Software Engineering Best Practices for Machine Learning Applications

Alex Serban, Koen van der Blom, Holger Hoos, Joost Visser

What? Why?

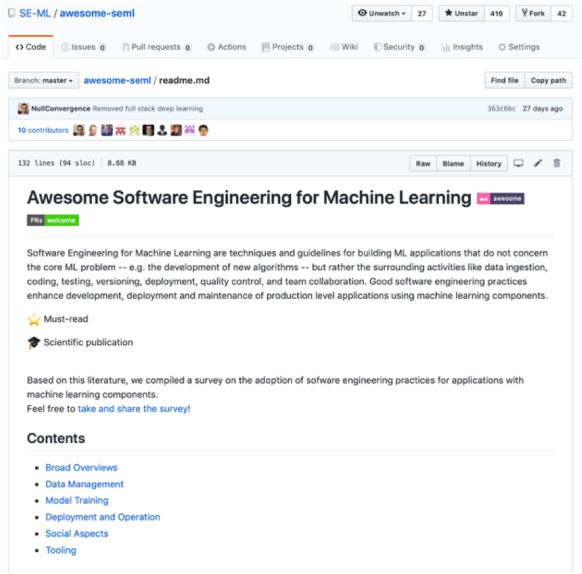
- Investigate which are the software engineering (SE) best practices for machine learning (ML)
- Validate if these practices are adopted by practitioners
- Investigate the effects of adopting these practices (e.g., traceability, software quality)
- Investigate which traditional software engineering best practices apply to ML

How?

- Performed a literature study to discover best practices for ML
- Designed a questionnaire to validate the adoption of practices with practitioners
- Interviewed practitioners to seek their challenges and validate the questionnaire
- Ran a survey with practitioners
- Interpreted the survey results

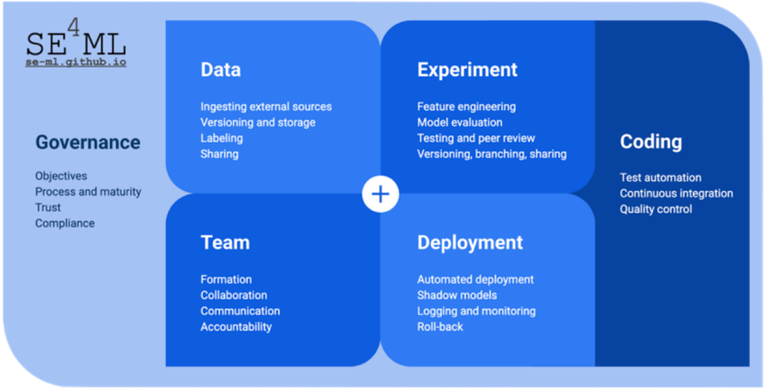
Literature review on SE for ML

- We reviewed over 50 articles, both from academic publications and grey literature
- The majority of articles come from grey literature
- We compiled a curated list of articles, available on Github, at: https://github.com/SE-ML/awesome-seml
- We welcome suggestions and contributions to the literature list, from the community



A catalog of SE practices for ML

- From the literature we compiled a catalog of 29 best practices for ML
- We classified the practices in 6 categories, corresponding to different stages of the broad ML development process
- We made the catalog of practices available online



https://se-ml.github.io/practices/

An example of best practice

Write Reusable Scripts for Data Cleaning and Merging

February, 2020 • Alex Serban, Joost Visser

← 3 / 29 • Data • basic →

Intent

Avoid untidy data wrangling scripts, reuse code and increase reproducibility.

Motivation

Data cleaning and merging are exploratory processes and tend to be less structured. Many times these processes involve manual steps or poorly structured code which can not be later reused or integrated in a pipeline.

Applicability

Reusable data cleaning scripts should be written for any ML application that does not use raw or standard data sets.

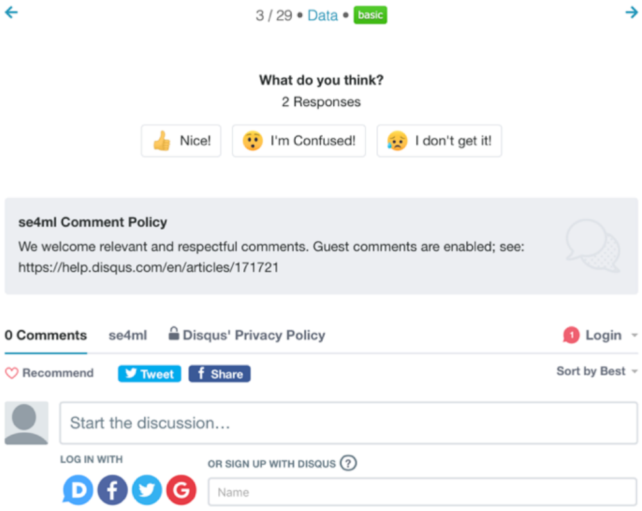
Description

Most of the times, training machine learning models is preceded by an exploratory phase, in which non-structured code is written or manual steps are performed in order to get the data in the right format, or merge several data sources. Especially when using notebooks, there is a tendency to write ad-hoc data processing scripts, which depend on variables already stored in memory when running previous cells.

Before moving to the training phase, it is important to convert this code into reusable scripts and move it into methods which can be called and *tested* individually. This will enable code reuse and ease integration into processing pipelines.

Related

- Test all Feature Extraction Code



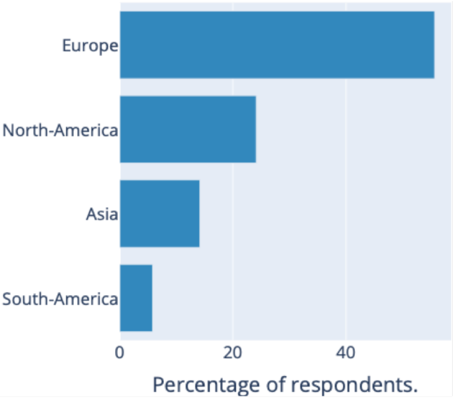
The catalog welcomes comments and contributions from the community

Survey design

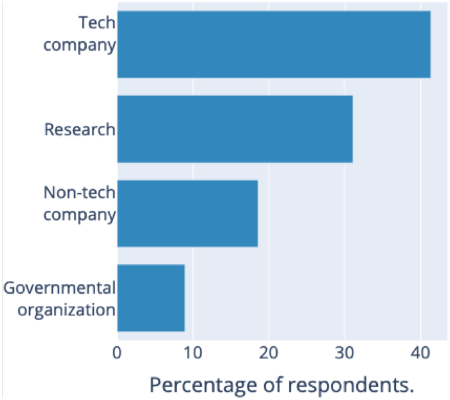
- The survey is an observational study, i.e., we ask teams of participants if they adopt the practices at the moment they fill in the questionnaire
- The questionnaire is divided in 4 parts: Preliminaries, Practice adoption questions, Effects, and Others
- In the preliminaries we asked participants about their background so we can assign them to groups
- The practice questions validate the adoption of practices
- The effects questions allow us to test the hypothesis that adopting a set of practices leads to an intended effect

Survey demographics

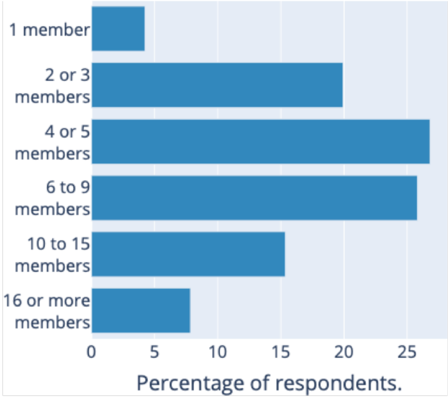
Based on 313 valid answers



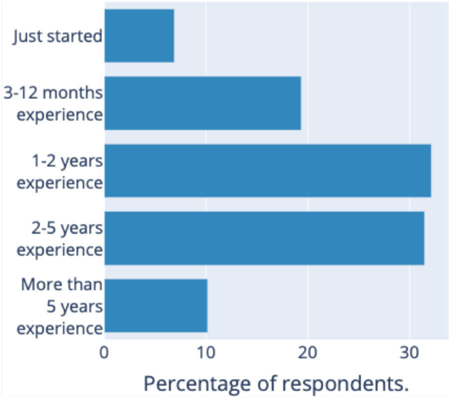
Grouped by continents



Grouped by organization type



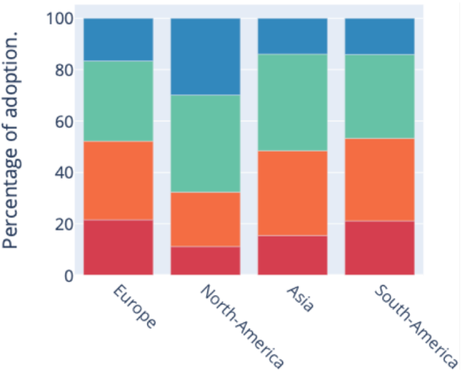
Grouped by team size



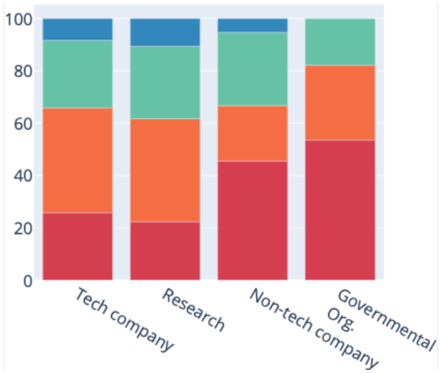
Grouped by team experience

Adoption rate based on demographics

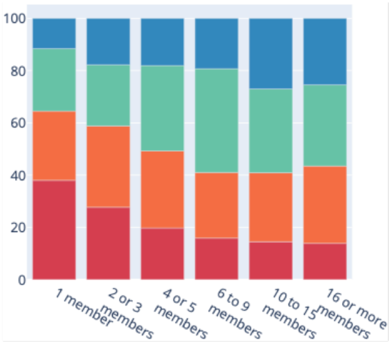
- North America has a different practice adoption
- Most trends are expected (e.g., the adoption of practices increases with team size)
- A small deviation for teams with more than 5 years of experience can be observed (due to practice novelty?)



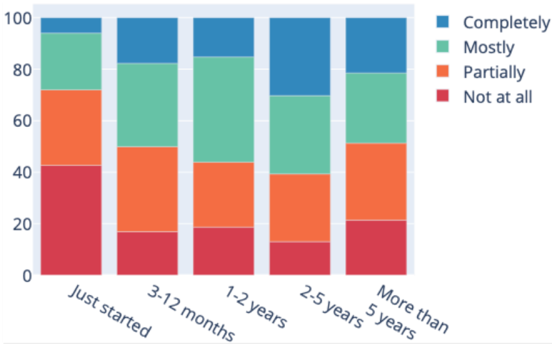
Grouped by continents



Grouped by organization type



Grouped by team size



Grouped by team experience

Adoption rate for individual practices

- We use a simple algorithm to rank the practices by their adoption rate
- The algorithm measures the percentage of “at least completely”, ”at least mostly” and “at least partially” adoption for each practice
- This method is meant to remove some noise stemming from uncertain answers, which lay on the boundaries

Nr.	Title	Class	Type	Ref.	Rank
1	Use Sanity Checks for All External Data Sources	Data	N	[15, 42]	22
2	Check that Input Data is Complete, Balanced and Well Distributed	Data	N	[1, 14, 37, 42, 45]	18
3	Write Reusable Scripts for Data Cleaning and Merging	Data	N	[9, 15, 42]	5
4	Ensure Data Labeling is Performed in a Strictly Controlled Process	Data	N	[10, 18, 38, 40]	11
5	Make Data Sets Available on Shared Infrastructure (private or public)	Data	N	[25, 30, 31, 37]	14
6	Share a Clearly Defined Training Objective within the Team	Training	N	[3, 37, 57]	2
7	Capture the Training Objective in a Metric that is Easy to Measure and Understand	Training	N	[3, 7, 49, 57]	1
8	Test all Feature Extraction Code	Training	M	[14, 44]	23
9	Assign an Owner to Each Feature and Document its Rationale	Training	M	[57]	29
10	Actively Remove or Archive Features That are Not Used	Training	N	[45, 57]	28
11	Peer Review Training Scripts	Training	M	[8]	20
12	Enable Parallel Training Experiments	Training	N	[44, 47]	6
13	Automate Hyper-Parameter Optimization and Model Selection	Training	N	[28, 36]	26
14	Continuously Measure Model Quality and Performance	Training	N	[20, 57]	4
15	Share Status and Outcomes of Experiments Within the Team	Training	N	[25, 34]	7
16	Use Versioning for Data, Model, Configurations and Training Scripts	Training	M	[25, 27, 34, 37, 47, 50, 53]	3
17	Run Automated Regression Tests	Coding	T	[14, 47]	27
18	Use Continuous Integration	Coding	T	[14, 44]	16
19	Use Static Analysis to Check Code Quality	Coding	T	[51]	24
20	Automate Model Deployment	Deployment	M	[4, 43, 49, 50]	15
21	Continuously Monitor the Behavior of Deployed Models	Deployment	N	[4, 13, 20, 44, 47]	12
22	Enable Shadow Deployment	Deployment	M	[4, 13, 50, 53]	25
23	Perform Checks to Detect Skews between Models	Deployment	N	[13, 20, 44, 57]	17
24	Enable Automatic Roll Backs for Production Models	Deployment	M	[4, 44]	13
25	Log Production Predictions with the Model’s Version and Input Data	Deployment	M	[5, 27, 37]	19
26	Use A Collaborative Development Platform	Team	T	[]	8
27	Work Against a Shared Backlog	Team	T	[46, 48]	9
28	Communicate, Align, and Collaborate With Multidisciplinary Team Members	Team	T	[21]	10
29	Enforce Fairness and Privacy	Governance	N	[2, 6, 14]	21

Top 5 most adopted practices

Nr.	Title	Class	Type	Ref.	Rank
1	Use Sanity Checks for All External Data Sources	Data	N	[15, 42]	22
2	Check that Input Data is Complete, Balanced and Well Distributed	Data	N	[1, 14, 37, 42, 45]	18
3	Write Reusable Scripts for Data Cleaning and Merging	Data	N	[9, 15, 42]	5
4	Ensure Data Labeling is Performed in a Strictly Controlled Process	Data	N	[10, 18, 38, 40]	11
5	Make Data Sets Available on Shared Infrastructure (private or public)	Data	N	[25, 30, 31, 37]	14
6	Share a Clearly Defined Training Objective within the Team	Training	N	[3, 37, 57]	2
7	Capture the Training Objective in a Metric that is Easy to Measure and Understand	Training	N	[3, 7, 49, 57]	1
8	Test all Feature Extraction Code	Training	M	[14, 44]	23
9	Assign an Owner to Each Feature and Document its Rationale	Training	M	[57]	29
10	Actively Remove or Archive Features That are Not Used	Training	N	[45, 57]	28
11	Peer Review Training Scripts	Training	M	[8]	20
12	Enable Parallel Training Experiments	Training	N	[44, 47]	6
13	Automate Hyper-Parameter Optimization and Model Selection	Training	N	[28, 36]	26
14	Continuously Measure Model Quality and Performance	Training	N	[20, 57]	4
15	Share Status and Outcomes of Experiments Within the Team	Training	N	[25, 34]	7
16	Use Versioning for Data, Model, Configurations and Training Scripts	Training	M	[25, 27, 34, 37, 47, 50, 53]	3
17	Run Automated Regression Tests	Coding	T	[14, 47]	27
18	Use Continuous Integration	Coding	T	[14, 44]	16
19	Use Static Analysis to Check Code Quality	Coding	T	[51]	24
20	Automate Model Deployment	Deployment	M	[4, 43, 49, 50]	15
21	Continuously Monitor the Behavior of Deployed Models	Deployment	N	[4, 13, 20, 44, 47]	12
22	Enable Shadow Deployment	Deployment	M	[4, 13, 50, 53]	25
23	Perform Checks to Detect Skews between Models	Deployment	N	[13, 20, 44, 57]	17
24	Enable Automatic Roll Backs for Production Models	Deployment	M	[4, 44]	13
25	Log Production Predictions with the Model's Version and Input Data	Deployment	M	[5, 27, 37]	19
26	Use A Collaborative Development Platform	Team	T	[]	8
27	Work Against a Shared Backlog	Team	T	[46, 48]	9
28	Communicate, Align, and Collaborate With Multidisciplinary Team Members	Team	T	[21]	10
29	Enforce Fairness and Privacy	Governance	N	[2, 6, 14]	21

Top 5 least adopted practices

Nr.	Title	Class	Type	Ref.	Rank
1	Use Sanity Checks for All External Data Sources	Data	N	[15, 42]	22
2	Check that Input Data is Complete, Balanced and Well Distributed	Data	N	[1, 14, 37, 42, 45]	18
3	Write Reusable Scripts for Data Cleaning and Merging	Data	N	[9, 15, 42]	5
4	Ensure Data Labeling is Performed in a Strictly Controlled Process	Data	N	[10, 18, 38, 40]	11
5	Make Data Sets Available on Shared Infrastructure (private or public)	Data	N	[25, 30, 31, 37]	14
6	Share a Clearly Defined Training Objective within the Team	Training	N	[3, 37, 57]	2
7	Capture the Training Objective in a Metric that is Easy to Measure and Understand	Training	N	[3, 7, 49, 57]	1
8	Test all Feature Extraction Code	Training	M	[14, 44]	23
9	Assign an Owner to Each Feature and Document its Rationale	Training	M	[57]	29
10	Actively Remove or Archive Features That are Not Used	Training	N	[45, 57]	28
11	Peer Review Training Scripts	Training	M	[8]	20
12	Enable Parallel Training Experiments	Training	N	[44, 47]	6
13	Automate Hyper-Parameter Optimization and Model Selection	Training	N	[28, 36]	26
14	Continuously Measure Model Quality and Performance	Training	N	[20, 57]	4
15	Share Status and Outcomes of Experiments Within the Team	Training	N	[25, 34]	7
16	Use Versioning for Data, Model, Configurations and Training Scripts	Training	M	[25, 27, 34, 37, 47, 50, 53]	3
17	Run Automated Regression Tests	Coding	T	[14, 47]	27
18	Use Continuous Integration	Coding	T	[14, 44]	16
19	Use Static Analysis to Check Code Quality	Coding	T	[51]	24
20	Automate Model Deployment	Deployment	M	[4, 43, 49, 50]	15
21	Continuously Monitor the Behavior of Deployed Models	Deployment	N	[4, 13, 20, 44, 47]	12
22	Enable Shadow Deployment	Deployment	M	[4, 13, 50, 53]	25
23	Perform Checks to Detect Skews between Models	Deployment	N	[13, 20, 44, 57]	17
24	Enable Automatic Roll Backs for Production Models	Deployment	M	[4, 44]	13
25	Log Production Predictions with the Model's Version and Input Data	Deployment	M	[5, 27, 37]	19
26	Use A Collaborative Development Platform	Team	T	[]	8
27	Work Against a Shared Backlog	Team	T	[46, 48]	9
28	Communicate, Align, and Collaborate With Multidisciplinary Team Members	Team	T	[21]	10
29	Enforce Fairness and Privacy	Governance	N	[2, 6, 14]	21

Adoption of practices based on practice type

- We defined 3 types of practices: Traditional (SE practices), Modified (from traditional for ML) and New (only for ML)
- The new practices are the most adopted, which means ML teams focus on ML practices

Practice Type	At least high adoption	At least medium adoption	At least low adoption
Traditional	15.6 %	47.8 %	76.8 %
Modified	11.3 %	42.0 %	76.9 %
New	**16.9** %	**50.0** %	**83.9** %

Alex Serban
cs.ru.nl/~aserban

Adoption of practices based on the data type

- For the three most used data types, the adoption of practices is consistent
- This result implies that the practices apply equally to the data types
- For the other data types, we plan to collect more data

Data Type	Perc. of respondents	Adoption At least high	At least medium	At least low
Tabular Data	31.7%	18.0%	50.1%	70.2%
Text	29.7%	19.3%	52.6%	71.4%
Images, Videos	26.4%	19.3%	50.5%	71.5%
Audio	8.8%	24.42%	55.8%	72.6%
Time Series	2.6%	28.2%	60.3%	72.6%
Graphs	0.5%	-	-	-

Relationship between practices and effects

- 4 effects were built in the survey, as shown below
- In all cases, the adoption of practices correlates strongly with the effects

Effects	Description	Practices	p-value	R^2
Agility	The team can quickly experiment with new data and algorithms, and quickly assess and deploy new models	12, 18, 22, 24, 28	$7 \cdot 10^{-4}$	0.84
Software Quality	The software produced is of high quality (technical and functional)	9, 10, 11, 17, 18, 19	$5 \cdot 10^{-3}$	0.95
Team Effectiveness	Experts with different skill sets (e.g., data science, software development, operations) collaborate efficiently	6, 26, 27, 28	$1 \cdot 10^{-5}$	0.98
Traceability	Outcomes of production models can easily be traced back to model configuration and input data	3, 5, 16, 25, 27	$4 \cdot 10^{-6}$	0.75

Predicting effects from practice adoption

- We train 4 ML regression models for each effect, to predict the effects from the practices
- In all cases, the effects can be well predicted from the practices

Effects	Practices	MSE / R^2 / ρ Linear Regression	MSE / R^2 / ρ RF	MSE / R^2 / ρ RF Grid Search	MSE / R^2 / ρ AutoML
Agility	12, 18, 20, 21, 22, 28	0.69 / 0.44 / 0.68	0.27 / 0.78 / 0.92	0.25 / 0.80 / 0.92	**0.24 / 0.82 / 0.92**
Software Quality	9, 10, 11, 17, 18, 19	0.35 / 0.71 / 0.83	**0.12 / 0.90 / 0.91**	0.17 / 0.87 / 0.91	0.17 / 0.87 / 0.91
Team Effectiveness	6, 26, 27, 28	0.45 / 0.63 / 0.87	0.25 / 0.80 / 0.90	0.19 / 0.84 / 0.92	**0.18 / 0.85 / 0.92**
Traceability	3, 5, 16, 21, 25, 27	0.38 / 0.69 / 0.80	0.22 / 0.82 / 0.90	**0.21 / 0.83 / 0.93**	0.22 / 0.82 / 0.93

Practice importance for each effect

- For each effect, we plot the practice importance as revealed by the predictive models, and their adoption rate
- The practice importance is measured by the Shapley values
- The plot gives an overview of which practice to adopt first, in order to maximize return of investment for achieving a desired effect

Conclusions

Materials:

- reading list: https://github.com/SE-ML/awesome-seml
- catalog of practices: https://se-ml.github.io/practices
- survey: https://se-ml.github.io/survey/
- news: https://se-ml.github.io

Alex Serban
cs.ru.nl/~aserban