Best Practices: Software Engineering, Machine Learning, and AutoML

Koen van der Blom

Universiteit Leiden
The Netherlands
Koen van der Blom

• Background
  • The Hague University of Applied Sciences
    • Bachelor Informatica
  • Leiden University
    • MSc Comp. Sci. → PhD → Postdoc

• Research
  • Meta-algorithmics, automated ML+AI
  • Multi-objective optimisation
  • Evolutionary computation and swarm intelligence
  • Software engineering for ML
Machine learning

• Many techniques

• You learned
  • How they work
  • How you can use them

• Industry ‘real-world’
  • ML as part of a larger system
Image search

(imaginary system)
Image search

(Imaginary system)

Koen van der Blom

Best Practices: SE, ML, AutoML
Image search

Input

Hash input

Check DB

ML model

Web crawler

Output

(imaginary system)
Image search

- Hash input
- Check DB
- Output
- Input
- ML model
- Web crawler

Interface design (visual + technical)

(imaginary system)
Image search

- Find (source of) original or similar image

- Input image
  - Did someone look for it before?
    - Check input URL/image;
    - Hash function;
    - Basic/cheap features
  - Yes: Look up in database, done!
  - No: Run through ML model

- Interface: Also part of the system!
  - Receive user input
  - Display results
Engineering software with ML

• Software engineering (SE)
  + Machine learning

• Best practices: Guidelines, can have exceptions!
Engineering software with ML

• Software engineering (SE)
  + Machine learning

• Best practices: Guidelines, can have exceptions!

• Improve
  • Agility
  • Software quality
  • Team effectiveness
  • Traceability
Effects of best practices

• Agility: Ability to easily adapt or add functionality
  • Can we quickly add new functionality?
  • Is it easy to make a change, e.g., based on user feedback?
Effects of best practices

- Agility: Ability to easily adapt or add functionality
  - Can we quickly add new functionality?
  - Is it easy to make a change, e.g., based on user feedback?

- Software quality: High technical and functional quality
  - Does the design make sense for what we are trying to achieve?
Effects of best practices

- Agility: Ability to easily adapt or add functionality
  - Can we quickly add new functionality?
  - Is it easy to make a change, e.g., based on user feedback?

- Software quality: High technical and functional quality
  - Does the design make sense for what we are trying to achieve?

- Team effectiveness: Efficient collaboration between professionals with different skills
  - Are we getting the most out of the team we have?
Effects of best practices

• Agility: Ability to easily adapt or add functionality
  • Can we quickly add new functionality?
  • Is it easy to make a change, e.g., based on user feedback?

• Software quality: High technical and functional quality
  • Does the design make sense for what we are trying to achieve?

• Team effectiveness: Efficient collaboration between professionals with different skills
  • Are we getting the most out of the team we have?

• Traceability: Trace work items in the development lifecycle
  • Why did we develop this code?
  • Does it do what we intended?
Best practices for SE

• What do you already know?
Best practices for SE

• What do you already know?

• Things that may be familiar
  • Collaborative code management (e.g., git)
  • Testing (unit tests, regression tests)
  • Documentation (comments, diagrams)
  • Development methodology (e.g., agile/scrum)
More best practices for SE

• Automate deployment
  • and rollback in case of errors
More best practices for SE

- Automate deployment
  - and rollback in case of errors

- Shared backlog

Image by Jeff.lasovski – License: https://creativecommons.org/licenses/by-sa/3.0/deed.en
More best practices for SE

• Automate deployment
  • and rollback in case of errors

• Shared backlog

• Continuous integration
  • Automatic build/compilation, static analysis, testing
More best practices for SE

• Automate deployment
  • and rollback in case of errors

• Shared backlog

• Continuous integration
  • Automatic build/compilation, static analysis, testing

• Peer review
More best practices for SE

• Automate deployment
  • and rollback in case of errors

• Shared backlog

• Continuous integration
  • Automatic build/compilation, static analysis, testing

• Peer review

• Versioning
Software with ML components

• SE with ML is different from just SE
• Do engineering practices still apply?
• Do we have to change practices?
• Do we need new practices?
Software with ML components

• SE with ML is different from just SE
• Do engineering practices still apply?
• Do we have to change practices?
• Do we need new practices?
• Yes ...all of those!
Practices that still apply

• Still working with code
  • Automated regression testing → Does it still work?
  • Continuous integration → Automatically build/compile
  • Static analysis → Check code quality

• Still a team effort
  • Collaborative development platform → Integrate changes
  • Use a shared backlog → Task status, priority
  • Communicate → Still working towards the same goal?
Practices that still apply

• Limited attention for these practices in ML

<table>
<thead>
<tr>
<th>Title</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run Automated Regression Tests</td>
<td>27</td>
</tr>
<tr>
<td>Use Continuous Integration</td>
<td>16</td>
</tr>
<tr>
<td>Use Static Analysis to Check Code Quality</td>
<td>24</td>
</tr>
<tr>
<td>Use A Collaborative Development Platform</td>
<td>8</td>
</tr>
<tr>
<td>Work Against a Shared Backlog</td>
<td>9</td>
</tr>
<tr>
<td>Communicate, Align, and Collaborate With Multidisciplinary Team Members</td>
<td>10</td>
</tr>
</tbody>
</table>

Out of 29

From: Serban et al. 2020

• Adopt and you are ahead of the competition!
What is different with ML?
What is different with ML?

• Use data

• Create ML models

• Optimise ML system for accuracy
  • Choose best features
  • Optimise parameter settings
  • Etc.

• Run on new data, from real people...
Practices that change

• Testing $\rightarrow$ Feature extraction code
Practices that change

- Testing → Feature extraction code

- Documentation → What does each feature (property) mean, why does it make sense to use?
Practices that change

- Testing → Feature extraction code
- Documentation → What does each feature (property) mean, why does it make sense to use?
- Peer review → Training scripts
Practices that change

• Testing $\rightarrow$ Feature extraction code

• Documentation $\rightarrow$ What does each feature (property) mean, why does it make sense to use?

• Peer review $\rightarrow$ Training scripts

• Versioning $\rightarrow$ Datasets, models, etc.
Practices that change

• Testing → Feature extraction code

• Documentation → What does each feature (property) mean, why does it make sense to use?

• Peer review → Training scripts

• Versioning → Datasets, models, etc.

• Automate → Model deployment, roll backs
Practices that change

- Testing $\rightarrow$ Feature extraction code
- Documentation $\rightarrow$ What does each feature (property) mean, why does it make sense to use?
- Peer review $\rightarrow$ Training scripts
- Versioning $\rightarrow$ Datasets, models, etc.
- Automate $\rightarrow$ Model deployment, roll backs
- Logging $\rightarrow$ Every prediction a model makes, including version numbers and input data
New practices

• Check the data is complete, balanced, well distributed
New practices

• Check the data is complete, balanced, well distributed

• Create reusable data cleaning and merging scripts
New practices

• Check the data is complete, balanced, well distributed

• Create reusable data cleaning and merging scripts

• Make sure data labelling is done as controlled process
New practices

• Check the data is complete, balanced, well distributed

• Create reusable data cleaning and merging scripts

• Make sure data labelling is done as controlled process

• Share data sets → ensure everyone works on the same
New practices

• Share clear training objectives in the team
New practices

• Share clear training objectives in the team

• Use training metrics that are easy to measure and understand
New practices

• Share clear training objectives in the team

• Use training metrics that are easy to measure and understand

• Remove or archive unused features
New practices

• Share clear training objectives in the team

• Use training metrics that are easy to measure and understand

• Remove or archive unused features

• Enable parallel training experiments
New practices

- Share clear training objectives in the team
- Use training metrics that are easy to measure and understand
- Remove or archive unused features
- Enable parallel training experiments
- Share experimental results → Avoid repetition
New practices

• Share clear training objectives in the team

• Use training metrics that are easy to measure and understand

• Remove or archive unused features

• Enable parallel training experiments

• Share experimental results → Avoid repetition

• Monitor deployed models
Automated machine learning

• Feature generation
• Feature selection

• Model selection
• Algorithm selection

• Hyperparameter optimisation

• Algorithm configuration
• Neural architecture search
Automated machine learning

- Feature generation
- Feature selection
- Model selection
- Algorithm selection
- Algorithm configuration
- Neural architecture search
- Hyperparameter optimisation

but all automated!
Automated machine learning

- Feature generation
- Feature selection
- Model selection
- Algorithm selection but all automated!
- Hyperparameter optimisation
- Algorithm configuration
- Neural architecture search

...and more
Automated selection

- Which model/algorithm to choose?

- Best is not always the same
  - Even with relatively small changes!

- Manually run things, see what works where
Automated selection

• Which model/algorithm to choose?
  • Best is not always the same
    • Even with relatively small changes!
  • Manually run things, see what works where

• Or: Automated ML system
  • E.g., Automate choice of model to deploy
  • Predict which model to use for which input
Automated tuning / configuration

• Which settings/parameters to use?

• Which algorithm/model architecture works best?

• Difficult to know what works well
  • Try many things → takes a lot of human time
  • Find improvements → but is it really the best?
Automated tuning / configuration

• Which settings/parameters to use?

• Which algorithm/model architecture works best?

• Difficult to know what works well
  • Try many things → takes a lot of human time
  • Find improvements → but is it really the best?

• Or: Automated ML system

• E.g. Neural architecture search
  • Optimised (deep) neural network for our use case
All is well then, ...or is it?

• Can we trust the system?

• Uses data, but which data, and how?

• Are the results biased?

• Do the users know it is an ML system, can they raise concerns when things go wrong?

• Can we explain why the ML system does what it does?
Trustworthy ML

• Test for social bias in training data
Trustworthy ML

• Test for social bias in training data

• Stop discriminatory attributes from being used as features
Trustworthy ML

• Test for social bias in training data

• Stop discriminatory attributes from being used as features

• Watch out for subgroup bias
Trustworthy ML

- Test for social bias in training data
- Stop discriminatory attributes from being used as features
- Watch out for subgroup bias
- Use privacy preserving ML techniques
  - E.g., federated learning
Trustworthy ML

• Assure application security
  • Data
  • Manipulation of system behaviour
Trustworthy ML

• Assure application security
  • Data
  • Manipulation of system behaviour

• Perform risk assessments
Trustworthy ML

• Assure application security
  • Data
  • Manipulation of system behaviour

• Perform risk assessments

• Provide audit trails
Trustworthy ML

• Assure application security
  • Data
  • Manipulation of system behaviour

• Perform risk assessments

• Provide audit trails

• Have your application audited
Trustworthy ML

- Establish responsible AI values
  - What does it mean to be responsible?
Trustworthy ML

• Establish responsible AI values
  • What does it mean to be responsible?

• Use interpretable models whenever possible
Trustworthy ML

• Establish responsible AI values
  • What does it mean to be responsible?

• Use interpretable models whenever possible

• Use team process for decision making
  • Higher accuracy ‘blackbox’ model or slightly less accurate interpretable model?
  • E.g., When do we accept a blackbox model?
Trustworthy ML

- Inform users about ML usage
Trustworthy ML

• Inform users about ML usage

• Provide safe channels to raise concerns
Trustworthy ML

• Inform users about ML usage

• Provide safe channels to raise concerns

• Explain results and decisions to users
Many practices, where to start?

• One step at a time
Many practices, where to start?

• One step at a time

• Aim: Rank by difficulty (work in progress)

| Use Versioning for Data, Model, Configurations and Training Scripts | Training | basic |
| Share Status and Outcomes of Experiments Within the Team | Training | basic |
| Run Automated Regression Tests | Coding | advanced |
| Use Continuous Integration | Coding | medium |
| Use Static Analysis to Check Code Quality | Coding | advanced |
Summary

• SE for ML is still SE
  • Don’t abandon best practices
  • Adapt and extend them

• AutoML to maximise performance

• Build a system we can trust

• Learn more: https://se-ml.github.io