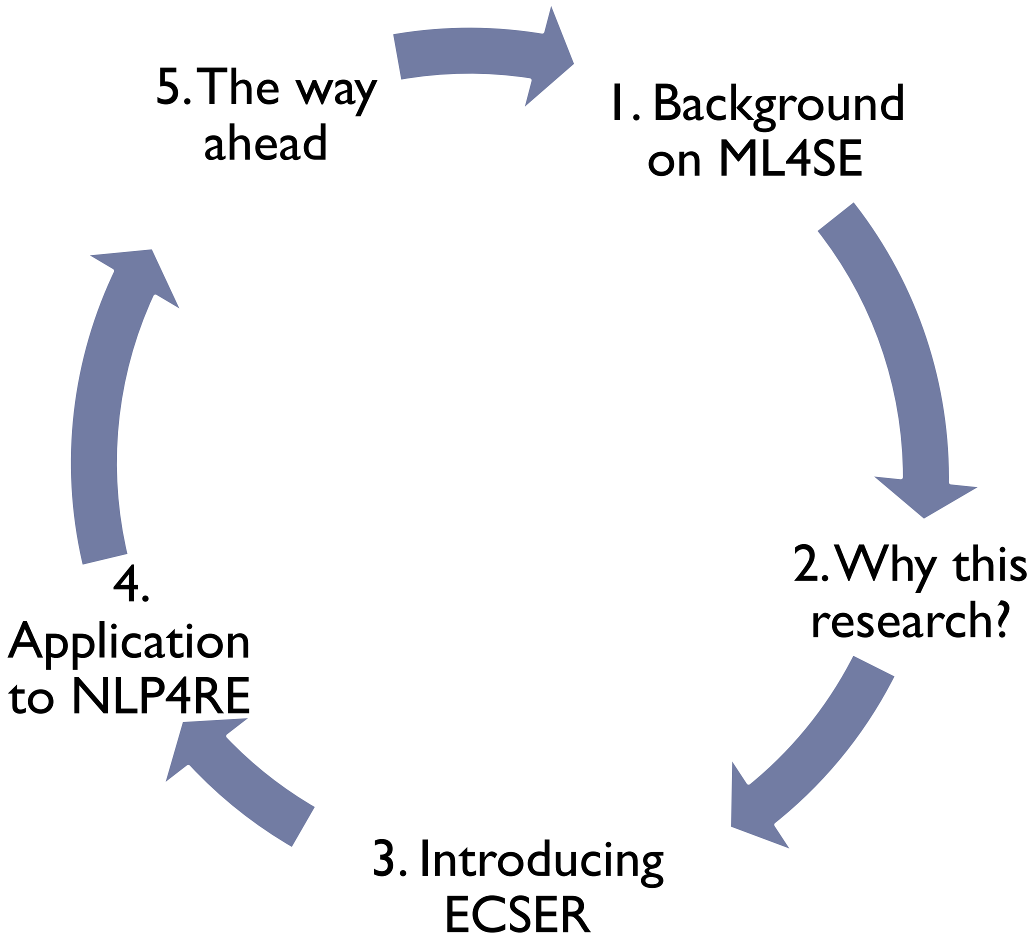
On the quest for more credible results in ML4SE research

Fabiano Dalpiaz
Requirements Engineering Lab
Utrecht University, the Netherlands
April 17, 2023

f.dalpiaz@uu.nl @FabianoDalpiaz

Outline and Acks



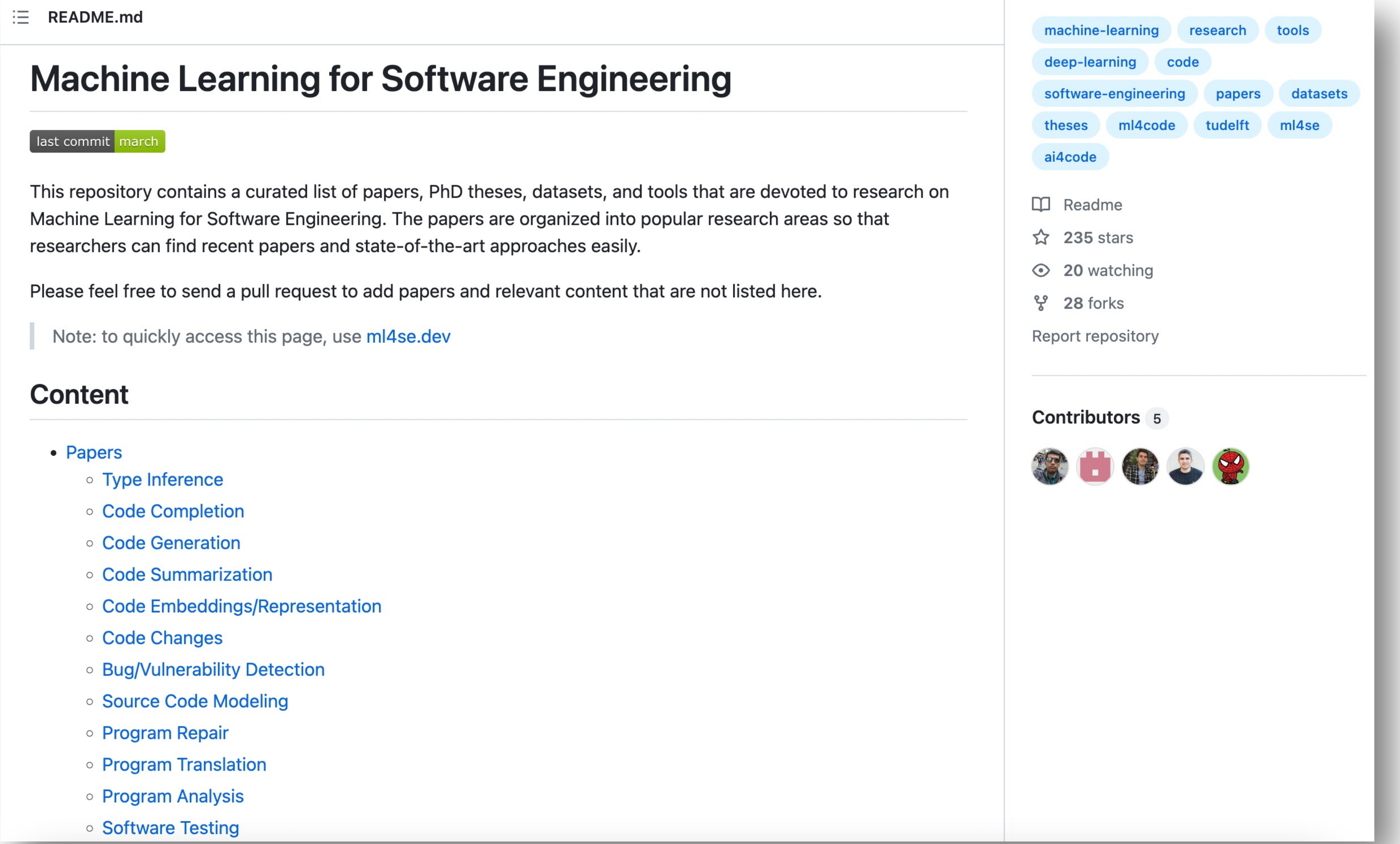
Dr. Davide Dell'Anna
Utrecht University
Netherlands

Dr. F. Başak Aydemir
Boğaziçi University
Turkey

Davide Dell'Anna, Fatma Basak Aydemir, Fabiano Dalpiaz: *Evaluating classifiers in SE research: the ECSER pipeline and two replication studies*. Empir. Softw. Eng. 28(1): 3 (2023)

1. Background on ML4SE

ML4SE research is (becoming) pervasive



README.md

Machine Learning for Software Engineering

last commit march

This repository contains a curated list of papers, PhD theses, datasets, and tools that are devoted to research on Machine Learning for Software Engineering. The papers are organized into popular research areas so that researchers can find recent papers and state-of-the-art approaches easily.

Please feel free to send a pull request to add papers and relevant content that are not listed here.

> Note: to quickly access this page, use ml4se.dev

Content

- Papers
  - Type Inference
  - Code Completion
  - Code Generation
  - Code Summarization
  - Code Embeddings/Representation
  - Code Changes
  - Bug/Vulnerability Detection
  - Source Code Modeling
  - Program Repair
  - Program Translation
  - Program Analysis
  - Software Testing

machine-learning research tools deep-learning code software-engineering papers datasets theses ml4code tudelft ml4se ai4code

Readme
235 stars
20 watching
28 forks
Report repository

Contributors 5

https://github.com/saltudelft/ml4se

- ML in the broad sense includes Neural Network architectures such as BERT, **GPTs**, …
- **Hundreds** of papers and tools

ML4SE research is (becoming) pervasive

README.md

Machine Learning for Software Engineering

last commit march

This repository contains a curated list of papers, PhD theses, datasets, and tools that are devoted to research on Machine Learning for Software Engineering. The papers are organized into popular research areas so that researchers can find recent papers and state-of-the-art approaches easily.

Please feel free to send a pull request to add papers and relevant content that are not listed here.

> Note: to quickly access this page, use ml4se.dev

Content

- Papers
  - Type Inference
  - Code Completion
  - Code Generation
  - Code Summarization
  - Code Embeddings/Representation
  - Code Changes
  - Bug/Vulnerability Detection
  - Source Code Modeling
  - Program Repair
  - Program Translation
  - Program Analysis
  - Software Testing

machine-learning research tools deep-learning code software-engineering papers datasets theses ml4code tudelft ml4se ai4code

Readme
235 stars
20 watching
28 forks
Report repository

Contributors 5

https://github.com/saltudelft/ml4se

- ML in the broad sense includes Neural Network architectures such as BERT, **GPTs**, …
- **Hundreds** of papers and tools
- Applied (hammer?) to **many SE problems**

Zoom-in on classification (roughly, supervised ML)

- Given
  - A set of **labels**
    - E.g., bug and feature request
  - A **labelled dataset**
    - E.g., a collection of user reviews each representing a bug, a feature request, both, or neither

Zoom-in on classification (roughly, supervised ML)

- Given
  - A set of **labels**
    - E.g., bug and feature request
  - A **labelled dataset**
    - E.g., a collection of user reviews each representing a bug, a feature request, both, or neither
- A classification algorithm builds a **model** that

Zoom-in on classification (roughly, supervised ML)

- Given
  - A set of **labels**
    - E.g., bug and feature request
  - A **labelled dataset**
    - E.g., a collection of user reviews each representing a bug, a feature request, both, or neither
- A classification algorithm builds a **model** that
  - describes the labelled dataset as accurately as possible

Zoom-in on classification (roughly, supervised ML)

- Given
  - A set of **labels**
    - E.g., bug and feature request
  - A **labelled dataset**
    - E.g., a collection of user reviews each representing a bug, a feature request, both, or neither
- A classification algorithm builds a **model** that
  - describes the labelled dataset as accurately as possible
  - **predicts accurately the labels of unseen datasets**

Zoom-in on classification (roughly, supervised ML)

- Given
  - A set of **labels**
    - E.g., bug and feature request
  - A **labelled dataset**
    - E.g., a collection of user reviews each representing a bug, a feature request, both, or neither
- A classification algorithm builds a **model** that
  - describes the labelled dataset as accurately as possible
  - **predicts accurately the labels of unseen datasets**

Table 2: Summary of the exploratory mapping study of the proceedings of the ICSE conference from the year 2019 through 2021.

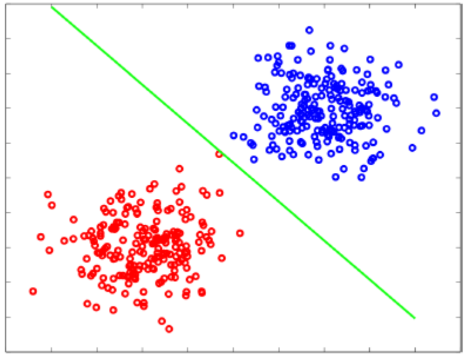
Year	2019		2020		2021		Total	
Accepted ICSE papers (Main track)	109		129		138		376	
Papers related to classification	19	(17.43%)	14	(10.85%)	27	(19.57%)	60	(15.96%)

An example of classification in NLP4RE

		Class
Requirement	Features	Functional
… *print a report* …	…	Yes
… *save the page* …	…	Yes
… *every three days* …	…	No
… *refresh the display* …	…	?

Train

Model



Predict

Zoom-in on classification, reprise

- Given
  - A set of **labels**
    - E.g., bug and feature request
  - A **labelled dataset**
    - E.g., a collection of user reviews each representing a bug, a feature request, both, or neither
- A classification algorithm builds a **model** that
  - describes the labelled dataset as accurately as possible
  - **predicts accurately the labels of unseen datasets**

Zoom-in on classification, reprise

- Given
  - A set of **labels**
    - E.g., bug and feature request
  - A **labelled dataset**
    - E.g., a collection of user reviews each representing a bug, a feature request, both, or neither
- A classification algorithm builds a **model** that
  - describes the labelled dataset as accurately as possible
  - **is expected to predict accurately the labels of unseen datasets**

Classification research in NLP4RE

Tool Type	Tool Name (Study ID)	No. Tools	Percent
Modeling	OICSI (S678), NL-OOPS (S553), EA-Miner (S499), CM-Builder (S343), Circe (S34), LIDA (S623), NIBA Toolset (S272), RETNA (S108), aToucan (S909), DBDT (S31), Cico (S34), NL2UMLviaSBVR (S70), RADD-NLI (S121), SUGAR (S190), GRACE (S208), AREMCD (S219), RUCM (S227), RSLingo (S266), Zen-ReqConfig (S482), TREx (S496), NAPLES (S499), GeNLangUML (S551), ConstraintSoup (S600), C&L (S707), AnModeler (S799), SBEAVER (S813), KCMP Dynamisch (S272), Xtext (S20), Kheops (S35), Visual Narrator (S683), ProcGap (S800), FeatureX (S772), CMT & FDE (S261), VoiceToModel (S765)	34	26.15%
Detection	ARM (S861), SREE (S812), RQA (S903), AnaCon (S41), REGICE (S55), NARCIA (S56), LELIE (S75), SRRDirector (S86), MIA (S114), KROSA (S178), NAI (S226), QuARS (S232), CAR (S252), CARL (S298), RAVEN (S303), ReqSAC (S370), RAT (S376), MaramaAIC (S395), RESI (S432), RECAA (S447), DeNom (S448), RETA (S450), AQUSA (S501), Dowser (S644), QAMiner (S661), LeCA (S701), S-HTC (S258), CNLP(S464), Pragmatic Ambiguity Detector (S256), ReqAligner (S663), REAssistant (S662)	31	23.85%
Extraction	findphrases (S13), AbstFinder (S307), FENL (S71), NAT2TESTSCR (S131), NLP-KAOS (S132), SAFE (S385), AUTOANNOTATOR (S433), UCTD (S453), GUEST (S598), Guidance Tool (S688), SpecQua (S743), NAT2TEST (S744), semMet (S777), Test2UseCase (S810), OCLgen (S845), Text2Policy (S872), GaiusT (S888), SNACC (S891), Doc2Spec (S897), ARSENAL (S915), MaTREx tool (S284), ELICA (S2), CHOReOS (S520), GuideGen (S907)	24	18.46%
Classification	ASUM (S129), RUBRIC (S223), WCC (S257), NFR2AC tool (S306), ALERTme (S332), PUMConf (337), FFRE (S341), AUR-BoW (S500), SEMIOS (S550), CRISTAL (S629), CoReq (S672), SD (S674), ACRE (S757), SOVA R-TC (S778), SMAA (S788), CSLabel (S892), HeRA (S718), NFR Locator (S758), SURF (S910), NFRFinder (S647)	20	15.38%
Tracing & Relating	Coparvo (S24), Trustrace (S25), Histrace (S25), CoChaIR (S26), HYPERDOCSY (S38), ReqSimile (S171), LGRTL (S198), CQV-UML (S400), TiQi (S651), REVERE (S717), LiMonE (S723), ESPRET (S792), COCAR (S805), RETRO (S934), WATson (S302)	15	11.54%
Search & Retrieval	RE-SWOT (S174), IntelliReq (S602), ReqWiki (S711), iMapper (S784), PriF (S802), WIKINA (S686)	6	4.62%
Total		**130**	**100%**

Liping Zhao, Waad Alhoshan, Alessio Ferrari, Keletso J. Letsholo, Muideen A. Ajagbe, Erol-Valeriu Chioasca, and Riza T. Batista-Navarro. *Natural Language Processing (NLP) for Requirements Engineering: A Systematic Mapping Study.* ACM Computing Surveys 54:3, 2022

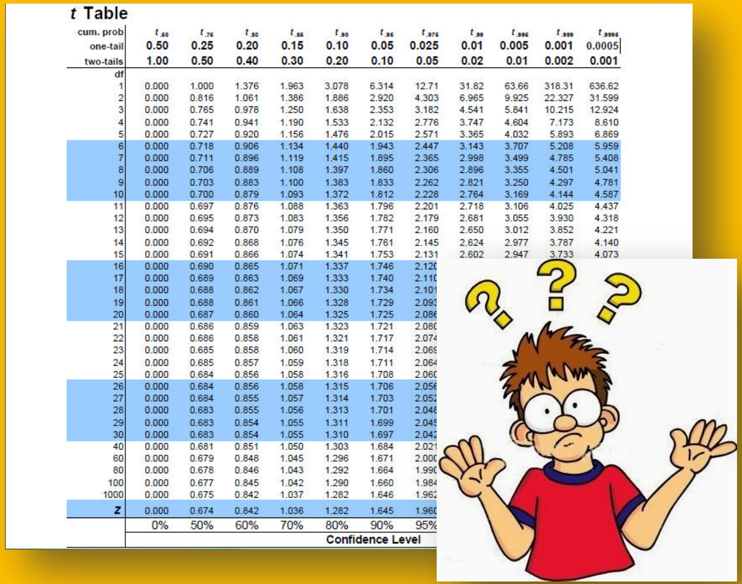
Classification research in NLP4RE

Tool Type	Tool Name (Study ID)	No. Tools	Percent
Modeling	OICSI (S678), NL-OOPS (S553), EA-Miner (S499), CM-Builder (S343), Circe (S34), LIDA (S623), NIBA Toolset (S272), RETNA (S108), aToucan (S909), DBDT (S31), Cico (S34), NL2UMLviaSBVR (S70), RADD-NLI (S121), SUGAR (S190), GRACE (S208), AREMCD (S219), RUCM (S227), RSLingo (S266), Zen-ReqConfig (S482), TREx (S496), NAPLES (S499), GeNLangUML (S551), ConstraintSoup (S600), C&L (S707), AnModeler (S799), SBEAVER (S813), KCMP Dynamisch (S272), Xtext (S20), Kheops (S35), Visual Narrator (S683), ProcGap (S800), FeatureX (S772), CMT & FDE (S261), VoiceToModel (S765)	34	26.15%
Detection	ARM (S861), SREE (S812), RQA (S903), AnaCon (S41), REGICE (S55), NARCIA (S56), LELIE (S75), SRRDirector (S86), MIA (S114), KROSA (S178), NAI (S226), QuARS (S232), CAR (S252), CARL (S298), RAVEN (S303), ReqSAC (S370), RAT (S376), MaramaAIC (S395), RESI (S432), RECAA (S447), DeNom (S448), RETA (S450), AQUSA (S501), Dowser (S644), QAMiner (S661), LeCA (S701), S-HTC (S258), CNLP(S464), Pragmatic Ambiguity Detector (S256), ReqAligner (S663), REAssistant (S662)	31	23.85%
Extraction	findphrases (S13), AbstFinder (S307), FENL (S71), NAT2TESTSCR (S131), NLP-KAOS (S132), SAFE (S385), AUTOANNOTATOR (S433), UCTD (S453), GUEST (S598), Guidance Tool (S688), SpecQua (S743), NAT2TEST (S744), semMet (S777), Test2UseCase (S810), OCLgen (S845), Text2Policy (S872), GaiusT (S888), SNACC (S891), Doc2Spec (S897), ARSENAL (S915), MaTREx tool (S284), ELICA (S2), CHOReOS (S520), GuideGen (S907)	24	18.46%
Classification	ASUM (S129), RUBRIC (S223), WCC (S257), NFR2AC tool (S306), ALERTme (S332), PUMConf (337), FFRE (S341), AUR-BoW (S500), SEMIOS (S550), CRISTAL (S629), CoReq (S672), SD (S674), ACRE (S757), SOVA R-TC (S778), SMAA (S788), CSLabel (S892), HeRA (S718), NFR Locator (S758), SURF (S910), NFRFinder (S647)	20	15.38%
Tracing & Relating	Coparvo (S24), Trustrace (S25), Histrace (S25), CoChaIR (S26), HYPERDOCSY (S38), ReqSimile (S171), LGRTL (S198), CQV-UML (S400), TiQi (S651), REVERE (S717), LiMonE (S723), ESPRET (S792), COCAR (S805), RETRO (S934), WATson (S302)	15	11.54%
Search & Retrieval	RE-SWOT (S174), IntelliReq (S602), ReqWiki (S711), iMapper (S784), PriF (S802), WIKINA (S686)	6	4.62%
Total		**130**	**100%**

Classification algorithms are used **not only by "requirements. classification" tools**, but also for tracing, defect detection, …

Liping Zhao, Waad Alhoshan, Alessio Ferrari, Keletso J. Letsholo, Muideen A. Ajagbe, Erol-Valeriu Chioasca, and Riza T. Batista-Navarro. *Natural Language Processing (NLP) for Requirements Engineering: A Systematic Mapping Study.* ACM Computing Surveys 54:3, 2022

2. Why this research?



t Table

cum. prob	$t_{.50}$	$t_{.75}$	$t_{.80}$	$t_{.85}$	$t_{.90}$	$t_{.95}$	$t_{.975}$	$t_{.99}$	$t_{.995}$	$t_{.999}$	$t_{.9995}$
one-tail	0.50	0.25	0.20	0.15	0.10	0.05	0.025	0.01	0.005	0.001	0.0005
two-tails	1.00	0.50	0.40	0.30	0.20	0.10	0.05	0.02	0.01	0.002	0.001
df											
1	0.000	1.000	1.376	1.963	3.078	6.314	12.71	31.82	63.66	318.31	636.62
2	0.000	0.816	1.061	1.386	1.886	2.920	4.303	6.965	9.925	22.327	31.599
3	0.000	0.765	0.978	1.250	1.638	2.353	3.182	4.541	5.841	10.215	12.924
4	0.000	0.741	0.941	1.190	1.533	2.132	2.776	3.747	4.604	7.173	8.610
5	0.000	0.727	0.920	1.156	1.476	2.015	2.571	3.365	4.032	5.893	6.869
6	0.000	0.718	0.906	1.134	1.440	1.943	2.447	3.143	3.707	5.208	5.959
7	0.000	0.711	0.896	1.119	1.415	1.895	2.365	2.998	3.499	4.785	5.408
8	0.000	0.706	0.889	1.108	1.397	1.860	2.306	2.896	3.355	4.501	5.041
9	0.000	0.703	0.883	1.100	1.383	1.833	2.262	2.821	3.250	4.297	4.781
10	0.000	0.700	0.879	1.093	1.372	1.812	2.228	2.764	3.169	4.144	4.587
11	0.000	0.697	0.876	1.088	1.363	1.796	2.201	2.718	3.106	4.025	4.437
12	0.000	0.695	0.873	1.083	1.356	1.782	2.179	2.681	3.055	3.930	4.318
13	0.000	0.694	0.870	1.079	1.350	1.771	2.160	2.650	3.012	3.852	4.221
14	0.000	0.692	0.868	1.076	1.345	1.761	2.145	2.624	2.977	3.787	4.140
15	0.000	0.691	0.866	1.074	1.341	1.753	2.131	2.602	2.947	3.733	4.073
16	0.000	0.690	0.865	1.071	1.337	1.746	2.120				
17	0.000	0.689	0.863	1.069	1.333	1.740	2.110				
18	0.000	0.688	0.862	1.067	1.330	1.734	2.101				
19	0.000	0.688	0.861	1.066	1.328	1.729	2.093				
20	0.000	0.687	0.860	1.064	1.325	1.725	2.086				
21	0.000	0.686	0.859	1.063	1.323	1.721	2.080				
22	0.000	0.686	0.858	1.061	1.321	1.717	2.074				
23	0.000	0.685	0.858	1.060	1.319	1.714	2.069				
24	0.000	0.685	0.857	1.059	1.318	1.711	2.064				
25	0.000	0.684	0.856	1.058	1.316	1.708	2.060				
26	0.000	0.684	0.856	1.058	1.315	1.706	2.056				
27	0.000	0.684	0.855	1.057	1.314	1.703	2.052				
28	0.000	0.683	0.855	1.056	1.313	1.701	2.048				
29	0.000	0.683	0.854	1.055	1.311	1.699	2.045				
30	0.000	0.683	0.854	1.055	1.310	1.697	2.042				
40	0.000	0.681	0.851	1.050	1.303	1.684	2.021				
60	0.000	0.679	0.848	1.045	1.296	1.671	2.000				
80	0.000	0.678	0.846	1.043	1.292	1.664	1.990				
100	0.000	0.677	0.845	1.042	1.290	1.660	1.984				
1000	0.000	0.675	0.842	1.037	1.282	1.646	1.962				
z	0.000	0.674	0.842	1.036	1.282	1.645	1.960				
	0%	50%	60%	70%	80%	90%	95%				
	Confidence Level										

Credible research?

- **Iris** is a requirements analyst who **wants to categorize** a large **collection of requirements** from their company

Credible research?

- **Iris** is a requirements analyst who **wants to categorize** a large **collection of requirements** from their company
- Iris comes across the following results from a prominent paper

Test set	**F**				**Q**			
	Prec	Rec	F1	AUC	Prec	Rec	F1	AUC
PROMISE train	0.981	0.984	0.982	1.00	0.985	1.000	0.990	1.00
PROMISE test	0.819	0.797	0.822	0.89	0.909	0.891	0.873	0.92

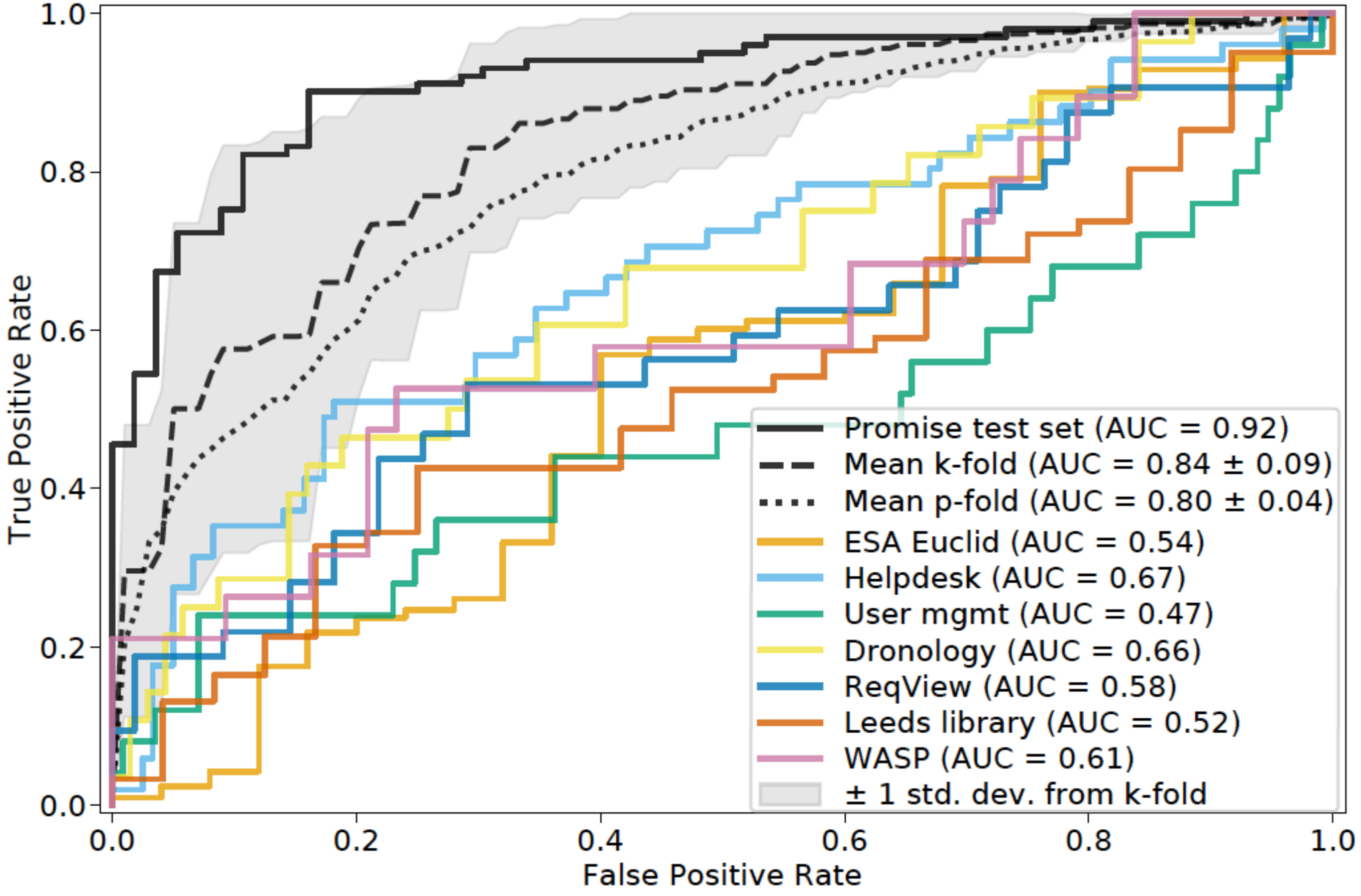
Credible research?

- **Iris** is a requirements analyst who **wants to categorize** a large **collection of requirements** from their company
- Iris comes across the following results from a prominent paper

Test set	F				Q			
	Prec	Rec	F1	AUC	Prec	Rec	F1	AUC
PROMISE train	0.981	0.984	0.982	1.00	0.985	1.000	0.990	1.00
PROMISE test	0.819	0.797	0.822	0.89	0.909	0.891	0.873	0.92

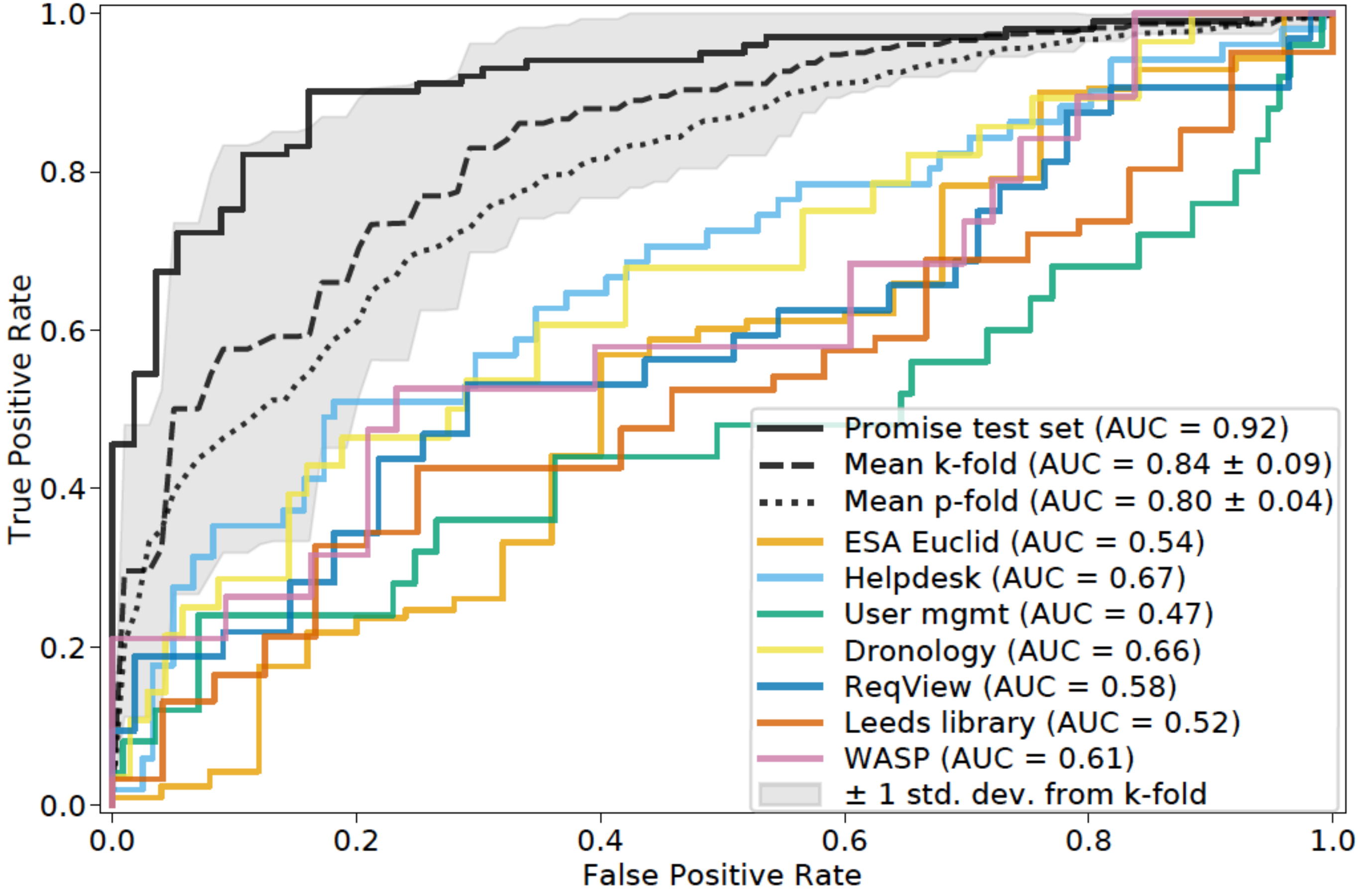
Can Iris trust that similar performance will be obtained on the company's dataset?

Credible research? Under certain assumptions



Fabiano Dalpiaz, Davide Dell'Anna , Fatma Basak Aydemir, Sercan Çevikol: *Requirements Classification with Interpretable Machine Learning and Dependency Parsing*. RE 2019: 142-152

Credible research? Under certain assumptions



Fabiano Dalpiaz, Davide Dell'Anna , Fatma Basak Aydemir, Sercan Çevikol: *Requirements Classification with Interpretable Machine Learning and Dependency Parsing*. RE 2019: 142-152

Does the dataset resemble PROMISE NFR?

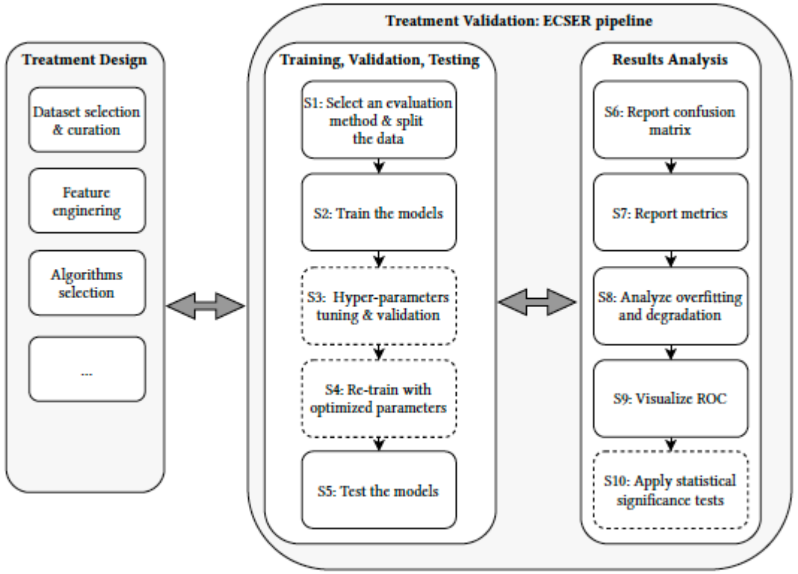
- ✅ Maybe the result can be transferred
- ❌ Iris may need to re-train the classifier, perhaps by labeling hundreds of reqs.

Research goal: toward credible results in ML4SE

- We aim to provide researchers with a **framework that enables and fosters publishing (more) credible results**

Research goal: toward credible results in ML4SE

- We aim to provide researchers with a **framework that enables and fosters publishing (more) credible results**
- ECSER pipeline: **Evaluating Classifiers in Software Engineering Research**



Why do we need ECSER?

- Bottom line: **we do not want to blame researchers**!
  - Our team made and still makes mistakes when reporting results

Why do we need ECSER?

- Bottom line: **we do not want to blame researchers**!
  - Our team made and still makes mistakes when reporting results
- So, why ECSER?
  - ML libraries, code snippets, ChatGPT make **ML accessible to non-experts**

Why do we need ECSER?

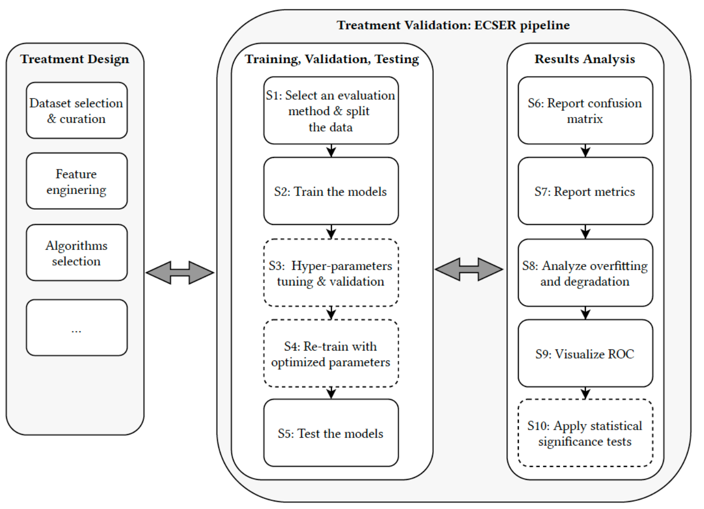
- Bottom line: **we do not want to blame researchers**!
  - Our team made and still makes mistakes when reporting results
- So, why ECSER?
  - ML libraries, code snippets, ChatGPT make **ML accessible to non-experts**
  - **Performance metrics** are often **chosen based on previous work**

Why do we need ECSER?

- Bottom line: **we do not want to blame researchers**!
  - Our team made and still makes mistakes when reporting results

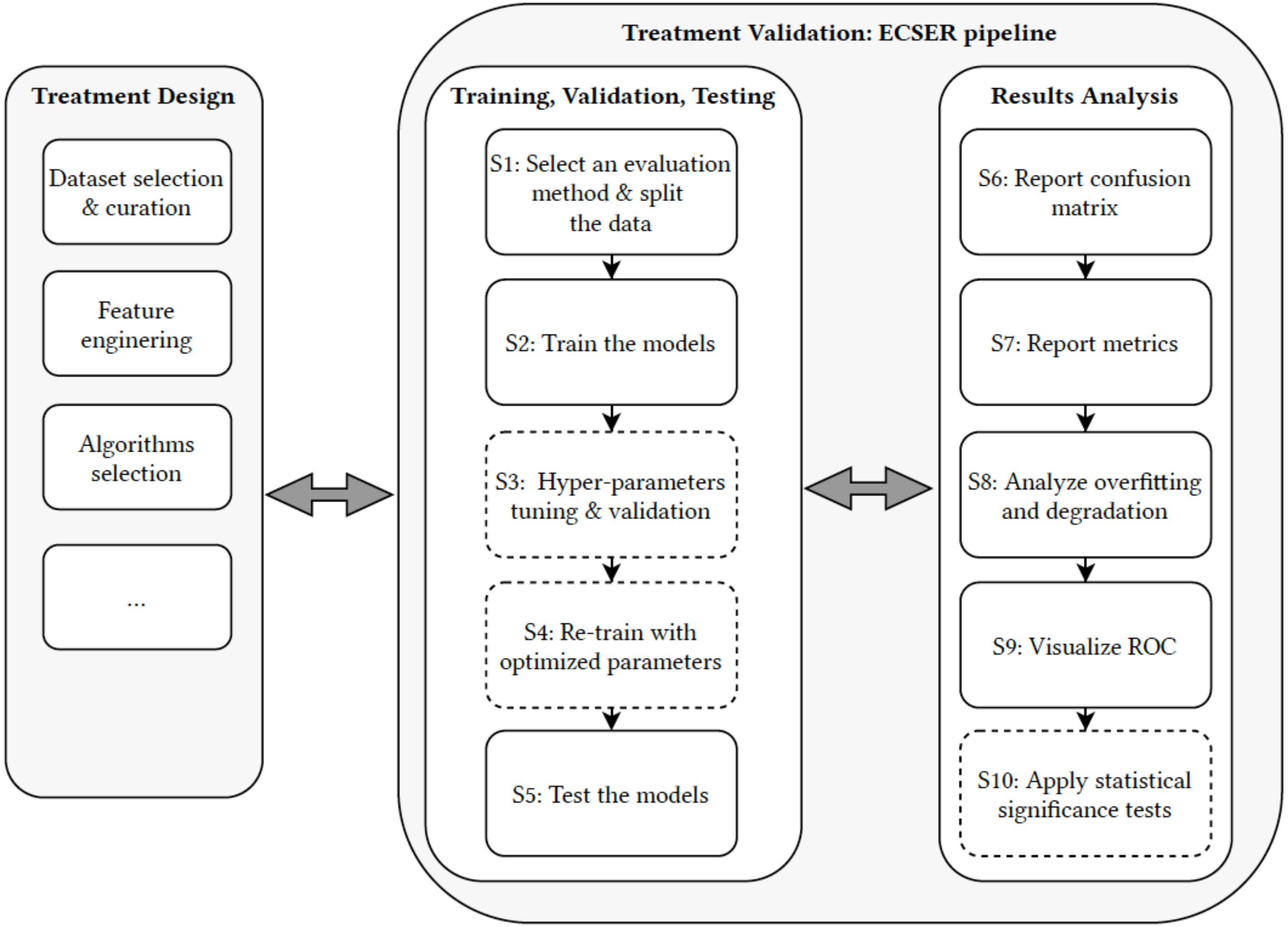
- So, why ECSER?
  - ML libraries, code snippets, ChatGPT make **ML accessible to non-experts**
  - **Performance metrics** are often **chosen based on previous work**
  - **Statistical analysis** on multiple datasets is **still rare**

3. Introducing ECSER



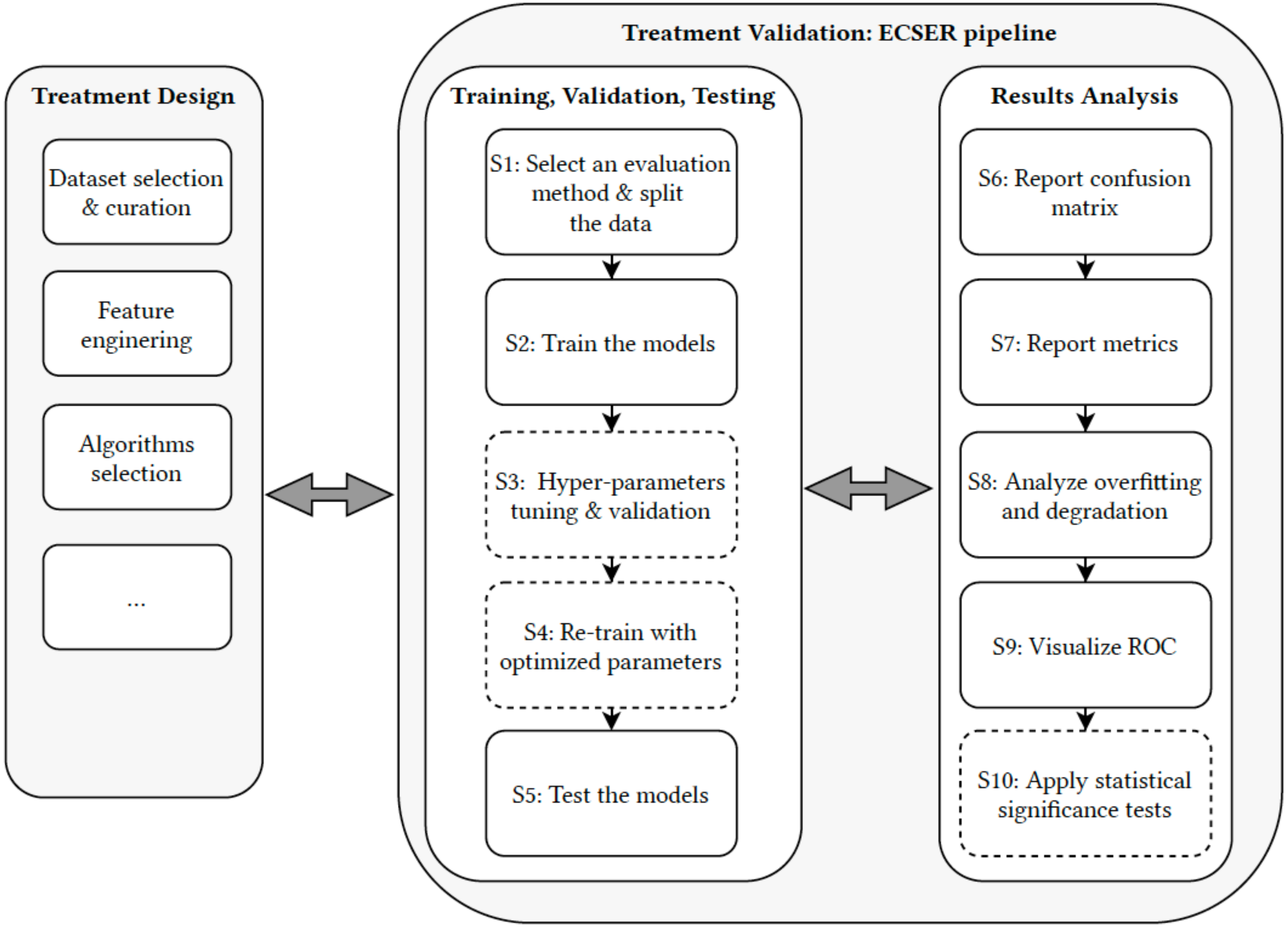
ECSER: an overview

- **ECSER focuses on Treatment Validation**
  - Treatment = a classifier
  - Two macro phases
  - Iterative, as typical in ML



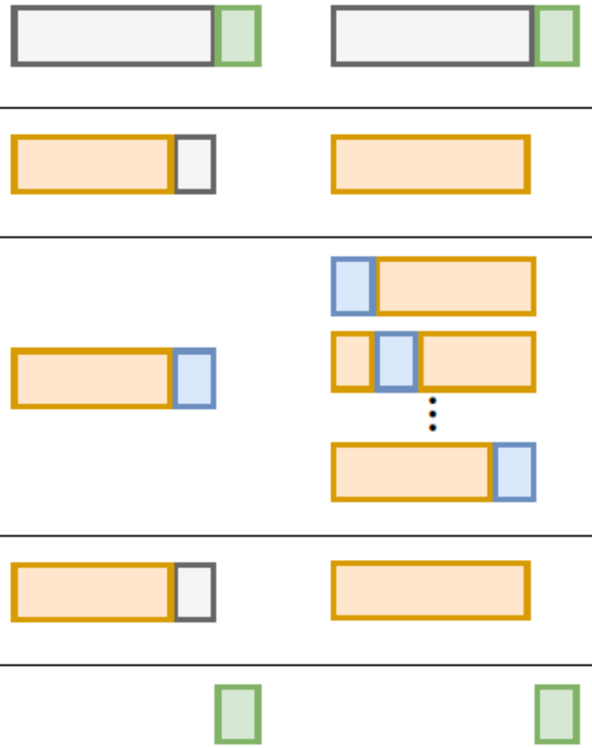
ECSER: an overview

- **ECSER focuses on Treatment Validation**
  - Treatment = a classifier
  - Two macro phases
  - Iterative, as typical in ML
- Treatment design is **outside the scope** of ECSER
  - Dataset selection & curation
  - Feature engineering
  - Algorithms selection



ECSER's highlight #1: data and models

Step	Classification Model	Holdout	X-Val
S1: Select an evaluation method & split the data	None: the test set is extracted for use in S4		
S2: Train the models	Fit non-test set with default hyper-parameters		
S3: Hyper-parameters tuning & validation	Search hyper-parameters that predict the validation set best		
S4: Re-train with optimized parameters	Fit non-test set with optimal hyper-parameters from S3		
S5: Test the models	Model from S4		



ECSER's highlight #2: p-fold cross-validation

- In SE, data originates from different projects
- p-fold cross-validation extends k-fold cross-validation with **per-project splits** (as opposed to random splits)

ECSER's highlight #2: p-fold cross-validation

- In SE, data originates from different projects
- p-fold cross-validation extends k-fold cross-validation with **per-project splits** (as opposed to random splits)
  1. Given a set P of projects, take a subset $S \subset P$ to train the classifier
  2. Test the classifier on the remaining $P \setminus S$

ECSER's highlight #2: p-fold cross-validation

- In SE, data originates from different projects
- p-fold cross-validation extends k-fold cross-validation with **per-project splits** (as opposed to random splits)
  1. Given a set P of projects, take a subset S⊂P to train the classifier
  2. Test the classifier on the remaining P \ S
  3. Take another subset S' of the same size of S
  4. Train the classifier on S'
  5. …

ECSER's highlight #2: p-fold cross-validation

- In SE, data originates from different projects
- p-fold cross-validation extends k-fold cross-validation with **per-project splits** (as opposed to random splits)
  1. Given a set P of projects, take a subset S⊂P to train the classifier
  2. Test the classifier on the remaining P \ S
  3. Take another subset S' of the same size of S
  4. Train the classifier on S'
  5. …

p-fold generally introduces more diversity than k-fold

Test set	F				Q			
	Prec	Rec	F1	AUC	Prec	Rec	F1	AUC
PROMISE train	0.981	0.984	0.982	1.00	0.985	1.000	0.990	1.00
PROMISE test	0.819	0.797	0.822	0.89	0.909	0.891	0.873	0.92
PROMISE k-fold	0.755	0.684	0.712	0.80	0.785	0.867	0.822	0.84
PROMISE p-fold	0.749	0.602	0.663	0.78	0.714	0.877	0.781	0.80

ECSER’s highlight #3: the confusion matrix

- Reporting on the confusion matrix provides transparency as it allows to derive all metrics and to easily inspect the results

$$
\begin{array}{cc}
 & \text{Predicted} \\
\begin{array}{c} \\ \end{array} & \begin{array}{cc} \text{Positive} & \text{Negative} \end{array} \\
\rotatebox{90}{\text{Actual}} \begin{array}{c} \text{Positive} \\ \text{Negative} \end{array} & \begin{bmatrix} \text{TP} & \text{FN} \\ \text{FP} & \text{TN} \end{bmatrix}
\end{array}
\qquad
\begin{array}{cc}
 & \text{Predicted} \\
\begin{array}{c} \\ \end{array} & \begin{array}{cc} \text{Positive} & \text{Negative} \end{array} \\
\rotatebox{90}{\text{Actual}} \begin{array}{c} \text{Positive} \\ \text{Negative} \end{array} & \begin{bmatrix} x & 0 \\ 0 & |D| - x \end{bmatrix}
\end{array}
$$

ECSER's highlight #3: the confusion matrix

- Reporting on the confusion matrix provides transparency as it allows to derive all metrics and to easily inspect the results

$$
\begin{array}{ccc}
 & & \text{Predicted} \\
 & & \begin{array}{cc}\text{Positive} & \text{Negative}\end{array} \\
\text{Actual} & \begin{array}{c}\text{Positive} \\ \text{Negative}\end{array} & \begin{bmatrix} \text{TP} & \text{FN} \\ \text{FP} & \text{TN} \end{bmatrix}
\end{array}
\qquad
\begin{array}{ccc}
 & & \text{Predicted} \\
 & & \begin{array}{cc}\text{Positive} & \text{Negative}\end{array} \\
\text{Actual} & \begin{array}{c}\text{Positive} \\ \text{Negative}\end{array} & \begin{bmatrix} x & 0 \\ 0 & |D| - x \end{bmatrix}
\end{array}
$$

Metric	Formula
Precision	$TP/(TP + FP)$
Recall (TPR)	$TP/(TP + FN)$
Specificity (TNR)	$TN/(TN + FP)$
Accuracy	$(TP + TN)/(TP + TN + FP + FN)$
F_1-score	$2 \cdot (Precision \cdot Recall)/(Precision + Recall)$
F_β-score	$(1+\beta^2)(Precision \cdot Recall)/((\beta^2 \cdot Precision) + Recall)$

ECSER's highlight #3: the confusion matrix

- Reporting on the confusion matrix provides transparency as it allows to derive all metrics and to easily inspect the results

		Gold Standard				
		None	Feature	Stability	Performance	Quality
Crowd	None	67	5	3	3	14
	Feature	4	94	1	1	2
	Stability	14	8	134	6	20
	Performance	4	5	3	29	19
	Quality	28	1	3	7	208

Martijn van Vliet, Eduard C. Groen, Fabiano Dalpiaz, Sjaak Brinkkemper: *Identifying and Classifying User Requirements in Online Feedback via Crowdsourcing*. REFSQ 2020: 143-159

ECSER's highlight #4: overfitting and degradation

- We suggest two specific metrics to better analyze performance

training set

validation set

test set

Step	Classification Model	Holdout	X-Val
S1	None: the test set is extracted for use in S4		
S2	Fit non-test set with default hyper-parameters		
S3	Search hyper-parameters that predict the validation set best		
S4	Fit non-test set with optimal hyper-parameters from S3		
S5	Model from S4		

ECSER's highlight #4: overfitting and degradation

- We suggest two specific metrics to better analyze performance

Overfitting = Test – Training

training set

validation set

test set

Step	Classification Model	Holdout	X-Val
S1	None: the test set is extracted for use in S4		
S2	Fit non-test set with default hyper-parameters		
S3	Search hyper-parameters that predict the validation set best		⋮
S4	Fit non-test set with optimal hyper-parameters from S3		
S5	Model from S4		

ECSER's highlight #4: overfitting and degradation

- We suggest two specific metrics to better analyze performance

Overfitting = Test – Training

Degradation = Test – Validation

training set

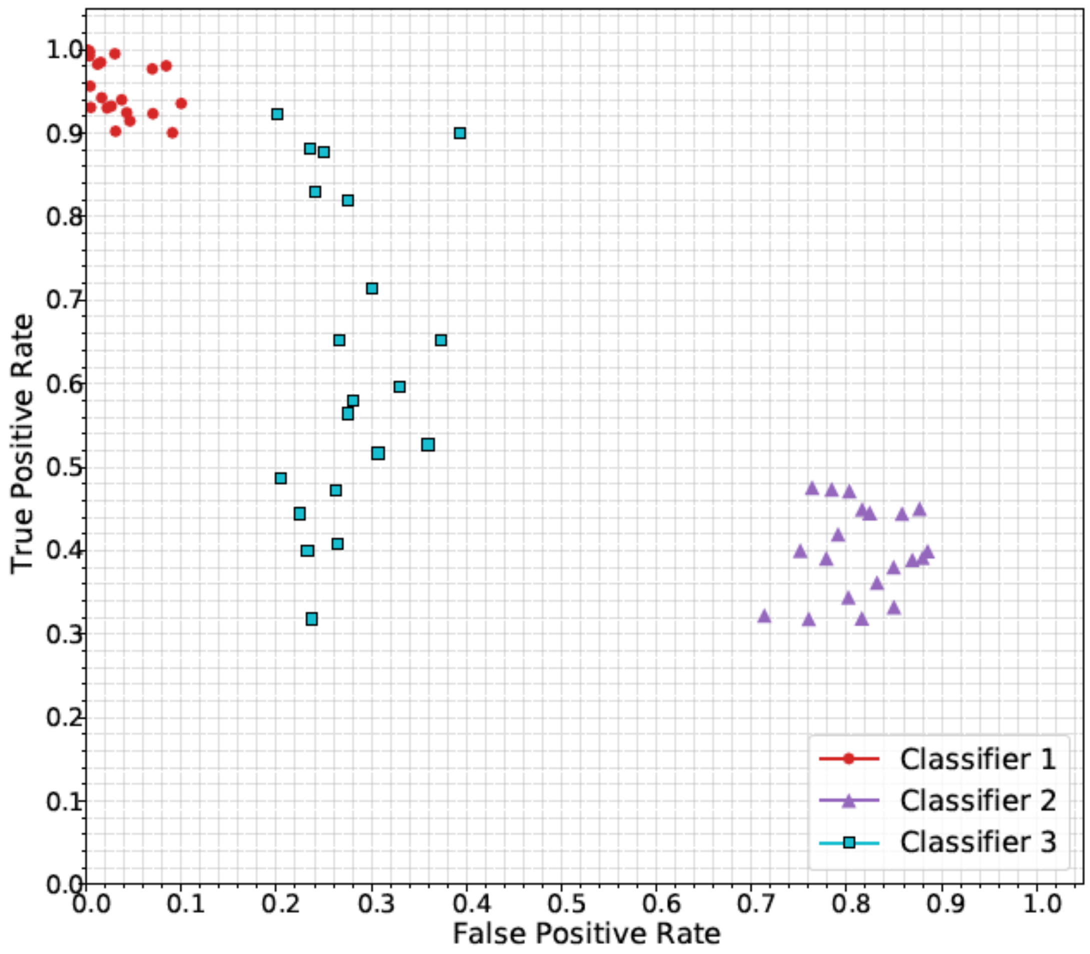
validation set

test set

Step	Classification Model	Holdout	X-Val
S1	None: the test set is extracted for use in S4		
S2	Fit non-test set with default hyper-parameters		
S3	Search hyper-parameters that predict the validation set best		
S4	Fit non-test set with optimal hyper-parameters from S3		
S5	Model from S4		

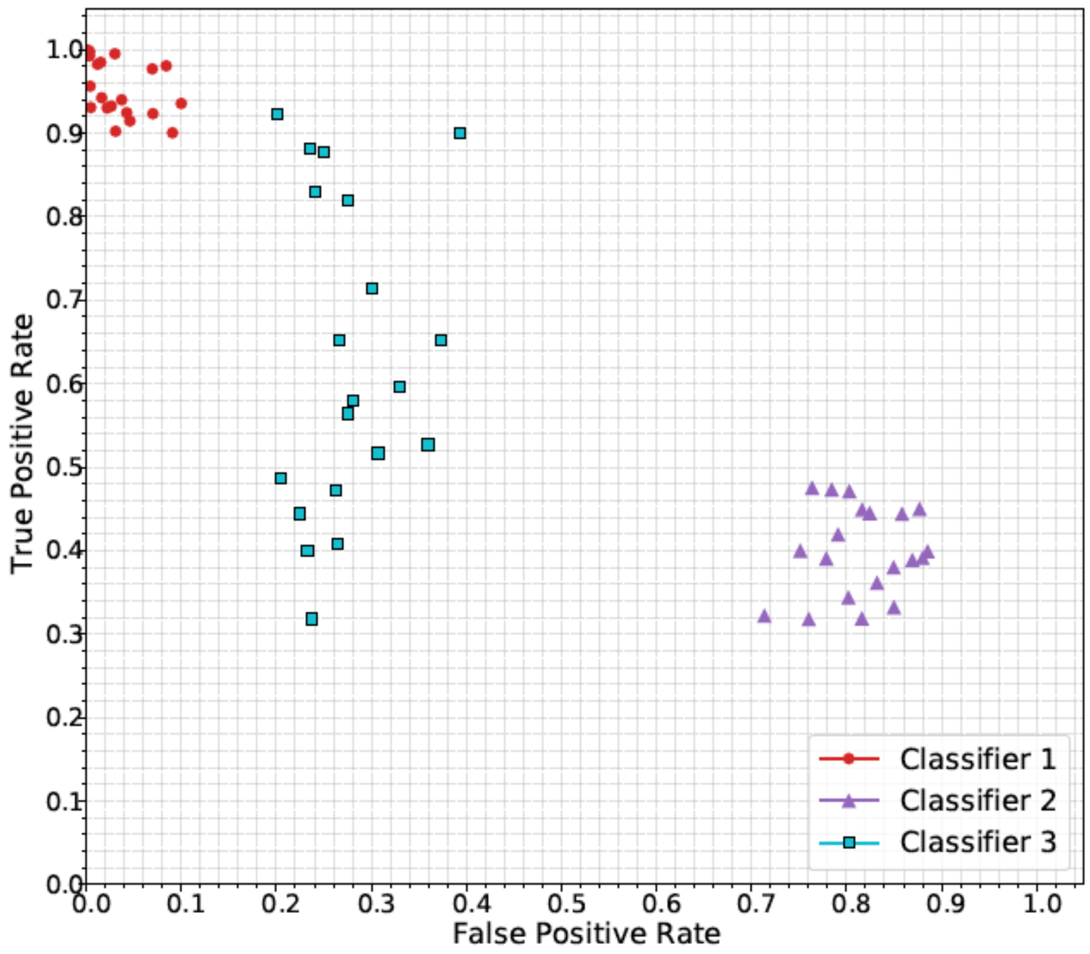
ECSER's highlight #5: the ROC plot

- The ROC plot can be used to visualize performance across multiple datasets



ECSER's highlight #5: the ROC plot

- The ROC plot can be used to visualize performance across multiple datasets
- … also, to explore the effect of the discrimination threshold between positives and negatives (not shown here)



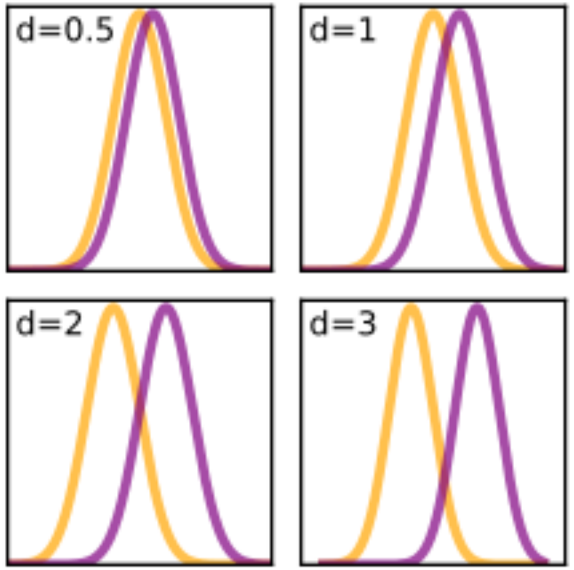
ECSER's highlight #6: statistical tests

- Which statistical test to use? ➡️

Test	Normal?	Same var?	Highlights	Suggested?
2+ *Classifiers: Pairwise Comparisons*				
Paired T	•		Sensitive to outliers [21], based on the absolute difference in performance	
Wilcoxon Signed-Rank			Based on ranks difference	•
Sign			Counts of wins, losses, ties. Weaker than Wilcoxon [21]	
Bayesian versions of Wilcoxon or Sign			Less affected by Type I Error. Requires definition of practical equivalence [8]	
3+ *Classifiers: Omnibus + Post-hoc test*				
Repeated measures ANOVA	•	•	Post-hoc: Tukey's HSD	
Friedman			Post-hoc: Nemenyi	•

ECSER's highlight #6: statistical tests

- Which statistical test to use? ➡️
- Not only p-value. Also, effect size! ⬇️



Test	Normal?	Same var?	Highlights	Suggested?
2+ Classifiers: Pairwise Comparisons				
Paired T	•		Sensitive to outliers [21], based on the absolute difference in performance	
Wilcoxon Signed-Rank			Based on ranks difference	•
Sign			Counts of wins, losses, ties. Weaker than Wilcoxon [21]	
Bayesian versions of Wilcoxon or Sign			Less affected by Type I Error. Requires definition of practical equivalence [8]	
3+ Classifiers: Omnibus + Post-hoc test				
Repeated measures ANOVA	•	•	Post-hoc: Tukey's HSD	
Friedman			Post-hoc: Nemenyi	•

4. Application to NLP4RE

Classification	ASUM (S129), RUBRIC (S223), WCC (S257), NFR2AC tool (S306), ALERTme (S332), PUMConf (337), FFRE (S341), AUR-BoW (S500), SEMIOS (S550), CRISTAL (S629), CoReq (S672), SD (S674), ACRE (S757), SOVA R-TC (S778), SMAA (S788), CSLabel (S892), HeRA (S718), NFR Locator (S758), SURF (S910), NFRFinder (S647)	20	15.38%

Classifying functional and quality requirements

- **Seminal classification problem** that aims at identifying NFRs (or qualities) for initial architectural design

Requirements Eng (2007) 12:103–120
DOI 10.1007/s00766-007-0045-1

ORIGINAL ARTICLE

Automated classification of non-functional requirements

**Jane Cleland-Huang · Raffaella Settimi ·
Xuchang Zou · Peter Solc**

Received: 3 November 2006 / Accepted: 22 February 2007 / Published online: 23 March 2007

Abstract This paper describes a technique for automating the detection and classification of non-functional requirements related to properties such as security, performance, and usability. Early detection of non-functional requirements enables them to be incorporated into the initial architectural design instead of being refactored in at a later date. The approach is used to detect and classify stakeholders' quality concerns across requirements speci-

is useful for supporting an analyst in tl
manually discovering NFRs, and furt
to quickly analyse large and complex
search for NFRs.

Keywords Non-functional requirem
Quality requirements · Classification

Classifying functional and quality requirements

- **Seminal classification problem** that aims at identifying NFRs (or qualities) for initial architectural design

- Dozens of tools in the literature
  - Keyword based, ML & DL classifiers, zero- and few-shot learning…

Requirements Eng (2007) 12:103–120
DOI 10.1007/s00766-007-0045-1

ORIGINAL ARTICLE

Automated classification of non-functional requirements

Jane Cleland-Huang · Raffaella Settimi ·
Xuchang Zou · Peter Solc

Received: 3 November 2006 / Accepted: 22 February 2007 / Published online: 23 March 2007

Abstract This paper describes a technique for automating the detection and classification of non-functional requirements related to properties such as security, performance, and usability. Early detection of non-functional requirements enables them to be incorporated into the initial architectural design instead of being refactored in at a later date. The approach is used to detect and classify stakeholders' quality concerns across requirements speci- is useful for supporting an analyst in tl manually discovering NFRs, and furt to quickly analyse large and complex search for NFRs.

Keywords Non-functional requirem Quality requirements · Classification

Classifying functional and quality requirements

- **Seminal classification problem** that aims at identifying NFRs (or qualities) for initial architectural design

- Dozens of tools in the literature
  - Keyword based, ML & DL classifiers, zero- and few-shot learning…
  - Often using the PROMISE NFR dataset

Requirements Eng (2007) 12:103–120
DOI 10.1007/s00766-007-0045-1

ORIGINAL ARTICLE

Automated classification of non-functional requirements

Jane Cleland-Huang · Raffaella Settimi · Xuchang Zou · Peter Solc

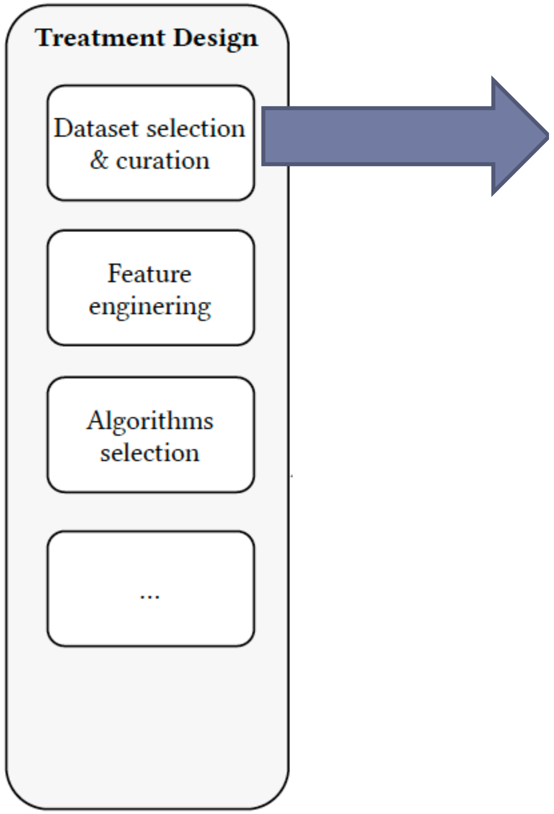
Received: 3 November 2006 / Accepted: 22 February 2007 / Published online: 23 March 2007

Abstract This paper describes a technique for automating the detection and classification of non-functional requirements related to properties such as security, performance, and usability. Early detection of non-functional requirements enables them to be incorporated into the initial architectural design instead of being refactored in at a later date. The approach is used to detect and classify stakeholders' quality concerns across requirements speci-

is useful for supporting an analyst in tl manually discovering NFRs, and furt to quickly analyse large and complex search for NFRs.

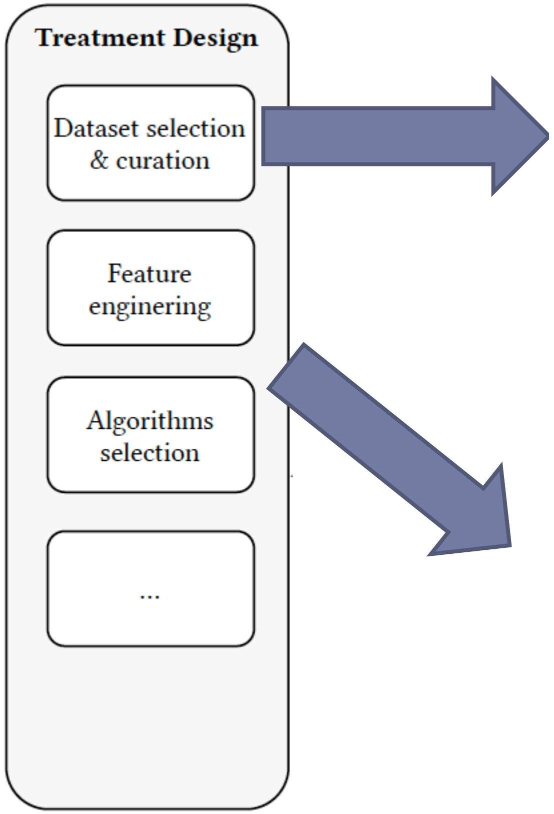
Keywords Non-functional requirem Quality requirements · Classification

Study design (prior to ECSER)



Data set	Public	New	Size	F	Q	Data set	Public	New	Size	F	Q
Dronology	✓		97	94	28	OAppT		✓	140	84	53
DUAP	✓	✓	148	138	110	PROMISE NFR	✓		625	310	382
ERec mgmt	✓	✓	228	163	149	RepReq		✓	99	40	47
ESA			236	91	211	ReqView	✓		87	75	32
Helpdesk			172	143	51	Streaming	✓	✓	291	135	233
Leeds Library	✓		85	44	61	User mgmt			138	126	25
NFR-Examples	✓	✓	130	15	117	WASP	✓		62	55	19
Totals									2538	1513	1518

Study design (prior to ECSER)



Data set	Public	New	Size	F	Q	Data set	Public	New	Size	F	Q
Dronology	✓		97	94	28	OAppT		✓	140	84	53
DUAP	✓	✓	148	138	110	PROMISE NFR	✓		625	310	382
ERec mgmt	✓	✓	228	163	149	RepReq		✓	99	40	47
ESA			236	91	211	ReqView	✓		87	75	32
Helpdesk			172	143	51	Streaming	✓	✓	291	135	233
Leeds Library	✓		85	44	61	User mgmt			138	126	25
NFR-Examples	✓	✓	130	15	117	WASP	✓		62	55	19
Totals									2538	1513	1518

Classifier	Year	ML algorithm	Distinctive characteristics
km500 [50]	2017	SVM	500 lexical and syntactical features (Word-level)
ling17 [18]	2019	SVM	17 linguistic features (Sentence-level)
norbert [39]	2020	Transfer learning	Word embedding (max seq. length 128), 10 epochs

S1. Evaluation method and data splitting

- Most of the literature uses PROMISE NFR
  - 625 requirements that pertain to 15 student projects
  - Generally, the studies only perform **validation**, **no testing**
  - We define two classifiers: *isFunctional* and *isQuality*

S1. Evaluation method and data splitting

- Most of the literature uses PROMISE NFR
  - 625 requirements that pertain to 15 student projects
  - Generally, the studies only perform **validation**, **no testing**
  - We define two classifiers: *isFunctional* and *isQuality*

- We use the **holdout method**
  - Training on 12 datasets, testing on the remaining one (repeat 13 times)
  - No hyper-parameter tuning (validation, S3-S4)

S2 & S5. Training and testing the model

- Training is performed on PROMISE NFR
  - In line with the literature

Data set	Public	New	Size	F	Q	Data set	Public	New	Size	F	Q
Dronology	✓		97	94	28	OAppT		✓	140	84	53
DUAP	✓	✓	148	138	110	PROMISE NFR	✓		625	310	382
ERec mgmt	✓	✓	228	163	149	RepReq		✓	99	40	47
ESA			236	91	211	ReqView	✓		87	75	32
Helpdesk			172	143	51	Streaming	✓	✓	291	135	233
Leeds Library	✓		85	44	61	User mgmt			138	126	25
NFR-Examples	✓	✓	130	15	117	WASP	✓		62	55	19
Totals									2538	1513	1518

S2 & S5. Training and testing the model

- Training is performed on PROMISE NFR
  - In line with the literature
- Testing is performed, as just said, according to the holdout method

Data set	Public	New	Size	F	Q	Data set	Public	New	Size	F	Q
Dronology	✓		97	94	28	OAppT		✓	140	84	53
DUAP	✓	✓	148	138	110	PROMISE NFR	✓		625	310	382
ERec mgmt	✓	✓	228	163	149	RepReq		✓	99	40	47
ESA			236	91	211	ReqView	✓		87	75	32
Helpdesk			172	143	51	Streaming	✓	✓	291	135	233
Leeds Library	✓		85	44	61	User mgmt			138	126	25
NFR-Examples	✓	✓	130	15	117	WASP	✓		62	55	19
Totals									2538	1513	1518

S6. Reporting the confusion matrix

- This is simply a presentation of the raw results…

		isF				*isQ*			
Data set	Classifier	TP	FP	TN	FN	TP	FP	TN	FN
Training (PROMISE NFR)	*ling17*	229	83	232	81	315	60	183	67
	km500	306	6	309	4	382	5	238	0
	norbert	301	10	305	9	382	27	216	0
Test (cumulative)	*ling17*	1009	321	365	194	673	258	495	463
	km500	655	185	501	548	806	377	376	330
	norbert	940	159	527	263	998	362	391	138

S6. Reporting the confusion matrix

- This is simply a presentation of the raw results…

		isF				*isQ*			
Data set	Classifier	TP	FP	TN	FN	TP	FP	TN	FN
Training (PROMISE NFR)	*ling17*	229	83	232	81	315	60	183	67
	km500	306	6	309	4	382	5	238	0
	norbert	301	10	305	9	382	27	216	0
Test (cumulative)	*ling17*	1009	321	365	194	673	258	495	463
	km500	655	185	501	548	806	377	376	330
	norbert	940	159	527	263	998	362	391	138

- But some aspects already stand out!

S7-S8. Performance and overfitting

- For simplicity, let's examine F_1 here

Task	Classifier	Training	Test	Overfitting (Test - Training)
		F_1		
isF	*ling17*	0.74	0.75 ± 0.11	0.01 ± 0.11
	km500	0.98	0.61 ± 0.09	-0.38 ± 0.09
	norbert	0.97	0.79 ± 0.09	-0.18 ± 0.09
isQ	*ling17*	0.80	0.62 ± 0.09	-0.18 ± 0.09
	km500	0.99	0.60 ± 0.12	-0.39 ± 0.12
	norbert	0.96	0.71 ± 0.13	-0.25 ± 0.13

S7-S8. Performance and overfitting

- For simplicity, let's examine F_1 here

Task	Classifier	Training	Test	Overfitting (Test - Training)
		F_1		
isF	*ling17*	0.74	0.75 ± 0.11	0.01 ± 0.11
	km500	0.98	0.61 ± 0.09	-0.38 ± 0.09
	norbert	0.97	0.79 ± 0.09	-0.18 ± 0.09
isQ	*ling17*	0.80	0.62 ± 0.09	-0.18 ± 0.09
	km500	0.99	0.60 ± 0.12	-0.39 ± 0.12
	norbert	0.96	0.71 ± 0.13	-0.25 ± 0.13

- Who's the winner?

S7-S8. Performance and overfitting

- For simplicity, let's examine F_1 here

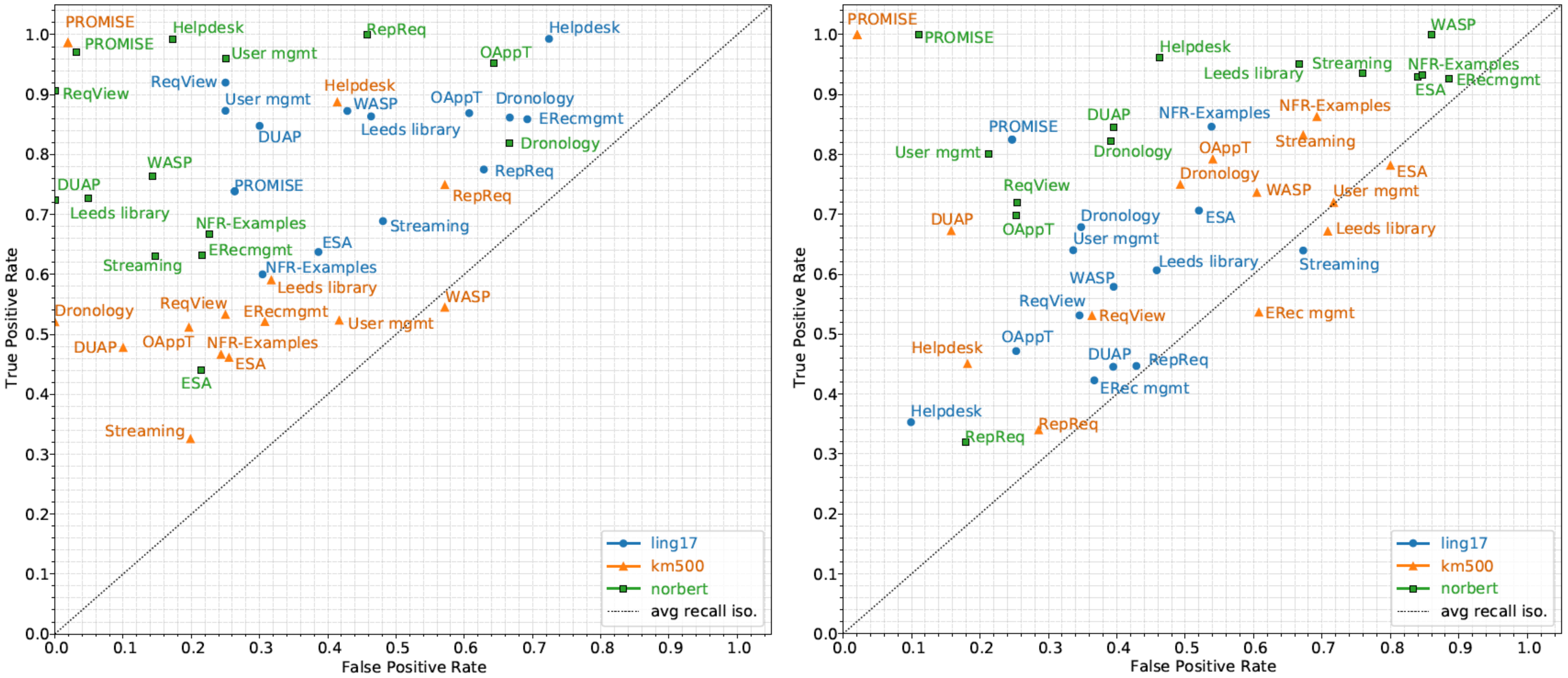
Task	Classifier	Training	Test	Overfitting (Test - Training)
		F_1		
isF	*ling17*	0.74	0.75 ± 0.11	0.01 ± 0.11
	km500	0.98	0.61 ± 0.09	-0.38 ± 0.09
	norbert	0.97	0.79 ± 0.09	-0.18 ± 0.09
isQ	*ling17*	0.80	0.62 ± 0.09	-0.18 ± 0.09
	km500	0.99	0.60 ± 0.12	-0.39 ± 0.12
	norbert	0.96	0.71 ± 0.13	-0.25 ± 0.13

- Who's the winner?
  - *km500* fits best the training set
  - *norbert* has the best performance on the test set
  - *ling17* has the smallest overfitting

S9. ROC Plot (for isFunctional)

- *norbert* is closer to the ROC heaven (top-left corner) for many datasets
- *ling17* tends to have more false positives
- *km500* has more false negatives

S9. ROC Plots (isF and isQ)



Worse performance for the isQ case (the *more interesting* class!)

S10. Statistical tests

- Is one of these classifiers significantly better?
- The results are mixed

		Omnibus	Post-Hoc/Cohen's d (magnitude)		
			ling17 vs *km500*	*ling17* vs *norbert*	*km500* vs *norbert*
isF	Prec	p^f=0.002**	0.059 (none)	0.37 (small)	0.314 (small)
	Rec	p^a=0.0**	2.152 (large)	0.236 (small)	1.528 (large)
	F_1	p^a=0.0**	1.39 (large)	0.43 (small)	1.989 (large)
isQ	Prec	p^a=0.066			
	Rec	p^f=0.0**	0.683 (medium)	1.659 (large)	0.977 (large)
	F_1	p^a=0.014*	0.134 (none)	0.778 (medium)	0.807 (large)

S10. Statistical tests

- Is one of these classifiers significantly better?
- The results are mixed
  - Yes, for *km500* vs. *norbert* in the isFunctional case

		Omnibus	Post-Hoc/Cohen's d (magnitude)		
			ling17 vs *km500*	*ling17* vs *norbert*	*km500* vs *norbert*
isF	Prec	p^f=0.002**	0.059 (none)	0.37 (small)	0.314 (small)
	Rec	p^a=0.0**	2.152 (large)	0.236 (small)	1.528 (large)
	F_1	p^a=0.0**	1.39 (large)	0.43 (small)	1.989 (large)
isQ	Prec	p^a=0.066			
	Rec	p^f=0.0**	0.683 (medium)	1.659 (large)	0.977 (large)
	F_1	p^a=0.014*	0.134 (none)	0.778 (medium)	0.807 (large)

S10. Statistical tests

- Is one of these classifiers significantly better?
- The results are mixed
  - Yes, for *km500* vs. *norbert* in the isFunctional case
  - Almost never for isQuality (only recall when comparing *ling17* and *norbert*)

		Omnibus	Post-Hoc/Cohen's d (magnitude)		
			ling17 vs *km500*	*ling17* vs *norbert*	*km500* vs *norbert*
isF	Prec	p^f=0.002**	0.059 (none)	0.37 (small)	0.314 (small)
	Rec	p^a=0.0**	2.152 (large)	0.236 (small)	1.528 (large)
	F_1	p^a=0.0**	1.39 (large)	0.43 (small)	1.989 (large)
isQ	Prec	p^a=0.066			
	Rec	p^f=0.0**	0.683 (medium)	1.659 (large)	0.977 (large)
	F_1	p^a=0.014*	0.134 (none)	0.778 (medium)	0.807 (large)

In summary

- We confirm that *norbert* outperforms both *ling17* and *km500* on unseen data
  - But hardly in a statistical sense (could be due to insufficient data points)

In summary

- We confirm that *norbert* outperforms both *ling17* and *km500* on unseen data
  - But hardly in a statistical sense (could be due to insufficient data points)

- The "losers" still have good properties:
  - *ling17* has the smallest overfitting
  - *km500* fits best the training data

In summary

- We confirm that *norbert* outperforms both *ling17* and *km500* on unseen data
  - But hardly in a statistical sense (could be due to insufficient data points)

- The "losers" still have good properties:
  - *ling17* has the smallest overfitting
  - *km500* fits best the training data

- For *norbert*, the original paper showed equivalent performance for isQ and isF. This is not the case in our experiments on the test sets.

5. The way ahead

A second case on flaky tests

- **Flaky tests** are tests with non-deterministic outcomes on the same code

Alshammari, Abdulrahman, Christopher Morris, Michael Hilton, and Jonathan Bell. *Flakeflagger: Predicting flakiness without rerunning tests*. In 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), pp. 1572-1584. IEEE, 2021.

A second case on flaky tests

- **Flaky tests** are tests with non-deterministic outcomes on the same code
- We took three approaches from the literature
  - **FF** (FlakeFlagger): an approach based on machine learning
  - **Voc**: a keyword-based approach to determine flakiness
  - **VocFF**: a combination of the previous two

Alshammari, Abdulrahman, Christopher Morris, Michael Hilton, and Jonathan Bell. *Flakeflagger: Predicting flakiness without rerunning tests*. In 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), pp. 1572-1584. IEEE, 2021.

A second case on flaky tests

- **Flaky tests** are tests with non-deterministic outcomes on the same code
- We took three approaches from the literature
  - **FF** (FlakeFlagger): an approach based on machine learning
  - **Voc**: a keyword-based approach to determine flakiness
  - **VocFF**: a combination of the previous two
- Previous results showed that FF and VocFF outperform Voc
  - They reported performance based on cross-validation (**no test set**)

Alshammari, Abdulrahman, Christopher Morris, Michael Hilton, and Jonathan Bell. *Flakeflagger: Predicting flakiness without rerunning tests*. In 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), pp. 1572-1584. IEEE, 2021.

How did we create a test set?

- We start from their dataset (22 projects)
- We order the projects by # of flaky tests
- We alternatively assign the projects with more positives to train and test set

Project	Tests	Flaky	Data splitting	
			Train set	Test set
spring-boot	2,108	160	✓	
hbase	431	145		✓
alluxio	187	116	✓	
okhttp	810	100		✓
ambari	324	52	✓	
hector	142	33		✓
activiti	2,043	32	✓	
java-websocket	145	23		✓
wildfly	1,023	23	✓	
httpcore	712	22		✓
logback	805	22	✓	
incubator-dubbo	2,174	19		✓
http-request	163	18	✓	
wro4j	1,135	16		✓
orbit	86	7	✓	
undertow	183	7	✓	
achilles	1,317	4	✓	
elastic-job-lite	558	3	✓	
zxing	345	2	✓	
assertj-core	6,261	1	✓	
handlebars.java	420	1	✓	
ninja	307	1	✓	
commons-exec	55	0	✓	
jimfs	212	0	✓	
Train set total	16,397	449		
Test set total	5,549	358		

Results, quick overview

- Training and validation as in the original paper, but…

	Classifier	Precision	Recall	F_1
Training	*FF*	1.00	1.00	1.00
	Voc	0.13	0.89	0.23
	VocFF	1.00	1.00	1.00
Validation	*FF*	0.71 ± 0.05	0.78 ± 0.07	0.74 ± 0.04
	Voc	0.12 ± 0.02	0.77 ± 0.08	0.21 ± 0.03
	VocFF	0.75 ± 0.04	0.79 ± 0.06	0.77 ± 0.03
Tests	*FF*	0.09 ± 0.19	0.05 ± 0.07	0.03 ± 0.04
	Voc	0.15 ± 0.17	0.34 ± 0.18	0.16 ± 0.15
	VocFF	0.12 ± 0.23	0.05 ± 0.06	0.06 ± 0.09

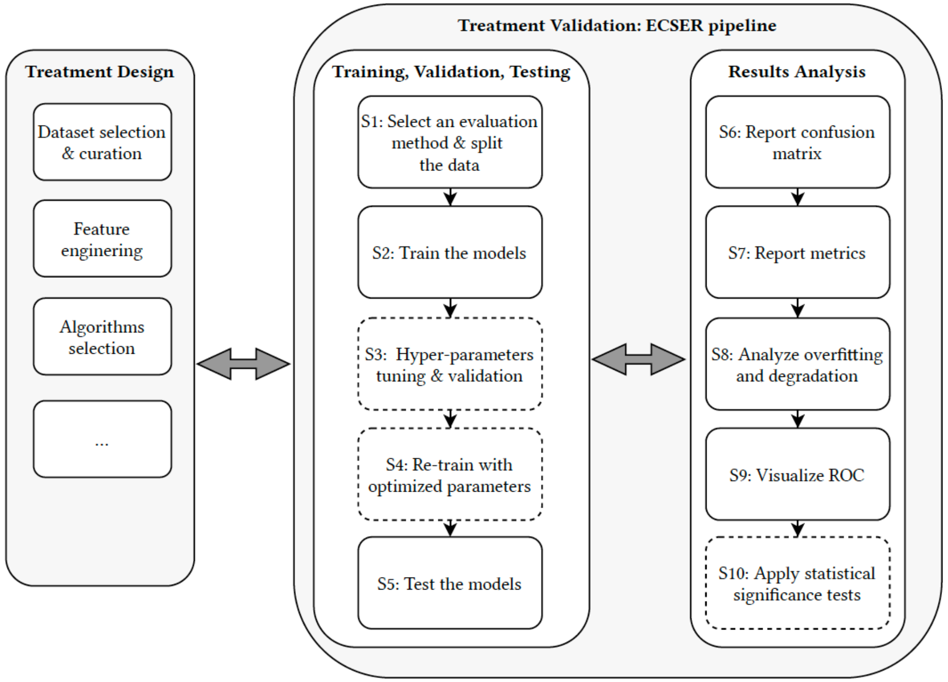
Results, quick overview

- Training and validation as in the original paper, but…
- Performance on the test set changes drastically: **contradictory results**
  - Voc is best when applied on unseen data

	Classifier	Precision	Recall	F_1
Training	*FF*	1.00	1.00	1.00
	Voc	0.13	0.89	0.23
	VocFF	1.00	1.00	1.00
Validation	*FF*	0.71 ± 0.05	0.78 ± 0.07	0.74 ± 0.04
	Voc	0.12 ± 0.02	0.77 ± 0.08	0.21 ± 0.03
	VocFF	0.75 ± 0.04	0.79 ± 0.06	0.77 ± 0.03
Tests	*FF*	0.09 ± 0.19	0.05 ± 0.07	0.03 ± 0.04
	Voc	0.15 ± 0.17	0.34 ± 0.18	0.16 ± 0.15
	VocFF	0.12 ± 0.23	0.05 ± 0.06	0.06 ± 0.09

What's next for the ML4SE and NLP4RE community?

What's next for the ML4SE and NLP4RE community?



What's next for the ML4SE and NLP4RE community?

Use multiple datasets, unless
(i) data labeling is practically possible
(ii) you can prove that real-world datasets are homogeneous

Project	Tests	Flaky	Data splitting Train set	Data splitting Test set
spring-boot	2,108	160	✓	
hbase	431	145		✓
alluxio	187	116	✓	
okhttp	810	100		✓
ambari	324	52	✓	
hector	142	33		✓
activiti	2,043	32	✓	
java-websocket	145	23		✓
wildfly	1,023	23	✓	
httpcore	712	22		✓
logback	805	22	✓	
incubator-dubbo	2,174	19		✓
http-request	163	18	✓	
wro4j	1,135	16		✓
orbit	86	7	✓	
undertow	183	7	✓	
achilles	1,317	4	✓	
elastic-job-lite	558	3	✓	
zxing	345	2	✓	
assertj-core	6,261	1	✓	
handlebars.java	420	1	✓	
ninja	307	1	✓	
commons-exec	55	0	✓	
jimfs	212	0	✓	
Train set total	16,397	449		
Test set total	5,549	358		

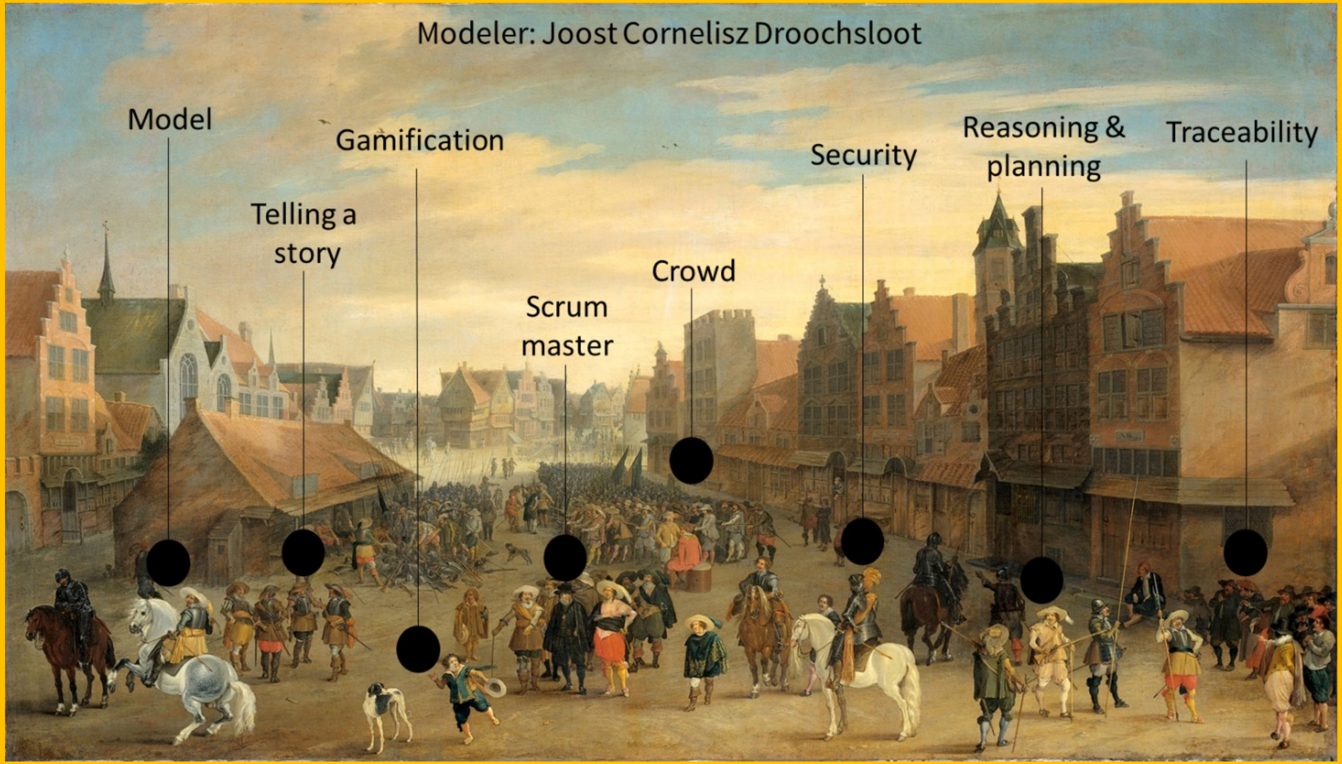
What’s next for the ML4SE and NLP4RE community?

Evolve ECSER and the research methods in the field

A few directions

- What happens with zero-shot learning where training is not necessary
- What are the “right” statistical tests?
- What are the most suitable metrics?
- Beyond classification – other ML tasks

Thank you for listening! Questions?



RE-Lab's research illustrated, 2018

f.dalpiaz@uu.nl

@FabianoDalpiaz