



Cofinanțat de
Uniunea Europeană

HUB Român de Inteligență Artificială
Cod SMIS: 351416



An empirical analysis about the usage of static source code metrics for Software Defect Prediction

Lect. PhD. Oneț-Marian Zsuzsanna

Babeș-Bolyai University

2026-06-05

Content

Introduction

Background

Duplicates in
CVDP

Follow-up study

Conclusions

- Introduction
- Background
- Previous study about duplicates in CVDP
- The follow-up study
 - Research Question
 - Methodology
 - Results and Analysis
- Conclusions and Further Work

Content

Introduction

Background

Duplicates in
CVDP

Follow-up study

Conclusions

- The first part of this presentation presents part of a study that we have performed last year with Lect. PhD. Hotea Diana.
- The second part presents a follow-up study that we started, which is still work in progress, but we believe that we already have a few interesting results.

Background on Software Defect Prediction

Content

Introduction

Background

Duplicates in
CVDP

Follow-up study

Conclusions

- Software Defect Prediction (SDP) consists in automatically identifying defective components in a software system $\Rightarrow$ supervised classification problem
 - instances are the components of a software system
 - each instance has a binary label
 - the feature vectors are in general the values of different software metrics computed for the component
 - static software metrics
 - process metrics
 - code churn related metrics
 - embeddings
 - etc.

- Considering where the train and test data comes from, SDP has three different scenarios:
 - Inner Version Defect Prediction (IVDP) - we have data from one single version of a software system, which is split into train and test data.
 - Cross Version Defect Prediction (CVDP) - we have data from several version of the software system. For a given version v_i we train the model on instances from previous version(s) and test it on v_i .
 - Cross Project Defect Prediction (CPDP) - we have data from version(s) of two different project. We use data from one project to train a model and test it on the data from the other project.

A study about duplicates in CVDP

Content

Introduction

Background

Duplicates in
CVDP

Follow-up study

Conclusions

- When we consider consecutive versions of a software system, we expect parts of the source code to be the same $\Rightarrow$ their representation might be the same as well.
- This might lead to *data leakage* in CVDP approaches, when the train and the test sets contain identical instances.
- In the 2025 ENASE conference, we have presented a study about this phenomenon [6]:

- Considering two consecutive versions of a software system $\mathcal{S}_{v1}$ and $\mathcal{S}_{v2}$, where for every instance we have the name of instance as well, we defined the following subsets:
 - $\mathcal{S}_{v1 \rightarrow v2}common$ - the set of those instances from $\mathcal{S}_{v2}$ for which exists an instance with the same name in $\mathcal{S}_{v1}$
 - $\mathcal{S}_{v1 \rightarrow v2}new$ - the set of those instances from $\mathcal{S}_{v2}$ for which does not exist an instance with the same name in $\mathcal{S}_{v1}$
- Furthermore, the set $\mathcal{S}_{v1 \rightarrow v2}common$ was split into two subsets:
 - $\mathcal{S}_{v1 \rightarrow v2}sameLabel$ - the set of those instances which have the same label in both versions $\mathcal{S}_{v1}$ and $\mathcal{S}_{v2}$.
 - $\mathcal{S}_{v1 \rightarrow v2}changedLabel$ - the set of those instances which have a different label in $\mathcal{S}_{v1}$ and $\mathcal{S}_{v2}$.

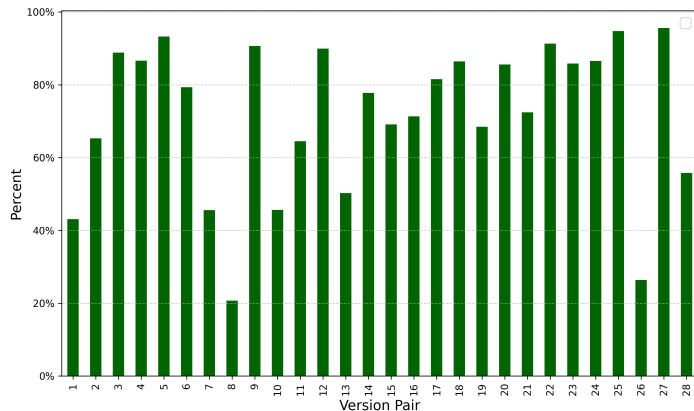
- We had two research questions:
 - RQ1¹: How prevalent is the occurrence of identical instances across two consecutive versions of the software system?
 - We count how many elements of $\mathcal{S}_{v1 \rightarrow v2}^{common}$ have identical feature vectors.
 - RQ2¹: How does the presence of identical instances influence the performance of CVDP models?
 - We trained different ML models on $\mathcal{S}_{v1}$ and tested it on different sets:
 - $\mathcal{S}_{v1}$ - the train data
 - $\mathcal{S}_{v2}$ - the test data
 - $\mathcal{S}_{v1 \rightarrow v2}^{new}$ - the newly introduced instances
 - $\mathcal{S}_{v1 \rightarrow v2}^{sameLabel}$ - instances whose label stayed the same
 - $\mathcal{S}_{v1 \rightarrow v2}^{changedLabel}$ - instances whose label changed

¹Simplified version of the original research question

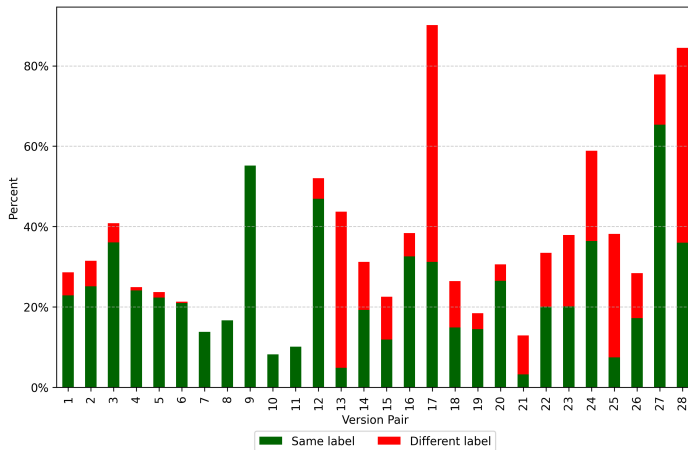
- As case study, we used all data sets from the Seacraft repository [1], where there are at least two versions and where class names are part of the feature list:
 - camel - versions 1.0, 1.2, 1.4, 1.6
 - jedit - versions 3.2, 4.0, 4.1, 4.2, 4.3
 - forest - versions 0.6, 0.7, 0.8
 - ivy - versions 1.1, 1.4, 2.0
 - log4j - versions 1.0, 1.1, 1.2
 - lucene - versions 2.0, 2.2, 2.4
 - poi - versions 1.5, 2.0, 2.5, 3.0
 - synapse - versions 1.0, 1.1, 1.2
 - velocity - versions 1.4, 1.5, 1.6
 - xalan - versions 2.4, 2.5, 2.6, 2.7
 - xerces - versions init, 1.2, 1.3, 1.4
- We had a total of 28 train - test pairs.

Results for RQ1:

- There are only two train-test pairs, where the percentage of common instances is below 40%, the average value is 72% (median is 79%).



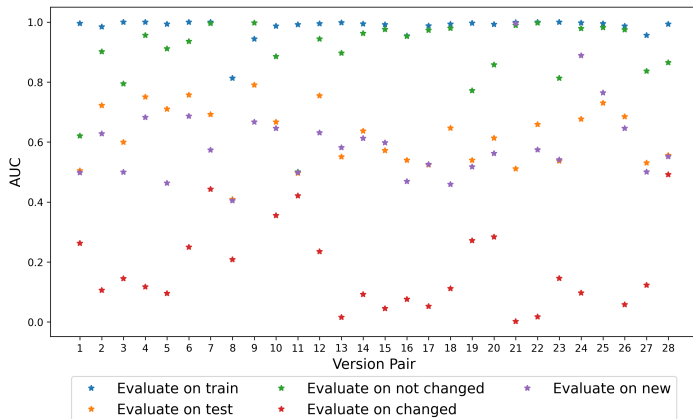
- On average 35% of common instances have identical representation (median is 31%) - but the value is very dataset specific.



● Results for RQ2:

● The model performance is:

- Exceptional for the train data and the test data where the label did not change
- Acceptable for the whole test set and the set of new instances
- Incredibly poor on the set of instances where the label changed



The follow-up study

Content

Introduction

Background

Duplicates in
CVDP

Follow-up study

Repositories

Research questions

Methodology

Results and analysis

Conclusions

- Core idea: ignore the file name attribute and compare directly at the representation
- Extend the previous study in two directions:
 - Consider all SDP scenarios (IVDP, CVDP and CPDP)
 - Consider more repositories and data sets
- We will only consider static source code metrics as features since they are the most frequently used in the literature.

Repositories and datasets

Content

Introduction

Background

Duplicates in
CVDP

Follow-up study

Repositories

Research questions

Methodology

Results and analysis

Conclusions

- To extend the experimental evaluation, we have tried to look up different repositories and datasets used for SDP.
- We have selected the following repositories for our study:

Repository	Year	Datasets	Features / Considered
AEEM [2]	2010	5	21 / 17
Seacraft [1]	2010	45 (11 projects)	20 / 20
ReLink [10]	2011	3	40 / 40
MJ12 [5]	2014	70 (18 projects)	24 / 20
GitHubBugDataset [9]	2016	103 (15 systems)	102 / 52
ELFF [8]	2016	69 (23 projects)	38 / 38
RNalitica [11]	2019	32 (9 projects)	65 / 54
BugHunter [3]	2020	15	95 / 52
Herbold [4]	2022	389 (37 projects)	4189 / 3624
Rebro [7]	2023	275 (39 projects)	12 / 12

- We preprocessed the datasets:
 - only the static source code features were kept (plus label)
 - Class label was transformed in 0 and 1 (if needed)
- Some datasets have 0 defects and some have very few instances (6 datasets have less than 10 instances). We have decided to keep them in the study.

Research questions

Content

Introduction

Background

Duplicates in
CVDP

Follow-up study
Repositories

Research questions

Methodology

Results and analysis

Conclusions

- RQ1: How prevalent is the occurrence of identical instances in datasets used for IVDP, CVDP and CPDP?

Content

Introduction

Background

Duplicates in
CVDP

Follow-up study

Repositories

Research questions

Methodology

Results and analysis

Conclusions

- In order to answer RQ1, we have identified *groups* in our data.
 - For IVDP, a *group* is made of all instances where the features are identical.
 - For CVDP and CPDP, where we have a train and a test set, a *group* is made of one instance of the test set and all instances from the train set which have identical features.
- For all scenarios, groups of size one represent unique data, while groups of size two or more, represent (possible) *data leakage*.

- For the CVDP scenario, the train set was represented by one version of a project and the test set was the next version. Only 7 out of the 10 repositories could have been used for this scenario.
- For the CPDP scenario, the train set was represented by all versions of the train project, while the test set was the current version of the test project.

Results for the IVDP scenario

Content

Introduction

Background

Duplicates in
CVDP

Follow-up study

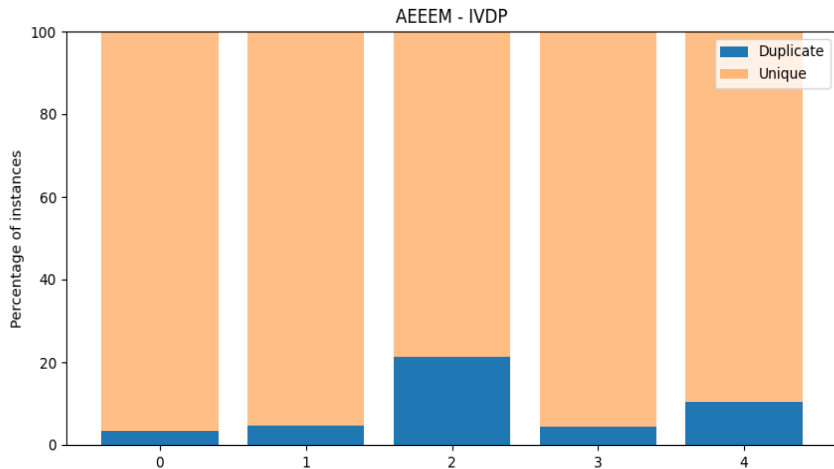
Repositories

Research questions

Methodology

Results and analysis

Conclusions



Content

Introduction

Background

Duplicates in
CVDP

Follow-up study

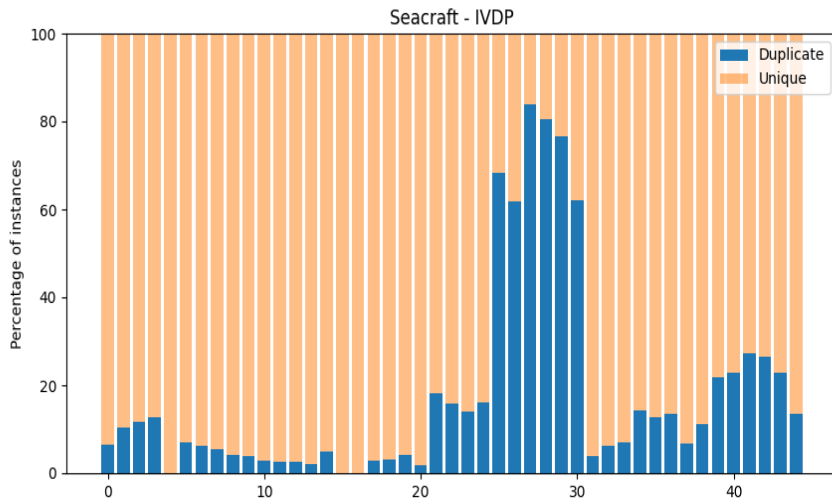
Repositories

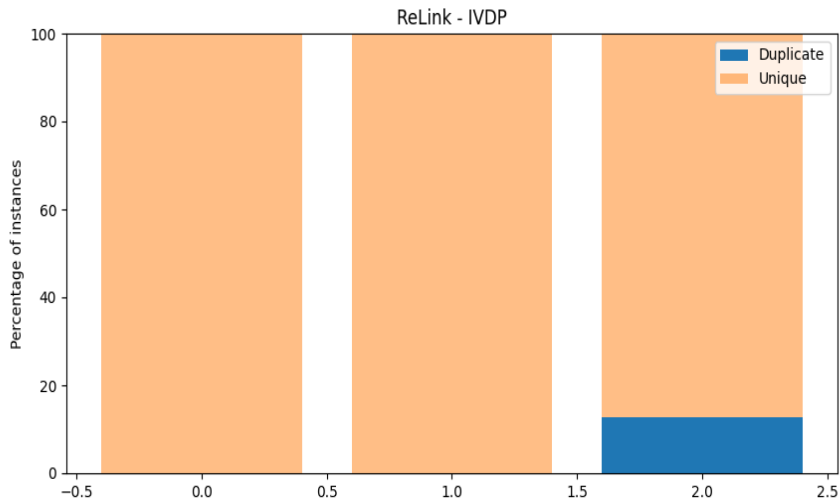
Research questions

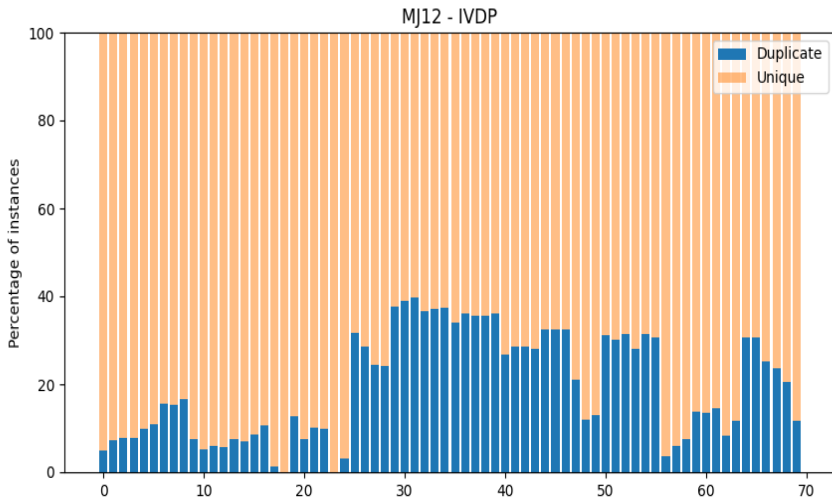
Methodology

Results and analysis

Conclusions







Content

Introduction

Background

Duplicates in
CVDP

Follow-up study

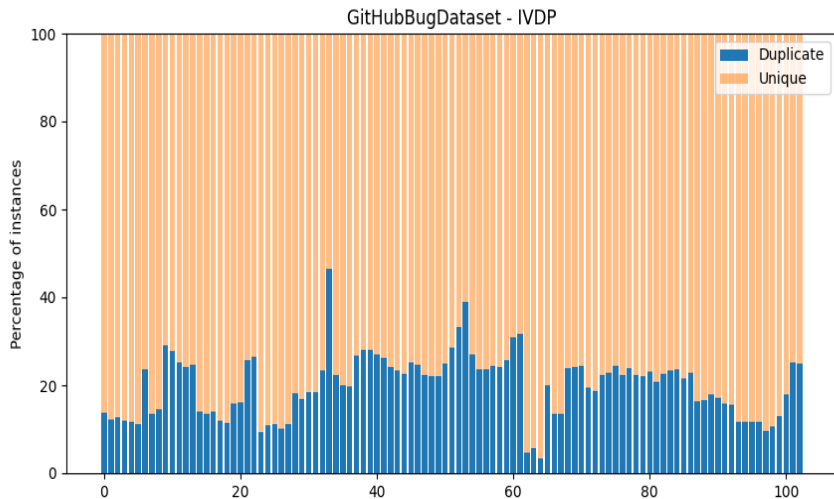
Repositories

Research questions

Methodology

Results and analysis

Conclusions



Content

Introduction

Background

Duplicates in
CVDP

Follow-up study

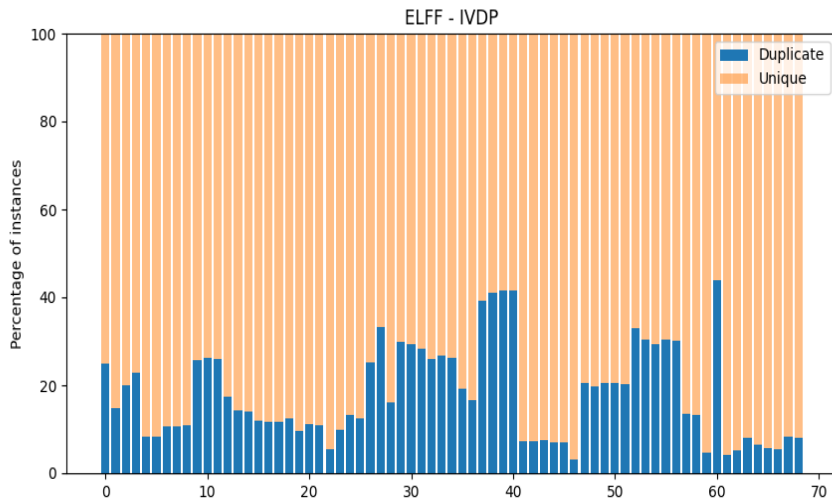
Repositories

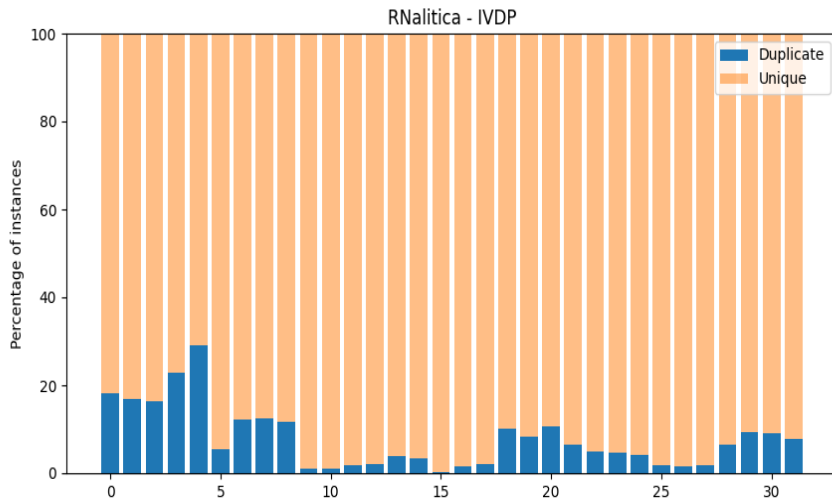
Research questions

Methodology

Results and analysis

Conclusions





Content

Introduction

Background

Duplicates in
CVDP

Follow-up study

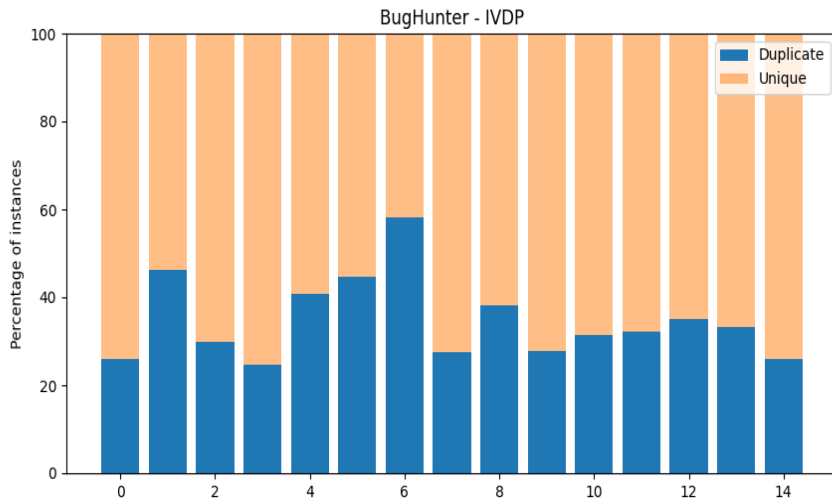
Repositories

Research questions

Methodology

Results and analysis

Conclusions



Content

Introduction

Background

Duplicates in
CVDP

Follow-up study

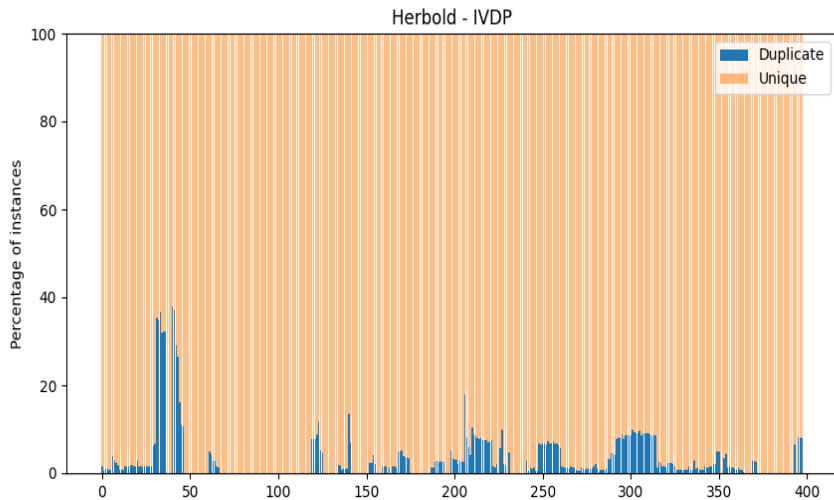
Repositories

Research questions

Methodology

Results and analysis

Conclusions



Content

Introduction

Background

Duplicates in
CVDP

Follow-up study

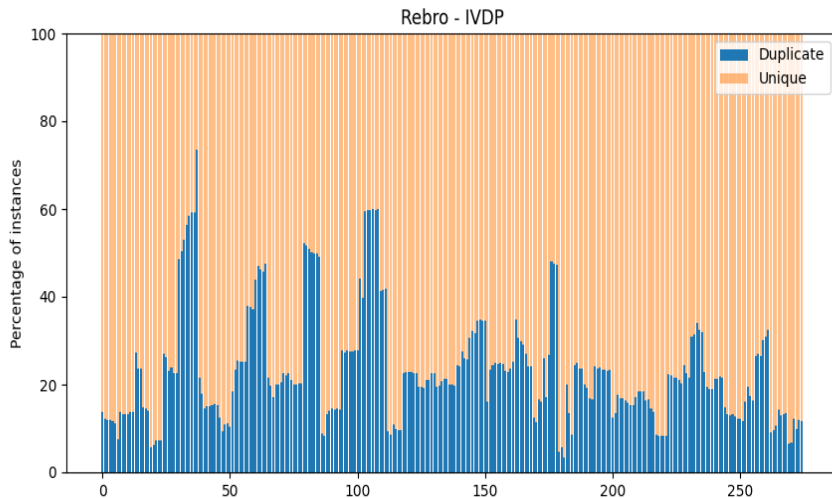
Repositories

Research questions

Methodology

Results and analysis

Conclusions



Repository	Datasets with no duplicates	Max group size	Median percentage duplicate
AEEEM	0	25	4.7
Seacraft	3	122	10.4
ReLink	2	13	0
MJ12	2	104	16.1
GitHubBugDataset	0	210	22
ELFF	0	451	14.2
RNalitica	0	65	5.9
BugHunter	0	230	32.3
Herbold	146	32	1.2
Rebro	2	13	21.2

Results for the CVDP scenario

Content

Introduction

Background

Duplicates in
CVDP

Follow-up study

Repositories

Research questions

Methodology

Results and analysis

Conclusions

Content

Introduction

Background

Duplicates in
CVDP

Follow-up study

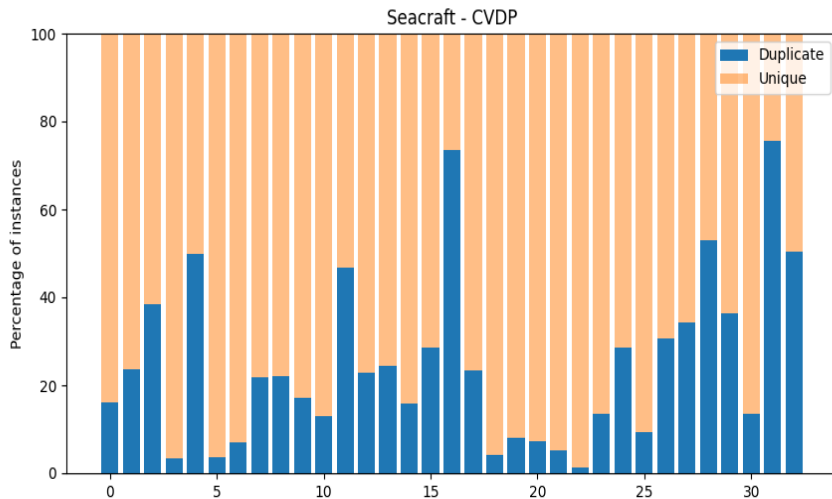
Repositories

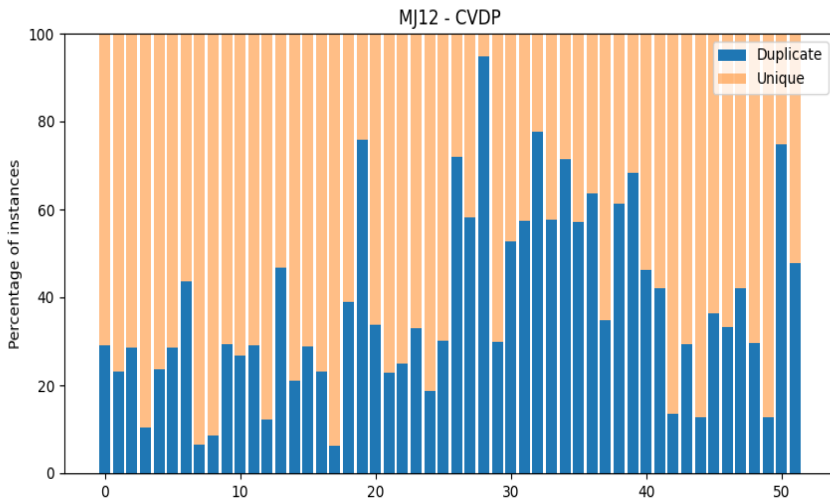
Research questions

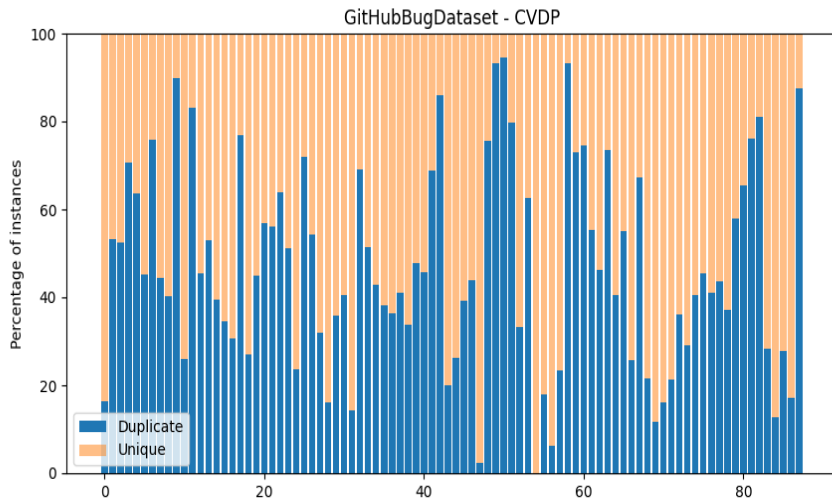
Methodology

Results and analysis

Conclusions







Content

Introduction

Background

Duplicates in
CVDP

Follow-up study

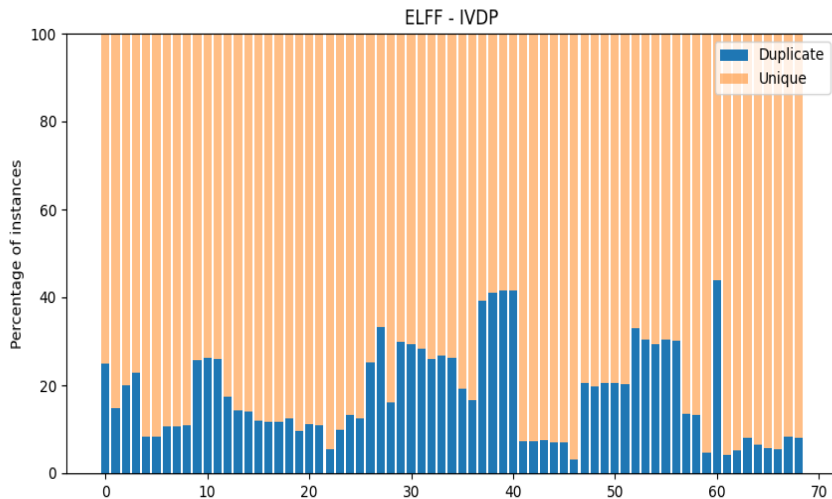
Repositories

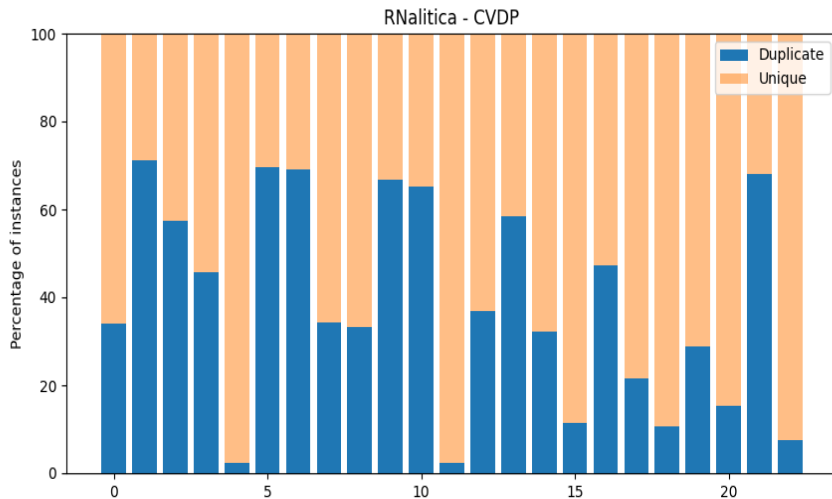
Research questions

Methodology

Results and analysis

Conclusions





Content

Introduction

Background

Duplicates in
CVDP

Follow-up study

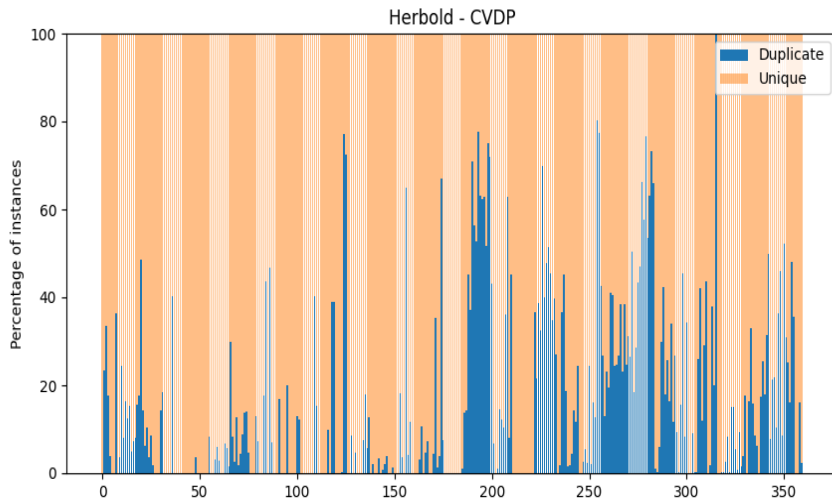
Repositories

Research questions

Methodology

Results and analysis

Conclusions



Content

Introduction

Background

Duplicates in
CVDP

Follow-up study

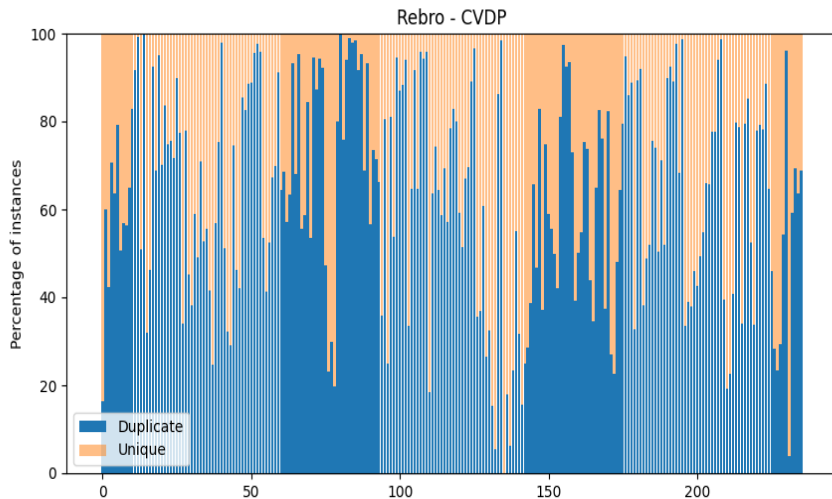
Repositories

Research questions

Methodology

Results and analysis

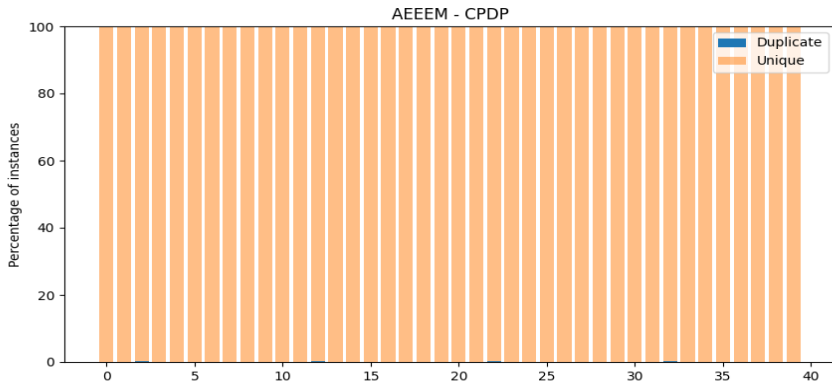
Conclusions



Repository	Datasets with no duplicates	Max group size	Median percentage duplicate
Seacraft	0	70	22.11
MJ12	0	39	34.18
GitHubBugDataset	1	209	44.21
ELFF	3	182	58.95
RNalitica	0	61	34.18
Herbold	120	15	6.82
Rebro	1	636	65.82

Results for the CPDP scenario

- Currently, we do not have results for the Rebro and Herbold repository



Content

Introduction

Background

Duplicates in
CVDP

Follow-up study

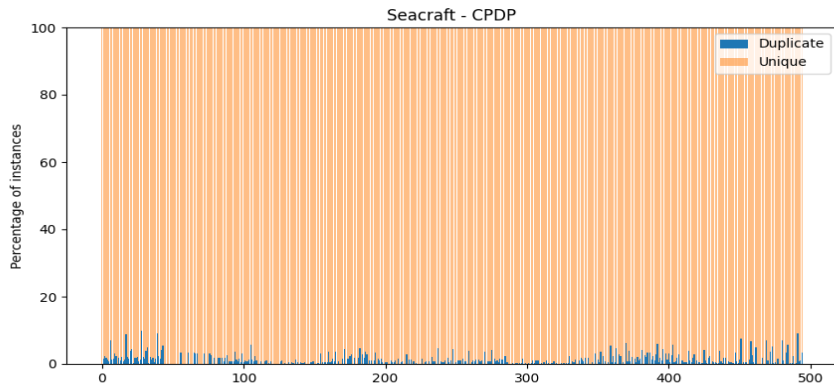
Repositories

Research questions

Methodology

Results and analysis

Conclusions



Content

Introduction

Background

Duplicates in
CVDP

Follow-up study

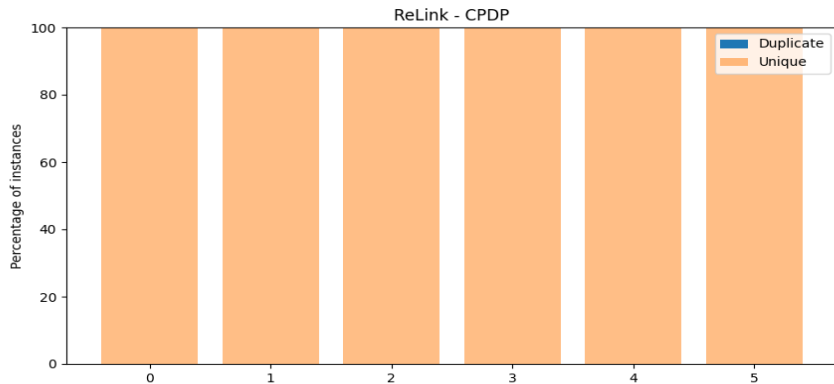
Repositories

Research questions

Methodology

Results and analysis

Conclusions



Content

Introduction

Background

Duplicates in
CVP

Follow-up study

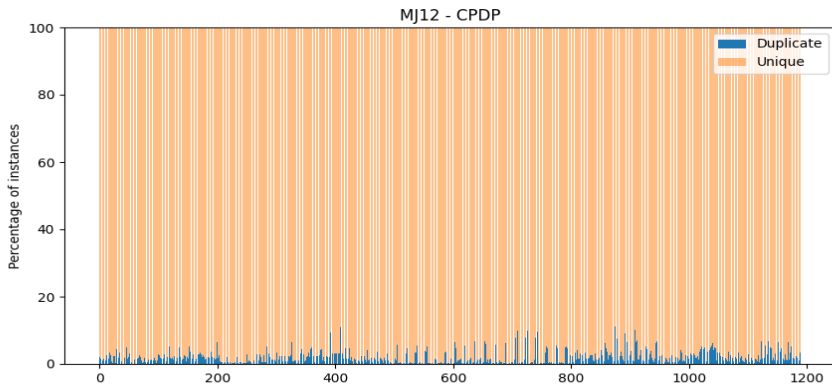
Repositories

Research questions

Methodology

Results and analysis

Conclusions



Content

Introduction

Background

Duplicates in
CVDP

Follow-up study

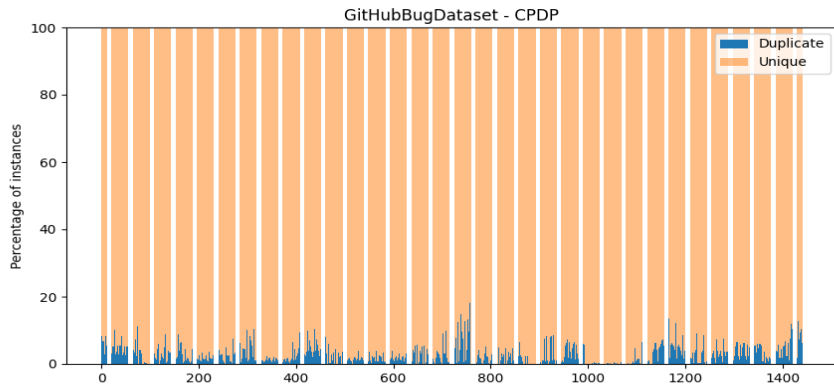
Repositories

Research questions

Methodology

Results and analysis

Conclusions



Content

Introduction

Background

Duplicates in
CVDP

Follow-up study

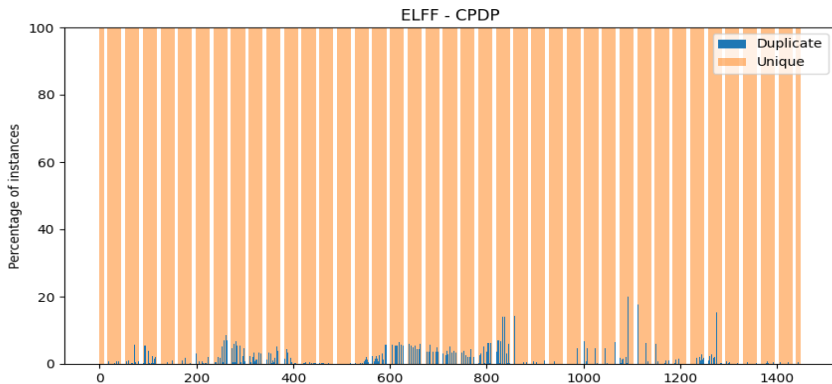
Repositories

Research questions

Methodology

Results and analysis

Conclusions



Content

Introduction

Background

Duplicates in
CVD

Follow-up study

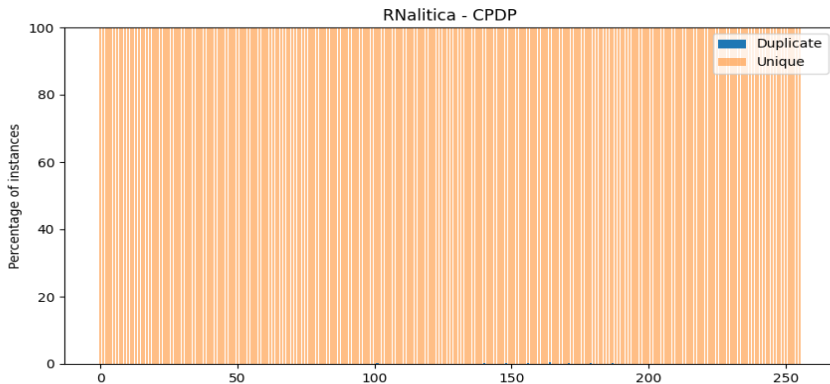
Repositories

Research questions

Methodology

Results and analysis

Conclusions



Content

Introduction

Background

Duplicates in
CVDP

Follow-up study

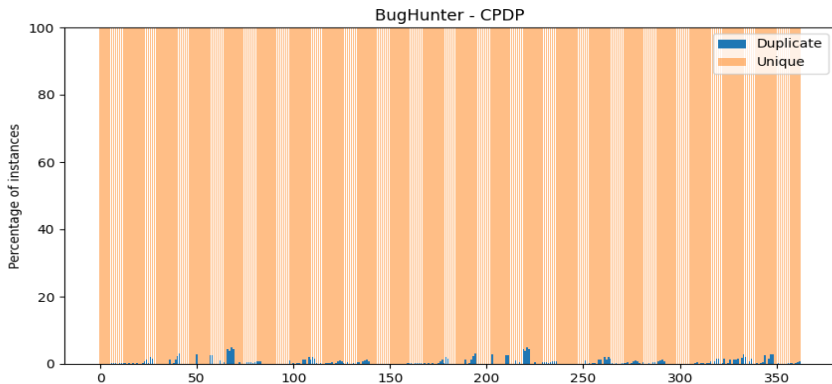
Repositories

Research questions

Methodology

Results and analysis

Conclusions



Content

Introduction

Background

Duplicates in
CVDP

Follow-up study

Repositories

Research questions

Methodology

Results and analysis

Conclusions

Repository	Datasets with no duplicates	Median percentage duplicate
AEEEM	32	0
Seacraft	135	0.69
ReLink	6	0
MJ12	33	1.33
GitHubBugDataset	144	2.07
ELFF	730	0
RNalitica	224	0
BugHunter	142	0.12

Comparison for the 3 scenarios

Content

Introduction

Background

Duplicates in
CVDP

Follow-up study

Repositories

Research questions

Methodology

Results and analysis

Conclusions

- Below we have the median percentage of duplicate instances for the three SDP scenarios

Repository	IVDP	CVDP	CPDP
AEEEM	4.7	NA	0
Seacraft	10.4	22.11	0.69
ReLink	0	NA	0
MJ12	16.1	34.18	1.33
GitHubBugDataset	22	44.21	2.07
ELFF	14.2	58.95	0
RNalitica	5.9	34.18	0
BugHunter	32.3	NA	0.12
Herbold	1.2	6.82	No result yet
Rebro	21.2	65.82	No result yet

Conclusions and future work

Content

Introduction

Background

Duplicates in
CVDP

Follow-up study

Conclusions

- The study about duplicates in CVDP showed us that the default CVDP evaluation scenario suffers from data leakage, which leads to overly optimistic performance indicators, hiding the poor performance on those instances which changed their label from one version to another.
- The follow-up study demonstrated that duplicate instances exist in other SDP scenarios as well, even if CVDP is the most affected.

- In the future we want to continue the follow-up study:
 - Look at the distribution of class labels inside the groups.
 - Look at the source code for the repositories where we have access to it, to see if identical representation means identical source code.
- Propose new evaluation methodologies, which avoid the *data leakage* caused by the duplicates.

Thank you for your attention!

Financial support from:

“Romanian Hub for Artificial Intelligence - HRIA”, Smart Growth, Digitization, and Financial Instruments Program, 2021–2027, MySMIS no. 351416.

Content

Introduction

Background

Duplicates in
CVDP

Follow-up study

Conclusions



The SEACRAFT repository of empirical software engineering data, 2017.



Marco D'Ambros, Michele Lanza, and Romain Robbes.

An extensive comparison of bug prediction approaches.

In 2010 7th IEEE working conference on mining software repositories (MSR 2010), pages 31–41. IEEE, 2010.



Rudolf Ferenc, Péter Gyimesi, Gábor Gyimesi, Zoltán Tóth, and Tibor Gyimóthy.

An automatically created novel bug dataset and its validation in bug prediction.

Journal of Systems and Software, 169:110691, 2020.



Steffen Herbold, Alexander Trautsch, Fabian Trautsch, and Benjamin Ledel.

Problems with szz and features: An empirical study of the state of practice of defect prediction data collection.

Empirical Software Engineering, 27(2):42, 2022.



Lech Madeyski and Marian Jureczko.

Which process metrics can significantly improve defect prediction models? an empirical study.

Software Quality Journal, 23(3):393–422, 2015.



Zsuzsanna Onet-Marian and Diana-Lucia Hotea.

Cross-version defect prediction: Does excessive train-test similarity affect the reliability of evaluation?

In *ENASE*, pages 304–315, 2025.



Dominik Arne Rebro, Stanislav Chren, and Bruno Rossi.

Source code metrics for software defects prediction.

In Proceedings of the 38th ACM/SIGAPP symposium on applied computing, pages 1469–1472, 2023.



Thomas Shippey, Tracy Hall, Steve Counsell, and David Bowes.

So you need more method level datasets for your software defect prediction? voilà!

In Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, pages 1–6, 2016.



Zoltán Tóth, Péter Gyimesi, and Rudolf Ferenc.

A public bug database of github projects and its application in bug prediction.

In International conference on computational science and its applications, pages 625–638. Springer, 2016.



Rongxin Wu, Hongyu Zhang, Sunghun Kim, and Shing-Chi Cheung.
Relink: recovering links between bugs and changes.

In Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering, pages 15–25, 2011.



Suraj Yatish, Jirayus Jiarpakdee, Patanamon Thongtanunam, and Chakkrit Tantithamthavorn.

Mining software defects: Should we consider affected releases?

In 2019 IEEE/ACM 41st international conference on software engineering (ICSE), pages 654–665. IEEE, 2019.