The Art and Science of Analyzing Software Data

FSE 2015

Leandro Minku: University of Birmingham, UK

Fayola Peters: Lero, University of Limerick, Ireland

http://www.cs.bham.ac.uk/~minkull/publications/fse15-tutorial.pdf

Who we are today…

Leandro L. Minku
University of Birmingham, UK
L.L.Minku@cs.bham.ac.uk

UNIVERSITY OF BIRMINGHAM

Fayola Peters
Lero, University of Limerick, Ireland
fayola.peters@lero.ie

Lero THE IRISH SOFTWARE RESEARCH CENTRE

Who we are tomorrow…

Leandro L. Minku
University of Leicester, UK
L.L.Minku@cs.bham.ac.uk

University of Leicester

Fayola Peters
Lero, University of Limerick, Ireland
fayola.peters@lero.ie

Lero THE IRISH SOFTWARE RESEARCH CENTRE

1. Introduction
2. Sharing data
3. Privacy and sharing
4. Sharing models
5. Summary

1. Introduction
2. Sharing data
3. Privacy and sharing
4. Sharing models
5. Summary

1a. Analyzing software data: why?
1b. The PROMISE project
1c. Analyzing software data: how?

1a. Analyzing software data: why?

In the 21^{st} century, too much data

E.g. PROMISE repository of SE data

- grown to 200+ standard projects
- 250,000+ spreadsheets

And a dozen other open-source repositories:

- E.g. see next page
- E.g Feb 2015
  - Mozilla Firefox : 1.1 million bug reports,
  - GitHub host 14+ million projects.

1a. Analyzing software data: why?

Repository	URL
Bug Prediction Dataset	http://bug.inf.usi.ch
Eclipse Bug Data	www.st.cs.uni-saarland.de/softevo/bug-data/eclipse
FLOSSMetrics	http://flossmetrics.org
FLOSSMole	http://flossmole.org
International Software Benchmarking Standards Group (IBSBSG)	www.isbsg.org
ohloh	www.ohloh.net
PROMISE	http://promisedata.googlecode.com
Qualitas Corpus	http://qualitascorpus.com
Software Artifact Repository	http://sir.unl.edu
SourceForge Research Data	http://zerlot.cse.nd.edu
Sourcerer Project	http://sourcerer.ics.uci.edu
Tukutuku	www.metriq.biz/tukutuku
Ultimate Debian Database	http://udd.debian.org

Impossible to browse all software project data!

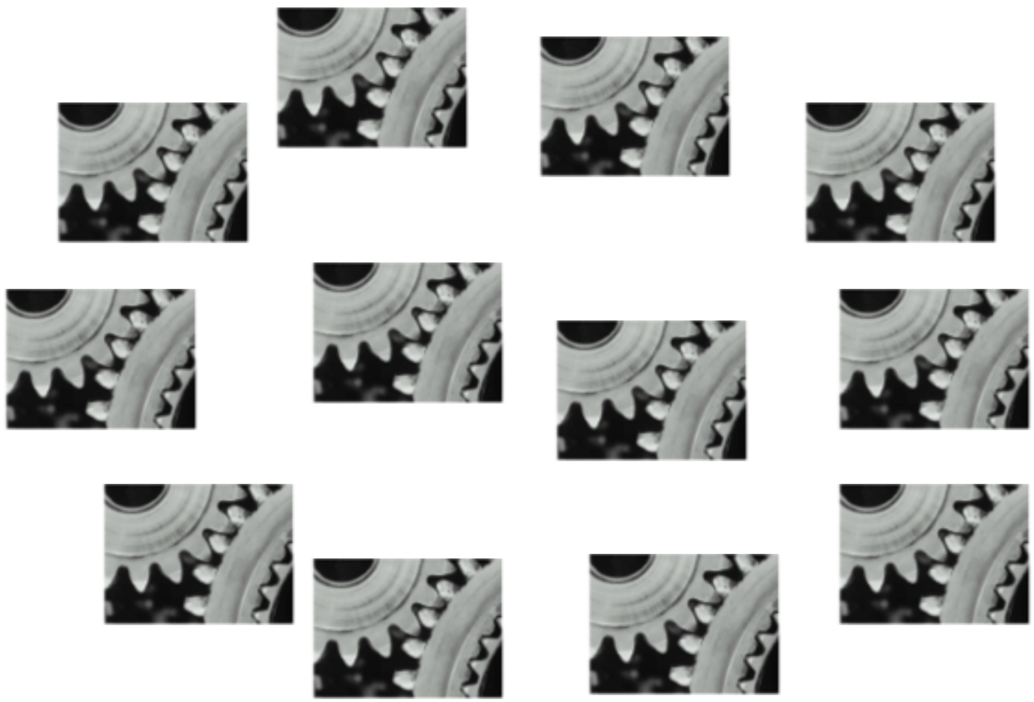
1a. Analyzing software data: why?

Repository	URL
Bug Prediction Dataset	http://bug.inf.usi.ch
Eclipse Bug Data	www.st.cs.uni-saarland.de/softevo/bug-data/eclipse
FLOSSMetrics	http://flossmetrics.org
FLOSSMole	http://flossmole.org
International Software Benchmarking Standards Group (IBSBSG)	www.isbsg.org
ohloh	www.ohloh.net
PROMISE	http://promisedata.googlecode.com
Qualitas Corpus	http://qualitascorpus.com
Software Artifact Repository	http://sir.unl.edu
SourceForge Research Data	http://zerlot.cse.nd.edu
Sourcerer Project	http://sourcerer.ics.uci.edu
Tukutuku	www.metriq.biz/tukutuku
Ultimate Debian Database	http://udd.debian.org

With the right tools, we can gain useful insights from software data!

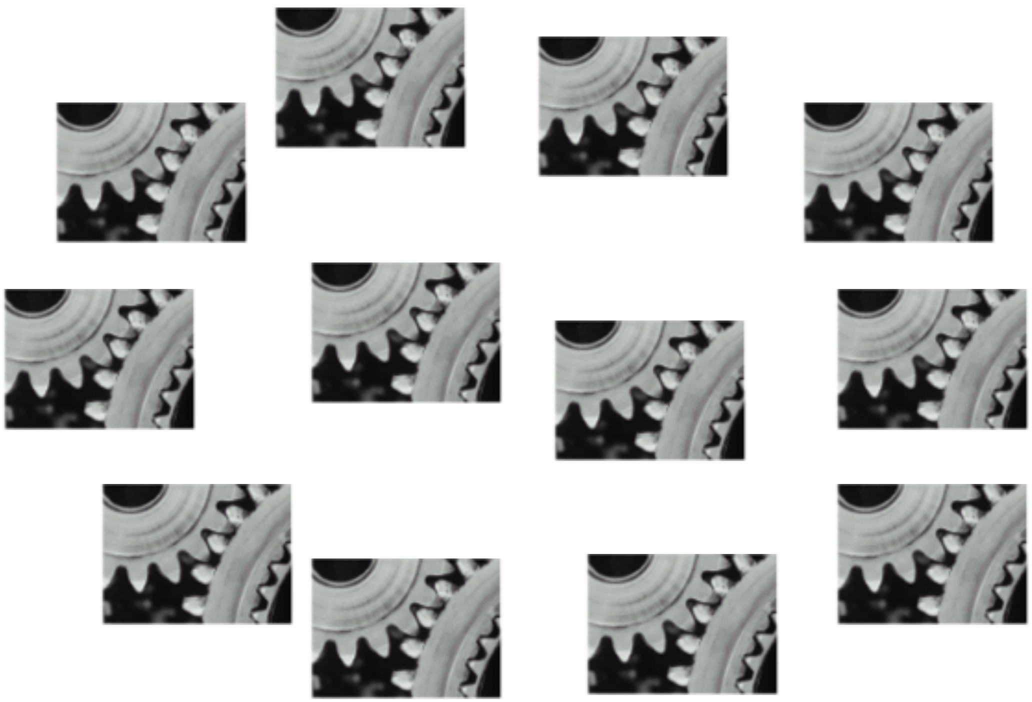
Example: Software Defect Prediction

Software code is composed of several components.



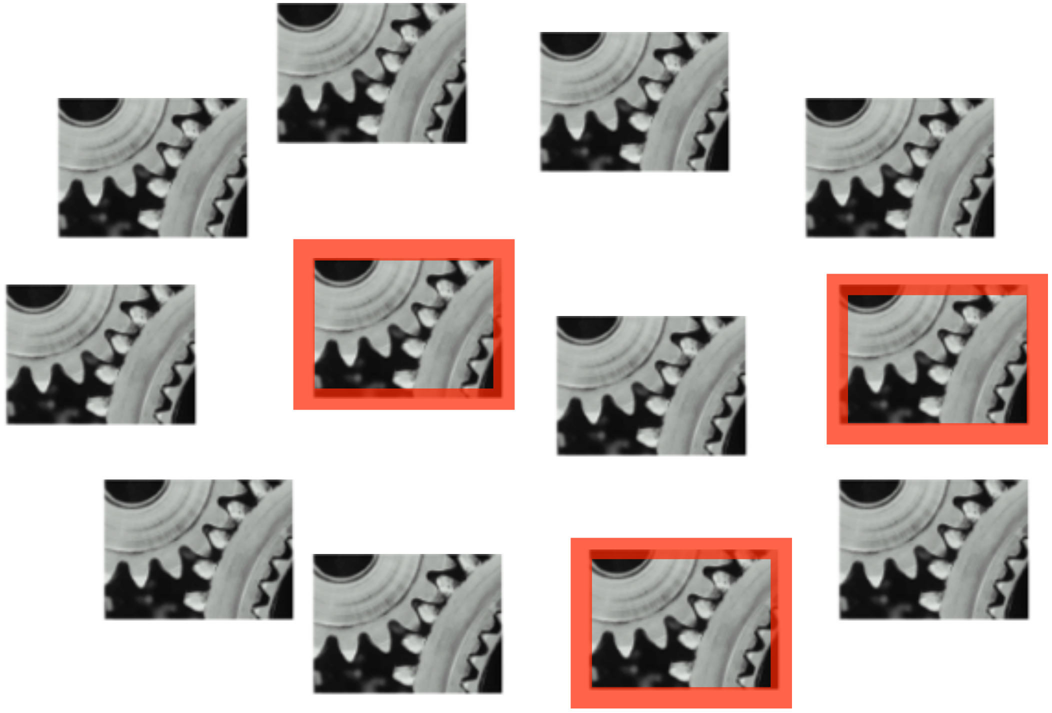
Example: Software Defect Prediction

Testing all these components can be very expensive.

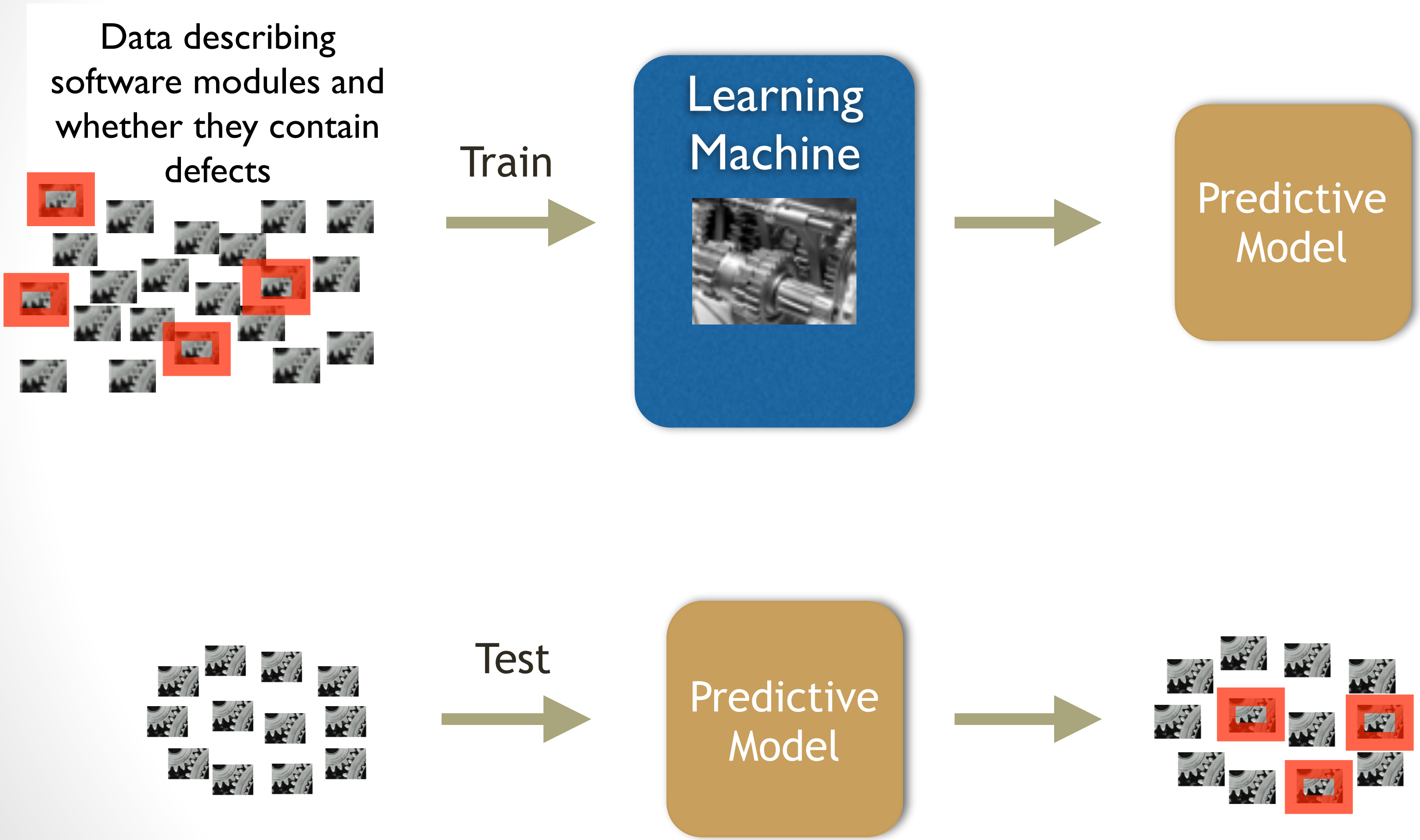


Example: Software Defect Prediction

If we know which components are likely to be defective, we can increase testing cost-effectiveness.



Example: Software Defect Prediction



Example: Software Effort Estimation

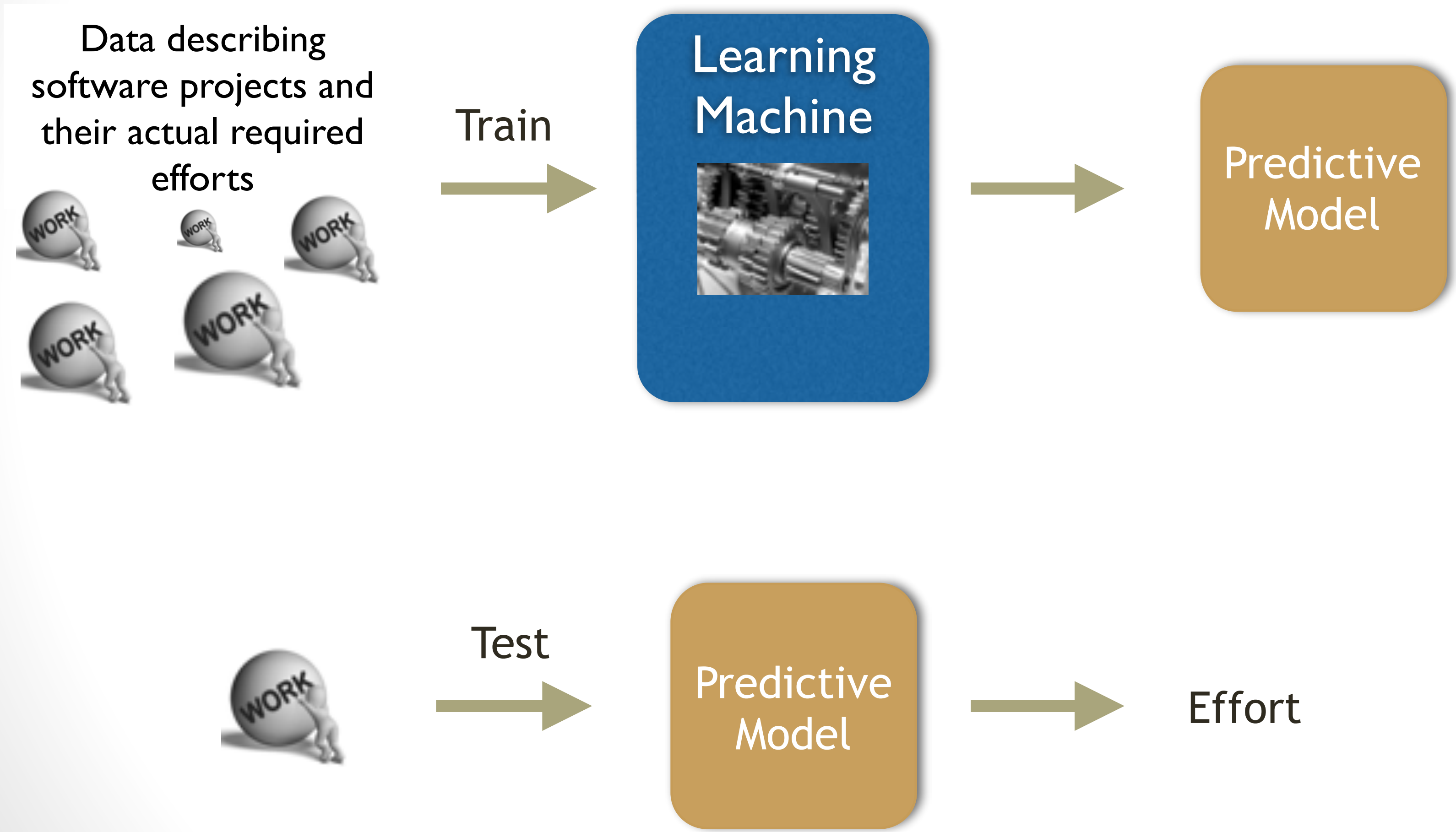
Estimation of the effort required to develop a software project.

- Effort is measured in person-hours, person-months, etc.
- Influenced by attributes such as required reliability, programming language, development type, team expertise, etc.
- Main factor influencing project cost.
- Overestimation vs underestimation.

Example: Software Effort Estimation

Nasa cancelled its incomplete Check-out Launch Control Software project after the initial $200M estimate was exceeded by another $200M.

Example: Software Effort Estimation



1a. Analyzing software data: why?

- Other examples of insights:
  - What team expertise to assign to a project so that it is more cost-efficient
  - How the productivity of a company changes over time
  - How to improve productivity
  - What commits are most likely to induce crashes
  - What developer to assign to what bug
  - What method has a bad smell
  - …

1b. The PROMISE Project

Serve all our data, on-line

The PROMISE repo
openscience.us/repo

#storingYourResearchData

- URL
  - openscience.us/repo
  - Data from 100s of projects
  - E.g. EUSE:
    - 250,000+ spreadsheets
- Oldest continuous repository of SE data
  - Version 0: 2002
  - For other repos, see Table 1 of goo.gl/UFZgnd

1b. The PROMISE Project

- "*Research has deserted the individual and entered the group. The individual worker find the problem too large, not too difficult. (They) must learn to work with others.*"
  - Theobald Smith
    American pathologist and microbiologist
    1859 -- 1934

1b. The PROMISE Project

PROMISE 2015

The 11th International Conference on Predictive Models and Data Analytics in Software Engineering

October 21, 2015, Beijing, China
Co-located with ESEM 2015 - 9th International Symposium on Empirical Software Engineering and Measurement
http://promisedata.org/2015/

Sponsored by:
Microsoft Research

CALL FOR PAPERS

PROMISE is an annual forum for researchers and practitioners to present, discuss and exchange ideas, results, expertise and experiences in construction and/or application of predictive models and data analytics in software engineering. Such models and analyses could be targeted at: planning, design, implementation, testing, maintenance, quality assurance, evaluation, process improvement, management, decision making, and risk assessment in software and systems development. PROMISE is distinguished from similar forums with its public data repository and focus on methodological details, providing a unique interdisciplinary venue for software engineering and data mining communities, and seeking for verifiable and repeatable experiments that are useful in practice.

Topics of Interest

Topics of interest include, but are not limited to:

Application oriented:

- using predictive models and software data analytics in policy and decision-making;
- predicting for cost, effort, quality, defects, business value;
- quantification and prediction of other intermediate or final properties of interest in software

1b. The PROMISE Project

If it works, try to make it better

- "The following is my valiant attempt to capture the difference (between PROMISE and MSR)"
- "To misquote George Box, I hope my model is more useful than it is wrong:
  - For the most part, the MSR community was mostly concerned with the *initial collection* of data sets from software projects.
  - Meanwhile, the PROMISE community emphasized the analysis of the data *after it was collected*."
- "The PROMISE people routinely posted all their data on a public repository
  - their new papers would re-analyze old data, in an attempt to improve that analysis.
  - In fact, I used to joke "PROMISE. Australian for repeatability" (apologies to the Fosters Brewing company). "

Dr. Prem Devanbu
UC Davis
General chair, MSR'14

1b. The PROMISE Project

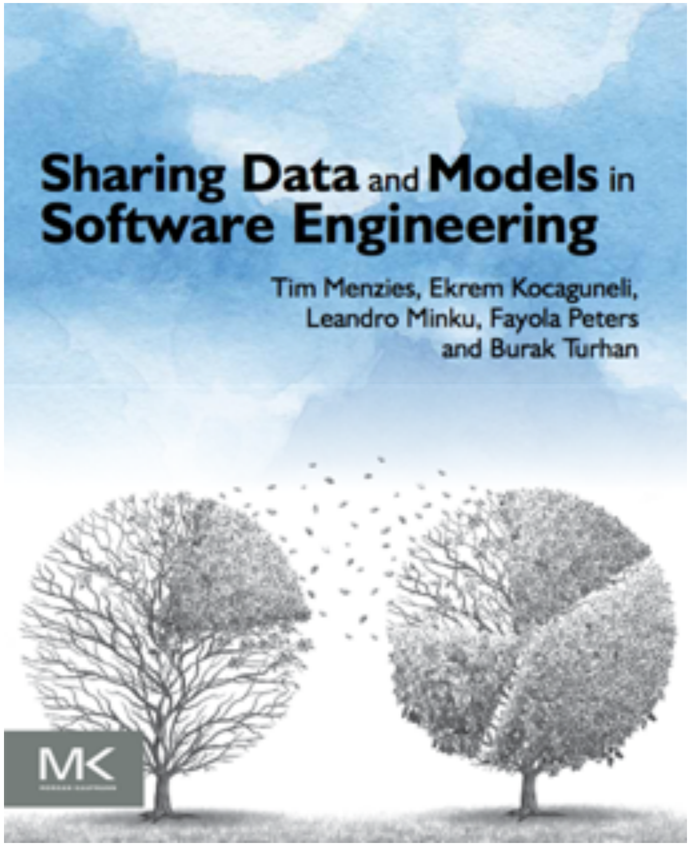
Challenges

- Initial, naïve, view:
  - Collect enough data …
  - … and the truth will emerge
- Reality:
  - The more data we collected …
  - … the more variance we observed
  - It's like the microscope zoomed in
    - to smash the slide
  - Conclusion instability
- So now we routinely slice the data
  - Find local lessons in local regions.

1c. Analyzing software data: how?

- Software engineering is so diverse
- What works <u>there</u> may not work <u>here</u>
- Need cost effective methods for finding best local lessons
- Every development team needs a data scientist

1c. Analyzing software data: how?



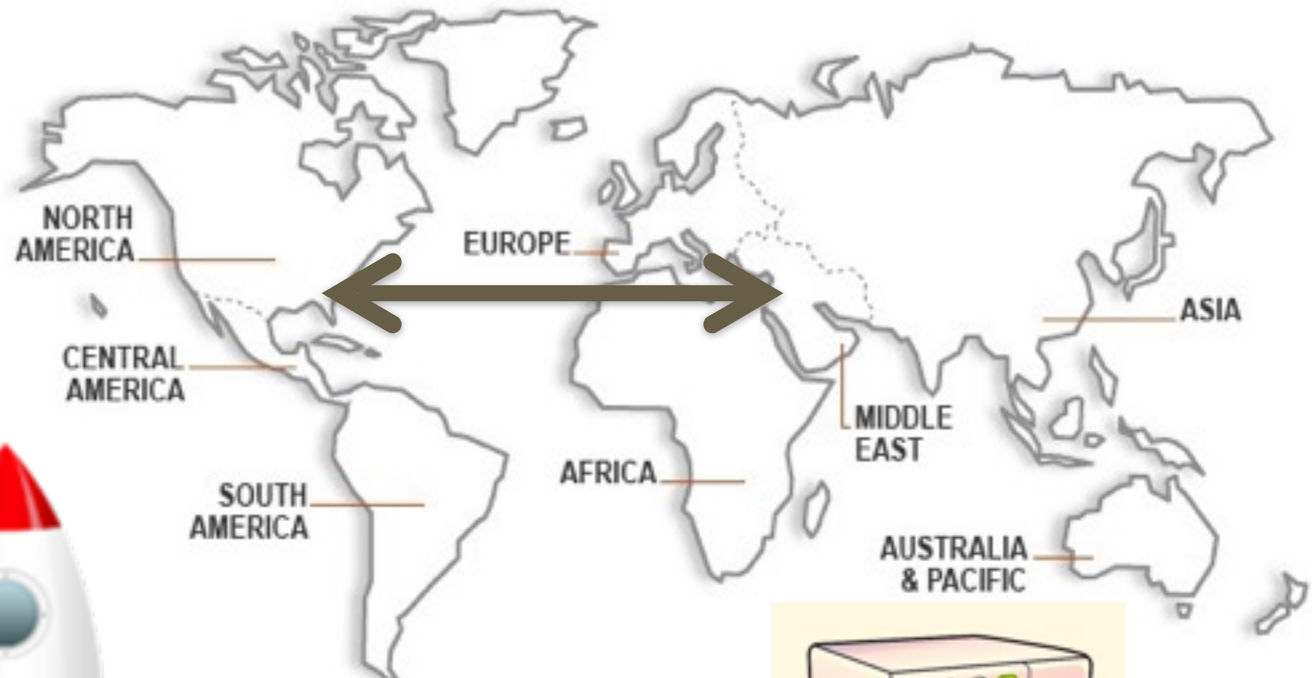
http://www.amazon.co.uk/Sharing-Data-Models-Software-Engineering/dp/0124172954

1. Introduction
2. Sharing data
3. Privacy and Sharing
4. Sharing models
5. Summary

Step 1: Throw most of it away

Step 2: Learn from the rest

From Turkish Washing Machines to NASA Space Ships

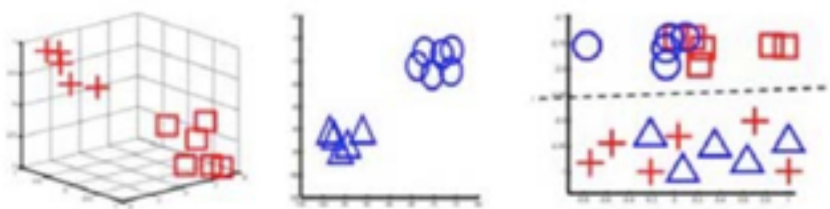


Burak Turhan, Tim Menzies, Ayşe B. Bener, and Justin Di Stefano. 2009. On the relative value of cross-company and within-company data for defect prediction. Empirical Softw. Eng. 14, 5 (October 2009),

Q: How to transfer data between projects? A: Be very cruel to the data

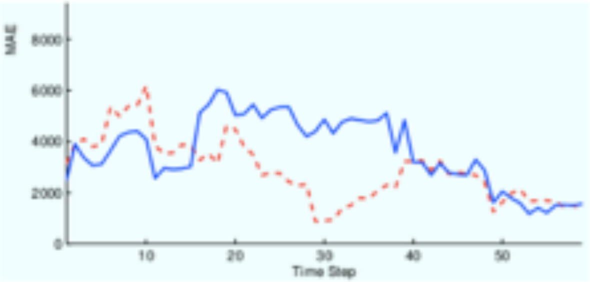
- **Ignore** most of the data
  - relevancy filtering: *Turhan ESEj'09; Peters TSE'13, ICSE'15*
  - variance filtering: *Kocaguneli TSE'12,TSE'13*
  - popularity filtering: *Kocaguneli PROMISE'12*

- **Contort** the data
  - spectral learning (working in PCA space or some other rotation): *Menzies TSE'13; Nam ICSE'13*



(a) 3-D data (b) 2-D data (c) Projected space

- Build a **bickering committee** of models of the data
  - Ensembles *Minku ICSE'14, PROMISE'12*



Ignoring Data -- Data Format

	Att1	*Att2*	…	*AttN*
Ex1	*1*	*high*	…	*Yes*
Ex2	*2*	*medium*	…	*No*
Ex3	*1.5*	*low*	…	*Yes*
…	…	…	…	…

Data Format

Observations / examples

	Att1	*Att2*	…	*AttN*
Ex1	*1*	*high*	…	*Yes*
Ex2	*2*	*medium*	…	*No*
Ex3	*1.5*	*low*	…	*Yes*
…	…	…	…	…

Data Format

Input / Independent Attributes

Observations / examples

	Att1	*Att2*	…	*AttN*
Ex1	*1*	*high*	…	*Yes*
Ex2	*2*	*medium*	…	*No*
Ex3	*1.5*	*low*	…	*Yes*
…	…	…	…	…

Data Format

Input / Independent Attributes

Output / Dependent Attribute

Observations / examples

	Att1	*Att2*	…	*AttN*
Ex1	*1*	*high*	…	*Yes*
Ex2	*2*	*medium*	…	*No*
Ex3	*1.5*	*low*	…	*Yes*
…	…	…	…	…

Example for Software Defect Prediction

Output / Dependent Attribute

Input / Independent Attributes

Observations / examples

	Size	*Number of operators*	*…*	*Bug/ No bug*
Component 1	*10*	*3*	…	*No*
Component 2	*20*	*6*	…	*No*
Component 3	*100*	*10*	…	*Yes*
…	…	…	…	…

Example for Software Effort Estimation

Input / Independent Attributes

Output / Dependent Attribute

Observations / examples

	Software Size	*Team Expertise*	…	*Effort*
Project 1	*1000*	*high*	…	*60*
Project 2	*200*	*medium*	…	*50*
Project 3	*150*	*low*	…	*50*
…				

Different Ways to Ignore Data

	Att1	*Att2*	…	*AttN*
Ex1	*1*	*high*	…	*Yes*
Ex2	*2*	*medium*	…	*No*
Ex3	*1.5*	*low*	…	*Yes*
…	…	…	…	…

How to ignore data?

Different Ways to Ignore Data

	Att1	At	…	AttN
Ex1	1	hig	…	Yes
Ex2	2	medi	…	No
Ex3	1.5	low	…	Yes
…	…	…	…	…

Prune columns

Different Ways to Ignore Data

	Att1	*Att2*	…	*AttN*
Ex1	***1***	***high***	…	***Yes***
Ex2	***2***	***medium***	…	***No***
Ex3				
…	…	…	…	…

Prune rows

Different Ways to Ignore Data

	Att1	*Att2*	…	*AttN*
Ex1	*1*	*high*	…	*Yes*
Ex2	*2*	*medium*	…	*No*
Ex3	*1.5*	*low*	…	*Yes*
…	…	…	…	…

Prune ranges

But Why Prune at All?
Why not use all the data?

- Outliers may confuse data analysis.
- Irrelevant features may make data analysis more difficult.

But Why Prune When Sharing Data? Why not use all the data?

The original vision of PROMISE	Software projects are different

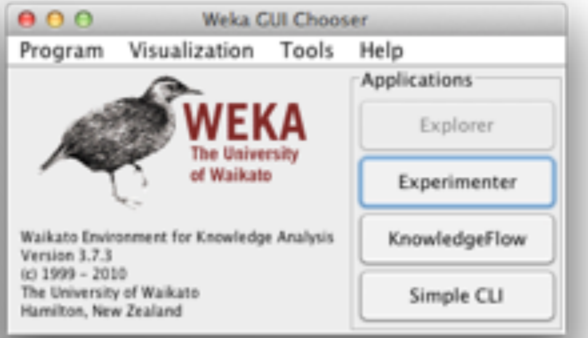
- With enough data, our knowledge will stabilize
- But the more data we collected …
  - … the more variance we observed
- Its like the microscope zoomed in
  - to smash the slide

- They change from place to place
- They change from time to time
- My lessons may not apply to you
- Your lessons may not even apply to you (tomorrow)
- Locality, locality, locality

Ignoring Data

Column pruning	Row pruning	Range pruning
• irrelevancy removal e.g. correlation-based feature selection • better predictions	• outliers • cross-company learning • handling missing values • privacy • anomaly detection • incremental learning	• contrast • goals

[Demo]



Ignoring Data

Column pruning	Row pruning	Range pruning
• irrelevancy removal • better predictions remove columns if that would lead to better predictions	• outliers • cross-company learning • handling missing values • privacy • anomaly detection • incremental learning	• contrast • goals

Ignoring Data

Column pruning	Row pruning	Range pruning
• irrelevancy removal • better predictions	• outliers • cross-company learning • handling missing values NN-filtering, TEAK, popularity-based filtering • privacy • anomaly detection • incremental learning	• contrast • goals

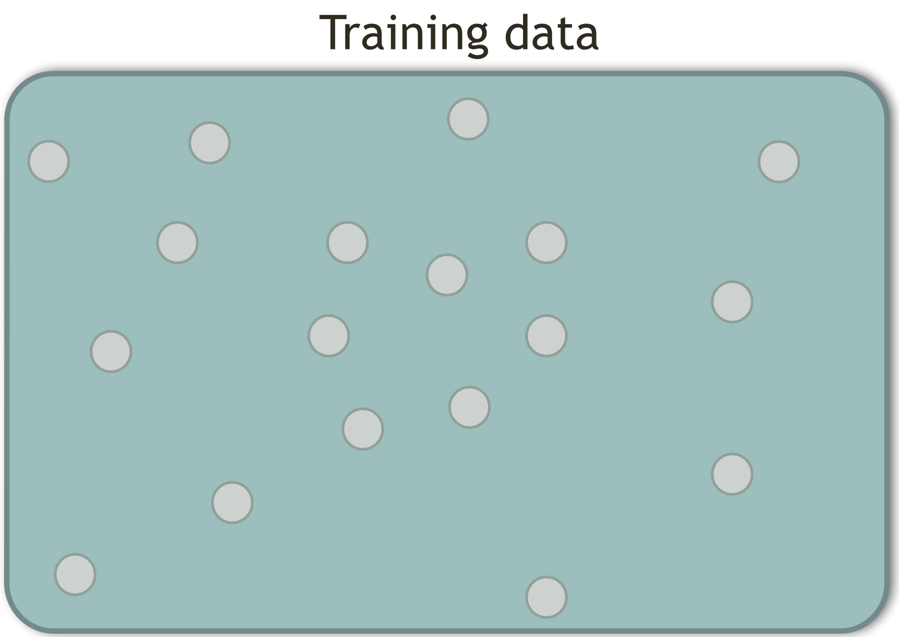
Nearest Neighbor (NN) Filtering

- Idea:
  - Step 1: Find the relevant data
  - Step 2: Build a predictor based on the relevant data

B. Turhan, T. Menzies, A. Bener, J. Distefano "On the Relative Value of Cross-Company and Within-Company Data for Defect Prediction", Empirical Software Engineering, 2009.

NN Filtering - Step 1

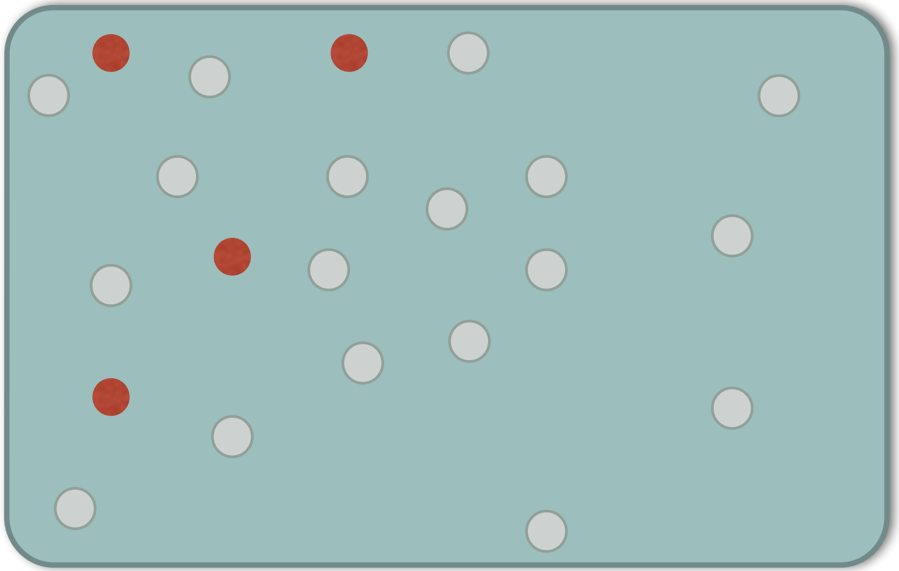
- Step 1: Find the relevant data



NN Filtering - Step 1

- Step 1: Find the relevant data

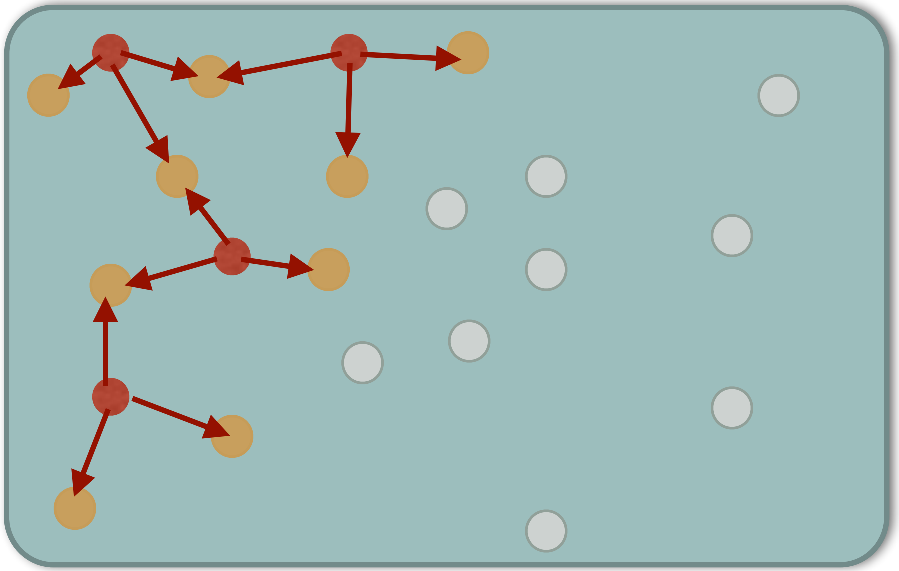
Training and test data



NN Filtering - Step 1

- Step 1: Find the relevant data

Find training data closest to test data



k-nearest neighbors

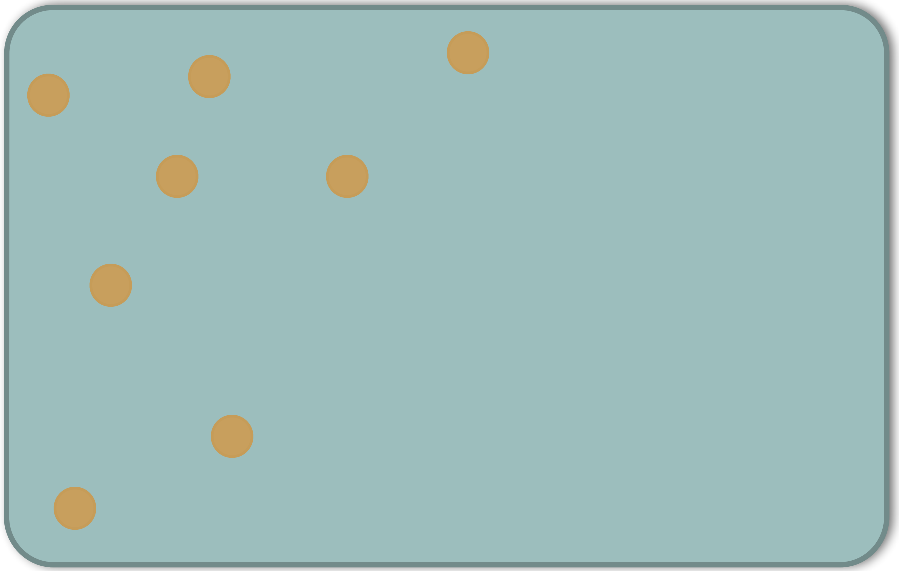
Euclidean distance based on input features

If you are dealing with prediction tasks, do not use the output attribute for this step!

NN Filtering - Step 1

- Step 1: Find the relevant data

Relevant training data



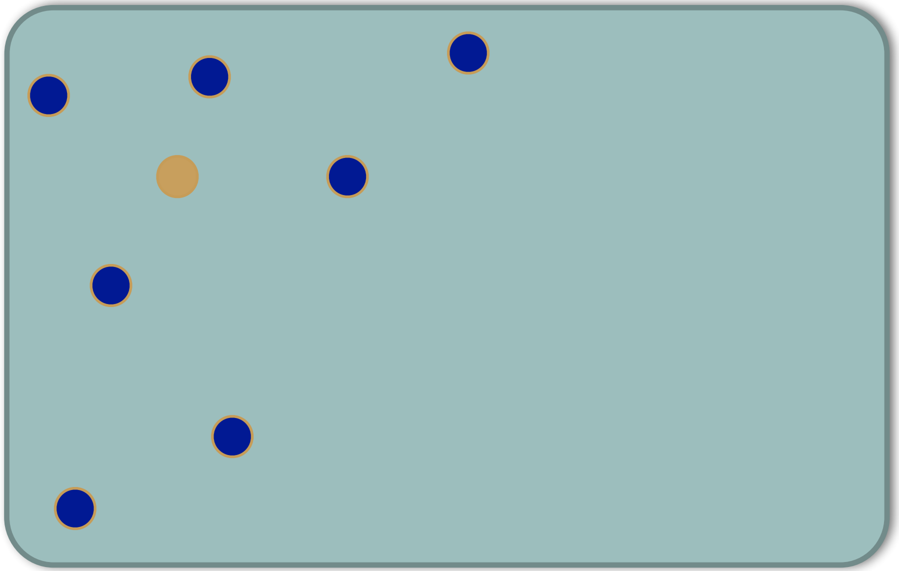
NN Filtering - Step 2

- Step 2: Build a predictor based on the relevant data

NN Filtering - Step 2

- Step 2: Build a predictor based on the relevant data

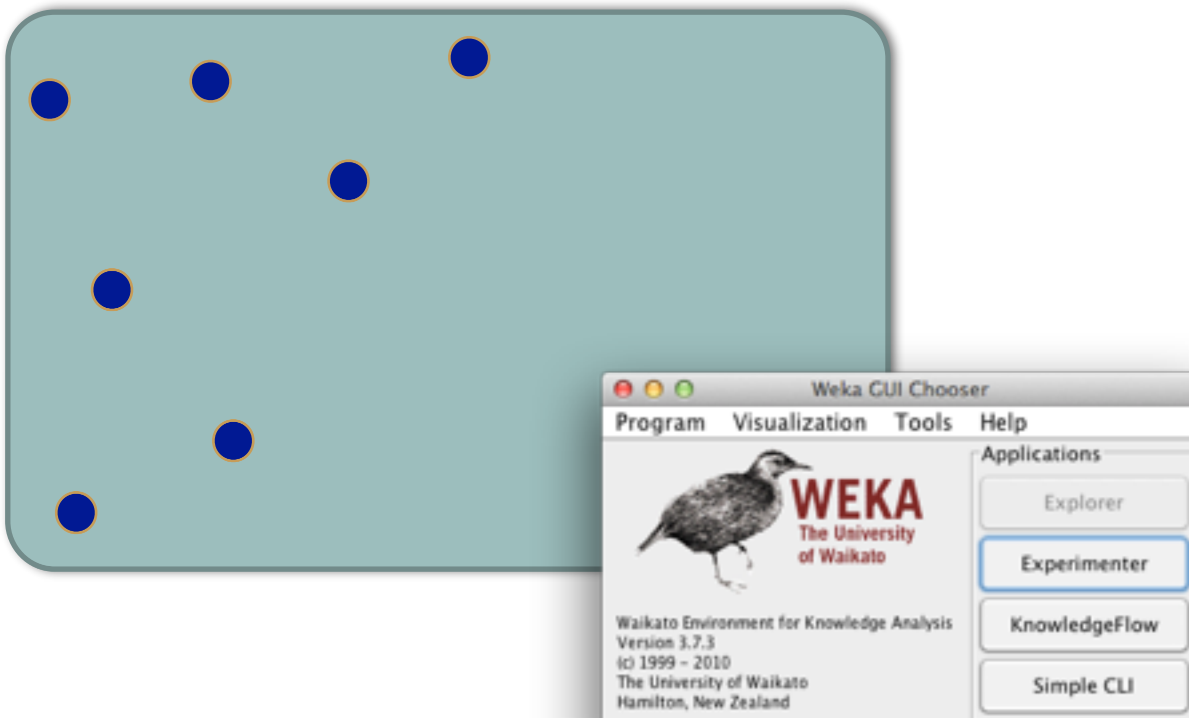
Take random sample of 90% of relevant training data



NN Filtering - Step 2

- Step 2: Build a predictor based on the relevant data

Build predictor, e.g., naive bayes



NN-Filtering Sample Result -- Software Defect Prediction

- CM1 software defect prediction when using data from other projects:
  - False positive: 91%
  - True positive: 98%

- When using NN-filtering with data from other projects:
  - False positive: 44%
  - True positive: 82%

- When using data from within a given project:
  - False positive: 33%
  - True positive: 80%

Why NN Filtering? When?

Why?

- NN filtering finds local regions that are relevant to a given context.
- It can transfer data between projects.

When?

- Helpful as an alternative when there is not much data from within a given environment.
  - E.g., defect predictor for first version of a software.
- Adequate when the number of neighbours is large enough to create an accurate model.
  - E.g., in software defect prediction.

Test Essential Assumption Knowledge (TEAK) is a relevancy filter that may be more adequate for smaller data sets.

Test Essential Assumption Knowledge (TEAK)

- Learning algorithms are based on assumptions.
  - E.g., linear regression assumes linearity, k-nearest neighbour assumes that locality implies homogeneity.

E. Kocaguneli, T. Menzies, A. Bener, J. Keung "Exploiting the Essential Assumptions of Analogy-Based Effort Estimation", IEEE Transactions on Software Engineering, 2012.

E. Kocaguneli, T. Menzies, E. Mendes "Transfer Learning in Effort Estimation", Empirical Software Engineering Journal, 2014.

Test Essential Assumption Knowledge (TEAK)

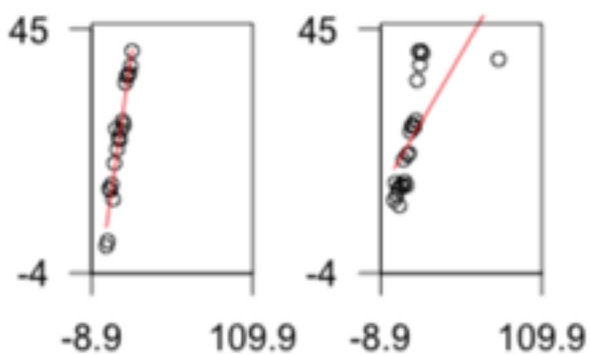
- Learning algorithms are based on assumptions.
  - E.g., linear regression assumes linearity, k-nearest neighbour assumes that locality implies homogeneity.

Assumptions are not always satisfied -- outliers!

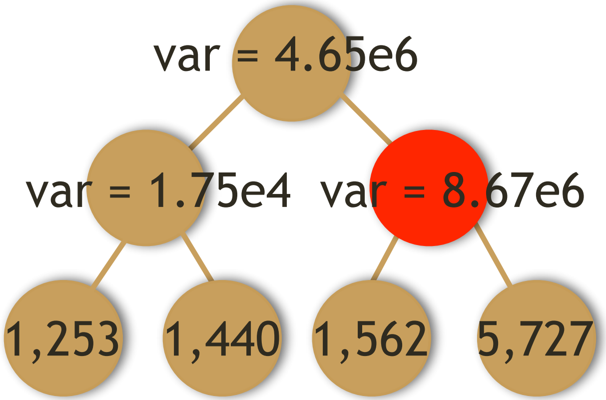
Figure from: http://nicodewet.com/2011/10/02/java-exception-rule-book/

TEAK - Eliminating Confusing Situations

Outliers can confuse algorithms, hindering their performance.



Linear regression



Greedy Agglomerative Clustering (GAC)
K-Nearest Neighbors

How TEAK Works

- Step 1: Select a prediction system
- Step 2: Identify its essential assumptions
- Step 3: Identify assumption violation
- Step 4: Remove violations
- Step 5: Execute the modified system

E. Kocaguneli, T. Menzies, A. Bener, J. Keung "Exploiting the Essential Assumptions of Analogy-Based Effort Estimation", IEEE Transactions on Software Engineering, 2012.

How TEAK Works

- Step 1: Select a prediction system
  - GAC k-NN
- Step 2: Identify its essential assumptions
- Step 3: Identify assumption violation
- Step 4: Remove violations
- Step 5: Execute the modified system

How TEAK Works

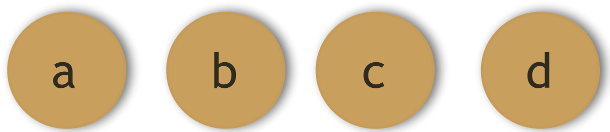
- Step 1: Select a prediction system
  - GAC k-NN
- Step 2: Identify its essential assumptions
  - Locality leads to homogeneity
- Step 3: Identify assumption violation
- Step 4: Remove violations
- Step 5: Execute the modified system

How TEAK Works

- Step 1: Select a prediction system
  - GAC k-NN
- Step 2: Identify its essential assumptions
  - Locality leads to homogeneity
- Step 3: Identify assumption violation
- Step 4: Remove violations
- Step 5: Execute the modified system

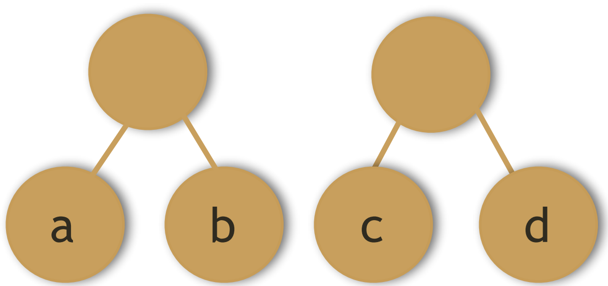
Identifying Assumption Violation for k-NN

- Create a tree by using GAC
  - For predictive tasks you would check the input attributes of the examples



Identifying Assumption Violation for k-NN

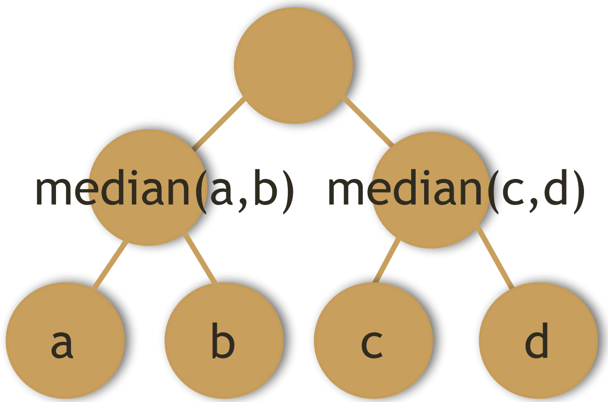
- Create a tree by using GAC
  - For predictive tasks you would check the input attributes of the examples



Group two closest pairs together based on input attributes

Identifying Assumption Violation for k-NN

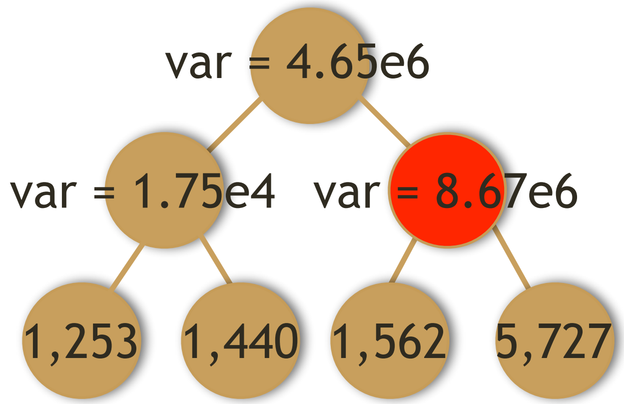
- Create a tree by using GAC
  - For predictive tasks you would check the input attributes of the examples



Group two closest pairs together based on input attributes

Identifying Assumption Violation for k-NN

- Create a tree by using GAC
- Traverse the tree to find increases in variance
  - For predictive tasks, this variance should be checked based on the output attribute



How TEAK Works

- Step 1: Select a prediction system
  - k-NN
- Step 2: Identify its essential assumptions
  - Locality leads to homogeneity
- Step 3: Identify assumption violation
- Step 4: Remove violations
  - Prune subtree that violates assumption
- Step 5: Execute the modified system

How TEAK Works

- Step 1: Select a prediction system
  - k-NN
- Step 2: Identify its essential assumptions
  - Locality leads to homogeneity
- Step 3: Identify assumption violation
- Step 4: Remove violations
  - Prune subtrees that violates assumption
- Step 5: Execute the modified system
  - Create a new GAC tree

Why TEAK? When?

Why?

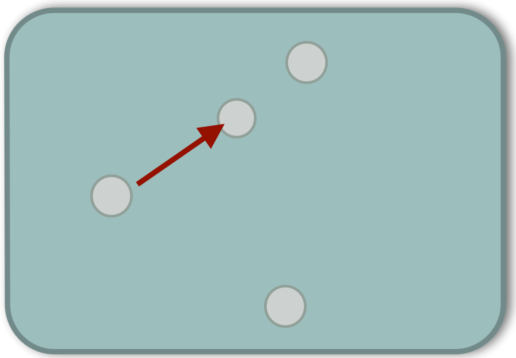
- TEAK eliminates examples that cause confusion and increase uncertainty of predictions
- It helps to improve models' predictive performance
- TEAK GAC k-NN can be used to remove not only confusing examples from within a given source, but also confusing examples from different sources
  - TEAK can thus be used for transfer learning

When?

- It is expected to be particularly useful when we don't have much data, i.e., when few outliers can cause great damage
  - E.g., software effort estimation

Popularity-Based Filtering

- Eliminate training examples that are unpopular, i.e., that are less often neighbors of other training examples.
- This has been shown to help overcoming problems with missing values.



k-nearest neighbors

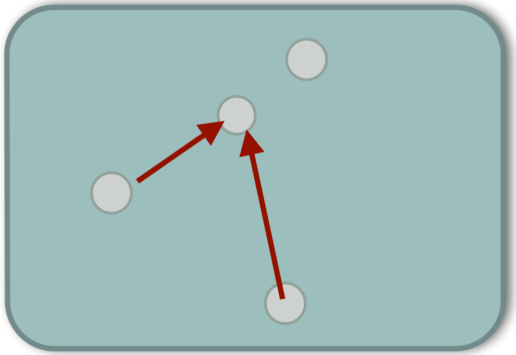
Euclidean distance based on input features

Ekrem Kocaguneli, Tim Menzies, Jairus Hihn and Byeong Ho Kang. Size Doesn't Matter? On the Value of Software Size Features for Effort Estimation

Popularity-Based Filtering

- Eliminate training examples that are unpopular, i.e., that are less often neighbors of other training examples.
- This has been shown to help overcoming problems with missing values.

Training data



k-nearest neighbors

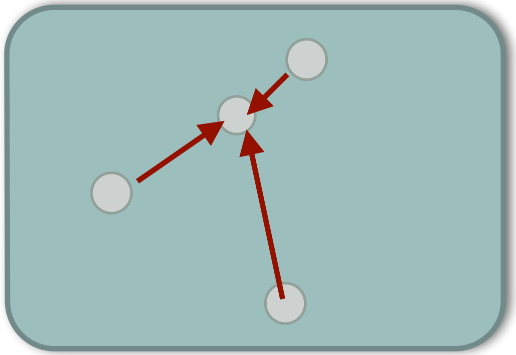
Euclidean distance based on input features

Ekrem Kocaguneli, Tim Menzies, Jairus Hihn and Byeong Ho Kang. Size Doesn't Matter? On the Value of Software Size Features for Effort Estimation

Popularity-Based Filtering

- Eliminate training examples that are unpopular, i.e., that are less often neighbors of other training examples.
- This has been shown to help overcoming problems with missing values.

Training data



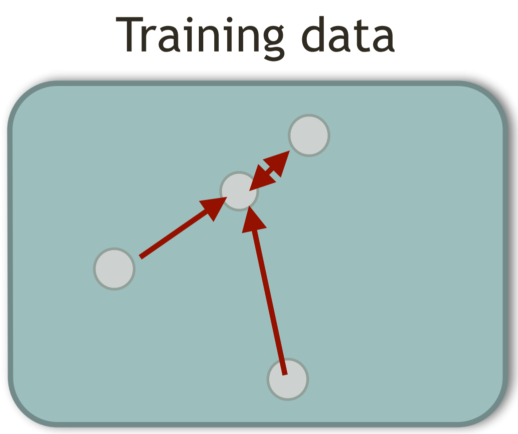
k-nearest neighbors

Euclidean distance based on input features

Ekrem Kocaguneli, Tim Menzies, Jairus Hihn and Byeong Ho Kang. Size Doesn't Matter? On the Value of Software Size Features for Effort Estimation

Popularity-Based Filtering

- Eliminate training examples that are unpopular, i.e., that are less often neighbors of other training examples.
- This has been shown to help overcoming problems with missing values.



k-nearest neighbors

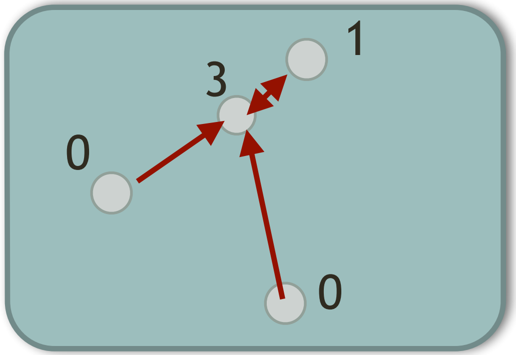
Euclidean distance based on input features

Ekrem Kocaguneli, Tim Menzies, Jairus Hihn and Byeong Ho Kang. Size Doesn’t Matter? On the Value of Software Size Features for Effort Estimation

Popularity-Based Filtering

- Eliminate training examples that are unpopular, i.e., that are less often neighbors of other training examples.
- This has been shown to help overcoming problems with missing values.

Training data



k-nearest neighbors

Euclidean distance based on input features

Ekrem Kocaguneli, Tim Menzies, Jairus Hihn and Byeong Ho Kang. Size Doesn't Matter? On the Value of Software Size Features for Effort Estimation

Popularity-Based Filtering

- Eliminate training examples that are unpopular, i.e., that are less often neighbors of other training examples.
- This has been shown to help overcoming problems with missing values.

k-nearest neighbors

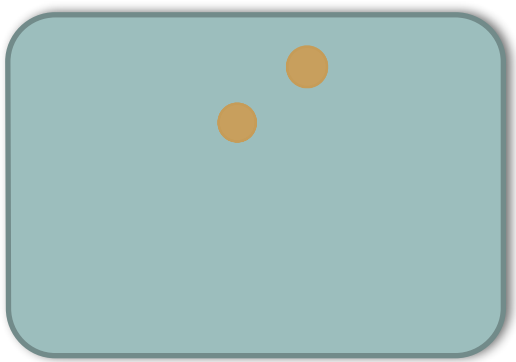
Euclidean distance based on input features

Ekrem Kocaguneli, Tim Menzies, Jairus Hihn and Byeong Ho Kang. Size Doesn’t Matter? On the Value of Software Size Features for Effort Estimation

Popularity-Based Filtering

- Eliminate training examples that are unpopular, i.e., that are less often neighbors of other training examples.
- This has been shown to help overcoming problems with missing values.

Relevant data



Add most popular examples that lead to considerable decreases in error

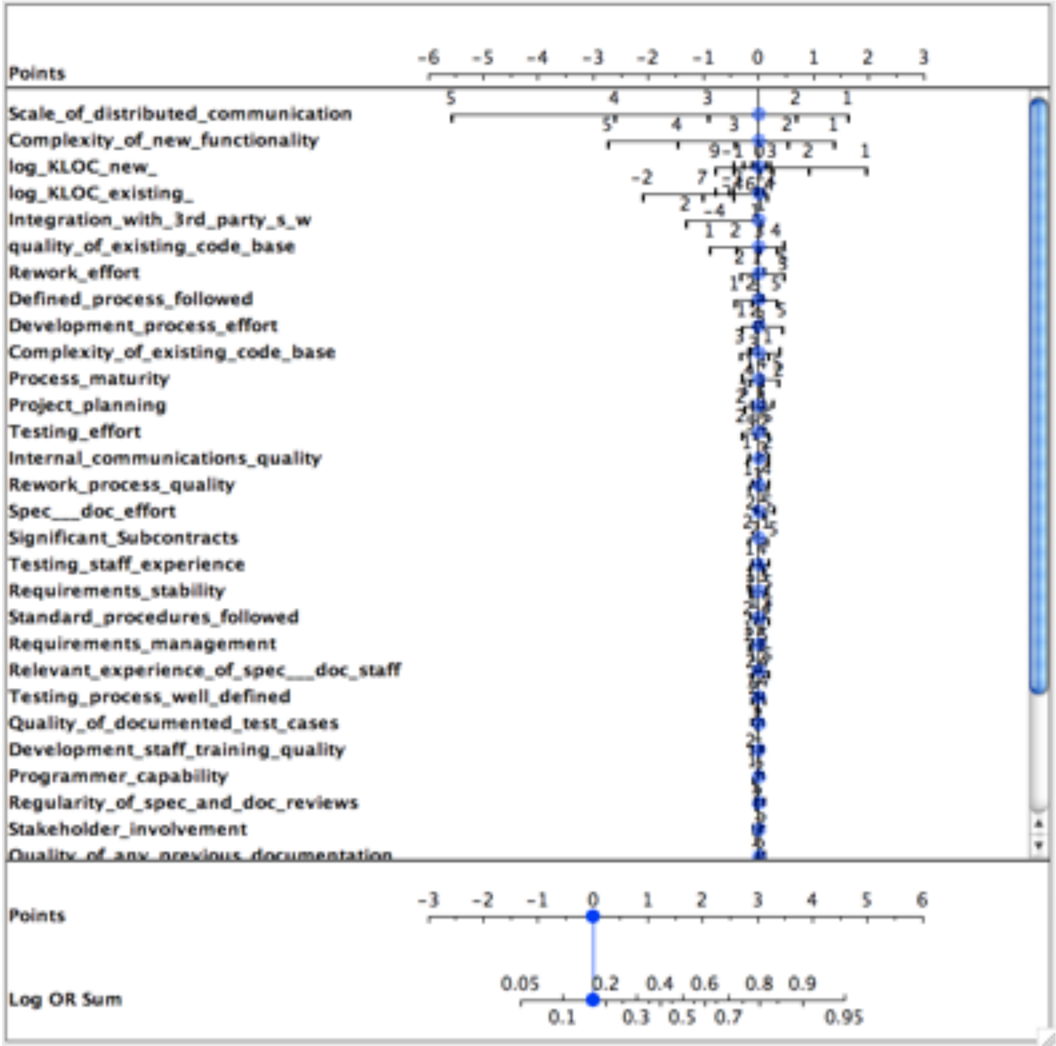
Ekrem Kocaguneli, Tim Menzies, Jairus Hihn and Byeong Ho Kang. Size Doesn't Matter? On the Value of Software Size Features for Effort Estimation

Ignoring Data

Column pruning	Row pruning	Range pruning
• irrelevancy removal • better predictions	• outliers • cross-company learning • handling missing values • privacy • anomaly detection • incremental learning	• contrast • goals

And What About Range Pruning?

- Classes **x**,**y**
  - **Fx**, **Fy**
    - frequency of discretized ranges in **x**,**y**
  - Log Odds Ratio
    - log(**Fx**/**Fy**)
    - Is zero if no difference in **x**,**y**
- E.g. Data from Norman Fenton's Bayes nets discussing software defects = **yes**, **no**
- Do most ranges contribute to determination of defects? **no**
- Restrict discussion to just most powerful ranges



Range Pruning

Contrast pruning

- prune away ranges that do not contribute to differences within the data.

Goal pruning

- prune away ranges that do not effect final decisions.

Learning from “powerful” ranges

find good housing in Boston

Decision tree learning on 14 features and 506 houses

WHICH

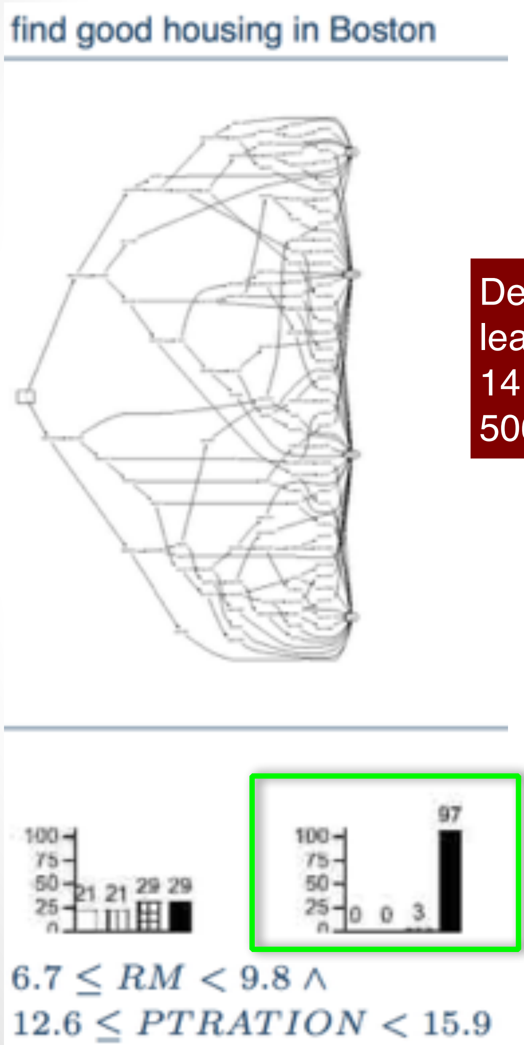
$6.7 \leq RM < 9.8 \wedge$
$12.6 \leq PTRATION < 15.9$

Contrast Pruning Example

- Generate tiny models
  - Sort all ranges by their power
- WHICH
  1. Select any pair (favouring those with most power)
  2. Combine pair, compute its power
  3. Sort back into the ranges
  4. Goto 1
- Initially:
  - stack contains single ranges
- Subsequently
  - stack sets of ranges

Tim Menzies, Zach Milton, Burak Turhan, Bojan Cukic, Yue Jiang, Ayse Basar Bener: Defect prediction from static code features: current results, limitations, new approaches. Autom. Softw. Eng. 17(4): 375-407 (2010)

Learning from “powerful” ranges



Decision tree learning on 14 features and 506 houses

WHICH

Goal Pruning Example

- Report only summary of data that affects a decision
  - Sort all ranges by their power
  - Find minority of ranges and columns that distinguish between groups.
- Question?
  1. What predicts for higher house cost?

Tim Menzies, Zach Milton, Burak Turhan, Bojan Cukic, Yue Jiang, Ayse Basar Bener: Defect prediction from static code features: current results, limitations, new approaches. Autom. Softw. Eng. 17(4): 375-407 (2010)

Advantage of Range Pruning

Learning defect predictors	Reasoning via analogy	Associated rule learning

Learning defect predictors

- If you just explore the ranges that survive row and column pruning,
  - is inference faster?

Reasoning via analogy

- Any nearest neighbour method runs faster with row/ column pruning
  - Fewer rows to search
  - Fewer columns to compare

Associated rule learning

- Mine only matching rules on demand:
  - E.g. ROSE, Zimmermann et al., TSE04.
  - Constraints on antecedent. Mine only rules which are related to the antecedent.

Zimmermann, Thomas, et al. "Mining version histories to guide software changes." In 26th International Conference on Software Engineering (ICSE) 2004.

Ignoring Data

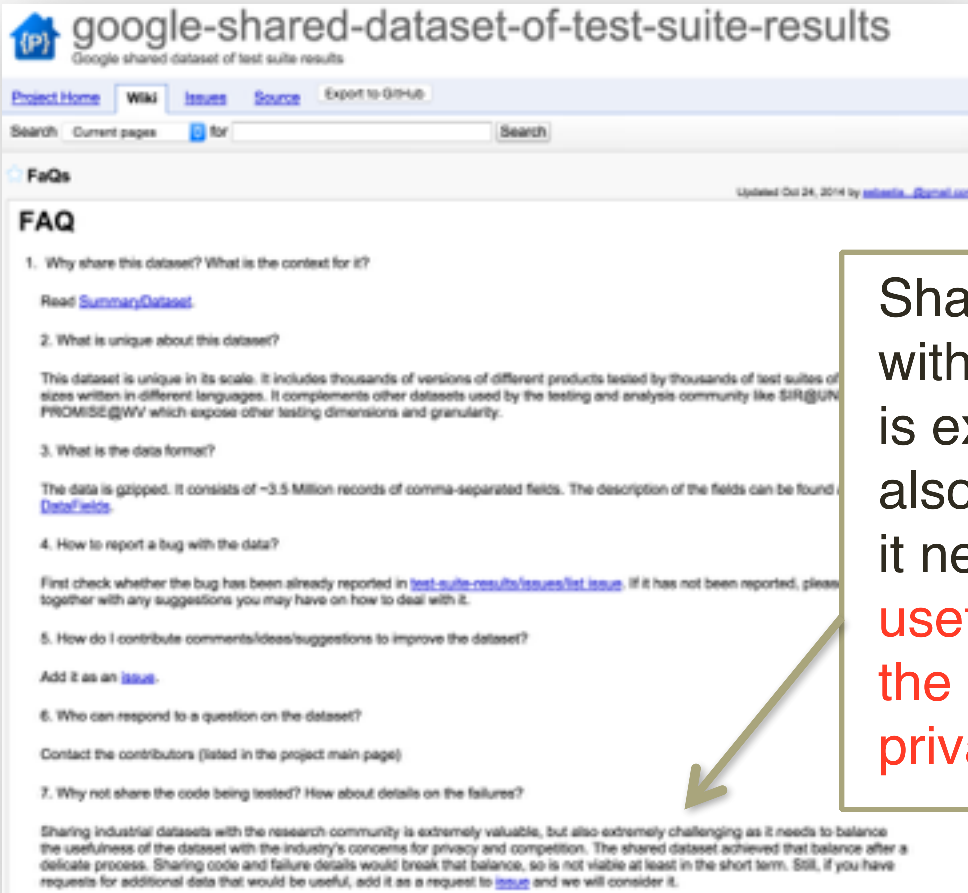
Column pruning	Row pruning	Range pruning
• irrelevancy removal • better predictions	• outliers • cross-company learning • handling missing values • privacy • anomaly detection • incremental learning LACE	• contrast • goals

1. Introduction
2. Sharing data
3. Privacy and sharing
4. Sharing models
5. Summary

Step 1: Throw most of it away

Step 2: Share the rest

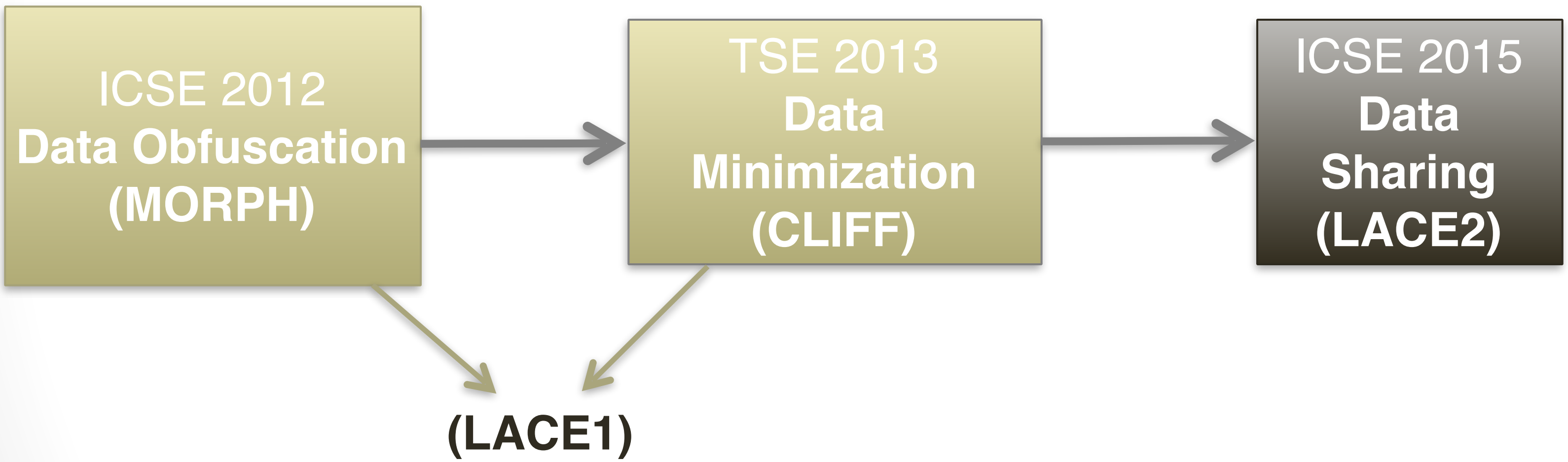
Balancing Usefulness & Privacy



Sharing industrial datasets with the research community is extremely valuable, but also extremely challenging as it needs to balance the usefulness of the dataset with the industry's concerns for privacy and competition.

S. Elbaum, A. Mclaughlin, and J. Penix, "The google dataset of testing results," June 2014. [Online].
Available: https://code.google.com/p/google-shared-dataset-of-test-suite-results

Challenge Accepted



F. Peters and T. Menzies, "Privacy and utility for defect prediction: Experiments with morph," in Proceedings of the 2012 International Conference on Software Engineering, ser. ICSE 2012. Piscataway, NJ, USA: IEEE Press, 2012, pp. 189–199.
F. Peters, T. Menzies, L. Gong, and H. Zhang, "Balancing privacy and utility in cross-company defect prediction," Software Engineering, IEEE Transactions on, vol. 39, no. 8, pp. 1054–1068, Aug 2013.

What We Want…

Features		Solution
Privacy	***Low sensitive attribute***	***?***
Utility	***Strong defect predictors.***	***?***
Cost	***Low memory requirements.***	***?***
	Fast runtime.	***?***

Sound Bites

LACE2 works

- because of the idea of software code re-use
  - In a set of programs, 32% were comprised of reused code (not including libraries). [Selby 2005]
- and one simple rule
  - don't share what others have already shared;

R. Selby, "Enabling reuse-based software development of large-scale systems," Software Engineering, IEEE Transactions on, vol. 31, no. 6, pp. 495–510, June 2005.

Research Questions

1. Does LACE2 offer more privacy than LACE1?
2. Does LACE2 offer more useful defect predictors than LACE1?
3. Are system costs of LACE2 (memory & runtime) worse than LACE1?

Roadmap

1. Privacy Threat (Sensitive Attribute Disclosure)
2. Cross Project Defect Prediction
3. LACE1 & LACE2
4. Experiments & Results
5. Why LACE?

Roadmap

1. Privacy Threat (Sensitive Attribute Disclosure)
2. Cross Project Defect Prediction
3. LACE1 & LACE2
4. Experiments & Results
5. Why LACE?

Sensitive Attribute Disclosure

- A privacy threat.
- Occurs when a target is associated with information about their sensitive attributes
  - e.g. software code complexity or actual software development times.
- **100 %** = zero sensitive attribute disclosure
- **0%** = total sensitive attribute disclosure

Queries	Original	Obfuscated	Breach
Q1	*0*	*0*	*yes*
Q2	*0*	*1*	*no*
Q3	*1*	*1*	*yes*
			no=1/3
			no=33%

J. Brickell and V. Shmatikov, “The cost of privacy: destruction of data-mining utility in anonymized data publishing,” in Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, ser. KDD ’08.
F. Peters and T. Menzies, “Privacy and utility for defect prediction: Experiments with morph,” in Proceedings of the 2012 International Conference on Software Engineering, ser. ICSE 2012. Piscataway, NJ, USA: IEEE Press, 2012, pp. 189–199.
F. Peters, T. Menzies, L. Gong, and H. Zhang, “Balancing privacy and utility in cross-company defect prediction,” Software Engineering, IEEE Transactions on, vol. 39, no. 8, pp. 1054–1068, Aug 2013.

Roadmap

1. Privacy Threat (Sensitive Attribute Disclosure)
2. Cross Project Defect Prediction
3. LACE1 & LACE2
4. Experiments & Results
5. Why LACE?

~~Cross Project~~ Defect Prediction

- For improving inspection efficiency

- But wait! I don't have enough data.

- Local data not always available [Zimmermann et al. 2009]
  - companies too small;
  - product in first release, no past data;
  - no time for data collection;

T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, "Cross-project defect prediction: a large scale experiment on data vs. domain vs. process." in ESEC/SIGSOFT FSE'09, 2009, pp. 91–100.

Cross Project Defect Prediction

- Use of data from other sources to build defect predictors for target data.
- Initial results (Zimmermann et al. 2009).

644 Cross Defect Prediction Experiments

T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, “Cross-project defect prediction: a large scale experiment on data vs. domain vs. process.” in ESEC/SIGSOFT FSE’09, 2009, pp. 91–100.

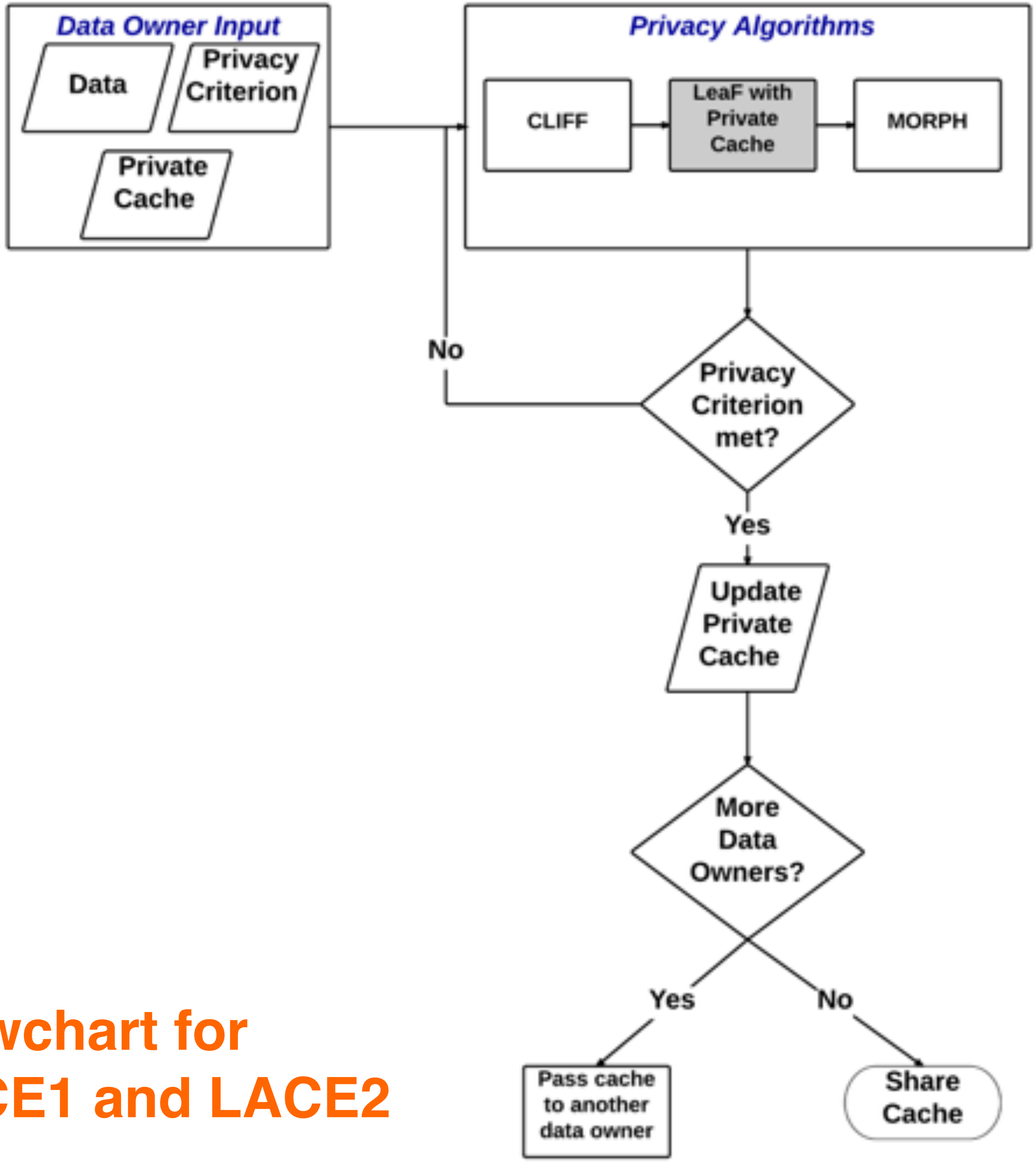
Cross Project Defect Prediction

- Use of data from other sources to build defect predictors for target data.
- Promising results when data from other sources are made similar to test data (Turhan et al. 2009, He et al. 2012,2013, Nam et al. 2013).
  - This raises privacy concerns;
  - Data must be shared.

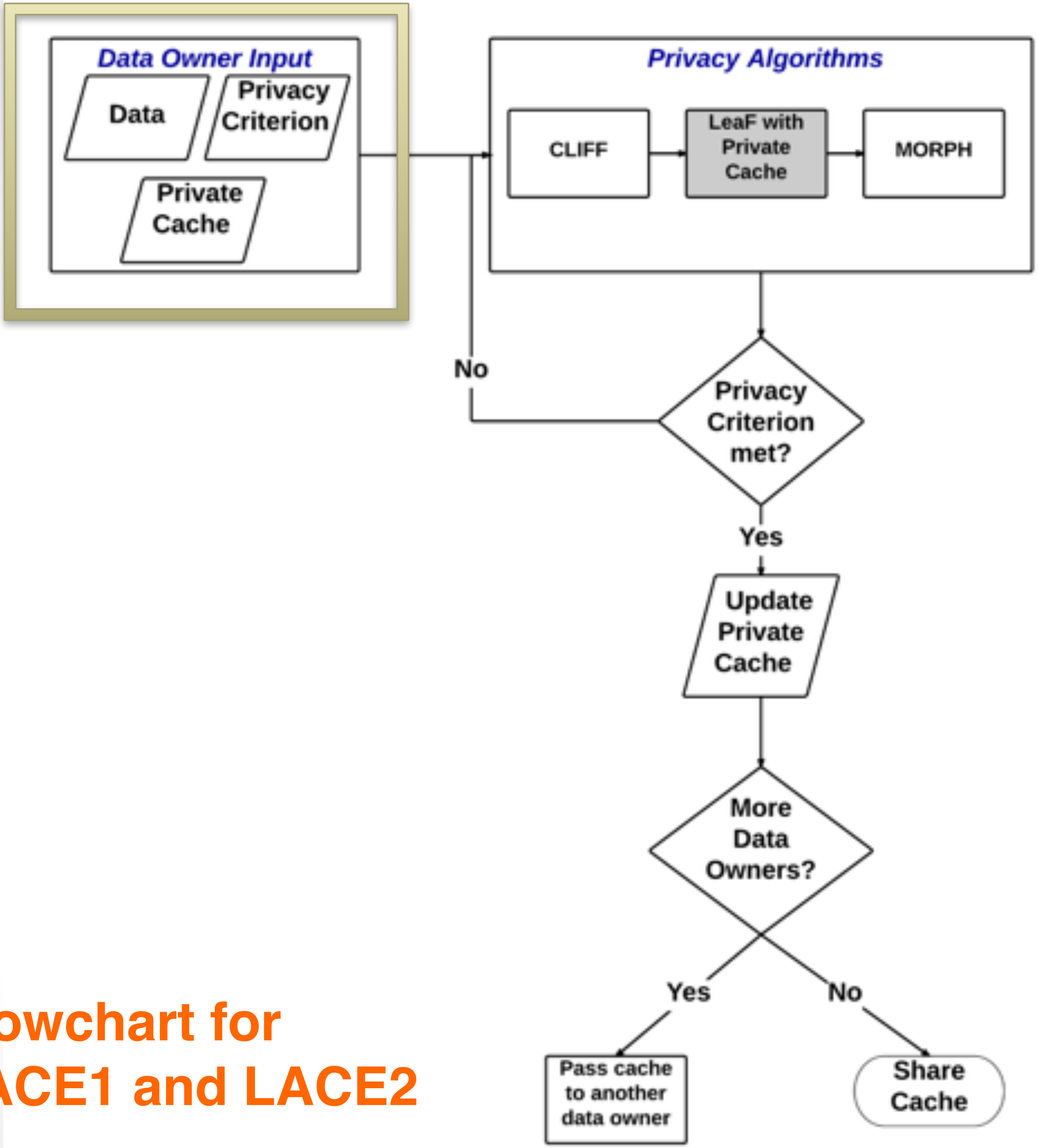
J. Nam, S. J. Pan, and S. Kim, “Transfer defect learning,” in ICSE’13. IEEE Press Piscataway, NJ, USA, 2013, pp. 802–811.
B. Turhan, T. Menzies, A. B. Bener, and J. Di Stefano, “On the relative value of cross-company and within-company data for defect prediction,” Empirical Software Engineering, vol. 14, pp. 540–578, 2009.
He, Zhimin, et al. "An investigation on the feasibility of cross-project defect prediction." Automated Software Engineering 19.2 (2012): 167-199.
He, Zhimin, et al. "Learning from open-source projects: An empirical study on defect prediction." Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on. IEEE, 2013.

Roadmap

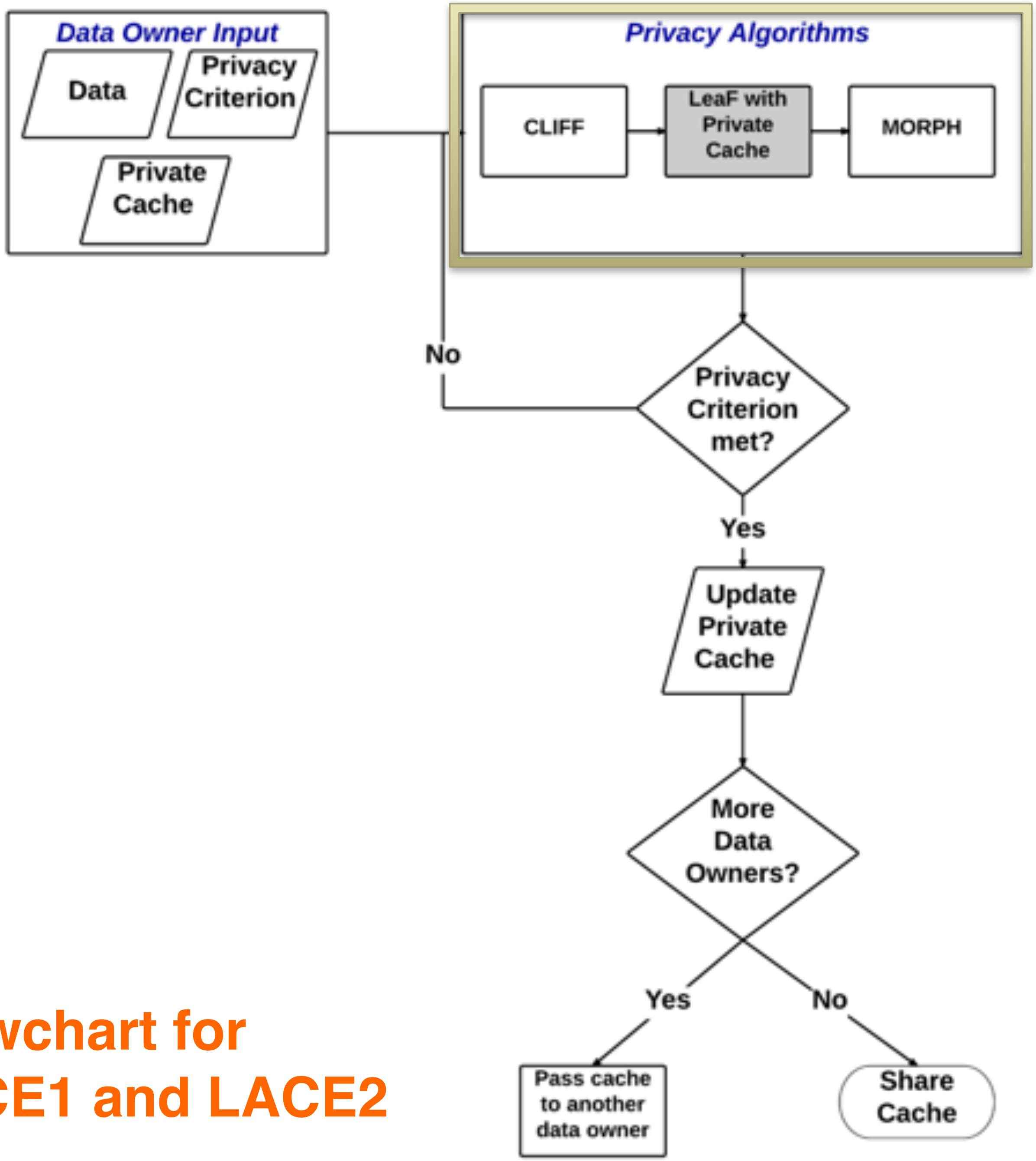
1. Privacy Threat (Sensitive Attribute Disclosure)
2. Cross Project Defect Prediction
3. LACE1 & LACE2
4. Experiments & Results
5. Why LACE?



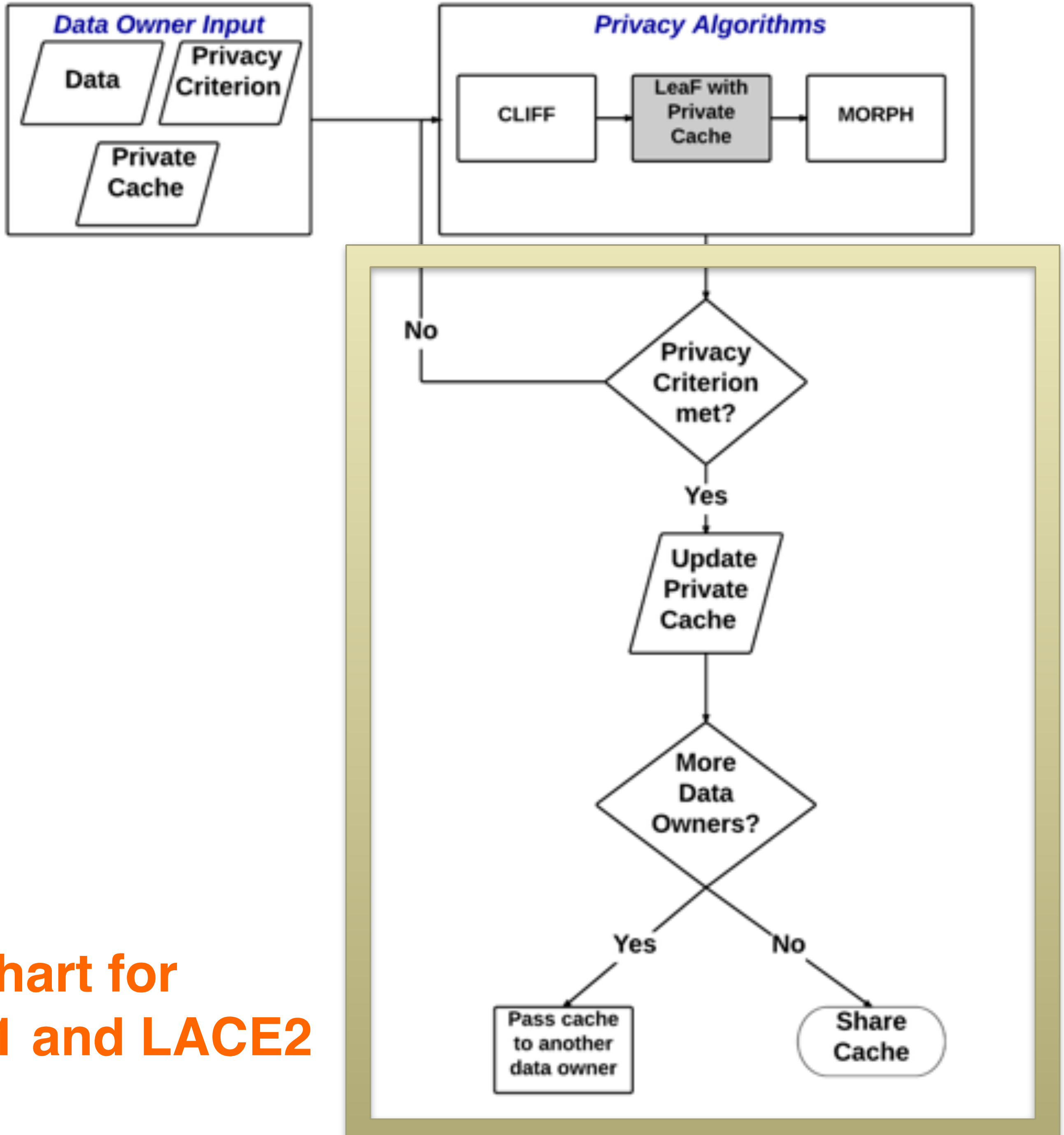
Flowchart for LACE1 and LACE2



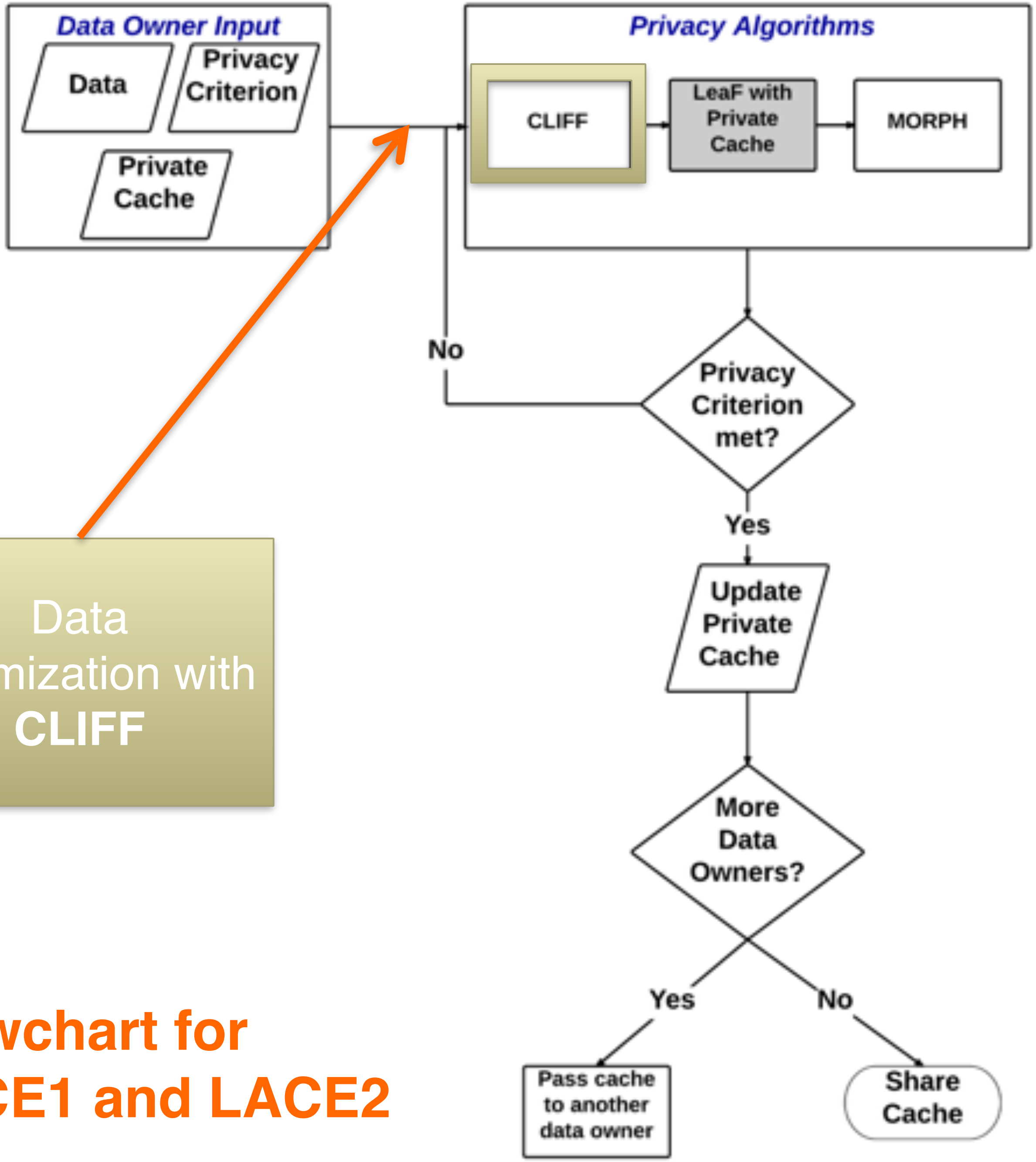
Flowchart for LACE1 and LACE2



Flowchart for LACE1 and LACE2



Flowchart for LACE1 and LACE2



Data Minimization with **CLIFF**

Flowchart for LACE1 and LACE2

Data Minimization

a	b	c	d	class
r1	*r1*	*r1*	*r2*	*yes*
r1	*r2*	*r3*	*r2*	*yes*
r1	*r3*	*r3*	*r3*	*yes*
r4	*r4*	*r4*	*r4*	*no*
r1	*r5*	*r5*	*r2*	*no*
r6	*r6*	*r6*	*r2*	*no*

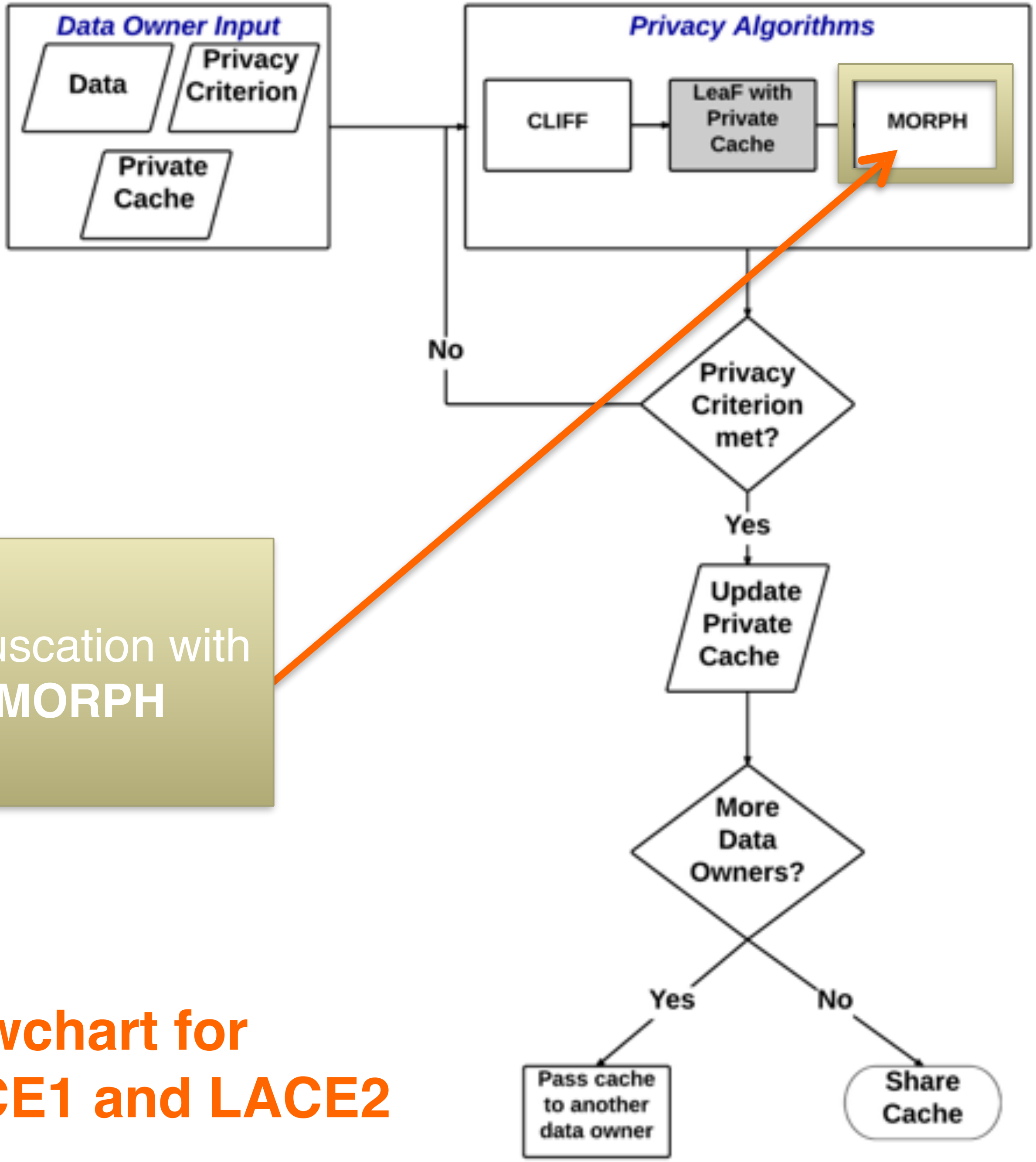
CLIFF: "a=r1" is powerful for selection for class=yes, i.e. more common in "yes" than "no".

- P(yes|r1) =

$$\frac{\text{like}(\text{yes}|r1)^2}{\text{like}(\text{yes}|r1) + \text{like}(\text{no}|r1)}$$

- Step 1: For each class find ranks of all values;
- Step 2: Multiply ranks of each row;
- Step 3: Select the most powerful rows of each class (top 20%).

F. Peters and T. Menzies, "Privacy and utility for defect prediction: Experiments with morph," in Proceedings of the 2012 International Conference on Software Engineering, ser. ICSE 2012. Piscataway, NJ, USA: IEEE Press, 2012, pp. 189–199.
F. Peters, T. Menzies, L. Gong, and H. Zhang, "Balancing privacy and utility in cross-company defect prediction," Software Engineering, IEEE Transactions on, vol. 39, no. 8, pp. 1054–1068, Aug 2013.



Obfuscation with **MORPH**

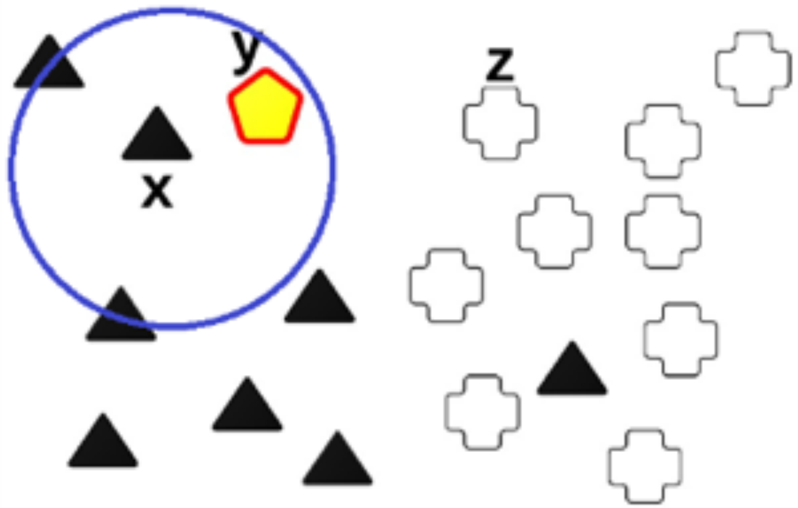
Flowchart for LACE1 and LACE2

Data Obfuscation

MORPH: Mutate the survivors no more than half the distance to their nearest unlike neighbor.

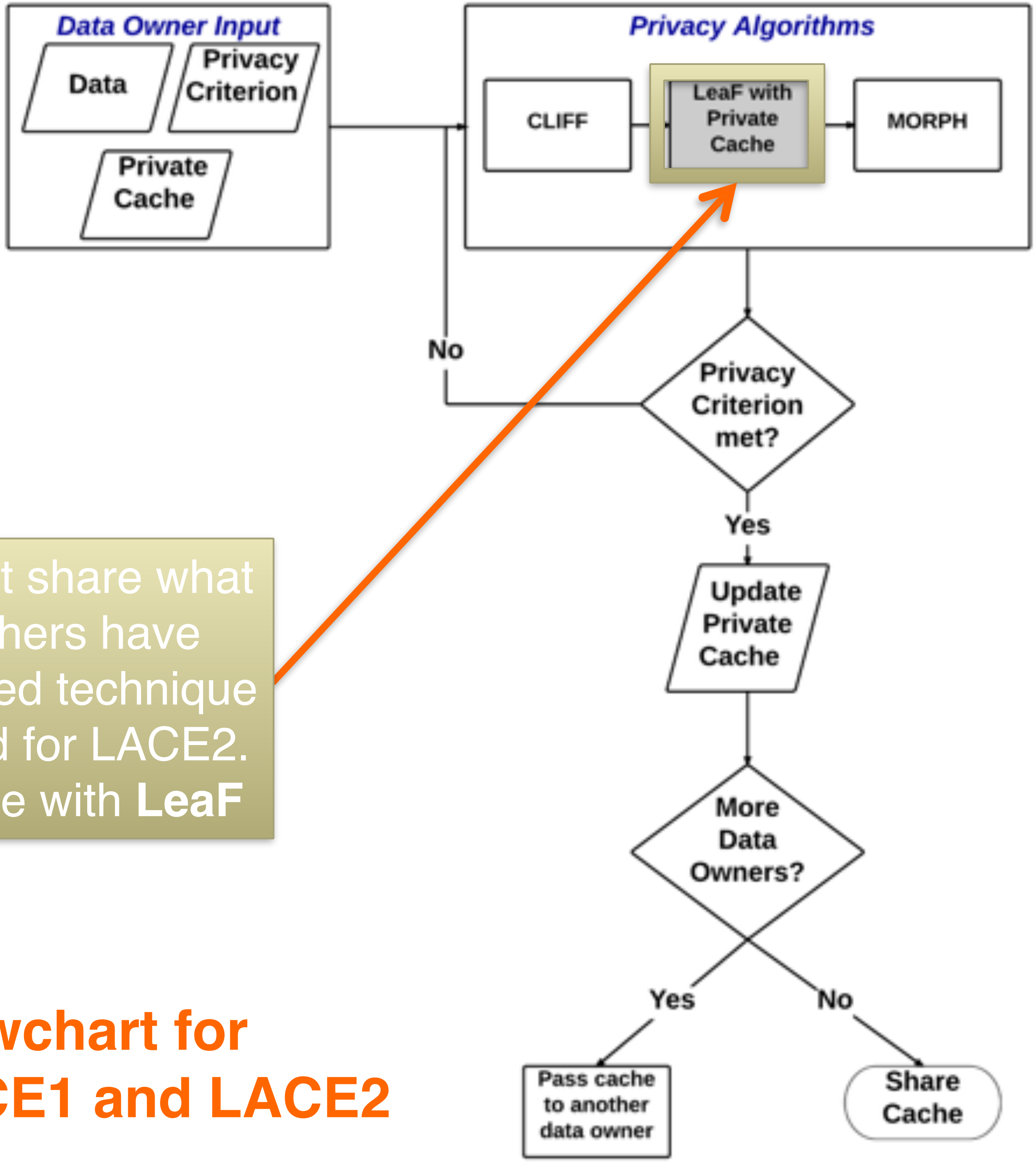
$$y = x \pm (x - z) * r$$

$$\alpha \leq r \leq \beta$$



- x is original instance;
- z is nearest unlike neighbor of x;
- y resulting MORPHed instance;
- r is random.

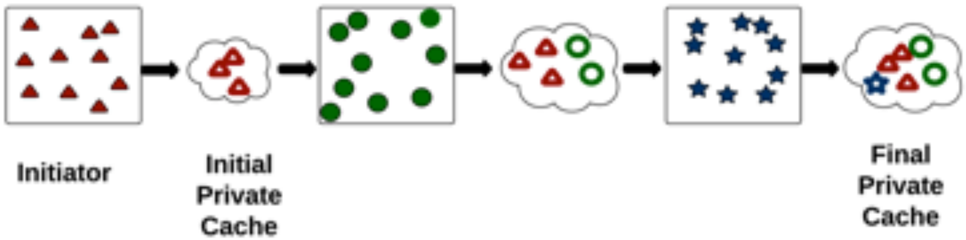
F. Peters and T. Menzies, “Privacy and utility for defect prediction: Experiments with morph,” in Proceedings of the 2012 International Conference on Software Engineering, ser. ICSE 2012. Piscataway, NJ, USA: IEEE Press, 2012, pp. 189–199.
F. Peters, T. Menzies, L. Gong, and H. Zhang, “Balancing privacy and utility in cross-company defect prediction,” Software Engineering, IEEE Transactions on, vol. 39, no. 8, pp. 1054–1068, Aug 2013.



Don't share what others have shared technique used for LACE2. Done with **LeaF**

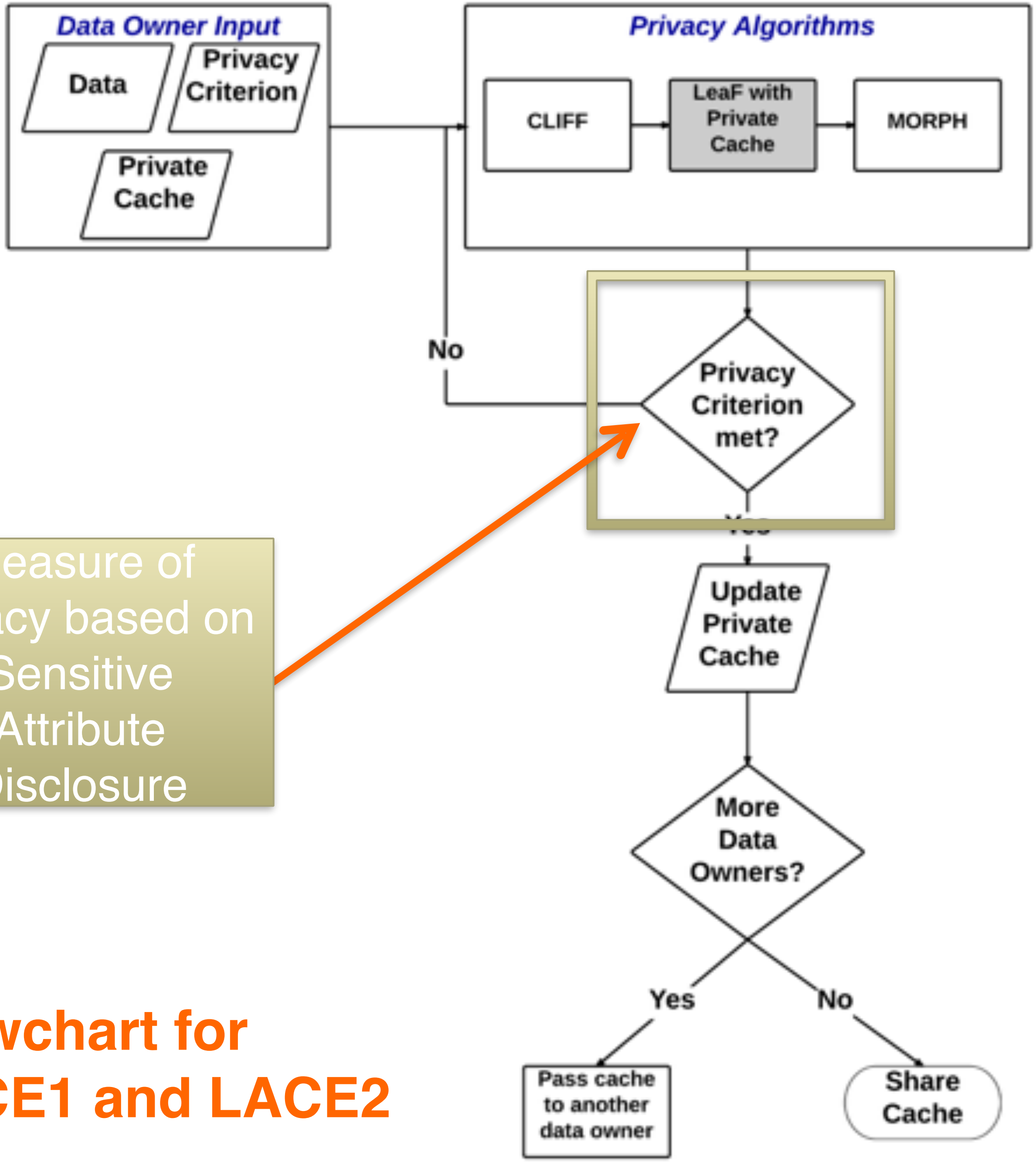
Flowchart for LACE1 and LACE2

Don’t Share What Others Share



- LACE2 : Learn from N software projects
  - from multiple data owners
- As you learn, play “pass the parcel”
  - The cache of reduced data
- Each data owner only adds its “leaders” to the passed cache
  - Morphing as they go
- Each data owner determines “leader” according to median distance
  - 100 random instances chosen
  - Find distance of nearest unlike neighbor for each
  - Get median distance

Duda, Richard O., Peter E. Hart, and David G. Stork. Pattern classification. John Wiley & Sons, 2012.
R. Selby, “Enabling reuse-based software development of large-scale systems,” Software Engineering, IEEE Transactions on, vol. 31, no. 6, pp. 495–510, June 2005.



Measure of privacy based on Sensitive Attribute Disclosure

Flowchart for LACE1 and LACE2

Roadmap

1. Privacy Threat (Sensitive Attribute Disclosure)
2. Cross Project Defect Prediction
3. LACE1 & LACE2
4. Experiments & Results
5. Why LACE?

Data

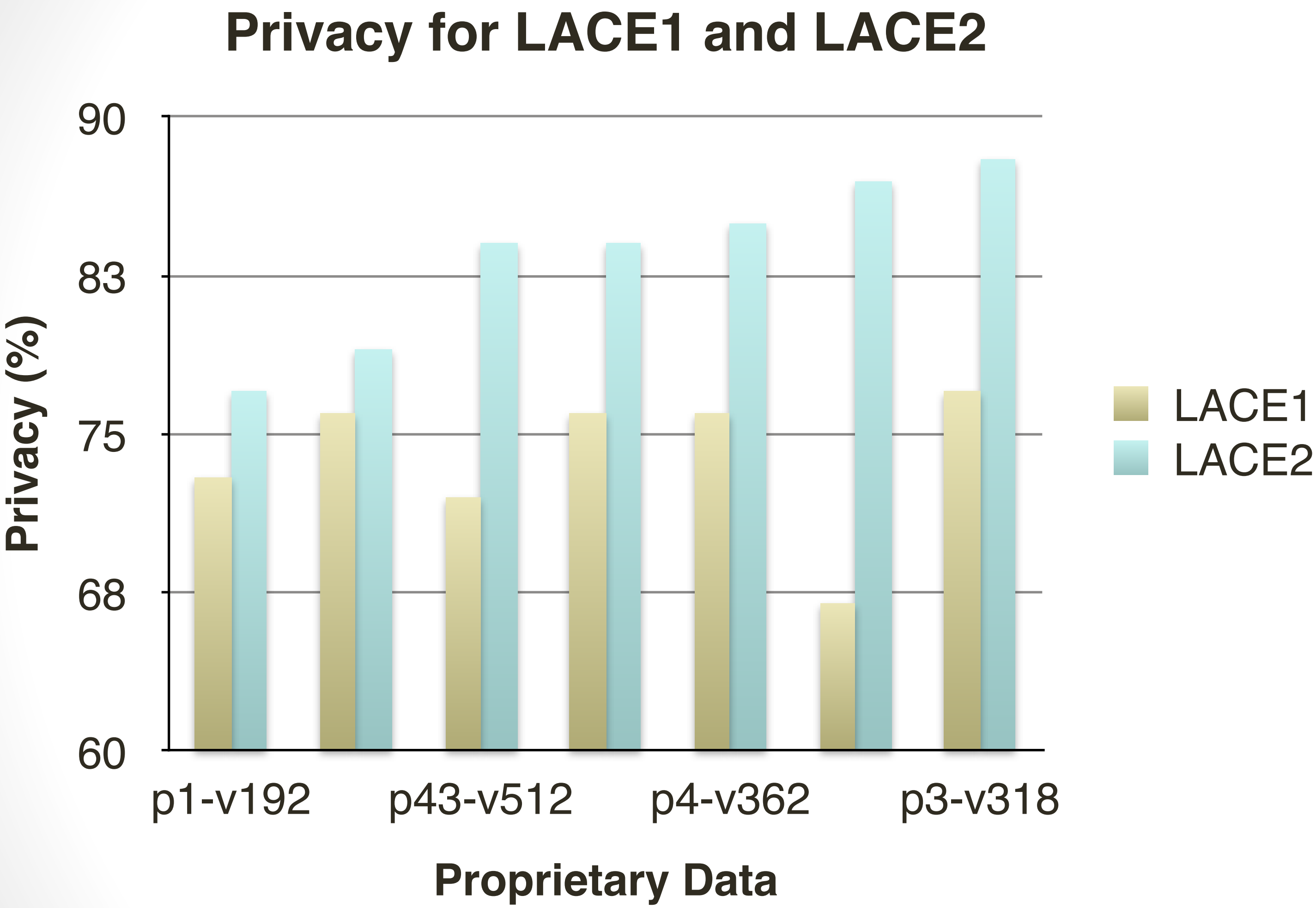
Defect Data	Type	# Instances	# Defects	% Defects
ant-1.7	open-source	1066	166	15.6
camel-1.6	open-source	1252	188	15.0
ivy-2.0	open-source	477	40	8.4
jEdit-4.1	open-source	644	79	12.3
lucene-2.4	open-source	536	203	37.9
poi-3.0	open-source	531	281	52.9
synapse-1.2	open-source	269	86	32.0
velocity-1.6	open-source	261	78	29.9
xalan-2.6	open-source	1170	411	35.1
xerces-1.3	open-source	545	69	12.7
prop1-ver192	proprietary	3692	85	2.3
prop2-ver276	proprietary	2472	334	13.5
prop3-ver318	proprietary	2440	365	15.0
prop4-ver362	proprietary	2865	213	7.4
prop5-ver185	proprietary	3260	268	8.2
prop42-ver454	proprietary	295	13	4.4
prop43-ver512	proprietary	2265	134	5.9

Experiment Design: RQ1

Does LACE2 offer more privacy than LACE1?

- 7 data owners follow LACE1 then LACE2 sharing techniques.
- Calculates the privacy level until privacy criterion (65%) is met.

Results: Privacy



RQ1: Does LACE2 offer more privacy than LACE1?

Result Summary

Features		LACE	LACE
Privacy	***Low sensitive attribute disclosure.***	***good***	***better***
Utility	***Strong defect predictors.***	***?***	
Cost	***Low memory requirements.***	***?***	
	Fast runtime.	***?***	

RQ1: Does LACE2 offer more privacy than LACE1?

Experiment Design: RQ2

Does LACE2 offer more useful defect predictors than LACE1?

- Cross project defect prediction experiment.
- Predictors built with k-nearest neighbour algorithm and private cache.

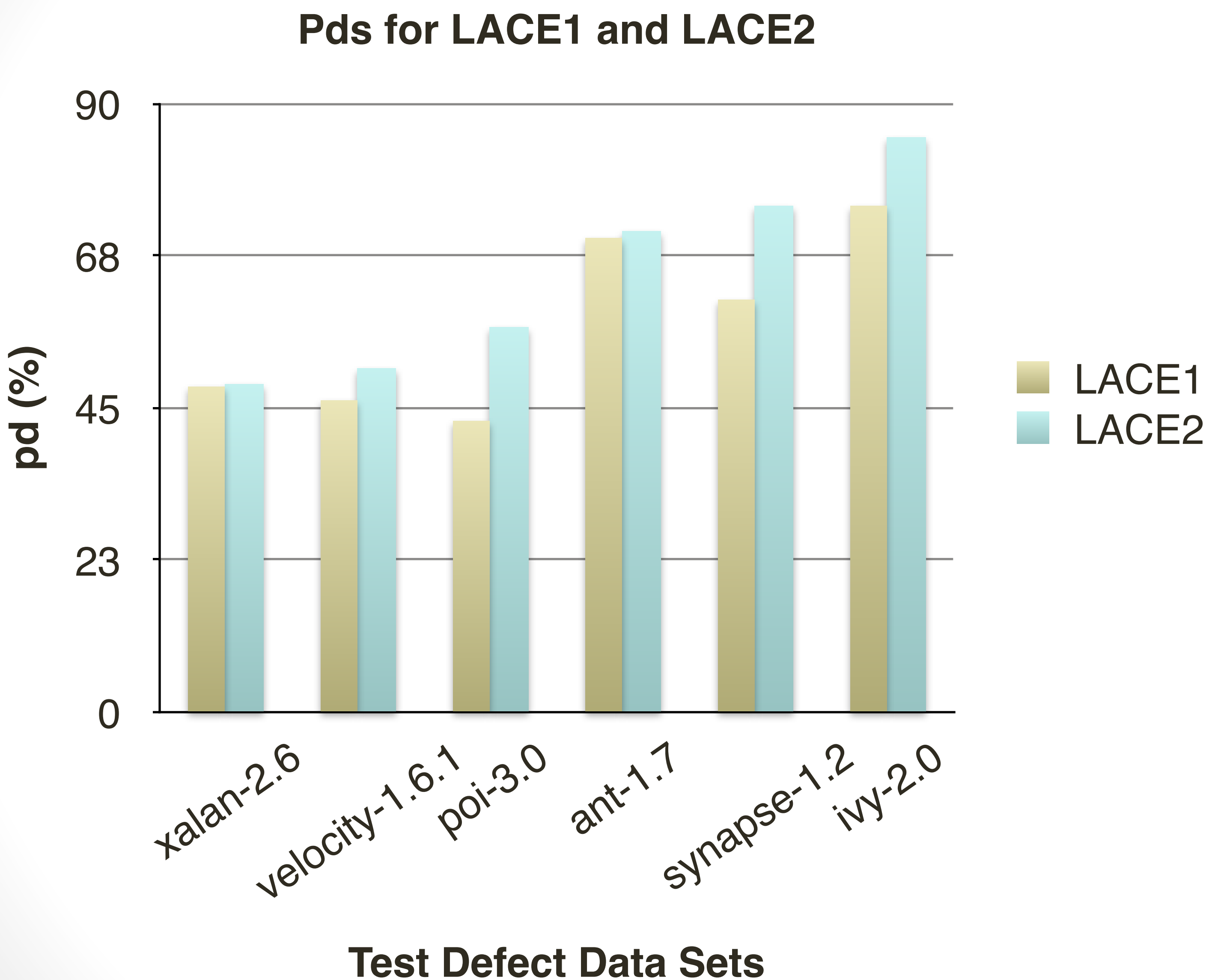
Performance Measures

- **TP (True Positive)**: defect-prone classes that are classified correctly;
- **FN (False Negative)**: defect-prone classes that are wrongly classified to be defect-free;
- **TN (True Negative)**: defect-free classes that are classified correctly;
- **FP (False Positive)**: defect-free classes that are wrongly classified to be defect-prone.

		Actual	
		yes	no
Predicted	yes	TP	FP
	no	FN	TN

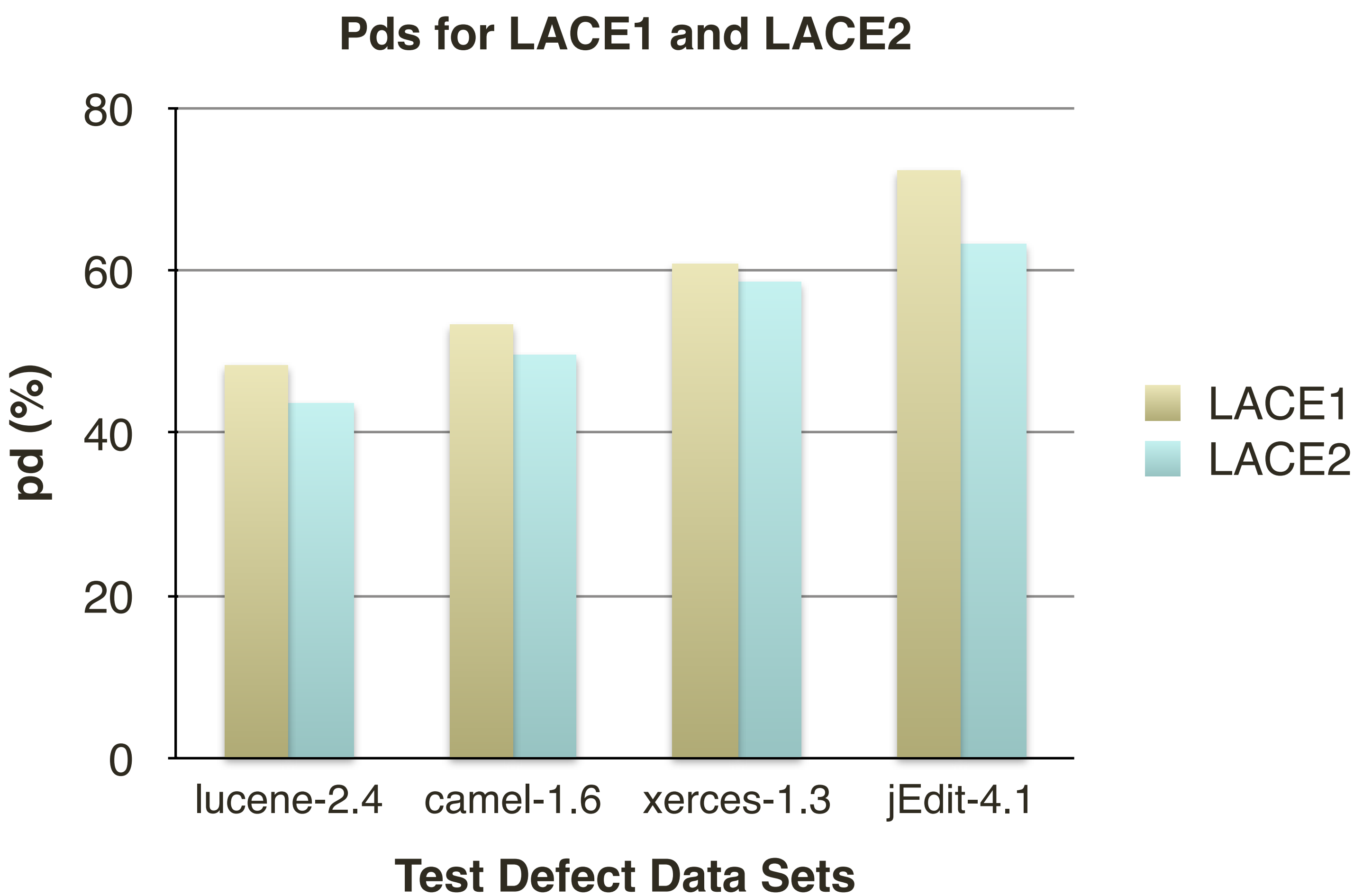
pd	$\frac{TP}{TP+FN}$
pf	$\frac{FP}{FP+TN}$
g-measure	$\frac{2*pd*(100-pf)}{pd+(100-pf)}$

Results: Defect Prediction



RQ2: Does LACE2 offer more useful defect predictors than LACE1?

Results: Defect Prediction



RQ2: Does LACE2 offer more useful defect predictors than LACE1?

Results: Defect Prediction

- Higher pfs (lower is best) than LACE1.

Pfs for LACE1 and LACE2		
Data	***LACE1***	***LACE2***
jEdit-4.1	***23.4***	***41.7***
ivy-2.0	***31.9***	***46.3***
xerces-1.3	***27.1***	***33.7***
ant-1.7	***34.3***	***36.8***
camel-1.6	***28.2***	***37.6***
lucene-2.4	***24.0***	***31.1***
xalan-2.6	***28.1***	***27.3***
velocity-1.6.1	***22.7***	***30.3***
synapse-1.2	***40.2***	***55.7***
poi-3.0	***16.4***	***23.8***

Results: Defect Prediction

- G-measures
  - No statistical difference between LACE1 and LACE2.

G-measures for LACE1 and LACE2		
Data	***LACE1***	***LACE2***
jEdit-4.1	***72.7***	***58.2***
ivy-2.0	***71.8***	***64.9***
xerces-1.3	***65.5***	***59.1***
ant-1.7	***67.6***	***64.9***
camel-1.6	***61.2***	***50.0***
lucene-2.4	***58.9***	***53.1***
xalan-2.6	***57.6***	***56.7***
velocity-1.6.1	***57.0***	***58.5***
synapse-1.2	***59.6***	***54.0***
poi-3.0	***57.0***	***63.9***

Result Summary

Features		LACE	LACE
Privacy	***Low sensitive attribute disclosure.***	***good***	***better***
Utility	***Strong defect predictors.***	***good***	***~good***
Cost	***Low memory requirements.***	***?***	
	Fast runtime.	***?***	

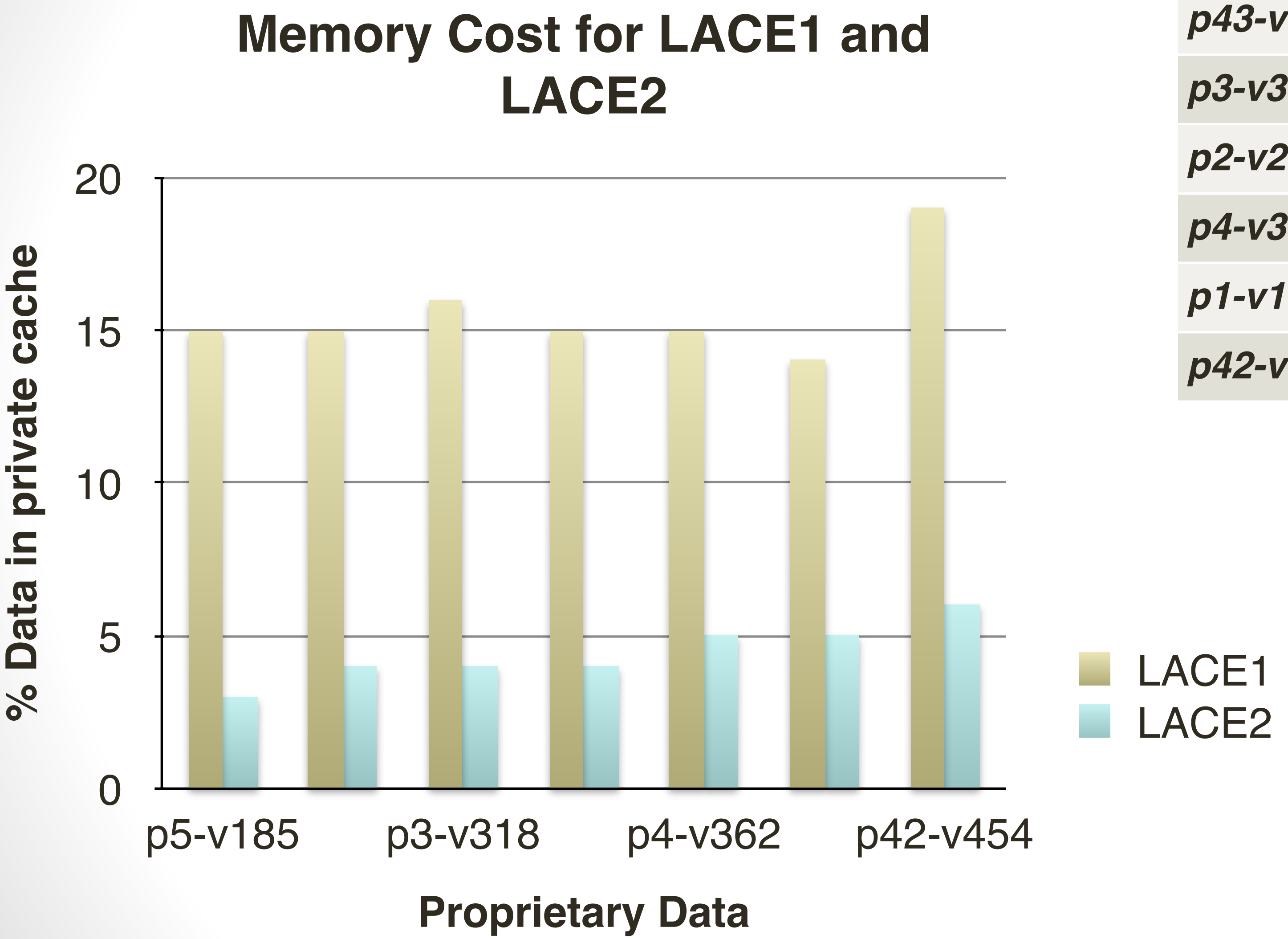
RQ2: Does LACE2 offer more useful defect predictors than LACE1?

Experiment Design: RQ3

Are system costs of LACE2 (memory & runtime) worse than LACE1?

- Memory = Calculated the percent of data each data owner contributes to the private cache.
- Runtime = Reported the time in seconds for creating each private cache for LACE1 and LACE2.

Results: Memory



Data	#
p5-v185	*3260*
p43-v512	*2265*
p3-v318	*2440*
p2-v276	*2472*
p4-v362	*2865*
p1-v192	*3692*
p42-v454	*295*

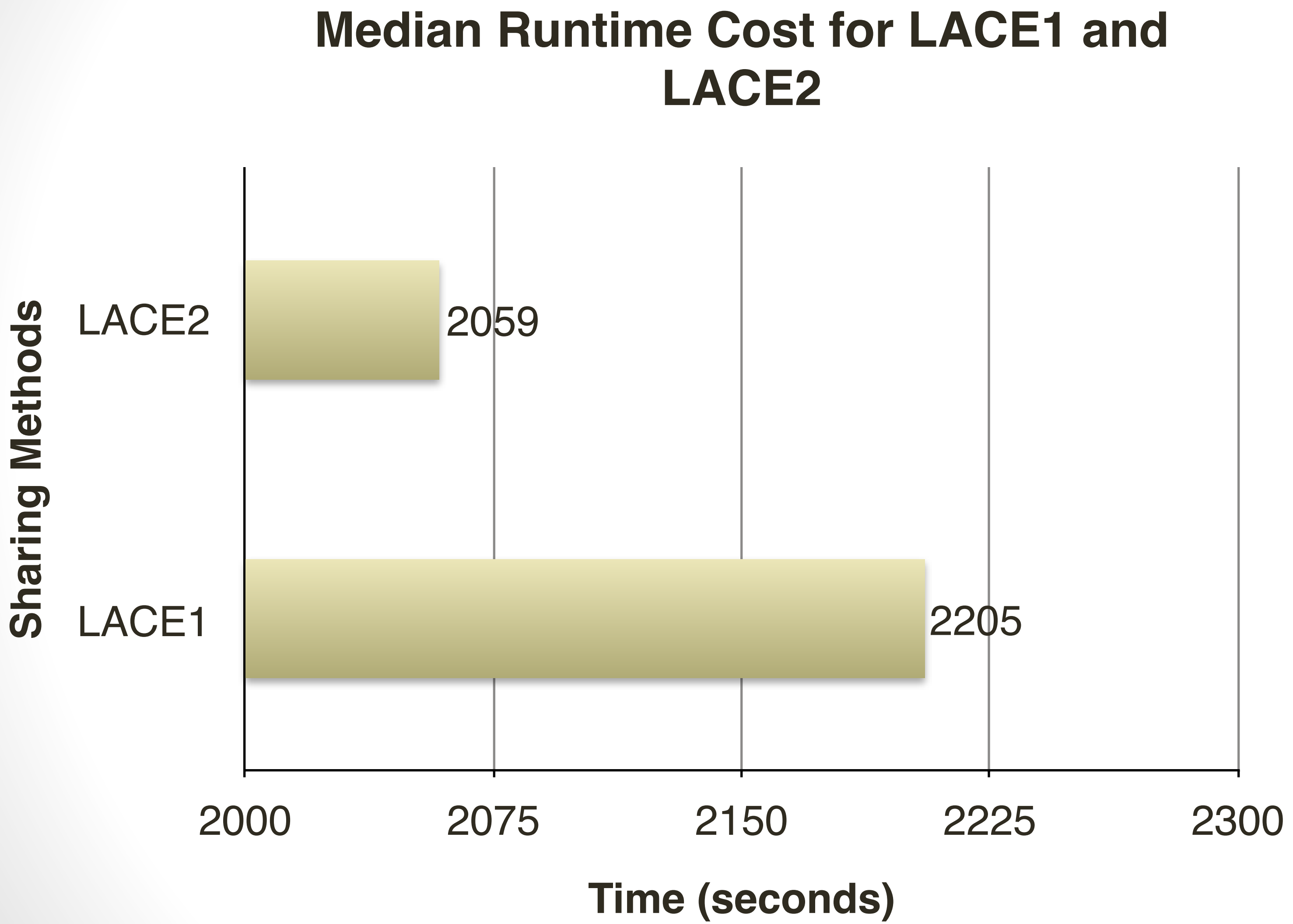
RQ3: Are system costs of LACE2 (memory) worse than LACE1?

Result Summary

Features		LACE	LACE
Privacy	***Low sensitive attribute disclosure.***	***good***	***better***
Utility	***Strong defect predictors.***	***good***	***~good***
Cost	***Low memory requirements.***	***good***	***better***
	Fast runtime.	***?***	***?***

RQ3: Are system costs of LACE2 (memory) worse than LACE1?

Results: Runtime



RQ3: Are system costs of LACE2 (runtime) worse than LACE1?

Result Summary

Features		LACE	LACE
Privacy	***Low sensitive attribute disclosure.***	***good***	***better***
Utility	***Strong defect predictors.***	***good***	***~good***
Cost	***Low memory requirements.***	***good***	***better***
	Fast runtime.	***good***	***good***

RQ3: Are system costs of LACE2 (runtime) worse than LACE1?

Roadmap

1. Privacy Threat (Sensitive Attribute Disclosure)
2. Cross Project Defect Prediction
3. LACE1 & LACE2
4. Experiments & Results
5. Why LACE?

Why LACE2?

- By using LACE2, you will be able to share a version of your data that is useful and satisfies your privacy criterion.
- LACE2 provides more privacy than LACE1.
  - Less data used.
  - Don’t share what others have shared.
- Comparable predictive efficacy to LACE1.
- LACE2’s sharing method, does not take more resources than LACE1.

Data from the Users Perspective


BY KATIE SHILTON

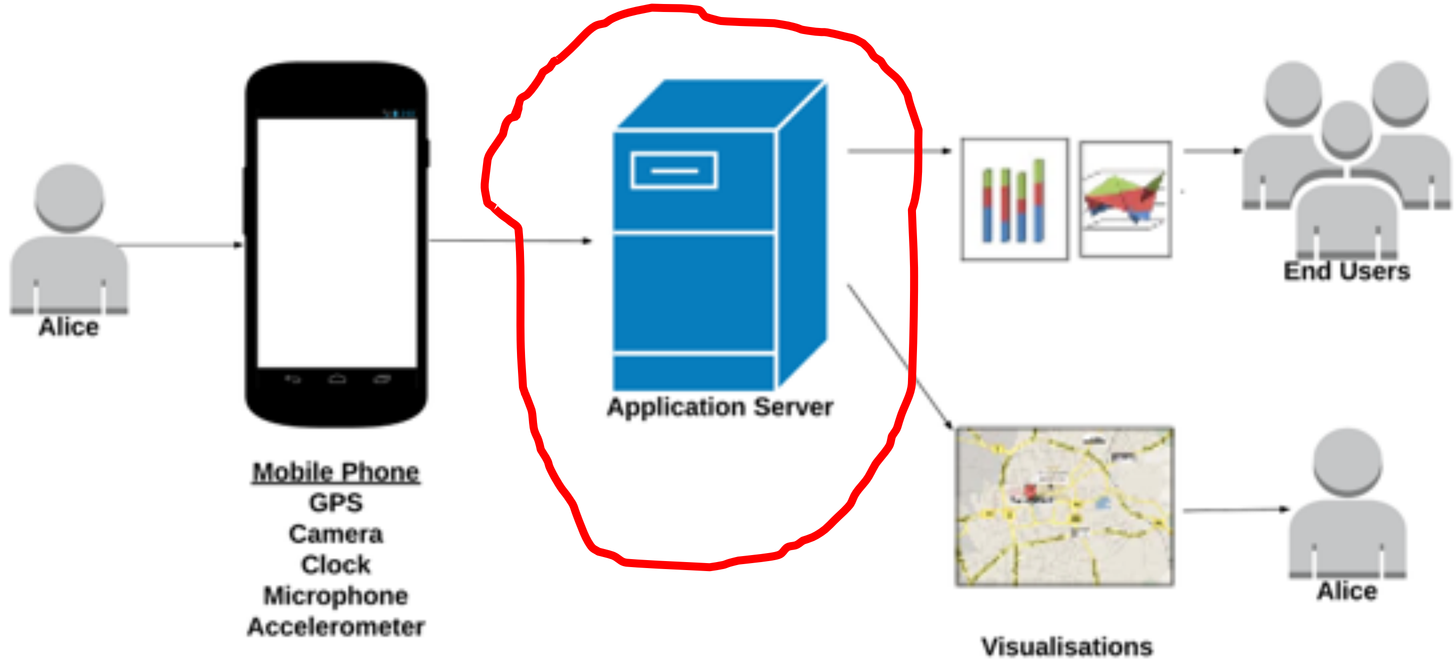
Four Billion Little Brothers?

Privacy, mobile phones, and ubiquitous data collection

Privacy is the ability to understand, choose, and control what personal information an individual shares, with whom, and for how long.

K. Shilton, "Four billion little brothers?: Privacy, mobile phones, and ubiquitous data collection," *Commun. ACM*, vol. 52, no. 11, pp. 48–53, Nov. 2009. [Online]. Available: http://doi.acm.org/10.1145/1592761.1592778

Some applications require personal data.



Attacker with access can breach user privacy.

The Conflict

Users have the opportunity to set privacy preferences but do not act on them in practice.

I. Krontiris, M. Langheinrich, and K. Shilton, "Trust and privacy in mobile experience sharing: future challenges and avenues for research," *Communications Magazine, IEEE*, vol. 52, no. 8, pp. 50–55, Aug 2014.

Privacy Zones Approach

Privacy by Design Principles

- Proactive not Reactive.
- Privacy as the default setting.
  - Set default privacy to only share privacy zone data.
    - In the zone = user’s habits (clusters)
    - Not in the zone = user’s irregular activities
  - User decides what to do with not in the zone data.
    - Ignore (Always share)
    - React (Obfuscate -> Success -> Share)
    - Prevent (Obfuscate -> No Success -> Do Not Share)
    - Terminate (End use of the application)

1. Introduction
2. Sharing data
3. Privacy and sharing
4. Sharing models
5. Summary

4a. Bagging
4b. Comba
4c. Multi-objective ensembles
4d. DCL
4e. Dycom

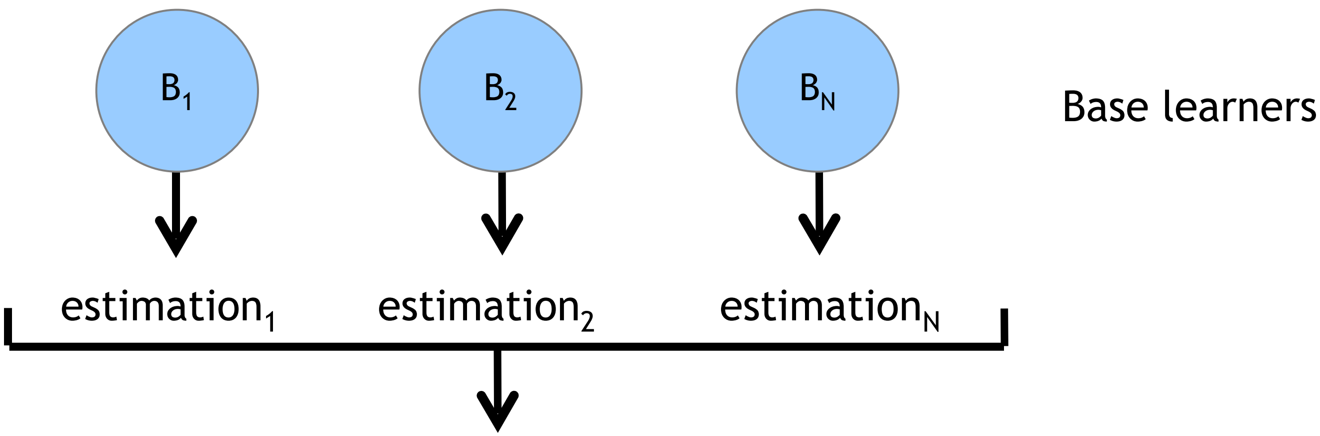
Ensembles and Wisdom of the Crowd

Committees of artificially generated experts with different views on how to solve a problem.

[Video -- BBC The Code -- Wisdom of the Crowd]
https://youtu.be/iOucwX7Z1HU

Ensembles

Sets of learning machines grouped together with the aim of improving predictive performance.

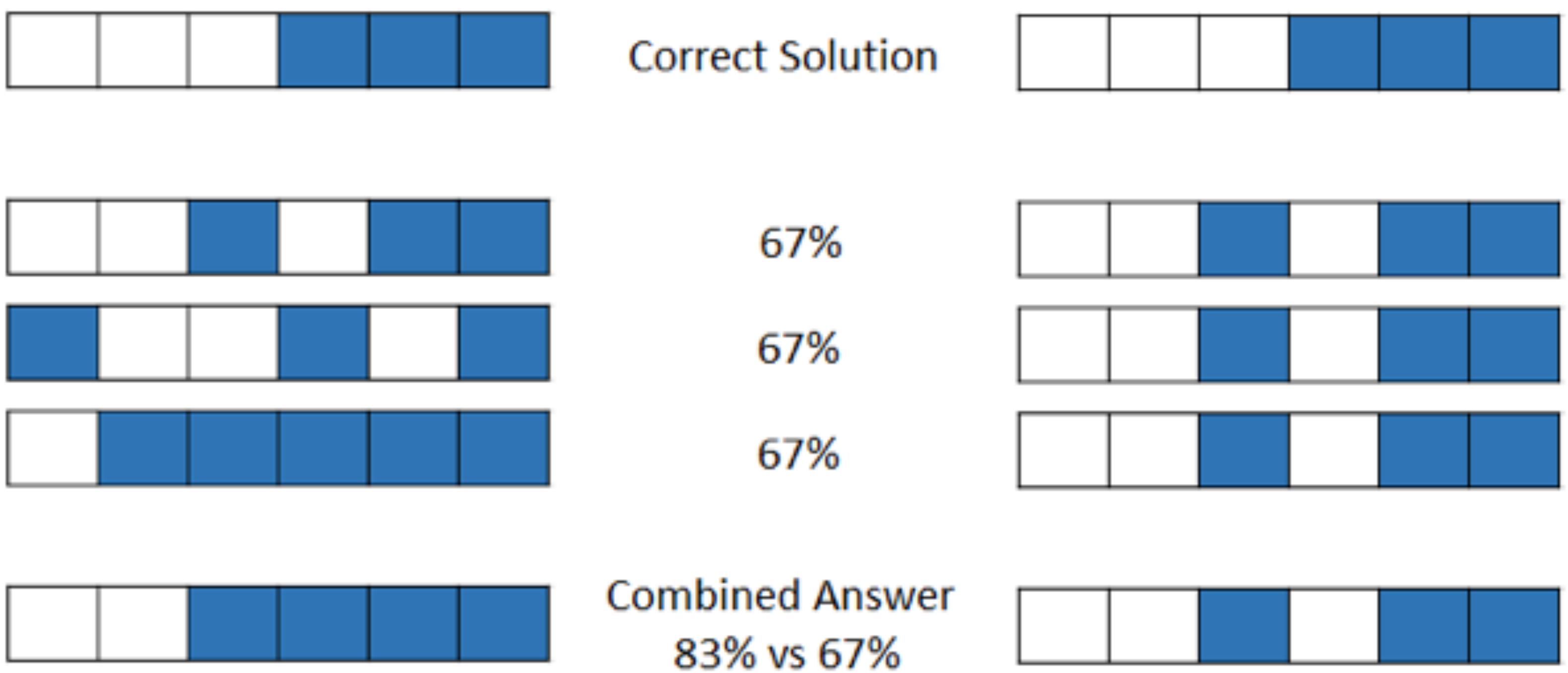


E.g.: ensemble estimation = $\Sigma\, w_i\, \text{estimation}_i$

T. Dietterich. Ensemble Methods in Machine Learning. Proceedings of the First International Workshop in Multiple Classifier Systems. 2000.

Ensemble Diversity

One of the keys: diversity, i.e., different base learners make different mistakes on the same instances.



Ensemble Versatility

Diversity can be used to address different issues when estimating software data.

- Models of the same environment
- Models with different goals
- Models of different environments

Ensemble Versatility

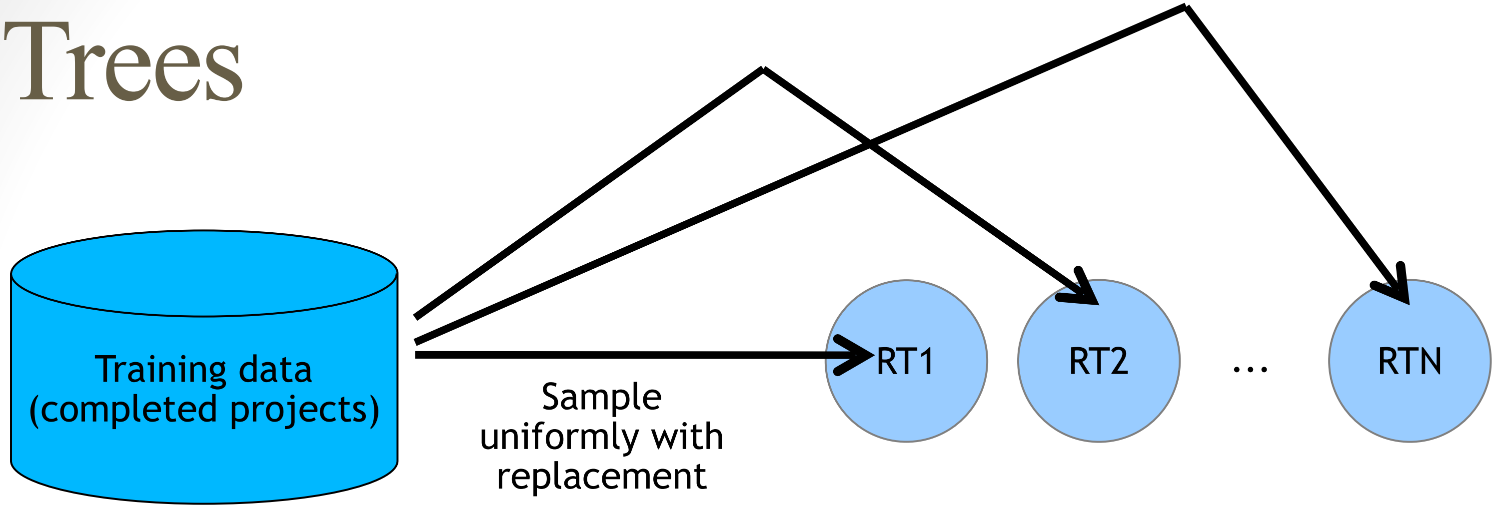
Diversity can be used to increase stability across data sets.

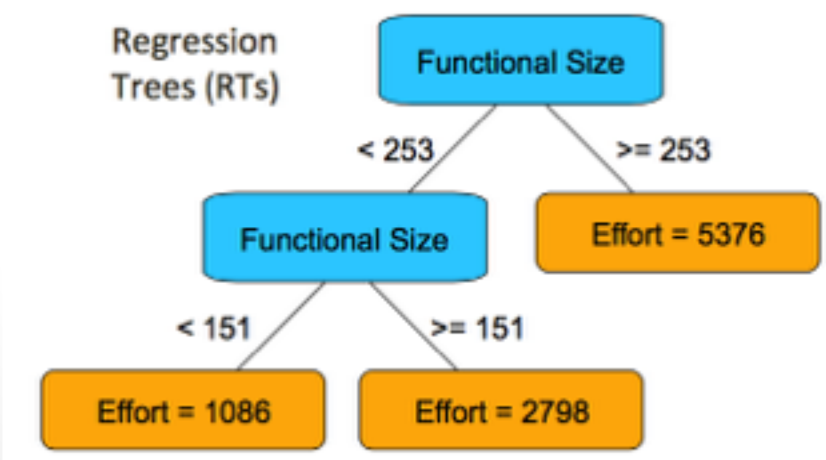
- Models of the same environment
- Models with different goals
- Models of different environments

Conclusion Instability

- Different predictive models perform differently on different data sets.
- Predictive models (e.g., RTs and MLPs) can be unstable when trained on different samples.
- Ensembles can help increasing conclusion stability across data sets.
  - Facilitates model choice.

Bagging Ensembles of Regression Trees





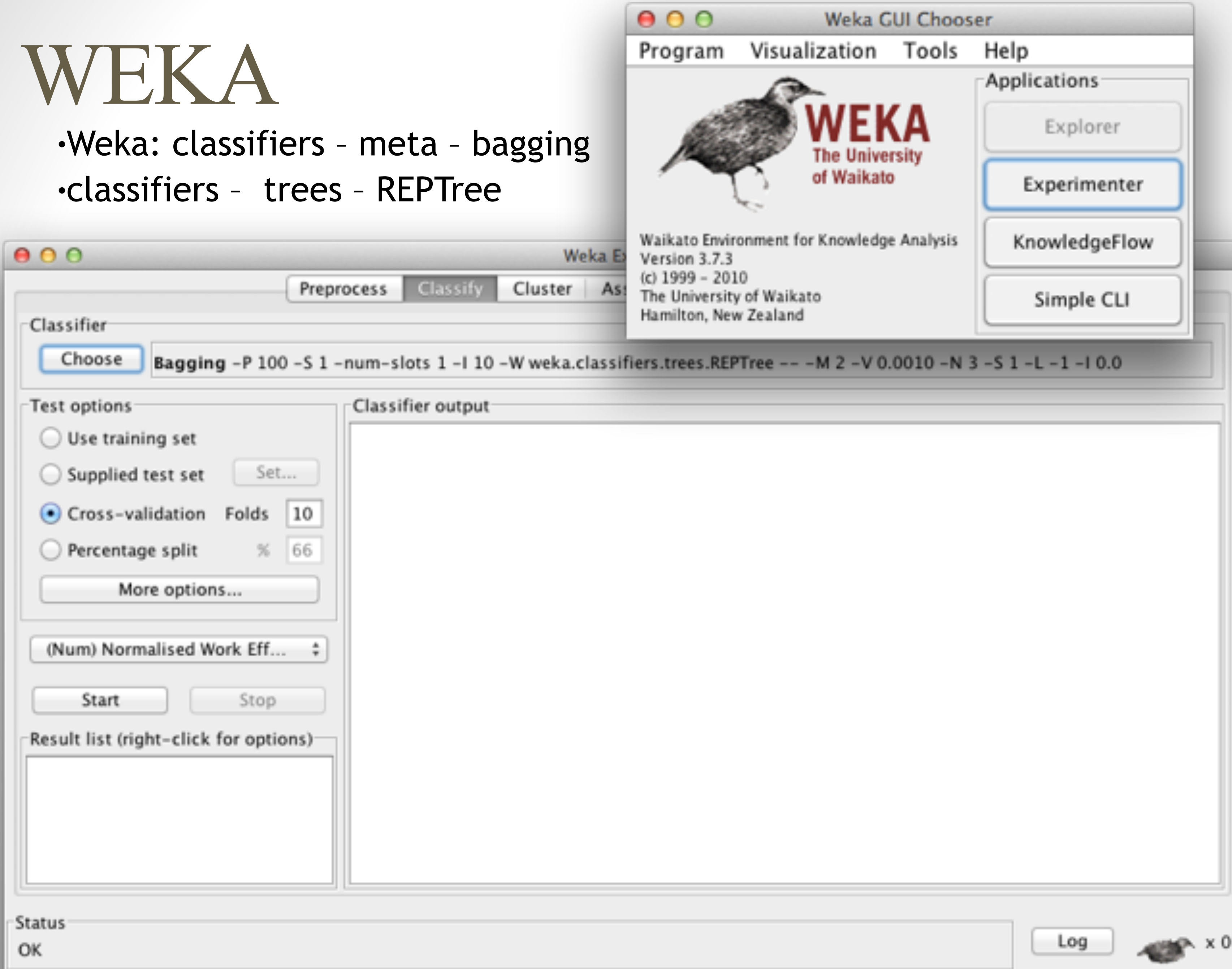
Regression Trees (RTs):

- **Local** methods.
- Divide projects according to attribute value.
- Most impactful attributes are in higher levels.
- Attributes with insignificant impact are not used.
- E.g., REPTrees.

L. Breiman. Bagging Predictors. Machine Learning 24(2):123-140, 1996.

WEKA

- Weka: classifiers – meta – bagging
- classifiers – trees – REPTree



Increasing Performance Rank Stability Across Data Sets

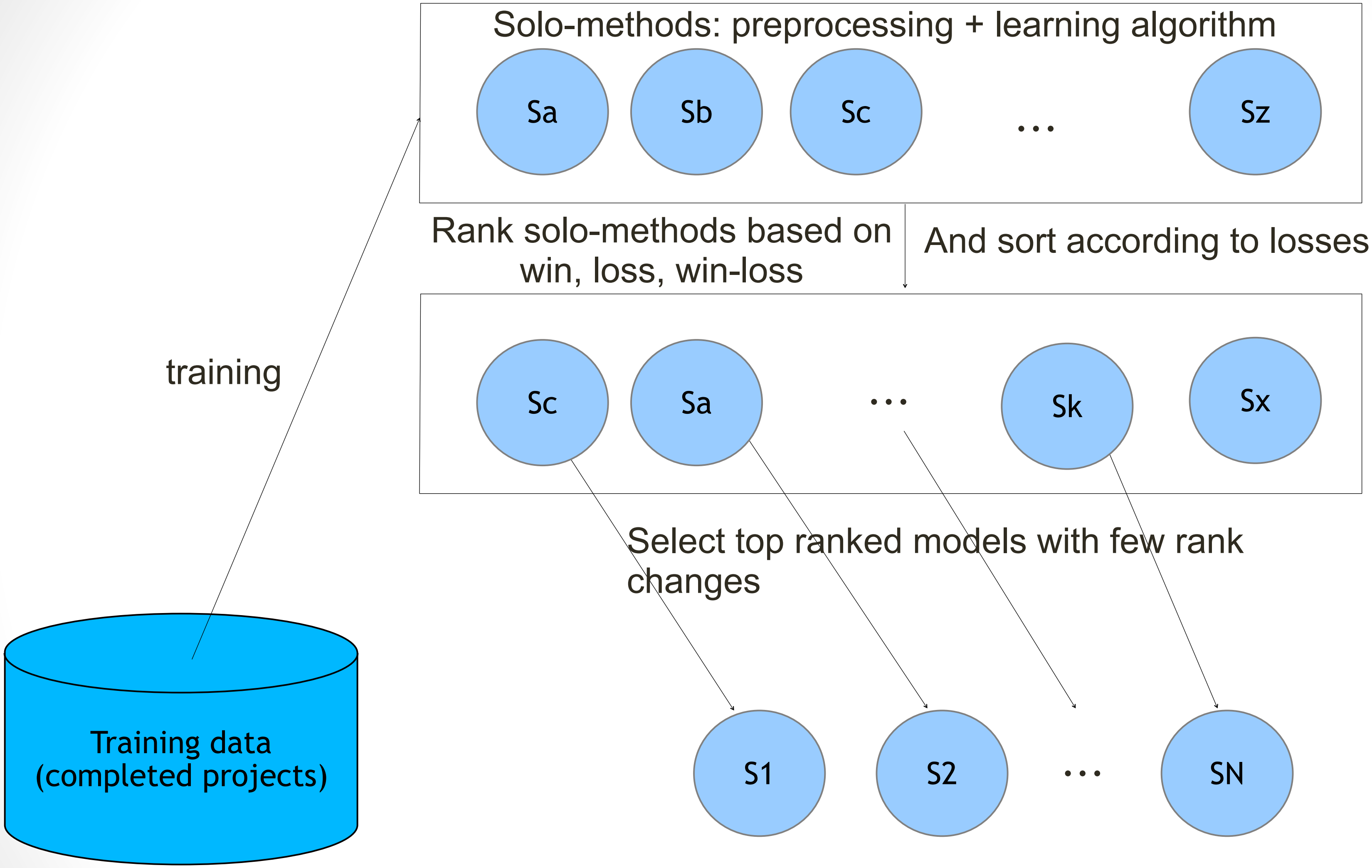
- Study with 13 data sets from PROMISE and ISBSG repositories.
- Bag+RTs:
  - Obtained the highest rank across data set in terms of Mean Absolute Error (MAE).
  - Rarely performed considerably worse (>0.1SA, SA = 1 – MAE / MAE_{rguess}) than the best approach:

Approaches	Number of Times
RT, Bag+RT	1
Bag+MLP, Bag+RBF, MLP	3
Rand+MLP	5
RBF	8
NCL+MLP	10

Approaches	Number of Times
RT	2
EM	4
K-NN, SC	5
K-Means	6

L. Minku, X. Yao. Ensembles and Locality: Insight on Improving Software Effort Estimation. Information and Software Technology 55(8):1512-1528, 2013.

Comba

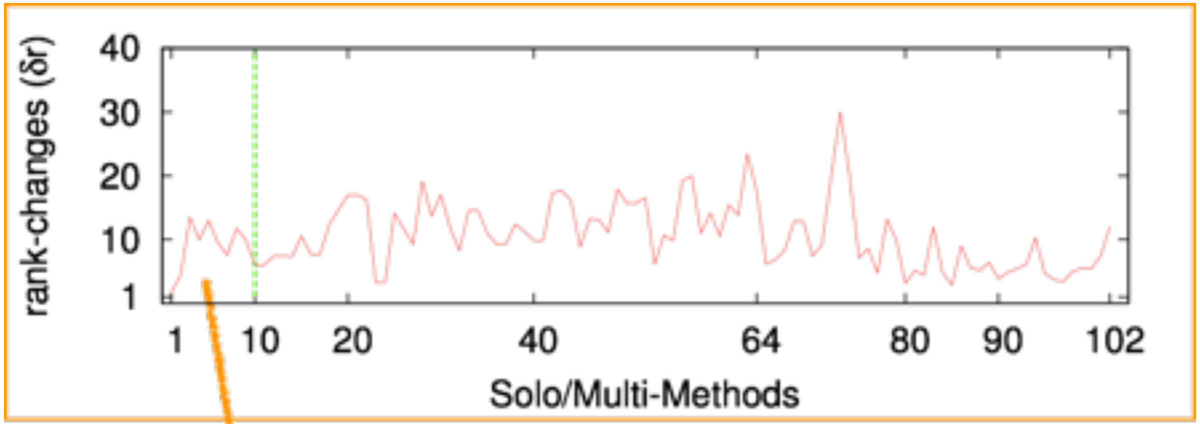


Kocaguneli, E., Menzies, T. and Keung, J. On the Value of Ensemble Effort Estimation. IEEE Transactions on Software Engineering, 8(6):1403 – 1416, 2012.

Increasing Rank Stability Across Data Sets

Combine top 2,4,8,13 solo-methods via mean, median and IRWM

Re-rank solo and multi-methods together according to #losses



Best 10 methods are all multi-methods..

The first ranked multi-method had very low rank-changes.

Ensemble Versatility

Diversity can be used to create models that perform well on different goals.

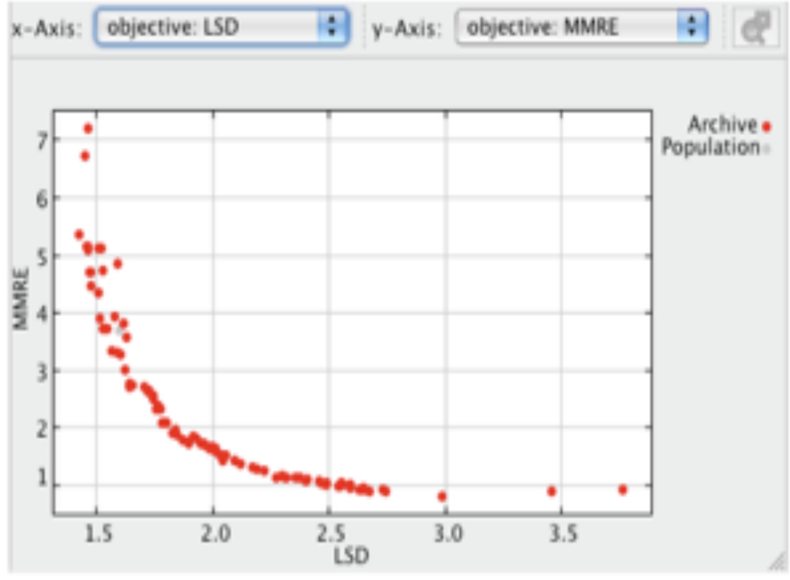
Models of the same environment

Models with different goals

Models of different environments

Multi-Objective Ensemble

- We may be interested in creating models that do well in terms of different objectives.
  - E.g., in software effort estimation, different performance measures capture different quality features.



- There is no agreed single measure.
- A model doing well for a certain measure may not do so well for another.

Multi-Objective Ensembles

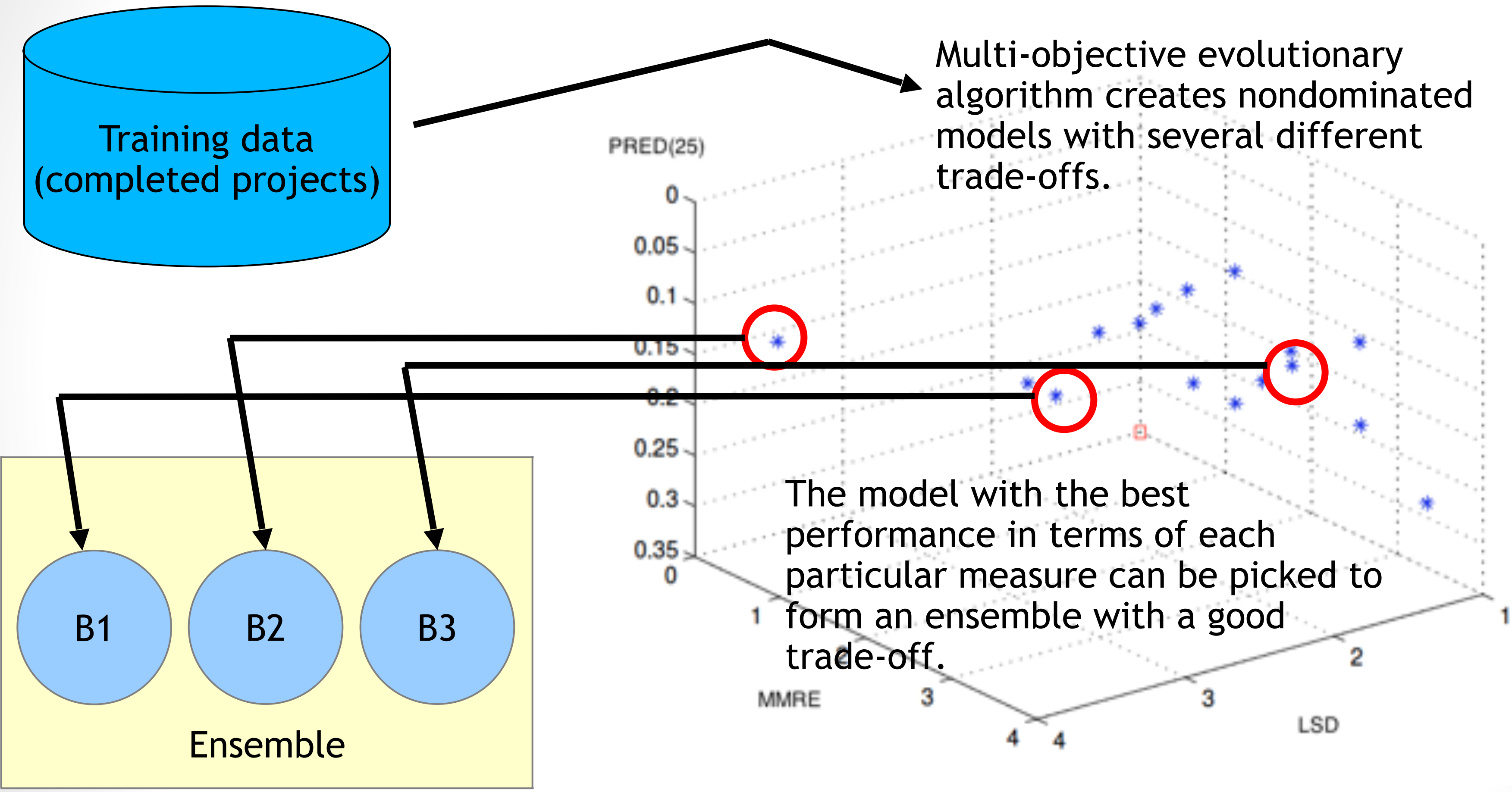
- We can view such problems (e.g., software effort estimation) as a multi-objective learning problems.

- A multi-objective approach (e.g. Multi-Objective Evolutionary Algorithm (MOEA)) can be used to:
  - Create models that do well for different objectives, in particular for larger data sets (>=60).
  - Better understand the relationship among objectives.

[Video - https://youtu.be/sEEiGM9em8s]

L. Minku, X. Yao. Software Effort Estimation as a Multi-objective Learning Problem. ACM Transactions on Software Engineering and Methodology, 22(4):35, 2013.

Multi-Objective Ensembles



L. Minku, X. Yao. Software Effort Estimation as a Multi-objective Learning Problem. ACM Transactions on Software Engineering and Methodology, 22(4):35, 2013.

Improving Performance on Different Measures

- Sample result: Pareto ensemble of MLPs (ISBSG):

	LSD	MMRE	PRED(25)
Pareto Ensemble	1.10 +- 0.95	0.73 +- 0.29	0.26 +- 0.13
Backpropagation MLP	2.68 +- 2.71	2.03 +- 2.58	0.17 +- 0.11

- Important:
  - Using performance measures that behave differently from each other (low correlation) provide better results than using performance measures that are highly correlated.
    - More diversity.
  - This can even improve results in terms of other measures not used for training.

L. Minku, X. Yao. An Analysis of Multi-objective Evolutionary Algorithms for Training Ensemble Models Based on Different Performance Measures in Software Effort Estimation. PROMISE, 10p, 2013.

Ensemble Versatility

Diversity can be used to deal with changes and transfer knowledge.

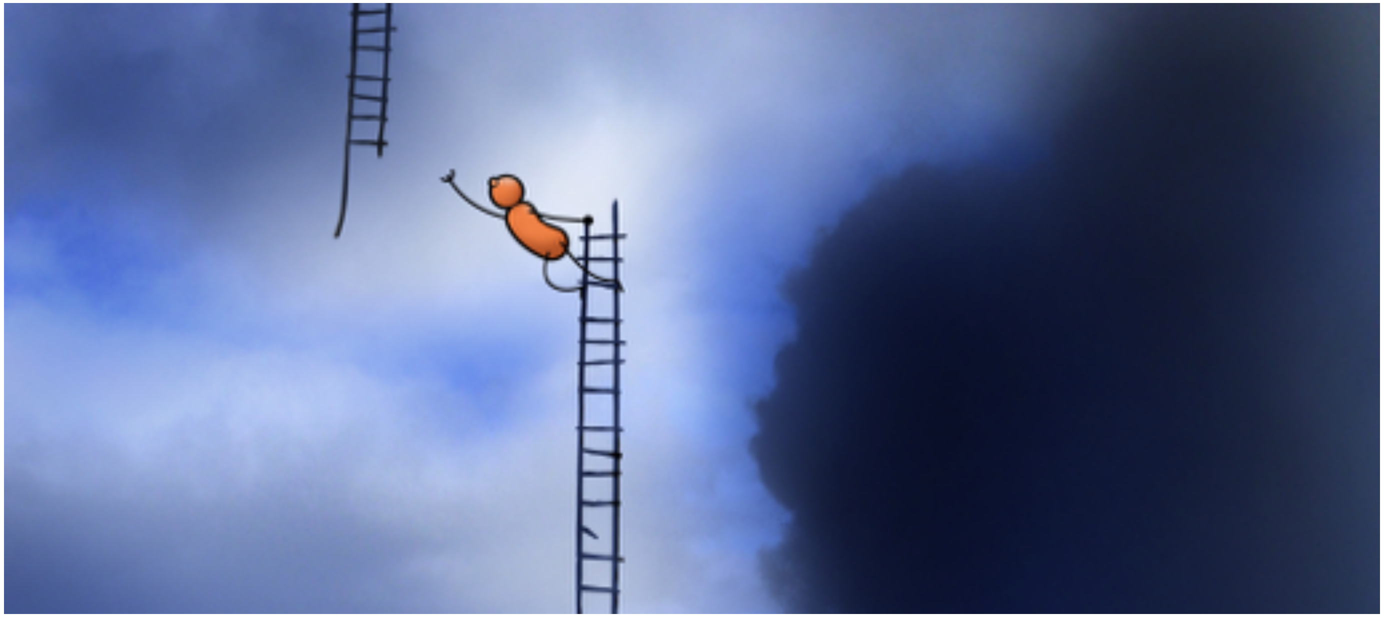
Models of the same environment

Models with different goals

Models of different environments

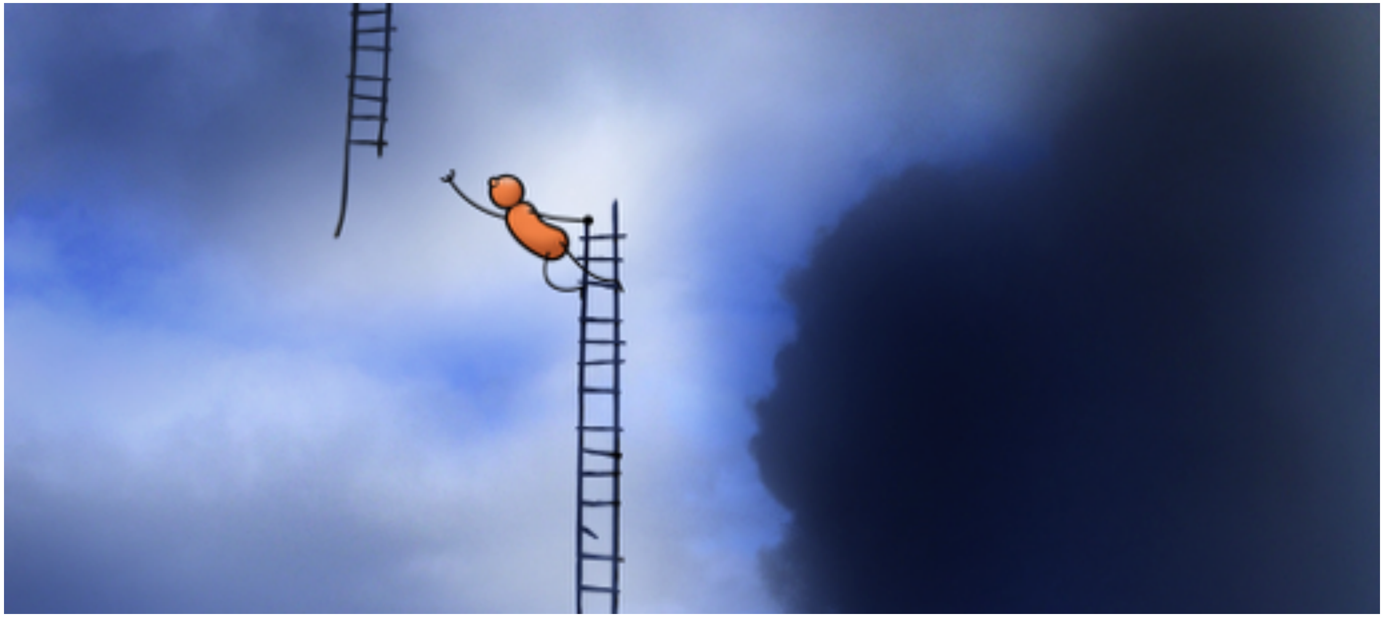
Companies' Changing Environments

Companies are not static entities – they can change with time *(concept drift)*.



Companies' Changing Environments

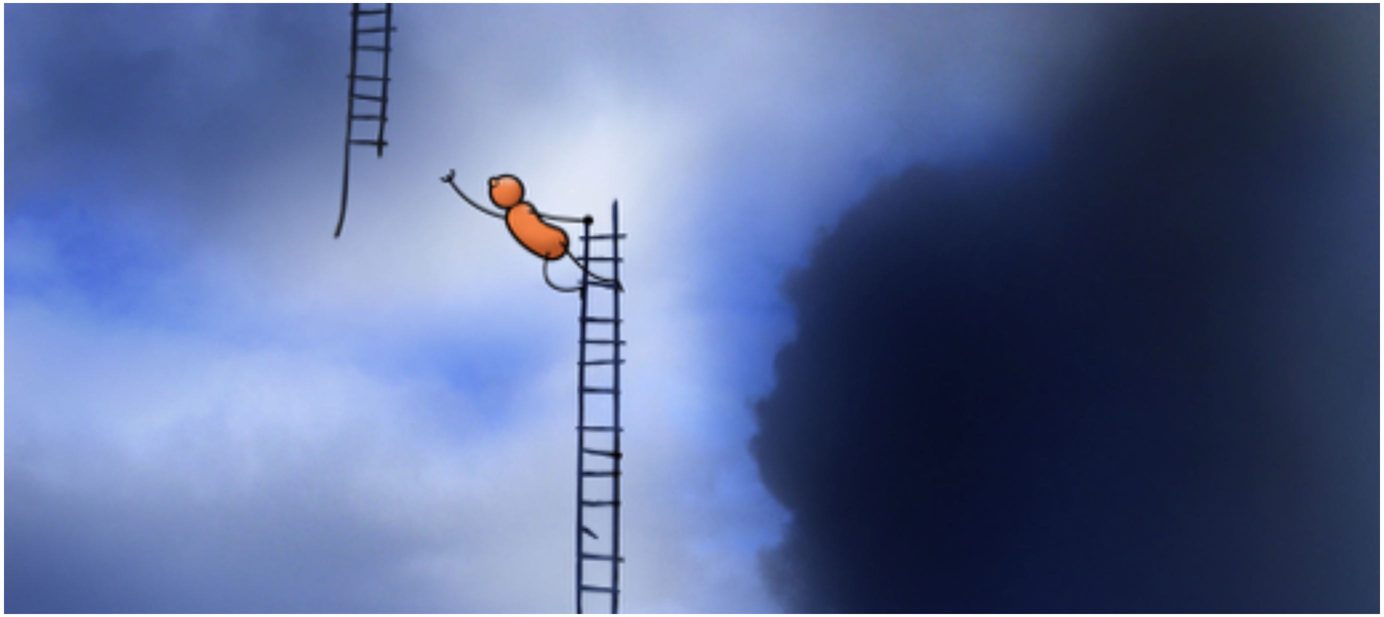
Companies are not static entities – they can change with time *(concept drift)*.



E.g., change in management strategy, development of new types of products, key employees leaving the company, etc.

Companies' Changing Environments

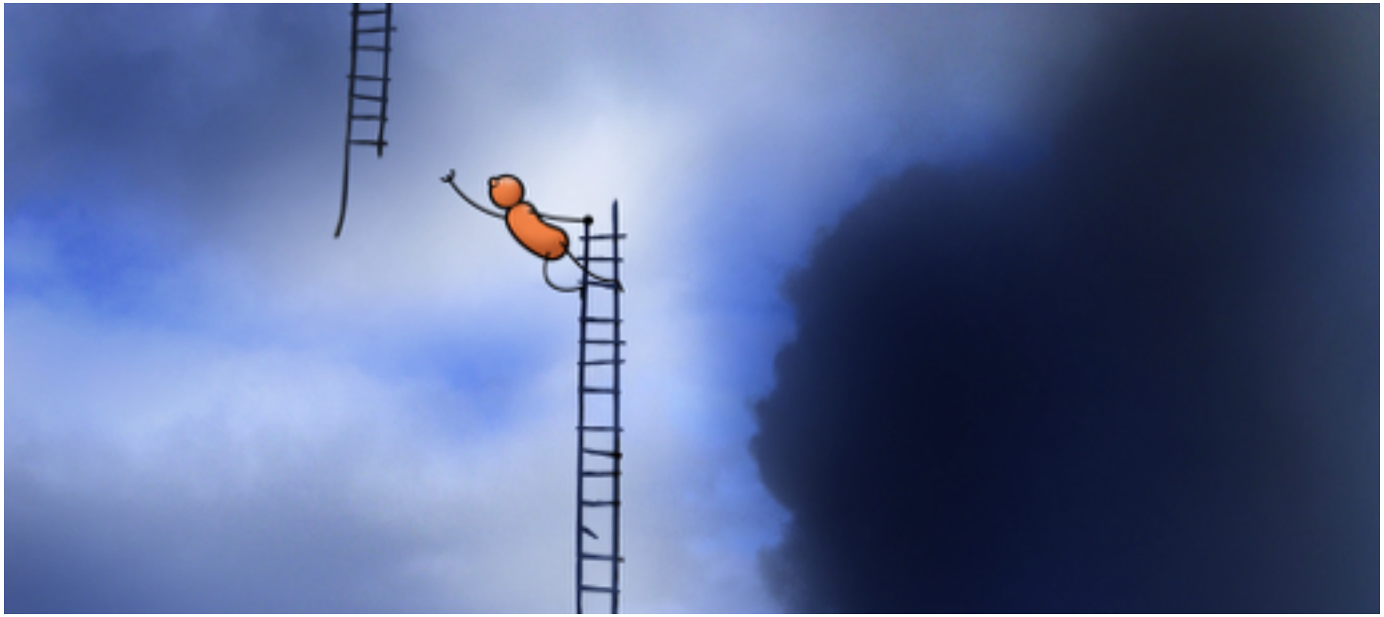
Companies are not static entities – they can change with time *(concept drift)*.



Changes may affect how well a given model describes the current situation of a company.

Companies' Changing Environments

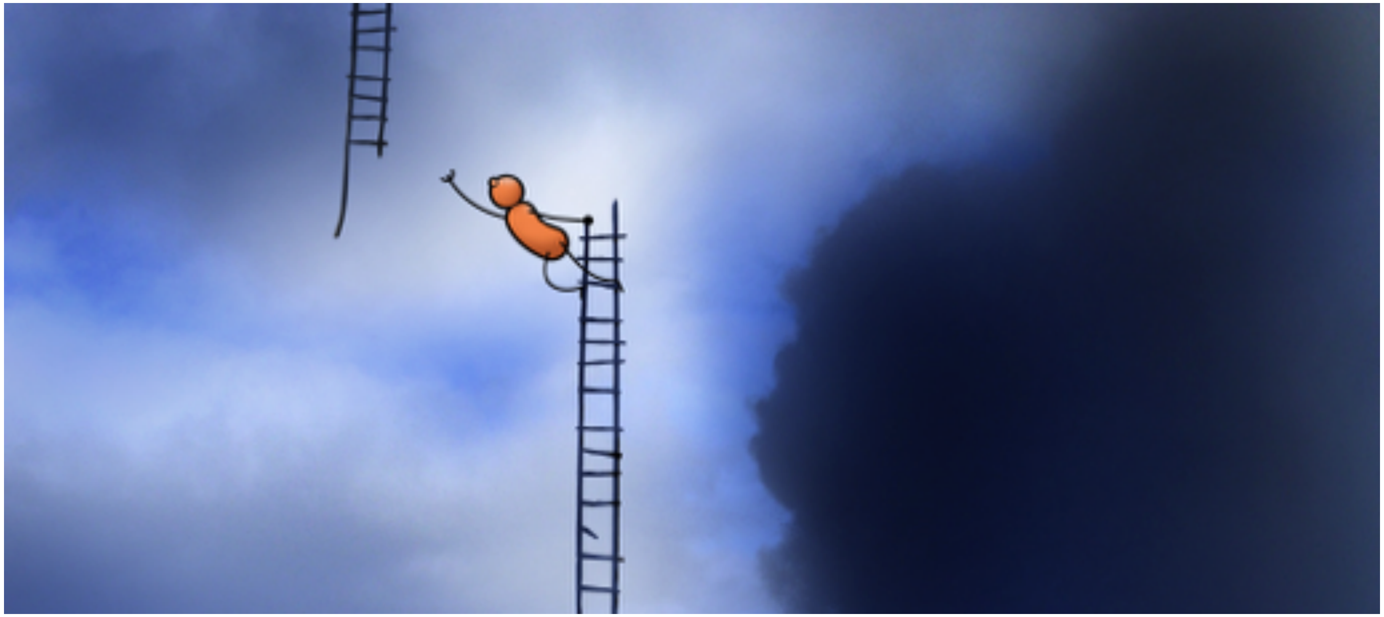
Companies are not static entities – they can change with time *(concept drift)*.



Software data analytics should consider temporal information!

Companies' Changing Environments

Companies are not static entities – they can change with time *(concept drift)*.

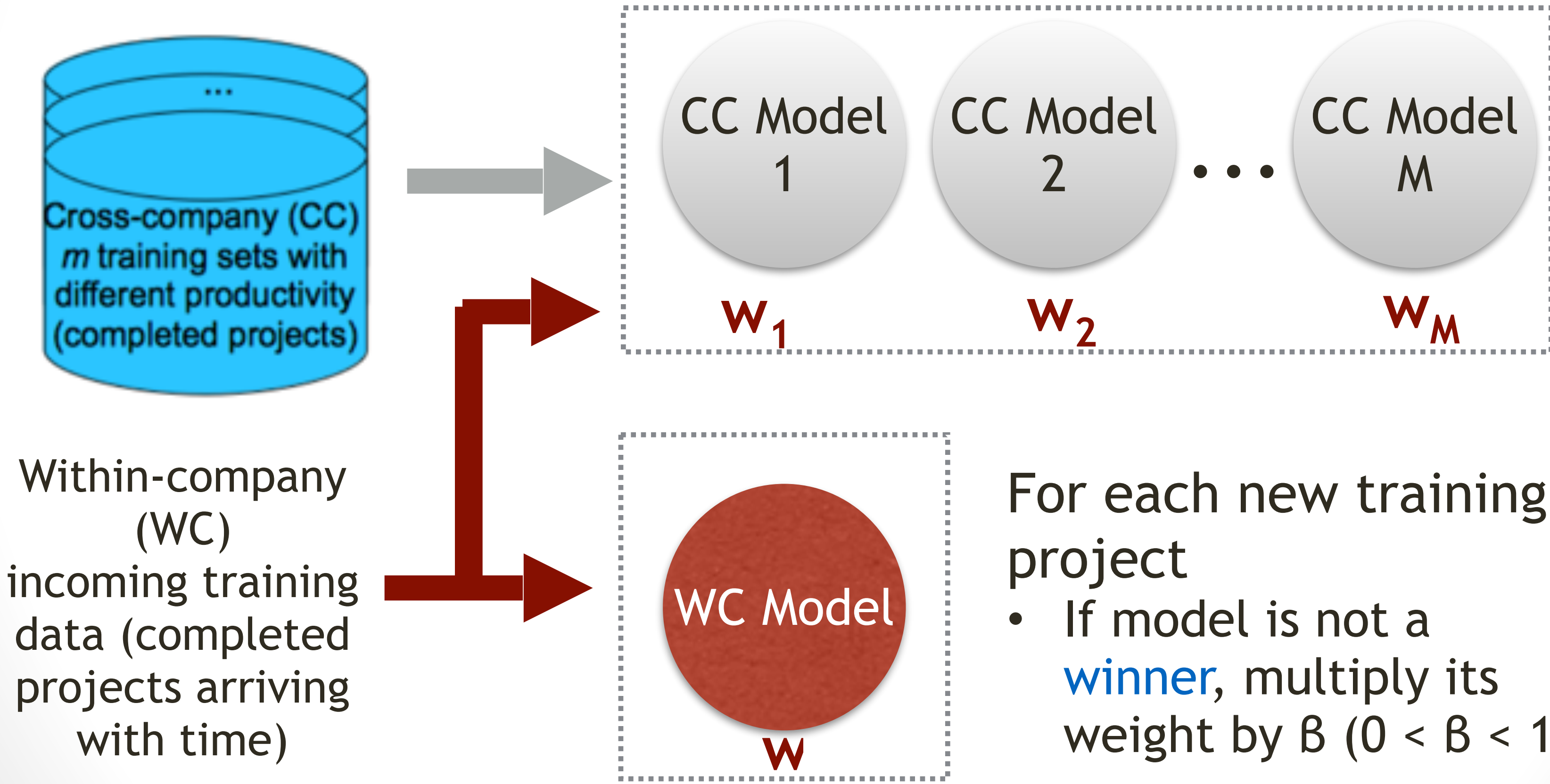


How to know when a model reflects well the current situation of a company?

How to update models throughout time?

Dynamic Cross-Company Learning (DCL)

DCL learns a weight to reflect the suitability of CC models.



For each new training project

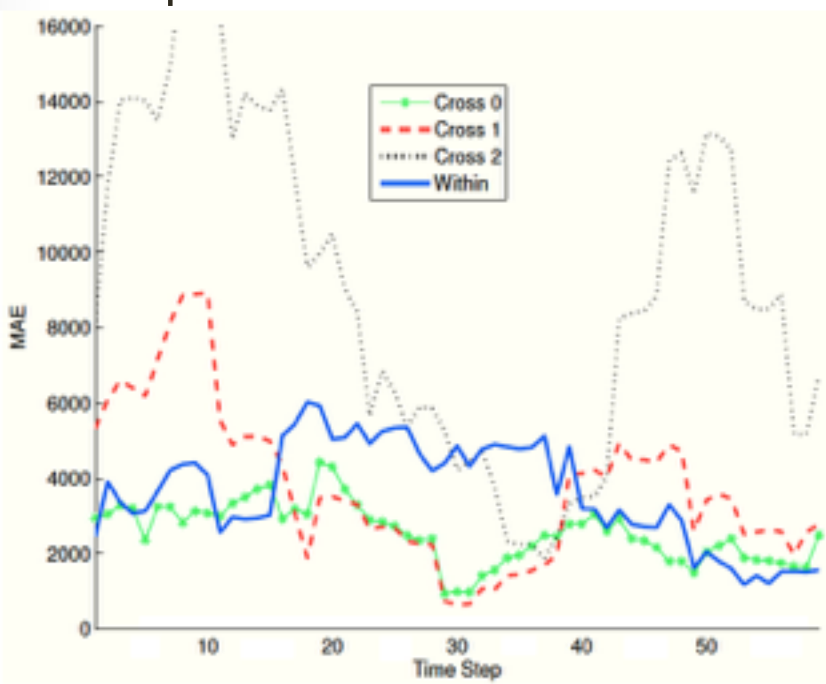
- If model is not a winner, multiply its weight by β (0 < β < 1)

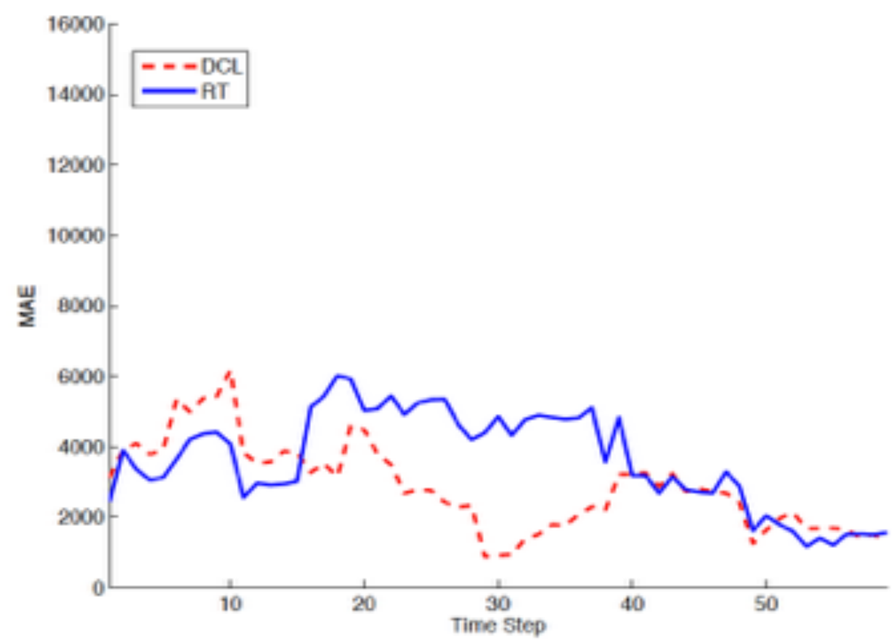
L. Minku, X. Yao. Can Cross-company Data Improve Performance in Software Effort Estimation? PROMISE, p. 69-78, 2012.

Improving Performance Throughout Time

- DCL can identify which model best represents our current situation.
- DCL adapts to changes by using CC models.
- DCL manages to use CC models to improve performance over WC models.

Sample Result





Predicting effort for a single company from ISBSG based on its projects and other companies' projects.

Why DCL? When?

Why?

- DCL is able to identify which model (CC or WC) best represents the current situation of a company.
  - It can be used for transfer learning.
  - It can deal with changes.
  - It can improve performance over WC models when CC models are useful.

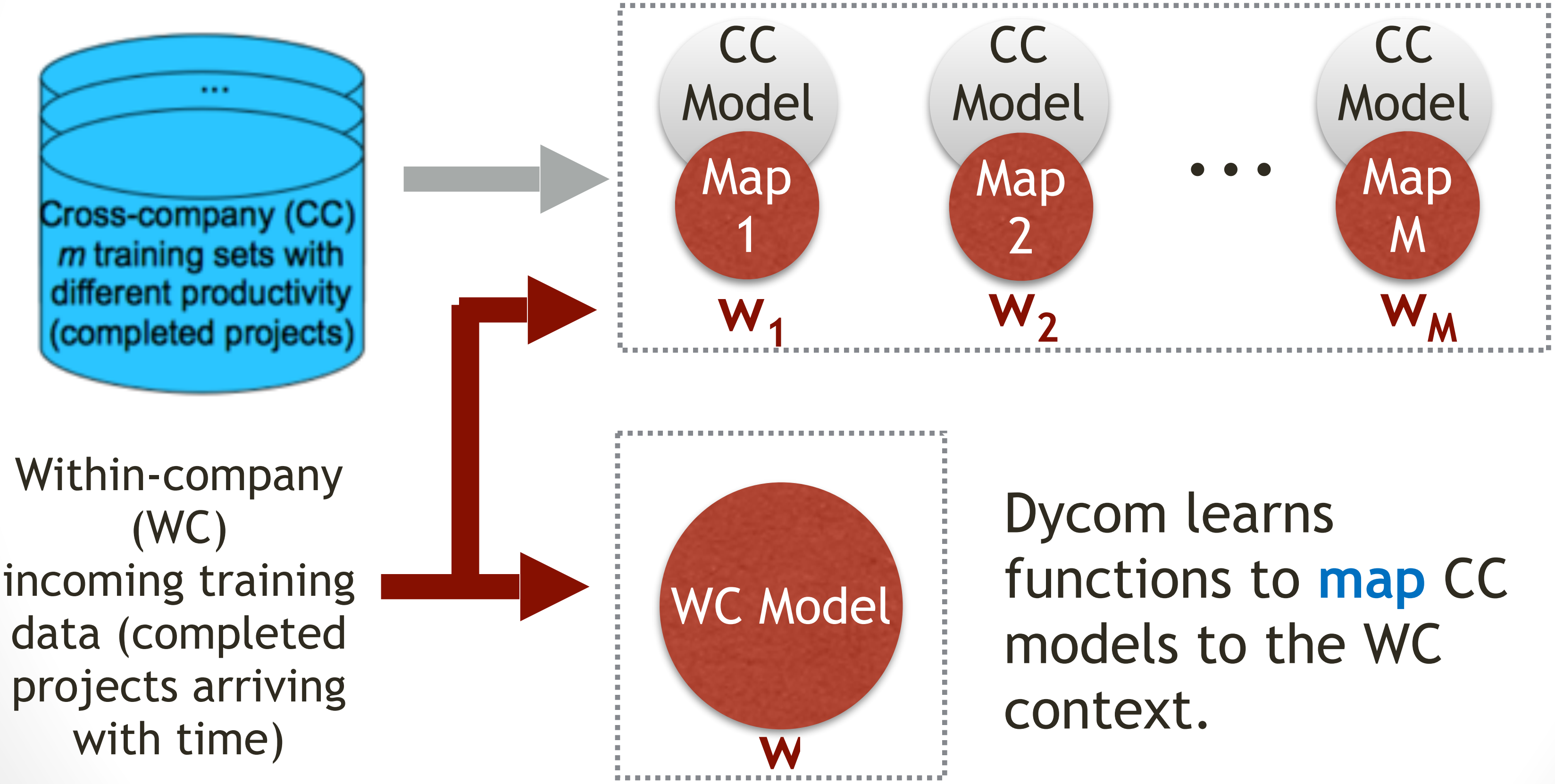
When?

- When one wishes to use CC data to improve predictive performance.
- When environments are likely to suffer changes.

If none of the CC models is useful, DCL will not be able to benefit from them.

Dynamic Cross-Company Mapped Model Learning (Dycom)

How to use CC models even when they are not directly helpful?



Dycom learns functions to **map** CC models to the WC context.

L. Minku, X. Yao. How to Make Best Use of Cross-Company Data in Software Effort Estimation? ICSE, p. 446-456, 2014.

Learning Mapping Function

$$(\hat{f}_B(\mathbf{x}), y) \xrightarrow{\text{train}} \hat{f}_A(\mathbf{x}) = \hat{g}_{BiA}(\hat{f}_{Bi}(\mathbf{x})) = \hat{f}_{Bi}(\mathbf{x}) \cdot b_i$$

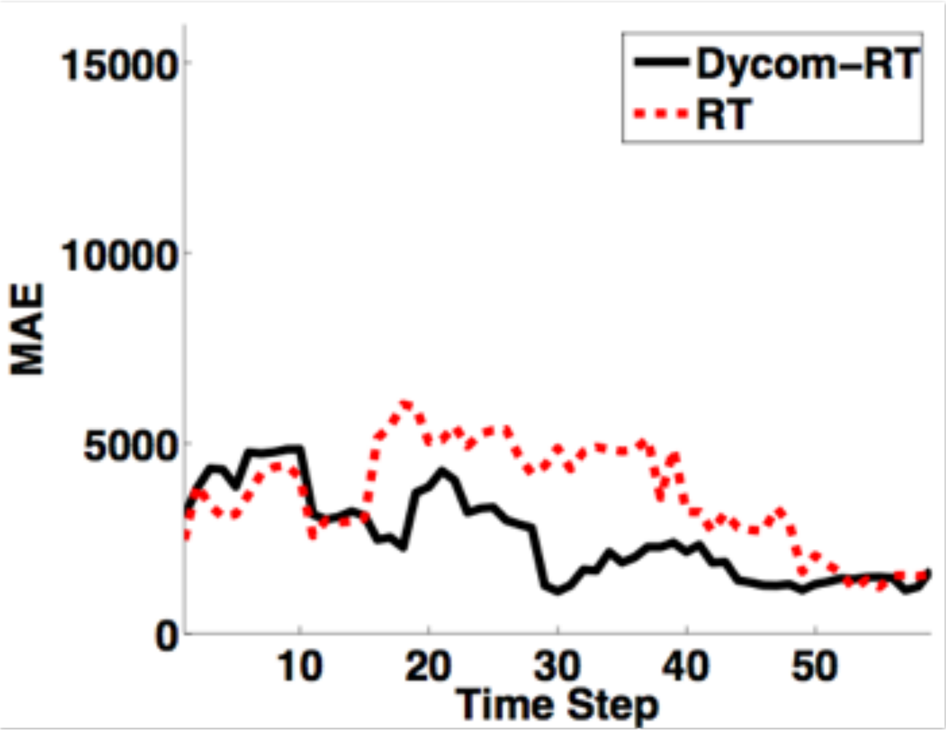
$$b_i = \begin{cases} 1, & \text{if no mapping training example has been received yet;} \\ \dfrac{y}{\hat{f}_{Bi}(\mathbf{x})}, & \text{if } (\hat{f}_{Bi}(\mathbf{x}), y) \text{ is the first mapping training example;} \\ lr \cdot \dfrac{y}{\hat{f}_{Bi}(\mathbf{x})} + (1 - lr) \cdot b_i, & \text{otherwise.} \end{cases}$$

where lr is a smoothing factor that allows tuning the emphasis on more recent examples.

L. Minku, X. Yao. How to Make Best Use of Cross-Company Data in Software Effort Estimation? ICSE, p. 446-456, 2014.

Reducing the Number of Required WC Training Examples

Sample Result



Dycom can achieve similar / better performance while using only 10% of WC data.

Why Dycom? When?

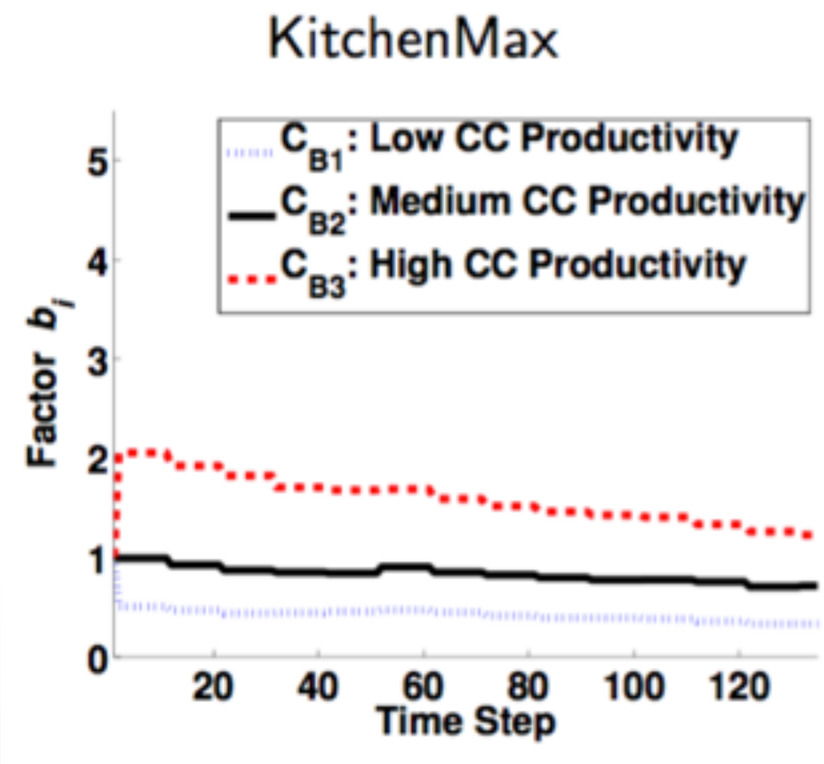
Why?

- Dycom is able to map models representing different contexts to the context we are interested in.
  - It can be used for transfer learning.
  - It can deal with changes.
  - It can reduce the number of required WC training examples.

When?

- Dycom is particularly useful when collection of WC training examples is expensive.
- When used for software effort estimation, Dycom can also provide insights into the productivity of a company over time.

Dycom Insights on Productivity

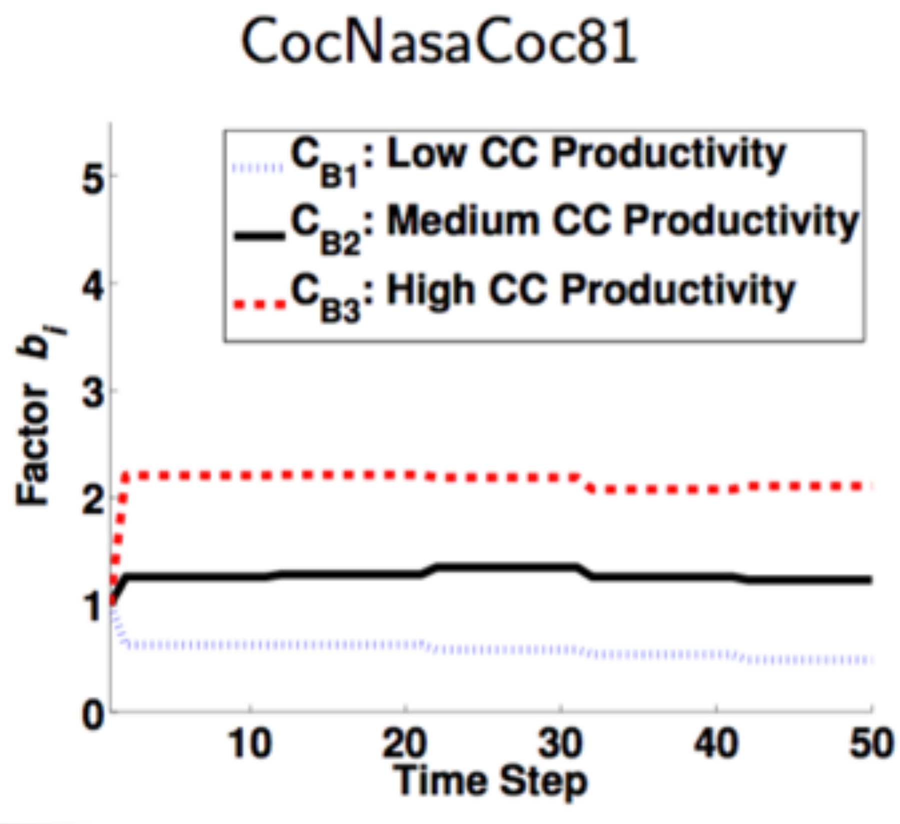


- Relationship between effort of different companies for the same projects.

$$\hat{f}_A(\mathbf{x}) = \hat{f}_{Bi}(\mathbf{x}) \cdot b_i$$

- Initially, our company needs initially 2x effort than company red.
- Later, it needs only 1.2x effort.

Dycom Insights on Productivity



$$\hat{f}_A(\mathbf{x}) = \hat{f}_{Bi}(\mathbf{x}) \cdot b_i$$

- Our company needs 2x effort than company red.
- How to improve our company?

Analysing Project Data

Number of projects with each feature value for the 20 CC projects from the medium productivity CC section and the first 20 WC projects:

Feature / Value	Lang. exp		Virtual mach. exp	
	CC	WC	CC	WC
Very low	1	0	1	0
Low	1	0	4	4
Nominal	8	8	8	16
High	10	12	7	0
Very high	0	0	0	0
Extremely high	0	0	0	0

Both the company and the medium CC section frequently use employees with high programming language experience.

Analysing Project Data

Number of projects with each feature value for the 20 CC projects from the medium productivity CC section and the first 20 WC projects:

Feature / Value	Lang. exp		Virtual mach. exp	
	CC	WC	CC	WC
Very low	1	0	1	0
Low	1	0	4	4
Nominal	8	8	8	16
High	10	12	7	0
Very high	0	0	0	0
Extremely high	0	0	0	0

Medium CC section uses more employees with high virtual machine experience. So, this is more likely to be a problem for the company. Sensitivity analysis and project manager knowledge could help to confirm that.

Ensemble Versatility

Diversity can be used to address different issues when estimating software data.

Increase stability across data sets.

Increase performance on different measures.

Models of the same environment

Models with different goals

Models of different environments

Deal with changes and transfer knowledge.

1. Introduction
2. Sharing data
3. Privacy and sharing
4. Sharing models
5. Summary

6a. The past
6b. The present
6c. The future

The past

- Models for individual data owners.
- Conclusion instability.

The present

- Reducing problems caused by conclusion instability.
- Finding local lessons from global data.
  - Accomplished for individual data owners as well as data owners who want to share data collaboratively.
  - Results are promising.

The future

- Privacy
  - Next step : focus on end user privacy
    - when using software apps that need personal info to function.

- Model-based reasoning
  - Gaining more insights from models.
  - Considering temporal aspects of software data.
  - Taking goals into account in decision-support tools.

End of our tale