Exploring the Relationship between Design Metrics and Software Diagnosability using Machine Learning

Thomas Dornberger, Sofie Kemper

2018-07-13

Research Question

- Is there a connection between software metrics and the applicability of spectrum-based fault localization techniques?

- Focus on the viewpoint of the developer

Structure

1) Data collection

2) Data exploration

3) Machine learning techniques

4) Results & discussion

Static and dynamic metrics

- LOC, SLOC, cyclomatic complexity, nesting depth assessment, …
  → Measure project size and complexity

- **Code Smells** like unused parameters, star imports, …
  → Hypothesis: Link between quality and software diagnosability

- Graph centrality, vertex/edge connectivity
  → Deeper analysis of code **complexity**
  → Hypothesis: Link between complexity and software diagnosability

Test suite and bug metrics

- Nr of relevant tests, **DDU** as density diversity uniqueness
  → Empirical evidence: Optimization of a test suite in terms of DDU improves software diagnosability

- Size of the bug, repair patterns and actions
  → Hypothesis: Close link between a specific bug and its diagnosability

Target Values

- Focus on the viewpoint of the developer

- #Methods to examine

Lang_5

Components	D-Star 2	Rank
Method A	0.8	1
Method B	0.67	2
Method C	0.67	2
Method D	0.67	2
Method Z	0.2	5

Target Values

- Focus on the viewpoint of the developer

- #Methods to examine

Lang_5

Components	D-Star 2	Rank
Method A	**0.8**	**1**
Method B	0.67	2
Method C	0.67	2
Method D	0.67	2
Method Z	0.2	5

#Methods to examine = **1**

Target Values

- Focus on the viewpoint of the developer

- #Methods to examine

Lang_5

Components	D-Star 2	Rank
Method A	0.8	1
Method B	**0.67**	**2**
Method C	0.67	2
Method D	0.67	2
Method Z	0.2	5

#Methods to examine = **3**

Target Values

- Focus on the viewpoint of the developer
- #Methods to examine
- Class labels

  perfect – good – medium – useless

Structure

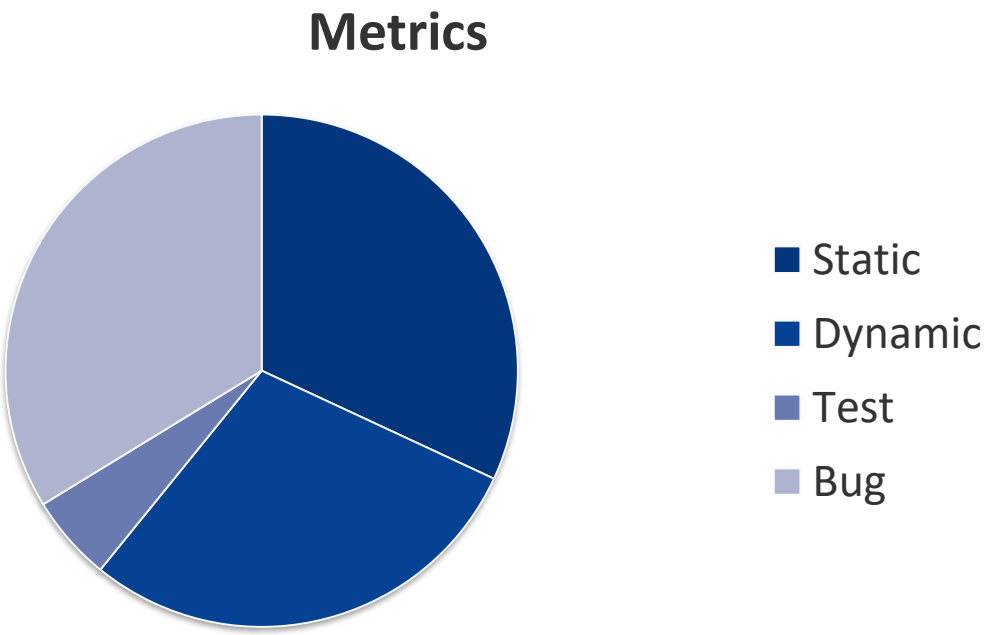
1) Data collection

2) Data exploration

3) Machine learning techniques

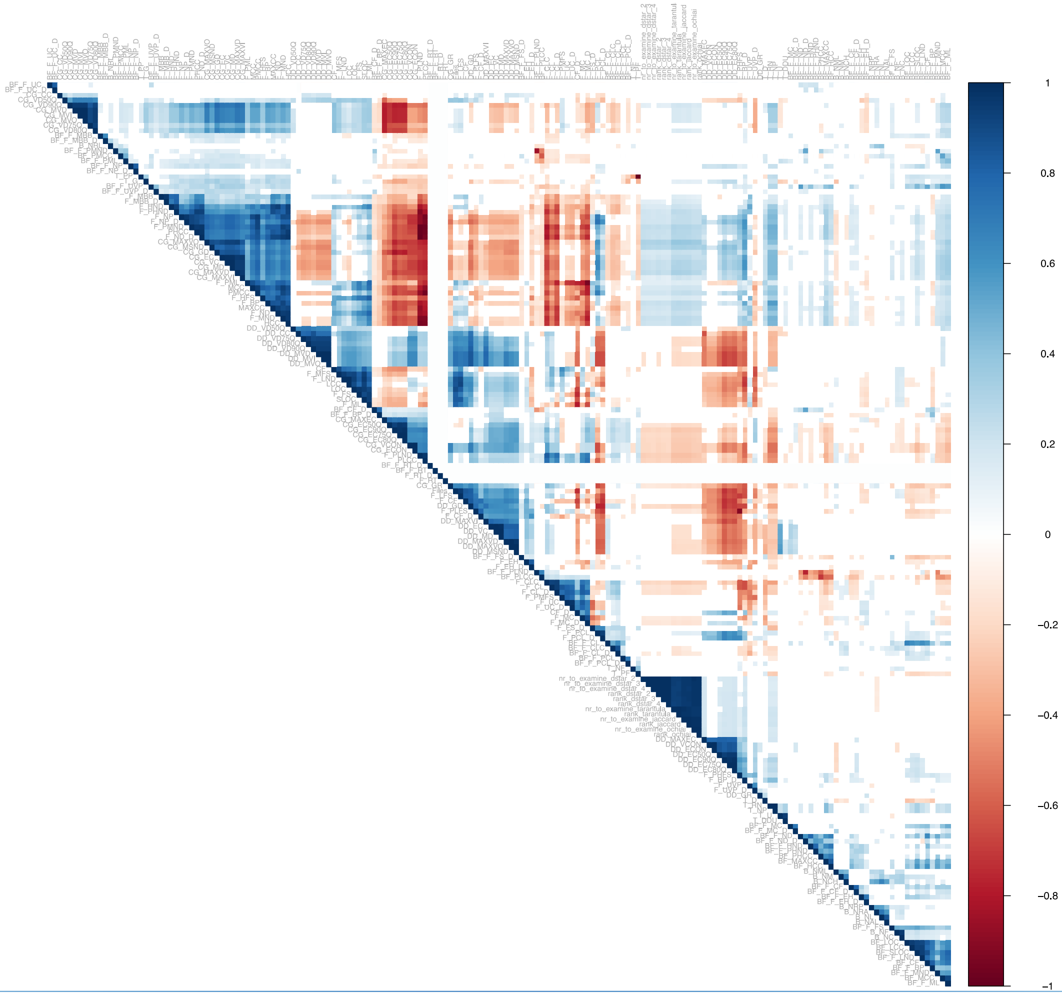
4) Results & discussion

Data Overview

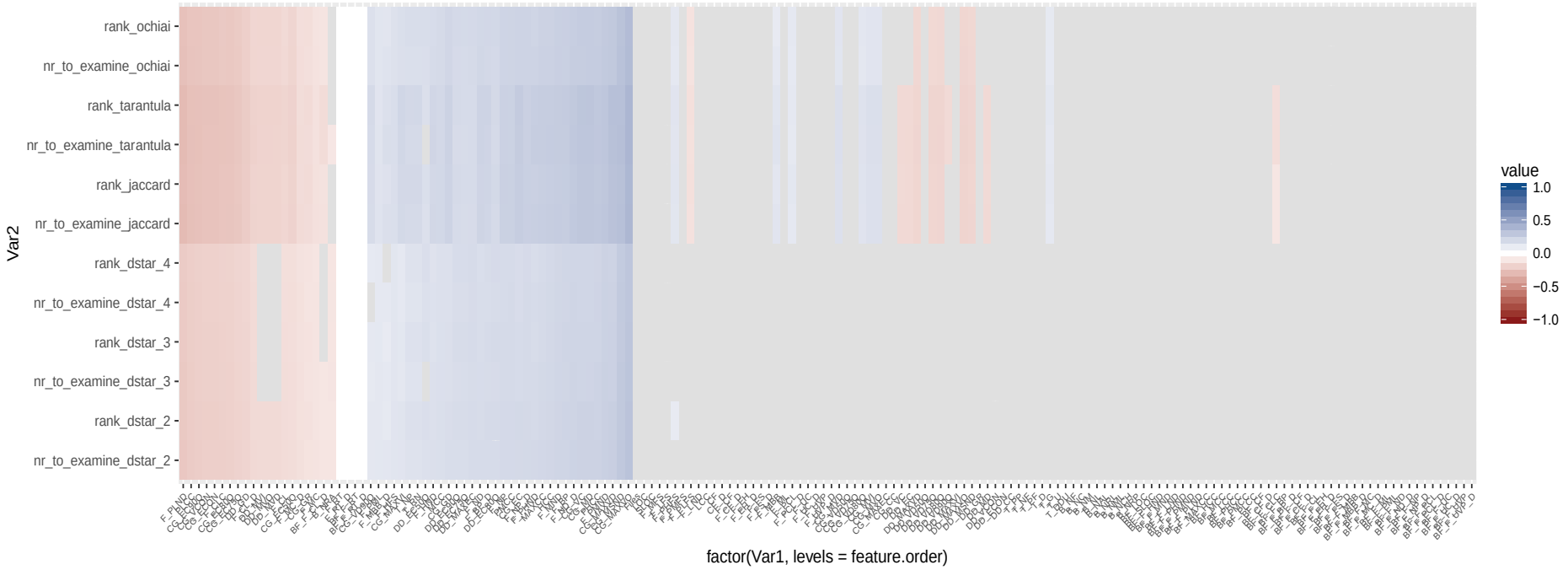


- 166 metrics collected in total
- Some missing data due to problems
- (e.g.,unreliable callgraph generation)

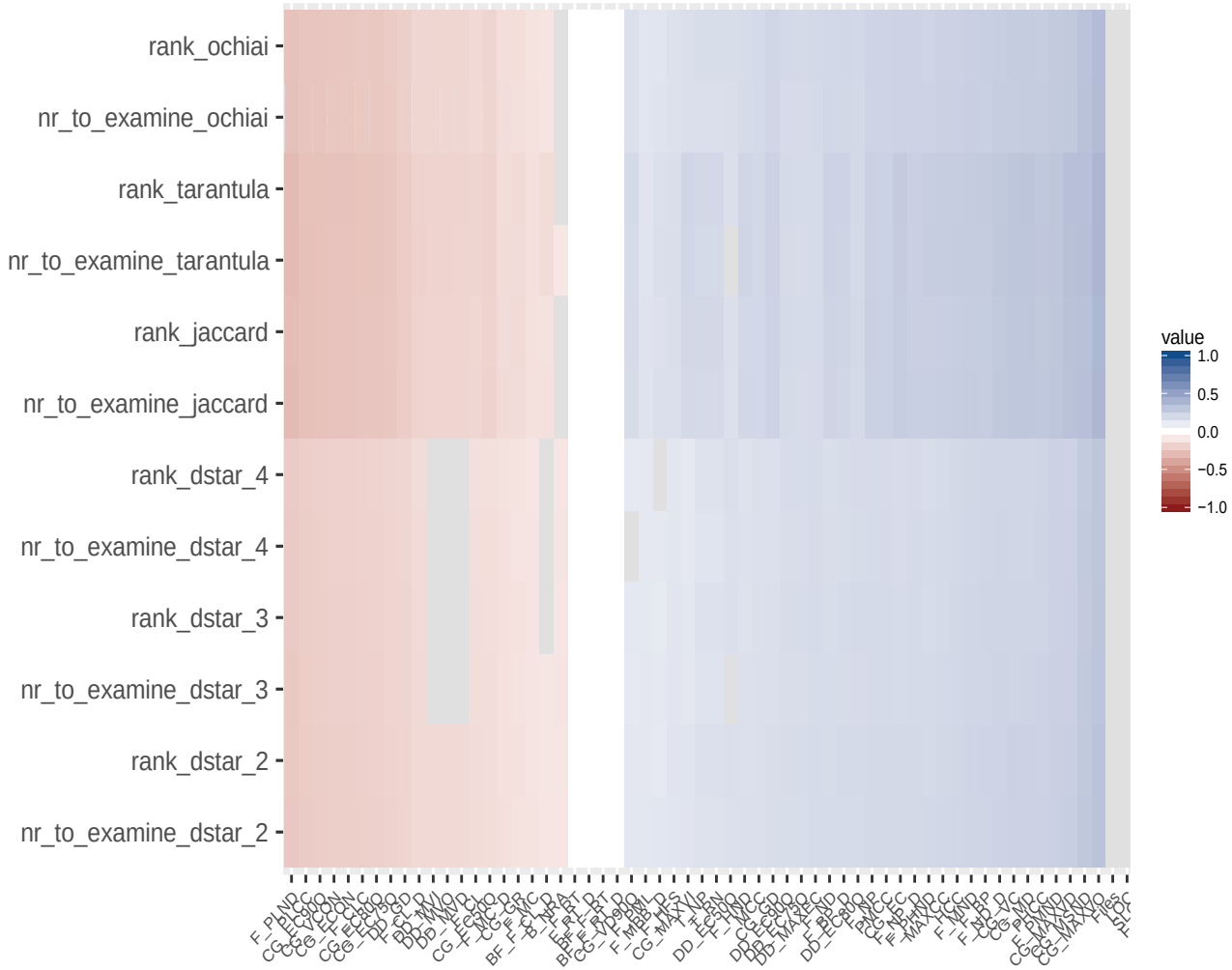
Feature-Feature Correlations



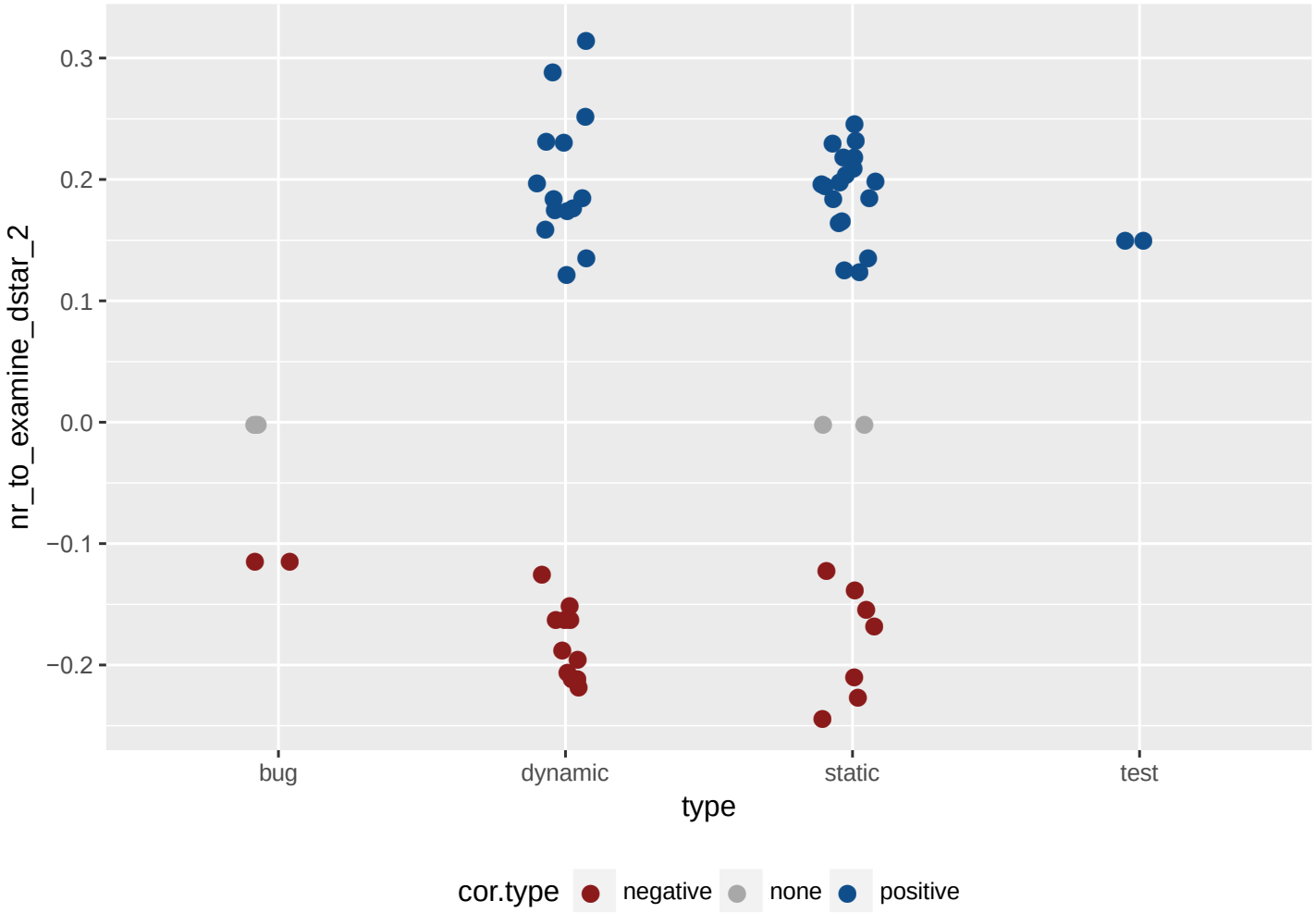
Feature-Target-Correlations



Feature-Target-Correlations



Correlated Features - Types



Structure

1) Data collection

2) Data exploration

3) Machine learning techniques

4) Results & discussion

Machine learning – Linear Regression I

Why choose least squares linear regression?

- Performant and easy to interpret
- Direct link to found correlations

- We use a train-test split to ensure generalizable results and evaluate with mean squared error (MSE) and adjusted R-Squared (R^2)

- First approach: random combinations of two features -> very bad results
- Refine by using only “important” features -> slight improvement

Machine learning – Linear Regression II

- Outliers and strongly scattered, high target values seem problematic
- Apply a log-transformation

- Significantly improved results: MSE ~ 0.9, R^2 ~ 0.3
- Still not a great result, but a starting point for other methods
- Linear regression might not be powerful enough

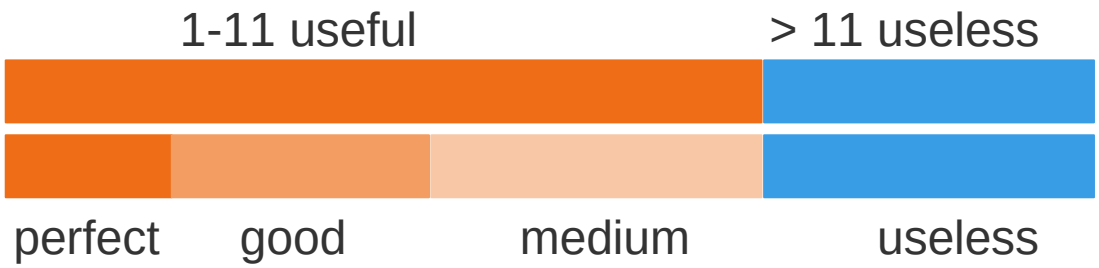
- Best models use **only dynamic** metrics!
  - Maximum vertex outdegree in callgraph
  - Eigenvector centrality distribution in the data dependency graph
  - Callgraph size: mean geodesic distance, edge count, or vertex count

Machine learning – Decision trees

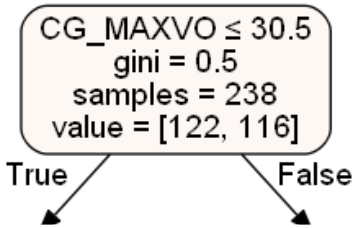
- Decision tree regression
  - Bad results, negative r2-scores
  - Problem: High target values are very strongly scattered

- Decision tree classification

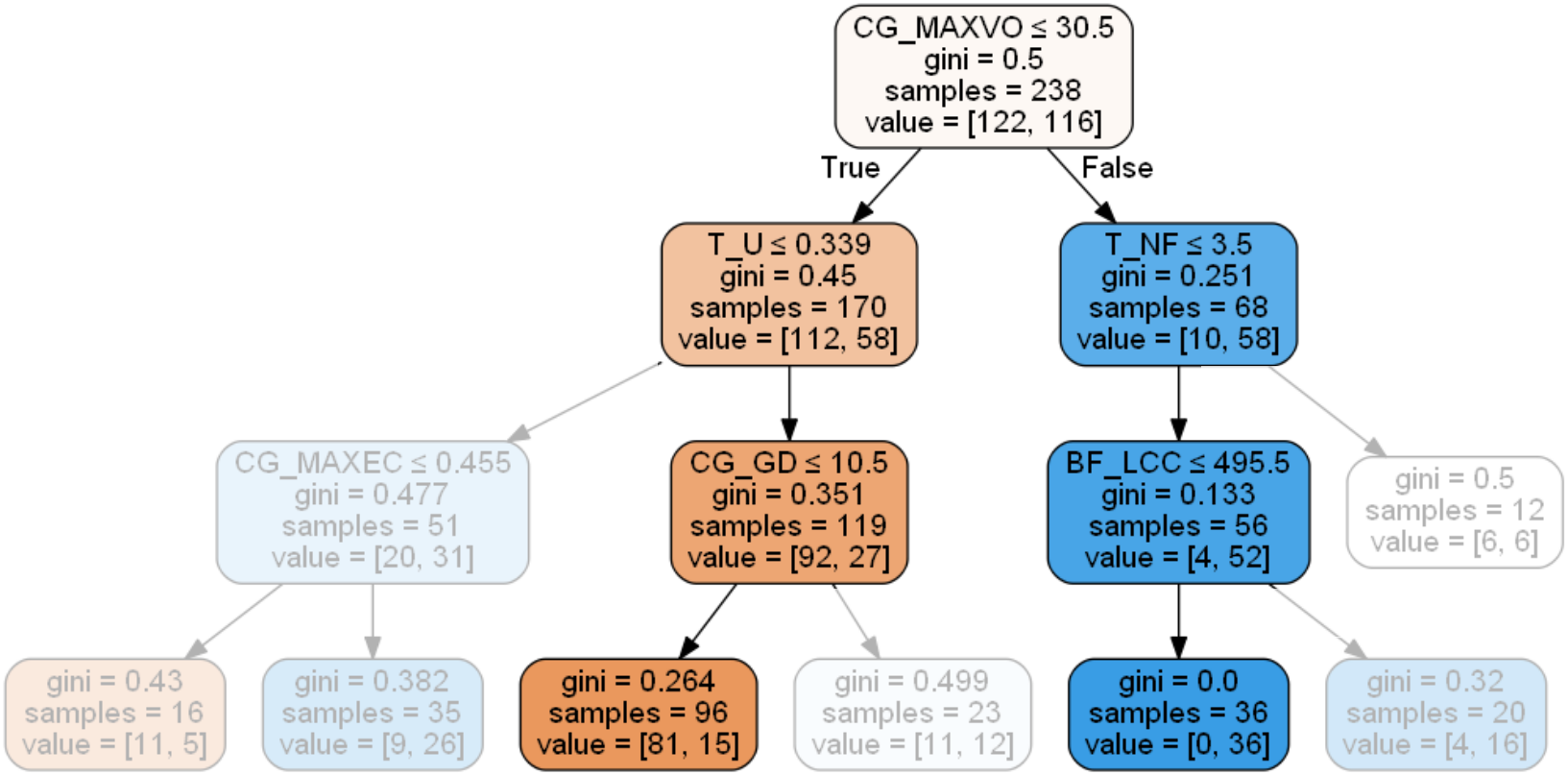
- Class labels:



Decision Tree – coarse-grained class labels



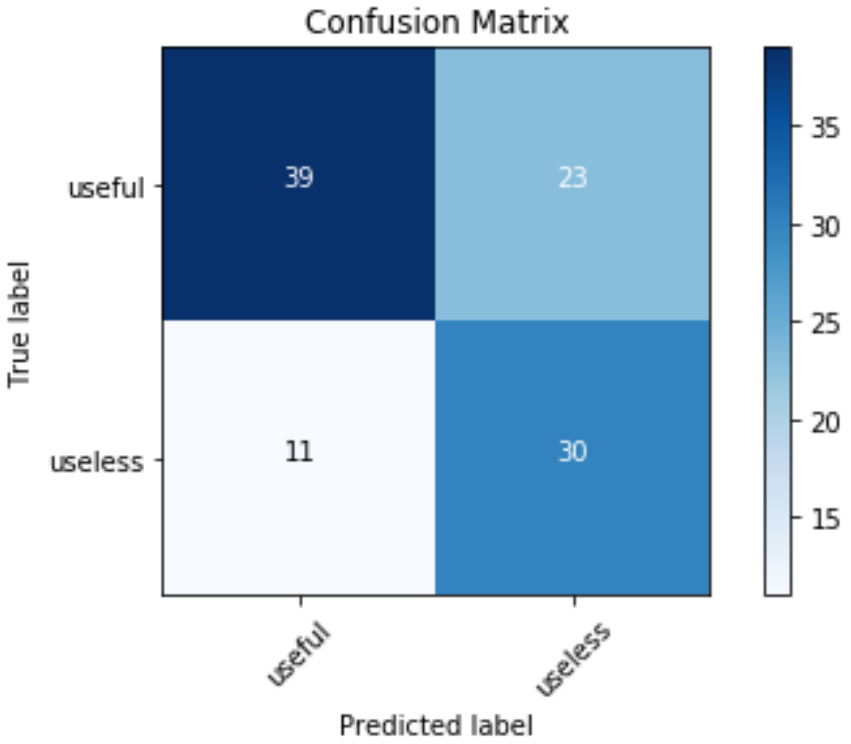
Decision Tree – coarse-grained class labels



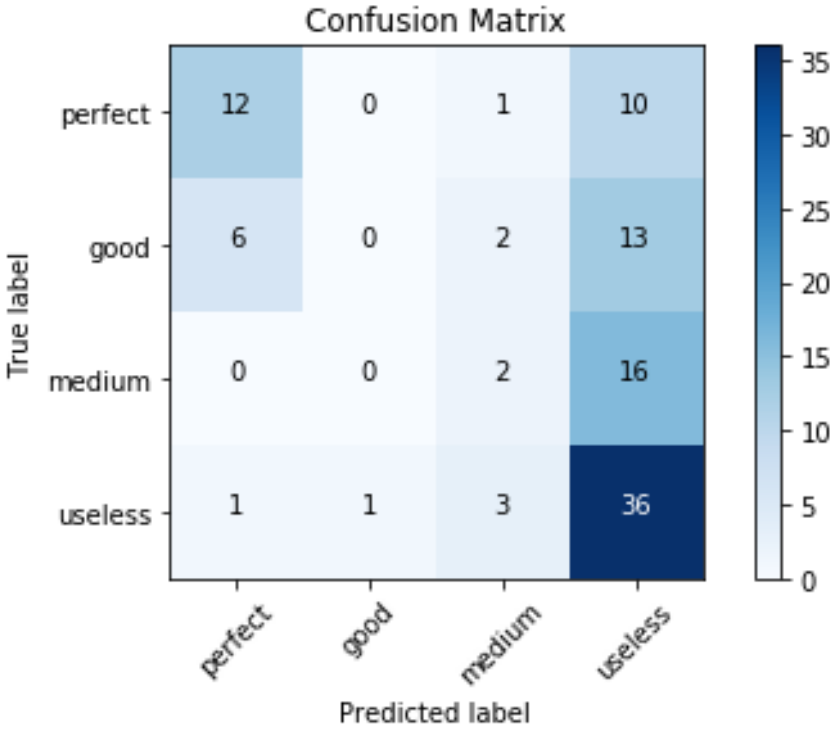
F1-score ≈ 0.64

accuracy-score ≈ 0.69

Decision Tree – coarse-grained class labels (II)



Decision Tree – fine-grained class labels



→ Samples predicted as perfect tend to be actually useful

accuracy-score (equal to F1 ‘mean’ score) ≈ 0.50

Structure

1) Data collection

2) Data exploration

3) Machine learning techniques

4) Results & discussion

Discussion: Insights from Models

- All three analysis methods share some important features:
  - Callgraph complexity and centrality, especially high quartiles of eigenvector centrality
- In linear regression:
  - Only dynamic metrics are important
  - The size of the data dependency graph is also used
- In decision trees:
  - Higher diversity of metrics: also some test metrics
- Decision trees tend to perform better (but only for classification)

Threats to Validity & Future work

- External validity is given: diverse and reliable data
- Internal validity is problematic: too little data; DDU and some other test metrics only computed on relevant tests

- Outlier elimination could significantly improve robustness and results
- Focus on dynamic complexity metrics to train more complex models

- Use auto-encoders or other dimensionality reduction techniques
- Clustering to find native classes in the data

Conclusion

- Most important: dynamic metrics, especially graph centrality and complexity

- Decision tree classifier: identification of software versions for which spectrum-based fault localization works well

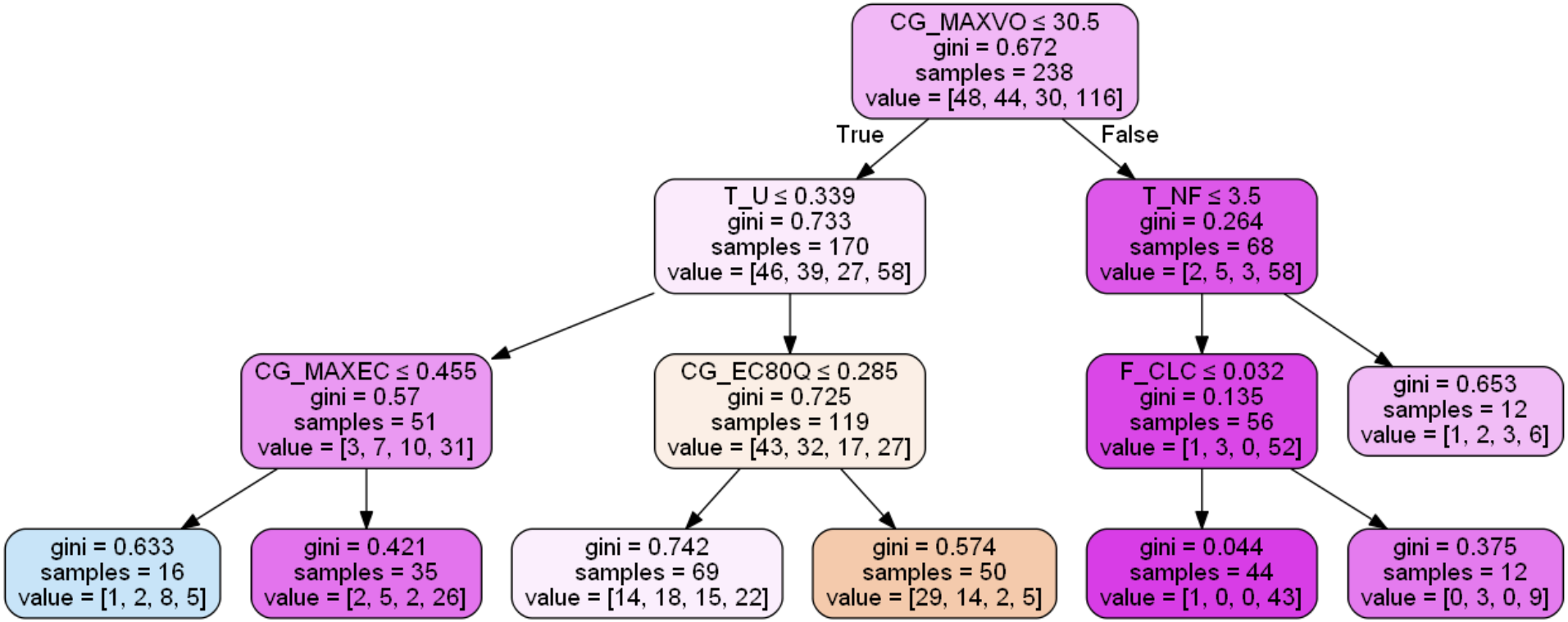
→ Help developer decide whether to use fault localization suggestions

Thank you for your attention!
Questions?

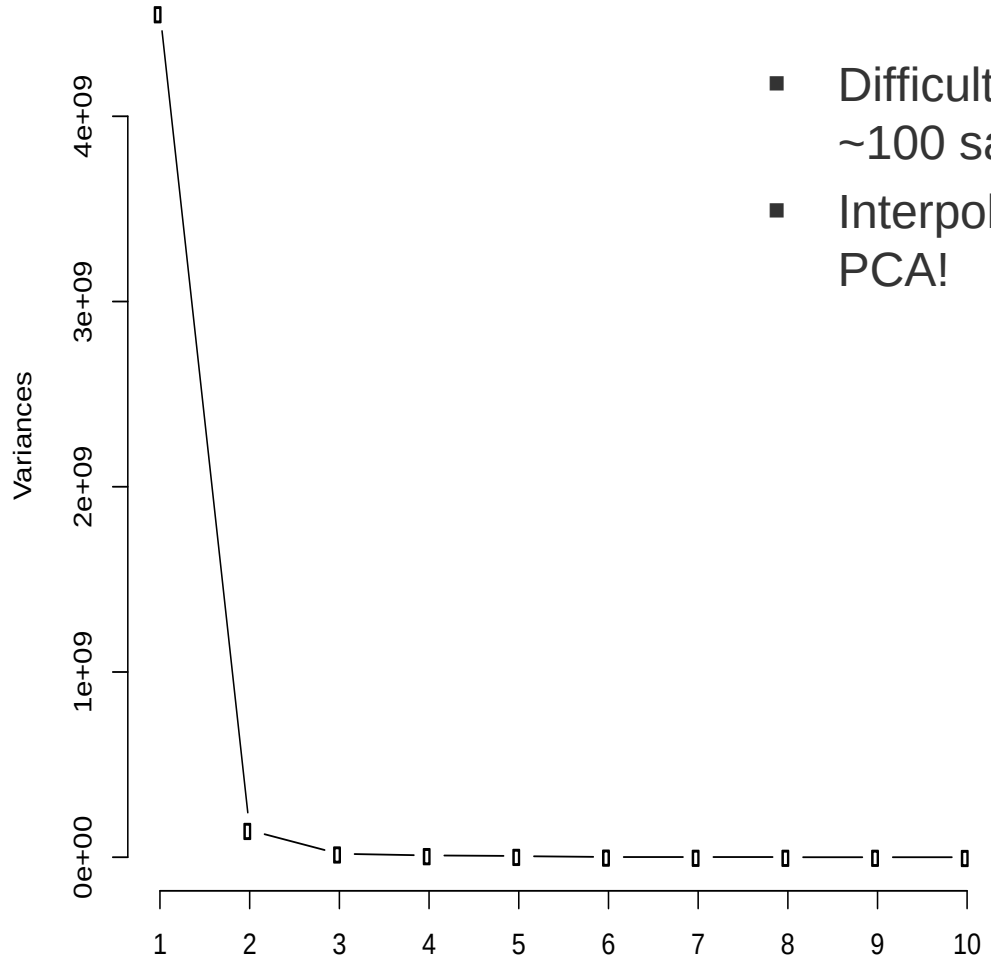
Backup (I)

- CG_MAXVO: maximum vertex outdegree in callgraph
- DD_EC90Q: 90th quartile of eigenvector centrality values in data dependency graph
- DD_MAXEC: maximum eigenvector centrality value in data dependency graph
- CG_MD: mean geodesic distance in callgraph
- CG_EC: number of edges in callgraph
- CG_VC: number of vertices in callgraph
- T_U: uniqueness of test suite (relevant tests)
- T_DDU: density-diversity-uniqueness of test suite (relevant tests)
- T_NF: absolute number of failing tests (relevant tests)

Backup (II) Decision Tree – fine-grained



Backup (III) - PCA



- Difficult due to missing values; only ~100 samples could actually be used
- Interpolation of missing values distorts PCA!