An Abstract Interpretation-Based Data Leakage Static Analysis

Filip Drobnjaković, Pavle Subotić, Caterina Urban

18th International Symposium on Theoretical Aspects of Software Engineering (TASE 2024)

Data Leakage in the Real World

Child Welfare

Proceedings of Machine Learning Research 81:1–15, 2018 Conference on Fairness, Accountability, and Transparency

A case study of algorithm-assisted decision making in child maltreatment hotline screening decisions

Alexandra Chouldechova
Heinz College
Carnegie Mellon University
Pittsburgh, PA, 15213, USA
Emily Putnam-Hornstein
Suzanne Dworak-Peck School of Social Work
University of Southern California
Los Angeles, CA, 90089, USA
Diana Benavides-Prado
Oleksandr Fialko
Rhema Vaithianathan
Centre for Social Data Analytics
Auckland University of Technology
Auckland, New Zealand

Editors: Sorelle A. Friedler and Christo Wilson

Abstract

Every year there are more than 3.6 million referrals made to child protection agencies across the US. The practice of screening calls is left to each jurisdiction to follow local practices and policies, potentially leading to large variation in the way in which referrals are treated across the country. Whilst increasing access to linked administrative data is available, it is difficult for welfare workers to make systematic use of historical information about all the children and adults on a single referral call. Risk prediction models that use routinely collected administrative data can help call workers to better identify cases that are likely to result in adverse outcomes. However, the use of predictive analytics in the area of child welfare is contentious. There

USA. We discuss the result
sis to-date, and also highlig
and data bias issues that p
for model evaluation and

1. Introduction

Every year there are more
rals made to child protect
US. It is estimated that
investigated for child ab
18 years (Kim et al., 201
cate that far from being
more children are being
fare agencies than prev
screening these referral
risdiction to follow local practices and
These practices usually involve caseworkers gath-
ering details about the adults and children associ-
ated with the alleged victim. Often, the decision
to investigate or not is made without
speaking with them.

https://www.aisnakeoil.com/p/the-bait-and-switch-behind-ai-risk

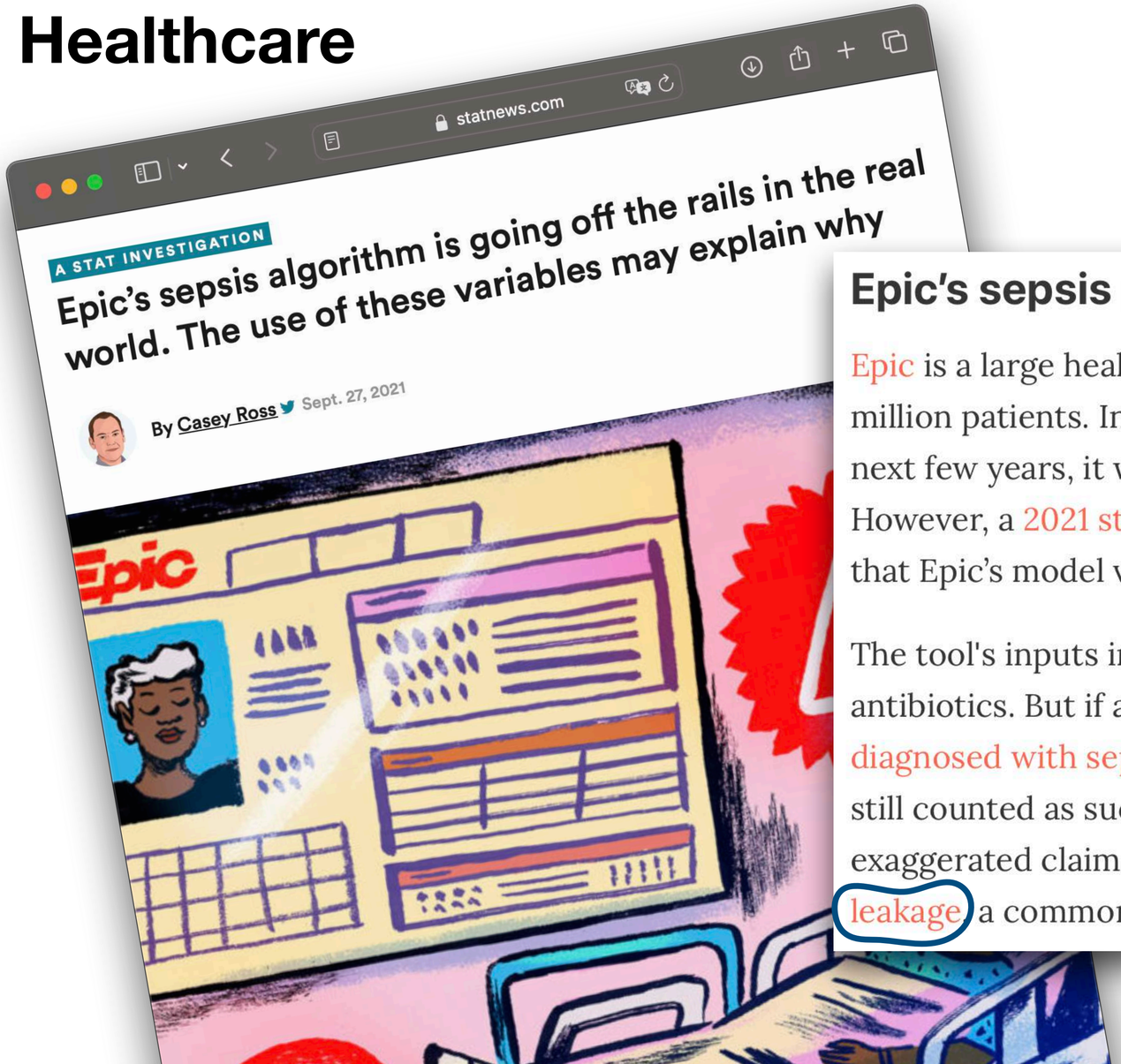
Family separation in Allegheny county

In 2016, Allegheny county in Pennsylvania adopted the Allegheny Family Screening Tool (AFST) to predict which children are at risk of maltreatment. AFST is used to decide which families should be investigated by social workers. In these investigations, social workers can forcibly remove children from their families and place them in foster care, even if there are no allegations of abuse—only poverty-based neglect.

Two years later, it was discovered that AFST suffered from data leakage, leading to exaggerated claims about its performance. In addition, the tool was systematically biased against Black families. When questioned, the creators trotted out the familiar defense that the final decision is always made by a human decision-maker.

Data Leakage in the Real World

Healthcare



https://www.aisnakeoil.com/p/the-bait-and-switch-behind-ai-risk

Epic's sepsis prediction debacle

Epic is a large healthcare software company. It stores health data for over 300 million patients. In 2017, Epic released a sepsis prediction model. Over the next few years, it was deployed in hundreds of hospitals across the U.S. However, a 2021 study from researchers at the University of Michigan found that Epic's model vastly underperformed compared to the developer's claims.

The tool's inputs included information about whether a patient was given antibiotics. But if a patient is given antibiotics, they have already been diagnosed with sepsis—making the tool's prediction useless. These cases were still counted as successes when the developer evaluated the tool, leading to exaggerated claims about how well it performed. This is an example of data leakage, a common error in building AI tools.

Example

[1]:
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

[2]:
```
data = pd.read_csv("data.csv")
X = data[["X_1", "X_2"]]
y = data[["y"]]
```

INPUT DATA READING

[3]:
```
min_max_scaler = MinMaxScaler()
X = min_max_scaler.fit_transform(X)
```

MIN-MAX NORMALIZATION

[4]:
```
X_train , X_test , y_train , y_test = train_test_split(X, y, test_size=0.025, random_state=2)
```

TRAIN/TEST SPLIT

[ ]:

[5]:
```
lr = LogisticRegression()
a = lr.fit(X_train , y_train)
```

TRAINING

[6]:
```
y_pred = lr.predict(X_test)
accuracy_score(y_test , y_pred)
```

TESTING

[6]: 0.67 ✓

Example

```
[1]: import numpy as np
     import pandas as pd
     from sklearn.preprocessing import MinMaxScaler
     from sklearn.model_selection import train_test_split
     from sklearn.linear_model import LogisticRegression
     from sklearn.metrics import accuracy_score

[2]: data = pd.read_csv("data.csv")
     X = data[["X_1", "X_2"]]
     y = data[["y"]]

[ ]:

[3]: X_train , X_test , y_train , y_test = train_test_split(X, y, test_size=0.025, random_state=2)

[4]: min_max_scaler = MinMaxScaler()
     X_train = min_max_scaler.fit_transform(X_train)
     X_test = min_max_scaler.fit_transform(X_test)

[5]: lr = LogisticRegression()
     a = lr.fit(X_train , y_train)

[6]: y_pred = lr.predict(X_test)
     accuracy_score(y_test , y_pred)

[6]: 0.33
```

INPUT DATA READING

TRAIN/TEST SPLIT

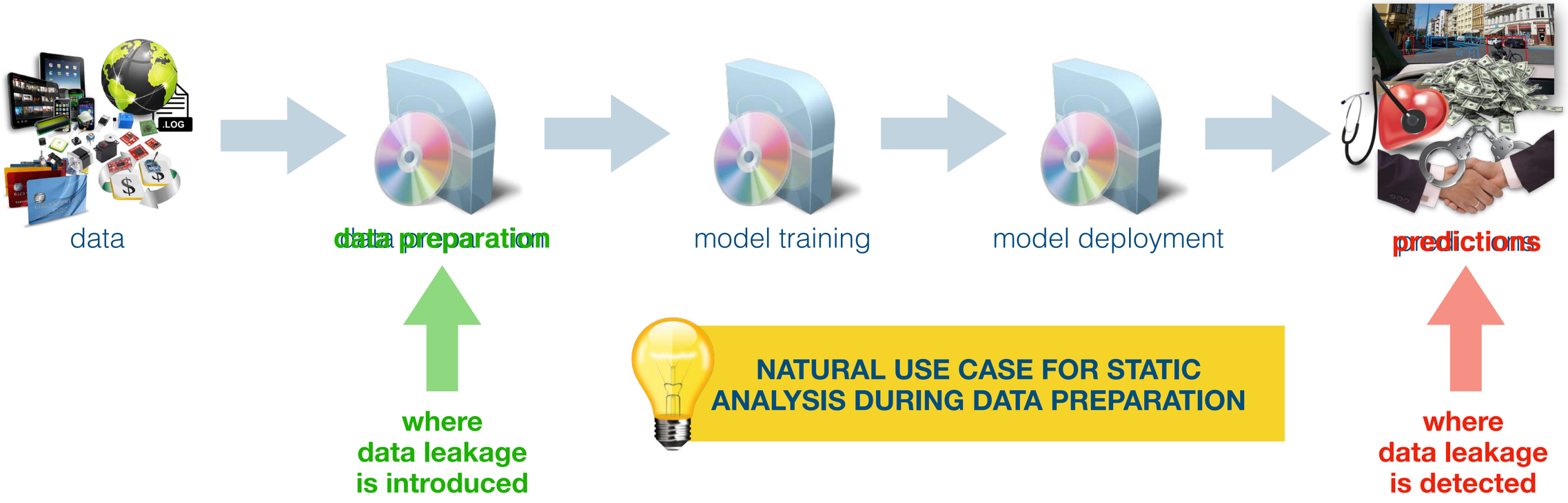
MIN-MAX NORMALIZATION

TRAINING

TESTING

Machine Learning Development Process

Machine Learning Pipeline

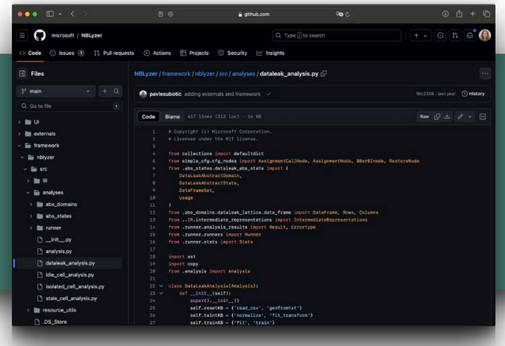


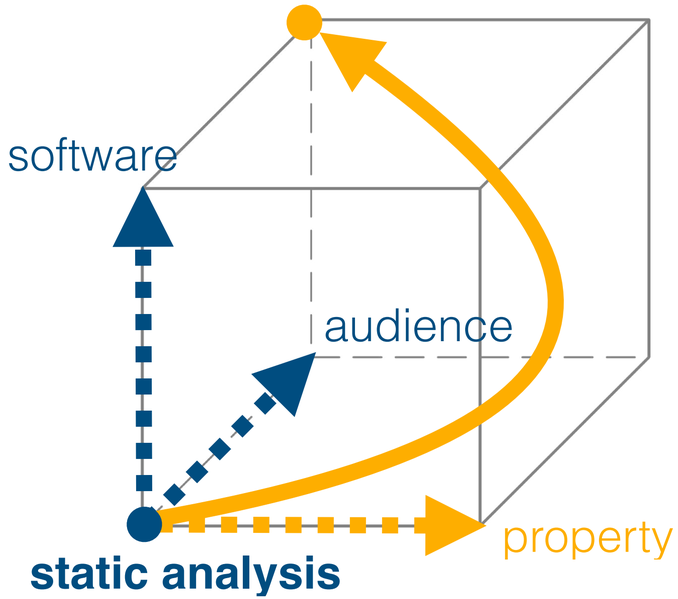
Data Leakage Static Analysis

NBLyzer

SOFTWARE ENGINEERING

practical tools
targeting specific programs





Data Leakage Static Analysis

This Paper: How To Get It Right

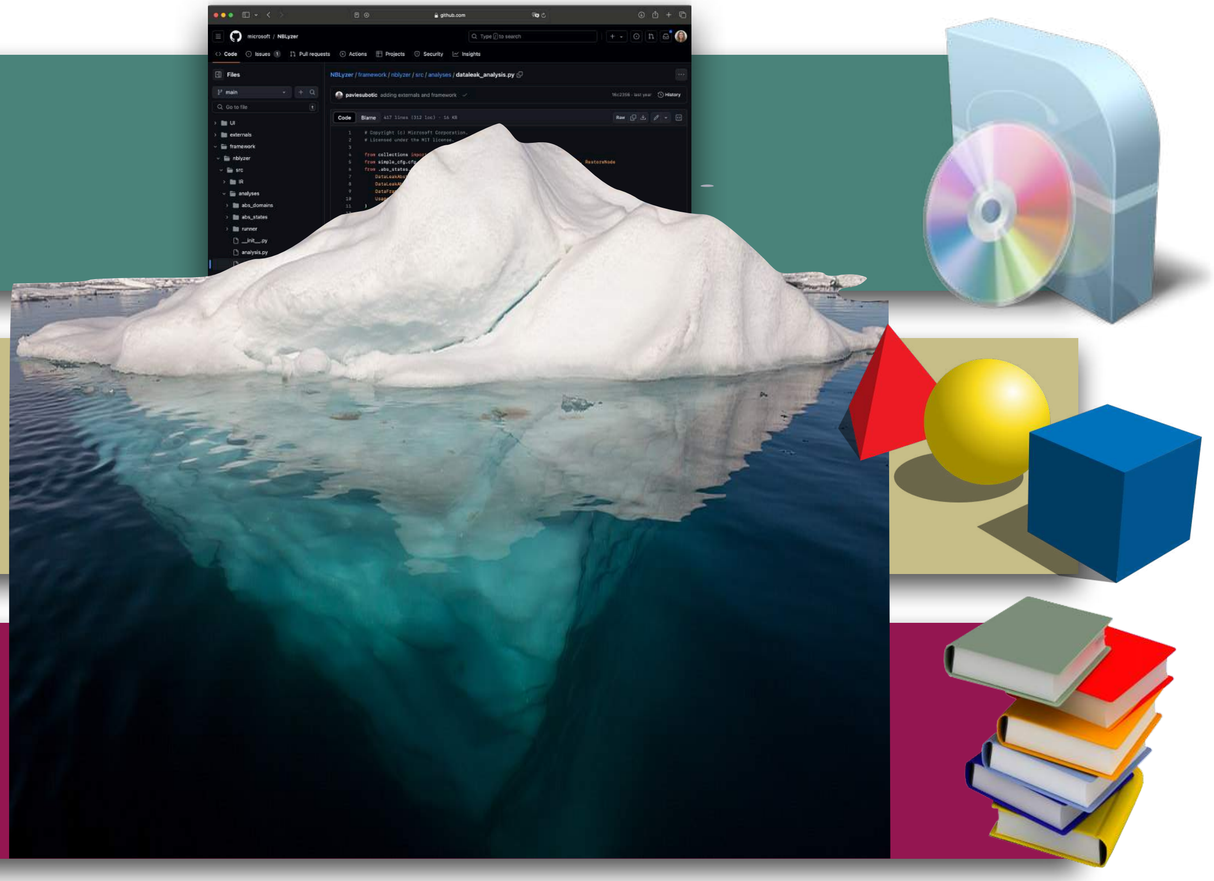
SOFTWARE ENGINEERING

practical tools
targeting specific programs

THEORETICAL ASPECTS

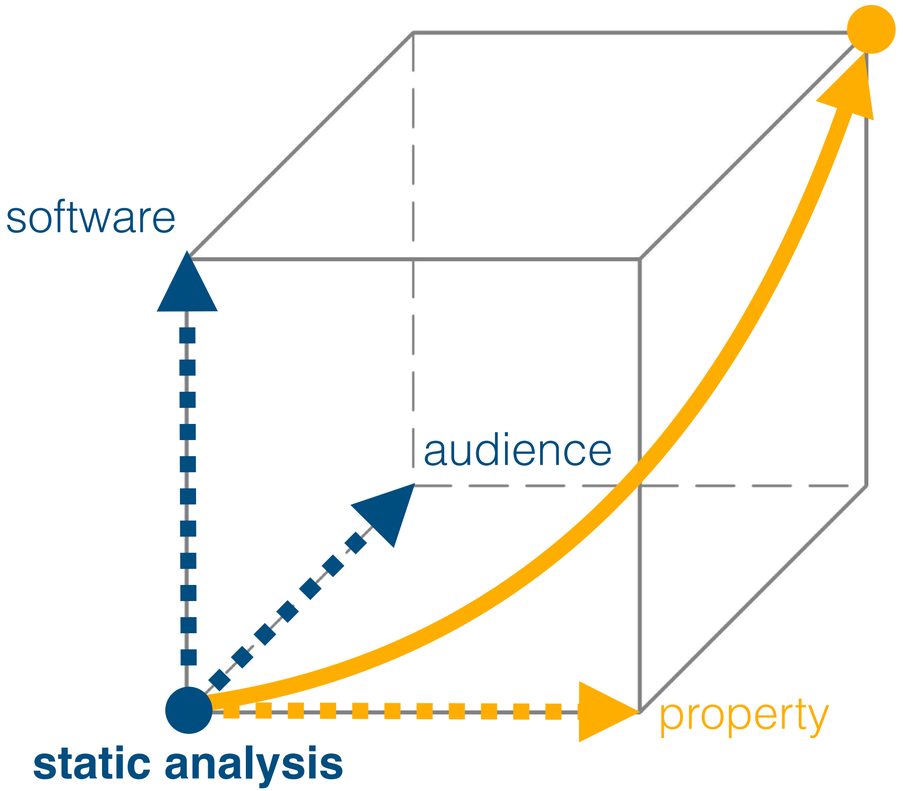
algorithmic approaches
to decide program properties

mathematical models
of the program behavior



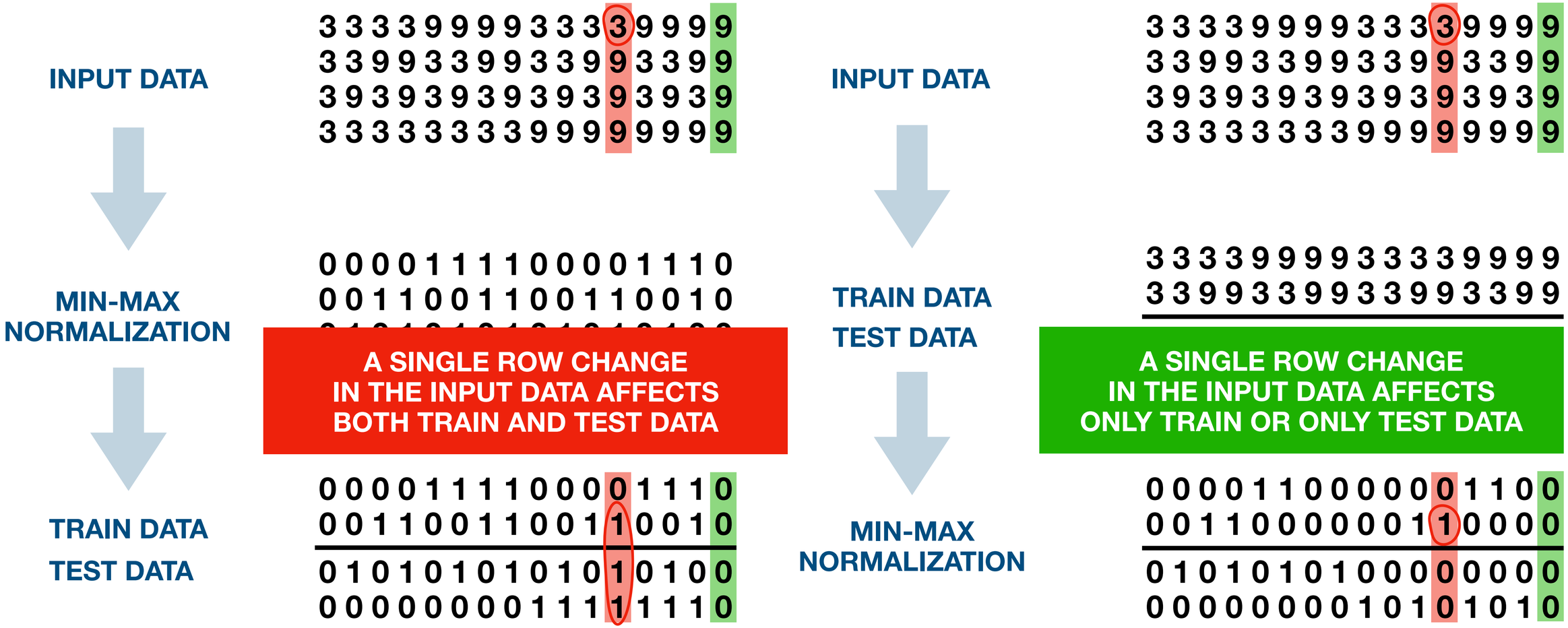
(Absence of) Data Leakage

Hyperproperty: Independence of Training and Testing Data



(Absence of) Data Leakage

Hyperproperty: Independence of Training and Testing Data



Data Leakage Static Analysis

Concrete Semantics

SOFTWARE ENGINEERING

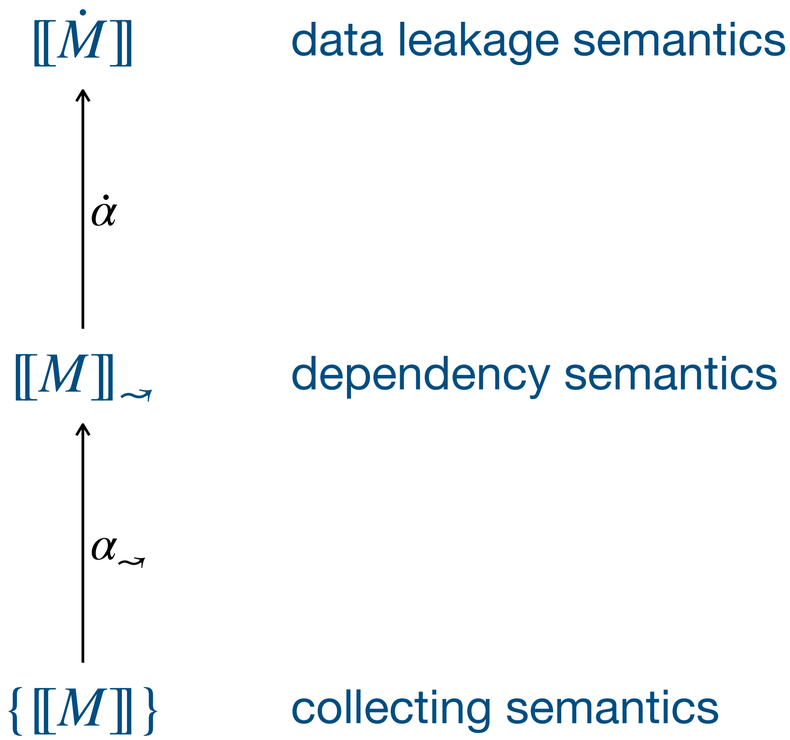
practical tools
targeting specific programs

THEORETICAL ASPECTS

algorithmic approaches
to decide program properties

mathematical models
of the program behavior

Hierarchy of Semantics



Hierarchy of Semantics

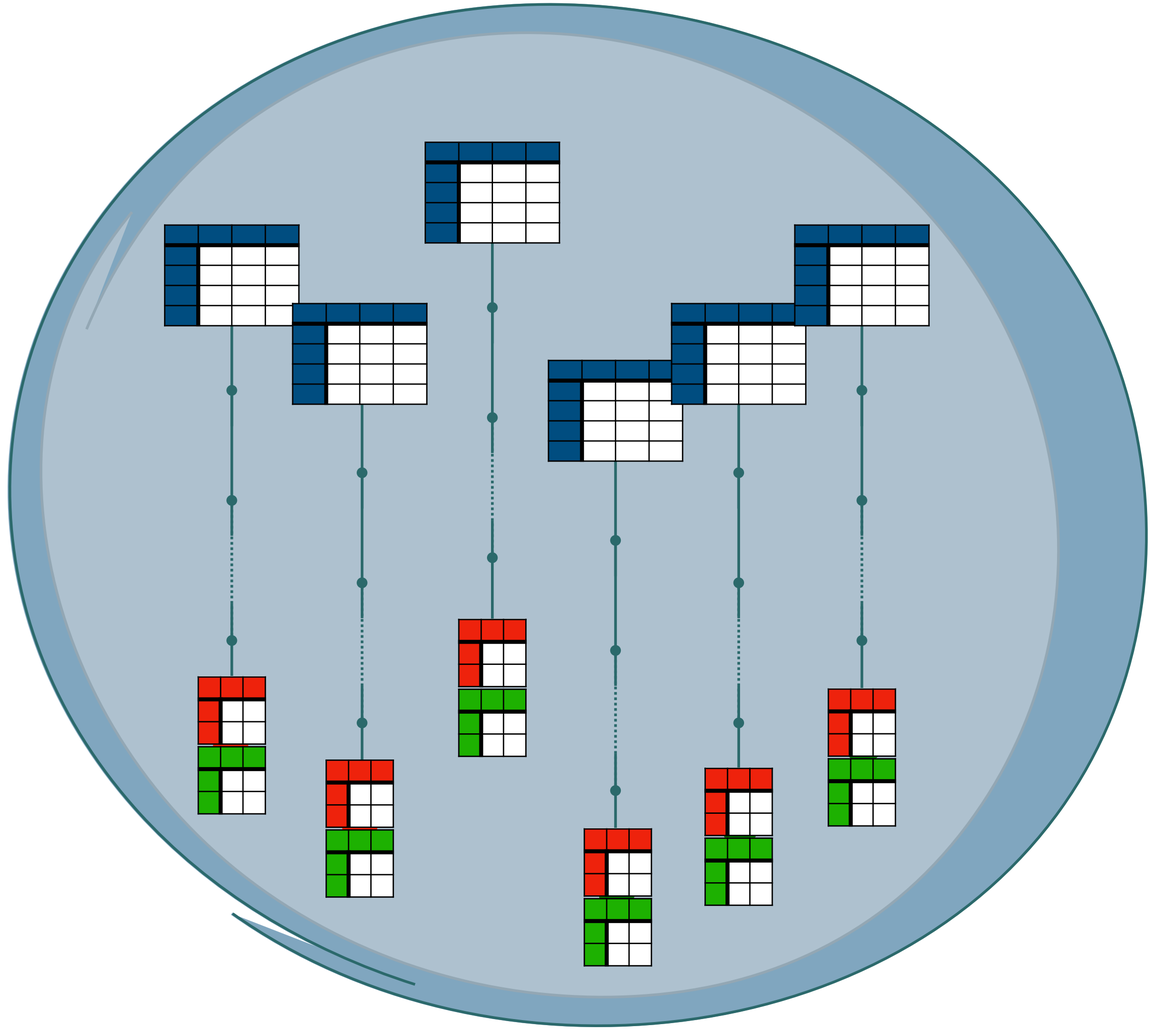
$[\![\dot{M}]\!]$ data leakage semantics

$\dot{\alpha}$

$[\![M]\!]_{\rightsquigarrow}$ dependency semantics

$\alpha_{\rightsquigarrow}$

$\{[\![M]\!]\}$ collecting semantics



Hierarchy of Semantics

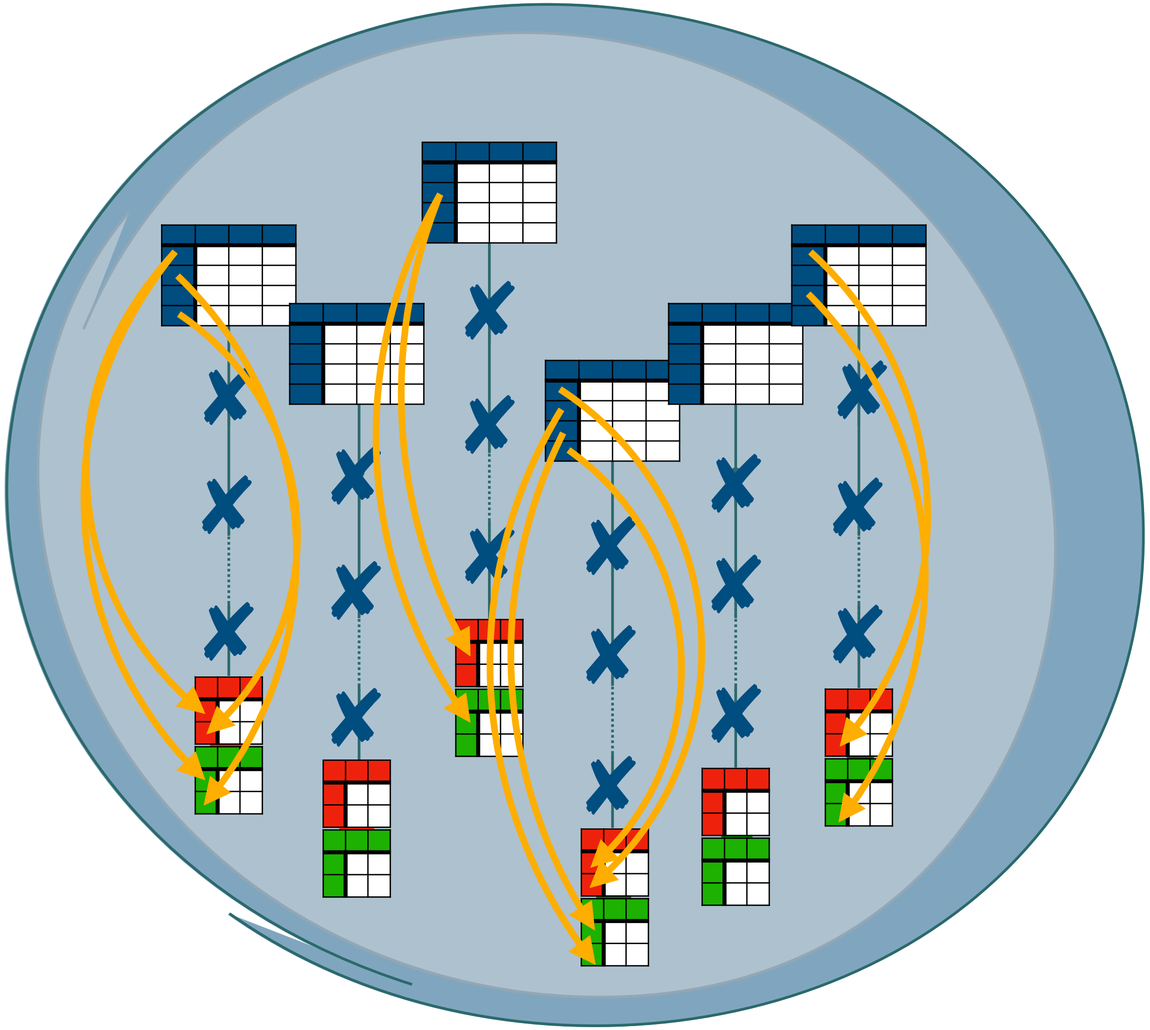
$[\![\dot{M}]\!]$ data leakage semantics

$\dot{\alpha}$

$[\![M]\!]_{\rightsquigarrow}$ dependency semantics

$\alpha_{\rightsquigarrow}$

$\{[\![M]\!]\}$ collecting semantics



Hierarchy of Semantics

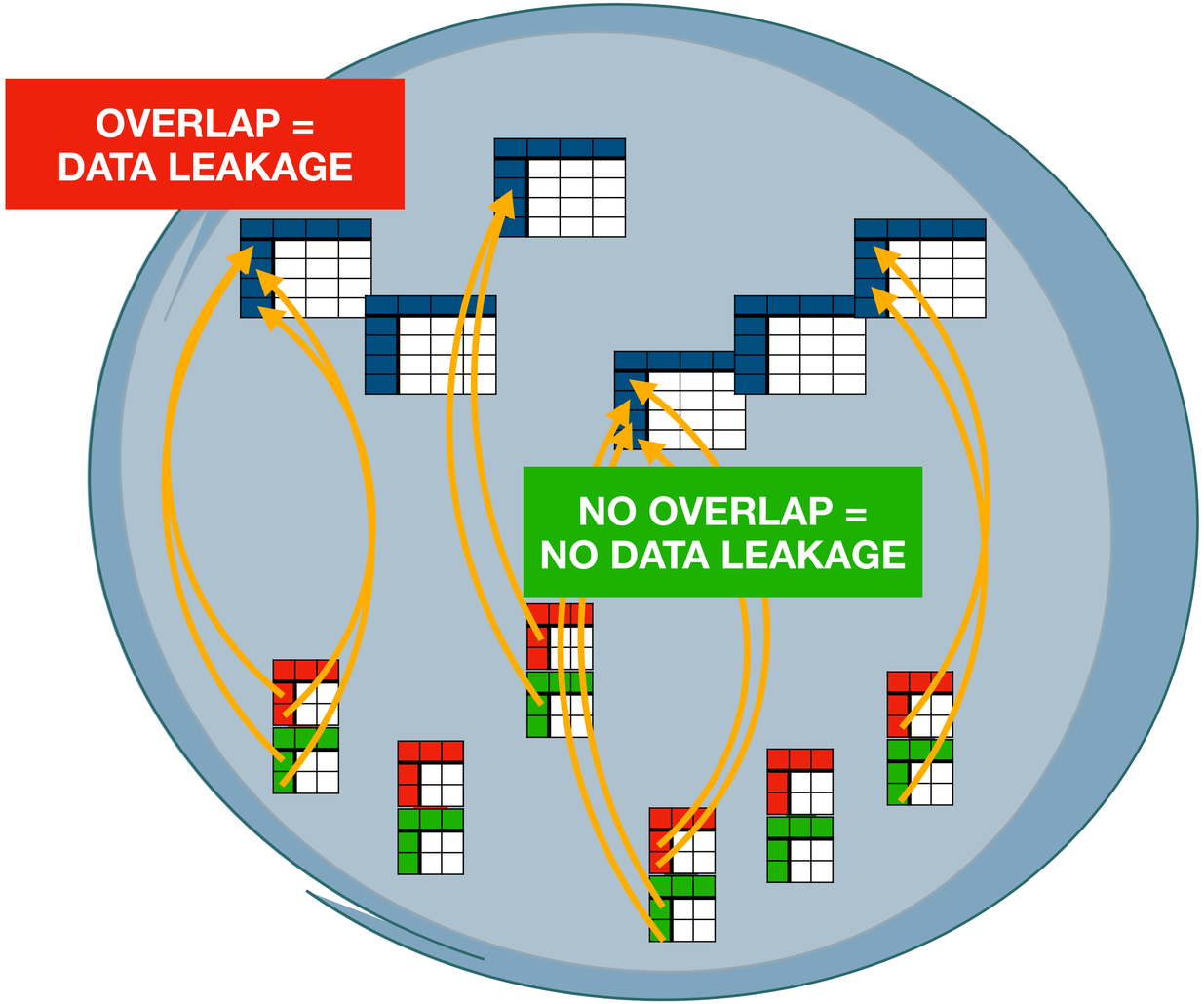
$[\![\dot{M}]\!]$ data leakage semantics

$\dot{\alpha}$

$[\![M]\!]_{\rightsquigarrow}$ dependency semantics

$\alpha_{\rightsquigarrow}$

$\{[\![M]\!]\}$ collecting semantics



Data Leakage Static Analysis

Abstract Semantics

SOFTWARE ENGINEERING

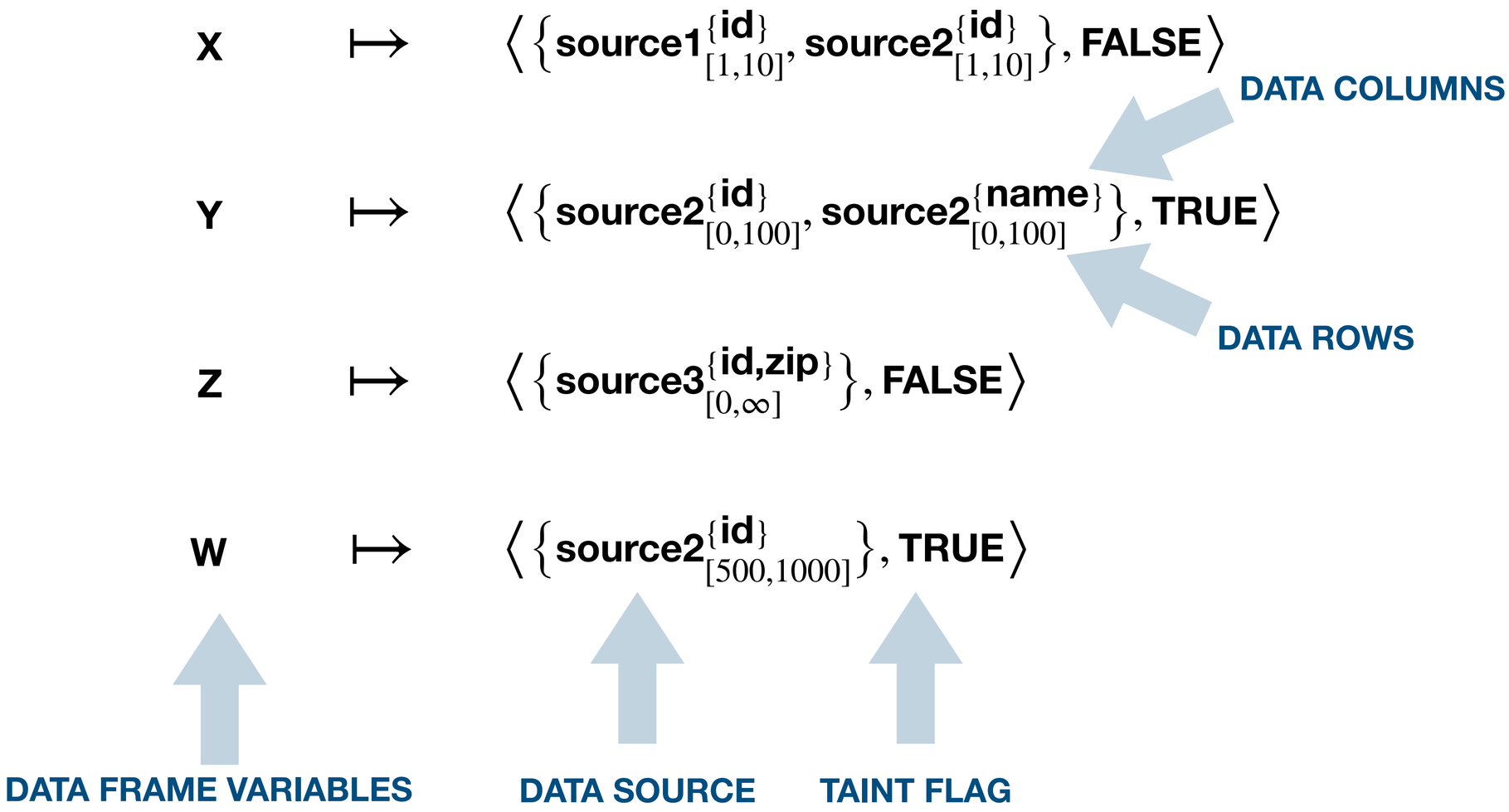
THEORETICAL ASPECTS

practical tools
targeting specific programs

algorithmic approaches
to decide program properties

mathematical models
of the program behavior

Data Frame Sources Abstract Domain



Example

```
[1]: import numpy as np
     import pandas as pd
     from sklearn.preprocessing import MinMaxScaler
     from sklearn.model_selection import train_test_split
     from sklearn.linear_model import LogisticRegression
     from sklearn.metrics import accuracy_score

[2]: data = pd.read_csv("data.csv")
     X = data[["X_1", "X_2"]]
     y = data[["y"]]

[3]: min_max_scaler = MinMaxScaler()
     X = min_max_scaler.fit_transform(X)

[4]: X_train , X_test , y_train , y_test = train_test_split(X, y, test_size=0.025, random_state=2)

[ ]:

[5]: lr = LogisticRegression()
     a = lr.fit(X_train , y_train)

[6]: y_pred = lr.predict(X_test)
     accuracy_score(y_test , y_pred)

[6]: 0.67
```

INPUT DATA READING

$$X \mapsto \left\langle \left\{ \text{data}_{[0,\infty]}^{\{X_1,X_2\}} \right\}, \text{FALSE} \right\rangle$$

MIN-MAX NORMALIZATION

$$X \mapsto \left\langle \left\{ \text{data}_{[0,\infty]}^{\{X_1,X_2\}} \right\}, \text{TRUE} \right\rangle$$

TRAIN/TEST SPLIT

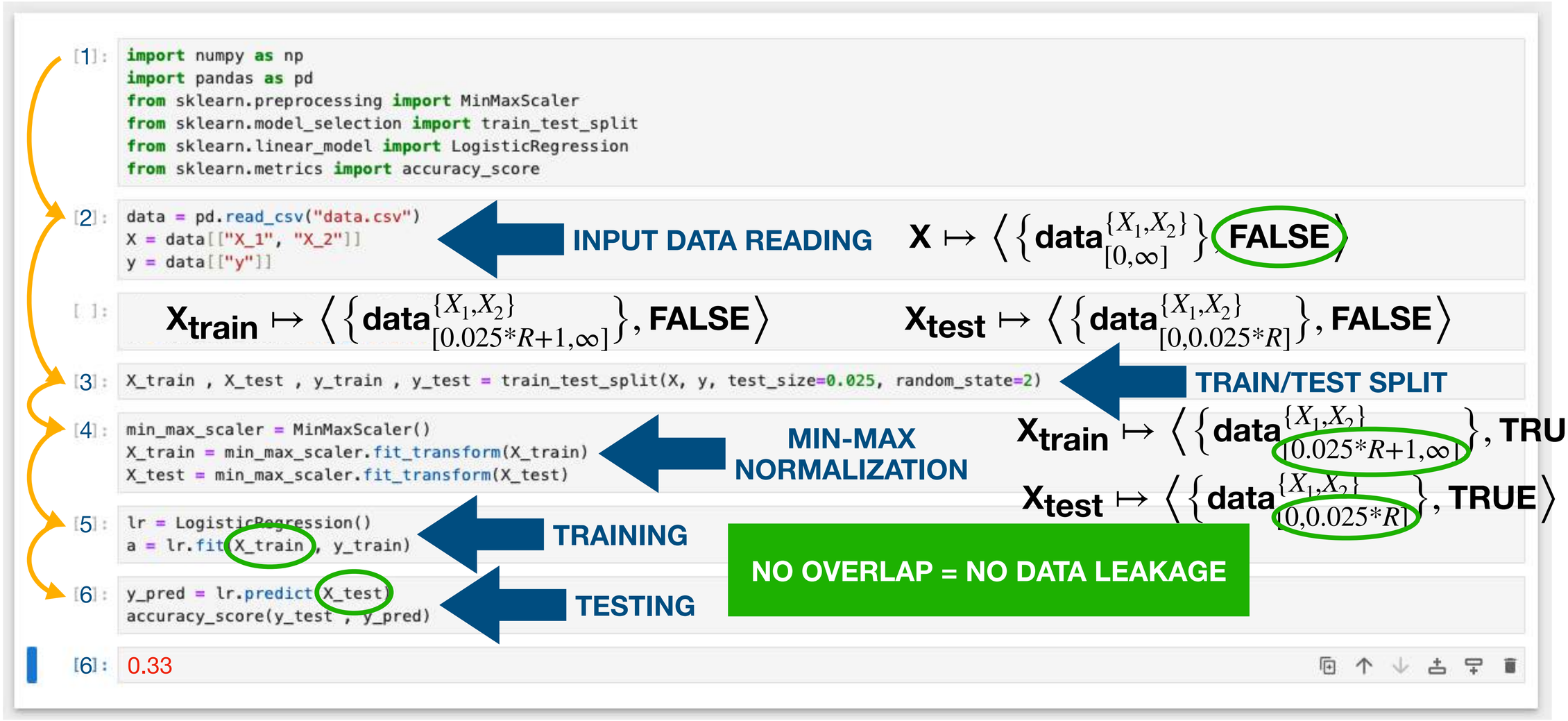
$$X_{\text{train}} \mapsto \left\langle \left\{ \text{data}_{[0,\infty]}^{\{X_1,X_2\}} \right\}, \text{TRUE} \right\rangle \qquad X_{\text{test}} \mapsto \left\langle \left\{ \text{data}_{[0,\infty]}^{\{X_1,X_2\}} \right\}, \text{TRUE} \right\rangle$$

TRAINING

TESTING

(TAINT) OVERLAP = DATA LEAKAGE

Example


[1]:
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
[2]:
data = pd.read_csv("data.csv")
X = data[["X_1", "X_2"]]
y = data[["y"]]
INPUT DATA READING
$X \mapsto \langle \{ data^{\{X_1,X_2\}}_{[0,\infty]} \}, FALSE \rangle$
[ ]:
$X_{train} \mapsto \langle \{ data^{\{X_1,X_2\}}_{[0.025*R+1,\infty]} \}, FALSE \rangle$
$X_{test} \mapsto \langle \{ data^{\{X_1,X_2\}}_{[0,0.025*R]} \}, FALSE \rangle$
[3]:
X_train , X_test , y_train , y_test = train_test_split(X, y, test_size=0.025, random_state=2)
TRAIN/TEST SPLIT
[4]:
min_max_scaler = MinMaxScaler()
X_train = min_max_scaler.fit_transform(X_train)
X_test = min_max_scaler.fit_transform(X_test)
MIN-MAX NORMALIZATION
$X_{train} \mapsto \langle \{ data^{\{X_1,X_2\}}_{[0.025*R+1,\infty]} \}, TRUE \rangle$
$X_{test} \mapsto \langle \{ data^{\{X_1,X_2\}}_{[0,0.025*R]} \}, TRUE \rangle$
[5]:
lr = LogisticRegression()
a = lr.fit(X_train , y_train)
TRAINING
[6]:
y_pred = lr.predict(X_test)
accuracy_score(y_test , y_pred)
TESTING
NO OVERLAP = NO DATA LEAKAGE
[6]:
0.33

Data Leakage Static Analysis

Implementation

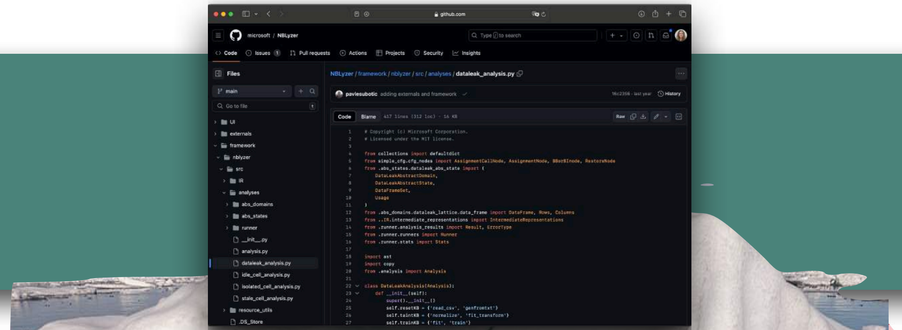
SOFTWARE ENGINEERING

practical tools
targeting specific programs

THEORETICAL ASPECTS

algorithmic approaches
to decide program properties

mathematical models
of the program behavior



Experimental Evaluation

7378 Executions in 2111 Notebooks from Kaggle

Implementation	True Positives		False Positives
	Taint Data Leakage	Overlap Data Leakage	
NBLyzer + Original Data Leakage Analysis	10	0	2
NBLyzer + Our Data Leakage Analysis	**10**	**15**	**2**

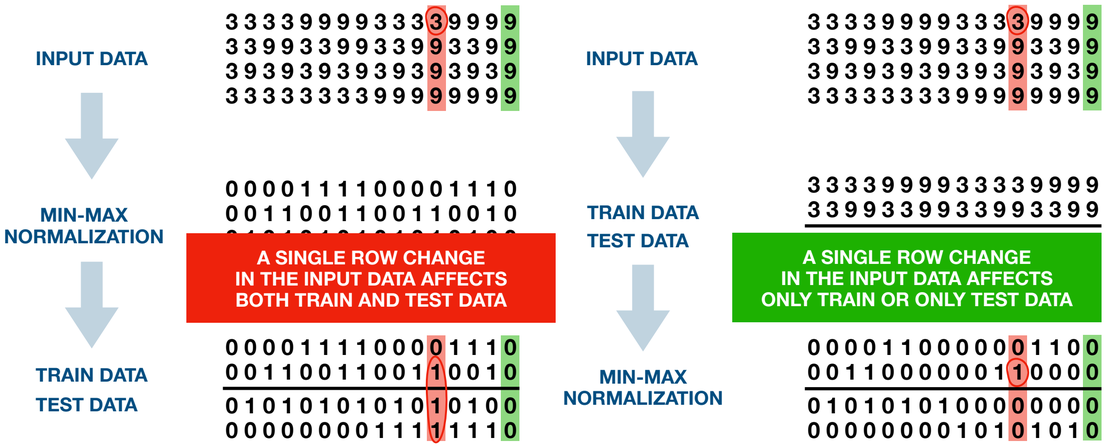
IN 5 NOTEBOOKS

IN 11 NOTEBOOKS

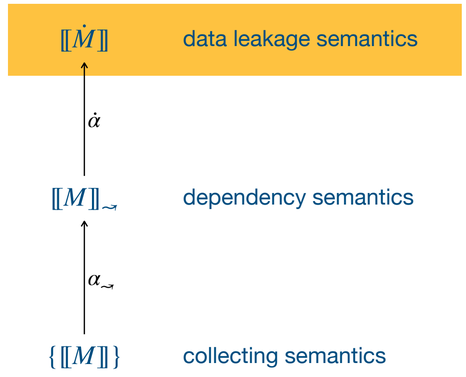
CONFIRMED BY 4 DATA SCIENTISTS AT MICROSOFT

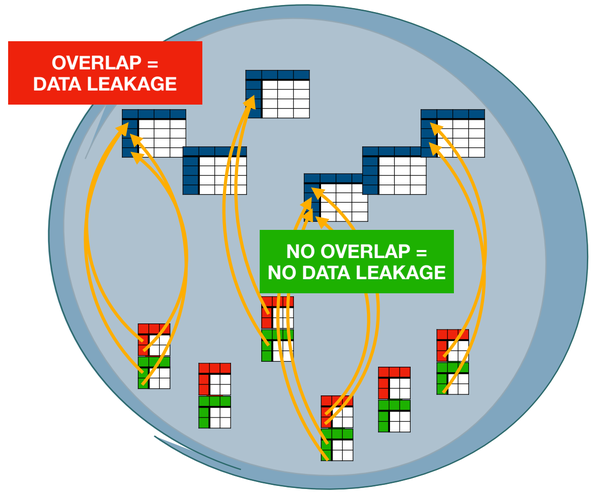
(Absence of) Data Leakage

Hyperproperty: Independence of Training and Testing Data



Hierarchy of Semantics





Data Frame Sources Abstract Domain

$$X \mapsto \langle \{ \text{source1}^{\{\text{id}\}}_{[1,10]}, \text{source2}^{\{\text{id}\}}_{[1,10]} \}, \text{FALSE} \rangle$$

$$Y \mapsto \langle \{ \text{source2}^{\{\text{id}\}}_{[0,100]}, \text{source2}^{\{\text{name}\}}_{[0,100]} \}, \text{TRUE} \rangle$$

$$Z \mapsto \langle \{ \text{source3}^{\{\text{id,zip}\}}_{[0,\infty]} \}, \text{FALSE} \rangle$$

$$W \mapsto \langle \{ \text{source2}^{\{\text{id}\}}_{[500,1000]} \}, \text{TRUE} \rangle$$

DATA COLUMNS

DATA ROWS

DATA FRAME VARIABLES

DATA SOURCE

TAINT FLAG

Experimental Evaluation

7378 Executions in 2111 Notebooks from Kaggle

Implementation	True Positives		False Positives
	Taint Data Leakage	Overlap Data Leakage	
NBLyzer + Original Data Leakage Analysis	10	0	2
NBLyzer + Our Data Leakage Analysis	**10**	**15**	**2**

IN 5 NOTEBOOKS

IN 11 NOTEBOOKS

CONFIRMED BY 4 DATA SCIENTI… AT MICRO…

THANKS!