Machine Learning for Software Engineering and Software Engineering for Machine Learning — A Two Way Path?

Leandro L. Minku

University of Leicester, UK

Data Security: Network Intrusion Detection, Malware Prediction

Personal Security: Security Screening, Face Recognition…

Self-Driving Cars

TESLA'S AUTOPILOT FIRST DEADLY CAR CRASH

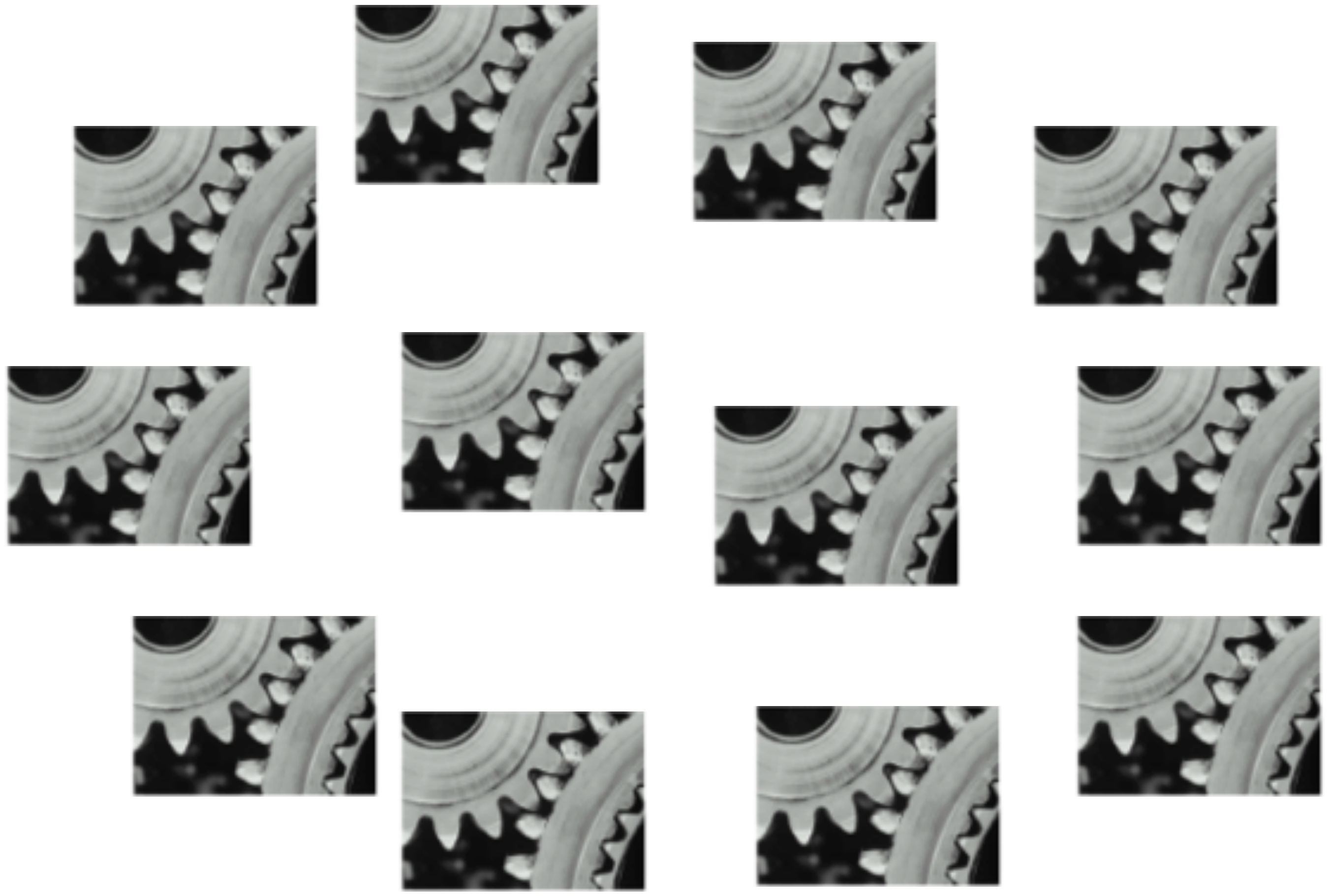
The automaker believes that the combination of a white trailer and brightly lit sky may have made the trailer difficult to see.

$\text{Machine Learning} \longrightarrow \text{Software Engineering}$

Software Defect Prediction

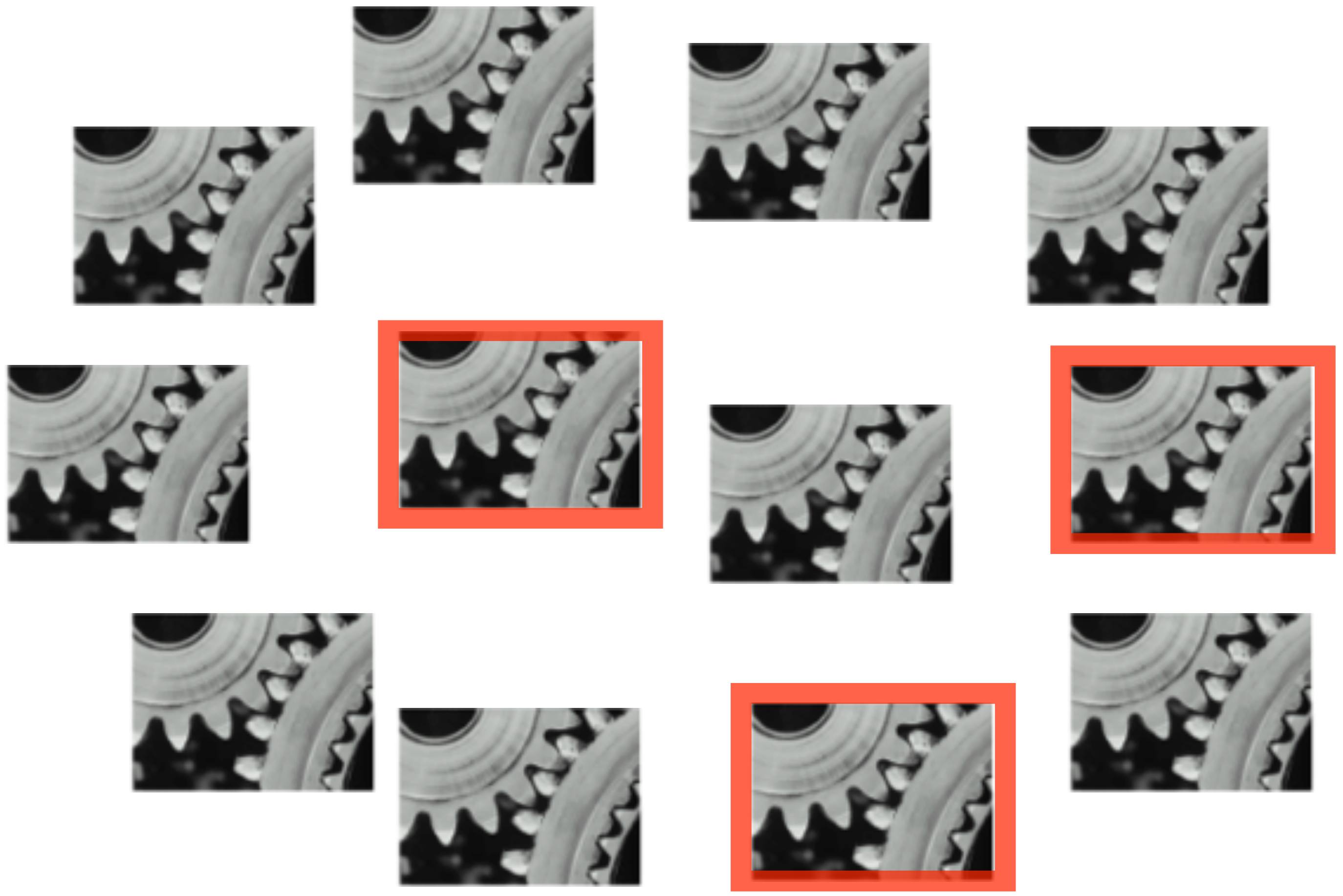
Software is composed of several components.

Testing all these components can be very expensive.



Software Defect Prediction

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



Software Defect Prediction as a Machine Learning Problem

Components of previous versions of the software

Module id	x1 = branch count	x2 = LOC	x3 = halstead	…	y = defective ?
1	18	1000	1	…	No
2	30	900	10	…	Yes
3	20	5000	3	…	Yes
…	…	…	…	…	…

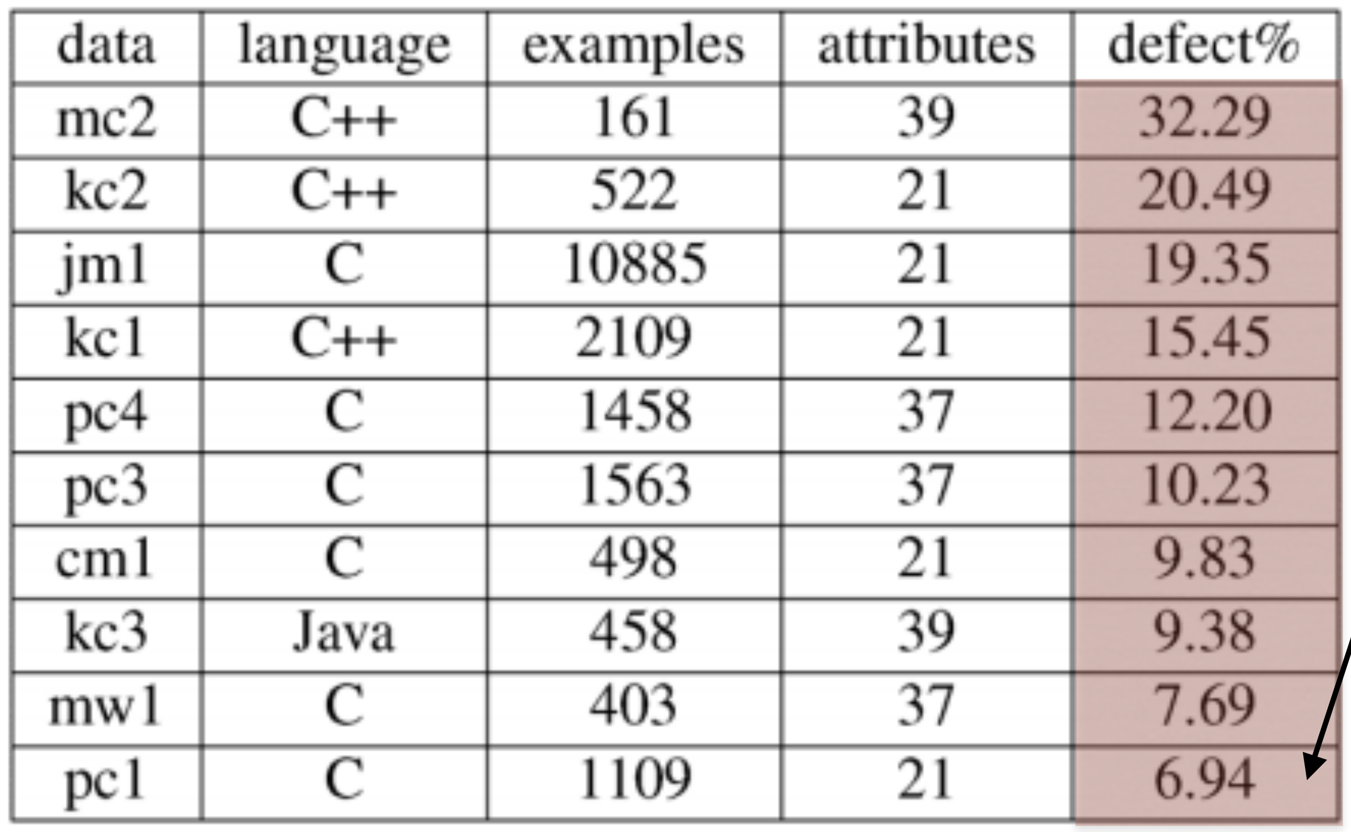
→ Machine Learning Algorithm →

New component **x** for new version of the software →

→ Yes/No

Class Imbalance

TABLE I
PROMISE DATA SETS, SORTED IN ORDER OF THE IMBALANCE RATE
(DEFECT%: THE PERCENTAGE OF DEFECTIVE MODULES)



data	language	examples	attributes	defect%
mc2	C++	161	39	32.29
kc2	C++	522	21	20.49
jm1	C	10885	21	19.35
kc1	C++	2109	21	15.45
pc4	C	1458	37	12.20
pc3	C	1563	37	10.23
cm1	C	498	21	9.83
kc3	Java	458	39	9.38
mw1	C	403	37	7.69
pc1	C	1109	21	6.94

E.g., an algorithm predicting all examples as non-defective would have 93.06% accuracy.

Problem: machine learning algorithms typically give the same importance to each training example.

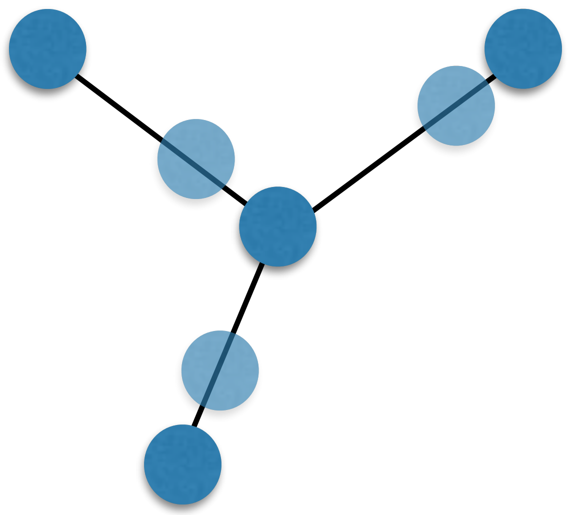
Source: S. Wang and X. Yao. Using Class Imbalance Learning for Software Defect Prediction, IEEE TR 62(2):434-443

Evaluating Classifiers for Class Imbalanced Data

- Accuracy is inadequate.
  - *(TP + TN) / (P + N)*
- Precision is inadequate.
  - *TP / (TP + FP)*
- Recall on each class separately is more adequate.
  - *TP / P* and *TN / N*.
- F-measure: not very adequate.
  - Harmonic mean of precision and recall.
- G-mean is more adequate.
  - $\sqrt{TP/P * TN/N}$
- ROC Curve is more adequate.
  - Recall on positive class *(TP / P)* vs False Alarms *(FP / N)*

Strategies to Deal With Class Imbalance

- Change the loss function of the machine learning algorithm to give the same level of attention to all classes.
- Use resampling techniques, e.g., SMOTE.

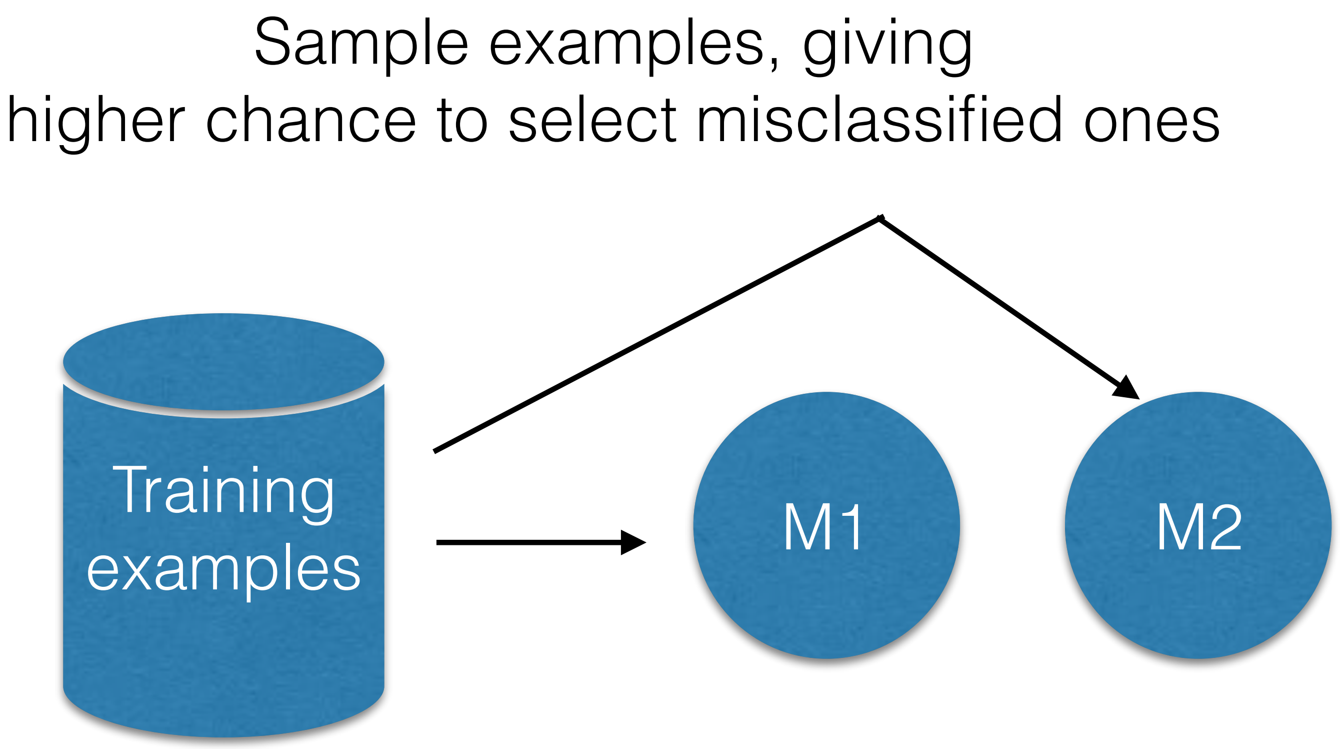


N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. J. Artif. Intell. Res., vol. 16, pp. 341–378, 2002.

Song, L., Minku, L. Yao, X. A Novel Automated Approach for Software Effort Estimation based on Data Augmentation, FSE 2018.

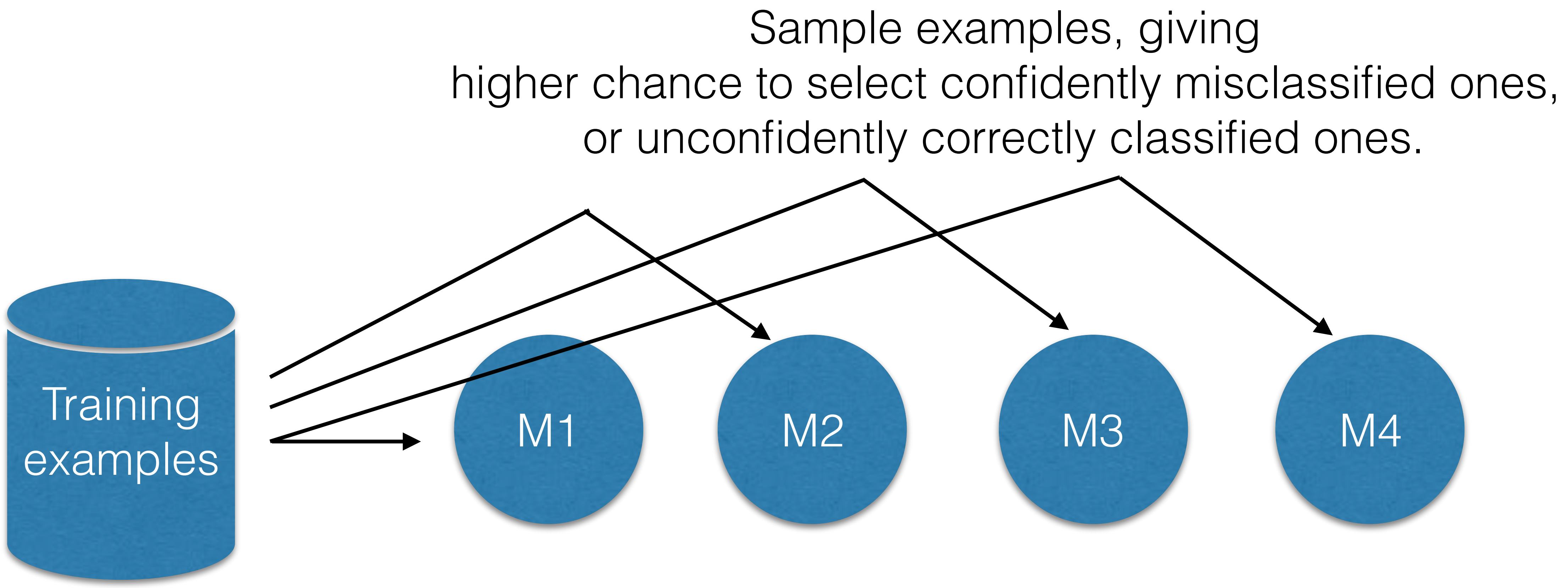
Strategies to Deal With Class Imbalance

- E.g., Adaboost.NC:
  - Ensemble method + strategy to deal with class imbalance.
  - Base models are created incrementally.


Sample examples, giving higher chance to select misclassified ones
Training examples
M1
M2

Strategies to Deal With Class Imbalance

- E.g., Adaboost.NC:
  - Ensemble method + strategy to deal with class imbalance.
  - Base models are created incrementally.


Sample examples, giving
higher chance to select confidently misclassified ones,
or unconfidently correctly classified ones.
Training examples
M1
M2
M3
M4

Sample Results (10 fold CV)

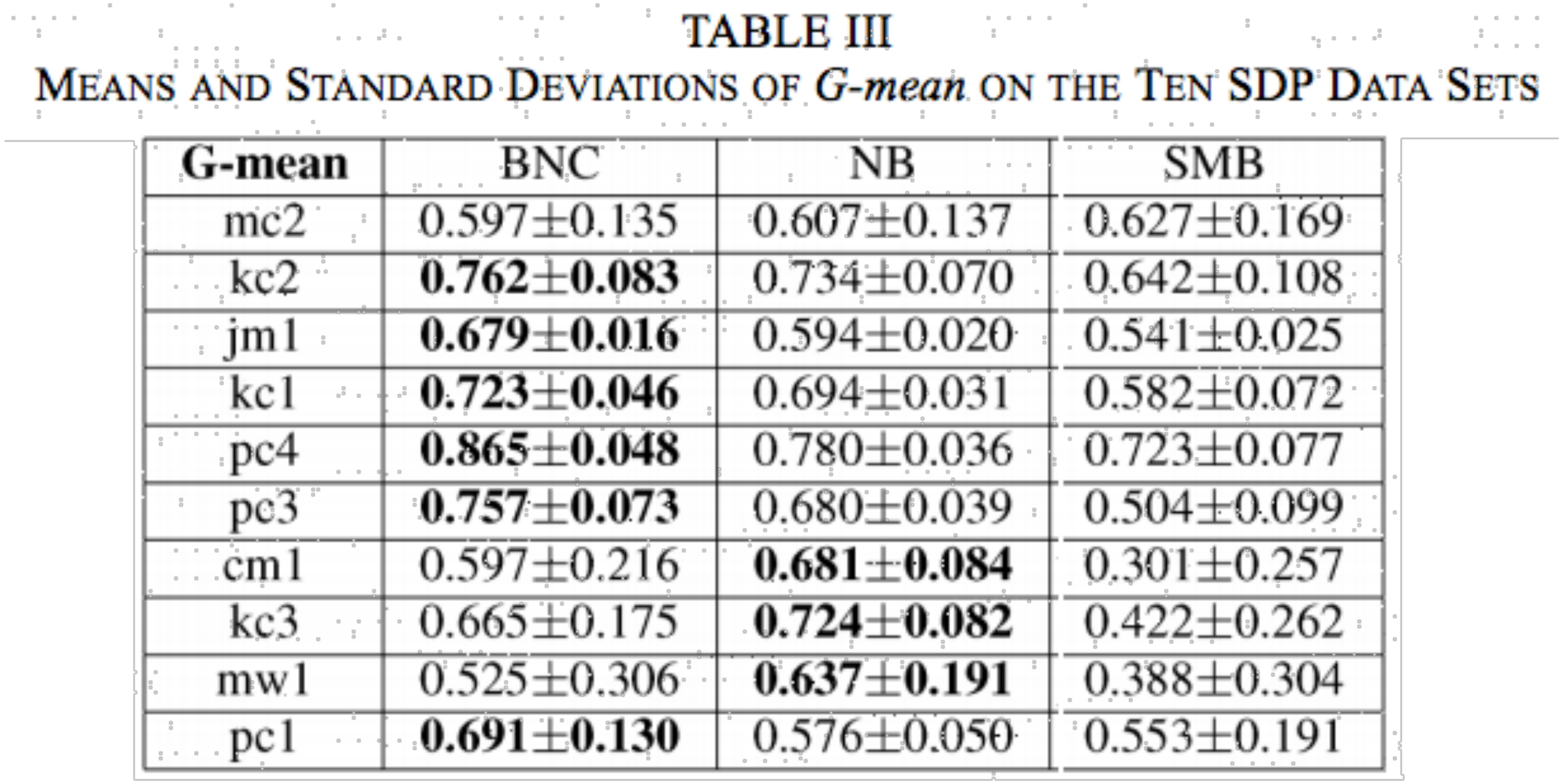


TABLE III
MEANS AND STANDARD DEVIATIONS OF *G-mean* ON THE TEN SDP DATA SETS

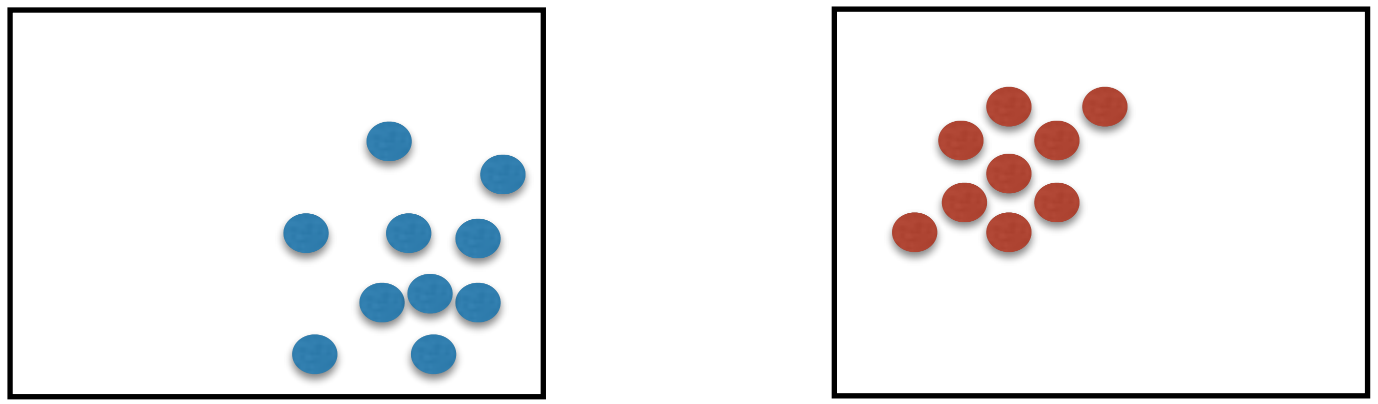
G-mean	BNC	NB	SMB
mc2	0.597±0.135	0.607±0.137	0.627±0.169
kc2	**0.762±0.083**	0.734±0.070	0.642±0.108
jm1	**0.679±0.016**	0.594±0.020	0.541±0.025
kc1	**0.723±0.046**	0.694±0.031	0.582±0.072
pc4	**0.865±0.048**	0.780±0.036	0.723±0.077
pc3	**0.757±0.073**	0.680±0.039	0.504±0.099
cm1	0.597±0.216	**0.681±0.084**	0.301±0.257
kc3	0.665±0.175	**0.724±0.082**	0.422±0.262
mw1	0.525±0.306	**0.637±0.191**	0.388±0.304
pc1	**0.691±0.130**	0.576±0.050	0.553±0.191

A better understanding of when a given approach works well and when it doesn't is necessary in most ML for SE studies.

Source: S. Wang and X. Yao. Using Class Imbalance Learning for Software Defect Prediction, IEEE TR 62(2):434-443

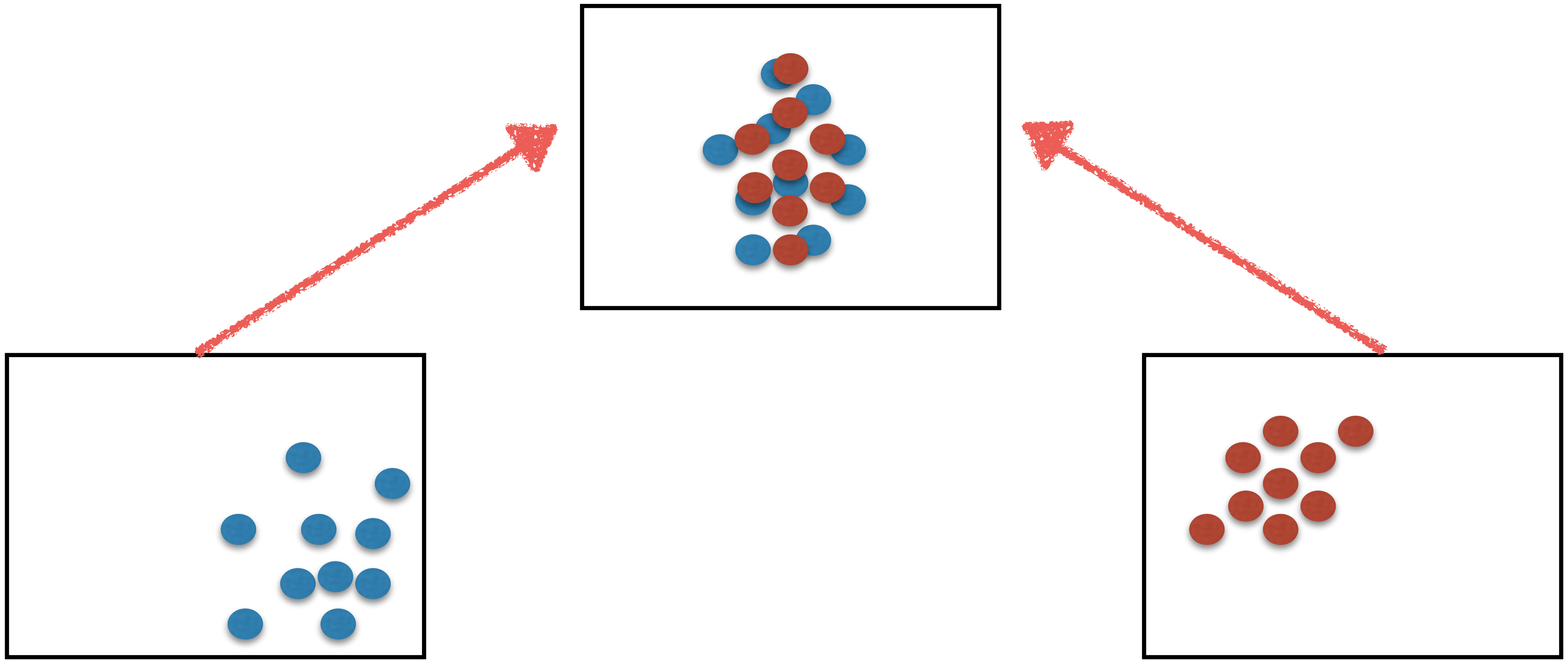
Cross-Project Software Defect Prediction

- Within-project software defect prediction can be applied only after an initial version of the software is available.
- Cross-project data could potentially be used.
- Problem: different projects may have different underlying distributions.



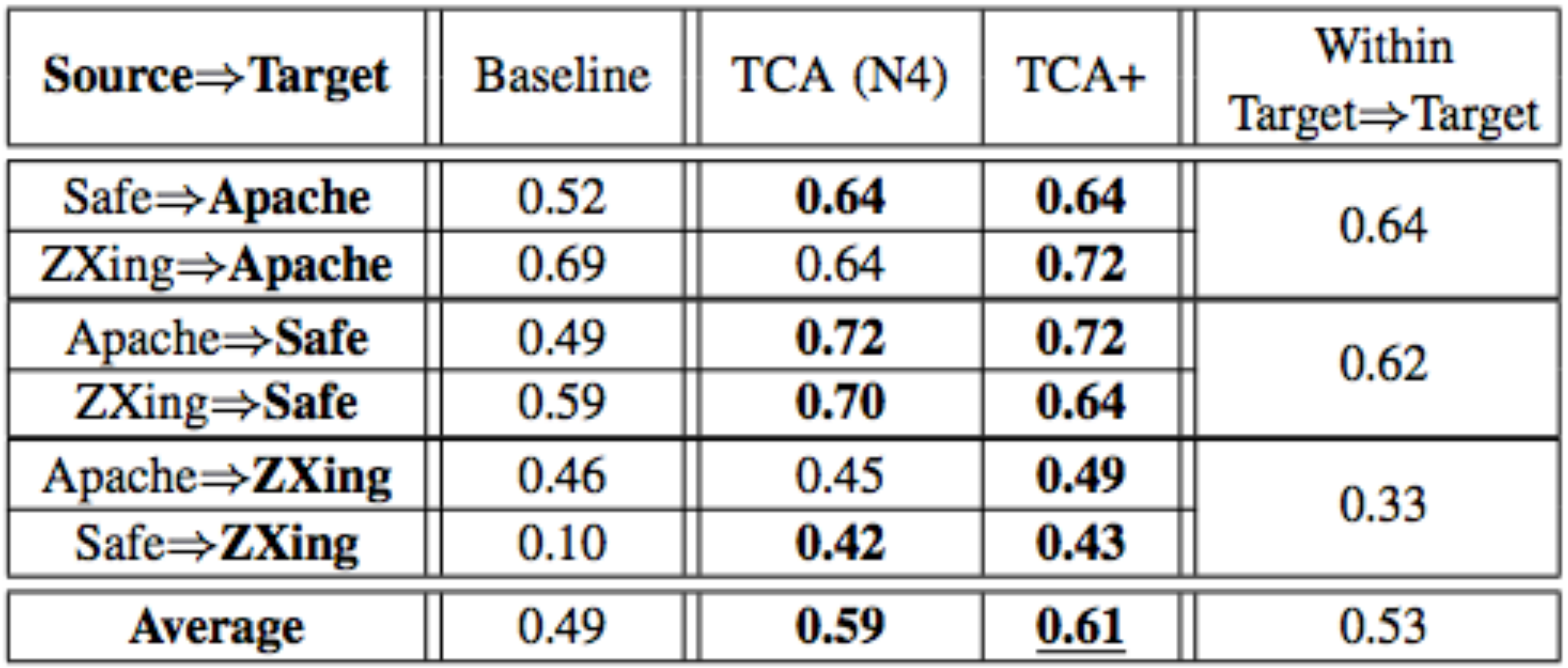
Transfer Learning

- Solution: learn a transformation of the distribution of input features from the source to the target project.
- The input feature distributions from the source and target projects are similar in the transformed space.



J. Nam, S.J. Pan and S. Kim. Transfer Defect Learning, ICSE 2013.

Sample Results (F-Measure)



Source⇒Target	Baseline	TCA (N4)	TCA+	Within Target⇒Target
Safe⇒**Apache**	0.52	**0.64**	**0.64**	0.64
ZXing⇒**Apache**	0.69	0.64	**0.72**	
Apache⇒**Safe**	0.49	**0.72**	**0.72**	0.62
ZXing⇒**Safe**	0.59	**0.70**	**0.64**	
Apache⇒**ZXing**	0.46	0.45	**0.49**	0.33
Safe⇒**ZXing**	0.10	**0.42**	**0.43**	
Average	0.49	**0.59**	**0.61**	0.53

Source: J. Nam, S.J. Pan and S. Kim. Transfer Defect Learning, ICSE 2013.

L. Minku and X. Yao. How to Make Best Use of Cross-company Data in Software Effort Estimation? ICSE 2014.

Test Case Generation

- Automatic generation of inputs for test cases, to compose test suites.
- Objectives:
    - Maximise number of crashes found.
    - Maximise coverage.
    - Minimise size of test suites.
    - Maximise number of mutants killed.

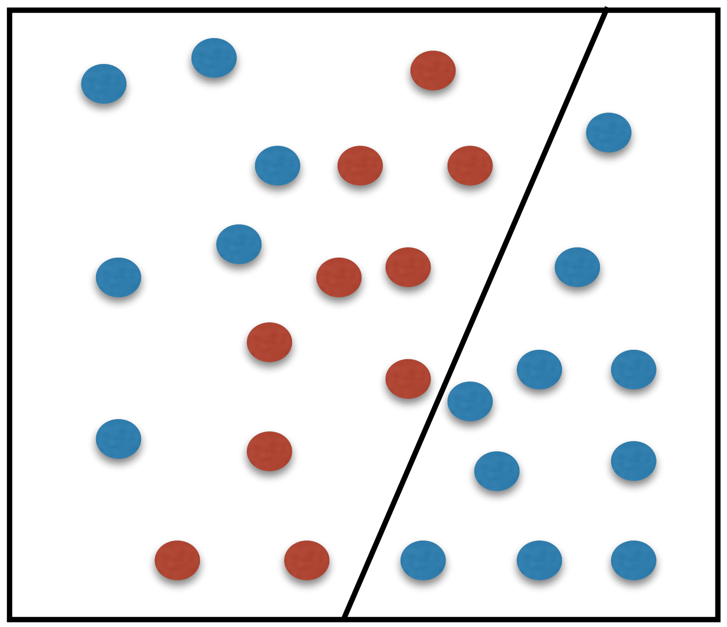
Ke Mao, Mark Harman, Yue Jia. Sapienz: Multi-objective Automated Testing for Android Applications. Proceedings of the 25th International Symposium on Software Testing and Analysis, Pages 94-105, 2016

Gordon Fraser and Andrea Arcuri. Whole Test Suite Generation. IEEE TSE 39(2):276–291, 2013.

$$\begin{tikzcd}\text{Machine Learning} \arrow[r, shift left, "?"] & \text{Software Engineering} \arrow[l, shift left]\end{tikzcd}$$

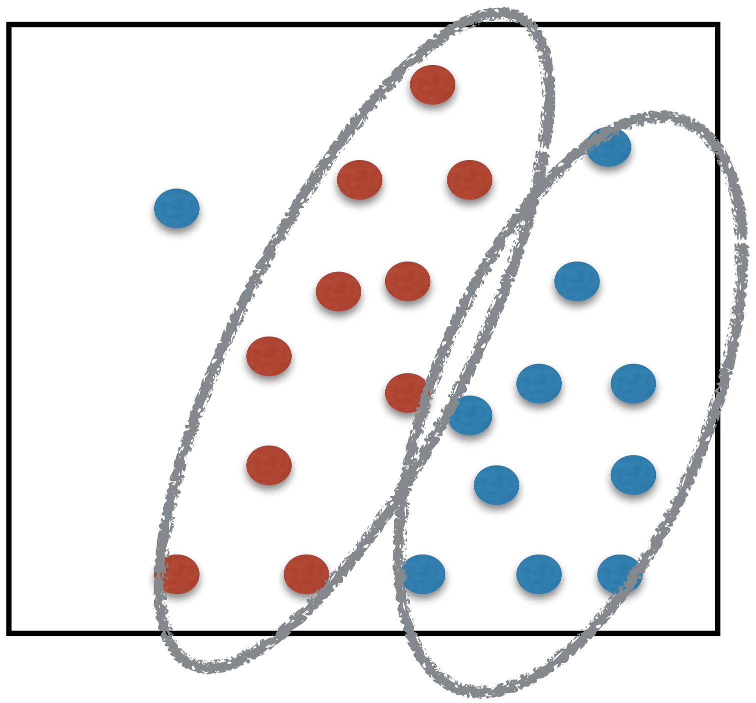
Issues Inherent to Machine Learning

- Machine learning builds models based on training examples.
- If a given region of the input space is not covered by the examples, the resulting model may perform poorly on that region.

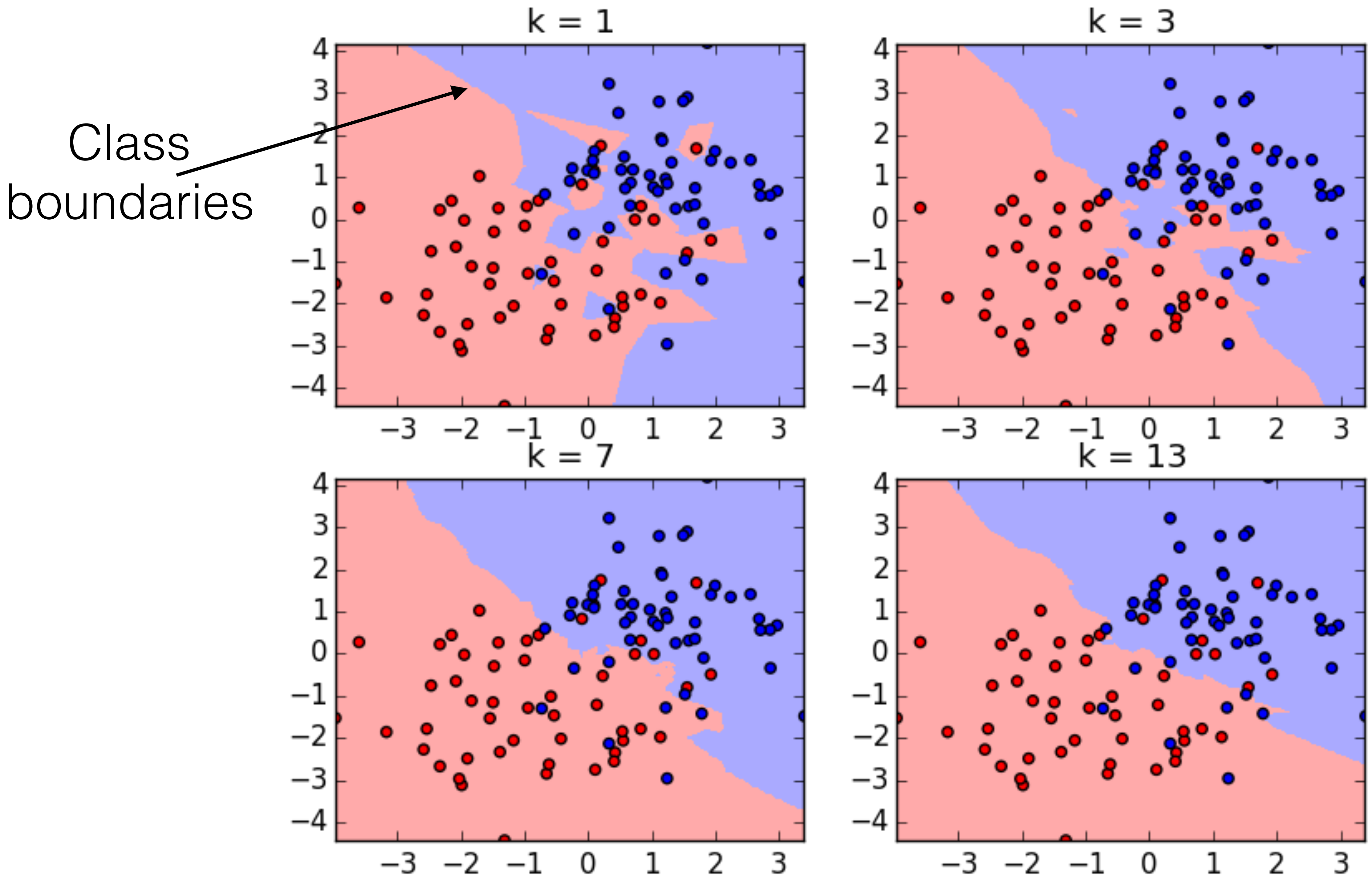


Potential but Partial ML Solutions

- Learn better: collect more representative data.
- Tell you can't predict: novelty detection algorithms.
- Request for more training data: active learning.



Overfitting and Underfitting



Potential but Partial ML Solutions

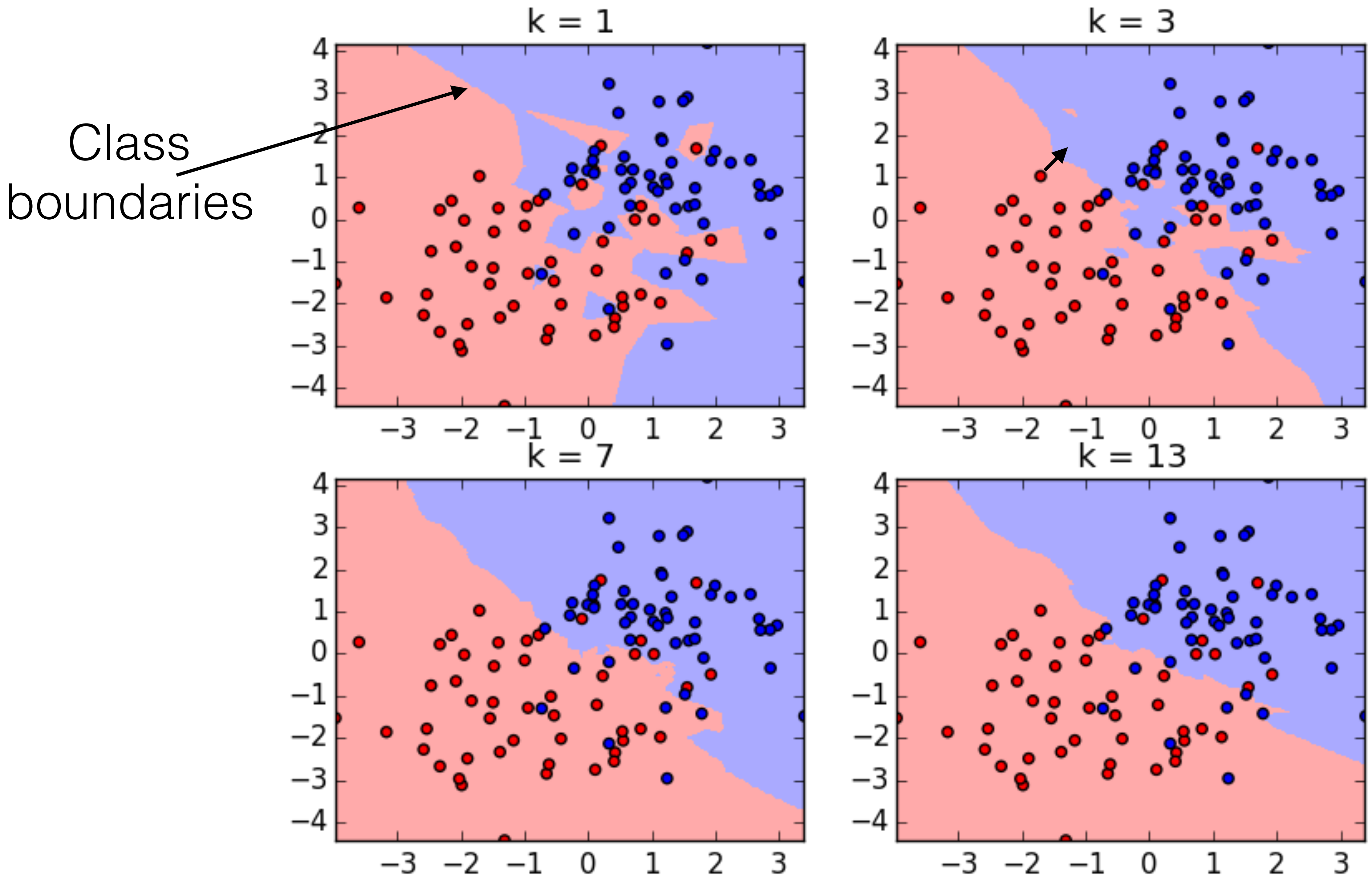
- Learn better:
  - Choose hyperparameters based on validation set.
  - Use regularisation term.
- Estimate how well you do:
  - Test predictive model on a data set not used for training or validation.

L. Song, L. Minku and X. Yao. The Impact of Parameter Tuning on Software Effort Estimation Using Learning Machines. PROMISE 2013.

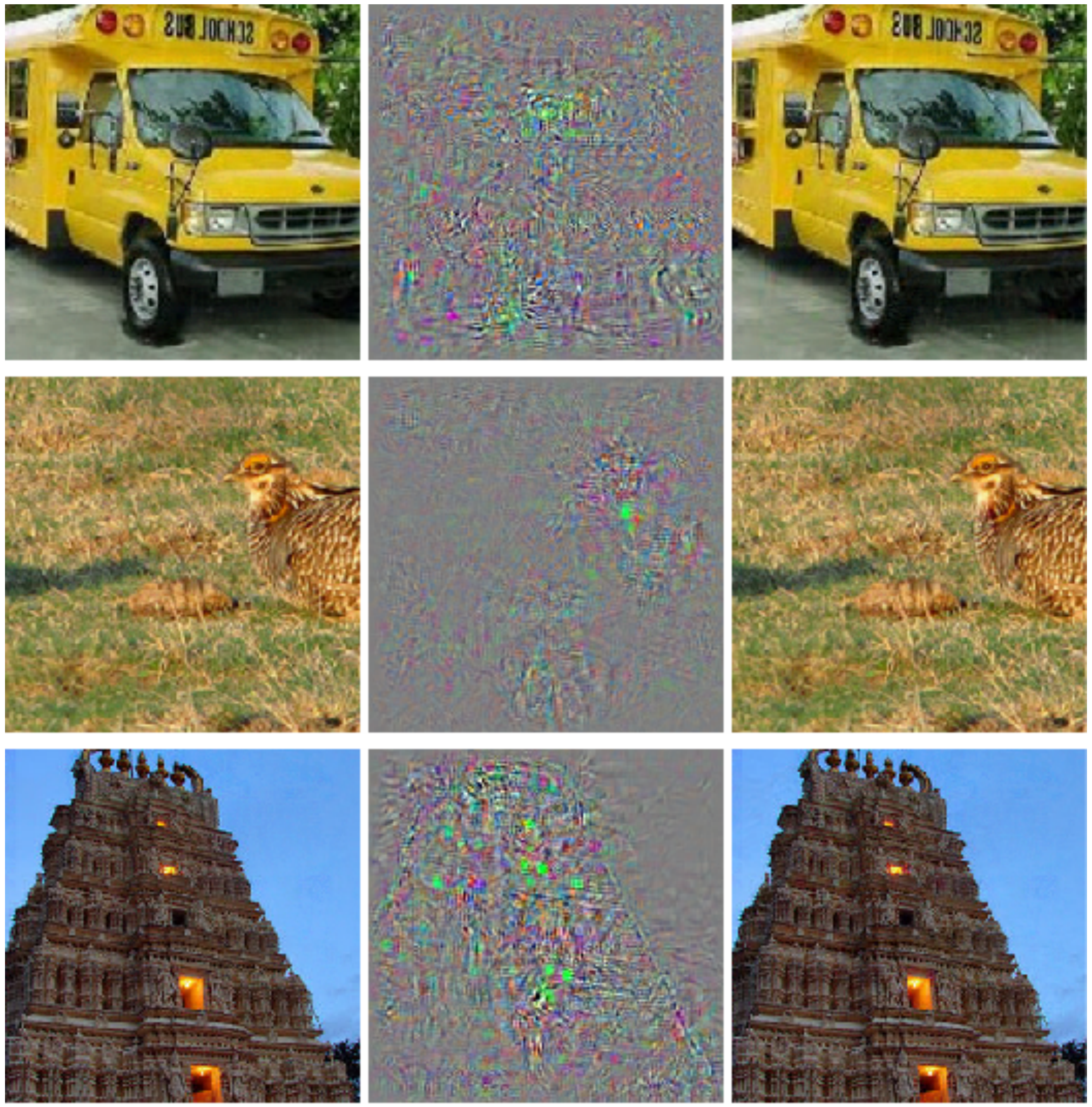
A. Agrawal and T. Menzies. Is “better data” better than “better data miners”? ICSE 2018.

J. Nam, S.J. Pan and S. Kim. Transfer Defect Learning, ICSE 2013.

Adversarial Examples



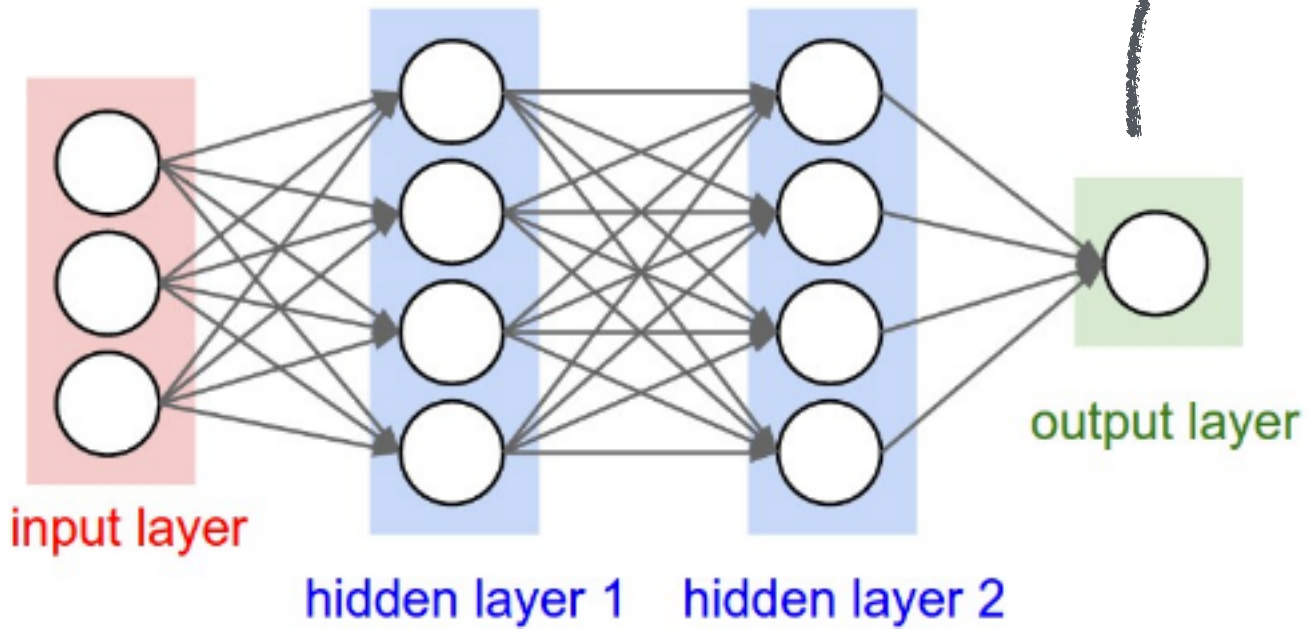
Adversarial Examples

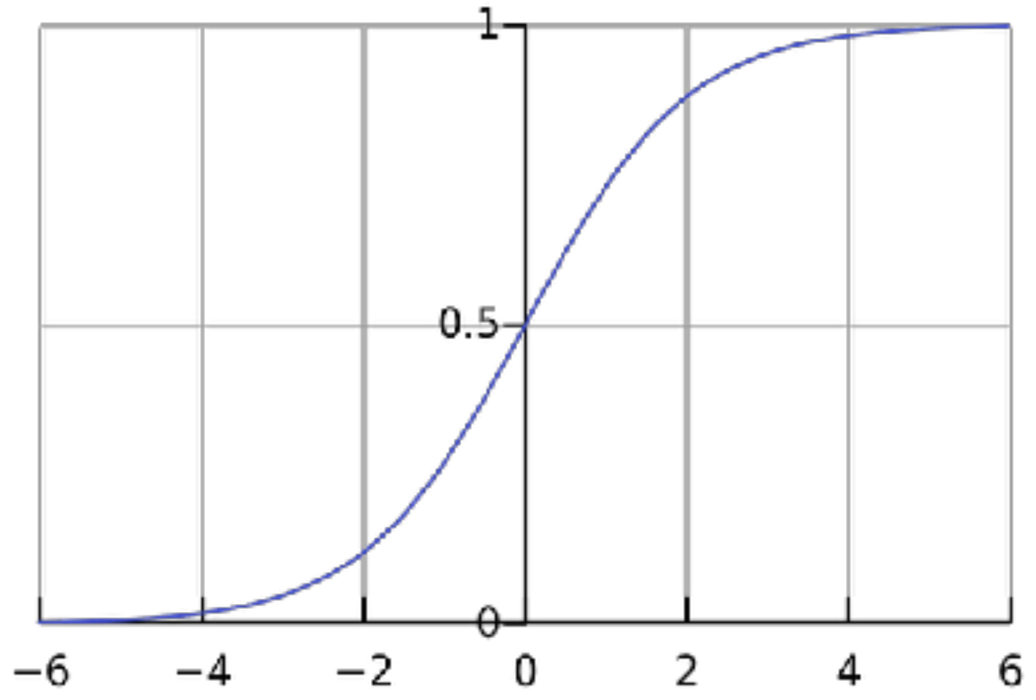


Szegedy et a. Intriguing properties of neural networks. https://arxiv.org/abs/1312.6199

Potential but Partial ML Solutions

- Learn better: find adversarial examples and train on them.
- Tell you can't predict: use confidence of predictions.





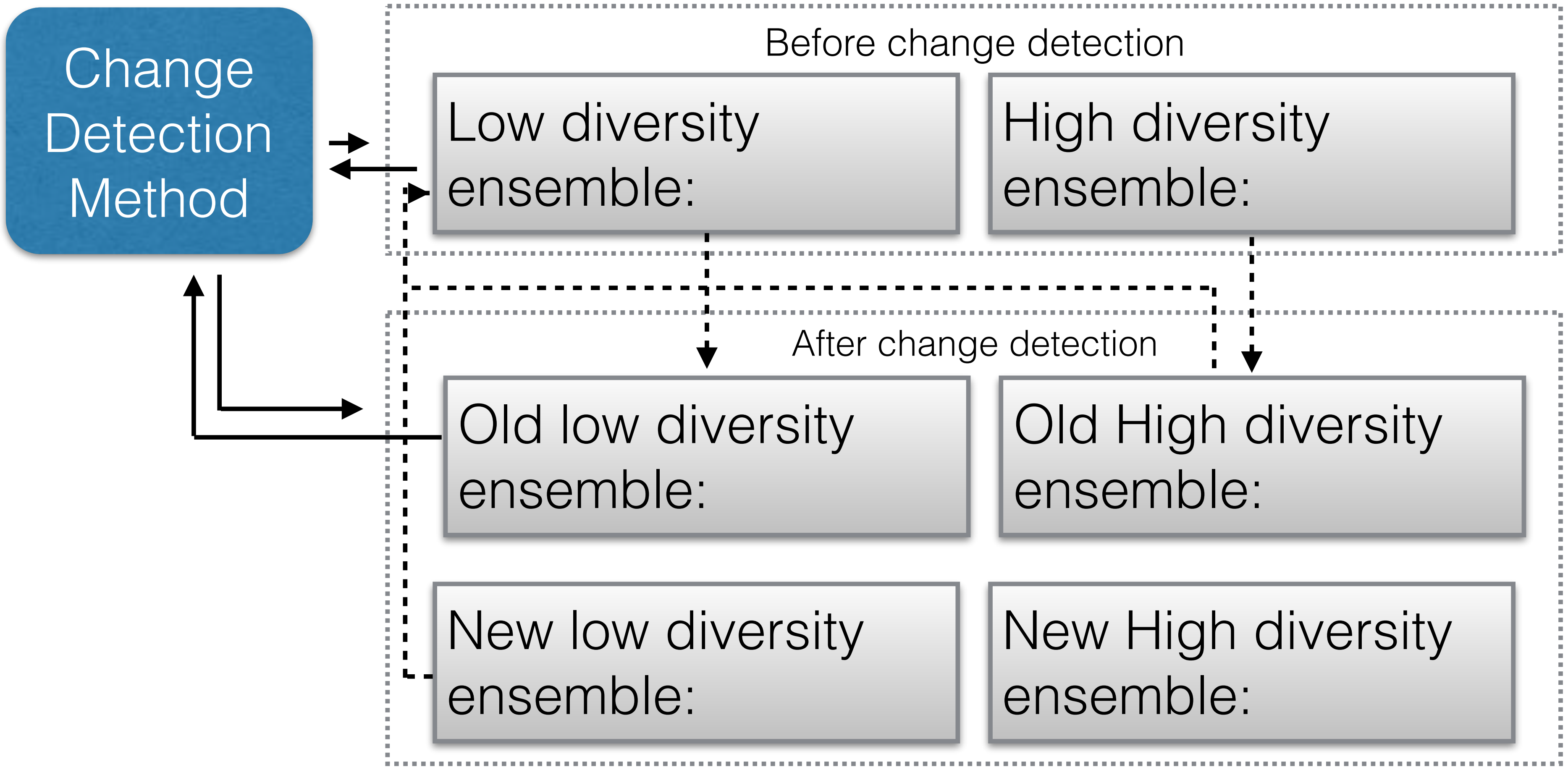
Concept Drift

Changes in the underlying distribution of the data over time, possibly affecting relationship between inputs and output.

L. Minku, X. Yao. DDD: A New Ensemble Approach For Dealing With Concept Drift, IEEE Transactions on Knowledge and Data Engineering, 24(4):619-633, 2012.

Potential but Partial ML Solutions

- Tell you can't predict: concept drift detection methods.
- Learn better: strategies to adapt to concept drift faster.



L. Minku, X. Yao. DDD: A New Ensemble Approach For Dealing With Concept Drift, IEEE Transactions on Knowledge and Data Engineering, 24(4):619-633, 2012.

There Will Always Be Mistakes…

- Should we really use machine learning for life critical applications?
    - It depends on whether machine learning is to be used to complement or replace humans.
    - It depends on whether we are ready for accepting the risk.
    - It depends on how well we can foresee mistakes and prevent drastic outcomes.
    - It depends on how many mistakes the machine learning will do compared to humans.

What Else Can Software Engineering Do To Help?

- Better testing based on ML weaknesses in mind, e.g., test cases close to decision boundaries.
- Continuous testing, as things may change over time.
- Requirements can be written with ML weaknesses in mind, e.g., data collection following certain guidelines.
- Architectures to enable switching away from ML when it is unlikely to perform well.
- Better frameworks for interoperability, to facilitate researchers and practitioners in comparing existing methods.