IRTF

Applied Networking Research Prize

AI/ML for Network Security: The Emperor has no Clothes

Arthur Selle Jacobs[1]

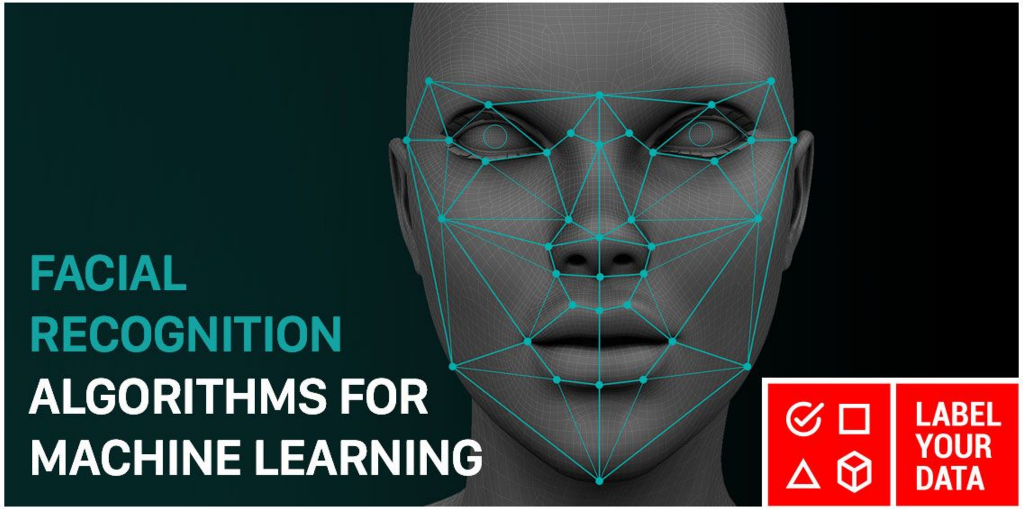
Roman Beltiukov[4] Walter Willinger[2] Ronaldo A. Ferreira[3]

Arpit Gupta[4] Lisandro Z. Granville[1]

March 27th, 2023

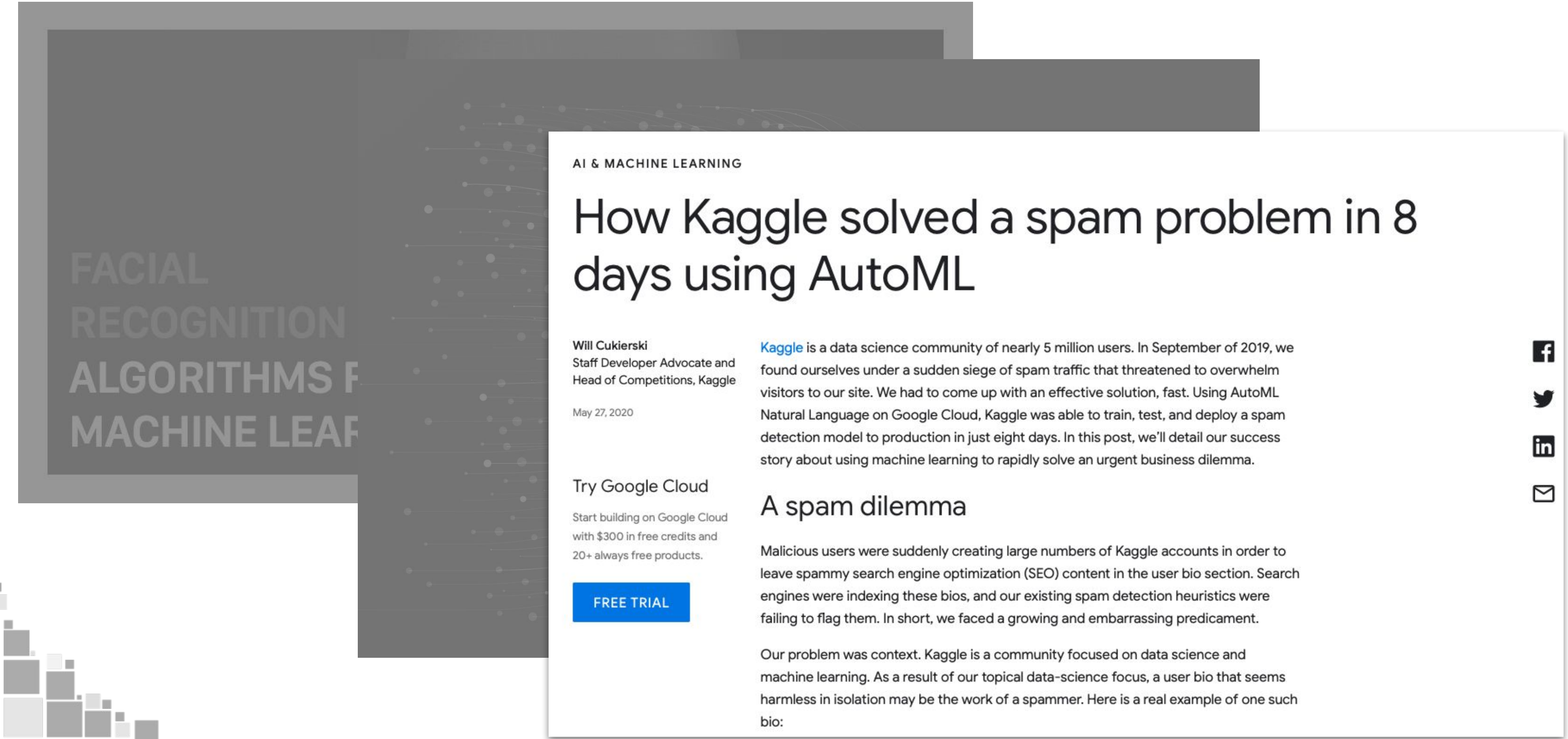
1 .inf INSTITUTO DE INFORMÁTICA UFRGS 2 NIKSUN 3 UFMS 4 University of California Santa Barbara

The Rise of AI



The Rise of AI

The Rise of AI



AI & MACHINE LEARNING

How Kaggle solved a spam problem in 8 days using AutoML

Will Cukierski
Staff Developer Advocate and Head of Competitions, Kaggle

May 27, 2020

Try Google Cloud

Start building on Google Cloud with $300 in free credits and 20+ always free products.

FREE TRIAL

Kaggle is a data science community of nearly 5 million users. In September of 2019, we found ourselves under a sudden siege of spam traffic that threatened to overwhelm visitors to our site. We had to come up with an effective solution, fast. Using AutoML Natural Language on Google Cloud, Kaggle was able to train, test, and deploy a spam detection model to production in just eight days. In this post, we'll detail our success story about using machine learning to rapidly solve an urgent business dilemma.

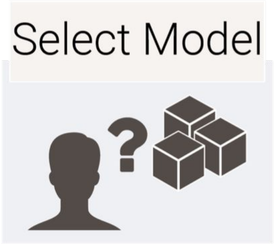
A spam dilemma

Malicious users were suddenly creating large numbers of Kaggle accounts in order to leave spammy search engine optimization (SEO) content in the user bio section. Search engines were indexing these bios, and our existing spam detection heuristics were failing to flag them. In short, we faced a growing and embarrassing predicament.

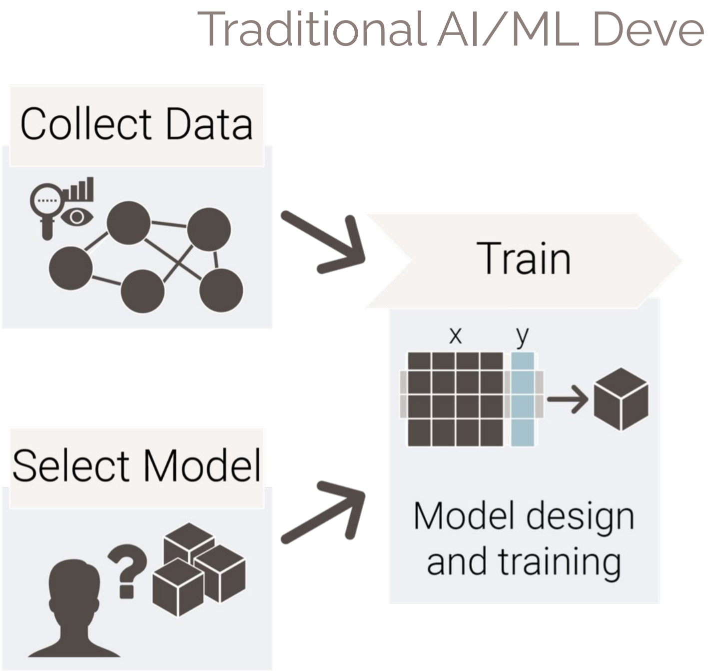
Our problem was context. Kaggle is a community focused on data science and machine learning. As a result of our topical data-science focus, a user bio that seems harmless in isolation may be the work of a spammer. Here is a real example of one such bio:

How does it work?

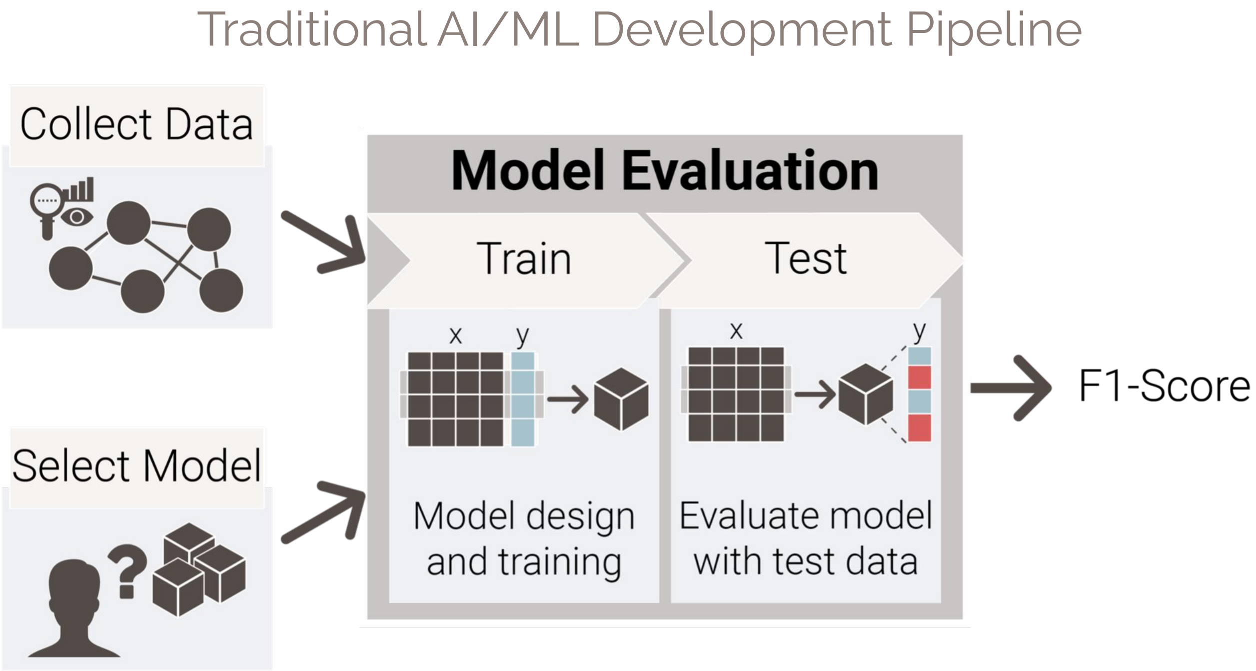
Traditional AI/ML Development Pipeline



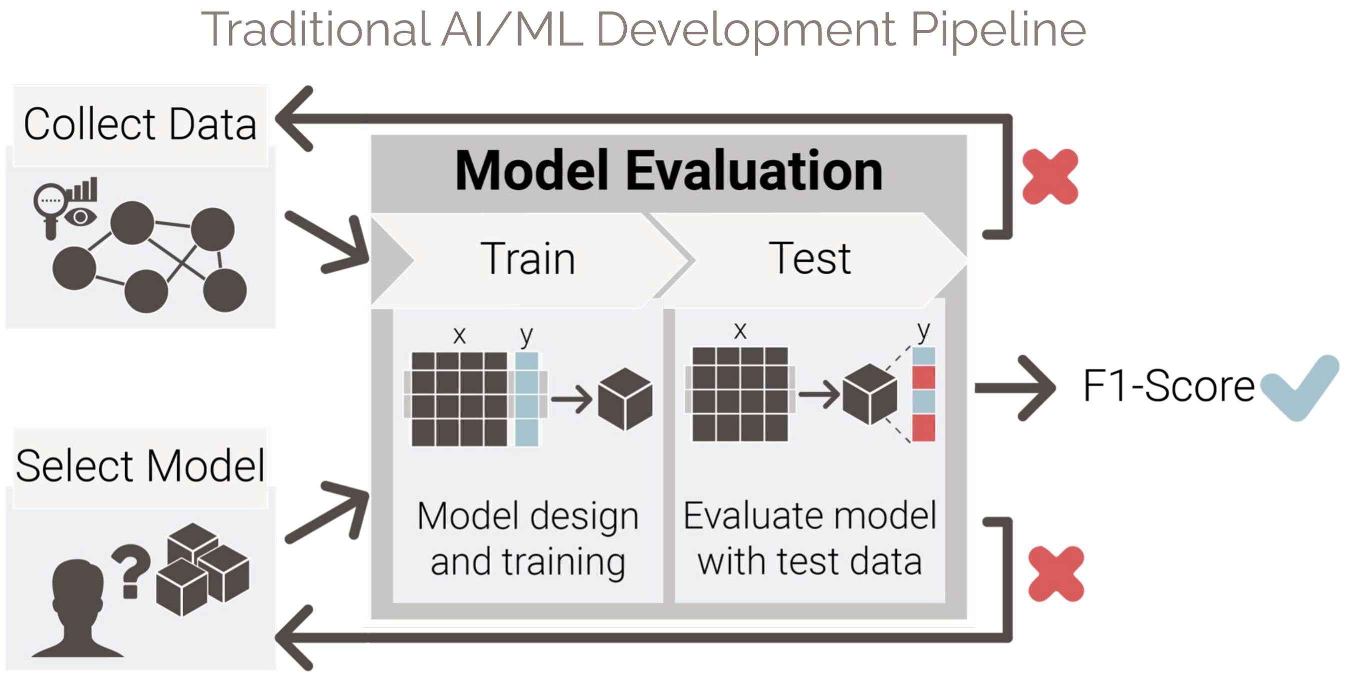
How does it work?



How does it work?



How does it work?



What about high-stakes decision making?

Why (and how) does the model work?

Self-driving Cars

When does the model not work?

Network Security

Underspecification issues!

Shortcut Learning

Model takes shortcuts to classify data!

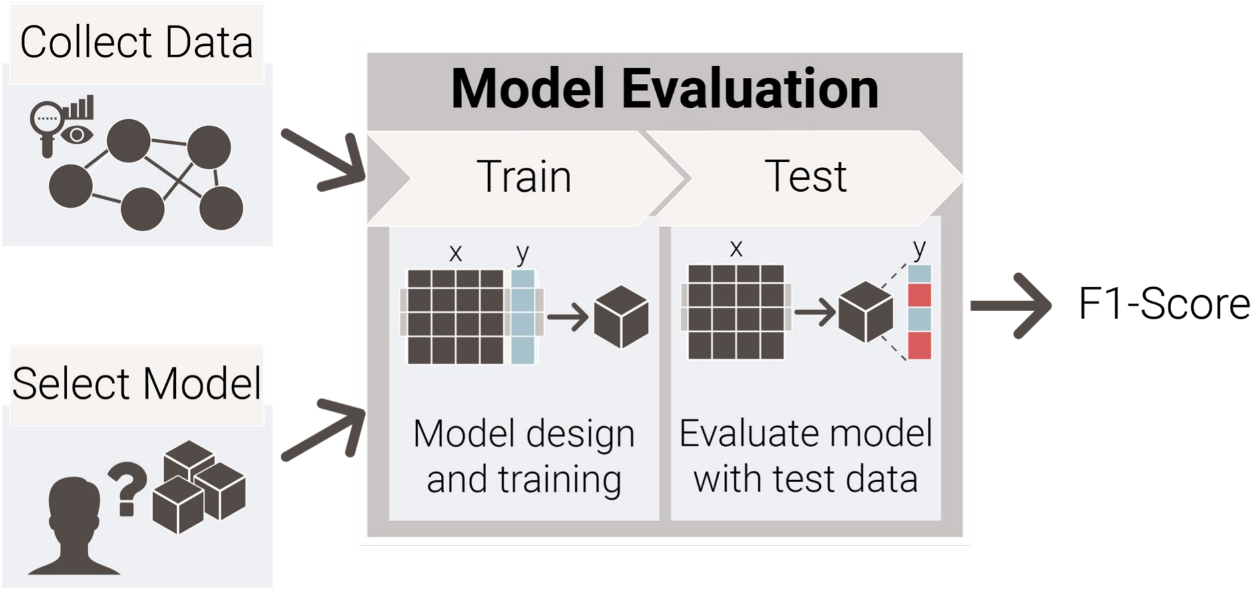
O.O.D. Samples

Model does not generalize!

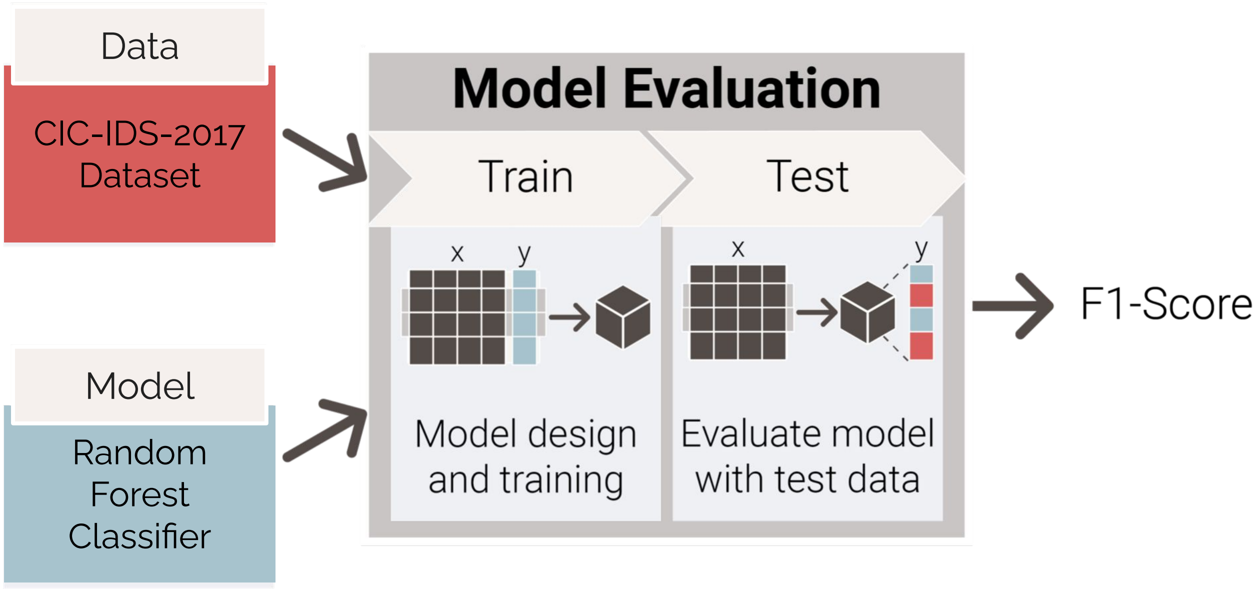
Spurious Correlations

Model picks up wrong correlations in the data!

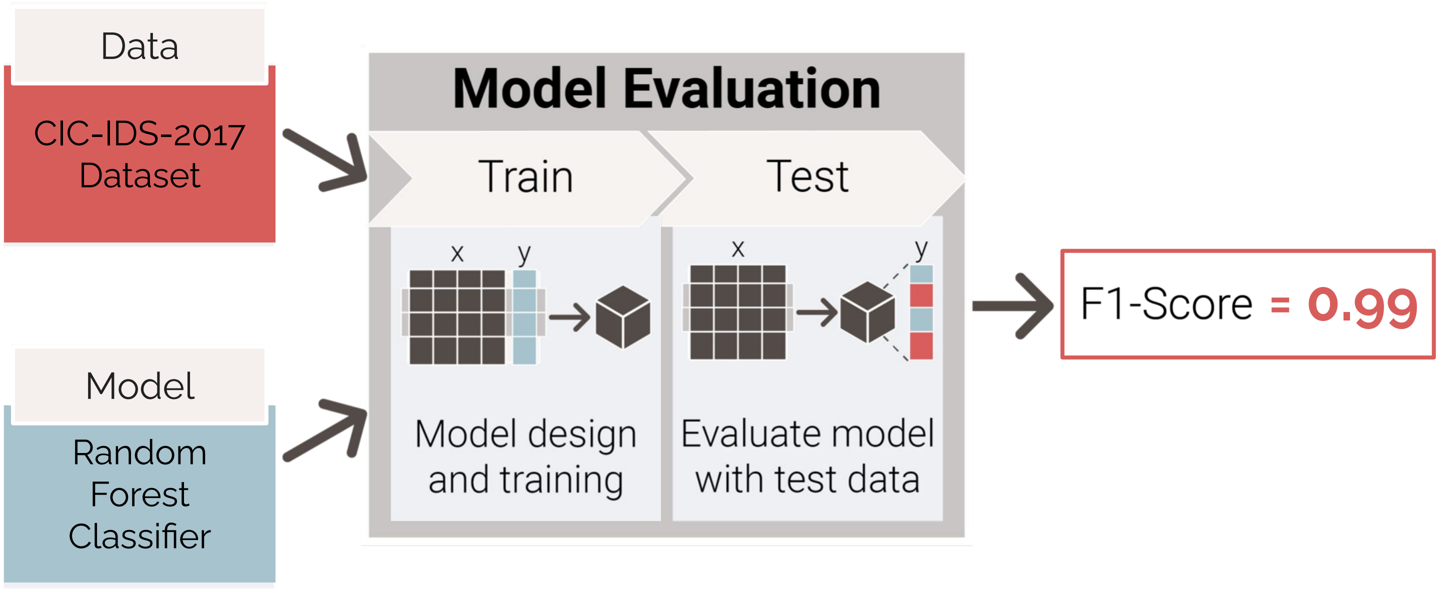
Consider this example...



Consider this example…



Consider this example…



Can you answer these questions?

Why (and how) does the model work?

When does the model not work?

Can you answer these questions?

Why (and how) does the model work?

When does the model not work?

Can you trust this model?

Can you trust this model?

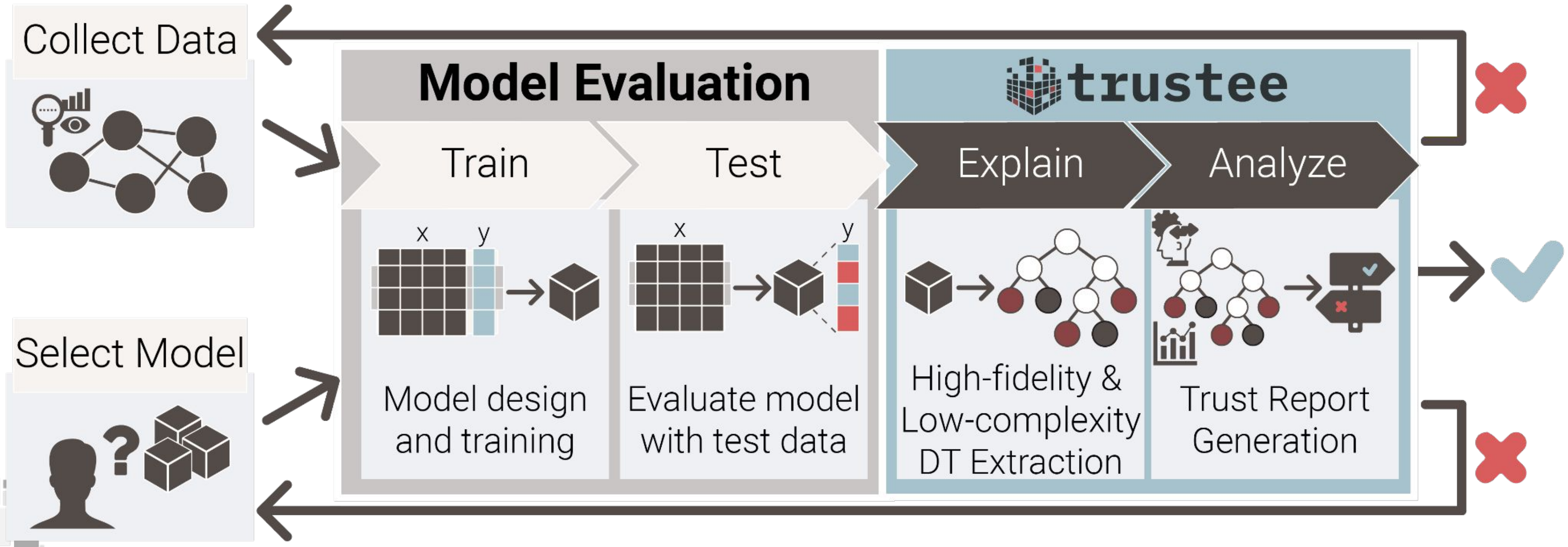
Trust in AI/ML model

=

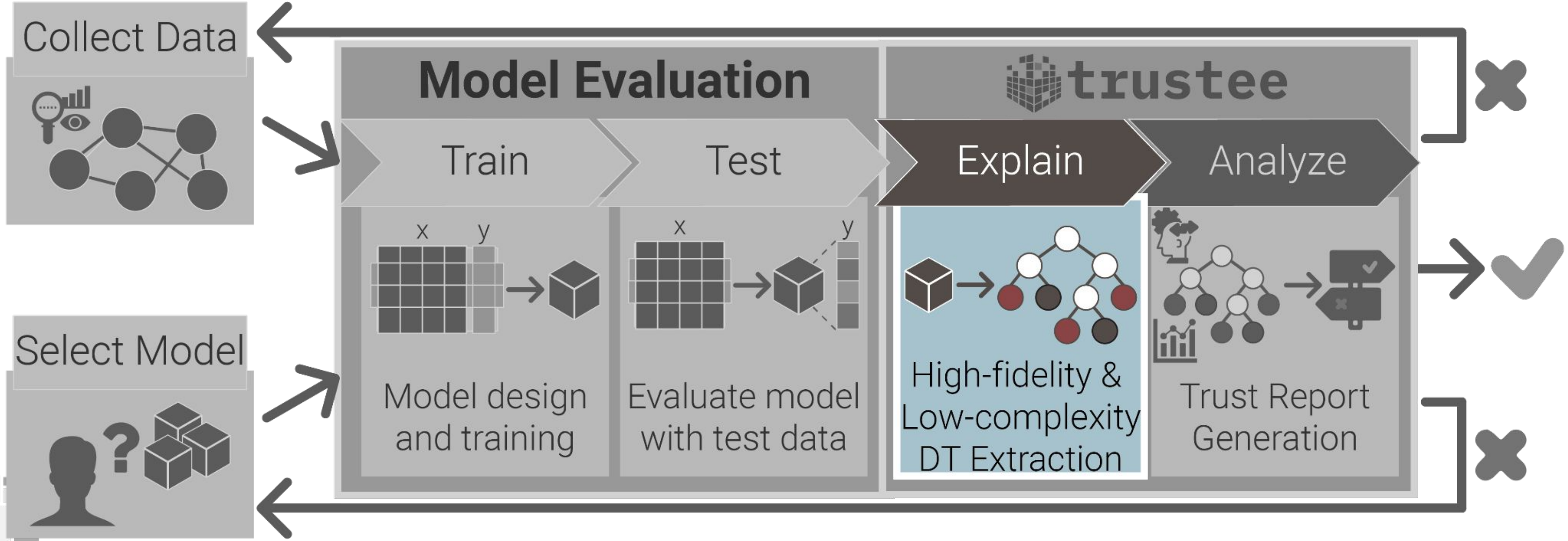
Hand over control to the AI/ML model

trustee

Augmented
AI/ML Development Pipeline



Augmented
AI/ML Development Pipeline



trustee

Explanation Requirements

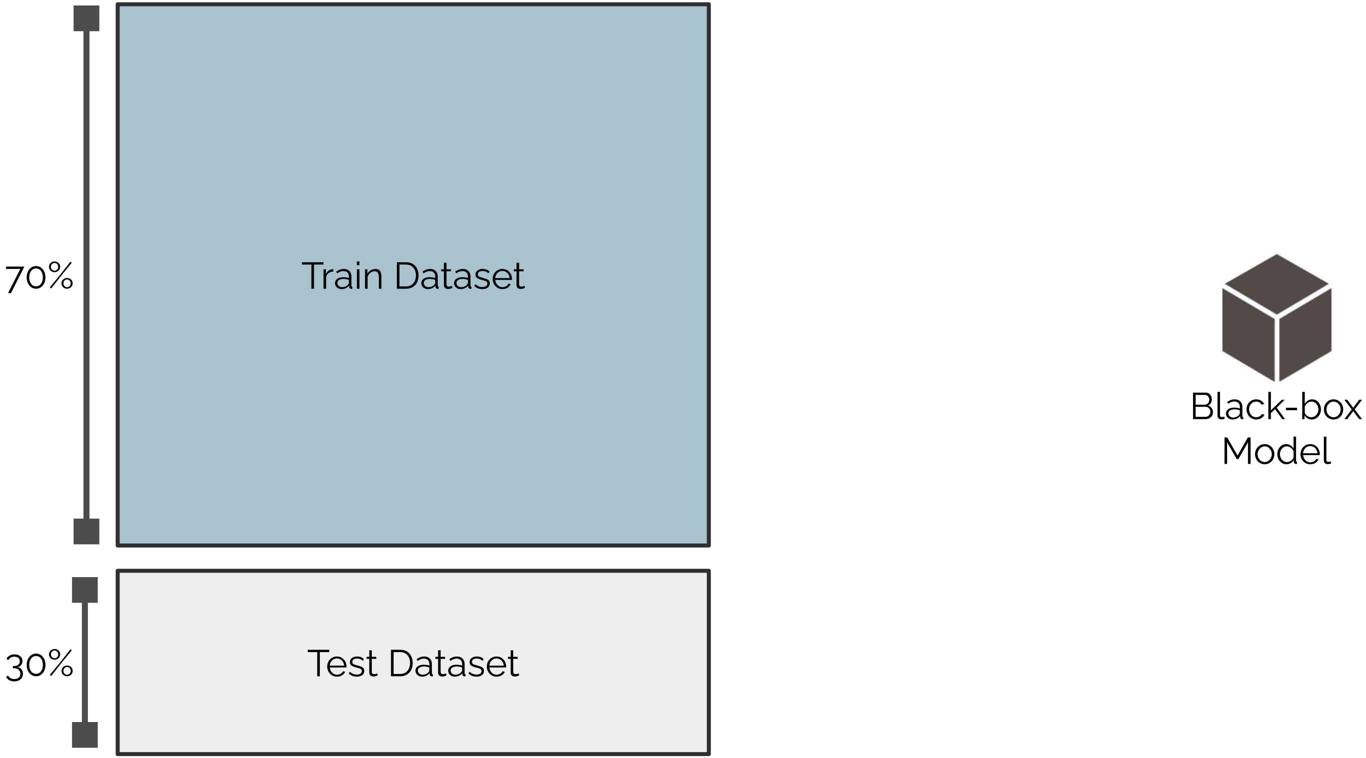
#1 Model Agnostic	#2 High Fidelity
#3 Low Complexity	#4 Stable

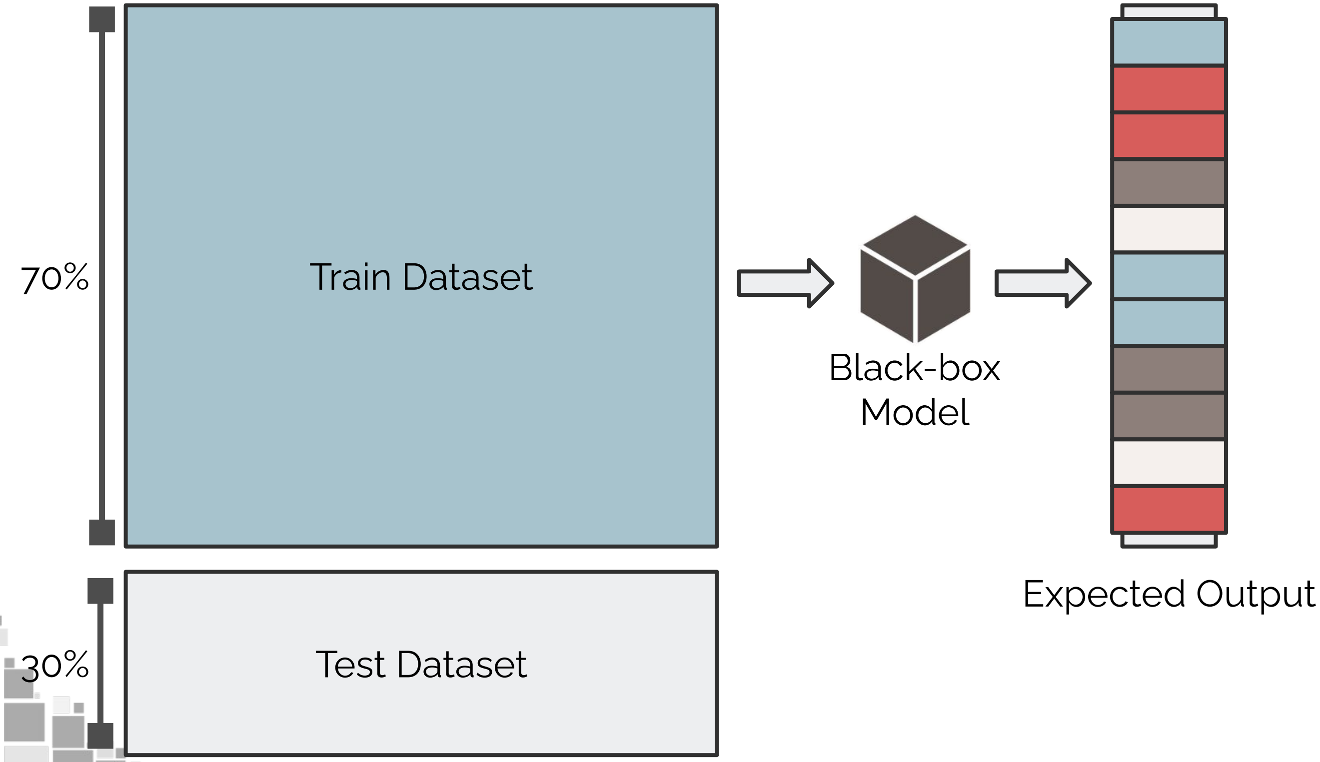
trustee

Dataset

Black-box
Model

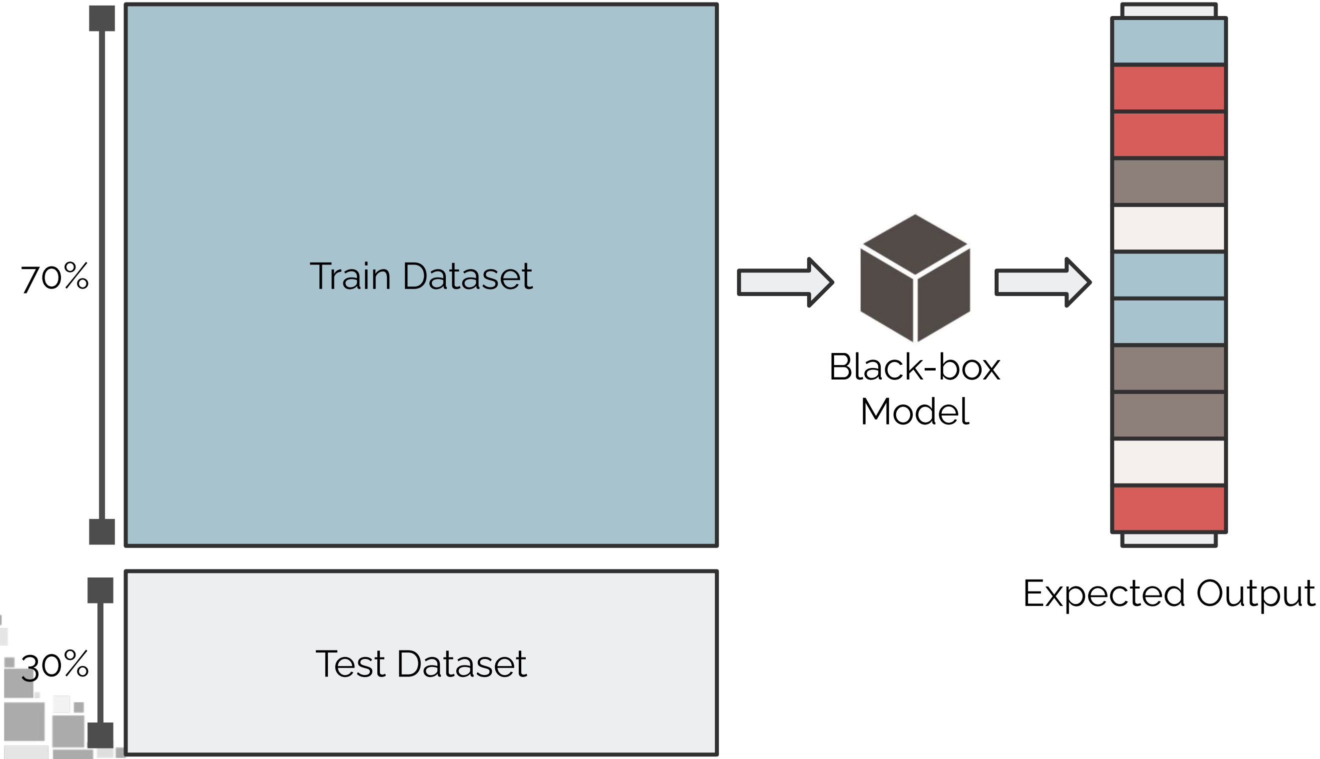
trustee

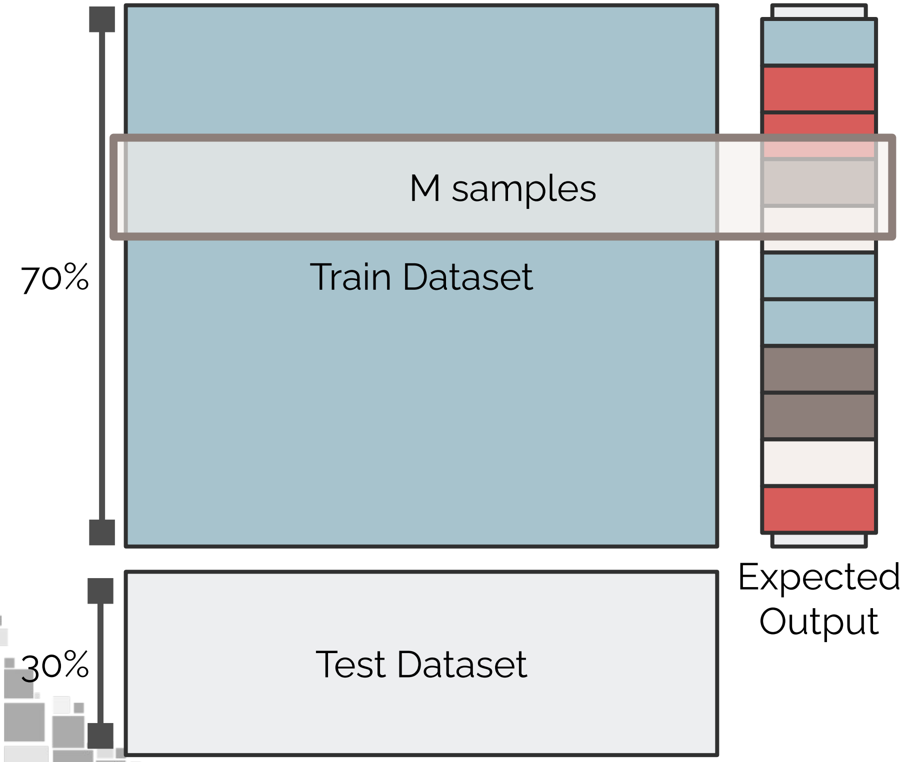



70%
Train Dataset
30%
Test Dataset
Black-box
Model
Expected Output

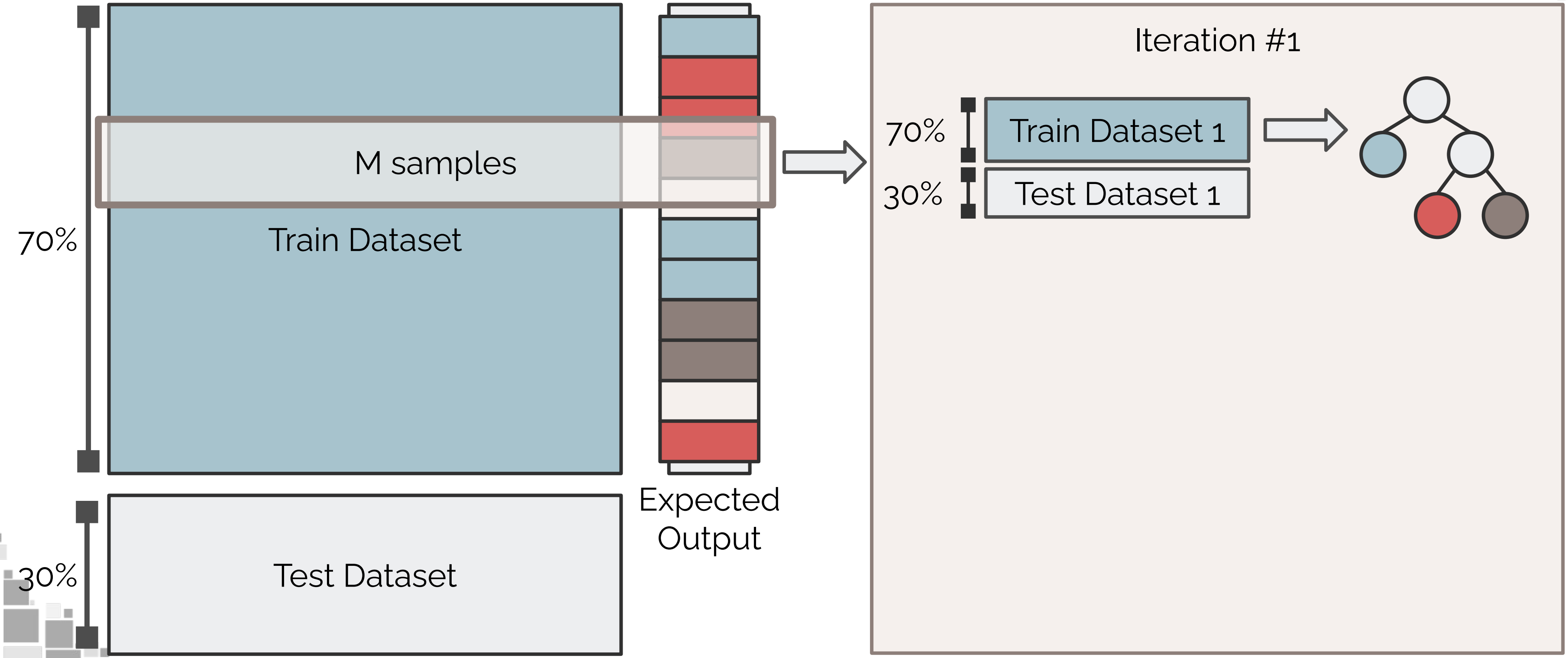
#1
Model
Agnostic

trustee



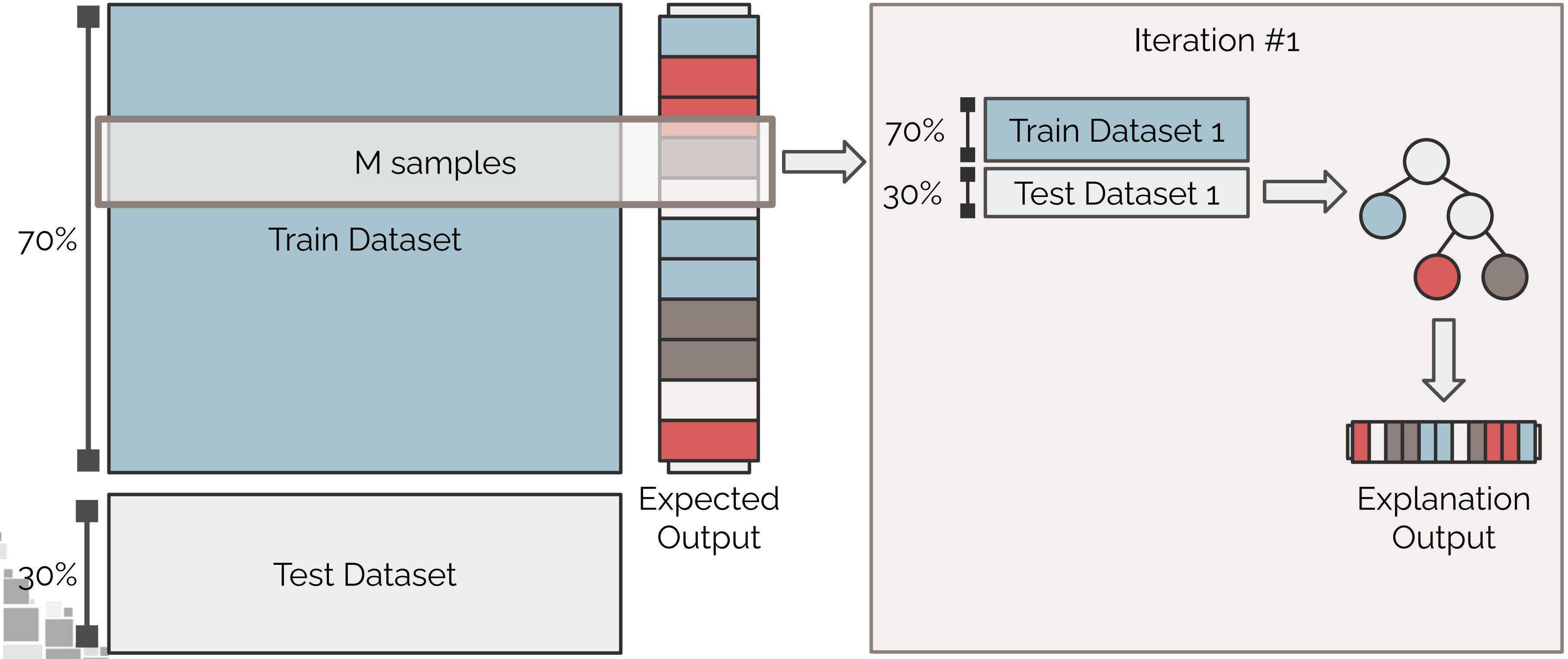

M samples
70%
Train Dataset
30%
Test Dataset
Expected
Output

trustee

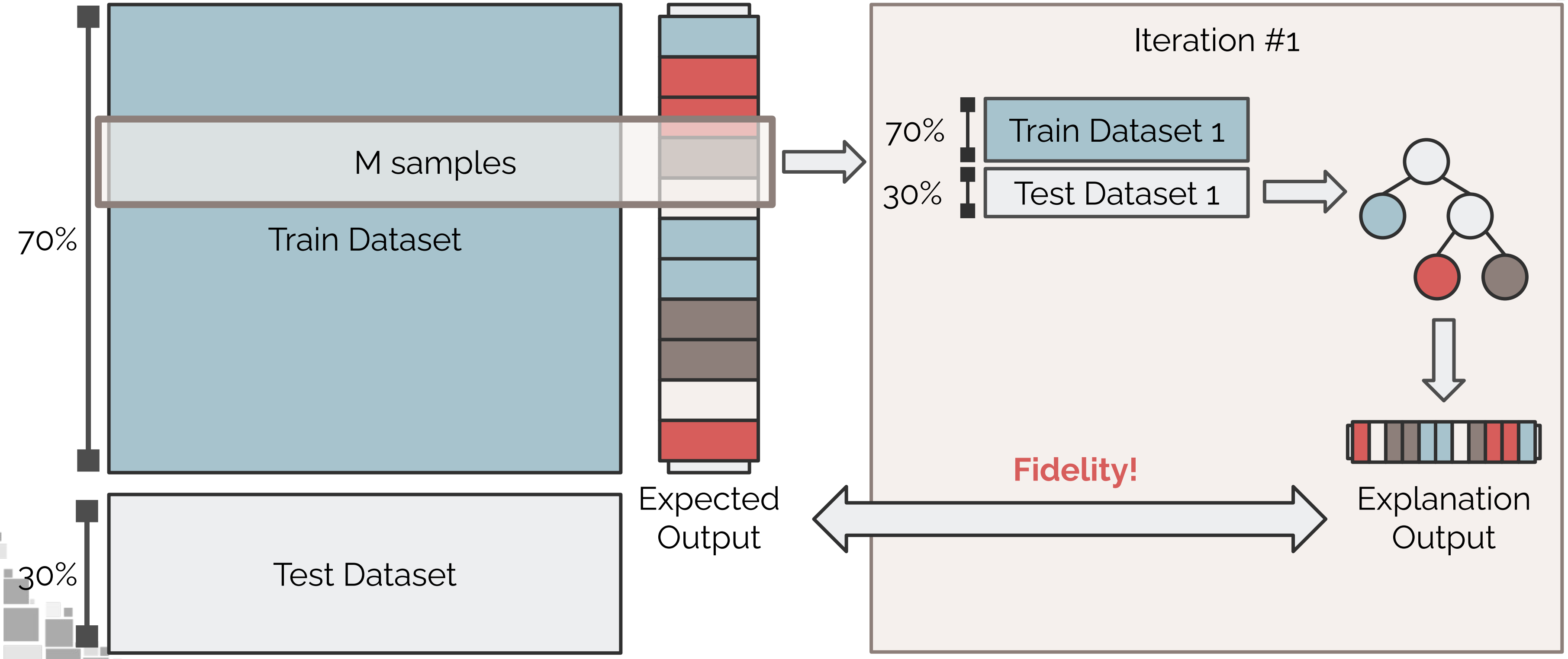


70%
30%
M samples
Train Dataset
Test Dataset
Expected
Output
Iteration #1
70%
30%
Train Dataset 1
Test Dataset 1

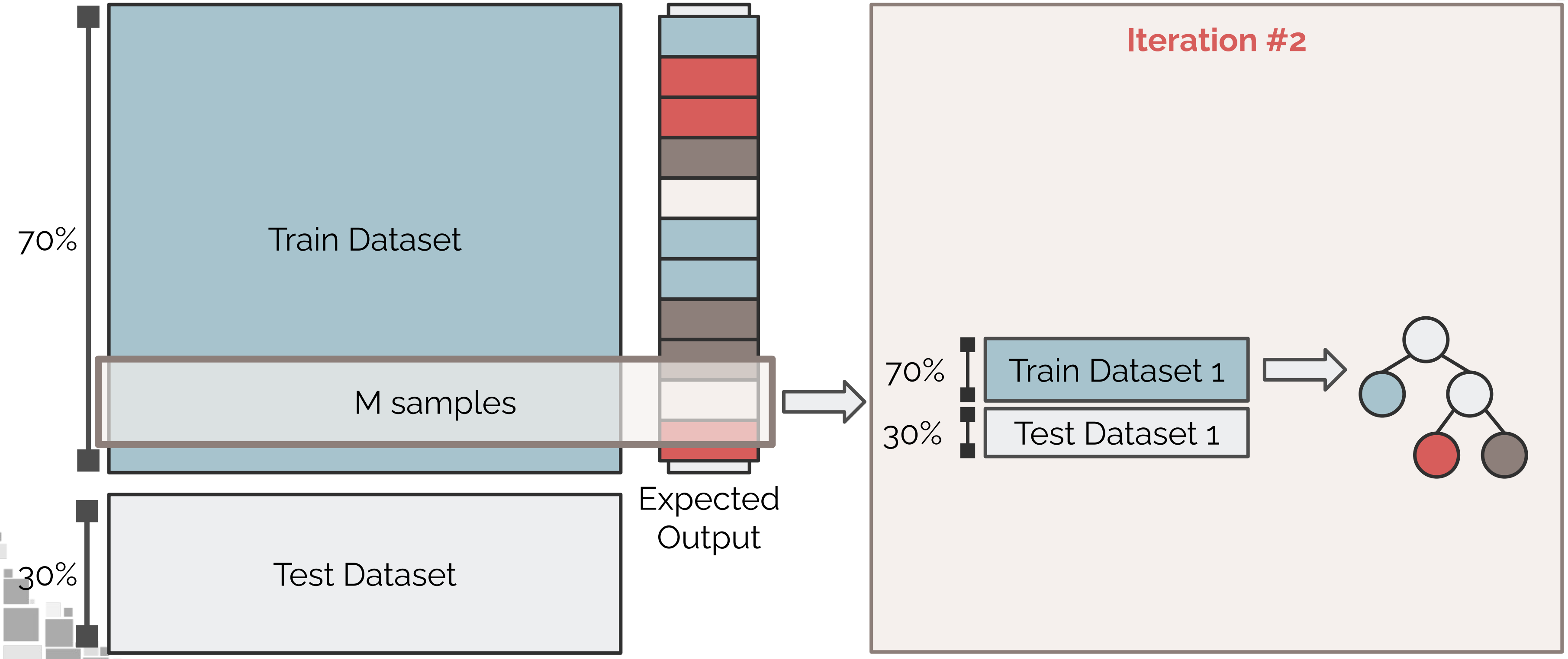
trustee


M samples
70%
Train Dataset
30%
Test Dataset
Expected
Output
Iteration #1
70%
Train Dataset 1
30%
Test Dataset 1
Explanation
Output

trustee

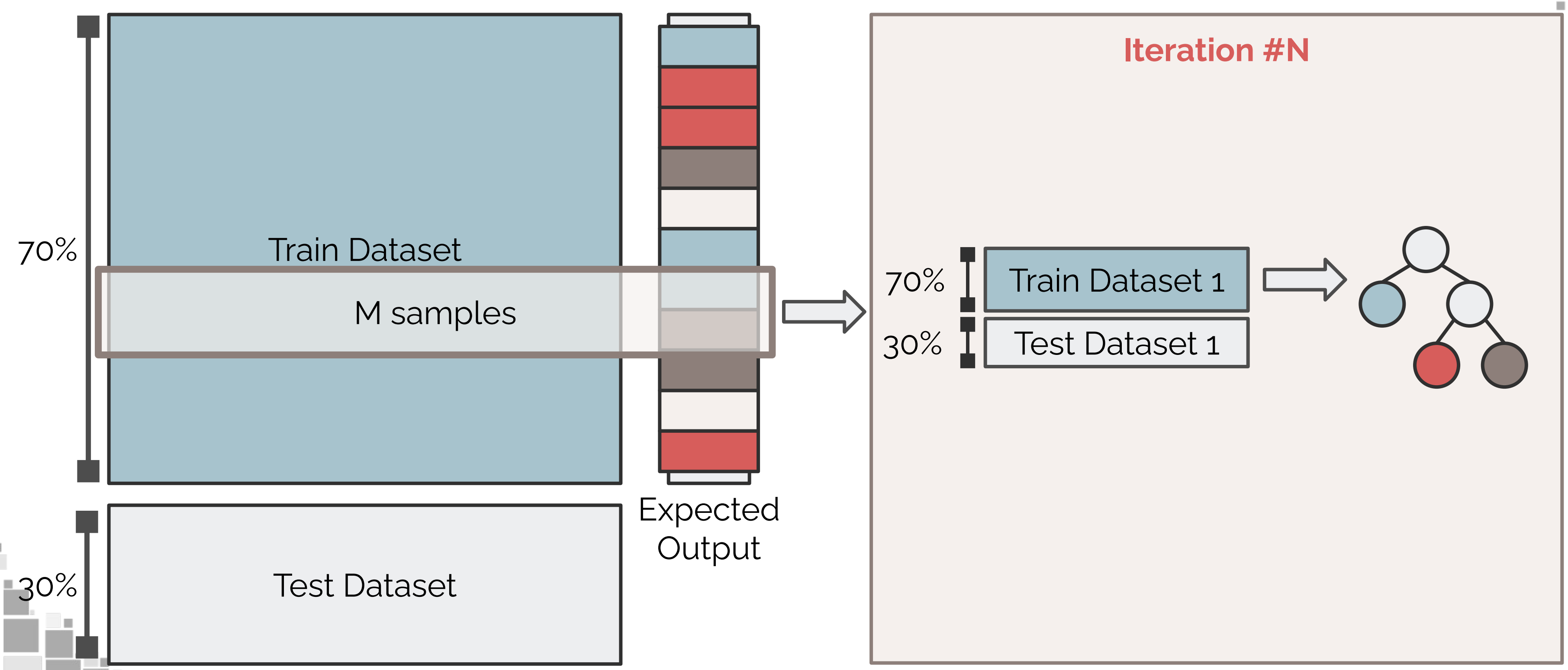


70%
M samples
Train Dataset
30%
Test Dataset
Expected
Output
Iteration #1
70%
Train Dataset 1
30%
Test Dataset 1
Fidelity!
Explanation
Output

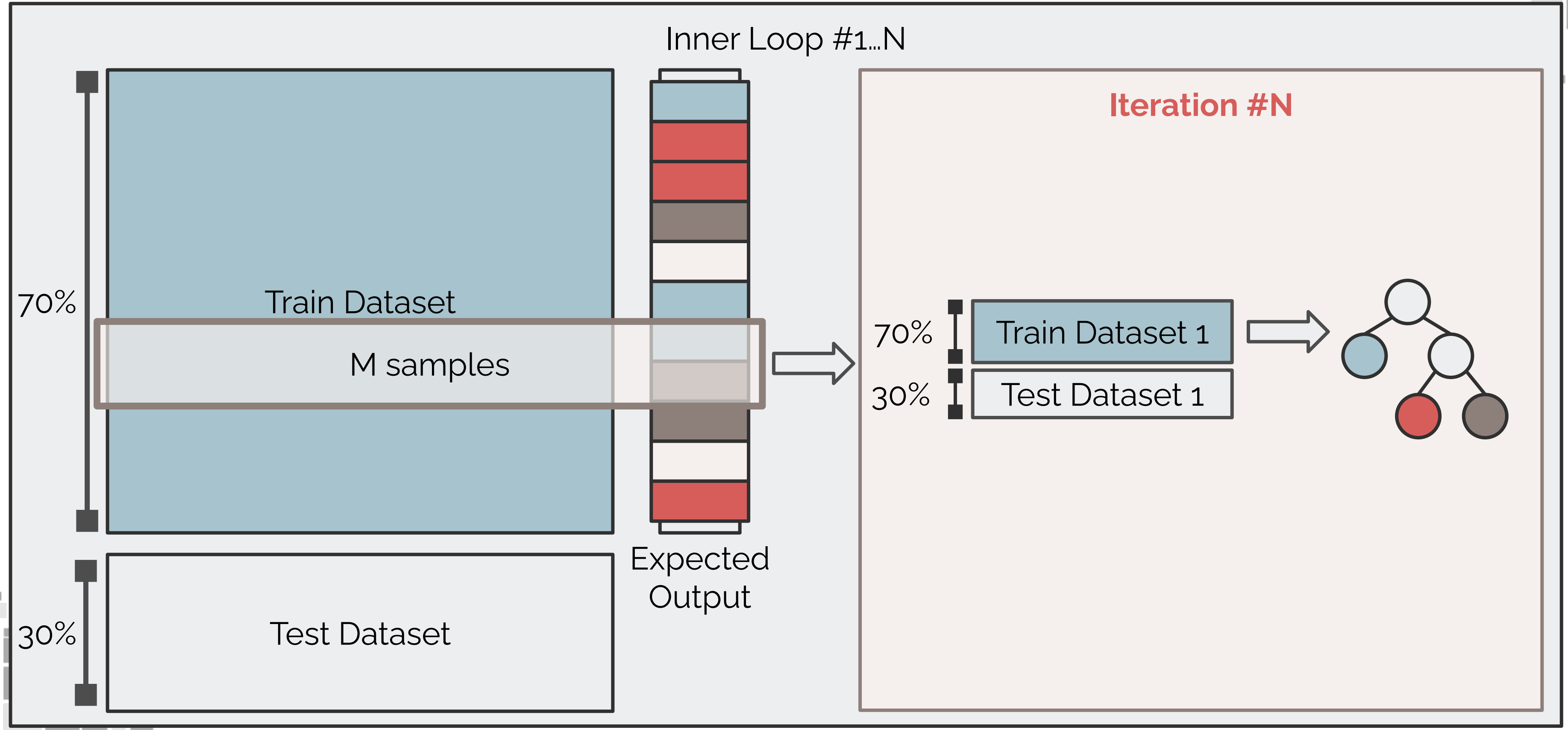


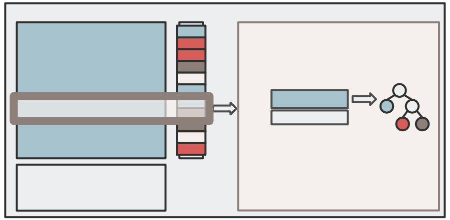
70%
Train Dataset
M samples
30%
Test Dataset
Expected
Output
Iteration #2
70%
Train Dataset 1
30%
Test Dataset 1

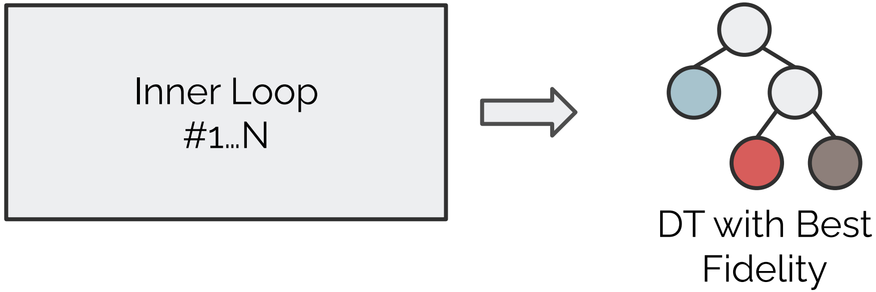
trustee


70%
Train Dataset
M samples
30%
Test Dataset
Expected
Output
Iteration #N
70%
Train Dataset 1
30%
Test Dataset 1

trustee

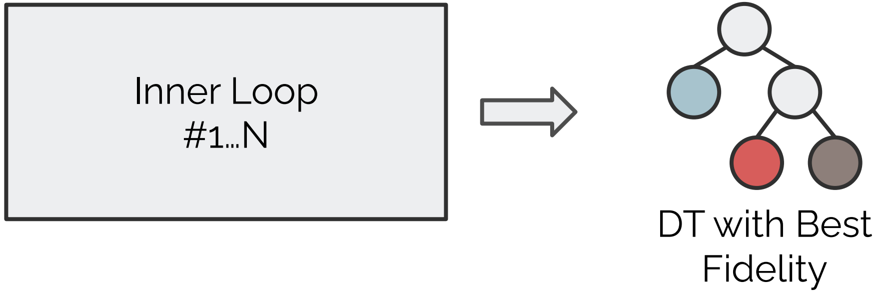

Inner Loop #1...N
70%
Train Dataset
M samples
30%
Test Dataset
Expected Output
Iteration #N
70%
Train Dataset 1
30%
Test Dataset 1



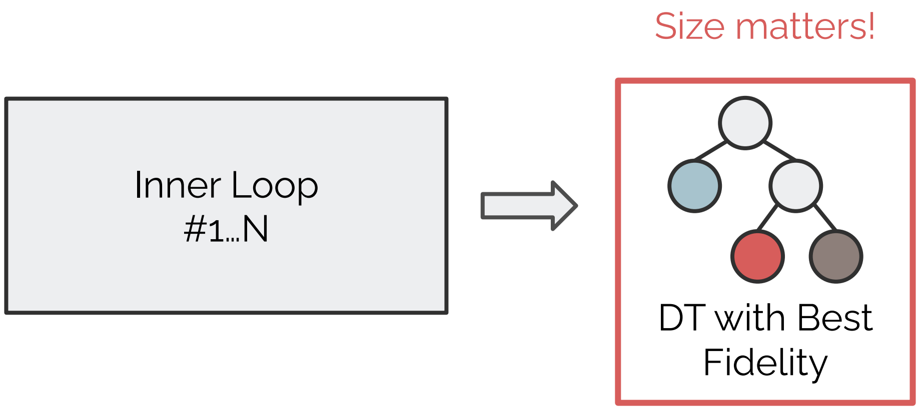

Inner Loop
#1...N
DT with Best
Fidelity

#2
High
Fidelity

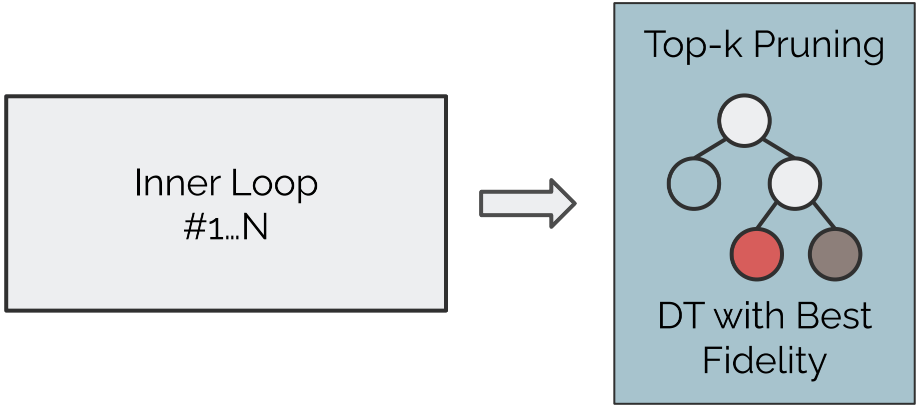
trustee



trustee

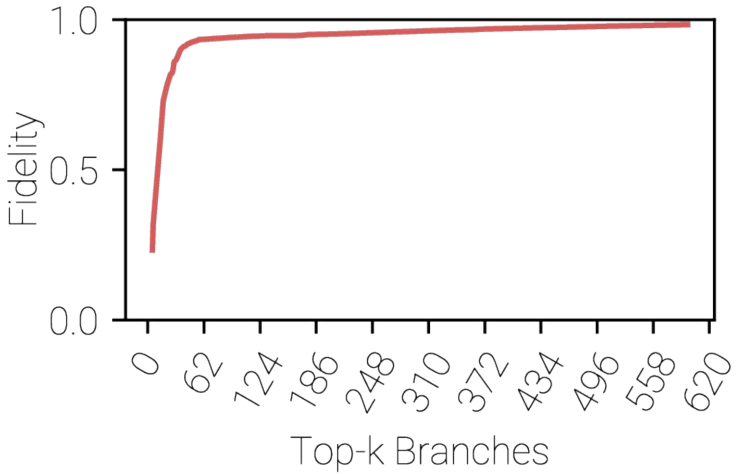


trustee

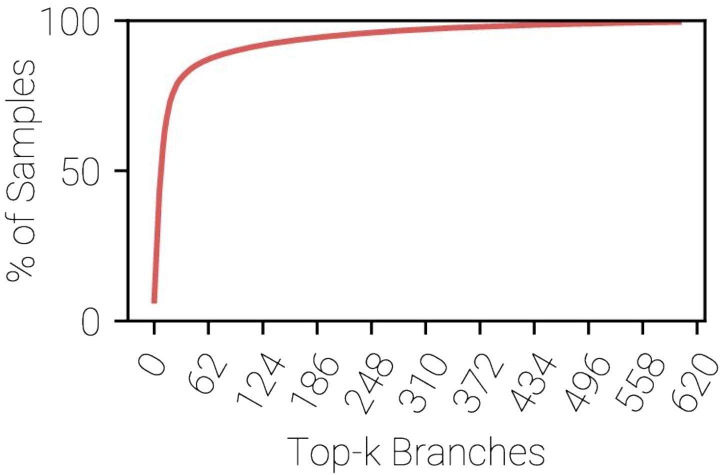


Top-k Pruning

Fidelity

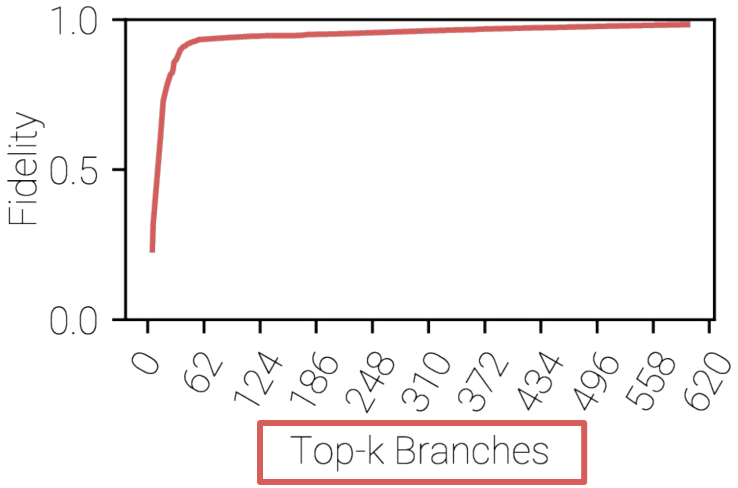


Samples

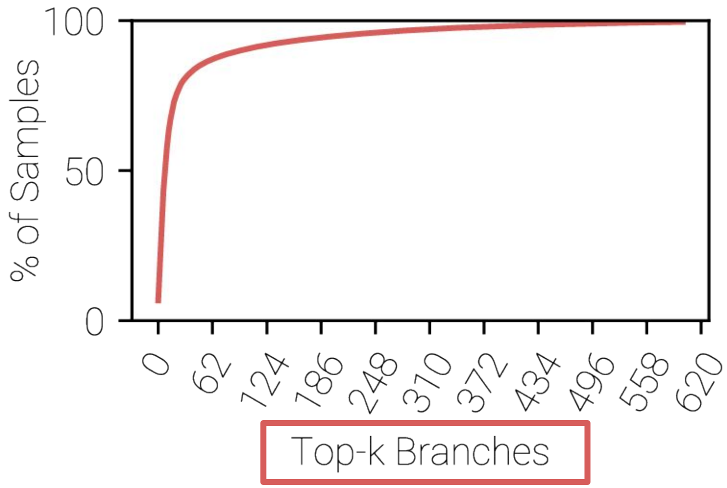


Top-k Pruning

Fidelity



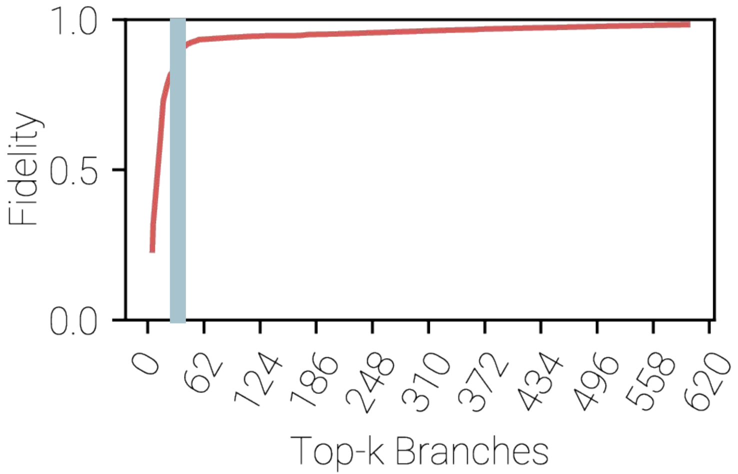
Samples



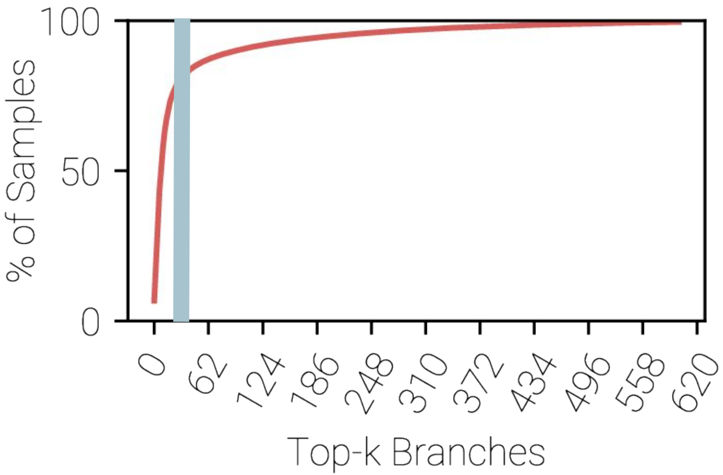
Diminishing returns!

Top-k Pruning

Fidelity



Samples

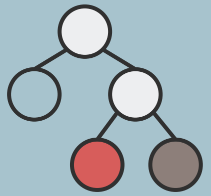


#3
Low
Complexity

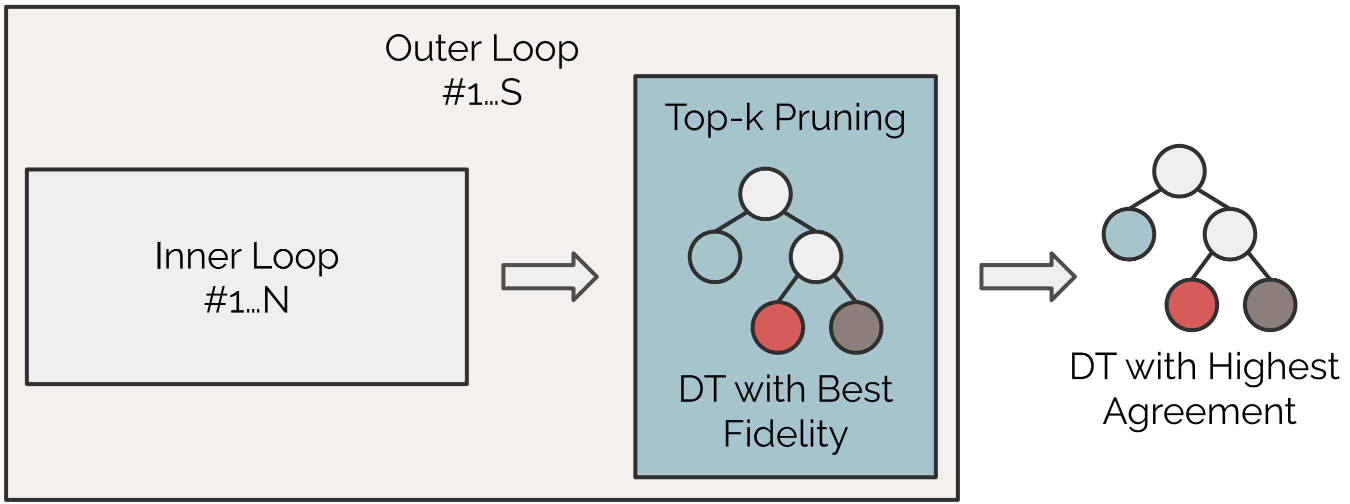
trustee

Inner Loop
#1…N

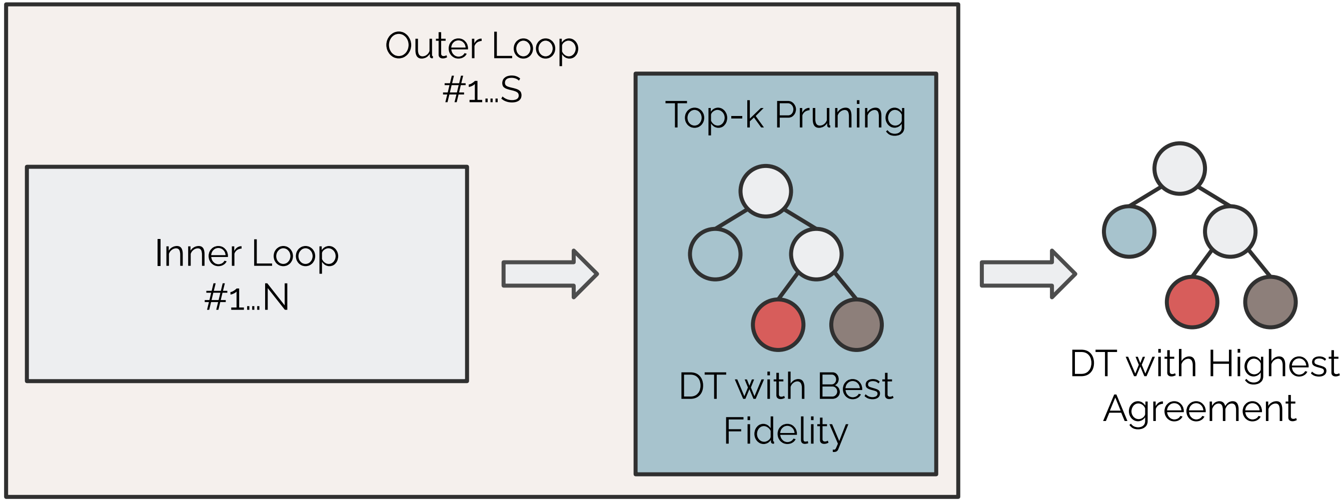
Top-k Pruning



DT with Best
Fidelity

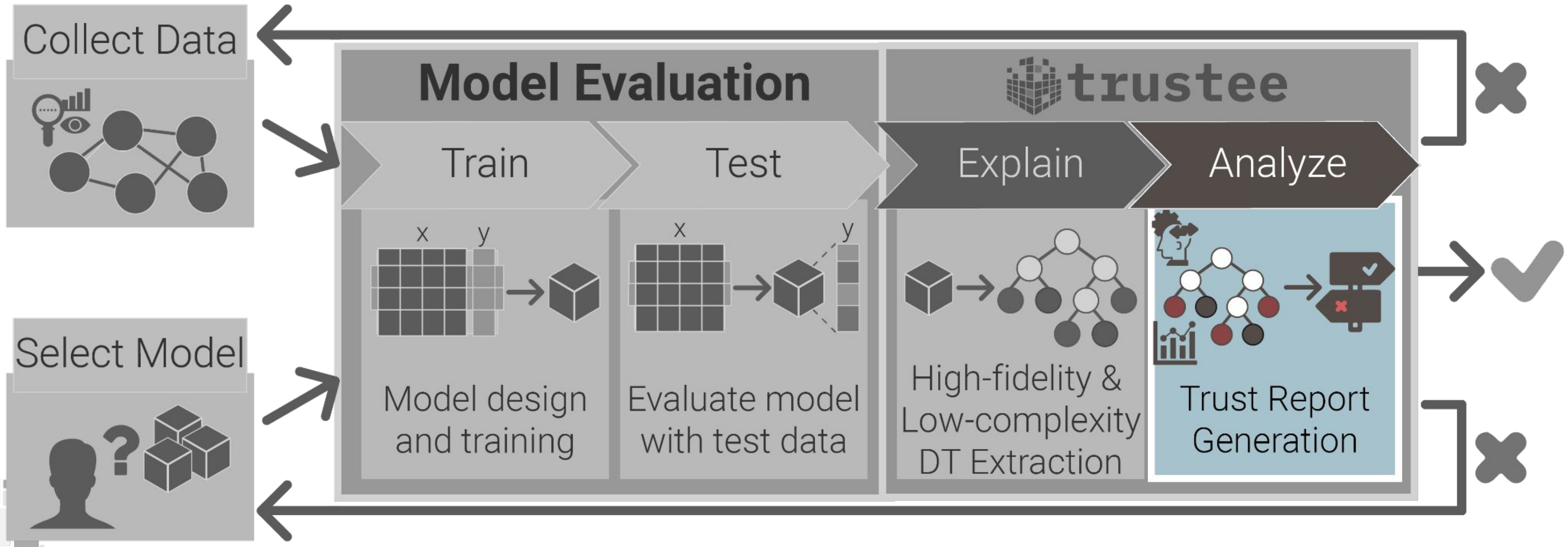

Outer Loop
#1…S
Inner Loop
#1…N
Top-k Pruning
DT with Best
Fidelity
DT with Highest
Agreement

#4
Stable


Outer Loop
#1…S
Inner Loop
#1…N
Top-k Pruning
DT with Best Fidelity
DT with Highest Agreement


Dataset
trustee
DT with Highest Agreement
Black-box Model

Augmented
AI/ML Development Pipeline



Generating Trust Reports

Underspecification issues!

(revisited)

Shortcut Learning	O.O.D. Samples	Spurious Correlations
Model takes shortcuts to classify data!	Model does not generalize!	Model makes the picks up wrong correlations in the data!

Generating Trust Reports

Classification Trust Report

Summary

Blackbox	Whitebox	Top-k Whitebox
Model: RandomForestClassifier	Explanation method: Trustee	Explanation method: Trustee
Dataset size: 947072	Model: DecisionTreeClassifier	Model: DecisionTreeClassifier
Train/Test Split: 70.00% / 30.00%	Iterations: 1	Iterations: 1
	Sample size: 50.00%	Sample size: 50.00%
	Decision Tree Info	Decision Tree Info
	Size: 2437	Size: 9
	Depth: 31	Depth: 4
	Leaves: 1219	Leaves: 5
		Top-k: 1
# Input features: 61	# Input features: 18 (29.51%)	# Input features: -
# Output classes: 5	# Output classes: 5 (100.00%)	# Output classes: 5 (100.00%)

Blackbox: Performance

	precision	recall	f1-score	support
0	1.000	0.912	0.954	24408
1	0.752	0.910	0.824	1872
2	0.929	0.827	0.875	10994
3	0.997	0.929	0.962	65188
4	0.958	0.997	0.978	181660
accuracy			0.967	284122
macro avg	0.927	0.915	0.918	284122
weighted avg	0.968	0.967	0.967	284122

Whitebox: Fidelity

	precision	recall	f1-score	support
0	1.000	1.000	1.000	22254
1	1.000	1.000	1.000	2265
2	0.969	0.965	0.967	9781
3	0.998	0.998	0.998	60768
4	0.998	0.998	0.998	189054
accuracy			0.997	284122
macro avg	0.993	0.992	0.993	284122
weighted avg	0.997	0.997	0.997	284122

Top-k Whitebox: Fidelity

	precision	recall	f1-score	support
0	0.000	0.000	0.000	22254
1	0.000	0.000	0.000	2265
2	0.000	0.000	0.000	9781
3	0.544	0.957	0.694	60768
4	0.875	0.821	0.847	189054
accuracy			0.751	284122
macro avg	0.284	0.356	0.308	284122
weighted avg	0.699	0.751	0.712	284122

Generating Trust Reports

Classification Trust Report

Summary

Blackbox		Whitebox		Top-k Whitebox	
Model:	RandomForestClassifier	Explanation method:	Trustee	Explanation method:	Trustee
Dataset size:	947072	Model:	DecisionTreeClassifier	Model:	DecisionTreeClassifier
Train/Test Split:	70.00% / 30.00%	Iterations:	1	Iterations:	1
		Sample size:	50.00%	Sample size:	50.00%
		Decision Tree Info		Decision Tree Info	
		Size:	2437	Size:	9
		Depth:	31	Depth:	4
		Leaves:	1219	Leaves:	5
# Input features:	61	# Input features:	18 (29.51%)	Top-k:	1
# Output classes:	5	# Output classes:	5 (100.00%)	# Input features:	-
				# Output classes:	5 (100.00%)

Performance (Blackbox)

	precision	recall	f1-score	support
0	1.000	0.912	0.954	24408
1	0.752	0.910	0.824	1872
2	0.929	0.827	0.875	10994
3	0.997	0.929	0.962	65188
4	0.958	0.997	0.978	181660
accuracy			0.967	284122
macro avg	0.927	0.915	0.918	284122
weighted avg	0.968	0.967	0.967	284122

Fidelity (Whitebox)

	precision	recall	f1-score	support
0	1.000	1.000	1.000	22254
1	1.000	1.000	1.000	2265
2	0.969	0.965	0.967	9781
3	0.998	0.998	0.998	60768
4	[illegible]	[illegible]	[illegible]	[illegible]
accuracy			0.997	284122
macro avg	0.993	0.992	0.993	284122
weighted avg	0.997	0.997	0.997	284122

Fidelity (Top-k Whitebox)

	precision	recall	f1-score	support
0	0.000	0.000	0.000	22254
1	0.000	0.000	0.000	2265
2	0.000	0.000	0.000	9781
3	0.544	0.957	0.694	60768
4	0.875	0.821	0.847	189054
accuracy			0.751	284122
macro avg	0.284	0.356	0.308	284122
weighted avg	0.699	0.751	0.712	284122

Generating Trust Reports

Class

Blackbox	
Model:	RandomForestClassifier
Dataset size:	947072
Train/Test Split:	70.00% / 30.00%
# Input features:	61
# Output classes:	5

Explanation m
Model:
Iteration
Sample si

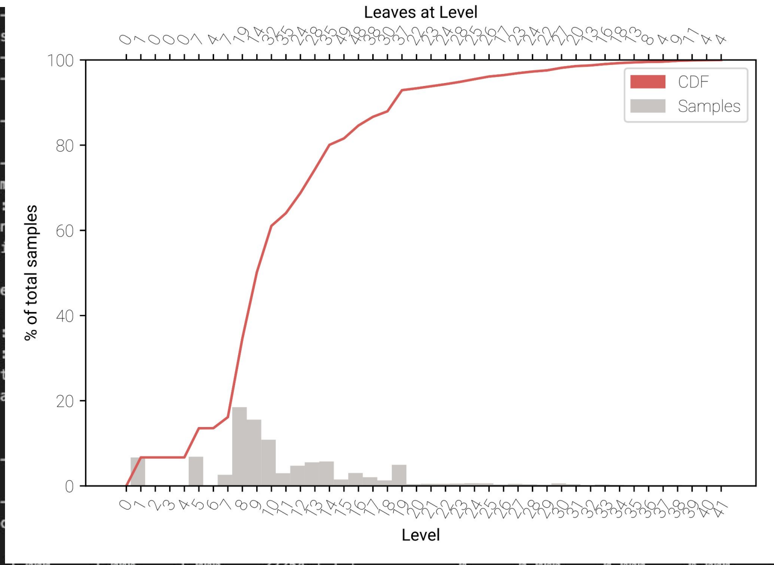
Decision Tre
Size:
Depth:
Leaves:
Input feat
Output cla

Performance

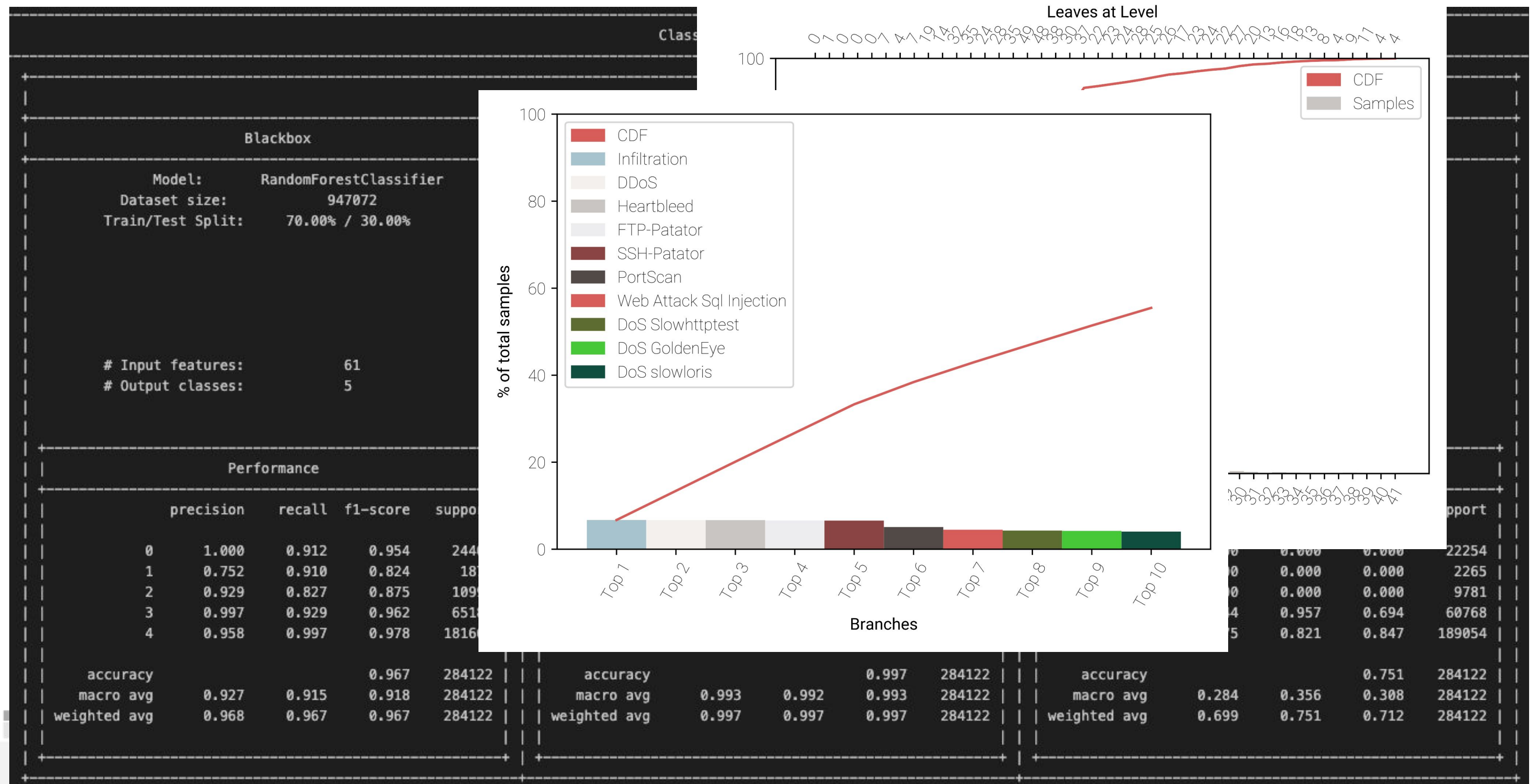
	precision	recall	f1-score	support
0	1.000	0.912	0.954	24408
1	0.752	0.910	0.824	1872
2	0.929	0.827	0.875	10994
3	0.997	0.929	0.962	65188
4	0.958	0.997	0.978	181660
accuracy			0.967	284122
macro avg	0.927	0.915	0.918	284122
weighted avg	0.968	0.967	0.967	284122

	prec			
0	1.000	1.000	1.000	22254
1	1.000	1.000	1.000	2265
2	0.969	0.965	0.967	9781
3	0.998	0.998	0.998	60768
4	0.998	0.998	0.998	189054
accuracy			0.997	284122
macro avg	0.993	0.992	0.993	284122
weighted avg	0.997	0.997	0.997	284122

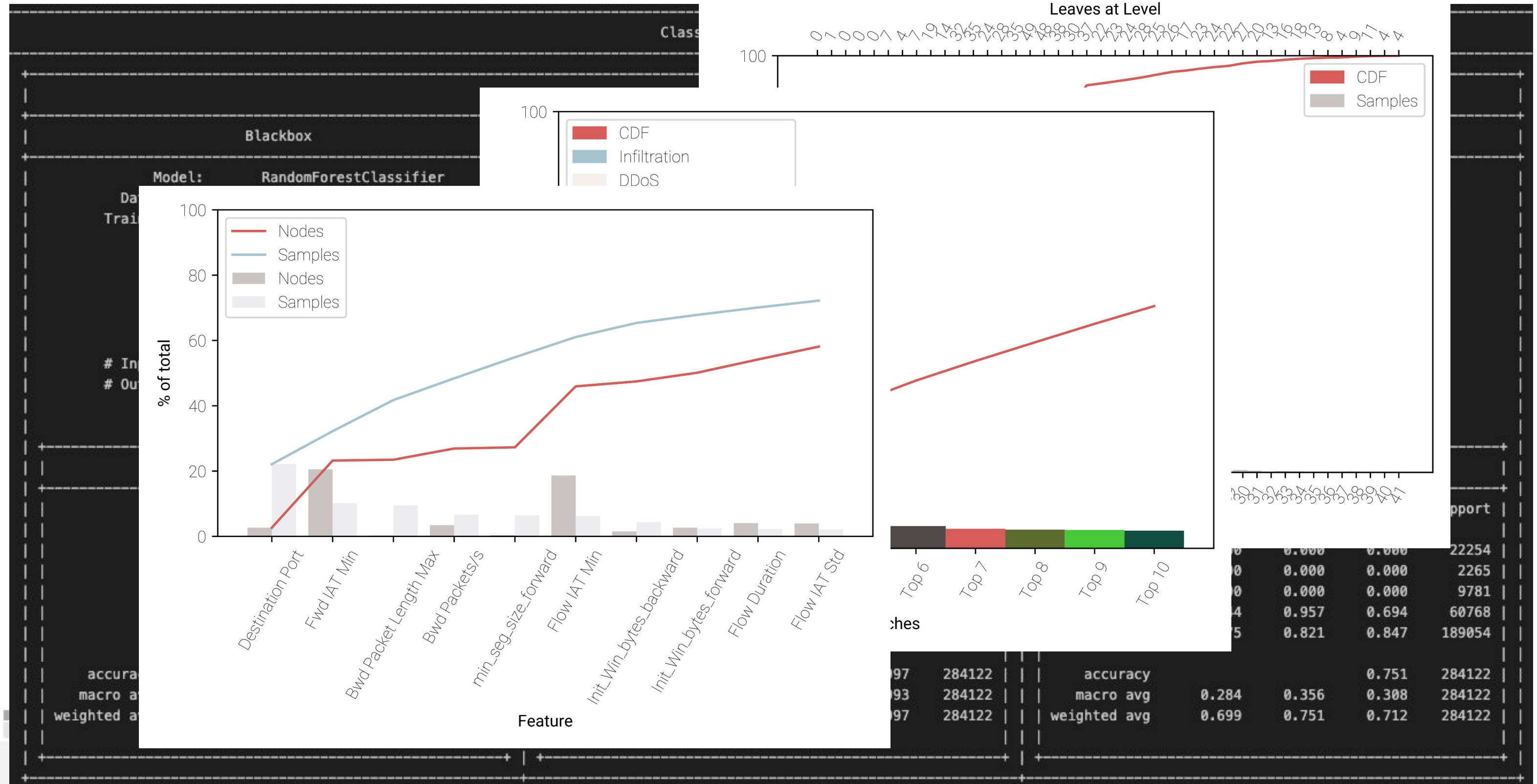
				pport
0	0.000	0.000	0.000	22254
1	0.000	0.000	0.000	2265
2	0.000	0.000	0.000	9781
3	0.544	0.957	0.694	60768
4	0.875	0.821	0.847	189054
accuracy			0.751	284122
macro avg	0.284	0.356	0.308	284122
weighted avg	0.699	0.751	0.712	284122



Generating Trust Reports



Generating Trust Reports



Use Cases

Use Case #1: Detecting VPN vs Non-VPN Traffic

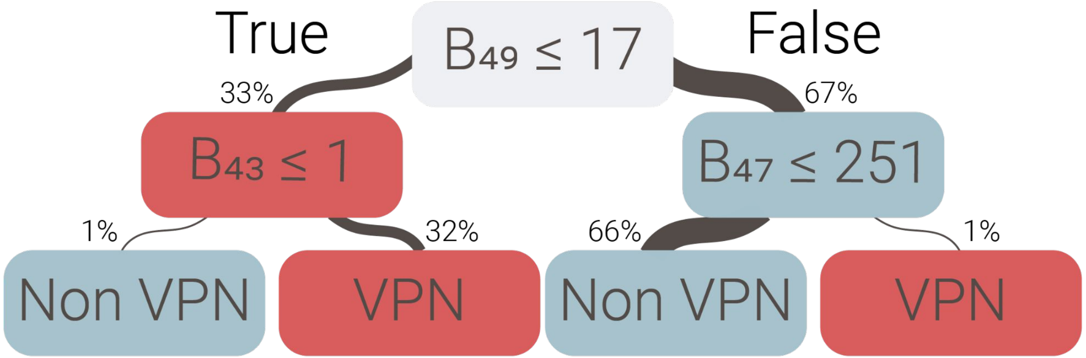
Problem Setup

- Selected publication:
  - *"End-to-end encrypted traffic classification with one-dimensional convolution neural networks"* — Wang et al., 2017
- Proposal:
  - **Model**: 1D-CNN to classify traffic between encrypted VPN traffic and non-encrypted traffic (i.e. VPN vs Non-VPN)
  - **Features**: first 784 raw bytes of each PCAP file
  - **Dataset**: ISCX VPN-nonVPN 2016 [https://www.unb.ca/cic/datasets/vpn.html]
- Results:
  - Reported F1-score: 0.99
  - Reproduced F1-score: 0.959

Use Case #1: Detecting VPN vs Non-VPN Traffic

Explanation

Fidelity: 1.000
No pruning
7 nodes



Use Case #1: Detecting VPN vs Non-VPN Traffic

Explanation

Non VPN

Destination MAC Address | Source MAC Address

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Pcap Meta (0)	161	178	195	212	0	2	0	4	0	0	0	0	0	0	0	0	0	0	255	255
(20)	0	0	0	1	85	65	10	69	0	5	80	24	0	0	0	64	0	0	0	64
Eth (40)	1	0	94	0	0	252	184	172	111	54	28	162	8	0	69	0	0	50	65	228
IPv4 (60)	0	0	1	17	34	185	131	202	240	87	224	0	0	252	201	86	20	235	0	...

VPN

Total Length | Frag.Off. | Protocol

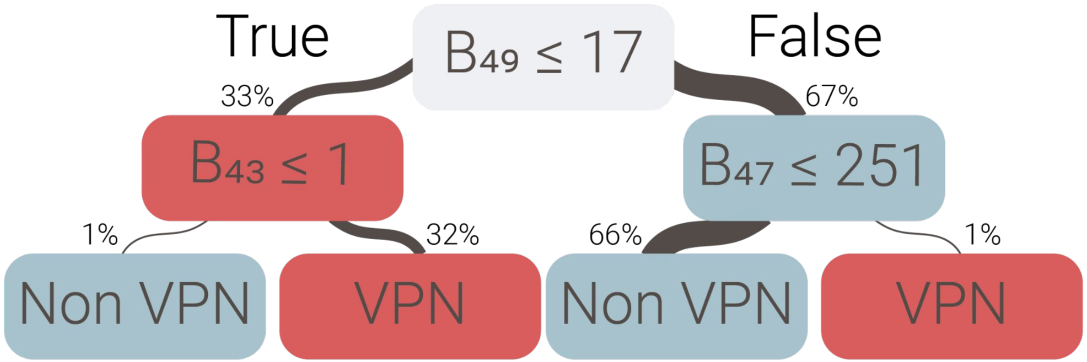
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Pcap Meta (0)	161	178	195	212	0	2	0	4	0	0	0	0	0	0	0	0	0	0	255	255
(20)	0	0	0	101	85	45	101	91	0	0	111	11	0	0	0	56	0	0	0	56
IPv4 (40)	69	0	0	56	99	213	[illegible]	0	[illegible]	17	[illegible]	254	10	8	0	10	69	171	255	36
UDP (60)	146	214	13	150	0	36	120	43	0	1	0	8	33	18	164	66	52	167	9	...

Use Case #1: Detecting VPN vs Non-VPN Traffic

Explanation

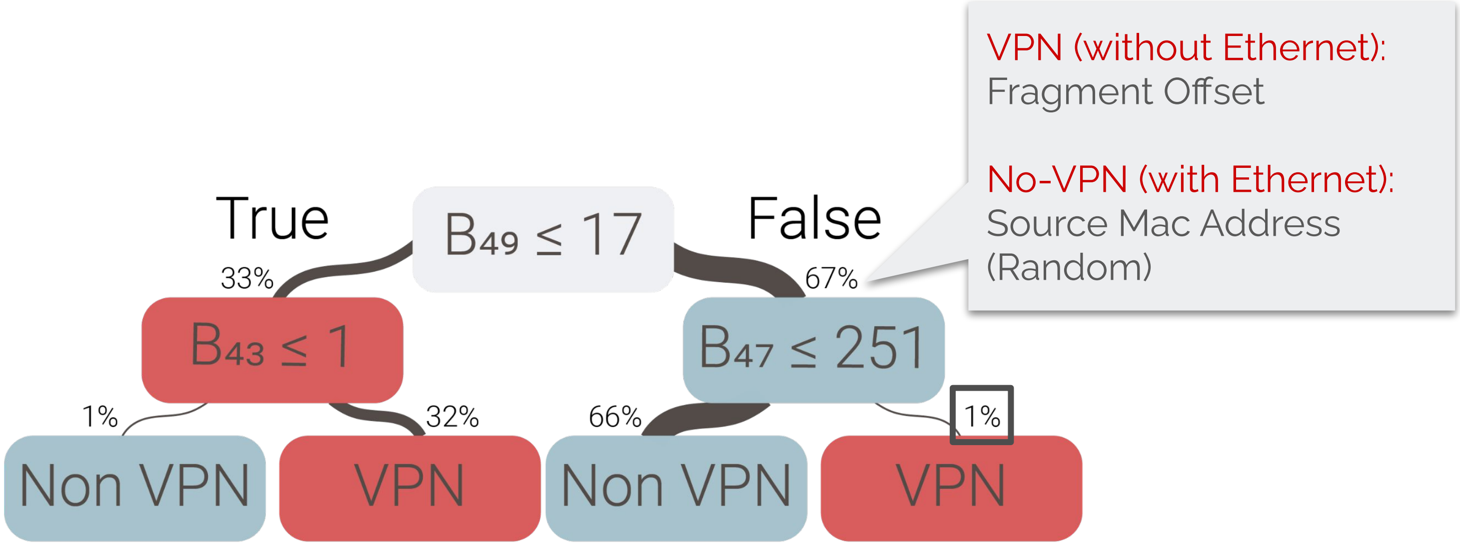
VPN (without Ethernet):
IPv4 Protocol (6 or 17)

No-VPN (with Ethernet):
Source Mac Address
(Random)



Use Case #1: Detecting VPN vs Non-VPN Traffic

Explanation

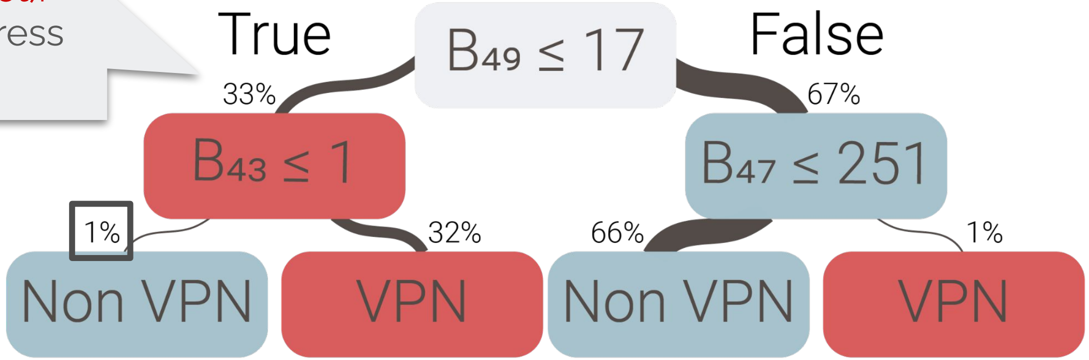


Use Case #1: Detecting VPN vs Non-VPN Traffic

Explanation

VPN (without Ethernet):
IP Total Length

No-VPN (with Ethernet):
Destination Mac Address
(Always 0)



Use Case #1: Detecting VPN vs Non-VPN Traffic

Validation

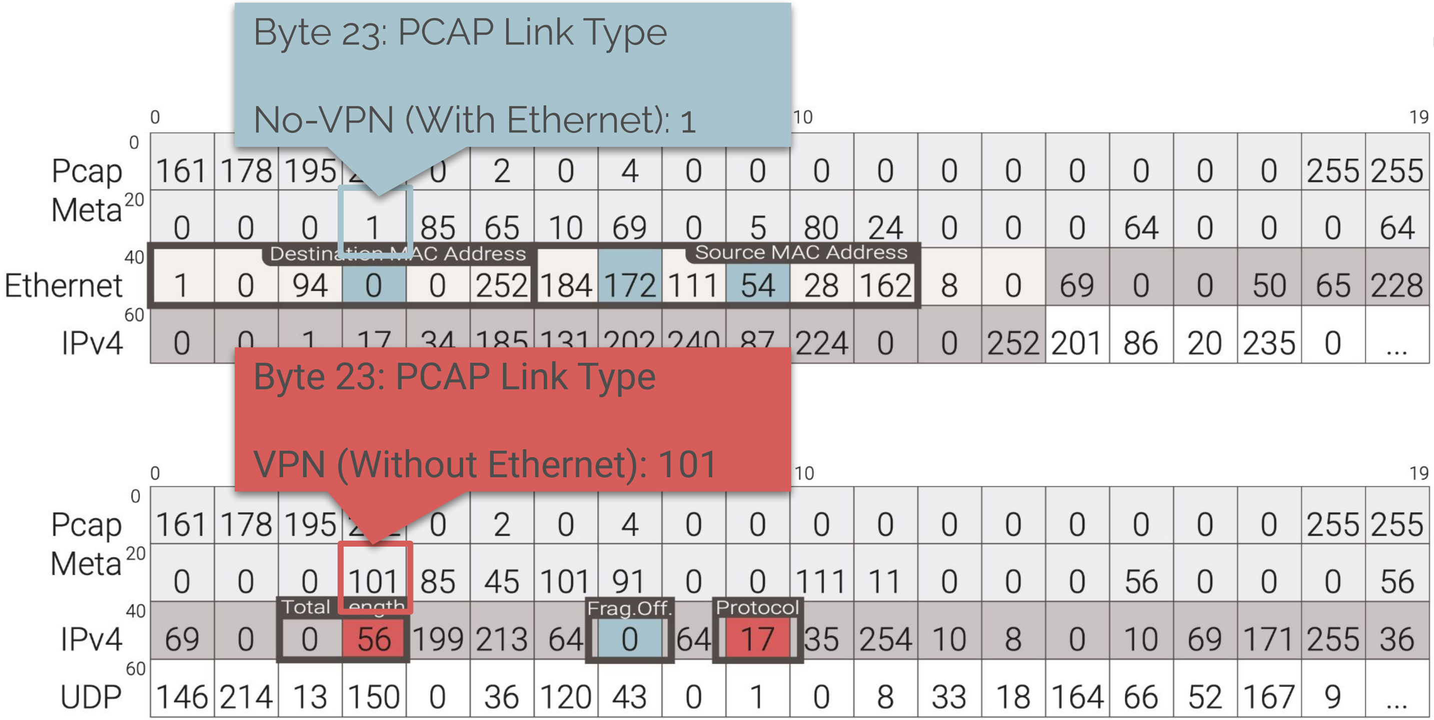
- Validation dataset:
  - Tampering with packet headers from original PCAPs

Validation Dataset	Avg. Precision	Avg. Recall	Avg. F1
Untampered	0.959	0.956	0.955
Tampered-43-47-49	0.959	0.956	0.955

Use Case #1: Detecting VPN vs Non-VPN Traffic

No VPN

VPN



Use Case #1: Detecting VPN vs Non-VPN Traffic

Validation

- Validation dataset:
  - Tampering with packet headers from original PCAPs

Validation Dataset	Avg. Precision	Avg. Recall	Avg. F1
Untampered	0.959	0.956	0.955
Tampered-43-47-49	0.959	0.956	0.955
Tampered-32-to-63	0.889	0.867	0.856
Tampered-0-to-63	0.831	0.757	0.734
Tampered-0-to-127	0.753	0.555	**0.398**

Use Case #1: Detecting VPN vs Non-VPN Traffic

Validation

- Validation dataset:
    - Tampering with packet headers from original PCAPs

Validation Dataset	Avg. Precision	Avg. Recall	Avg. F1
Untampered	0.959	0.956	0.955
Tampered-43-47-49	0.959	0.956	0.955
Tampered-32-to-63	0.889	0.867	0.856
Tampered-0-to-63	0.831	0.757	0.734
Tampered-0-to-127	0.753	0.555	**0.398**

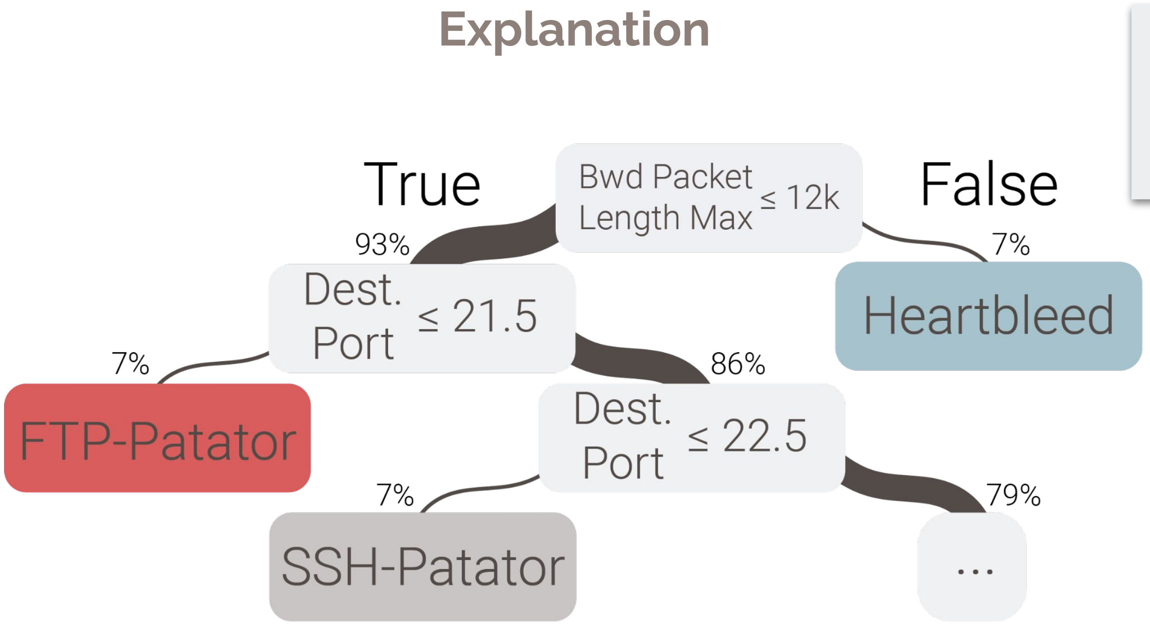
Takeaway: the model suffers from shortcut learning!

Use Case #2: Detecting Heartbleed Traffic

Problem Setup

- Selected publications:
    - *Many papers that rely on the CIC-IDS-2017 dataset*
    - *"Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization" — Sharafaldin et al., 2018*
- Proposal:
    - **Model**: Random Forest to classify traffic between benign traffic and 13 different attacks (e.g. PortScan, DDoS, Heartbleed)
    - **Features**: 78 pre-computed features, from flow statistics (e.g. flow duration, mean IAT)
    - **Dataset**: CIC-IDS-2017 [https://www.unb.ca/cic/datasets/ids-2017.html]
- Results:
    - Reported F1-score: 0.99
    - Reproduced F1-score: 0.99

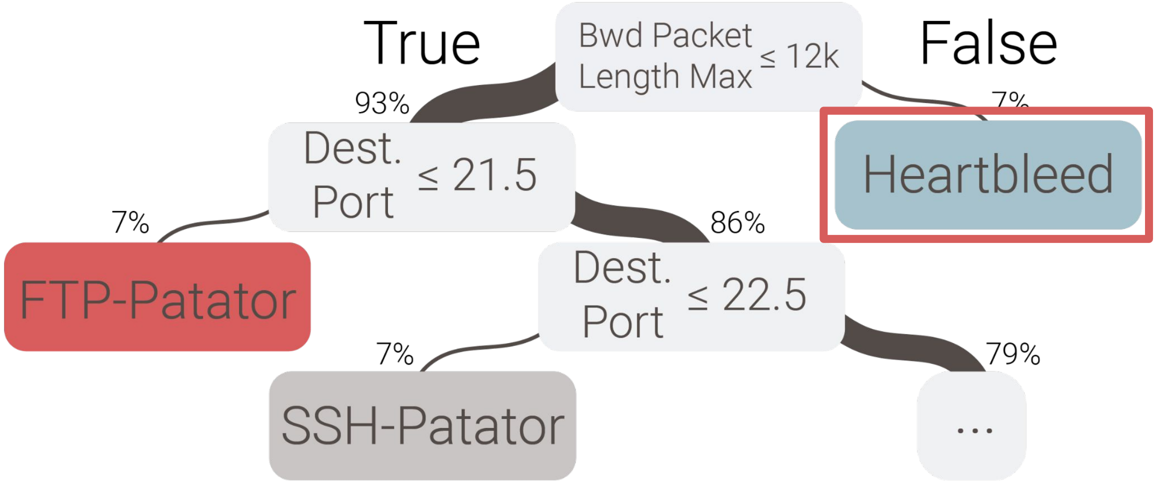
Use Case #2: Detecting Heartbleed Traffic



Fidelity: 0.99
Top-3 pruning
6 nodes

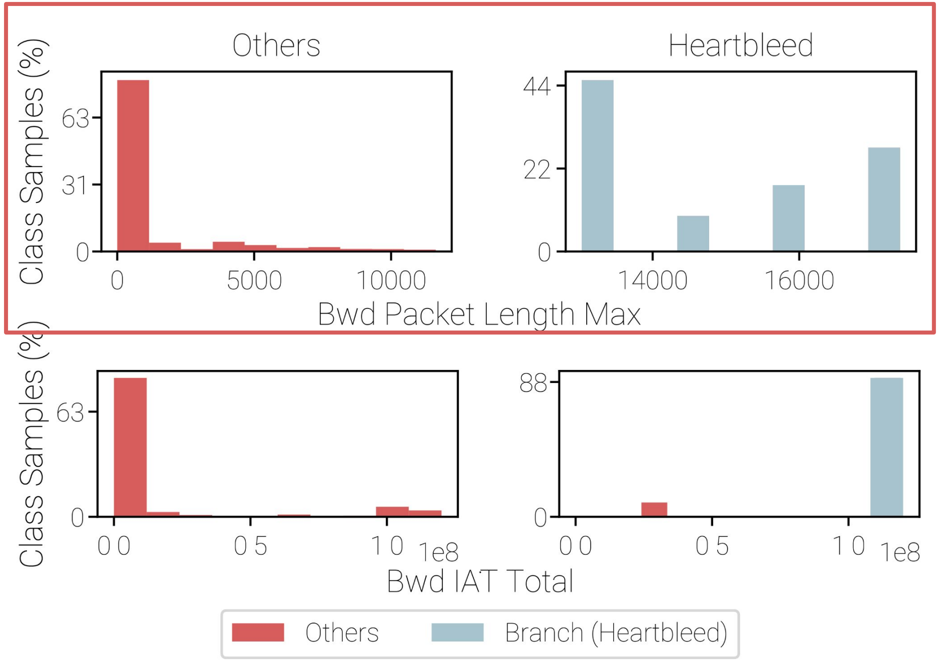
Use Case #2: Detecting Heartbleed Traffic

Explanation



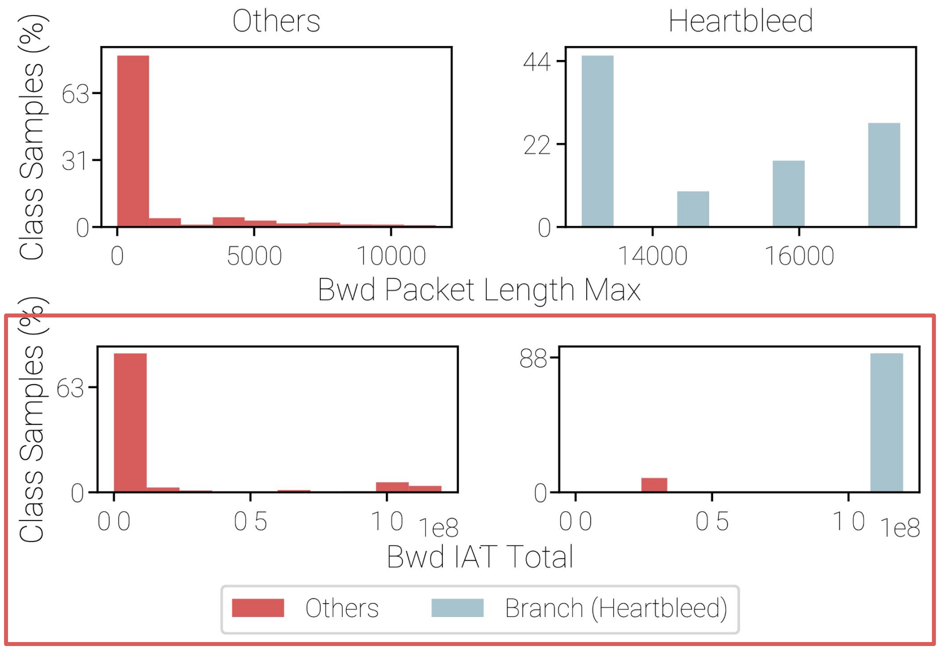
Use Case #2: Detecting Heartbleed Traffic

Explanation



Use Case #2: Detecting Heartbleed Traffic

Explanation



Use Case #2: Detecting Heartbleed Traffic

- Heartbleed attack:
  - An attacker sends an HTTPS heartbeat message with a value in the size field bigger than the message
    - e.g., 16k bytes packet with 64k bytes size value
  - A vulnerable server responds with a message with the size equal to the value specified in the size field and reveals information stored locally in its memory
    - e.g. server returns 64k bytes (16k from packet and 48k from memory)

- In the CIC-IDS-2017 dataset:
  - HTTPS connection was never closed during the duration of the attack
    - Huge number of backward bytes and very high IAT in the flow!

Use Case #2: Detecting Heartbleed Traffic

Validation

- Validation dataset:
  - 1000 new heartbleed flows closing connection after every heartbeat
  - Backward bytes and IAT similar to benign traffic

Class	Precision	Recall	F1
Heartbleed (i.i.d.)	1.000	1.000	1.000
Heartbleed (o.o.d)	0.000	0.000	0.000

Use Case #2: Detecting Heartbleed Traffic

Validation

- Validation dataset:
  - 1000 new heartbleed flows closing connection after every heartbeat
  - Backward bytes and IAT similar to benign traffic

Class	Precision	Recall	F1
Heartbleed (i.i.d.)	1.000	1.000	1.000
Heartbleed (o.o.d)	0.000	0.000	0.000

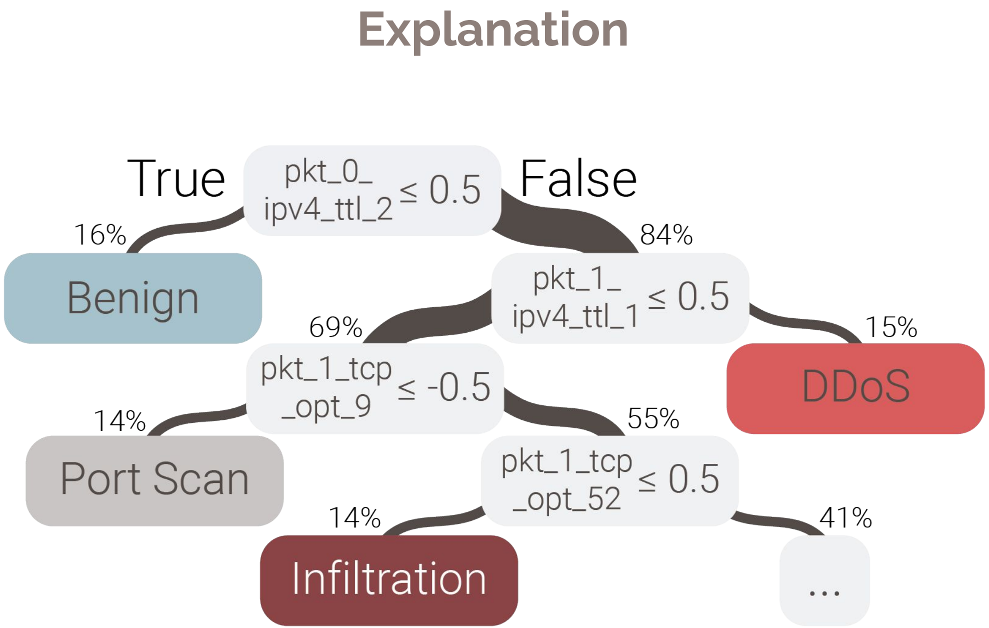
Takeaway: the model is overfitted to training data and fails to identify o.o.d. samples!

Use Case #3: Inferring Malicious Traffic for IDS

Problem Setup

- Selected publications:
  - *"New Directions in Automated Traffic Analysis" — Holland et al., 2020*
- Proposal:
  - **Model**: nPrintML, an AutoML model for an Intrusion Detection System (IDS)
  - **Features**: 4,480 features with values -1, 0, or 1, each feature represents a bit of a set of pre-established protocol headers.
  - **Dataset**: CIC-IDS-2017 [https://www.unb.ca/cic/datasets/ids-2017.html]
- Results:
  - Reported F1-score: 0.99
  - Reproduced F1-score: 0.99

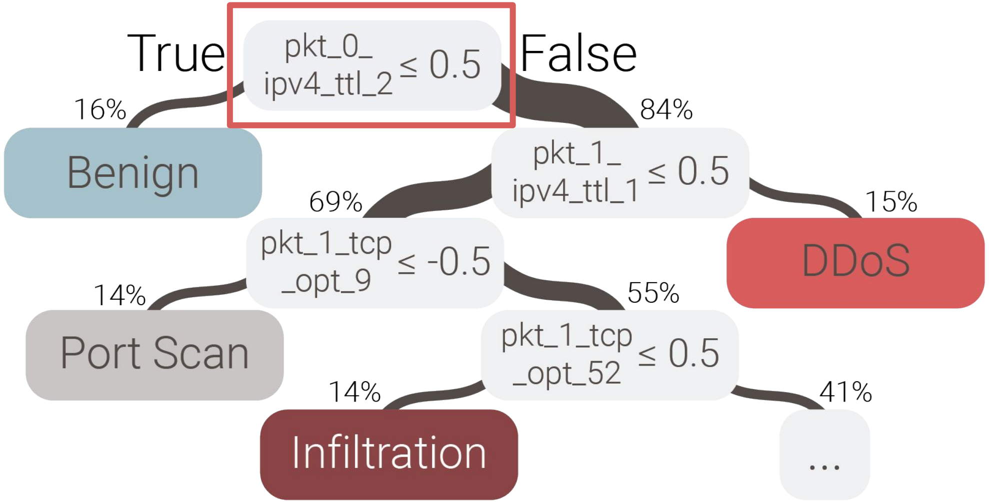
Use Case #3: Inferring Malicious Traffic for IDS



Fidelity: 0.99
Top-4 pruning
8 nodes

Use Case #3: Inferring Malicious Traffic for IDS

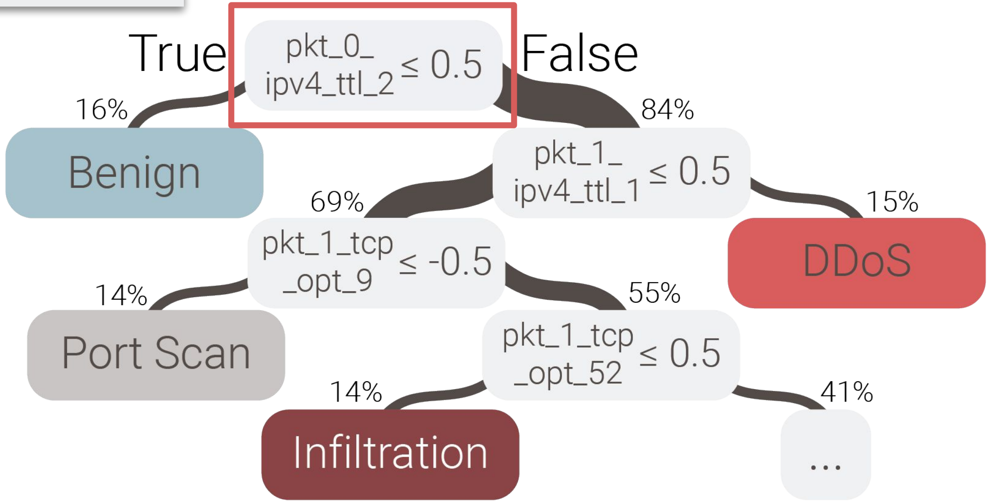
Explanation



Use Case #3: Inferring Malicious Traffic for IDS

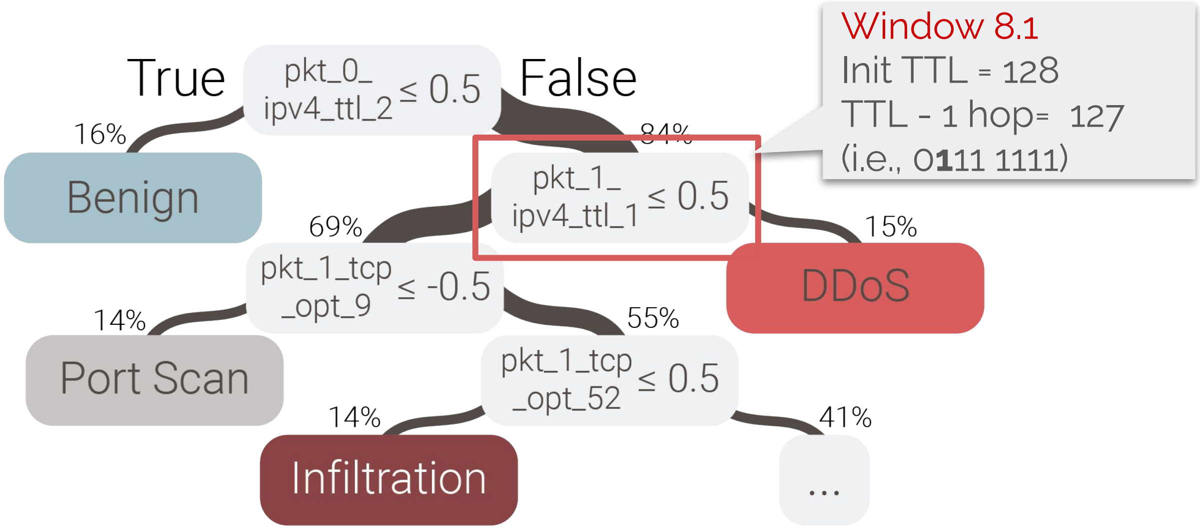
Kali Linux
Init TTL = 64
TTL - 1 hop = 63
(i.e., 00**11** 1111)

Explanation



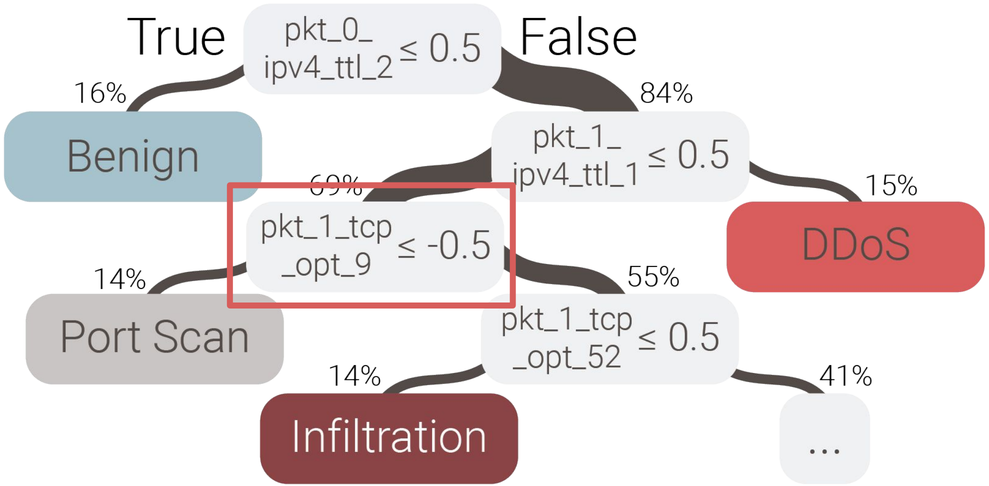
Use Case #3: Inferring Malicious Traffic for IDS

Explanation



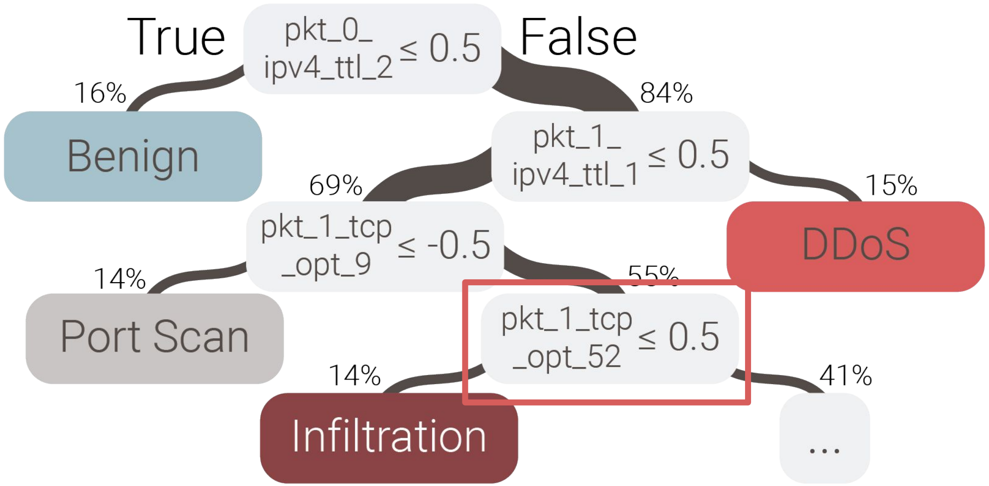
Use Case #3: Inferring Malicious Traffic for IDS

Explanation



Use Case #3: Inferring Malicious Traffic for IDS

Explanation



Use Case #3: Inferring Malicious Traffic for IDS

Validation

- Validation dataset:
  - Curated balanced dataset with 4,047 flows from real-world traffic in UCSB network
  - Used Suricata-IDS to generate flow labels

Class	Precision	Recall	F1
Benign	0.653	0.806	0.722
DoS	0.000	0.000	0.000
Port Scan	0.120	0.143	0.130
Average	0.256	0.315	0.282

Use Case #3: Inferring Malicious Traffic for IDS

Validation

- Validation dataset:
  - Curated balanced dataset with 4,047 flows from real-world traffic in UCSB network
  - Used Suricata-IDS to generate flow labels

Class	Precision	Recall	F1
Benign	0.653	0.806	0.722
DoS	0.000	0.000	0.000
Port Scan	0.120	0.143	0.130
Average	0.256	0.315	0.282

Takeaway: the model suffers from spurious correlations in the training data!

Other Use Cases

Problem	Model(s)	Dataset(s)	Trustee Fidelity	Inductive Bias
Detect VPN traffic (Wang *et al.*, ISI'17)	1-D CNN	ISCX VPN-nonVPN	1.00	Shortcut learning
Detect Heartbleed traffic (Sharafaldin *et al.*, ICISSP'18)	RFC	CIC-IDS-2017	0.99	O.O.D.
Detect Malicious traffic (IDS) (Holland *et al.*, CCS'21)	nPrintML	CIC-IDS-2017	0.99	Spurious Correlation
Anomaly Detection (Mirsky *et al.*, NDSS'18)	Kitsune	Mirai dataset	0.99	O.O.D
OS Fingerprinting (Holland *et al.*, CCS'21)	nPrintML	CIC-IDS-2017	0.99	O.O.D
IoT Device Fingerprinting (Xiong *et al.*, HotNets'19)	Iisy	UNSW-IoT	0.99	Shortcut learning
Adaptive Bit-rate (Mao *et al.*, SIGCOMM'17)	Pensieve	HSDPA Norway	0.99	O.O.D

Other Use Cases

Problem	Model(s)	Dataset(s)	Trustee Fidelity	Inductive Bias
Detect VPN traffic (Wang *et al.*, ISI'17)	1-D CNN	ISCX VPN-nonVPN	1.00	Shortcut learning
Detect Heartbleed traffic (Sharafaldin *et al.*, ICISSP'18)	RFC	CIC-IDS-2017	0.99	O.O.D.
Detect Malicious traffic (IDS) (Holland *et al.*, CCS'21)	nPrintML	CIC-IDS-2017	0.99	Spurious Correlation
Anomaly Detection (Mirsky *et al.*, NDSS'18)	Kitsune	Mirai dataset	0.99	O.O.D
OS Fingerprinting (Holland *et al.*, CCS'21)	nPrintML	CIC-IDS-2017	0.99	O.O.D
IoT Device Fingerprinting (Xiong *et al.*, HotNets'19)	Iisy	UNSW-IoT	0.99	Shortcut learning
Adaptive Bit-rate (Mao *et al.*, SIGCOMM'17)	Pensieve	HSDPA Norway	0.99	O.O.D

Other Use Cases

Problem	Model(s)	Dataset(s)	Trustee Fidelity	Inductive Bias
Detect VPN traffic (Wang *et al.*, ISI'17)	1-D CNN	ISCX VPN-nonVPN	1.00	Shortcut learning
Detect Heartbleed traffic (Sharafaldin *et al.*, ICISSP'18)	RFC	CIC-IDS-2017	0.99	O.O.D.
Detect Malicious traffic (IDS) (Holland *et al.*, CCS'21)	nPrintML	CIC-IDS-2017	0.99	Spurious Correlation
Anomaly Detection (Mirsky *et al.*, NDSS'18)	Kitsune	Mirai dataset	0.99	O.O.D
OS Fingerprinting (Holland *et al.*, CCS'21)	nPrintML	CIC-IDS-2017	0.99	O.O.D
IoT Device Fingerprinting (Xiong *et al.*, HotNets'19)	Iisy	UNSW-IoT	0.99	Shortcut learning
Adaptive Bit-rate (Mao *et al.*, SIGCOMM'17)	Pensieve	HSDPA Norway	0.99	O.O.D

Additional details (see paper)

Algorithmic Description of Trustee

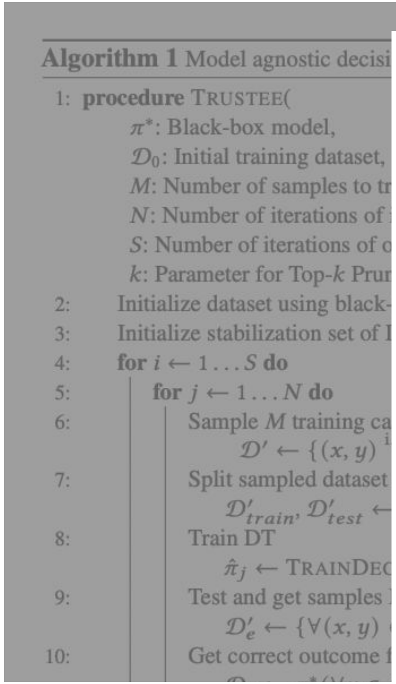
Algorithm 1 Model agnostic decision tree explanation extraction.

procedure TRUSTEE(
  π^*: Black-box model,
  $\mathcal{D}_0$: Initial training dataset,
  M: Number of samples to train the decision tree,
  N: Number of iterations of inner loop,
  S: Number of iterations of outer loop,
  k: Parameter for Top-k Pruning),

  Initialize dataset using black-box $\mathcal{D} \leftarrow \pi^*(\forall x \in \mathcal{D}_0)$
  Initialize stabilization set of DTs $\mathcal{R} \leftarrow \emptyset$
  for $i \leftarrow 1 \ldots S$ **do**
    for $j \leftarrow 1 \ldots N$ **do**
      Sample M training cases uniformly from $\mathcal{D}$
        $\mathcal{D}' \leftarrow \{(x, y) \overset{\text{i.i.d.}}{\sim} U(\mathcal{D})\}$
      Split sampled dataset for training and testing
        $\mathcal{D}'_{train}, \mathcal{D}'_{test} \leftarrow \text{TRAINTESTSPLIT}(\mathcal{D}')$
      Train DT
        $\hat{\pi}_j \leftarrow \text{TRAINDECISIONTREE}(\mathcal{D}'_{train})$
      Test and get samples DT misclassifies
        $\mathcal{D}'_e \leftarrow \{\forall (x, y) \in \mathcal{D}'_{test} \mid \hat{\pi}_j(x) \neq \pi^*(x)\}$
      Get correct outcome from black-box

Additional details (see paper)

Algorithmic Description of Trustee


Algorithm 1 Model agnostic decisi

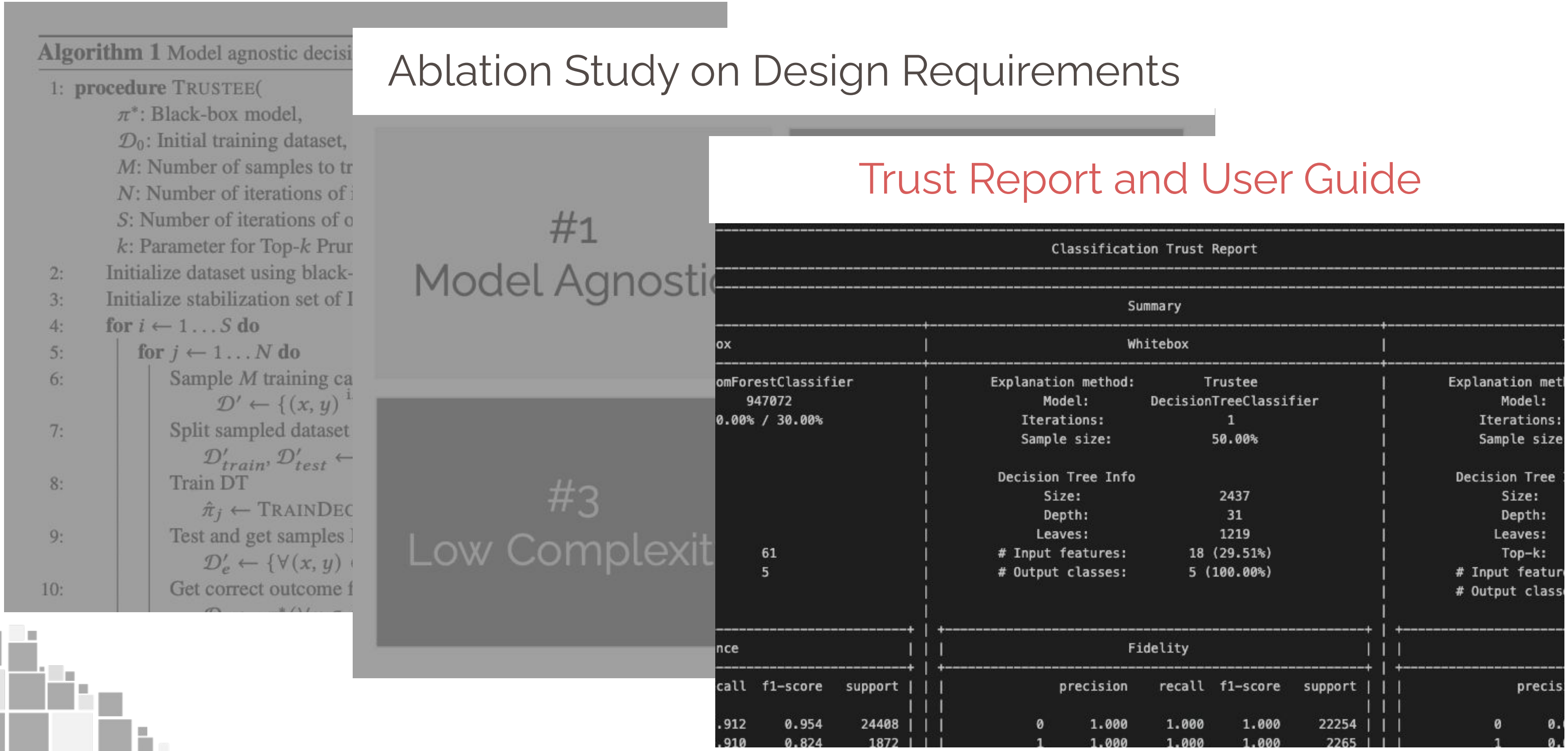
procedure TRUSTEE(
π^*: Black-box model,
$\mathcal{D}_0$: Initial training dataset,
M: Number of samples to tr
N: Number of iterations of i
S: Number of iterations of o
k: Parameter for Top-k Prun
Initialize dataset using black-
Initialize stabilization set of I
for $i \leftarrow 1 \ldots S$ **do**
for $j \leftarrow 1 \ldots N$ **do**
Sample M training ca
$\mathcal{D}' \leftarrow \{(x, y)\}$ i
Split sampled dataset
$\mathcal{D}'_{train}, \mathcal{D}'_{test} \leftarrow$
Train DT
$\hat{\pi}_j \leftarrow$ TRAINDEC
Test and get samples I
$\mathcal{D}'_e \leftarrow \{\forall (x, y)$
Get correct outcome f

Ablation Study on Design Requirements

#1 Model Agnostic	#2 High Fidelity
#3 Low Complexity	#4 Stable

Additional details (see paper)

Algorithmic Description of Trustee



Trustee Python package

Search projects

Help Sponsors Log in Register

trustee 1.1.1

`pip install trustee`

Latest version

Released: Aug 28, 2022

This package implements the Trustee framework to extract decision tree explanation from black-box ML models.

Navigation

- Project description
- Release history
- Download files

Project links

- Homepage
- Repository

Statistics

GitHub statistics:

- Stars: 2
- Forks: 0
- Open issues/PRs: 0

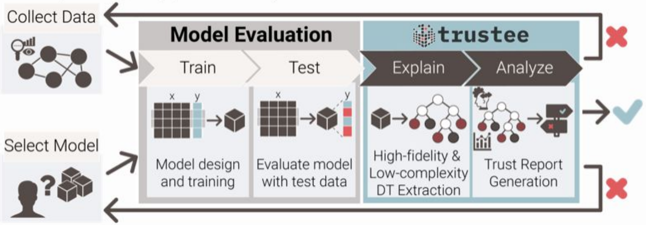
View statistics for this project via

Project description

trustee

This package implements the `trustee` framework to extract decision tree explanation from black-box ML models. For more information, please visit the documentation website.

Standard AI/ML development pipeline extended by Trustee.



Downloads 43	Downloads 175	Downloads 6k
Daily	Weekly	Total

Trustee 1.1.1 documentation

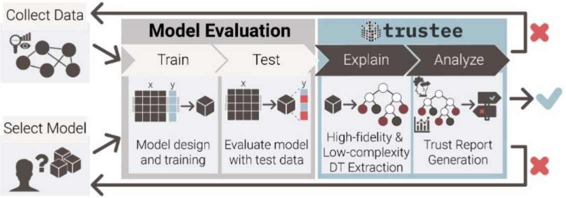
Search

- API
- Examples
  - TrustReport for Classification
  - TrustReport for Regression
  - RegressionTrustee
  - ClassificationTrustee

Github Repo Use Cases Tech Report

Overview

Trustee is a framework to extract decision tree explanation from black-box ML models.



Standard AI/ML development pipeline extended by Trustee.

Getting Started

This section contains basic information and instructions to get started with Trustee.

Python Version

Trustee supports Python >=3.7.

Install Trustee

Use the following command to install Trustee:

```
$ pip install trustee
```

Sample Code

```
from sklearn import datasets
```

Conclusions

1. ML in high-stakes requires trust
2. Trustee improves trust!
3. Trustee can be used with any existing model
4. Trustee is ready to be used!
   - Just download our Python package

Thank you!

Arthur Jacobs
asjacobs@inf.ufrgs.br

trustee

https://trusteeml.github.io

Trustee Python package

- https://pypi.org/project/trustee/

Trustee Repository

- https://github.com/TrusteeML/trustee

Use Cases Repository

- https://github.com/TrusteeML/emperor

.inf INSTITUTO DE INFORMÁTICA UFRGS
NIKSUN
UFMS
UNIVERSITY OF CALIFORNIA SANTA BARBARA

Backup

Existing approaches

Method	Model Agnostic	High Fidelity	Domain-specific Pruning
Trepan	✓	—	—
dtextract	✓	—	—
VIPER	—	—	—
Metis	—	—	—
trustee	✓	✓	✓

Use Case #4: Anomaly Detection for Mirai Attacks

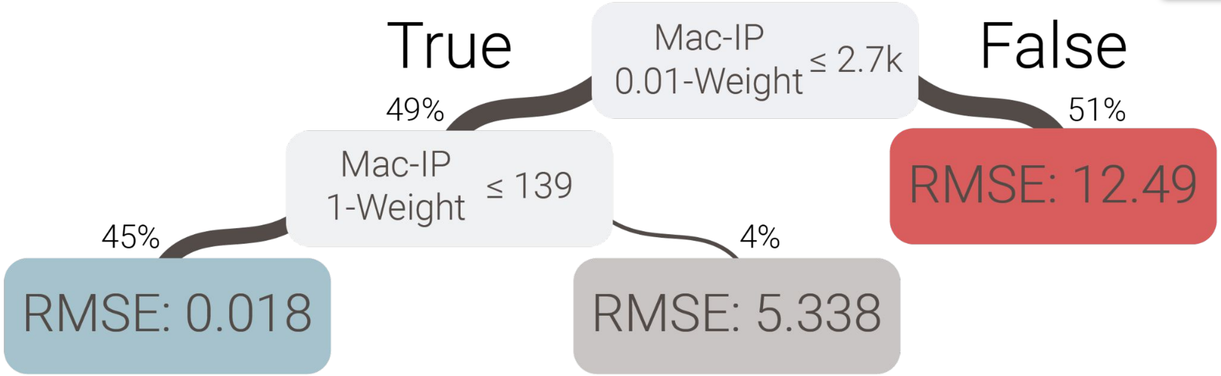
Problem Setup

- Selected publications:
  - *"Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection" — Mirsky et al., 2018*
- Proposal:
  - **Model**: Kitsune, an ensemble of neural networks, trained with unsupervised learning, for anomaly detection
  - **Features**: 110 features based on traffic statistics (*e.g.*, number of packets per **time window**).
  - **Dataset**: synthetic Mirai attack trace.
- Results:
  - Reported R-squared: 0.99
  - Reproduced R-squared: 0.99

Use Case #4: Anomaly Detection for Mirai Attacks

Explanation

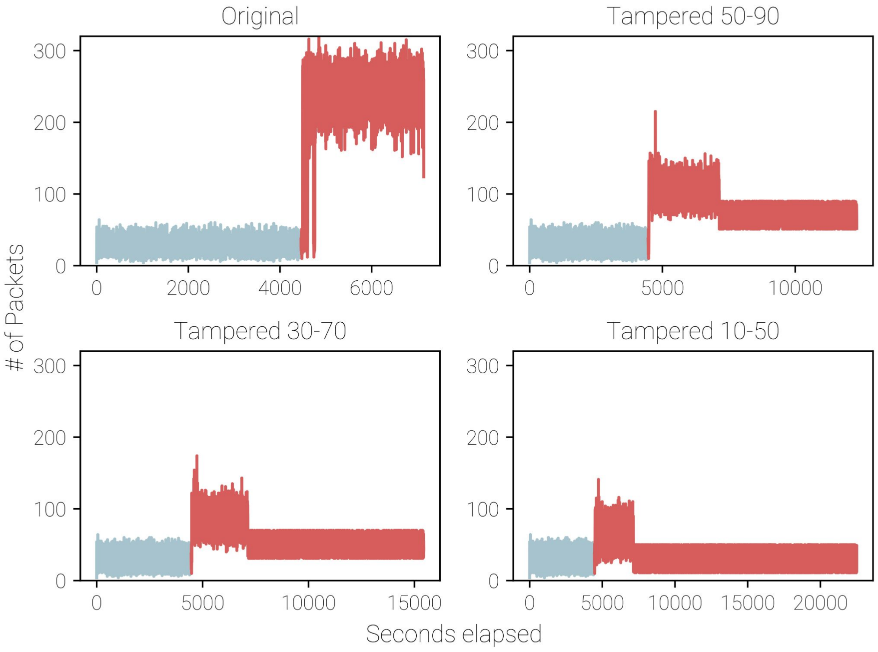
Fidelity: 0.99
Top-3 pruning
5 nodes



Use Case #4: Anomaly Detection for Mirai Attacks

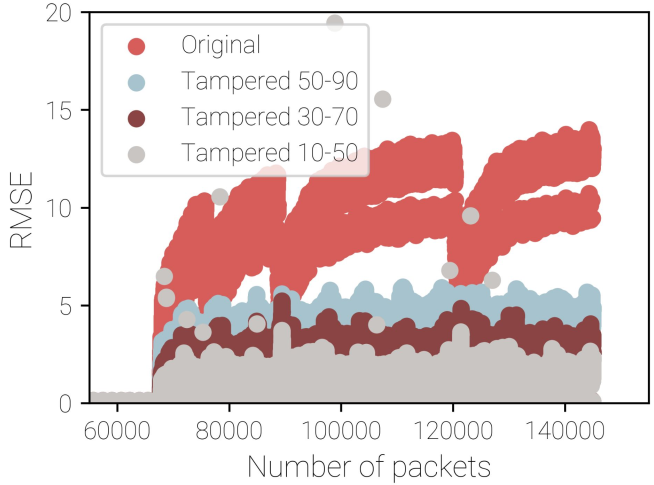
Validation

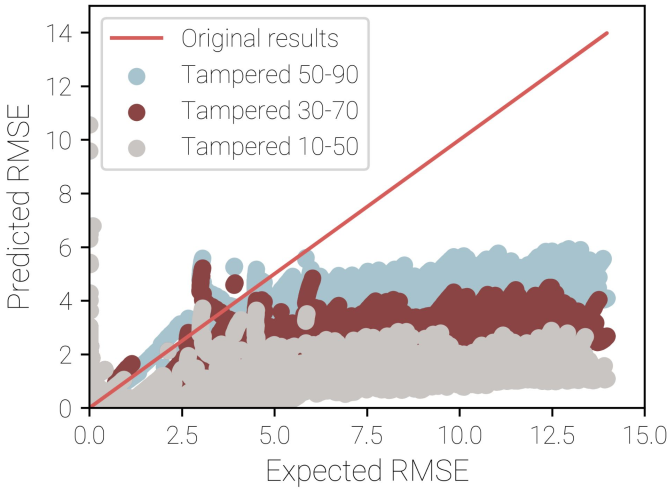
- Validation datasets:



Use Case #4: Anomaly Detection for Mirai Attacks

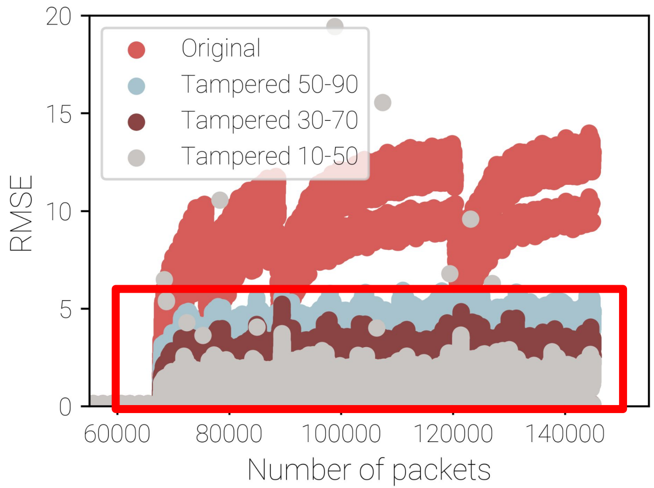
Validation

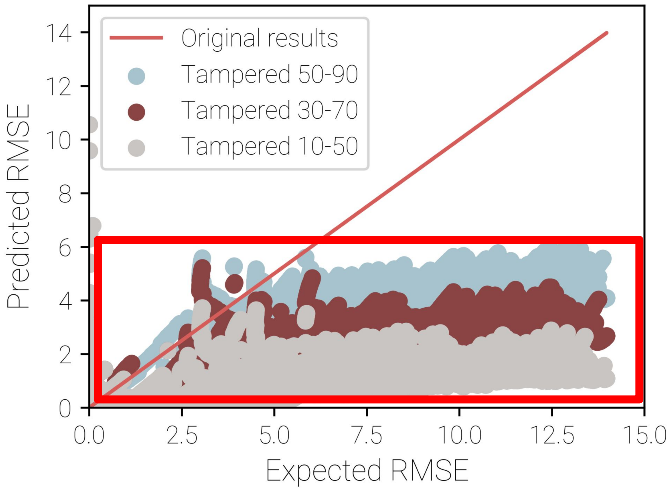




Use Case #4: Anomaly Detection for Mirai Attacks

Validation





Takeaway: the model is overfitted to training data and fails to identify o.o.d. samples!