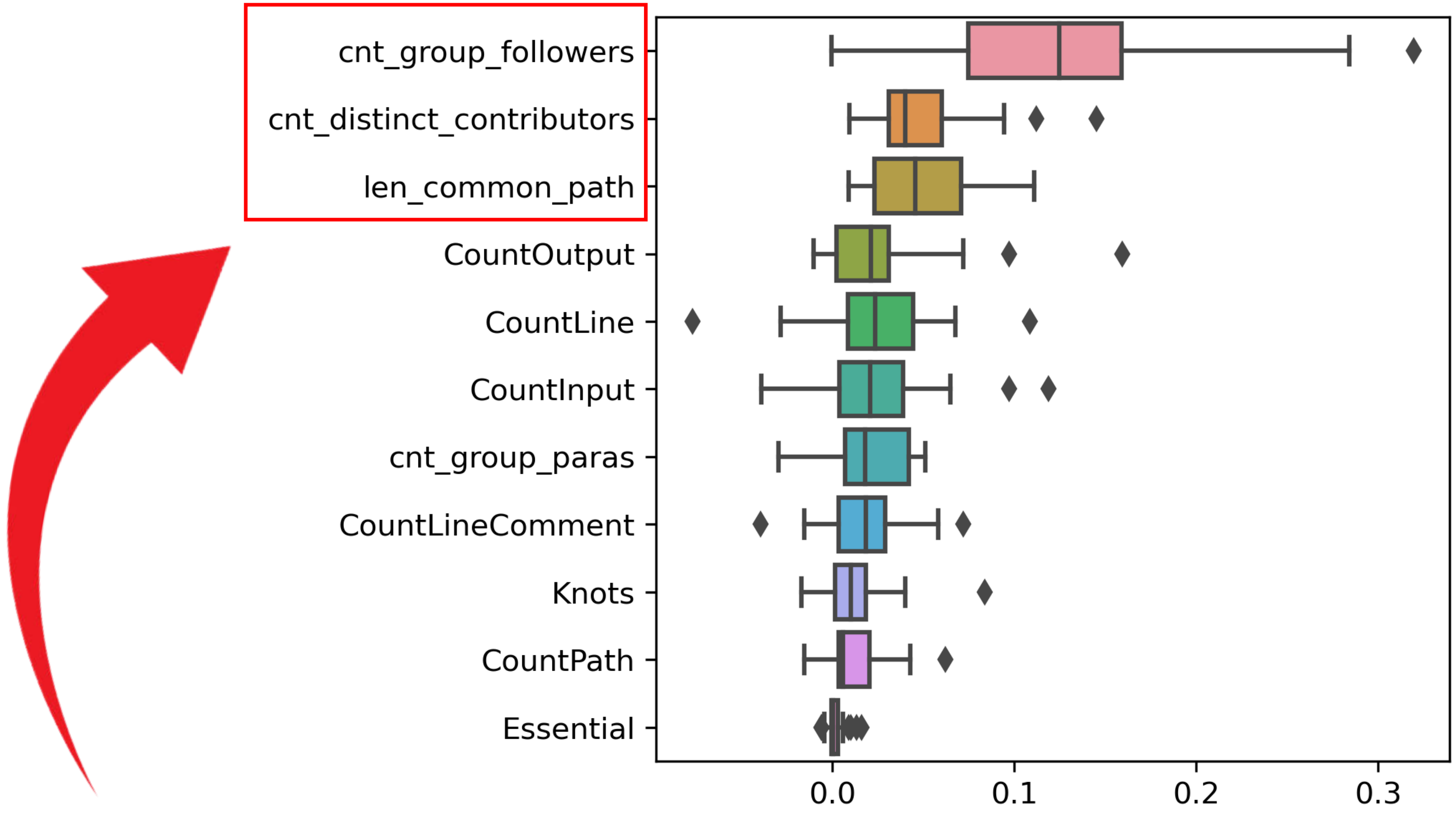


# RQ1.The AUC achieved by ML classifiers in classifying the reusable code clones



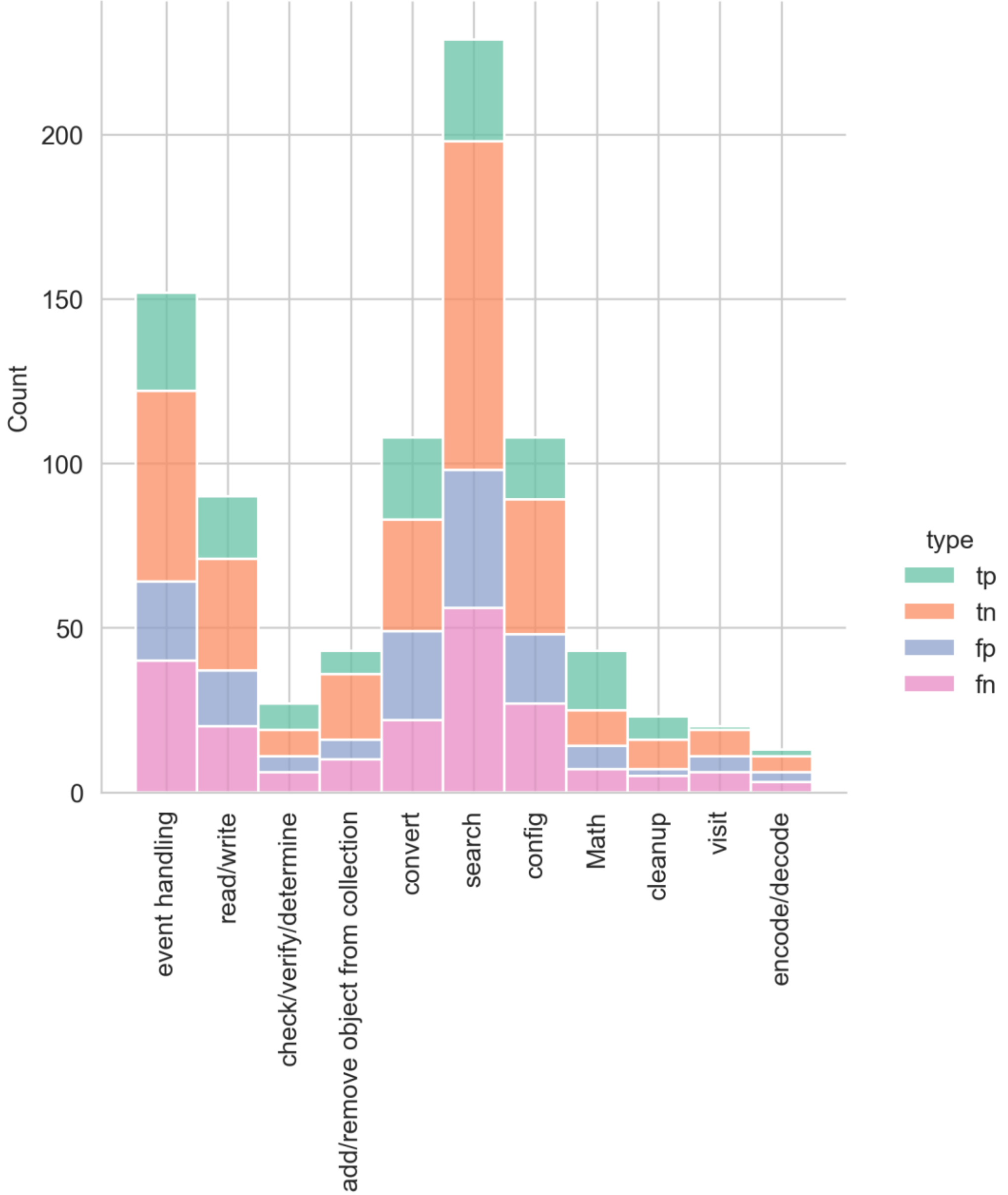

Our Random Forest classifier achieves the best AUC (i.e., 0.79) in determining reusable code clones.

# RQ2. Features that have the most explanatory power in distinguishing reusable/non-reusable clones



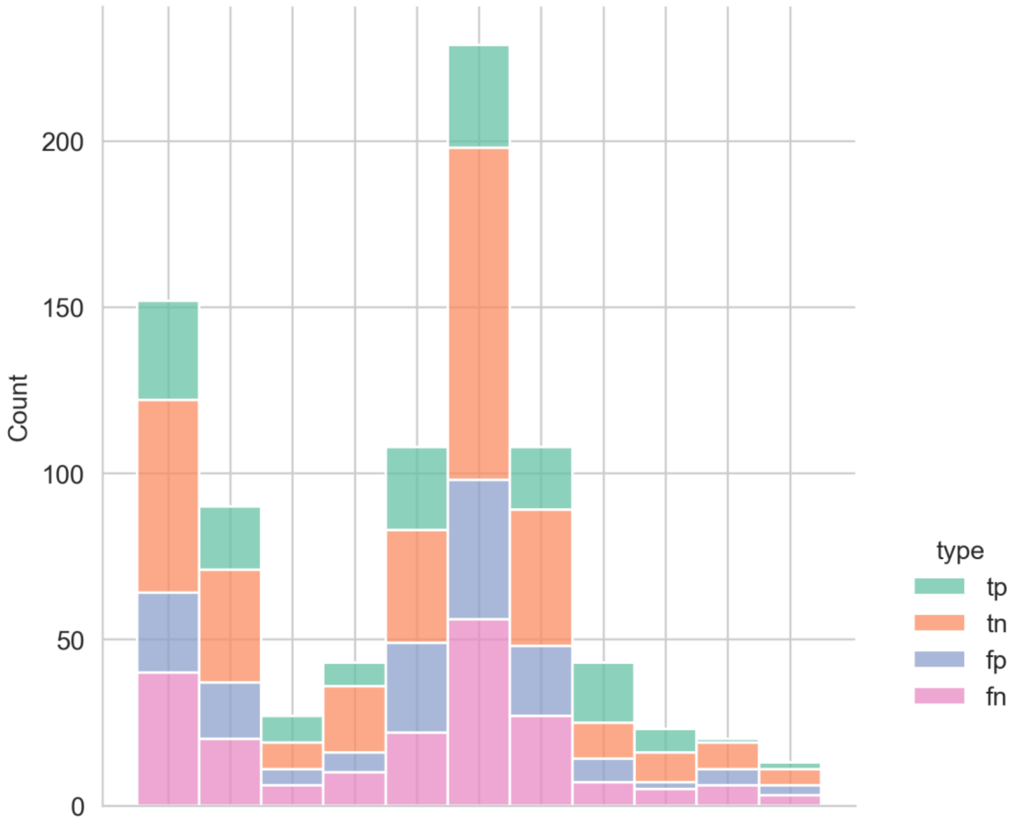

The features number of followers, number of contributors, and length of common paths provide the most explanatory power.

# RQ3. The functionalities of reusable code clones




| Function Category | Category Description |
|---|---|
| **Search** | Retrieve related information to the given object |
| **Config** | Prepare and initialize working environment |
| **Check/Determine** | Check specific status |
| **Add/Remove from collection** | Append/add/insert/remove/delete/exclude |
| **Handle events** | Listen to an event and take handling actions |
| **Convert** | Convert an object from one type to another |
| **Read/Write** | Read/write from/to db, file, stream, buffer |
| **Math** | Process calculations |
| **Encode/Decode** | Encode or decode an object |
| **Visit** | Traverse a collection |
| **Cleanup** | Cleanup working environment |

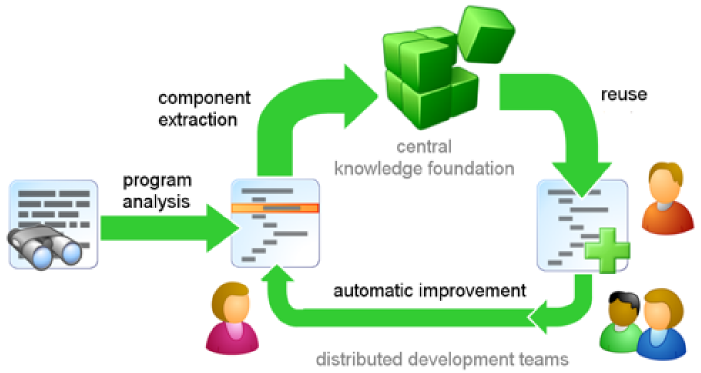
# RQ3. The functionalities of reusable code clones




| Function Category | Category Description |
|---|---|
| **Search** | Retrieve related information to the given object |
| **Config** | Prepare and initialize working environment |
| **Check/Determine** | Check specific status |
| **Add/Remove from collection** | Append/add/insert/remove/delete/exclude |
| **Handle events** | Listen to an event and take handling actions |
| **Convert** | Convert an object from one type to another |
| **Read/Write** | Read/write from/to db, file, stream, buffer |
| **Math** | Process calculations |
| **Encode/Decode** | Encode or decode an object |
| **Visit** | Traverse a collection |
| **Cleanup** | Cleanup working environment |

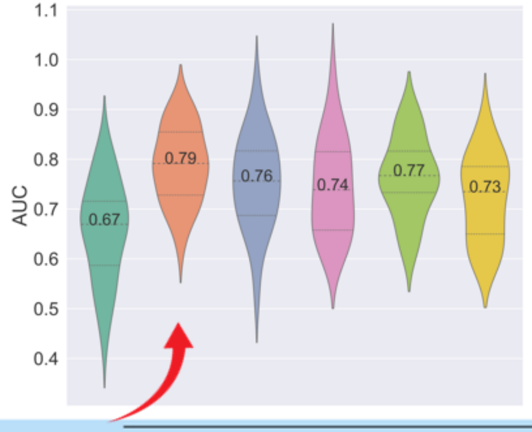
11 functionality categories derived from the clone groups reveal the intention of the code, the top three categories are: search, event-handling, and convert.

# Reusing code clones to improve code quality




To assist development and maintenance team in enhancing code quality, we aim to pinpoint high-quality code clones for reliable reuse.

# RQ1.The AUC achieved by ML classifiers in classifying the reusable code clones

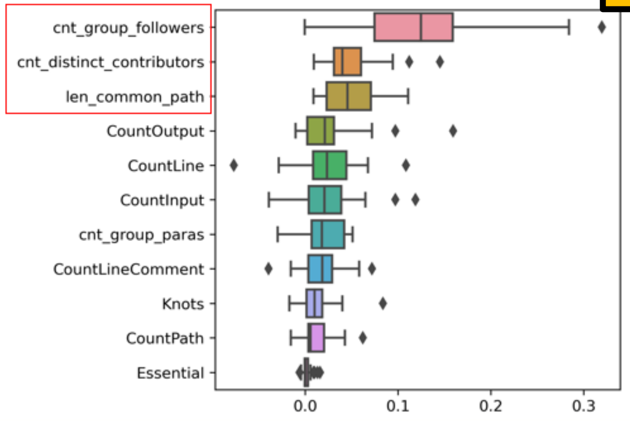



Our Random Forest classifier achieves the best AUC (i.e., 0.79) in determining reusable code clones.

Software Evolution & Analytics Lab

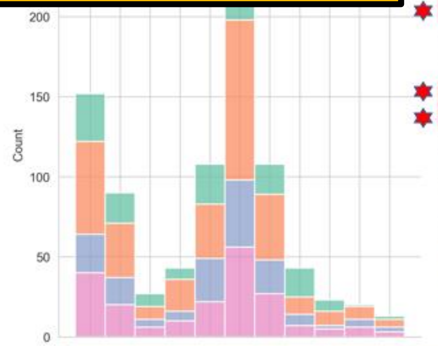
**chunli.yu@queensu.ca**

# RQ2. Features that have the most expla in distinguishing reusable/non-reusa




The features number of followers, number of contributors, and length of common paths provide the most explanatory power.

# alities of reusable code clones




| Function Category | Category Description |
|---|---|
| Search | Retrieve related information to the given object |
| Config | Prepare and initialize working environment |
| Check/Determine | Check specific status |
| Add/Remove from collection | Append/add/insert/remove/delete/exclude |
| Handle events | Listen to an event and take handling actions |
| Convert | Convert an object from one type to another |
| Read/Write | Read/write from/to db, file, stream, buffer |
| Math | Process calculations |
| Encode/Decode | Encode or decode an object |
| Visit | Traverse a collection |
| Cleanup | Cleanup working environment |

11 functionality categories derived from the clone groups reveal the intention of the code, the top three categories are: search, event-handling, and convert.