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

Arthur Selle Jacobs¹

Roman Beltiukov⁴ Walter Willinger² Ronaldo A. Ferreira³

Arpit Gupta⁴ Lisandro Z. Granville¹

March 27th, 2023
The Rise of AI

FACIAL RECOGNITION
ALGORITHMS FOR MACHINE LEARNING
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

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

Collect Data

Select Model
How does it work?

Traditional AI/ML Development Pipeline

1. Collect Data
2. Select Model
3. Train
   - Model design and training
How does it work?

Traditional AI/ML Development Pipeline

Colllect Data

Model Evaluation

Train

Model design and training

Test

Evaluate model with test data

F1-Score
How does it work?

Traditional AI/ML Development Pipeline

Collect Data

Select Model

Model Evaluation

Train

Test

Model design and training

Evaluate model with test data

F1-Score
What about high-stakes decision making?

Why (and how) does the model work?

When does the model not work?

Self-driving Cars

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...

Data
CIC-IDS-2017 Dataset

Model
Random Forest Classifier

Model Evaluation

Train
Model design and training

Test
Evaluate model with test data

F1-Score
Consider this example...

**Model Evaluation**

Data
- CIC-IDS-2017 Dataset

Model
- Random Forest Classifier

Train
- Model design and training

Test
- Evaluate model with test data

F1-Score = 0.99
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
Augmented AI/ML Development Pipeline

Collect Data

Model Evaluation
- Train
  - Model design and training
- Test
  - Evaluate model with test data

Select Model

trustee
- Explain
  - High-fidelity & Low-complexity DT Extraction
- Analyze
  - Trust Report Generation
Augmented AI/ML Development Pipeline

Collect Data

Select Model

Model Evaluation

Train

Model design and training

Test

Evaluate model with test data

Explain

High-fidelity & Low-complexity DT Extraction

Analyze

Trust Report Generation
Explanation Requirements

#1 Model Agnostic

#2 High Fidelity

#3 Low Complexity

#4 Stable
Dataset

Black-box Model
Train Dataset

Test Dataset

70%

30%

Black-box Model
Train Dataset

70%

Test Dataset

30%

#1
Model
Agnostic

Black-box
Model

Expected Output
Train Dataset

Test Dataset

M samples

70%

30%

Expected Output

26
Iteration #1

Train Dataset 1

Test Dataset 1

M samples

Train Dataset

Test Dataset

Expected Output

70%

30%

27
Iteration #1

Train Dataset 1
Test Dataset 1

30%
70%

Explanation
Output

M samples

Train Dataset

Expected Output

70%
30%

Train Dataset 1
Test Dataset 1

Explanation Output

28
Iteration #1

Train Dataset 1

Test Dataset 1

30%

70%

M samples

Train Dataset

Test Dataset

Expected Output

Fidelity!

Explanation Output
Iteration #2

Train Dataset 1
Test Dataset 1

30%
70%

M samples

Expected Output

70%
30%

Train Dataset
Test Dataset
Iteration #N

Train Dataset

70%

M samples

Test Dataset

30%

Expected Output

70%

30%

Train Dataset 1

Test Dataset 1
Inner Loop #1...N

Iteration #N

Train Dataset 1

Test Dataset 1

Expected Output

M samples

Train Dataset

Test Dataset

70%

30%

70%

30%
#2
High Fidelity

Inner Loop
#1...N

DT with Best Fidelity
Size matters!

Inner Loop #1…N

DT with Best Fidelity
Inner Loop #1…N

Top-k Pruning

DT with Best Fidelity
Top-k Pruning

Fidelity

Samples

Fidelity

% of Samples

Top-k Branches

Top-k Branches
Top-k Pruning

Fidelity

Samples

Diminishing returns!
Top-k Pruning

Fidelity

Samples

Top-k Branches

Fidelity

% of Samples

Top-k Branches
Top-k Pruning

DT with Best Fidelity

Inner Loop
#1…N

#3
Low Complexity
Outer Loop
#1…S

Inner Loop
#1…N

Top-k Pruning
DT with Best Fidelity

DT with Highest Agreement
Outer Loop
#1…S

Inner Loop
#1…N

Top-k Pruning
DT with Best Fidelity

DT with Highest Agreement

#4 Stable
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

<table>
<thead>
<tr>
<th>Model: RandomForestClassifier</th>
<th>Explanation method: Trustee</th>
<th>Model: DecisionTreeClassifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset size: 947/872</td>
<td>iterations: 1</td>
<td>iterations: 1</td>
</tr>
<tr>
<td>Train/Test Split: 70.00% / 30.00%</td>
<td>sample size: 50.00%</td>
<td>sample size: 50.00%</td>
</tr>
<tr>
<td></td>
<td>Decision Tree Info</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>size: 2437</td>
<td></td>
</tr>
<tr>
<td></td>
<td>depth: 31</td>
<td></td>
</tr>
<tr>
<td></td>
<td>leaves: 1210</td>
<td></td>
</tr>
<tr>
<td></td>
<td># Input features: 38 (29.51%)</td>
<td># Input features: 5 (100.00%)</td>
</tr>
<tr>
<td></td>
<td># Output classes: 5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Performance

<table>
<thead>
<tr>
<th>precision</th>
<th>recall</th>
<th>F1-score</th>
<th>support</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.000</td>
<td>0.912</td>
<td>0.954</td>
</tr>
<tr>
<td>1</td>
<td>0.752</td>
<td>0.910</td>
<td>0.824</td>
</tr>
<tr>
<td>2</td>
<td>0.929</td>
<td>0.927</td>
<td>0.875</td>
</tr>
<tr>
<td>3</td>
<td>0.997</td>
<td>0.997</td>
<td>0.978</td>
</tr>
<tr>
<td>4</td>
<td>0.998</td>
<td>0.997</td>
<td>0.978</td>
</tr>
<tr>
<td>accuracy</td>
<td>0.967</td>
<td></td>
<td></td>
</tr>
<tr>
<td>macro avg</td>
<td>0.927</td>
<td></td>
<td></td>
</tr>
<tr>
<td>weighted avg</td>
<td>0.968</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Fidelity

<table>
<thead>
<tr>
<th>precision</th>
<th>recall</th>
<th>F1-score</th>
<th>support</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>1</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>2</td>
<td>0.969</td>
<td>0.965</td>
<td>0.967</td>
</tr>
<tr>
<td>3</td>
<td>0.996</td>
<td>0.998</td>
<td>0.998</td>
</tr>
<tr>
<td>4</td>
<td>0.998</td>
<td>0.998</td>
<td>0.998</td>
</tr>
<tr>
<td>accuracy</td>
<td>0.997</td>
<td></td>
<td></td>
</tr>
<tr>
<td>macro avg</td>
<td>0.993</td>
<td></td>
<td></td>
</tr>
<tr>
<td>weighted avg</td>
<td>0.997</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Generating Trust Reports

### Classification Trust Report

#### Summary

<table>
<thead>
<tr>
<th>Blackbox</th>
<th>Whitebox</th>
<th>Top-k Whitebox</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model:</td>
<td>Explanation method:</td>
<td>Exploration method:</td>
</tr>
<tr>
<td>RandomForestClassifier</td>
<td>Trustee</td>
<td>DecisionTreeClassifier</td>
</tr>
<tr>
<td>Dataset size: 947/772</td>
<td>Iterations: 1</td>
<td>Iterations: 1</td>
</tr>
<tr>
<td>Train/Test Split: 70.00% / 30.00%</td>
<td>Sample size: 50.00%</td>
<td>Sample size: 50.00%</td>
</tr>
</tbody>
</table>

#### Decision Tree Info

- Size: 2437
- Depth: 31
- Leaves: 1219

- # Input features: 61
- # Output classes: 5

#### Performance

<table>
<thead>
<tr>
<th></th>
<th>precision</th>
<th>recall</th>
<th>f1-score</th>
<th>support</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.000</td>
<td>0.912</td>
<td>0.954</td>
<td>24488</td>
</tr>
<tr>
<td>1</td>
<td>0.755</td>
<td>0.910</td>
<td>0.824</td>
<td>1872</td>
</tr>
<tr>
<td>2</td>
<td>0.929</td>
<td>0.827</td>
<td>0.875</td>
<td>18994</td>
</tr>
<tr>
<td>3</td>
<td>0.997</td>
<td>0.929</td>
<td>0.962</td>
<td>65988</td>
</tr>
<tr>
<td>4</td>
<td>0.958</td>
<td>0.997</td>
<td>0.978</td>
<td>181684</td>
</tr>
</tbody>
</table>

**accuracy** | 0.967 | 284122
**macro avg** | 0.927 | 0.915 | 0.918 | 284122
**weighted avg** | 0.968 | 0.967 | 0.967 | 284122

#### Fidelity

<table>
<thead>
<tr>
<th></th>
<th>precision</th>
<th>recall</th>
<th>f1-score</th>
<th>support</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>22254</td>
</tr>
<tr>
<td>1</td>
<td>1.000</td>
<td>0.989</td>
<td>0.997</td>
<td>2265</td>
</tr>
<tr>
<td>2</td>
<td>0.989</td>
<td>0.965</td>
<td>0.967</td>
<td>9781</td>
</tr>
<tr>
<td>3</td>
<td>0.998</td>
<td>0.998</td>
<td>0.998</td>
<td>68768</td>
</tr>
<tr>
<td>4</td>
<td>0.998</td>
<td>0.998</td>
<td>0.998</td>
<td>105924</td>
</tr>
</tbody>
</table>

**accuracy** | 0.997 | 284122
**macro avg** | 0.993 | 0.992 | 0.993 | 284122
**weighted avg** | 0.997 | 0.997 | 0.997 | 284122

#### Decision Tree Info

- Size: 9
- Depth: 4
- Leaves: 5

- Top-k: 1
- # Input features: 38 (29.51%)
- # Output classes: 5 (100.00%)
Generating Trust Reports

<table>
<thead>
<tr>
<th>Class</th>
<th>Train</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.000</td>
<td>0.912</td>
</tr>
<tr>
<td>1</td>
<td>0.752</td>
<td>0.910</td>
</tr>
<tr>
<td>2</td>
<td>0.929</td>
<td>0.827</td>
</tr>
<tr>
<td>3</td>
<td>0.997</td>
<td>0.929</td>
</tr>
<tr>
<td>4</td>
<td>0.958</td>
<td>0.997</td>
</tr>
</tbody>
</table>

Performance

<table>
<thead>
<tr>
<th>precision</th>
<th>recall</th>
<th>f1-score</th>
<th>support</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.000</td>
<td>0.912</td>
<td>24486</td>
</tr>
<tr>
<td>1</td>
<td>0.752</td>
<td>0.910</td>
<td>1872</td>
</tr>
<tr>
<td>2</td>
<td>0.929</td>
<td>0.827</td>
<td>18994</td>
</tr>
<tr>
<td>3</td>
<td>0.997</td>
<td>0.929</td>
<td>65185</td>
</tr>
<tr>
<td>4</td>
<td>0.958</td>
<td>0.997</td>
<td>181686</td>
</tr>
</tbody>
</table>

Leaves at Level

<table>
<thead>
<tr>
<th>Level</th>
<th>CDF</th>
<th>Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>1</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>2</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>3</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>4</td>
<td>0.751</td>
<td>0.751</td>
</tr>
</tbody>
</table>

Accuracy

<table>
<thead>
<tr>
<th>accuracy</th>
<th>macro avg</th>
<th>weighted avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.967</td>
<td>0.967</td>
<td>0.967</td>
</tr>
</tbody>
</table>
Generating Trust Reports

Blackbox

Model: RandomForestClassifier
Dataset size: 947872
Train/Test Split: 70.00% / 30.00%

# Input features: 61
# Output classes: 5

Performance

<table>
<thead>
<tr>
<th>precision</th>
<th>recall</th>
<th>F1-score</th>
<th>support</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.000</td>
<td>0.912</td>
<td>0.954</td>
</tr>
<tr>
<td>1</td>
<td>0.752</td>
<td>0.910</td>
<td>0.824</td>
</tr>
<tr>
<td>2</td>
<td>0.929</td>
<td>0.827</td>
<td>0.875</td>
</tr>
<tr>
<td>3</td>
<td>0.997</td>
<td>0.929</td>
<td>0.962</td>
</tr>
<tr>
<td>4</td>
<td>0.958</td>
<td>0.997</td>
<td>0.976</td>
</tr>
</tbody>
</table>

| accuracy  | 0.067  |
| macro avg | 0.927  | 0.915    | 0.918   | 284122  |
| weighted avg | 0.968  | 0.967    | 0.967   | 284122  |

Leaves at Level

- CDF
- Infiltration
- DDoS
- Heartbleed
- FTP-Patator
- SSH-Patator
- PortScan
- Web Attack Sql Injection
- DoS Slowhtttptest
- DoS GoldenEye
- DoS slowloris

Branches

| accuracy  | 0.993  | 0.992  | 0.993  | 284122  |
| macro avg | 0.356  | 0.356  | 0.356  | 284122  |
| weighted avg | 0.699  | 0.751  | 0.712  | 284122  |
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

- **B_{49} \leq 17**
  - True: 33% Non-VPN, 67% VPN
  - False: 1% Non-VPN, 99% VPN

- **B_{43} \leq 1**
  - Non-VPN: 1% Non-VPN, 99% VPN
  - VPN: 32% Non-VPN, 68% VPN

- **B_{47} \leq 251**
  - Non-VPN: 66% Non-VPN, 34% VPN
  - VPN: 1% Non-VPN, 99% VPN

Fidelity: 1.000
No pruning
7 nodes
# Use Case #1: Detecting VPN vs Non-VPN Traffic

**Explanation**

<table>
<thead>
<tr>
<th>PCap Meta</th>
<th>Non VPN</th>
<th>VPN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>161</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>178</td>
<td>10</td>
</tr>
<tr>
<td>0</td>
<td>195</td>
<td>54</td>
</tr>
<tr>
<td>4</td>
<td>212</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>85</td>
<td>172</td>
</tr>
<tr>
<td>0</td>
<td>65</td>
<td>111</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>54</td>
</tr>
<tr>
<td>0</td>
<td>69</td>
<td>28</td>
</tr>
<tr>
<td>0</td>
<td>80</td>
<td>162</td>
</tr>
<tr>
<td>0</td>
<td>24</td>
<td>8</td>
</tr>
<tr>
<td>0</td>
<td>64</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>0</td>
<td>64</td>
<td>228</td>
</tr>
</tbody>
</table>

**IPv4**

<table>
<thead>
<tr>
<th>Total Length</th>
<th>Frag Offset</th>
<th>Protocol</th>
<th>Non VPN</th>
<th>VPN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>69</td>
<td>0</td>
<td>56</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>99</td>
<td>64</td>
<td>17</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>213</td>
<td>0</td>
<td>254</td>
<td>10</td>
</tr>
<tr>
<td>0</td>
<td>6</td>
<td>0</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>69</td>
<td>171</td>
<td>255</td>
</tr>
<tr>
<td>0</td>
<td>33</td>
<td>18</td>
<td>164</td>
<td>66</td>
</tr>
<tr>
<td>0</td>
<td>52</td>
<td>167</td>
<td>9</td>
<td>...</td>
</tr>
</tbody>
</table>

**UDP**

| Non VPN | 146 | 214 | 13 | 150 | 0 | 36 | 120 | 43 | 0 | 1 | 0 | 8 | 33 | 18 | 164 | 66 | 52 | 167 | 9 | ... |

**Eth**

<table>
<thead>
<tr>
<th>Destination MAC Address</th>
<th>Source MAC Address</th>
<th>Non VPN</th>
<th>VPN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>94</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>252</td>
<td>184</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>172</td>
<td>111</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>54</td>
<td>28</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>69</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>50</td>
<td>65</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>228</td>
<td>64</td>
<td>0</td>
</tr>
</tbody>
</table>

**IPv4**

<table>
<thead>
<tr>
<th>Total Length</th>
<th>Frag Offset</th>
<th>Protocol</th>
<th>Non VPN</th>
<th>VPN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>69</td>
<td>0</td>
<td>56</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>99</td>
<td>64</td>
<td>17</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>213</td>
<td>0</td>
<td>254</td>
<td>10</td>
</tr>
<tr>
<td>0</td>
<td>6</td>
<td>0</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>69</td>
<td>171</td>
<td>255</td>
</tr>
<tr>
<td>0</td>
<td>33</td>
<td>18</td>
<td>164</td>
<td>66</td>
</tr>
<tr>
<td>0</td>
<td>52</td>
<td>167</td>
<td>9</td>
<td>...</td>
</tr>
</tbody>
</table>
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)

True: $B_{49} \leq 17$

- Non VPN: B_{43} \leq 1 (33%)
- VPN: B_{43} > 1 (1%)

False: $B_{47} \leq 251$

- Non VPN: B_{47} > 251 (66%)
- VPN: B_{47} > 251 (1%)
Use Case #1: Detecting VPN vs Non-VPN Traffic

**Explanation**

VPN (without Ethernet): Fragment Offset

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

- **B\(_{43}\) ≤ 1**
  - Non VPN: 33%
  - VPN: 1%

- **B\(_{49}\) ≤ 17**
  - True: 32%
  - False: 66%

- **B\(_{47}\) ≤ 251**
  - Non VPN: 67%
  - VPN: 1%
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)

Diagram:
- **VPN**:
  - **B43 ≤ 1**: 33%
  - **B49 ≤ 17**: 67%
- **Non-VPN**:
  - **B43 ≤ 1**: 1%
  - **B47 ≤ 251**: 66%
Use Case #1: Detecting VPN vs Non-VPN Traffic

Validation

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

<table>
<thead>
<tr>
<th>Validation Dataset</th>
<th>Avg. Precision</th>
<th>Avg. Recall</th>
<th>Avg. F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Untampered</td>
<td>0.959</td>
<td>0.956</td>
<td>0.955</td>
</tr>
<tr>
<td>Tampered-43-47-49</td>
<td>0.959</td>
<td>0.956</td>
<td>0.955</td>
</tr>
</tbody>
</table>
Use Case #1: Detecting VPN vs Non-VPN Traffic

No VPN

VPN

Byte 23: PCAP Link Type

No-VPN (With Ethernet): 1

VPN (Without Ethernet): 101
Use Case #1: Detecting VPN vs Non-VPN Traffic

Validation

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

<table>
<thead>
<tr>
<th>Validation Dataset</th>
<th>Avg. Precision</th>
<th>Avg. Recall</th>
<th>Avg. F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Untampered</td>
<td>0.959</td>
<td>0.956</td>
<td>0.955</td>
</tr>
<tr>
<td>Tampered-43-47-49</td>
<td>0.959</td>
<td>0.956</td>
<td>0.955</td>
</tr>
<tr>
<td>Tampered-32-to-63</td>
<td>0.889</td>
<td>0.867</td>
<td>0.856</td>
</tr>
<tr>
<td>Tampered-0-to-63</td>
<td>0.831</td>
<td>0.757</td>
<td>0.734</td>
</tr>
<tr>
<td>Tampered-0-to-127</td>
<td>0.753</td>
<td>0.555</td>
<td>0.398</td>
</tr>
</tbody>
</table>
Use Case #1: Detecting VPN vs Non-VPN Traffic

Validation

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

<table>
<thead>
<tr>
<th>Validation Dataset</th>
<th>Avg. Precision</th>
<th>Avg. Recall</th>
<th>Avg. F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Untampered</td>
<td>0.959</td>
<td>0.956</td>
<td>0.955</td>
</tr>
<tr>
<td>Tampered-43-47-49</td>
<td>0.959</td>
<td>0.956</td>
<td>0.955</td>
</tr>
<tr>
<td>Tampered-32-to-63</td>
<td>0.889</td>
<td>0.867</td>
<td>0.856</td>
</tr>
<tr>
<td>Tampered-0-to-63</td>
<td>0.831</td>
<td>0.757</td>
<td>0.734</td>
</tr>
<tr>
<td>Tampered-0-to-127</td>
<td>0.753</td>
<td>0.555</td>
<td><strong>0.398</strong></td>
</tr>
</tbody>
</table>

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)

- **Results:**
  - Reported F1-score: 0.99
  - Reproduced F1-score: 0.99
Use Case #2: Detecting Heartbleed Traffic

**Explanation**

True

- Dest. Port $\leq 21.5$
- Bwd Packet Length Max $\leq 12k$

False

- Dest. Port $\leq 22.5$
- Heartbleed

Fidelity: 0.99
Top-3 pruning
6 nodes
Use Case #2: Detecting Heartbleed Traffic

Explanation

True
- Dest. Port ≤ 21.5
- Bwd Packet Length Max ≤ 12k
- FTP-Patator
- SSH-Patator

False
- Dest. Port ≤ 22.5
- Heartbleed
- ...

93% 7%
86% 7%
79% 7%
Use Case #2: Detecting Heartbleed Traffic

Explanation

![Diagram showing distribution of Bwd Packet Length Max and Bwd IAT Total for Heartbleed and Others classes.](image-url)
Use Case #2: Detecting Heartbleed Traffic

Explanation

![Graphs showing Heartbleed traffic comparison](image)
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

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heartbleed (i.i.d.)</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>Heartbleed (o.o.d)</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Class</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heartbleed (i.i.d.)</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>Heartbleed (o.o.d)</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

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

Explanation

- **True**
  - pkt_0_ipv4_ttl_2 <= 0.5 (16%)
  - pkt_1_tcp_opt_9 <= -0.5 (14%)
  - Benign

- **False**
  - pkt_1_ipv4_ttl_1 <= 0.5 (84%)
  - pkt_1_tcp_opt_52 <= 0.5 (55%)
  - DDoS (15%)

- **Port Scan** (14%)
- **Infiltration** (41%)

Fidelity: 0.99
Top-4 pruning
8 nodes
Use Case #3: Inferring Malicious Traffic for IDS

**Explanation**

- **True**
  - **Benign**
    - `pkt_0.ipv4_ttl_2 ≤ 0.5`
    - 16%
  - **Port Scan**
    - `pkt_1_tcp_opt_9 ≤ -0.5`
    - 69%
    - 14%
  - **Infiltration**
    - `pkt_1_tcp_opt_52 ≤ 0.5`
    - 14%

- **False**
  - **DDoS**
    - `pkt_1_ipv4_ttl_1 ≤ 0.5`
    - 84%
    - 15%
  - **...**
    - 41%
Use Case #3: Inferring Malicious Traffic for IDS

**Explanation**

**Kali Linux**
Init TTL = 64
TTL - 1 hop = 63
(i.e., 0011 1111)

**True**

- \( pkt_0_{ipv4_ttl} \leq 0.5 \)
- \( pkt_1_{tcp_opt_9} \leq -0.5 \)

**False**

- \( pkt_1_{ipv4_ttl} \leq 0.5 \)
- \( pkt_1_{tcp_opt_52} \leq 0.5 \)

**Benign**
16%

**Port Scan**
14%

**Infiltration**
14%

**DDoS**
15%

...
Use Case #3: Inferring Malicious Traffic for IDS

Explanation

True

Benign

(pkt_0___ipv4_ttl_2 \leq 0.5) 16%

(pkt_1_tcp__opt_9 \leq -0.5) 14%

Port Scan

(pkt_0___ipv4_ttl_2 \leq 0.5) 16%

(pkt_1_tcp__opt_52 \leq 0.5) 14%

Infiltration

False

(pkt_1___ipv4_ttl_1 \leq 0.5) 69%

(pkt_1_tcp__opt_9 \leq -0.5) 55%

DDoS

Window 8.1

Init TTL = 128

TTL - 1 hop = 127

(i.e., 0111 1111)

...
Use Case #3: Inferring Malicious Traffic for IDS

Explanation

Benign

True

pkt_0_ipv4_ttl_2 ≤ 0.5

16%

False

pkt_1_ipv4_ttl_1 ≤ 0.5

84%

Port Scan

pkt_1_tcp_opt_9 ≤ -0.5

69%

Infiltration

pkt_1_tcp_opt_52 ≤ 0.5

14%

DDoS

pkt_1_tcp_opt_52 ≤ 0.5

15%

...
Use Case #3: Inferring Malicious Traffic for IDS

Explanation

- **Benign**
  - $\text{pkt}_0_{ \text{ipv4_ttl}_2} \leq 0.5$
  - $16\%$
  - $69\%$
  - Port Scan
  - $14\%$

- **DDoS**
  - $\text{pkt}_1_{ \text{ipv4_ttl}_1} \leq 0.5$
  - $84\%$
  - $55\%$
  - Infiltration
  - $41\%$

- **Infiltration**
  - $\text{pkt}_1_{ \text{tcp_opt}_52} \leq 0.5$
  - $14\%$

- **Port Scan**
  - $\text{pkt}_1_{ \text{tcp_opt}_9} \leq -0.5$
  - $15\%$

- **False**
  - $\text{pkt}_1_{ \text{ipv4_ttl}_1} \leq 0.5$
  - $84\%$
  - $55\%$
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

<table>
<thead>
<tr>
<th>Class</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benign</td>
<td>0.653</td>
<td>0.806</td>
<td>0.722</td>
</tr>
<tr>
<td>DoS</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Port Scan</td>
<td>0.120</td>
<td>0.143</td>
<td>0.130</td>
</tr>
<tr>
<td>Average</td>
<td>0.256</td>
<td>0.315</td>
<td>0.282</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Class</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benign</td>
<td>0.653</td>
<td>0.806</td>
<td>0.722</td>
</tr>
<tr>
<td>DoS</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Port Scan</td>
<td>0.120</td>
<td>0.143</td>
<td>0.130</td>
</tr>
<tr>
<td>Average</td>
<td>0.256</td>
<td>0.315</td>
<td>0.282</td>
</tr>
</tbody>
</table>

Takeaway: the model suffers from spurious correlations in the training data!
<table>
<thead>
<tr>
<th>Problem</th>
<th>Model(s)</th>
<th>Dataset(s)</th>
<th>Trustee Fidelity</th>
<th>Inductive Bias</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detect VPN traffic (Wang et al., ISI'17)</td>
<td>1-D CNN</td>
<td>ISCX VPN-nonVPN</td>
<td>1.00</td>
<td>Shortcut learning</td>
</tr>
<tr>
<td>Detect Heartbleed traffic (Sharafaldin et al., ICISSP'18)</td>
<td>RFC</td>
<td>CIC-IDS-2017</td>
<td>0.99</td>
<td>O.O.D.</td>
</tr>
<tr>
<td>Detect Malicious traffic (IDS) (Holland et al., CCS'21)</td>
<td>nPrintML</td>
<td>CIC-IDS-2017</td>
<td>0.99</td>
<td>Spurious Correlation</td>
</tr>
<tr>
<td>Anomaly Detection (Mirsky et al., NDSS'18)</td>
<td>Kitsune</td>
<td>Mirai dataset</td>
<td>0.99</td>
<td>O.O.D</td>
</tr>
<tr>
<td>OS Fingerprinting (Holland et al., CCS'21)</td>
<td>nPrintML</td>
<td>CIC-IDS-2017</td>
<td>0.99</td>
<td>O.O.D</td>
</tr>
<tr>
<td>IoT Device Fingerprinting (Xiong et al., HotNets'19)</td>
<td>lisy</td>
<td>UNSW-IoT</td>
<td>0.99</td>
<td>Shortcut learning</td>
</tr>
<tr>
<td>Adaptive Bit-rate (Mao et al., SIGCOMM'17)</td>
<td>Pensieve</td>
<td>HSDPA Norway</td>
<td>0.99</td>
<td>O.O.D</td>
</tr>
<tr>
<td>Problem</td>
<td>Model(s)</td>
<td>Dataset(s)</td>
<td>Trustee Fidelity</td>
<td>Inductive Bias</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>------------</td>
<td>------------------</td>
<td>------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Detect VPN traffic (Wang et al., ISI’17)</td>
<td>1-D CNN</td>
<td>ISCX VPN-nonVPN</td>
<td>1.00</td>
<td>Shortcut learning</td>
</tr>
<tr>
<td>Detect Heartbleed traffic (Sharafaldin et al., ICISSP’18)</td>
<td>RFC</td>
<td>CIC-IDS-2017</td>
<td>0.99</td>
<td>O.O.D.</td>
</tr>
<tr>
<td>Detect Malicious traffic (IDS) (Holland et al., CCS’21)</td>
<td>nPrintML</td>
<td>CIC-IDS-2017</td>
<td>0.99</td>
<td>Spurious Correlation</td>
</tr>
<tr>
<td>Anomaly Detection (Mirsky et al., NDSS’18)</td>
<td>Kitsune</td>
<td>Mirai dataset</td>
<td>0.99</td>
<td>O.O.D</td>
</tr>
<tr>
<td>OS Fingerprinting (Holland et al., CCS’21)</td>
<td>nPrintML</td>
<td>CIC-IDS-2017</td>
<td>0.99</td>
<td>O.O.D</td>
</tr>
<tr>
<td>IoT Device Fingerprinting (Xiong et al., HotNets’19)</td>
<td>lisy</td>
<td>UNSW-IoT</td>
<td>0.99</td>
<td>Shortcut learning</td>
</tr>
<tr>
<td>Adaptive Bit-rate (Mao et al., SIGCOMM’17)</td>
<td>Pensieve</td>
<td>HSDPA Norway</td>
<td>0.99</td>
<td>O.O.D</td>
</tr>
<tr>
<td>Problem</td>
<td>Model(s)</td>
<td>Dataset(s)</td>
<td>Trustee Fidelity</td>
<td>Inductive Bias</td>
</tr>
<tr>
<td>--------------------------------------------------------------</td>
<td>-----------</td>
<td>---------------------</td>
<td>------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>Detect VPN traffic (Wang et al., ISI’17)</td>
<td>1-D CNN</td>
<td>ISCX VPN-nonVPN</td>
<td>1.00</td>
<td>Shortcut learning</td>
</tr>
<tr>
<td>Detect Heartbleed traffic (Sharafaldin et al., ICISSP’18)</td>
<td>RFC</td>
<td>CIC-IDS-2017</td>
<td>0.99</td>
<td>O.O.D.</td>
</tr>
<tr>
<td>Detect Malicious traffic (IDS) (Holland et al., CCS’21)</td>
<td>nPrintML</td>
<td>CIC-IDS-2017</td>
<td>0.99</td>
<td>Spurious Correlation</td>
</tr>
<tr>
<td>Anomaly Detection (Mirsky et al., NDSS’18)</td>
<td>Kitsune</td>
<td>Mirai dataset</td>
<td>0.99</td>
<td>O.O.D</td>
</tr>
<tr>
<td>OS Fingerprinting (Holland et al., CCS’21)</td>
<td>nPrintML</td>
<td>CIC-IDS-2017</td>
<td>0.99</td>
<td>O.O.D</td>
</tr>
<tr>
<td>IoT Device Fingerprinting (Xiong et al., HotNets’19)</td>
<td>lisy</td>
<td>UNSW-IoT</td>
<td>0.99</td>
<td>Shortcut learning</td>
</tr>
<tr>
<td>Adaptive Bit-rate (Mao et al., SIGCOMM’17)</td>
<td>Pensieve</td>
<td>HSDPA Norway</td>
<td>0.99</td>
<td>O.O.D</td>
</tr>
</tbody>
</table>
Algorithmic Description of Trustee

Algorithm 1 Model agnostic decision tree explanation extraction.

1: procedure TRUSTEE(
2: \( \pi^* \): Black-box model,
3: \( \mathcal{D}_0 \): Initial training dataset,
4: \( M \): Number of samples to train the decision tree,
5: \( N \): Number of iterations of inner loop,
6: \( S \): Number of iterations of outer loop,
7: \( k \): Parameter for Top-\( k \) Pruning),
8: \( \mathcal{D} \leftarrow \pi^* (\forall x \in \mathcal{D}_0) \)
9: Initialize stabilization set of DTs \( \mathcal{R} \leftarrow \emptyset \)
10: for \( i \leftarrow 1 \ldots S \) do
11:     for \( j \leftarrow 1 \ldots N \) do
12:         Sample \( M \) training cases uniformly from \( \mathcal{D} \)
13:         \( \mathcal{D}' \leftarrow \{(x, y) \mid U(\mathcal{D})\} \)
14:     Split sampled dataset for training and testing
15:     \( \mathcal{D}_{train}', \mathcal{D}_{test}' \leftarrow \text{TRAINTESTSPLIT}(\mathcal{D}') \)
16:     Train DT
17:     \( \hat{\pi}_j \leftarrow \text{TRAINDECISIONTREE}(\mathcal{D}_{train}') \)
18:     Test and get samples DT misclassifies
19:     \( \mathcal{D}'_c \leftarrow \{(x, y) \in \mathcal{D}_{test}' \mid \hat{\pi}_j (x) \neq \pi^* (x)\} \)
20:     Get correct outcome from black-box
21:     \( \mathcal{D}_c \leftarrow U(\mathcal{D}') \)
Algorithmic Description of Trustee

Ablation Study on Design Requirements

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

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

Project description

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.

Overview

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

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

View sample code examples:
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 Python package
- https://pypi.org/project/trustee/

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

Use Cases Repository
- https://github.com/TrusteeML/emperor
### Existing approaches

<table>
<thead>
<tr>
<th>Method</th>
<th>Model Agnostic</th>
<th>High Fidelity</th>
<th>Domain-specific Pruning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trepan</td>
<td>✓</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td><em>dtextract</em></td>
<td>✓</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>VIPER</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Metis</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>trustee</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
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

True
- Mac-IP 1-Weight ≤ 139
- RMSE: 0.018
- 45%

False
- Mac-IP 0.01-Weight ≤ 2.7k
- RMSE: 12.49
- 51%

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!