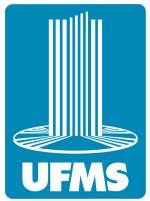
ACM SIGCOMM 2023 Tutorial:
Closed-Loop "ML for Networks" Pipelines

Trustee: An Augmented ML Pipeline for Explaining ML Models

Ronaldo A Ferreira
UFMS

September 10, 2023



Traditional AI/ML Pipeline

- The “traditional AI/ML pipeline” consists of:
  - A training task characterized by a model specification
  - A (labelled) training dataset
  - An independent and identically distributed (IID) evaluation procedure
  - A (single) metric (e.g., F1-score) that measures the model’s expected predictive performance on data drawn from the training distribution

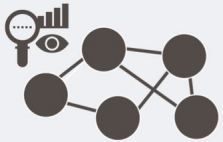
Traditional AI/ML Pipeline

- Main criticisms of the “traditional ML pipeline” include:
  - The pipeline’s output (i.e., best-performing ML model) is prone to suffer from the problem of underspecification
  - The evaluation of the pipeline’s output is agnostic to the particular inductive biases encoded by the trained model

How does it work?

Traditional AI/ML Development Pipeline

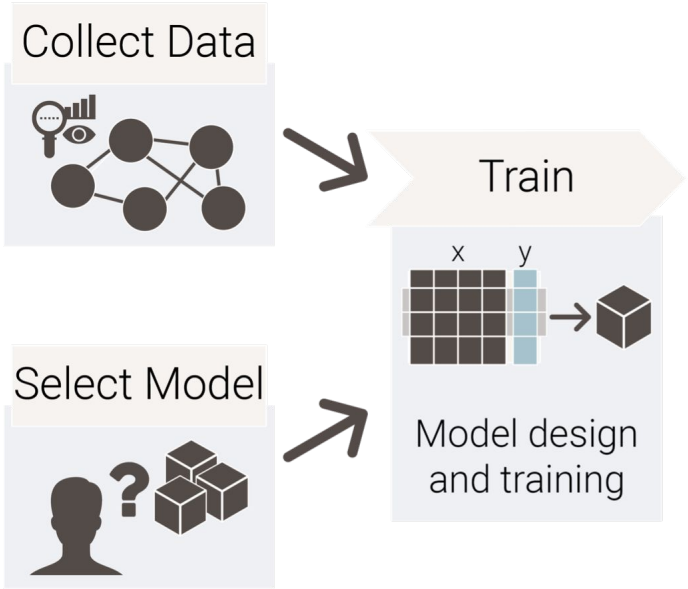
Collect Data



Select Model

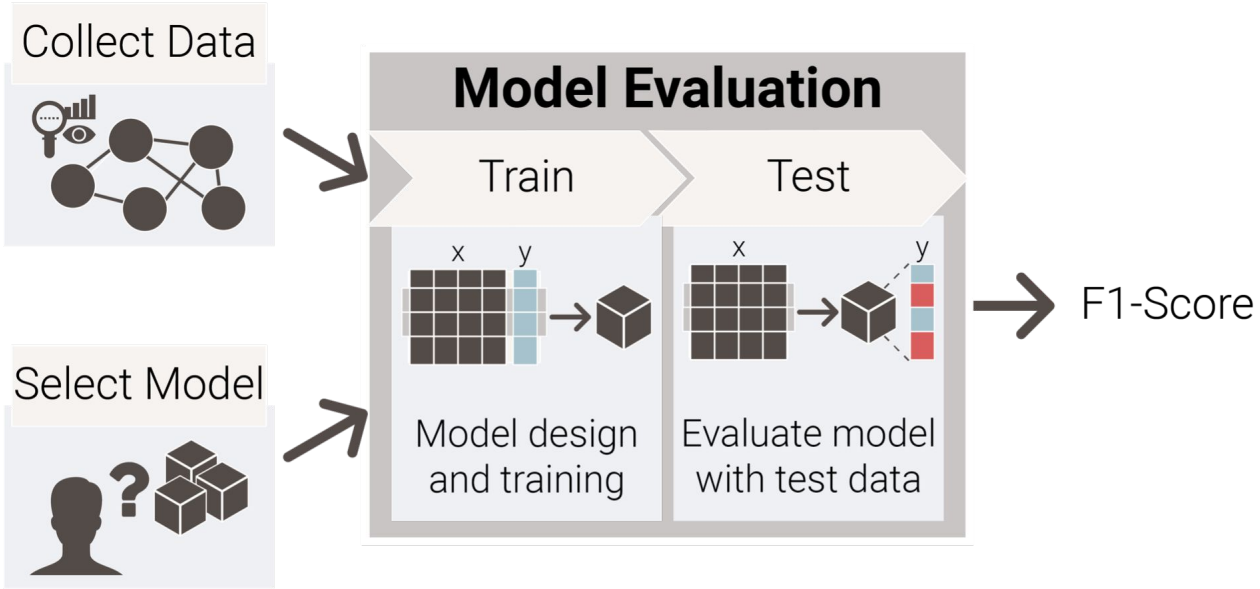
How does it work?

Traditional AI/ML Development Pipeline



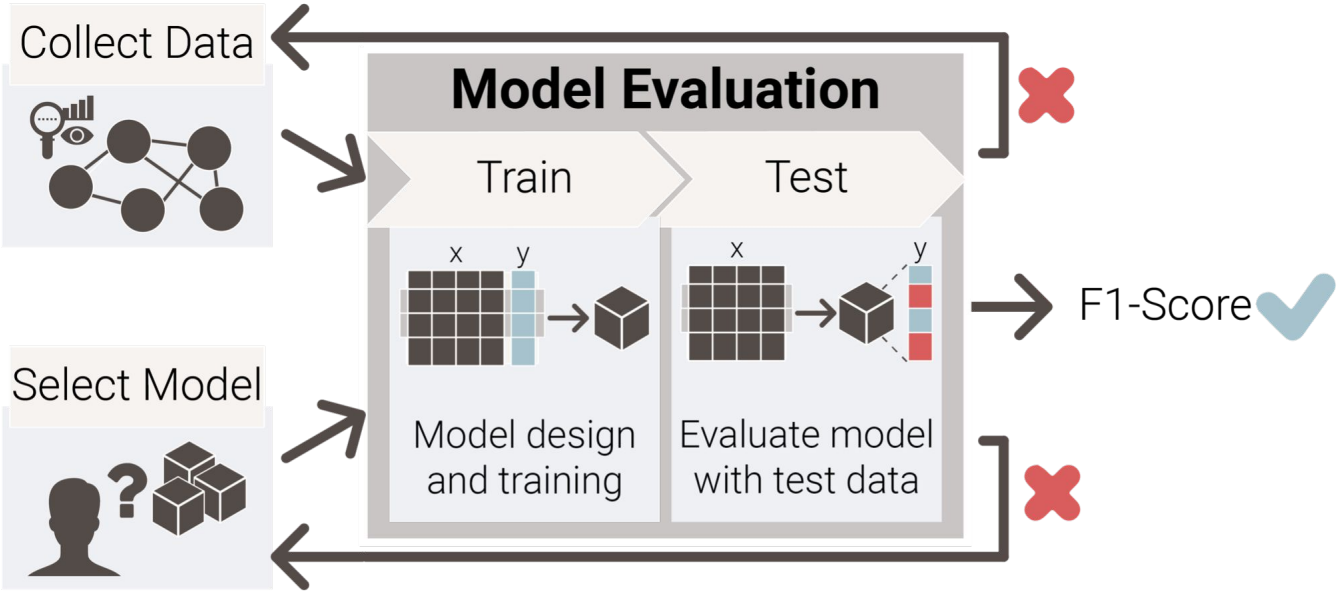
How does it work?

Traditional AI/ML Development Pipeline



How does it work?

Traditional AI/ML Development Pipeline



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!

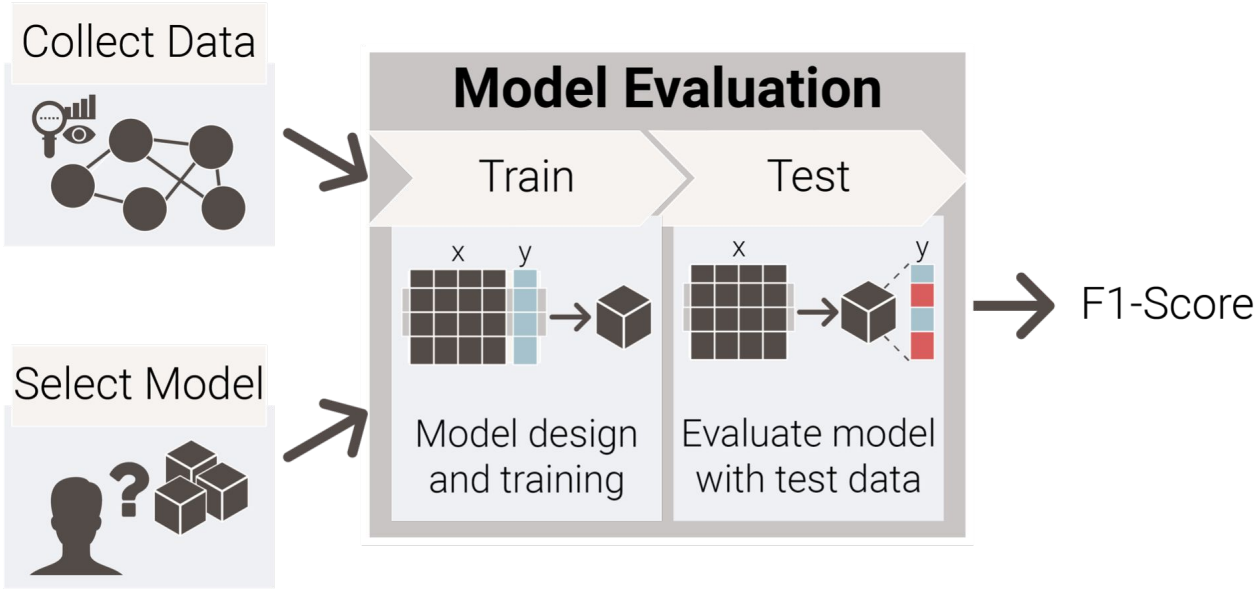
O.O.D. Samples

Model does not generalize!

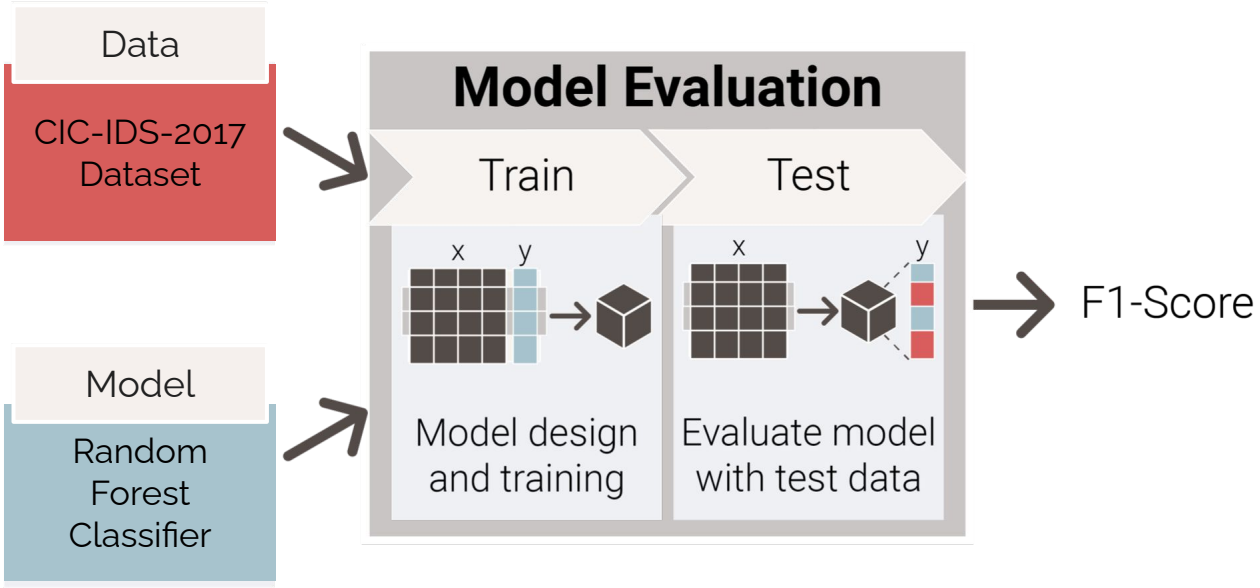
Spurious Correlations

Model picks up wrong correlations in the data!

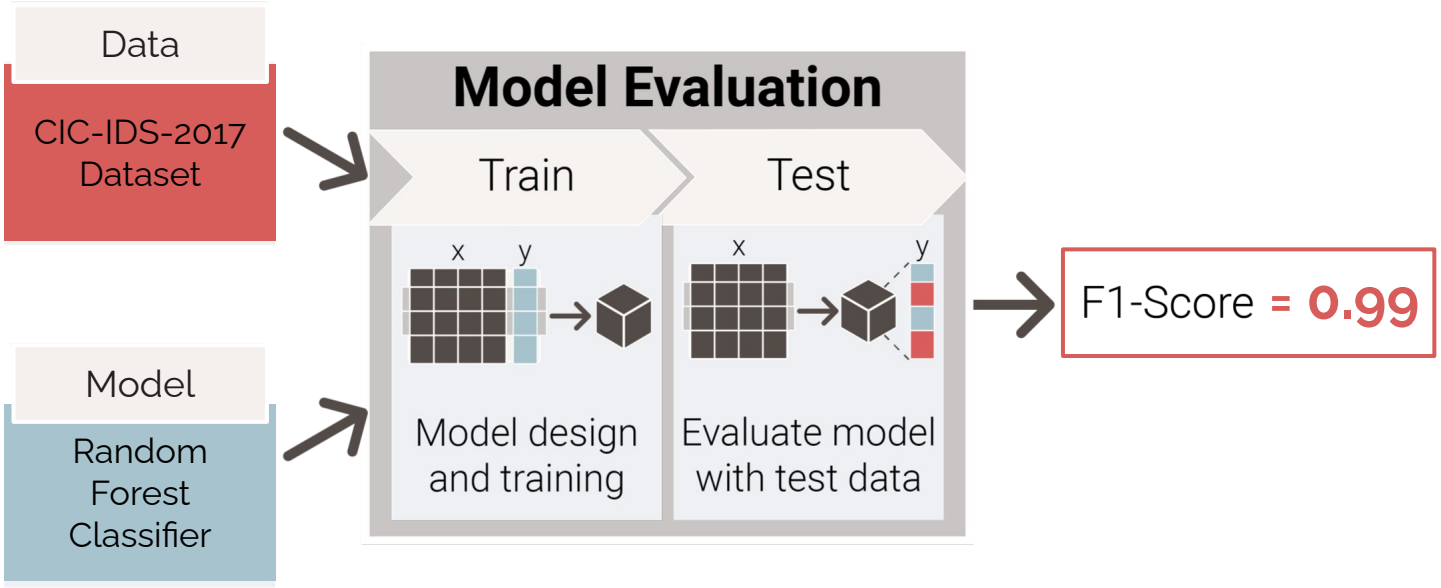
Consider this example…



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?

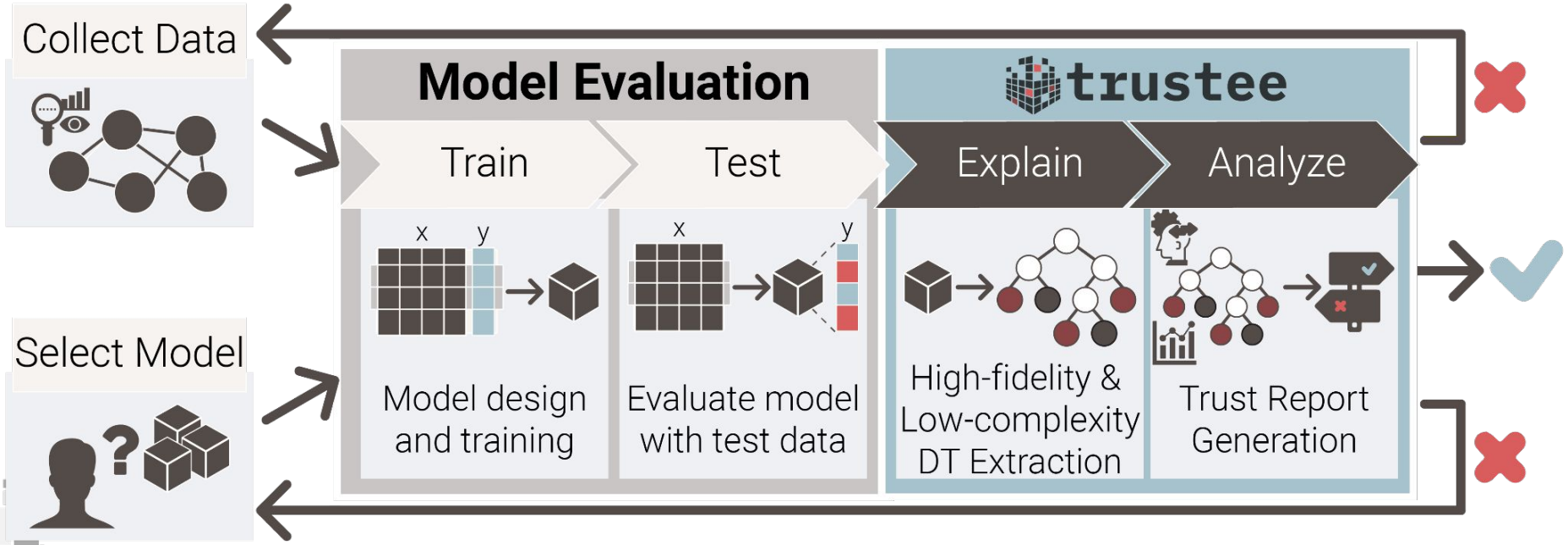
Trust in AI/ML model

=

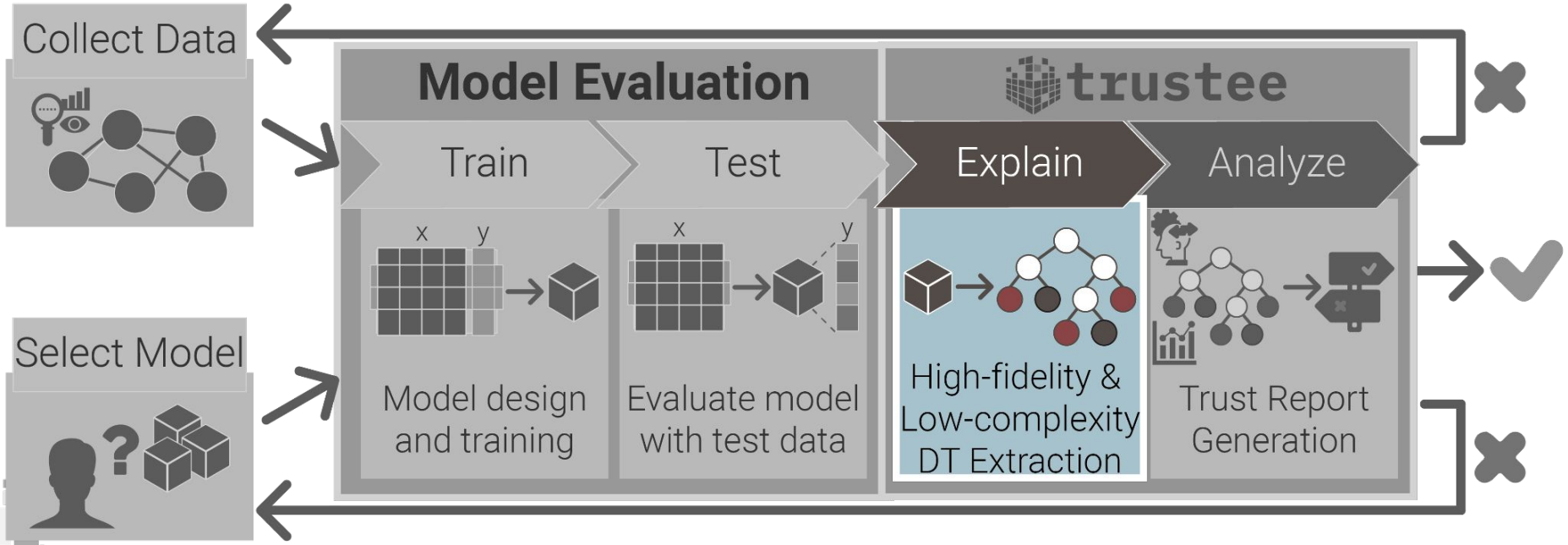
Hand over control to the AI/ML model


trustee

Augmented
AI/ML Development Pipeline



Augmented
AI/ML Development Pipeline



trustee

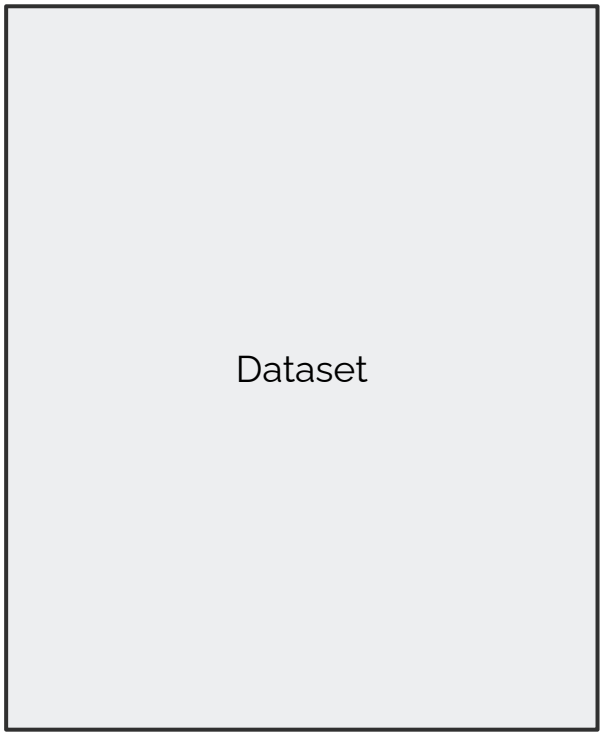
- Why do we need another XAI method?

trustee

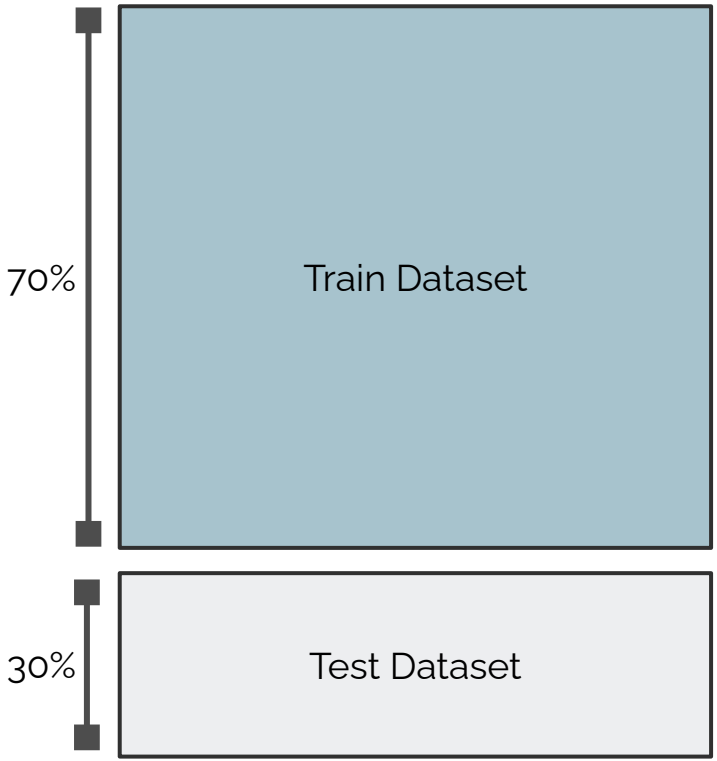
- Why do we need another XAI method?
- No existing global XAI method satisfies four requirements we desire

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

trustee

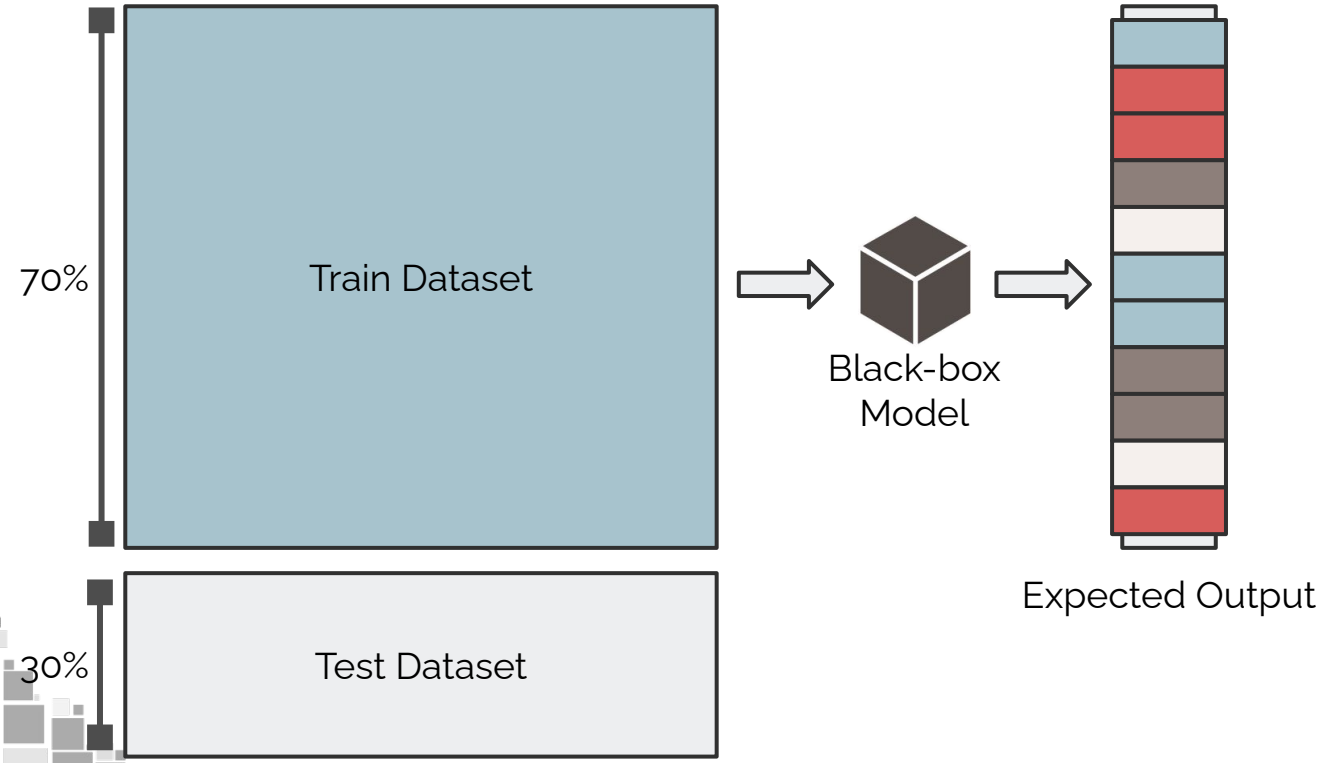


Black-box
Model

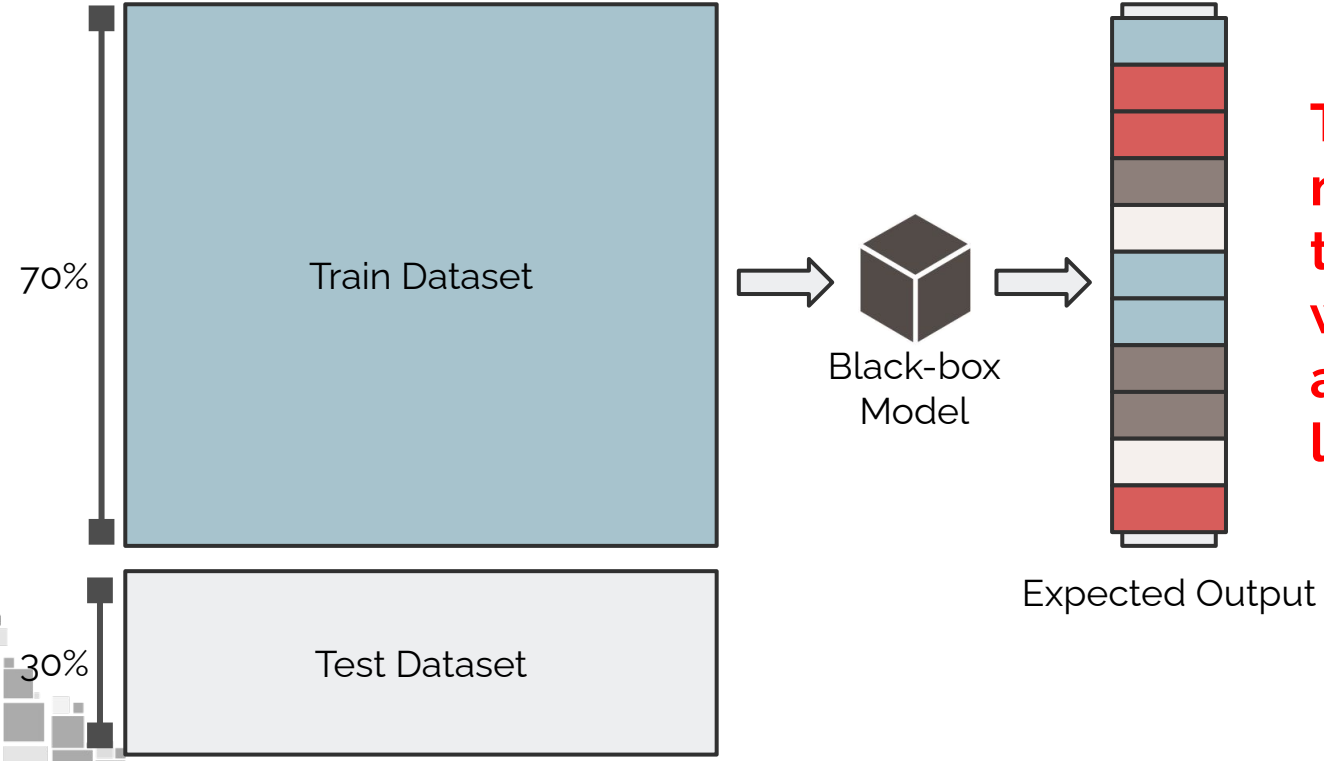


Black-box
Model

trustee


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

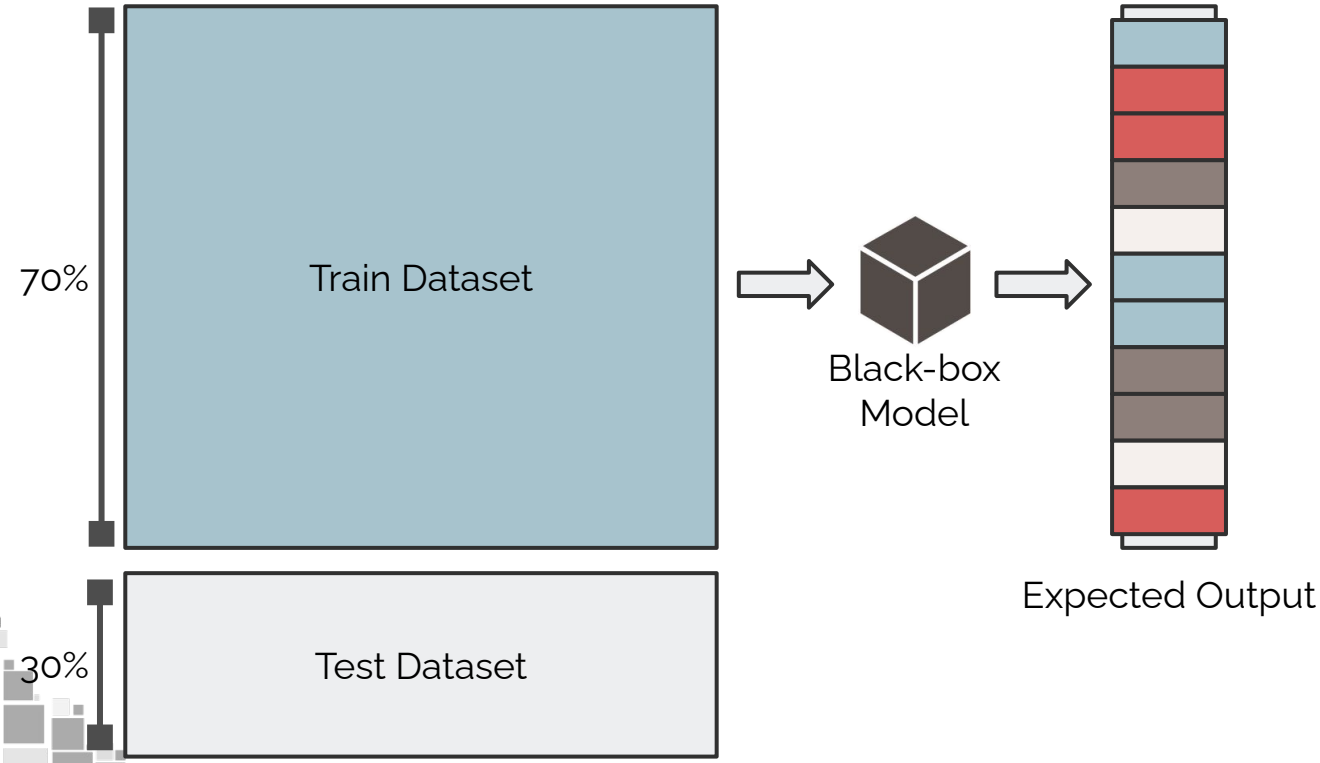
trustee



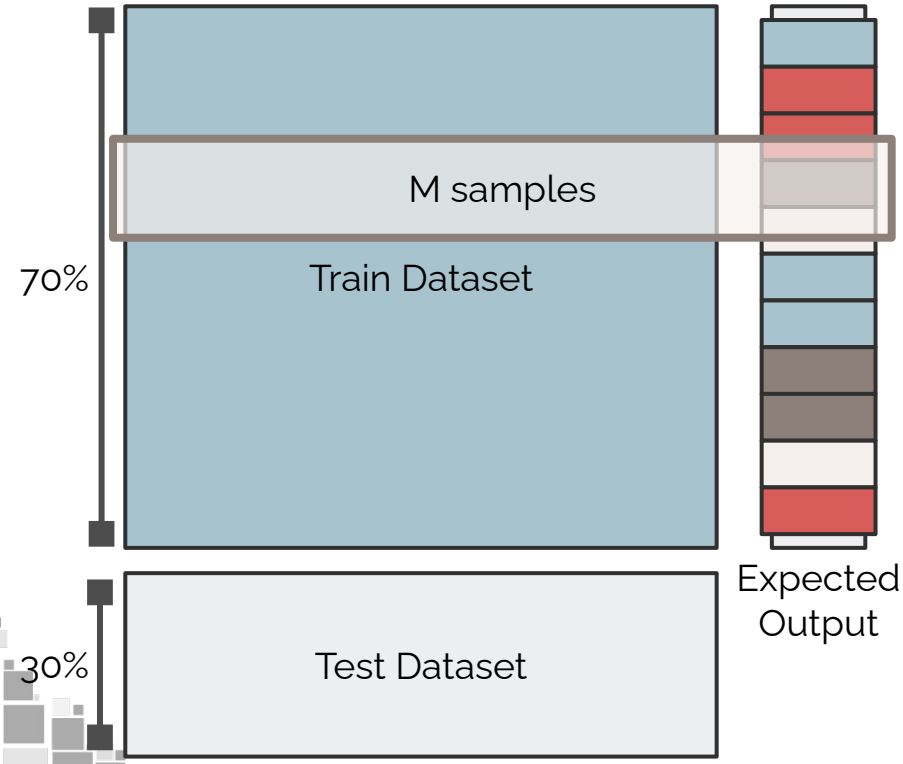
Trustee uses this model as an expert to train a student (a white-box model), akin to imitation learning.

#1
Model
Agnostic

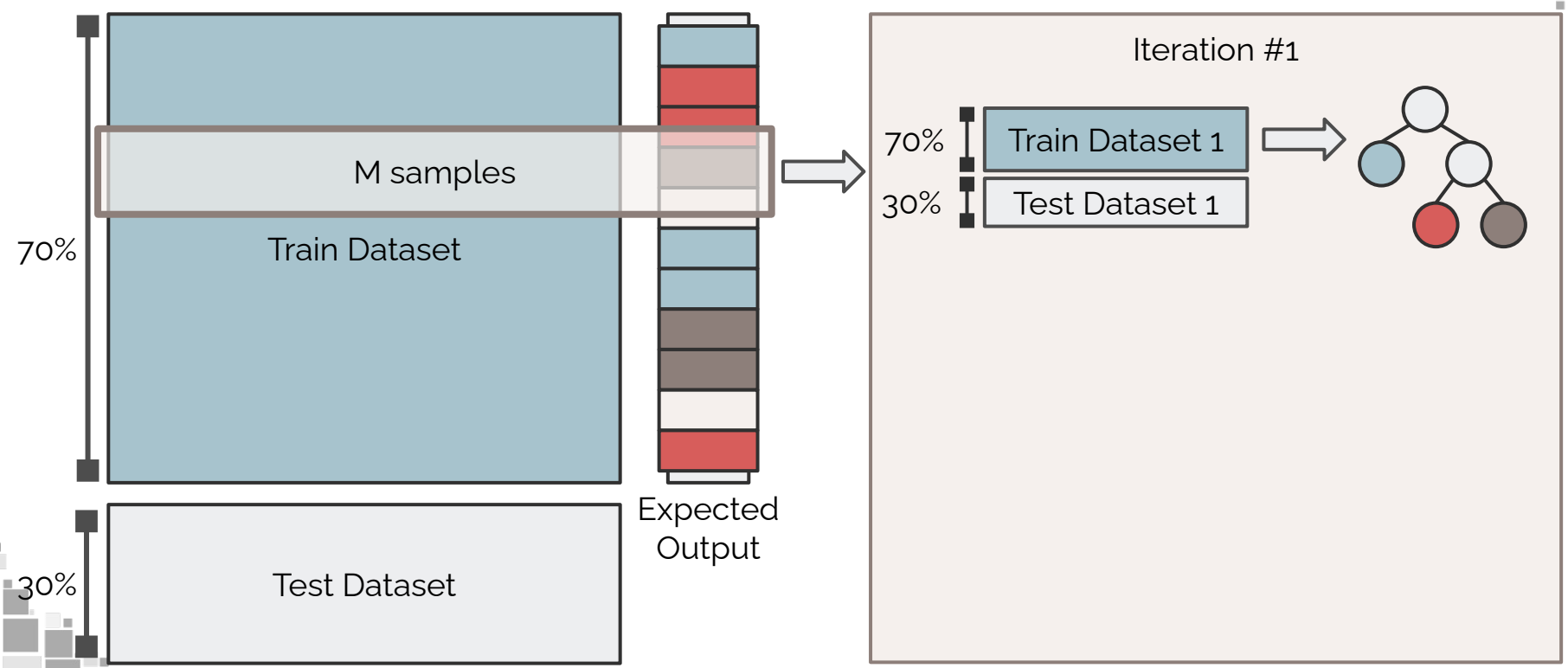
trustee



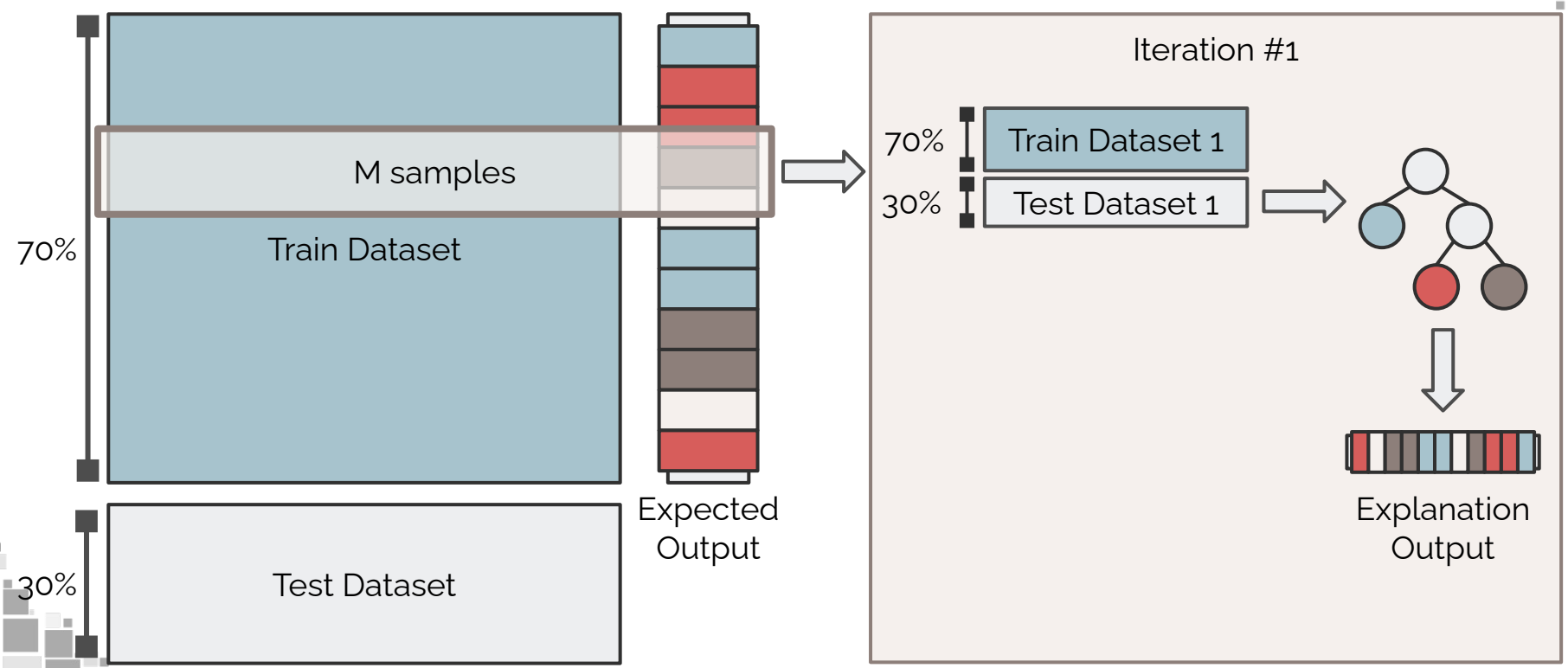
trustee



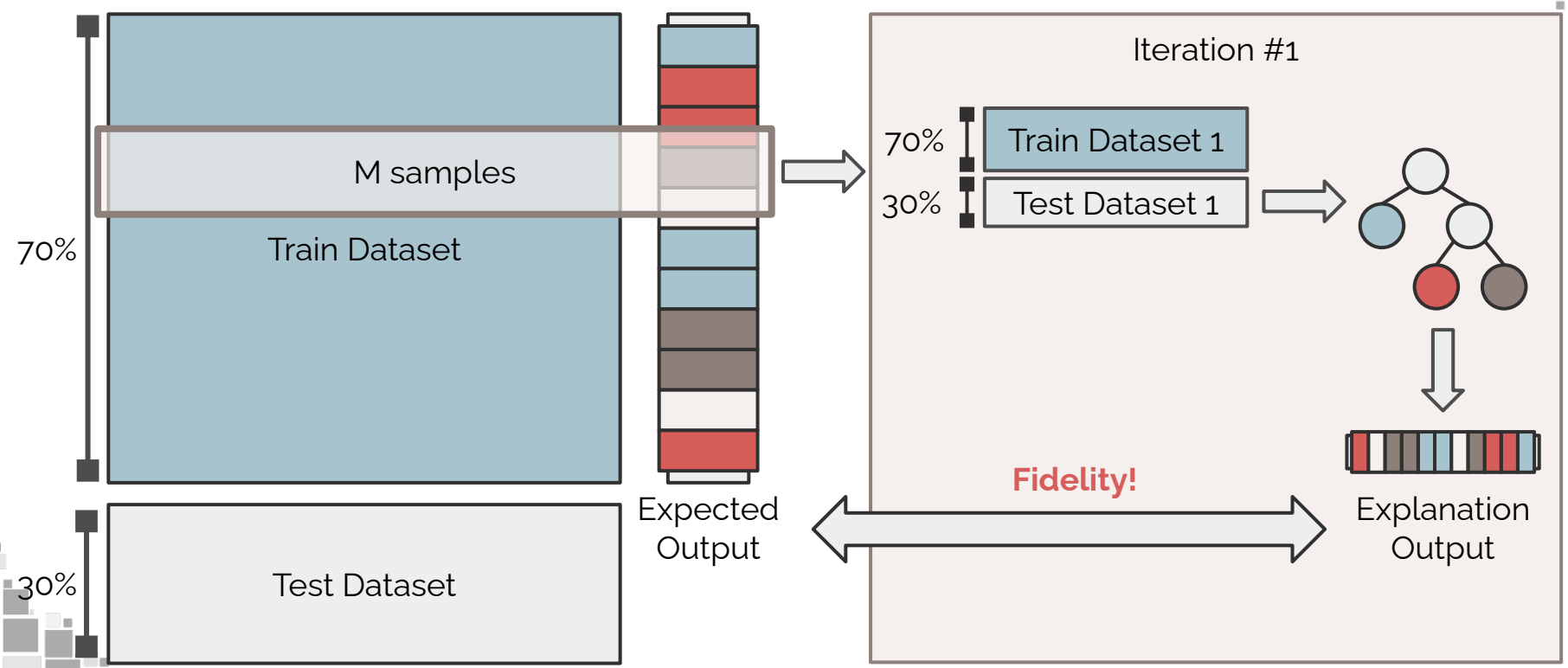
trustee


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

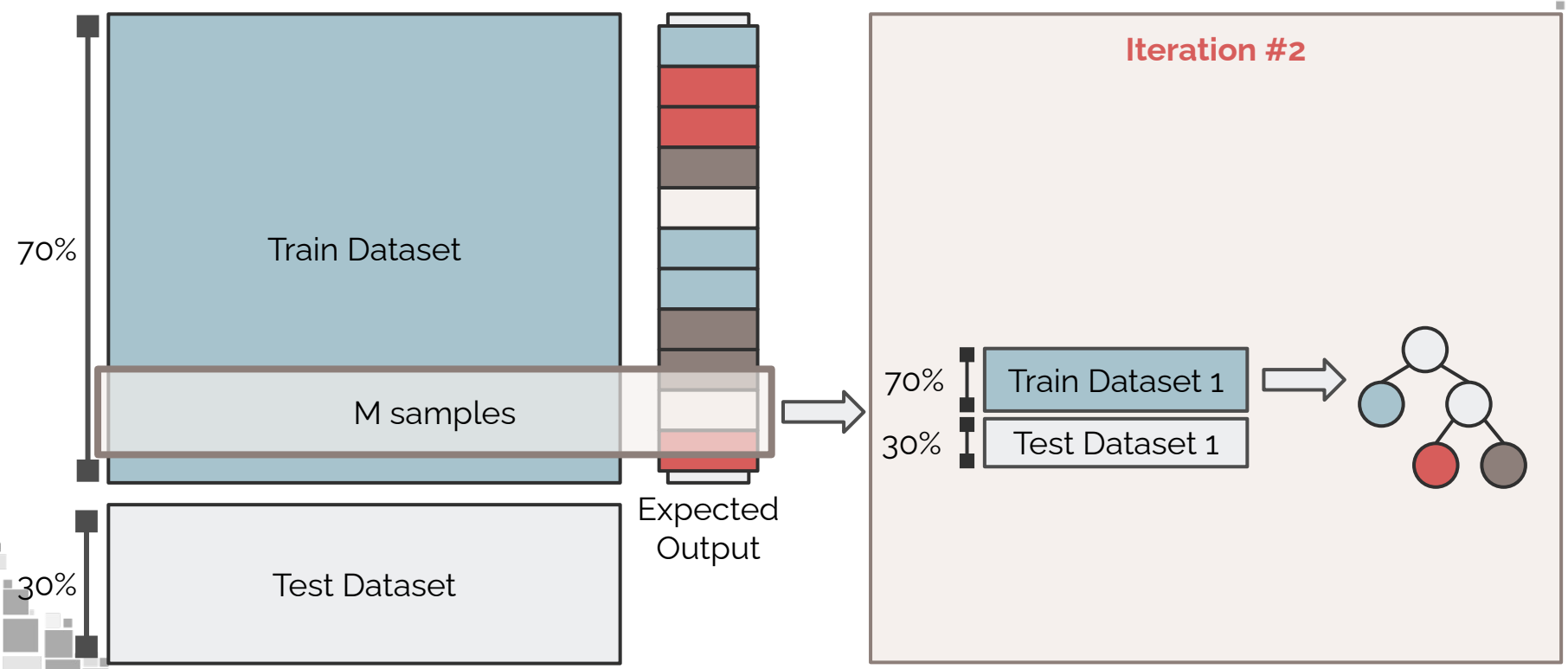
trustee


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

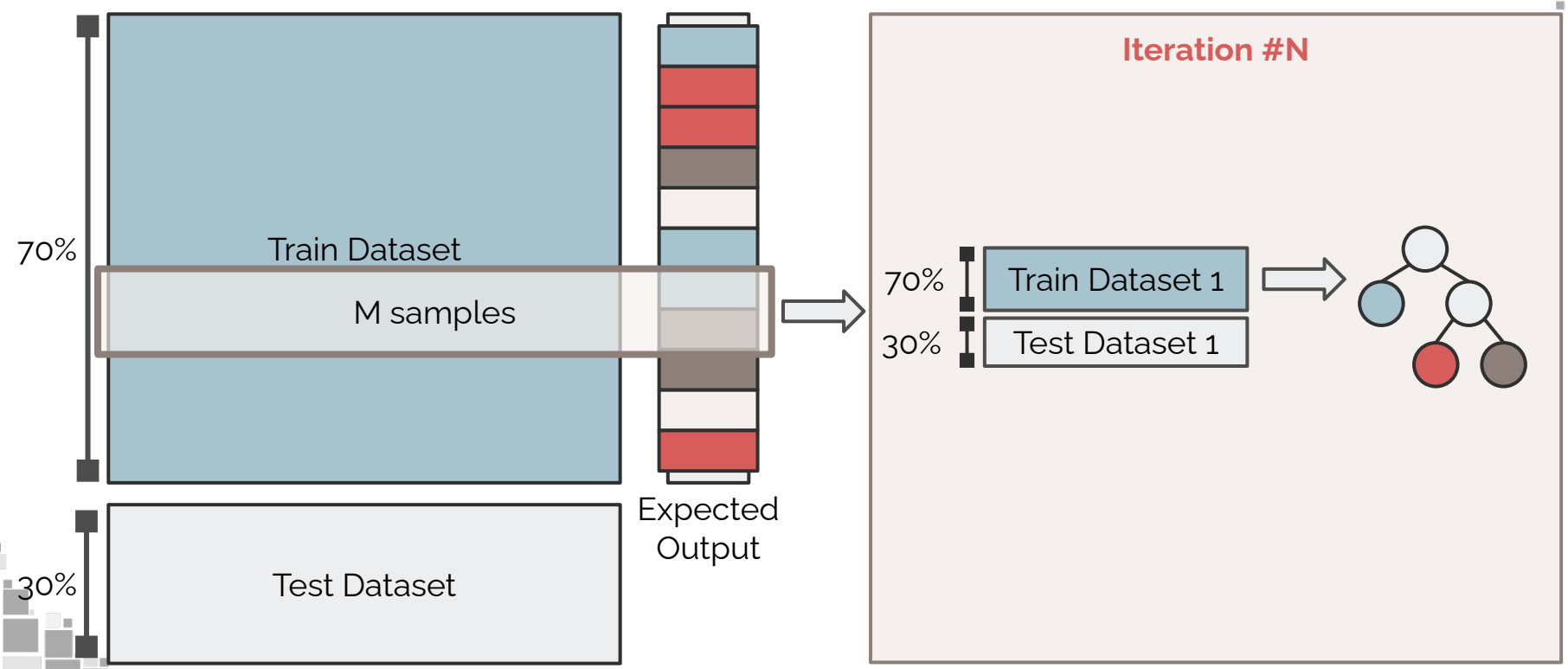
trustee


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

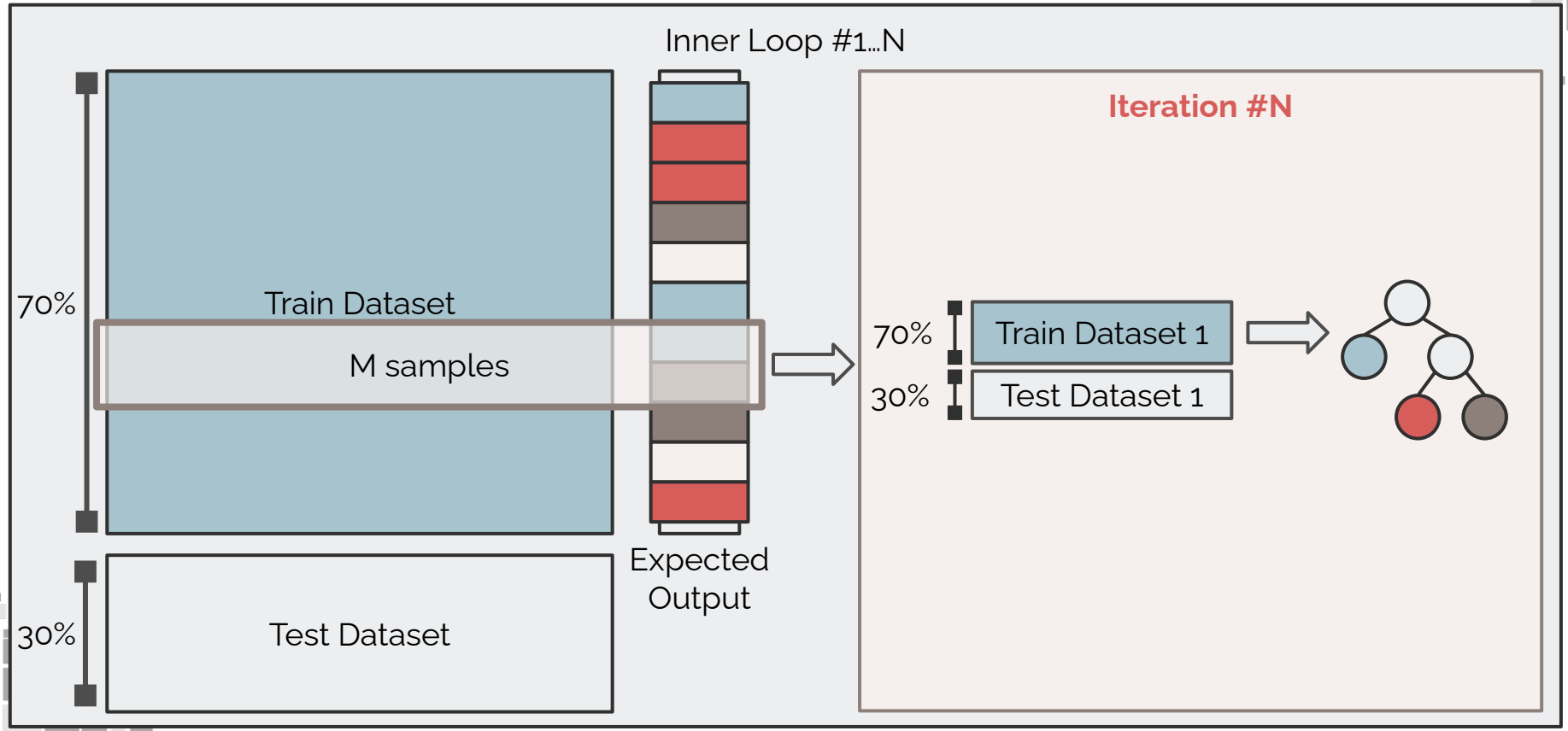
trustee


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

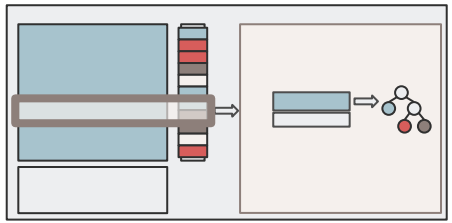
trustee

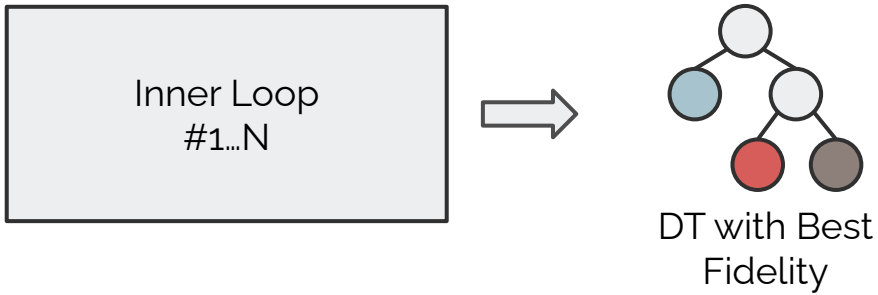

70%
Train Dataset
M samples
30%
Test Dataset
Expected Output
Iteration #N
70%
Train Dataset 1
30%
Test Dataset 1

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

trustee

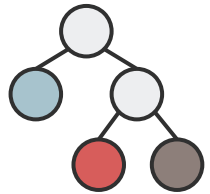



Inner Loop
#1…N
DT with Best
Fidelity

#2
High
Fidelity

trustee

Inner Loop
#1…N



DT with Best
Fidelity

trustee

How does Trustee achieve a high-fidelity explanation?

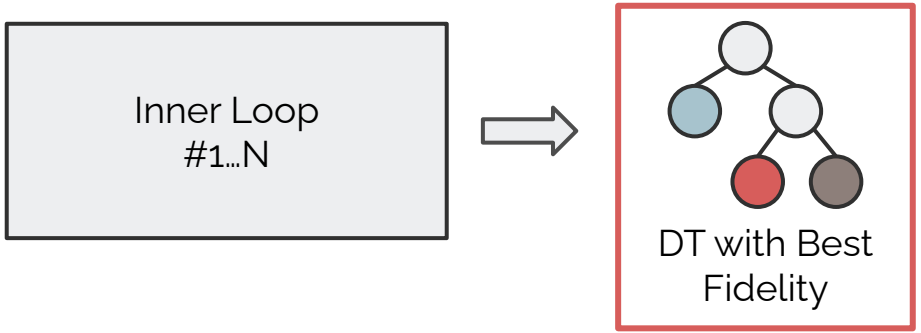
How does Trustee achieve a high-fidelity explanation?

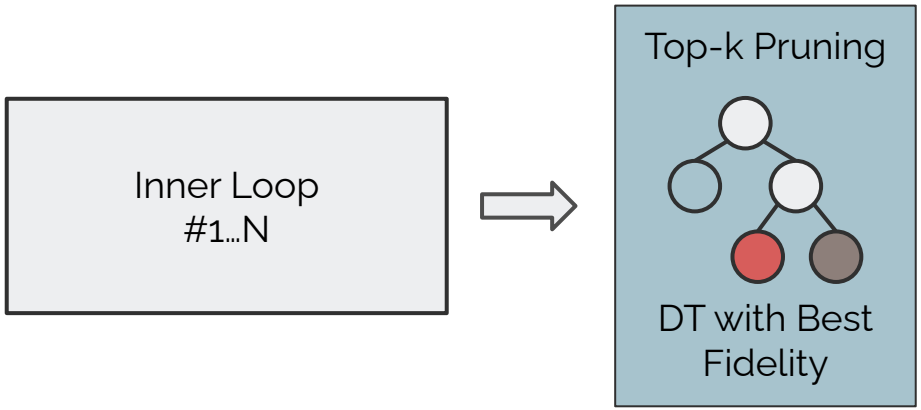
- It optimizes for fidelity
  - The computed explanation is not necessarily the most accurate
- It does data augmentation
  - It overcorrects for the data samples for which the DT model makes wrong decisions

trustee

The generated DT may be too large!

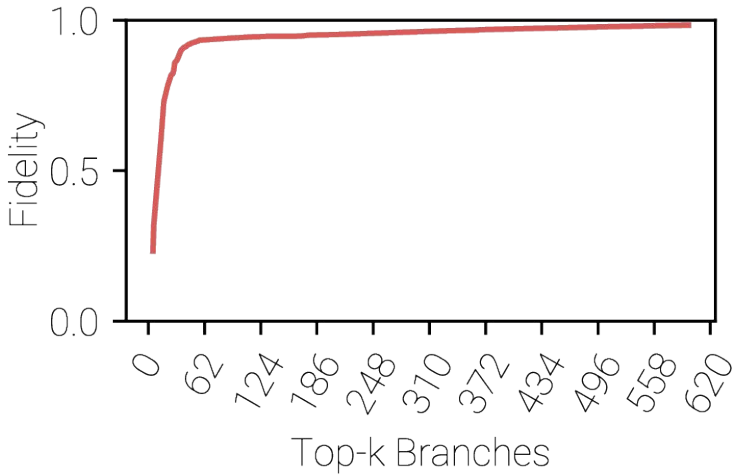
Size matters!



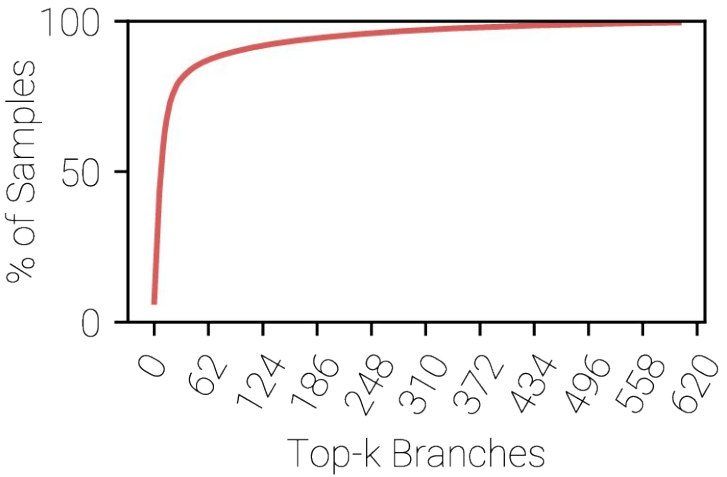

Inner Loop
#1…N
Top-k Pruning
DT with Best
Fidelity

Top-k Pruning

Fidelity

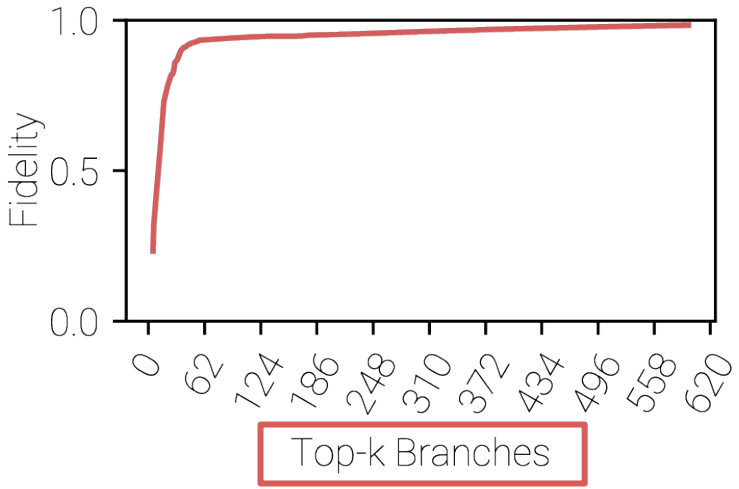


Samples

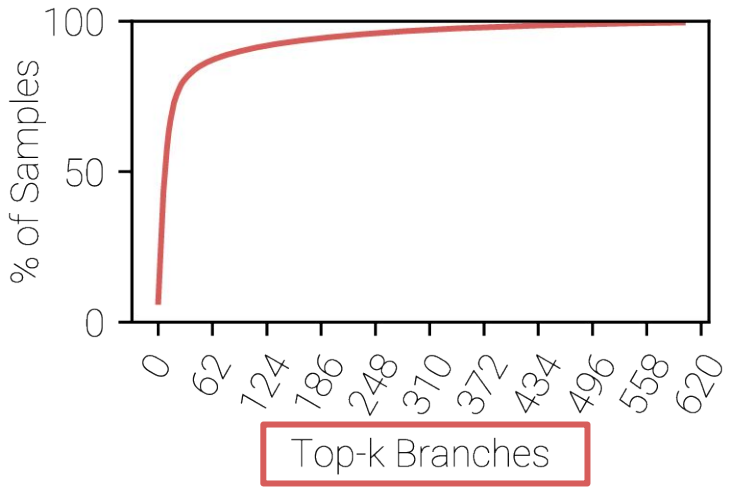


Top-k Pruning

Fidelity



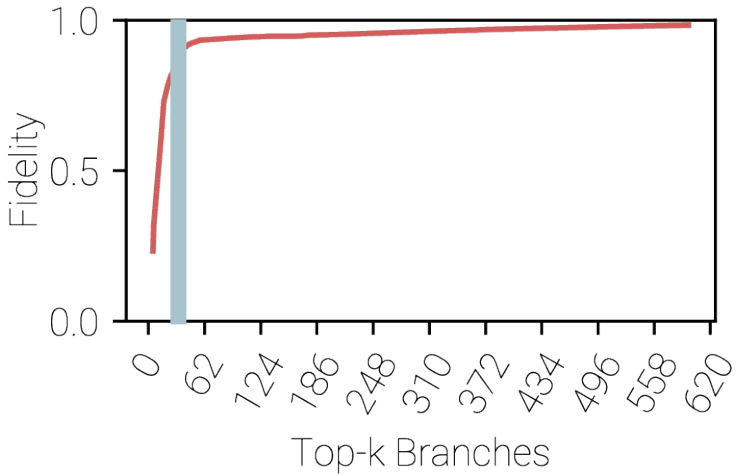
Samples



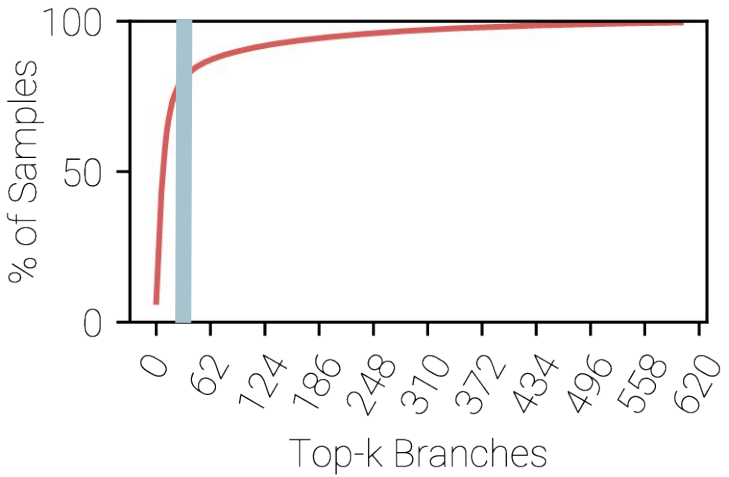
Diminishing returns!

Top-k Pruning

Fidelity



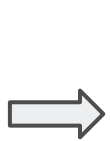
Samples



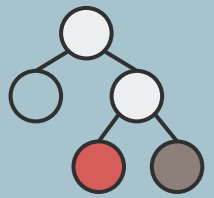
#3
Low
Complexity

trustee

Inner Loop
#1…N



Top-k Pruning



DT with Best
Fidelity

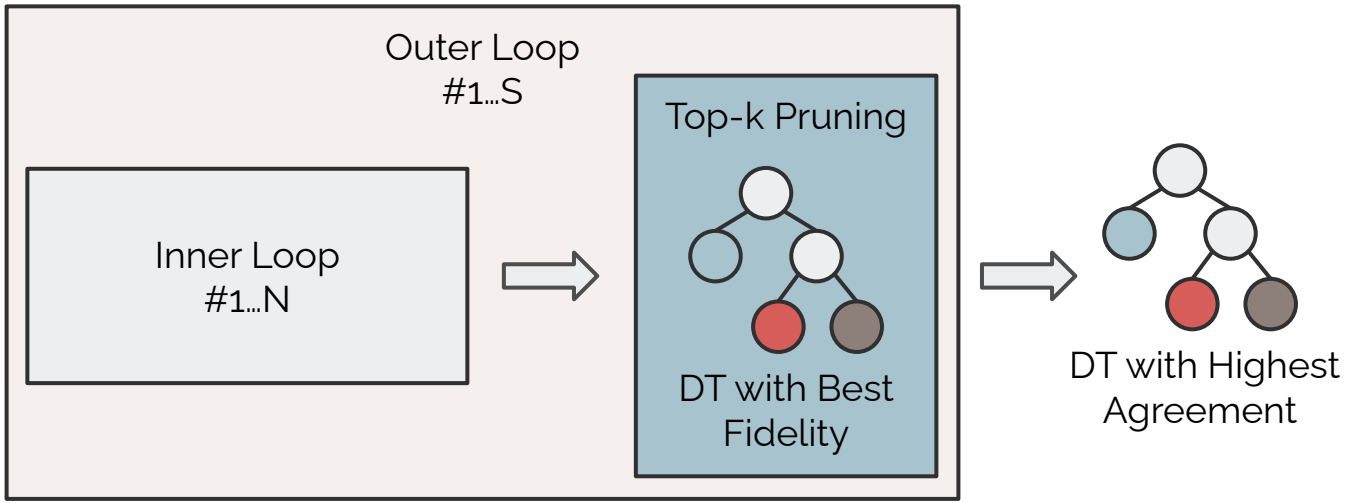
trustee

How does Trustee prevent obviously misleading explanations?

trustee

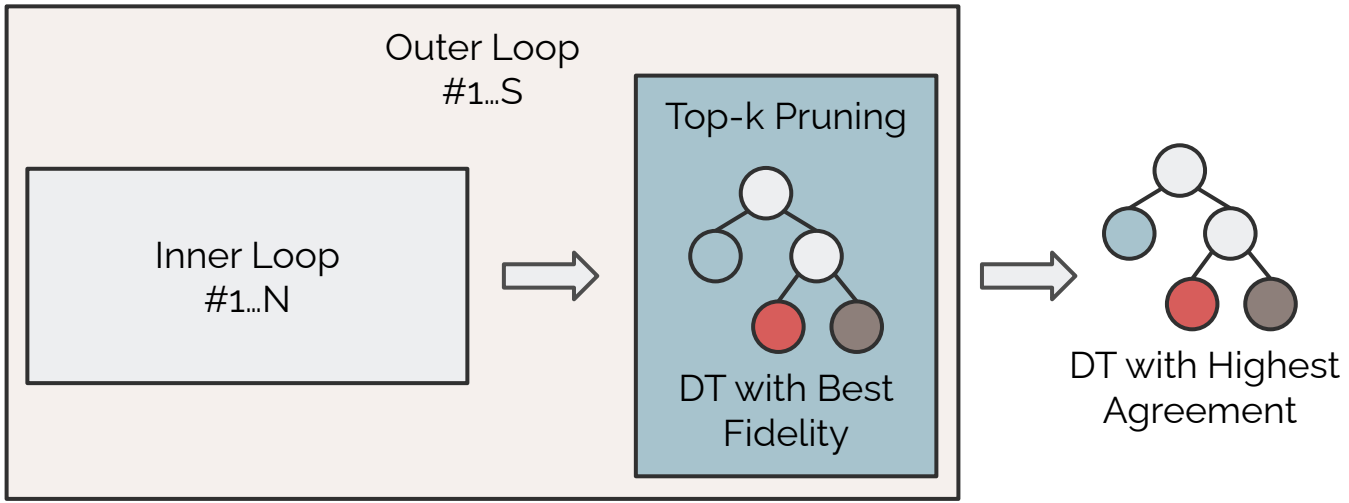
How does Trustee prevent obviously misleading explanations?

- It has an outer loop to select the DT with the highest mean agreement with the other explanations

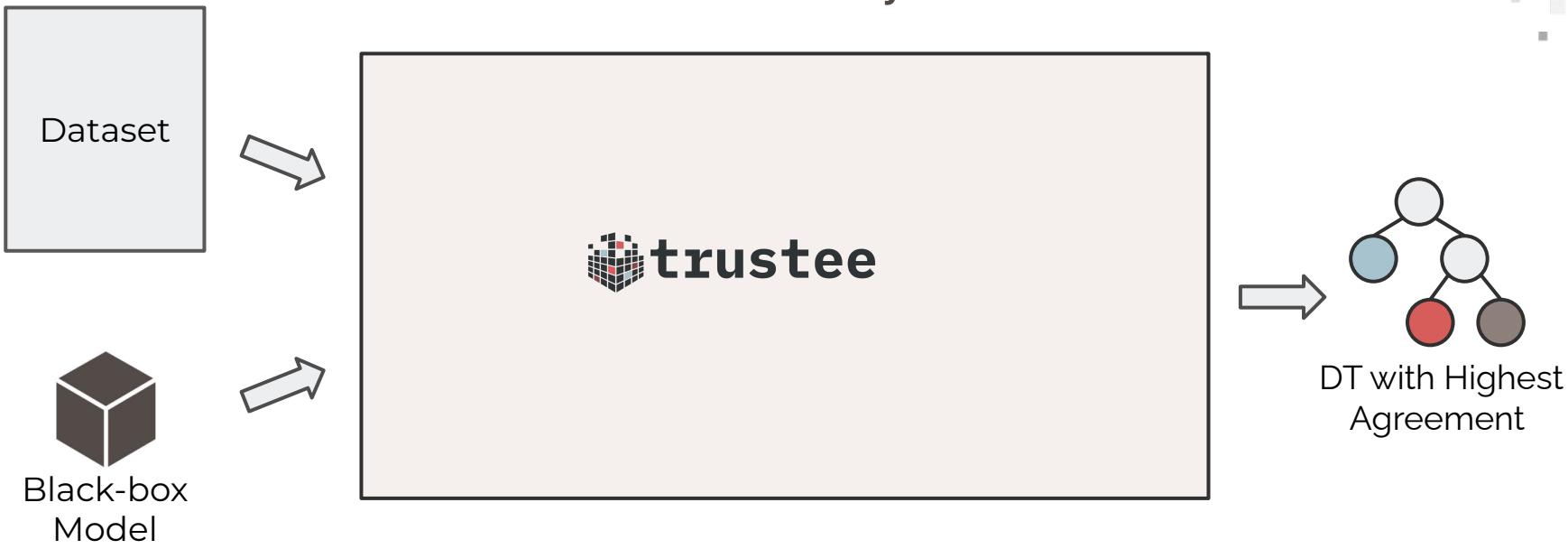


#4
Stable

trustee

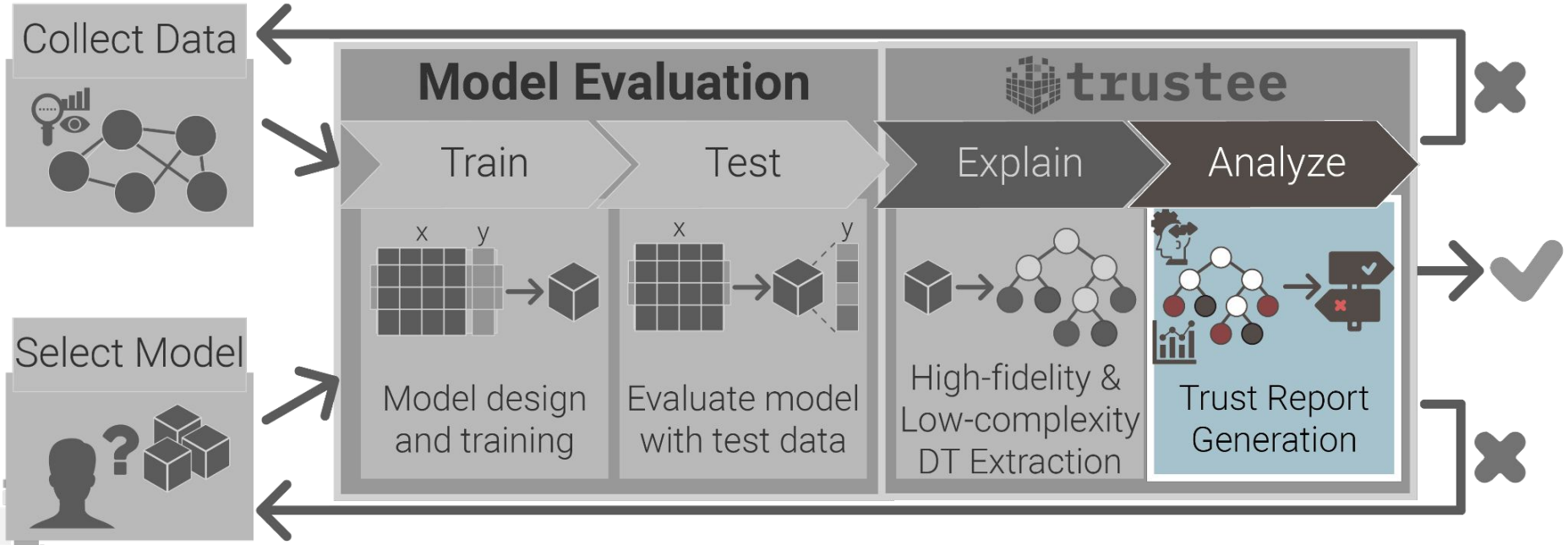


Summary



- Inner loop computes high-fidelity explanations
- Outer loop computes "the most stable explanation," i.e., the one with the highest mean agreement with the other explanations

Augmented
AI/ML Development Pipeline



Generating Trust Reports

Underspecification issues!

(revisited)

Shortcut Learning

Model takes shortcuts to classify data!

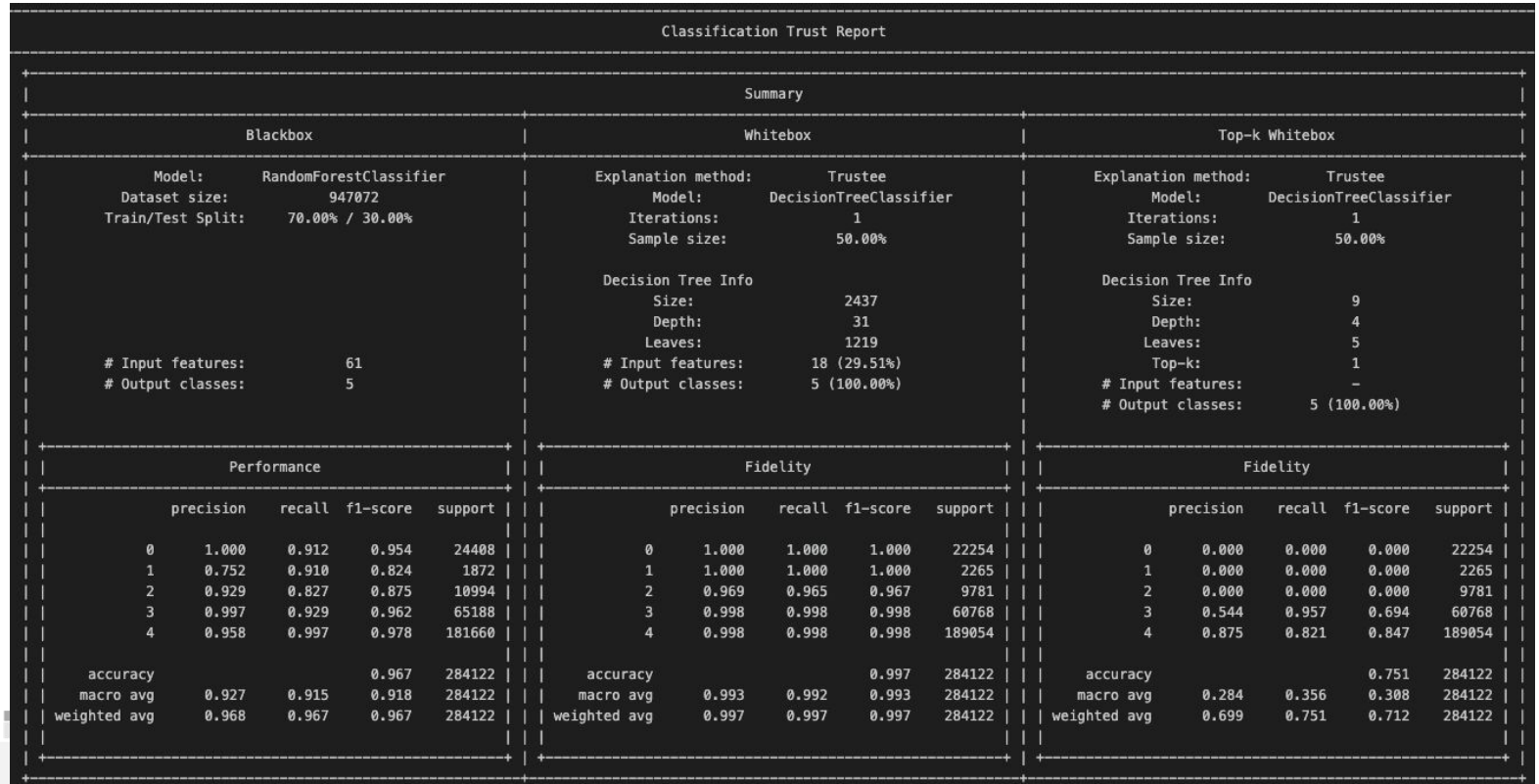
O.O.D. Samples

Model does not generalize!

Spurious Correlations

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
# 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

	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

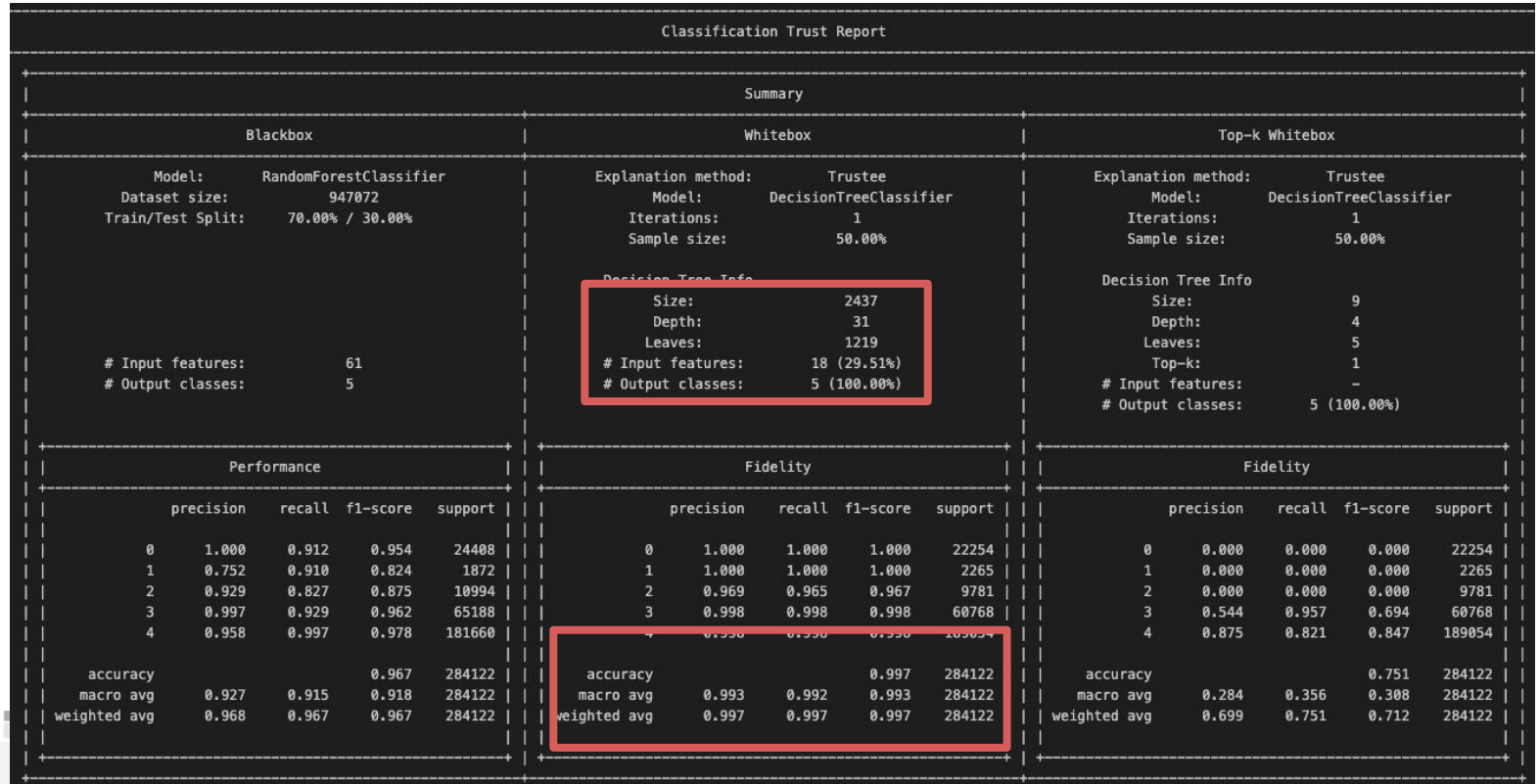
Fidelity

	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

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

	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

	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

	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

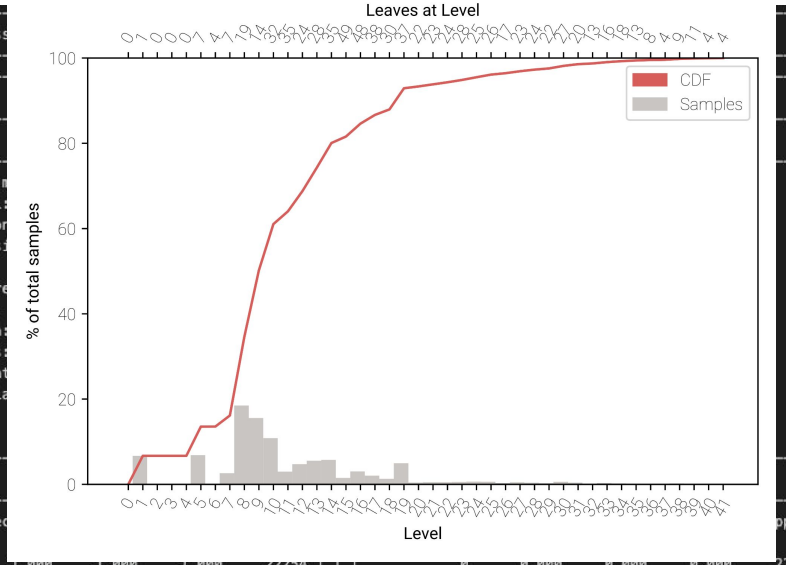
Generating Trust Reports

Blackbox

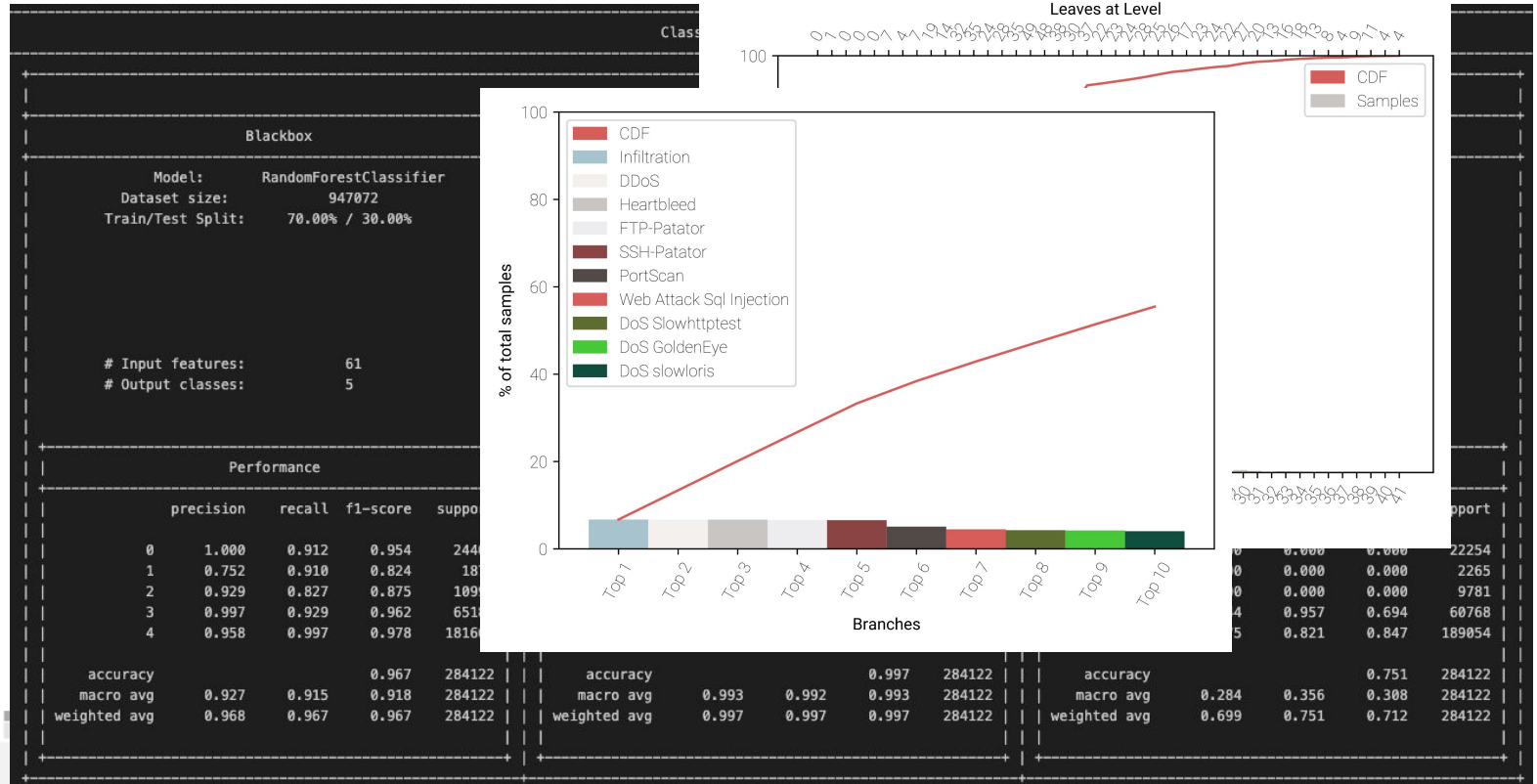
Model:	RandomForestClassifier
Dataset size:	947072
Train/Test Split:	70.00% / 30.00%
# Input features:	61
# Output classes:	5

Performance

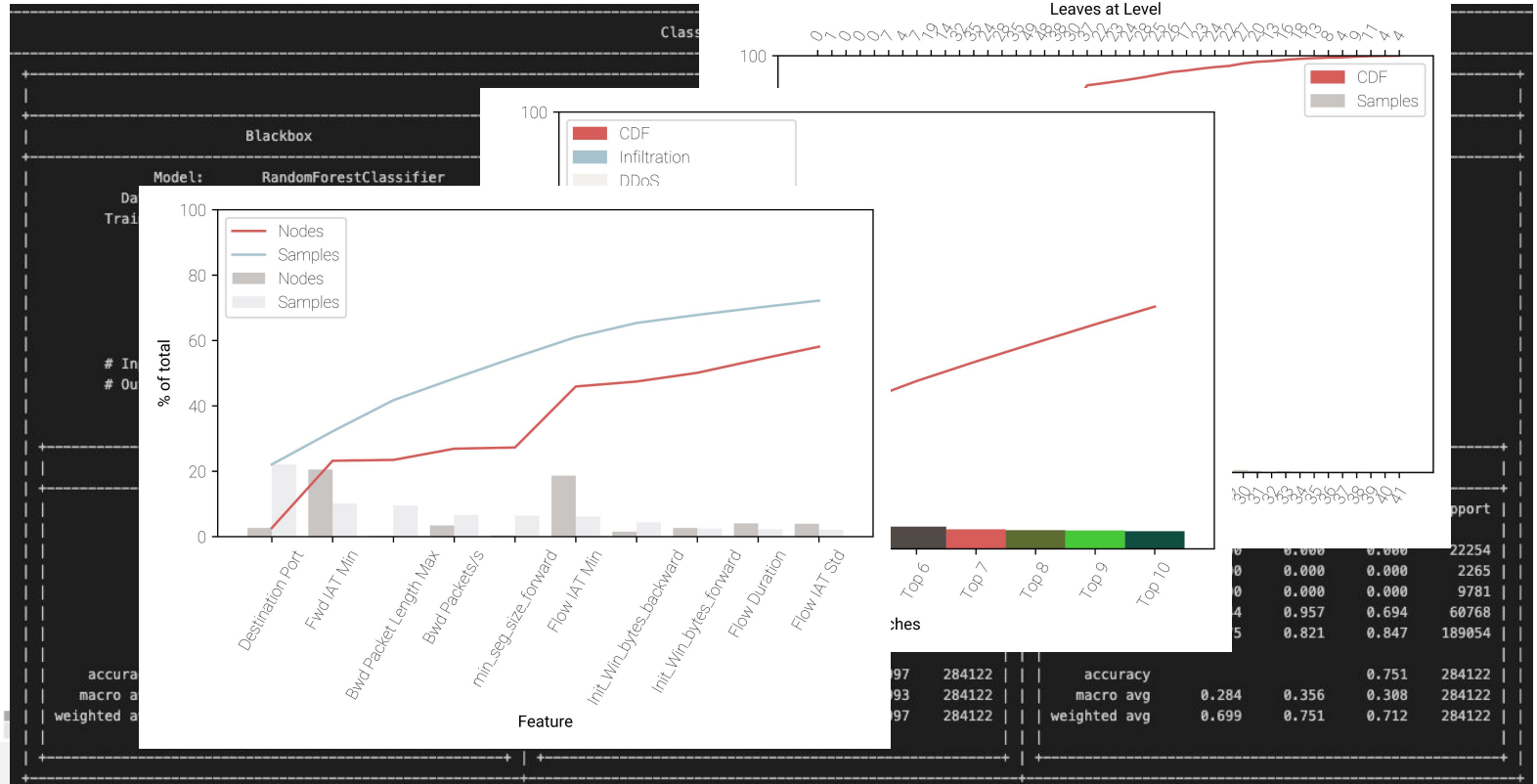
	precision	recall	f1-score	support
0	1.000	0.912	0.954	24408
1	0.752	0.910	0.824	1872
2	0.929	0.827	0.875	10994
3	0.997	0.929	0.962	65188
4	0.958	0.997	0.978	181660
accuracy			0.967	284122
macro avg	0.927	0.915	0.918	284122
weighted avg	0.968	0.967	0.967	284122



Generating Trust Reports



Generating Trust Reports



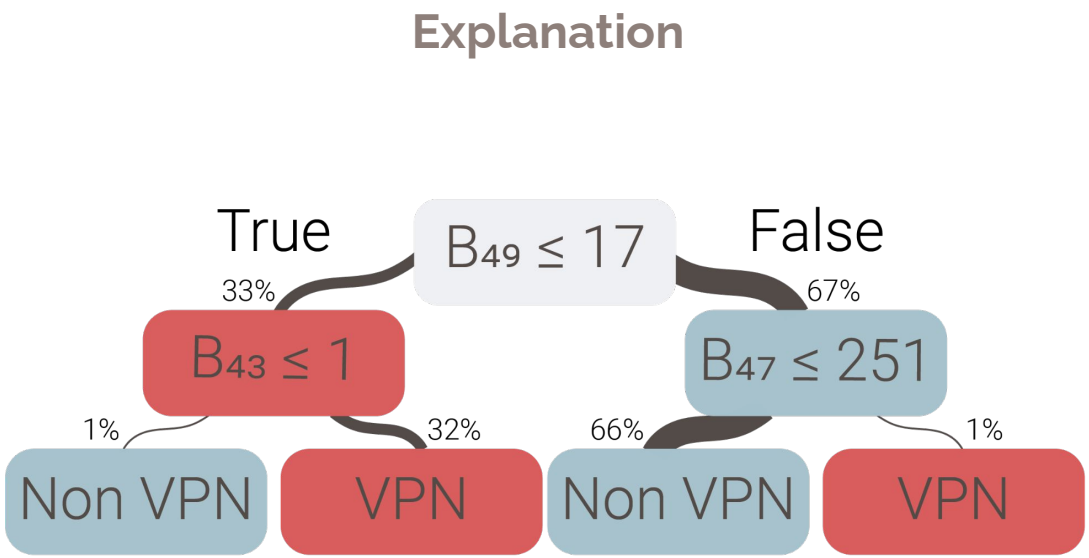
Use Cases

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

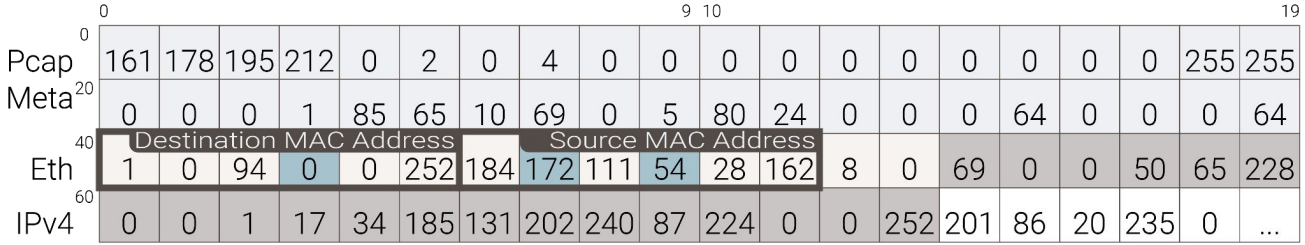


Fidelity: 1.000
No pruning
7 nodes

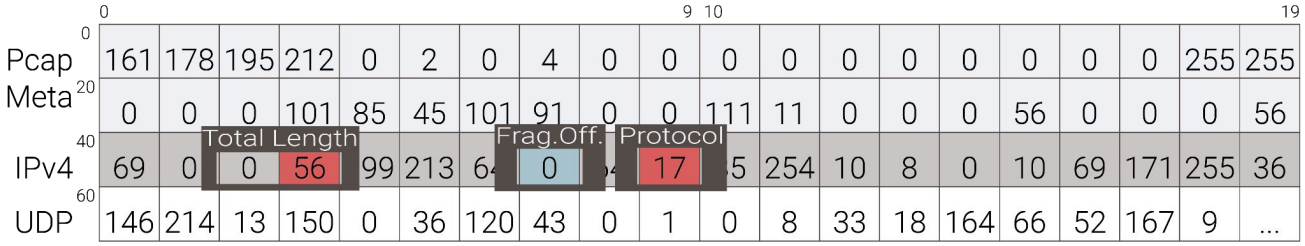
Use Case #1: Detecting VPN vs Non-VPN Traffic

Explanation

Non VPN



VPN

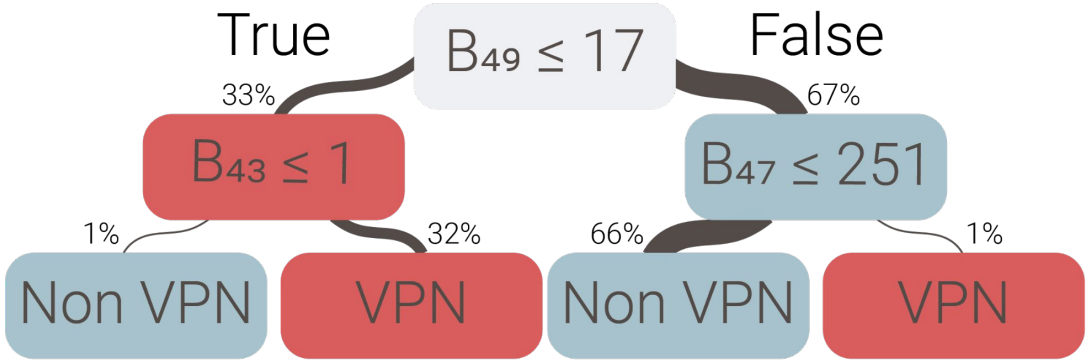


Use Case #1: Detecting VPN vs Non-VPN Traffic

Explanation

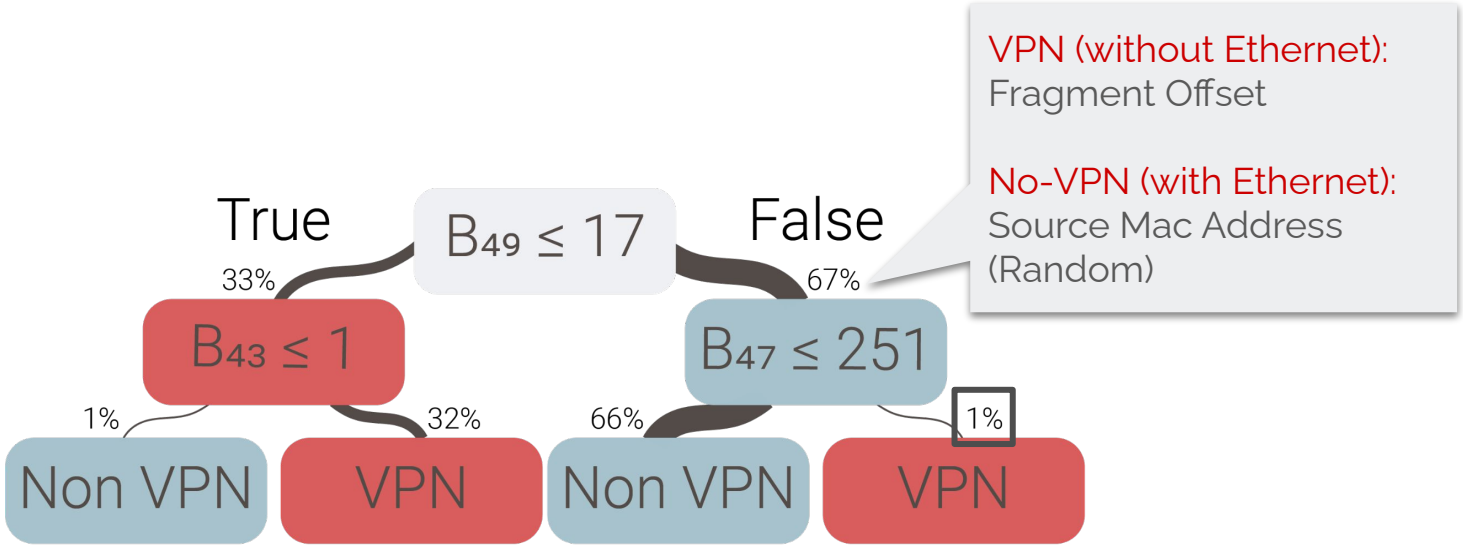
VPN (without Ethernet):
IPv4 Protocol (6 or 17)

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



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

Explanation

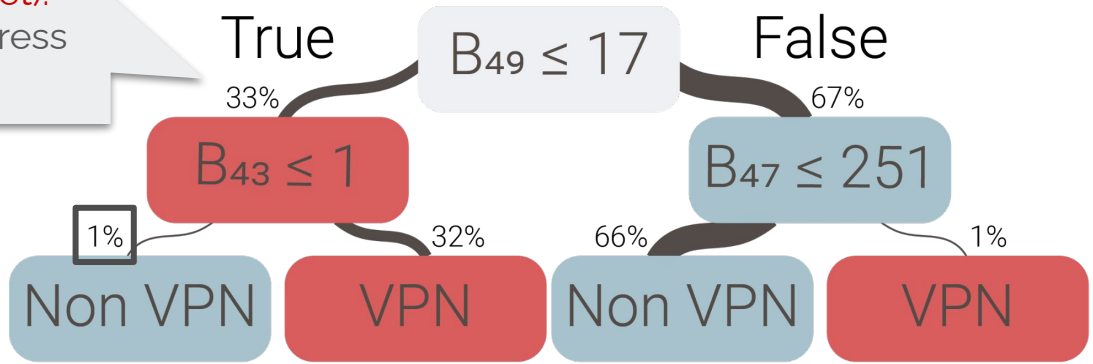


Use Case #1: Detecting VPN vs Non-VPN Traffic

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

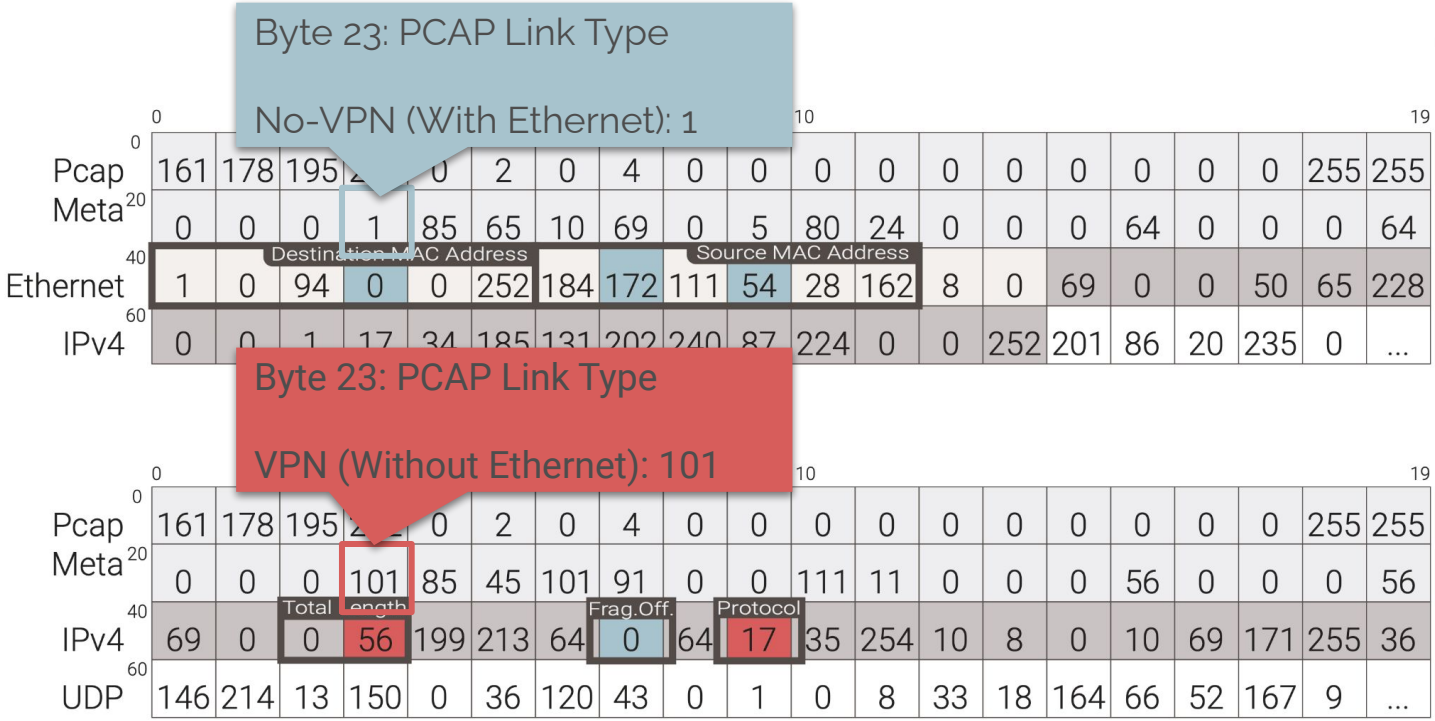
- Validation dataset:
  - Tampering with packet headers from original PCAPs

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**

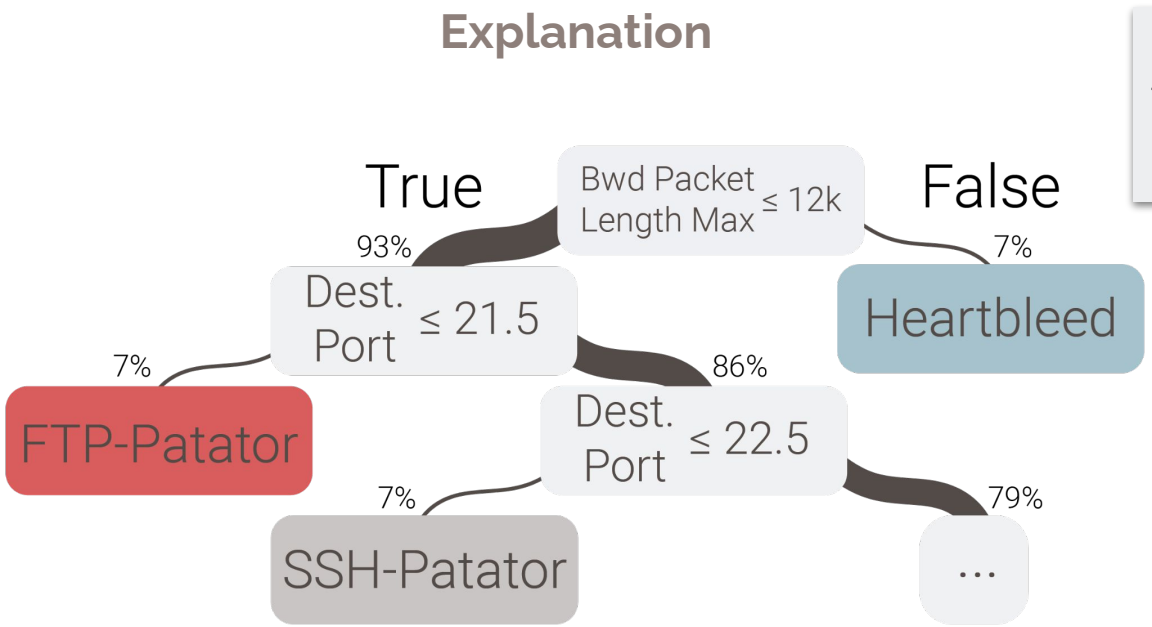
Takeaway: the model suffers from shortcut learning!

Use Case #2: Detecting Heartbleed Traffic

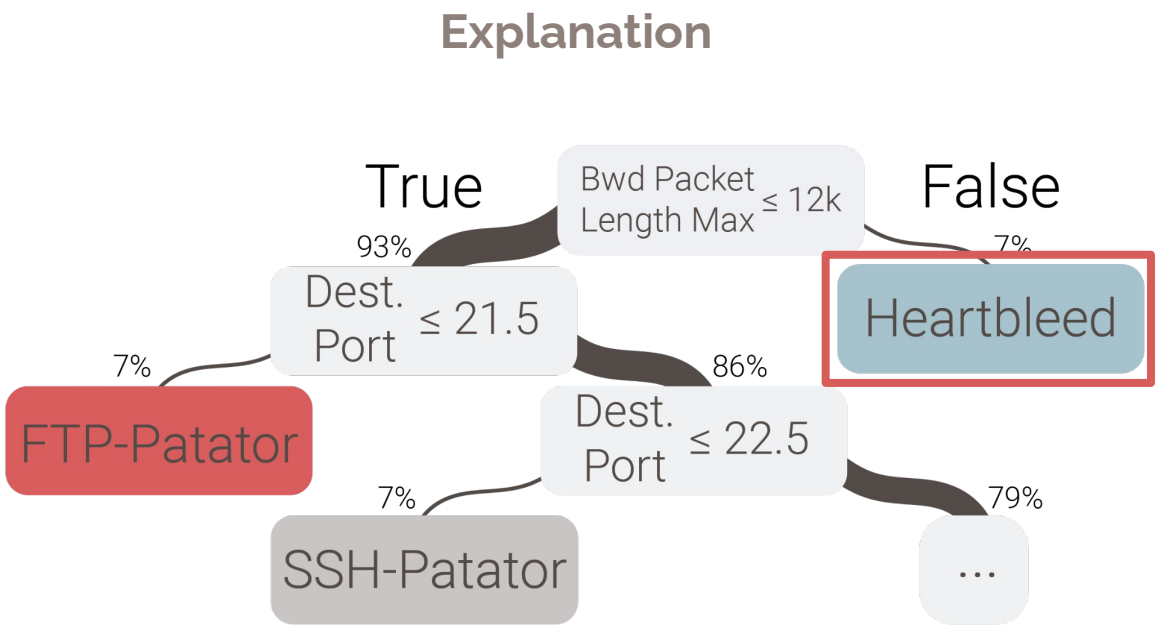
Problem Setup

- 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

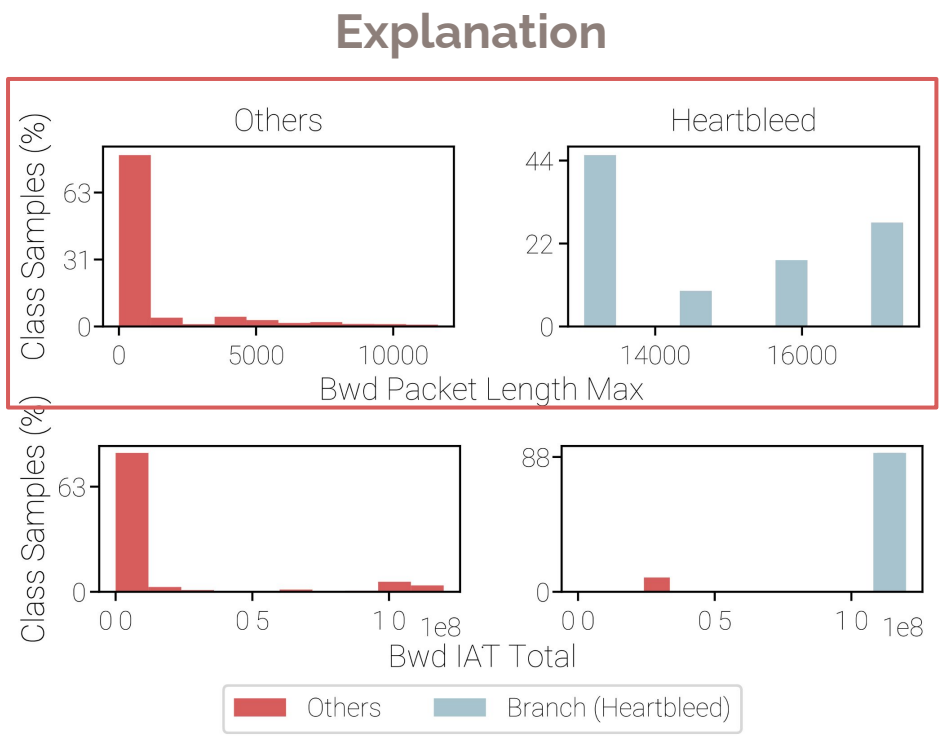
Use Case #2: Detecting Heartbleed Traffic



Use Case #2: Detecting Heartbleed Traffic

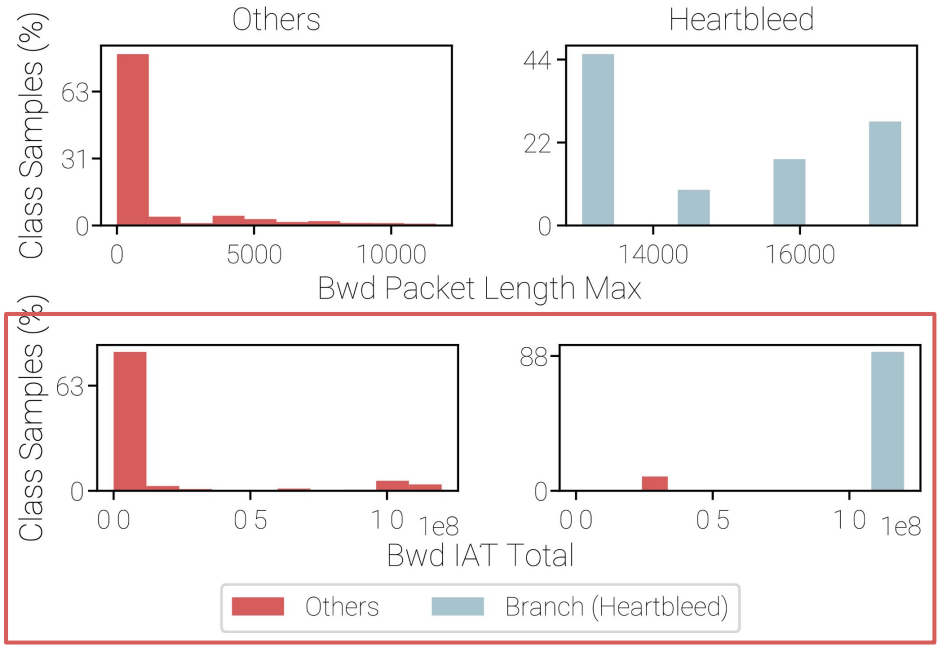


Use Case #2: Detecting Heartbleed Traffic



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

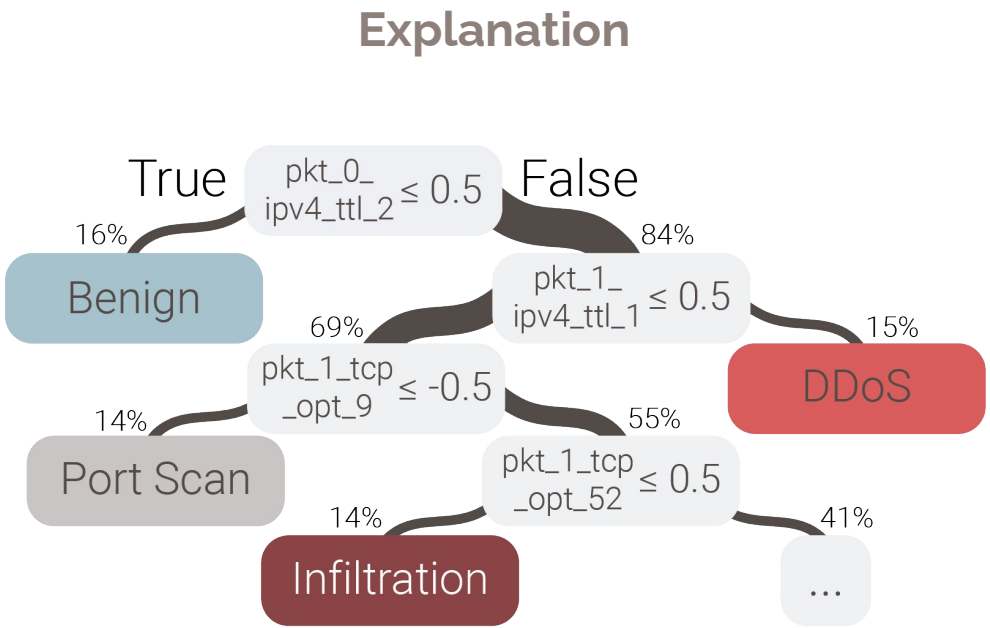
Takeaway: the model is overfitted to training data and fails to identify o.o.d. samples!

Use Case #3: Inferring Malicious Traffic for IDS

Problem Setup

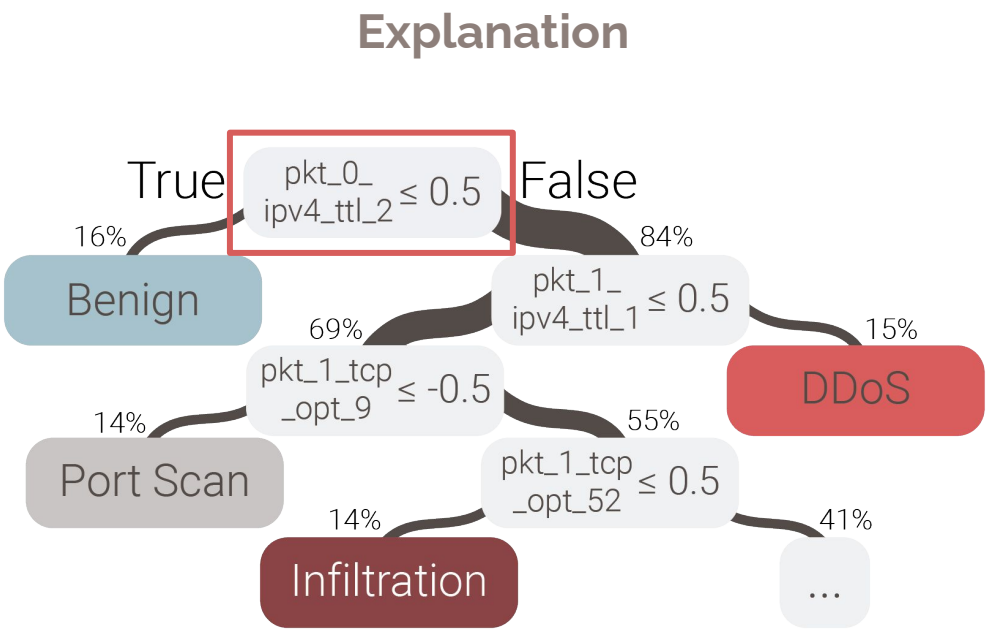
- Selected publications:
  - *"New Directions in Automated Traffic Analysis" — Holland et al., 2020*
- Proposal:
  - **Model**: nPrintML, an AutoML model for an Intrusion Detection System (IDS). It uses AutoGluton to ensemble multiple single models.
  - **Features**: 4,480 features with values -1, 0, 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



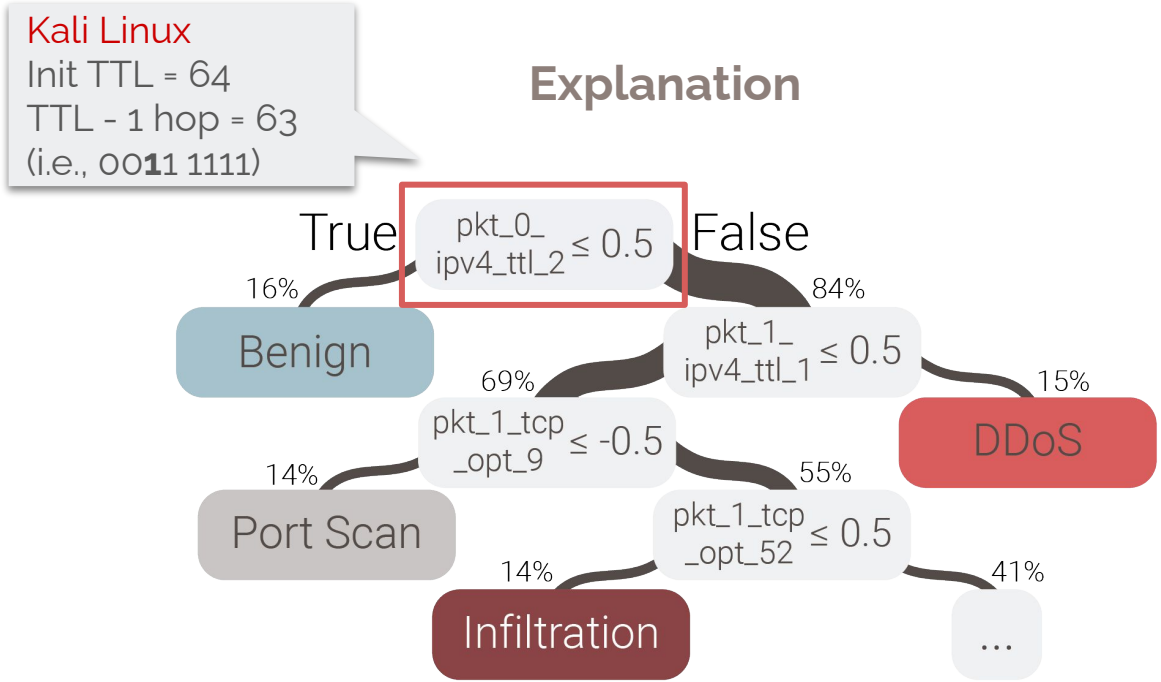
Fidelity: 0.99
Top-4 pruning
8 nodes

Use Case #3: Inferring Malicious Traffic for IDS

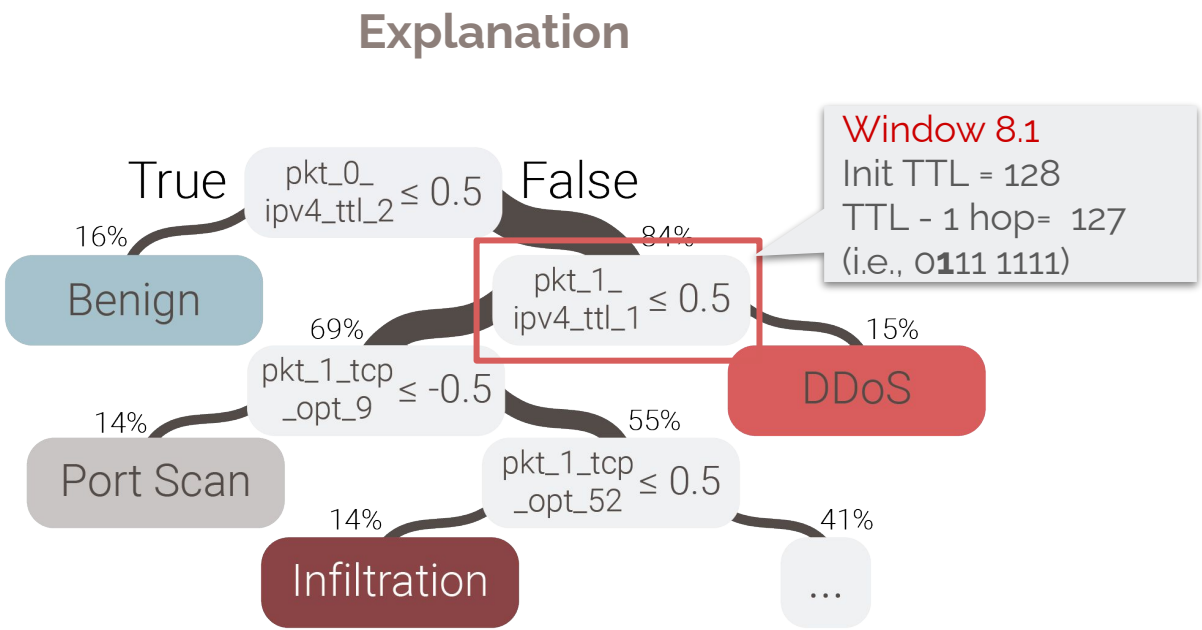


Use Case #3: Inferring Malicious Traffic for IDS

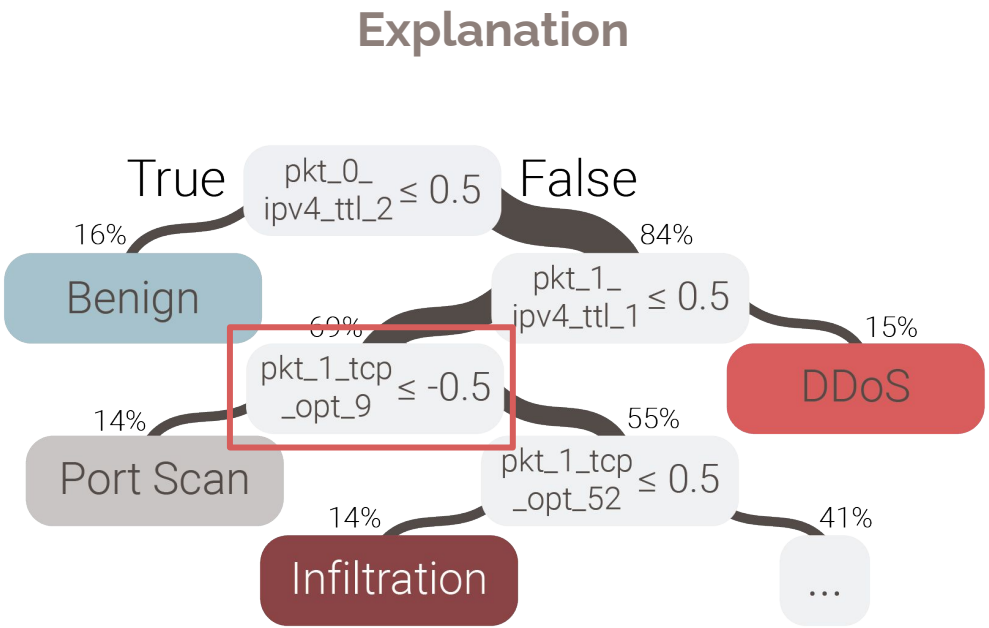
Kali Linux
Init TTL = 64
TTL - 1 hop = 63
(i.e., 00**11** 1111)



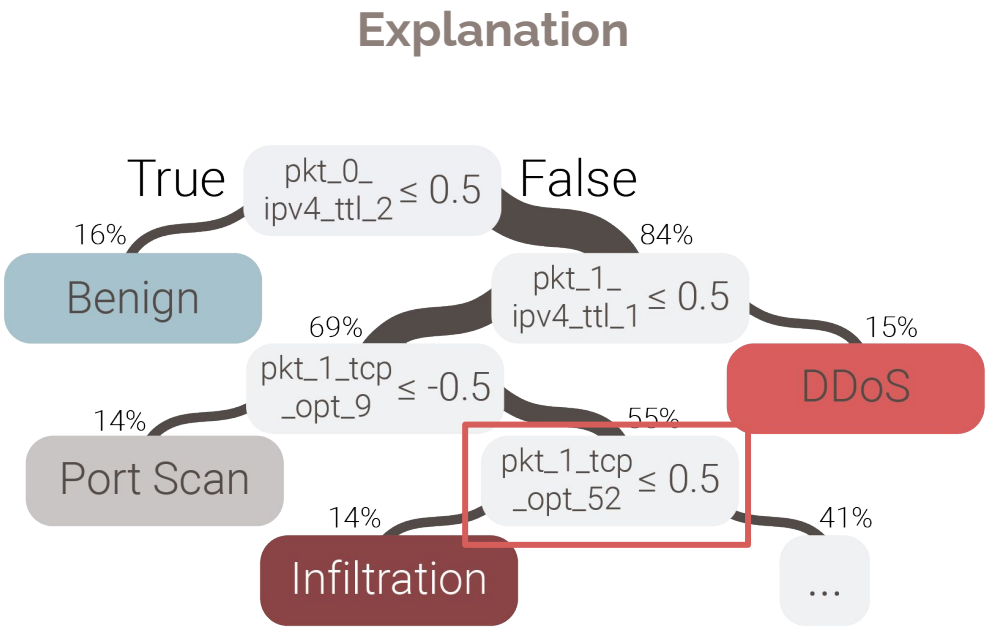
Use Case #3: Inferring Malicious Traffic for IDS



Use Case #3: Inferring Malicious Traffic for IDS



Use Case #3: Inferring Malicious Traffic for IDS



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!

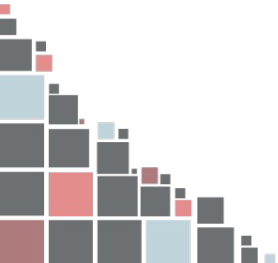
Other Use Cases

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

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!
    - More in the afternoon session

Thank you!



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

UFMS

.inf INSTITUTO DE INFORMATICA UFRGS

NIKSUN