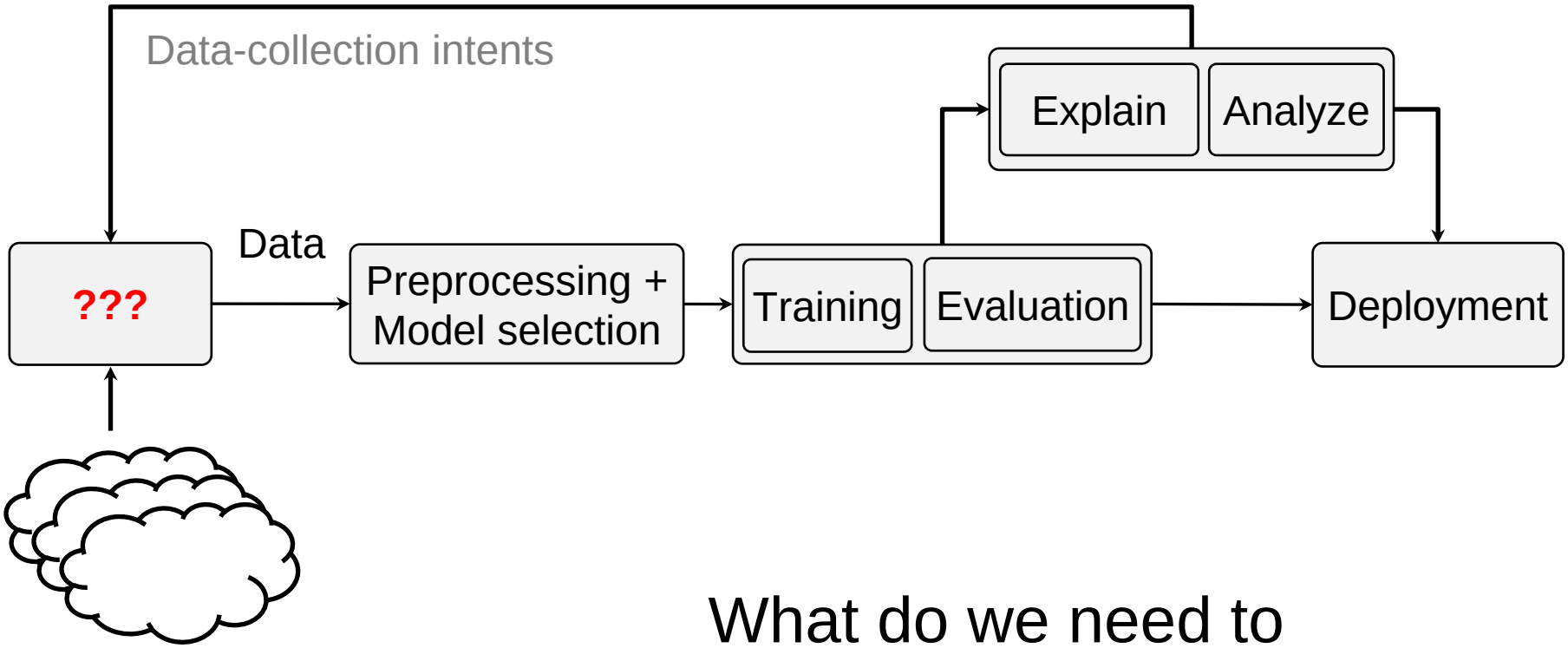
Collecting Data for “Beyond F1 Scores”?



Platforms &
Data Generators

What do we need to
make the data better?

Existing Data Collection Efforts

Key (**problematic**) attributes

- **Fragmented**

  Designed for specific learning problems and network environments

- **Monolithic**

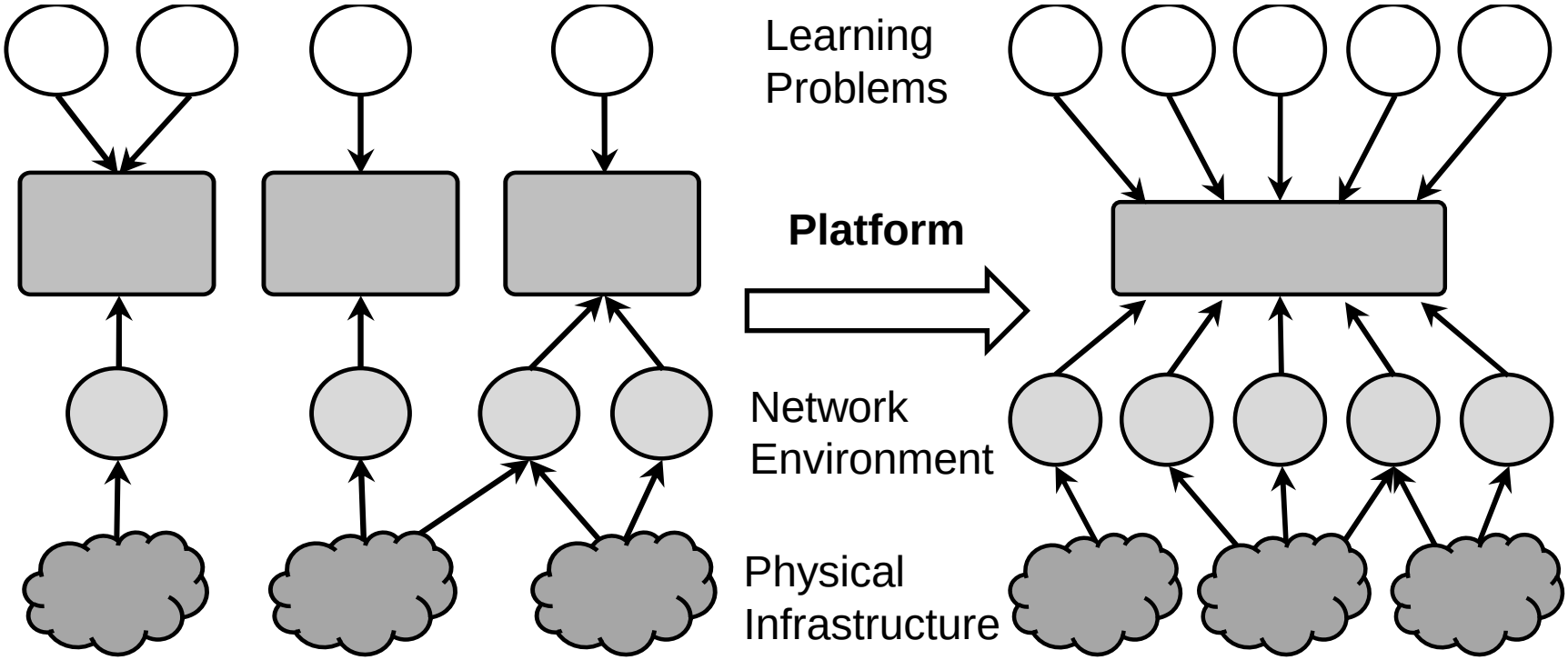
  Lack modularity, no clear separation between intents and mechanisms

- **One-off**

  Not suited for iterative data collection

A New Data-Collection Platform – netUnicorn

- Fragmented → **Unified (any problem & any platform)**
- Monolithic → **Modular (Tasks, Pipelines, Experiments)**
- One-off → **Iterative (Easy reproducibility)**



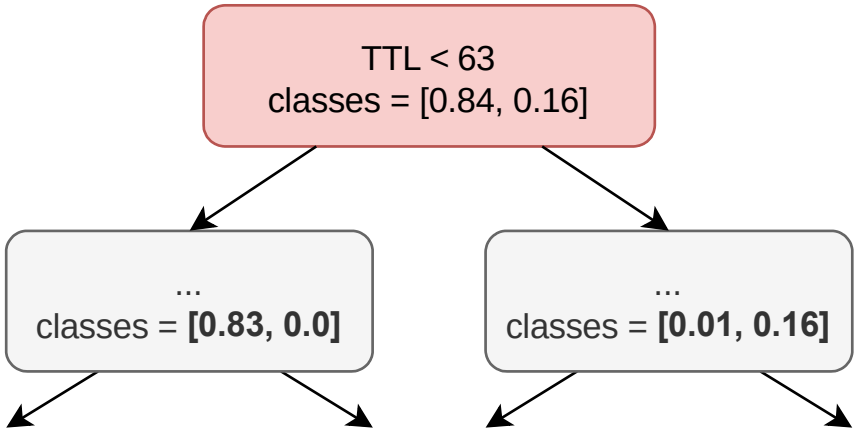
Key Disaggregations

- **Stakeholders**
  - Experimenters (intents) vs. Developers (mechanisms)
- **Infrastructures**
  - Connectors: easy modular way to add infrastructures
  - Nodes & NodePools: abstract way to represent targets
- **Programming Abstractions**
  - Developers:
    - Tasks, TaskDispatchers, Pipelines
  - Experimenters:
    - Nodes, Experiments
- **netUnicorn**
  - Users
  - Core Services
  - Executors

Illustrative Example – HTTP Bruteforce

- Learning problem:
  DDoS flows identification from network traffic (PCAPs)

Data: CIC-IDS-2017 Dataset
Model: Random Forest
F1-score: 0.99



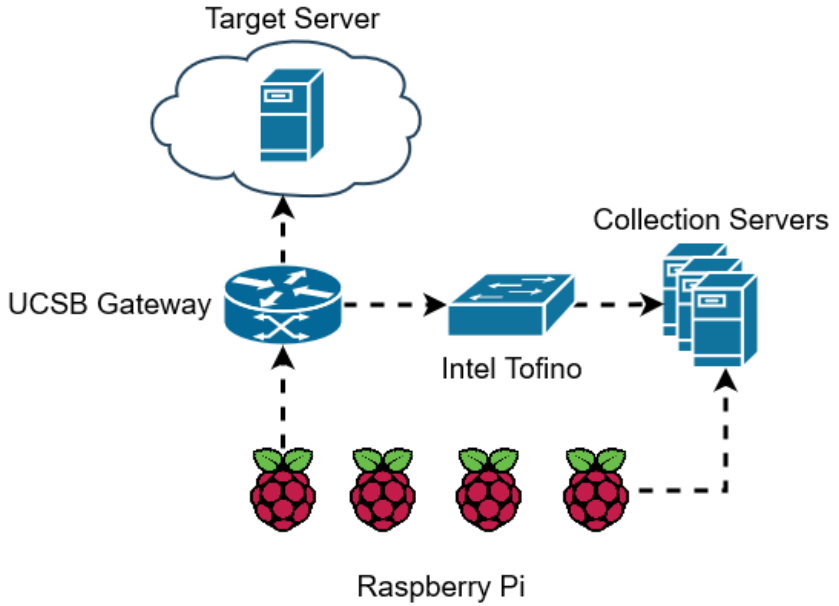
Observations

- All nodes sending benign traffic have the same TTL
- Model learns a shortcut – **generalization issues**

HTTP Bruteforce – implementation efforts

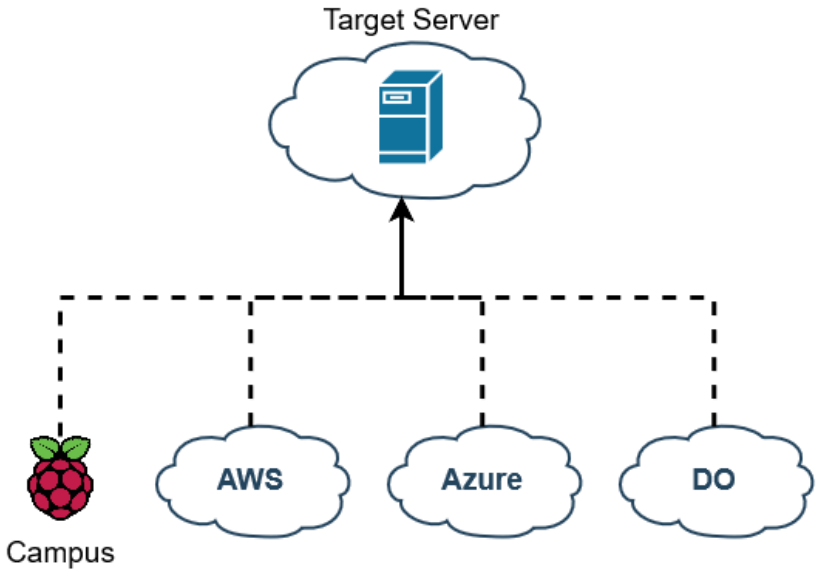
PINOT infrastructure

- Raspberry Pi devices over the whole campus
- Traffic mirroring @ border gateway
- 80 lines of code in total



Multi-cloud infrastructure

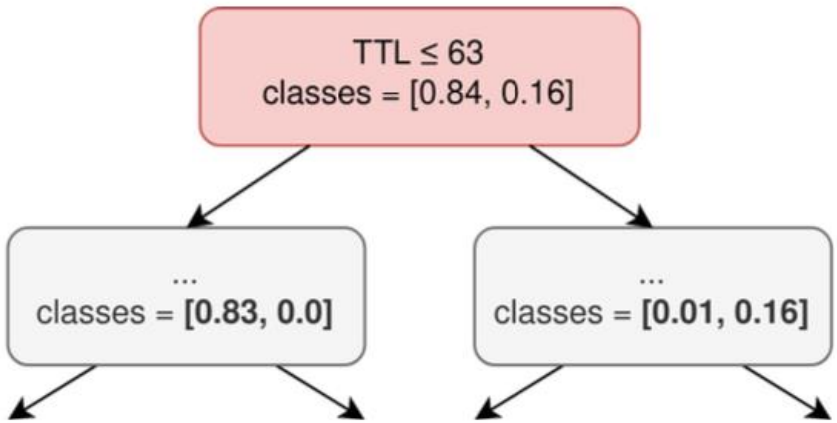
- VMs and containers in different clouds
- Extra 5 (!) lines of code

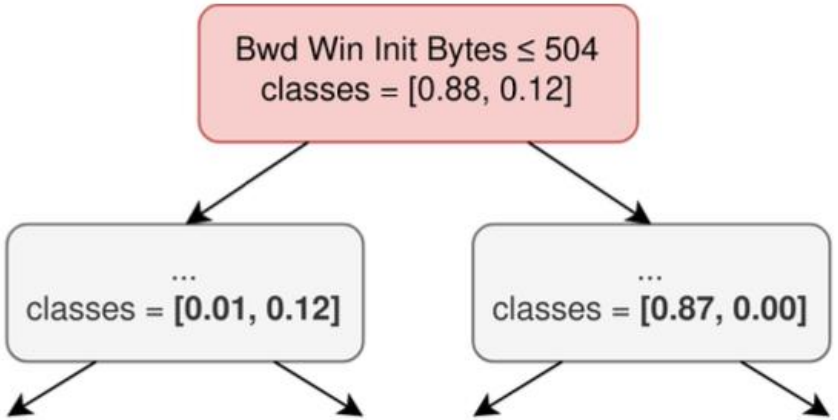


Fixing the dataset – step 1

- "Swapping" of nodes pools to remove TTL issue
  - + adding multiple clouds
  - 10 lines of code

- Recollected the data, explored with Trustee
  - New shortcut: Bwd Win Init Bytes
  - *Backward TCP Window – how many bytes server can accept (indicator of heavy server load)*

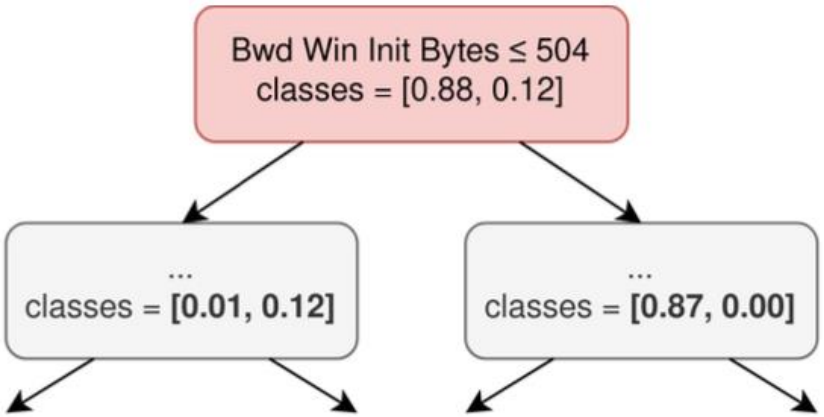


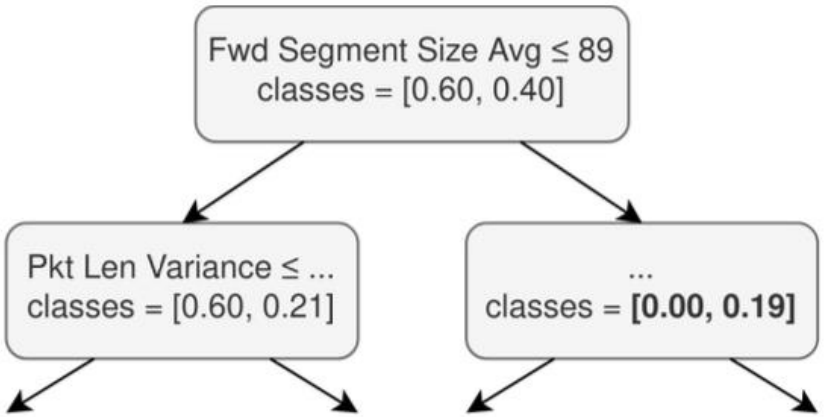


Fixing the dataset – step 2

- Introduced more benign traffic & slower bruteforce
  - +5 lines of code (w/o tasks implementation)

- Recollected the data, explored with Trustee
  - *Starts using valid features (small forward packet size & small packet length variance)*
  - *Shortcuts are not found (but possible)*





Results & Other examples

- From *simple static data collection* via **iterations** to **better datasets** and **generalizable models**
- Low efforts to implement iterations and usage of multiple infrastructures

- Continuation examples:
  - Explore differences in bruteforce data between different infrastructures
  - Explore the resulting dataset more to verify lack of problems *(and possibly iterate more)*

netUnicorn's iterative data collection helps developing ML models with better chance of being **generalizable**

Takeaways

- Data collection efforts should be:
  - Iteratively built to eliminate biases
  - Open, easy to reproduce, share, and implement
  - Adaptable to different infrastructures

- netUnicorn – modular platform for data collection
  - Wide range of learning problems
    - Speedtests, YouTube/Vimeo/Twitch QoE, Wi-Fi measurements, video identification, network attacks identifications, …
  - Wide range of supported infrastructures
    - PINOT, Mininet, AWS, MS Azure, Kubernetes, SaltStack, SSH, …

rbeltiukov@ucsb.edu https://netunicorn.cs.ucsb.edu