An Analysis of Clustering Fragility

Xin Yin Vincenzo Musco Iulian Neamtiu

New Jersey Institute of Technology

Software Engineers Are Supposed to Embrace ML…

5 great AI-powered home devices that will improve your life today

courtesy www.t3.com

Computer Science > Software Engineering

Software Engineers vs. Machine Learning Algorithms: An Empirical Study Assessing Performance and Reuse Tasks

Nathalia Nascimento, Carlos Lucena, Paulo Alencar, Donald Cowan

(Submitted on 4 Feb 2018 (v1), last revised 7 Feb 2018 (this version, v2))

Several papers have recently contained reports on applying machine learning (ML) to the automation of software engineering (SE) tasks, such as project management, modeling and development. However, there appear to be no approaches comparing how software engineers fare against machine-learning algorithms as applied to specific software development tasks. Such a comparison is essential to gain insight into which tasks are better performed by humans and which by machine learning and how cooperative work or human-in-the-loop processes can be implemented more effectively. In this paper, we present an empirical study that compares how software engineers and machine-learning algorithms perform and reuse tasks. The empirical study involves the synthesis of the control structure of an autonomous streetlight application. Our approach consists of four steps. First, we solved the problem using machine learning to determine specific performance and reuse tasks. Second, we asked software engineers with different domain knowledge levels to provide a solution to the same tasks. Third, we compared how software engineers fare against machine-learning algorithms when accomplishing the performance and reuse tasks based on criteria such as energy consumption and safety. Finally, we analyzed the results to understand which tasks are better performed by either humans or algorithms so that they can work together more effectively. Such an understanding and the resulting human-in-the-loop approaches, which take into account the strengths and weaknesses of humans and machine-learning algorithms, are fundamental not only to provide a basis for cooperative work in support of software engineering, but also, in other areas.

Comments: 22 pages. To be submitted to IEEE Transactions on Software Engineering
Subjects: **Software Engineering (cs.SE)**; Artificial Intelligence (cs.AI); Human-Computer Interaction (cs.HC); Machine Learning (cs.LG); Neural and Evolutionary Computing (cs.NE)
Cite as: arXiv:1802.01096 [cs.SE]
(or arXiv:1802.01096v2 [cs.SE] for this version)

49,712 views | Jan 1, 2018, 08:33pm

Why Do Developers Find It Hard To Learn Machine Learning?

Janakiram MSV Contributor ⓘ

I cover Cloud Computing, Machine Learning, and Internet of Things

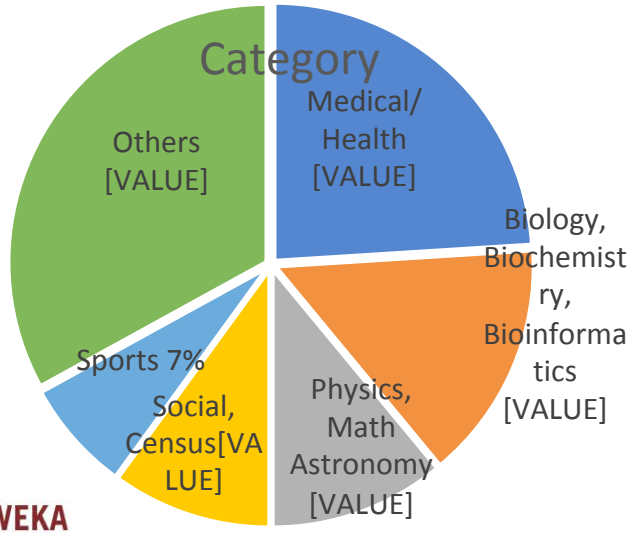
Machine learning (ML) is touted as the most critical skill of current times. Artificial intelligence (AI), an application of ML, is becoming pervasive. From autonomous vehicles to self-tuned databases, AI and ML are found everywhere. Industry analysts often refer to AI-driven automation as the job killer. Almost every domain and industry vertical are getting impacted by AI and ML. Platform companies with massive investments in AI research are shipping new tools and frameworks at a rapid pace.

…but can they rely on "ML as Black Box"?

Our focus: Clustering
Our results: Clustering is Fragile

- Clustering outcomes are **non-deterministic** across runs of the same algorithm/toolkit
- Accuracy can vary from **random** to **perfect**
- Some “unlucky” runs can be **worse-than-random**
- For the same algorithm, **different implementations** have **different accuracies**
- Toolkits might agree with ground truth but **disagree** between themselves

Setup



- PMLB benchmark, 162 datasets
- 7 toolkits:

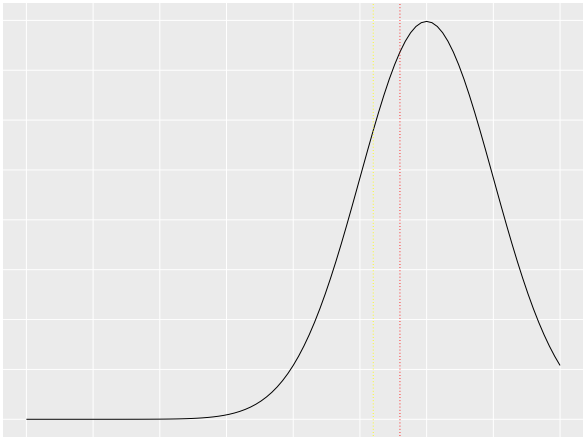
MATLAB mlpack R scikit learn 将軍 TensorFlow RESEARCH CLOUD WEKA The University of Waikato

- Algorithms: *K*-means, K-means++, Expectation-Maximization (Gaussian), Spectral Clustering, Hierarchical Clustering, DBSCAN, Affinity Propagation
- 30 runs for each dataset/algorithm/toolkit configuration (161,192 total)
- Accuracy: Adjusted Rand Index (ARI)

What do accuracy distributions look like?

R (Right skewed, **good** outliers)

L (Left skewed, **bad** outliers)



LR(both good and bad outliers

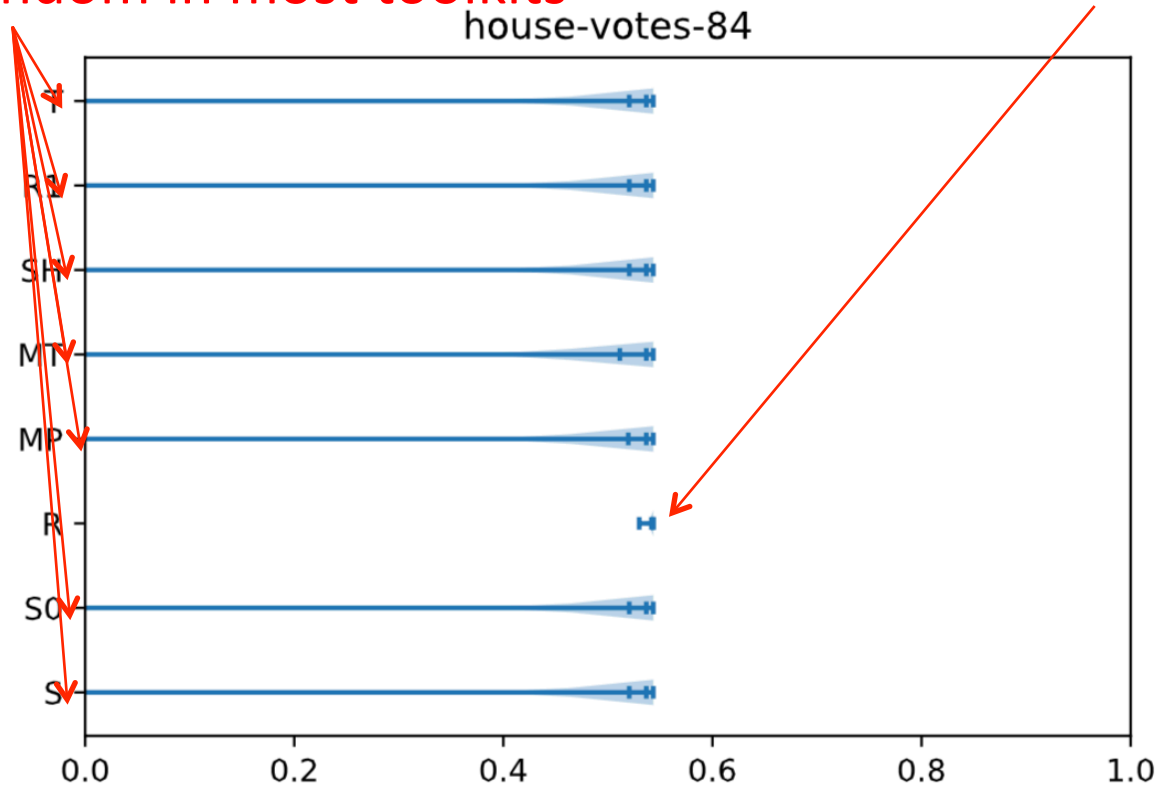
B(Bimodal)

U(Uniform)

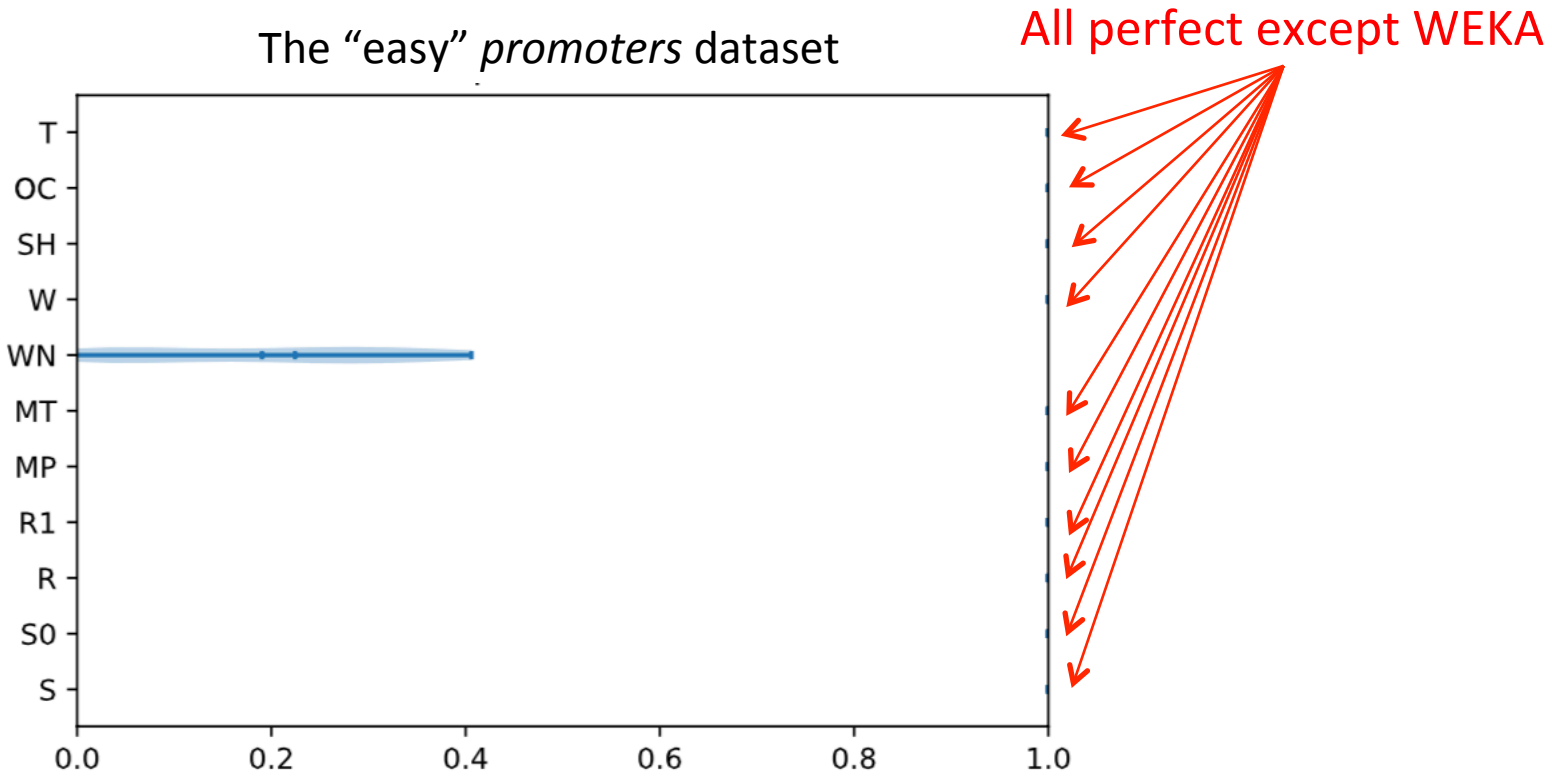
K-means can be worse than random!

Can be worse than random in most toolkits

R to the rescue



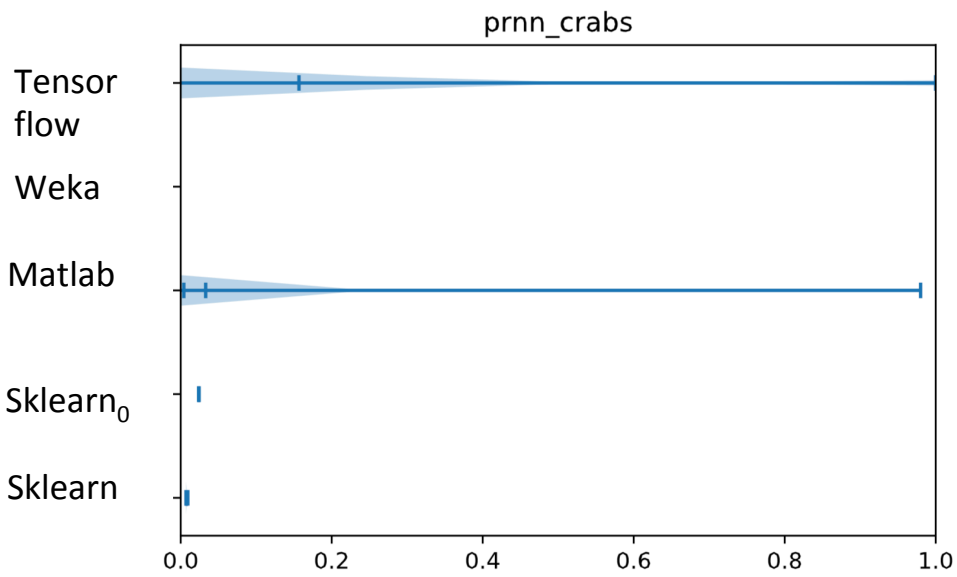
“Broken” implementations?
WEKA’s very low accuracy on K-means++



Confirmed with WEKA developers − need to use non-standard config. to achieve high accuracy

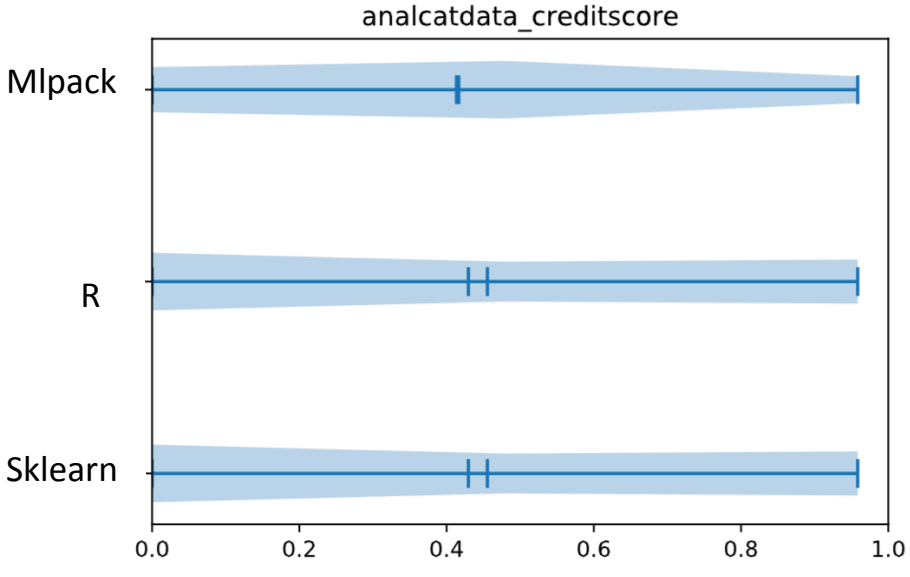
Variation across runs: from *worse-than-random* to *perfect*

Algorithm: Gaussian Mix



Accuracy: -0.005 .. 1

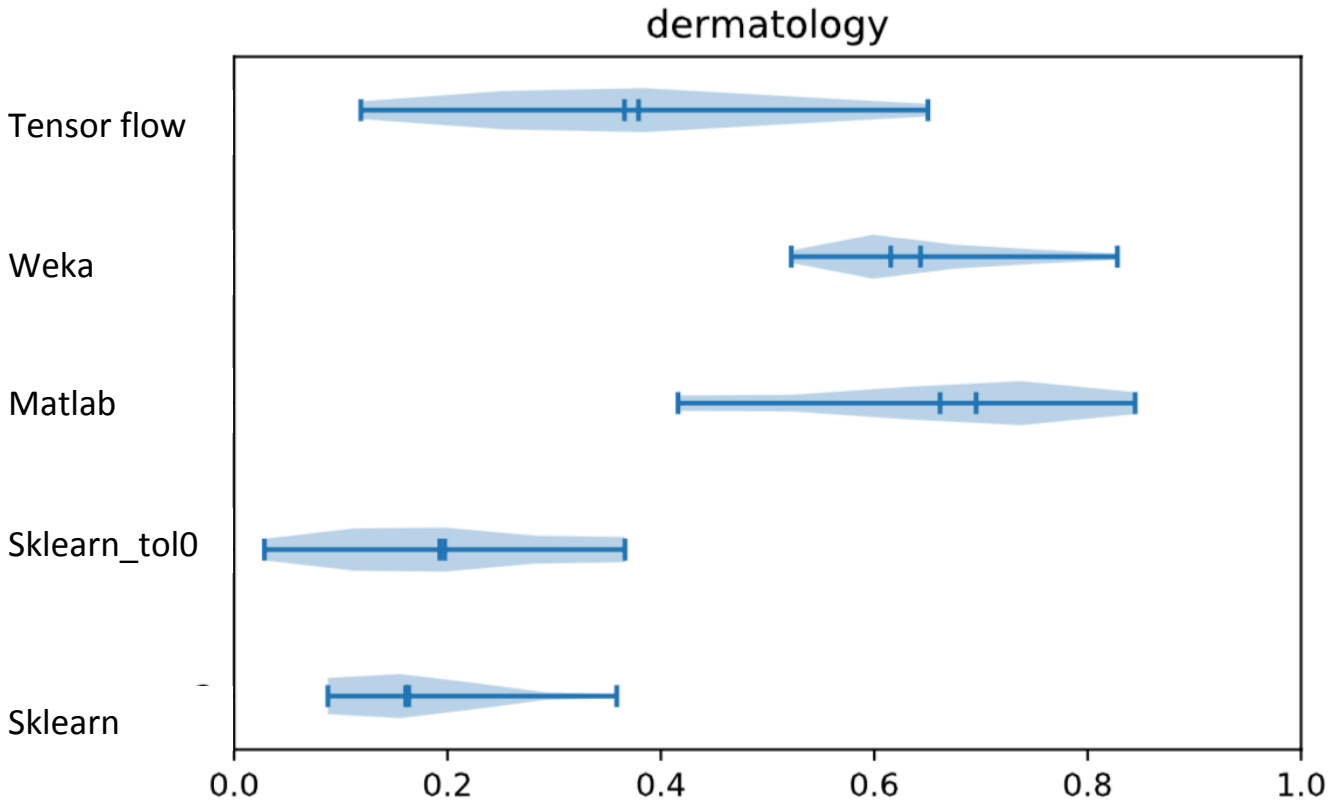
Algorithm: DBSCAN



Accuracy: 0 .. 0.958

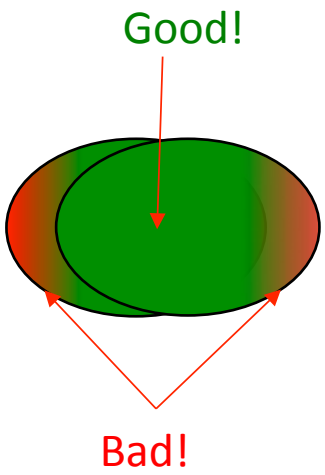
Gaussian mix…it's a mix, alright – you're at the mercy of the implementation

1776 out of 34987 are found!



Toolkits might agree with ground truth, but disagree with each other

Algorithm	Dataset	Toolkit1	Toolkit2	Toolkit1 vs. Ground truth	Toolkit2 vs. Ground truth	Toolkit Agreement
spectral	promoters	sklearnfast	R	0.889	0.962	0.853
gaussian	iris	weka	sklearn	0.759	0.904	0.693



Conclusions

- Pressure upon software engineers to “embrace” ML
- We set out to *manage expectations*
  - Large-scale study to quantify clustering outcomes across different algorithms, toolkits, and multiple runs
- Our findings*: large variations across all these dimensions*
- Implications for Software Engineers
  - Don’t assume algorithms are stable
  - Don’t assume implementations are reliable
- Implications for ML/SE researchers
  - Need **dependable clustering** and
  - More generally, we need **dependable ML**