# *AI4NETS*
## *AI/ML for Data Communication Networks*

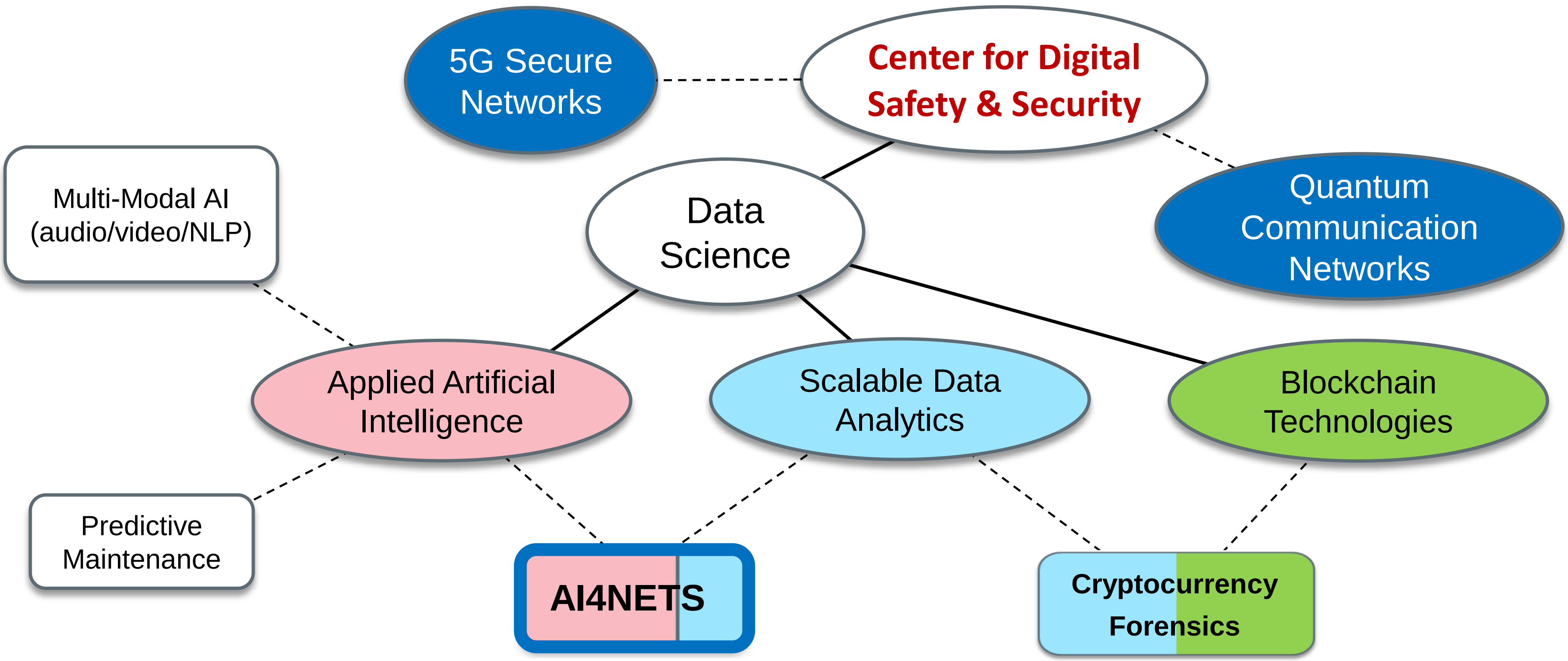

Dr. Pedro Casas

Senior Scientist

Data Science & Artificial Intelligence

AIT Austrian Institute of Technology @Vienna

# AI4NETS @AIT – Data Science Research




- AIT – **Austrian Institute of Technology** (~1300 FTEs in multiple applied research domains)
- DSS – Center for Digital Safety & Security (~300 FTEs)
- **Digital Insight Lab** (~25 FTEs), working on Data Science as core Research Field

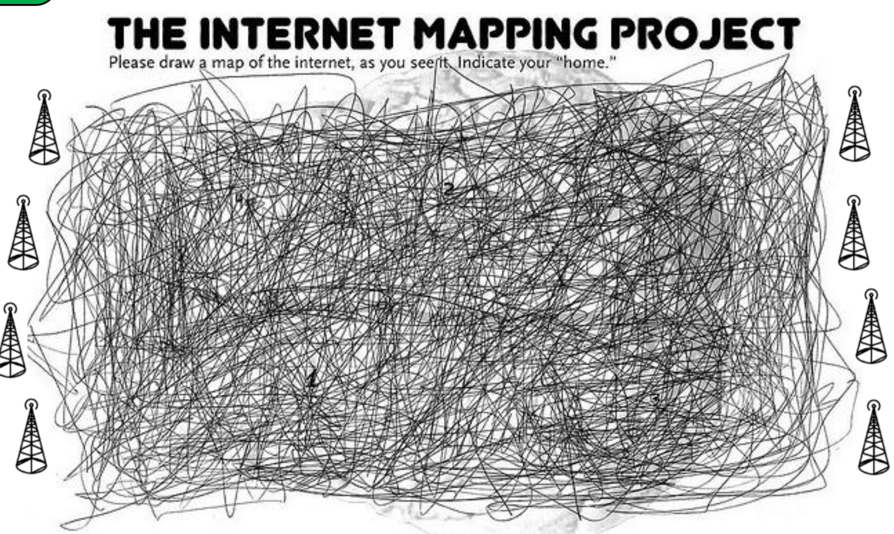

While AI has produced major breakthroughs in current data driven landscape...
…its successful application to data communication networks is still at a very early stage

# AI4NETS, Big Data Analytics, and Network Monitoring

**Ever-increasing performance requirements**

- **Our reliance on the Internet** makes us **victims of its success**, and **vulnerable to its shortcomings**

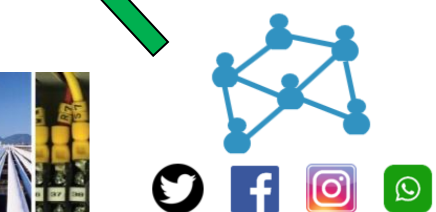
**Users**





media streaming


cloud platforms



online social networks

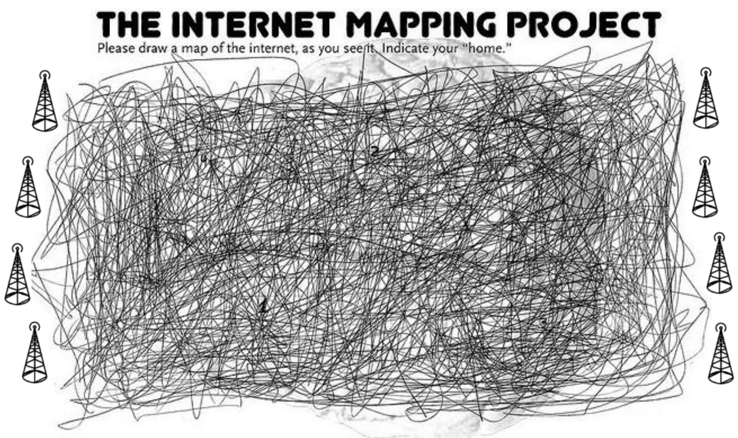
Contents

**Critical Infrastructure**

**Ever-increasing attack surface**

The **complexity of the Internet** requires **advanced monitoring and analysis** approaches to match the expectations in terms of **security**, **robustness**, **performance**, and **adaptability**

# AI4NETS, Big Data Analytics, and Network Monitoring




"**The Internet is the first thing that humanity has built that humanity doesn't understand,** the largest experiment in anarchy that we have ever had."

Eric Schmidt, former Google CEO

- **Hot Topic** in the agenda of **top Internet players**

## Goals of AI4NETS

- A **radical change in the way we manage** communication **networks**, relying on **AI & Big-Data Analytics**
- The vision – turn the **Internet "transparent"** and **"liquid"**
- More **secure & robust**, better **performance**, **greener**, and **self-adaptive to end-user needs** in real-time

# Two decades of AI4NETS

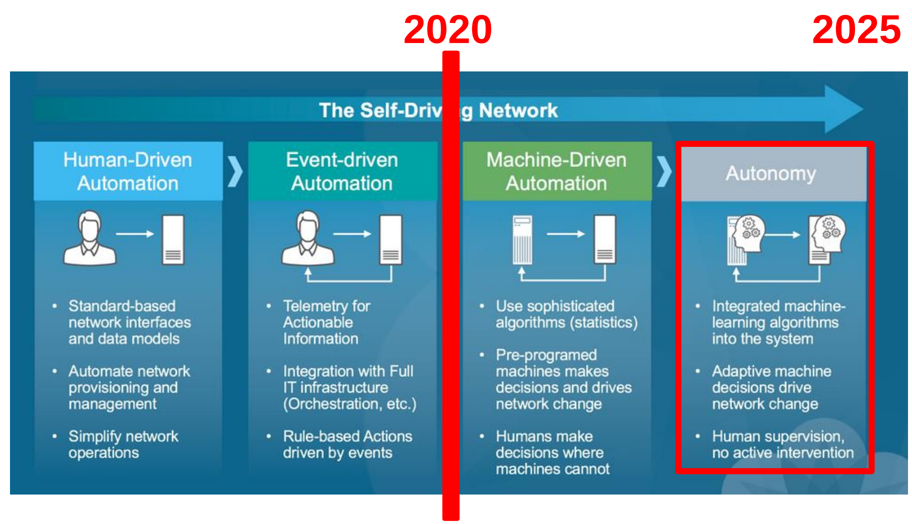

- **Cognitive Networking** (1998): networks with cognitive capabilities which could learn from past observations and behaviors, to better adapt to end-to-end requirements.

- Term **re-furbished along time**, referring to it as self-organizing networks, self-aware networks, **self-driving networks**, intelligent networks, etc.

- However, there is a **striking gap** between the **extensive academic research** and **the actual deployments of such AI-based systems in operational environments**.

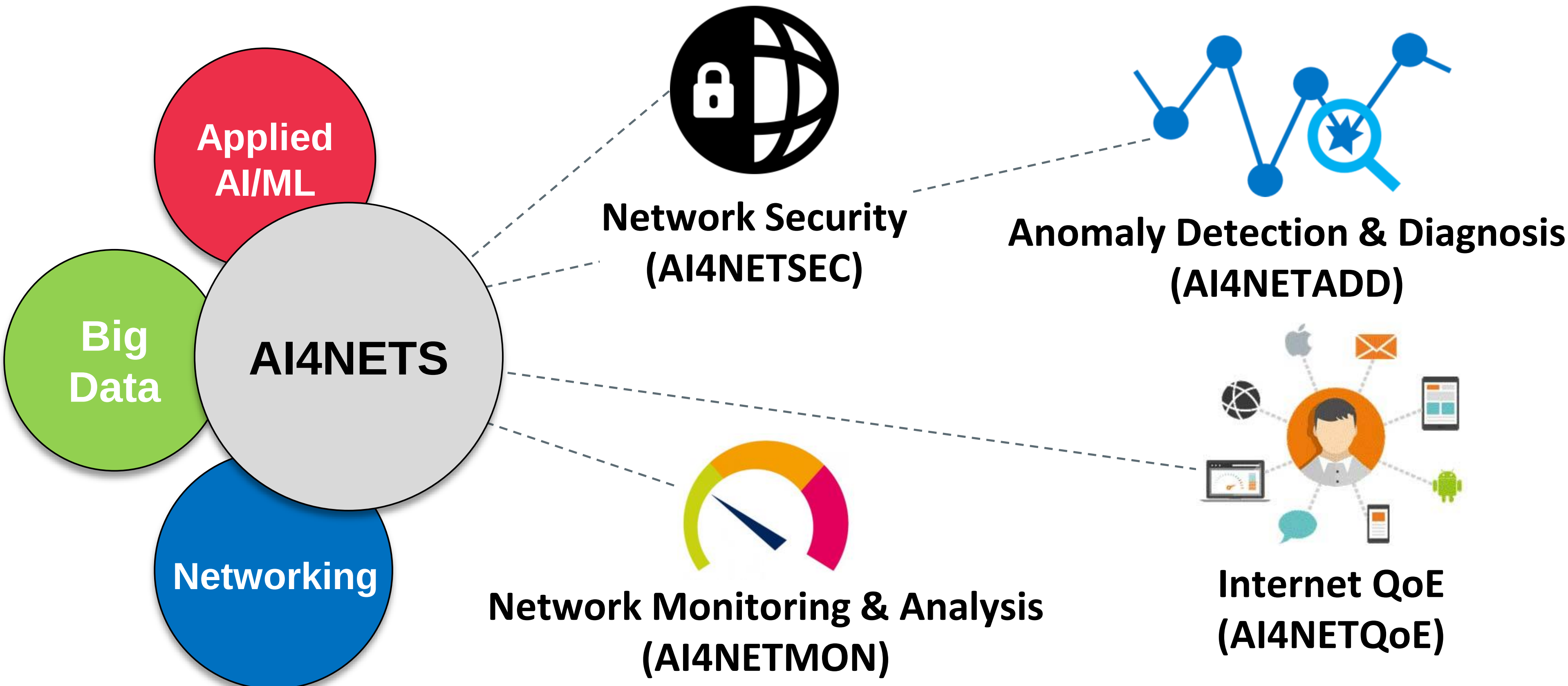
- **Why?** my take: there are still **many unsolved complex challenges** associated to the **analysis of Networking data through AI/ML.**

- **Hot Topic in the agenda of main Internet players:**
  - Network Operators
  - **Network Vendors (self-driving networks)**
  - Content Providers: the **Internet business of end-user engagement**



# AI4NETS – Application Domains in Networking

- AI4NETS represents the **meeting point** between applied **AI/ML**, **Big Data**, and **Networking**



# AI4NETS – Application to Other Domains

- **AI4NETS** represents a complex **context for AI/ML,** opening the door to **other fields**
- The **challenge is huge:**
  - AI4NETS **involves** all of the **major learning and big data challenges** (**the 4 Vs**)
  - **Massive volumes** of **complex** and heterogeneous **data** (**V**olume and **V**ariety)
  - **Fast** and **highly dynamic** streams of **data** (**V**elocity)
  - **Lack of ground truth** for learning (**V**eracity)


**IoT**

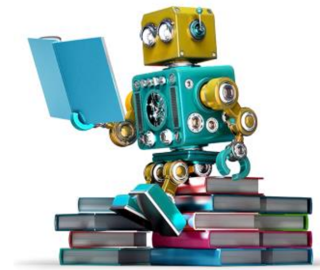

**Industry 4.0**

**Scalable Computing**

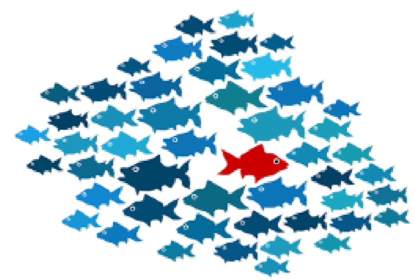
**Trading**

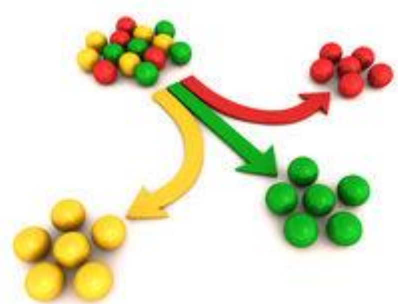
**Blockchains**

# Machine Learning for Network Analytics Use Cases – What do I do @AIT?



- Network Traffic Monitoring & Analysis
- End-User Experience (Internet-QoE)
- Cybersecurity & Anomaly Detection
- Network Performance Forecasting

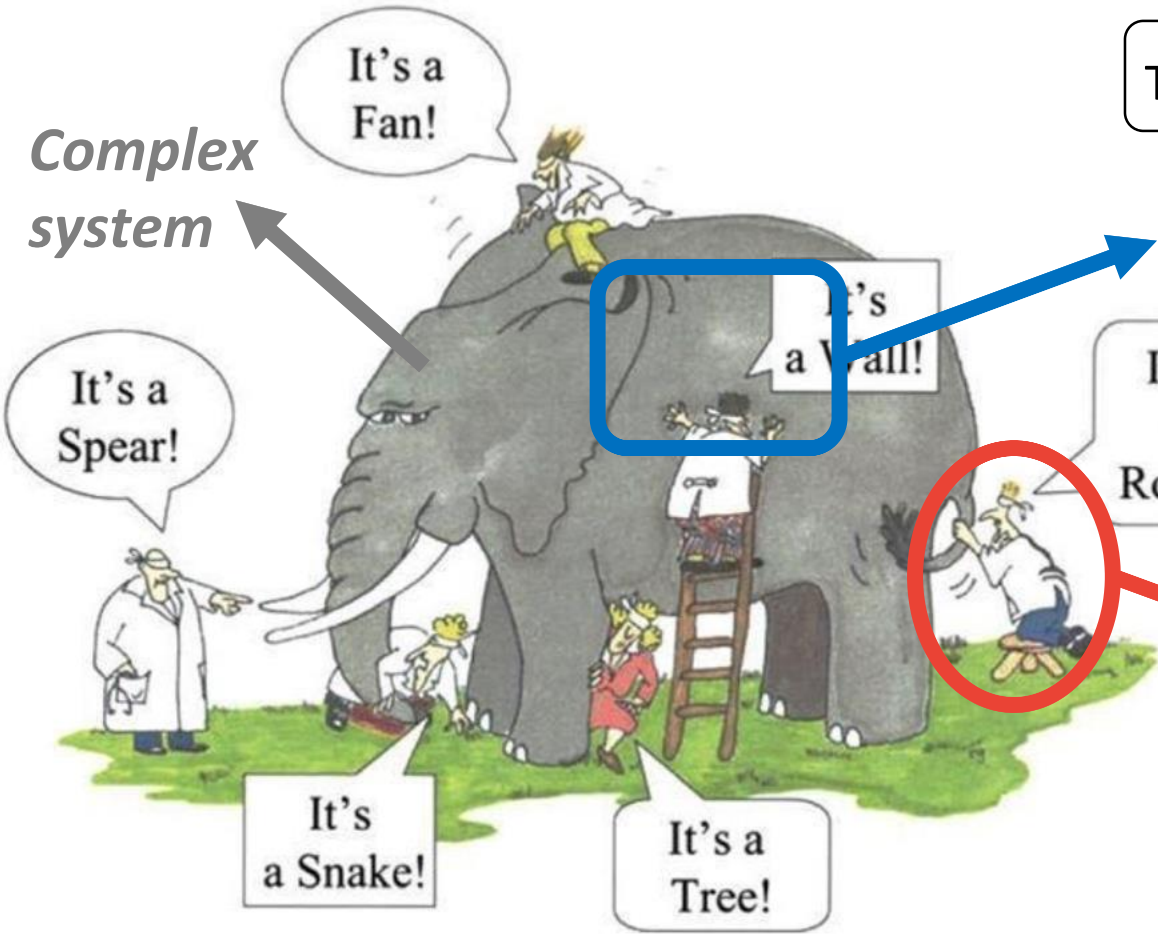





# AI/ML and a Data Driven Approach to Science


- ***Big Data*** + Machine ***(Deep) Learning*** = **data driven** analysis of complex systems

***Complex system***




The Achilles heel of ML/AI: ***BIAS***

***data is biased***: partial or misrepresentation of real system

***models are biased***: assumptions or hypotheses of behavior, mathematical properties, lack of transparency

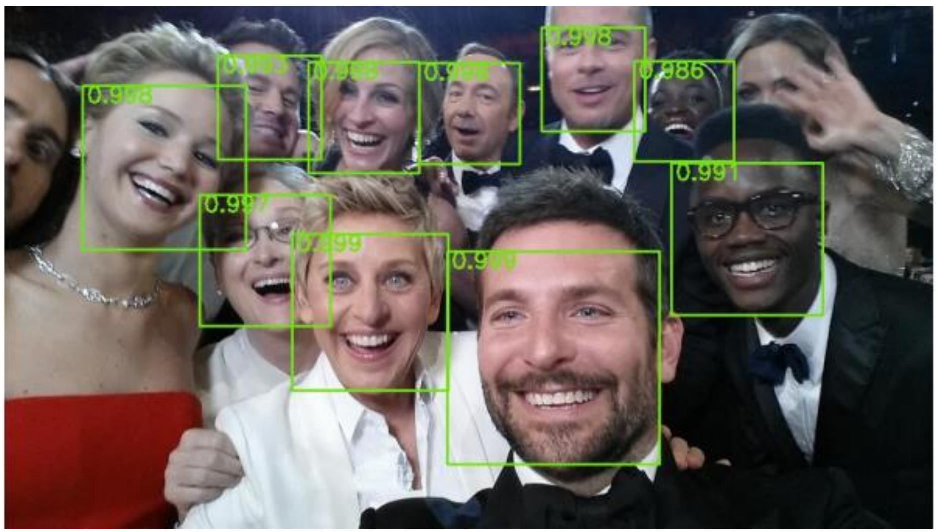
- The AI/ML model ***user is biased***, or unaware of the limitations of AI/ML: model **evaluation/testing**, model ***certification***, **correlation vs causality**

# What is Blocking AI/ML Success in Networking?

# What is Blocking AI Success in Networking?

- ***Data Complexity:*** the complexity (and heterogeneity) of the data related to Internet-like networks is one of the most significant bottlenecks to AI4NETS

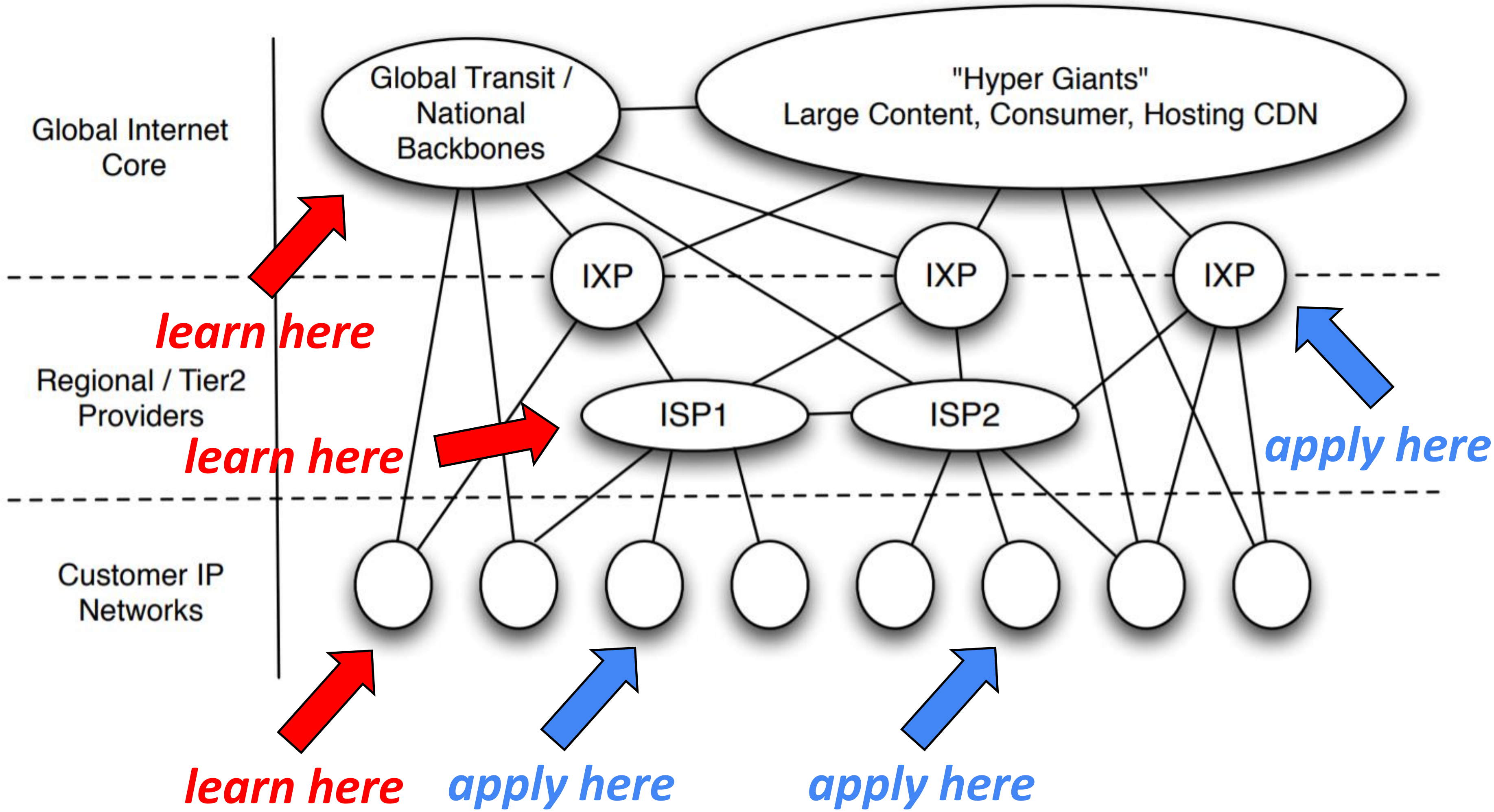
- The Internet, and in general large-scale networks, are a **complex tangle** of networks, technologies, applications, services, devices and end-users



- **AI has so far shown very successful results** generally in **data from** more **predictable** and **easy to understand sources** (natural sources)

# What is Blocking AI Success in Networking?

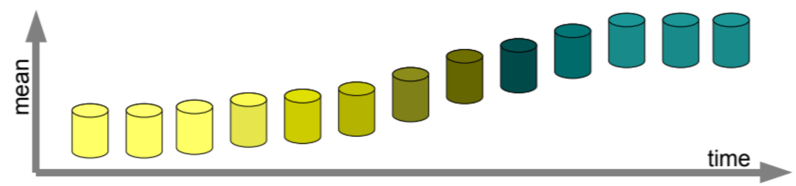
- ***Diversity of Network Data:*** besides complexity, network data often exhibits much more diversity than one would intuitively expect

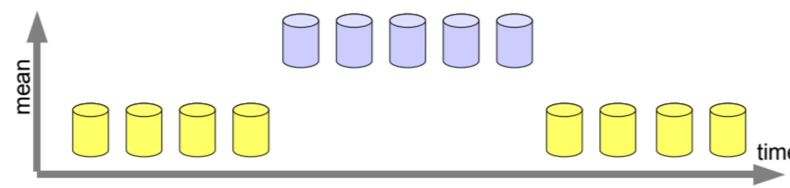


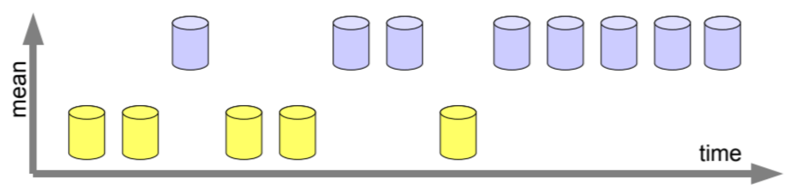
# What is Blocking AI Success in Networking?

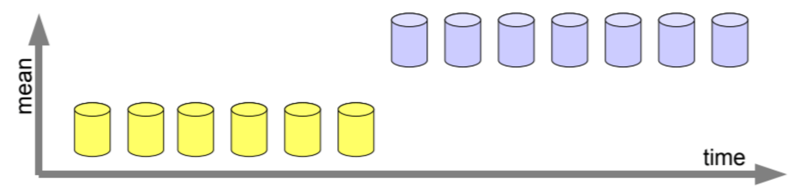
- ***Data Dynamics:*** networking data is non-stationary, generally comes in the form of data streams, and is full of constant concept drifts

*time*



# What is Blocking AI Success in Networking?

- ***Lack of Ground Truth:*** in the wild networking data is usually non-labeled
- ***Lack of Standardized and Representative Datasets***: datasets are generally biased, difficult to find appropriate public datasets to assess AI4NETS


- There is **no IMAGENET** or the like in Networking
- **Network data labeling**, and even **data interpretation**, is **too complex** for humans, even for **domain experts** (e.g., malware vs benign traffic instead of cat vs dog)
  - Easier for naturally generated data: images, text, audio

# What is Blocking AI Success in Networking?

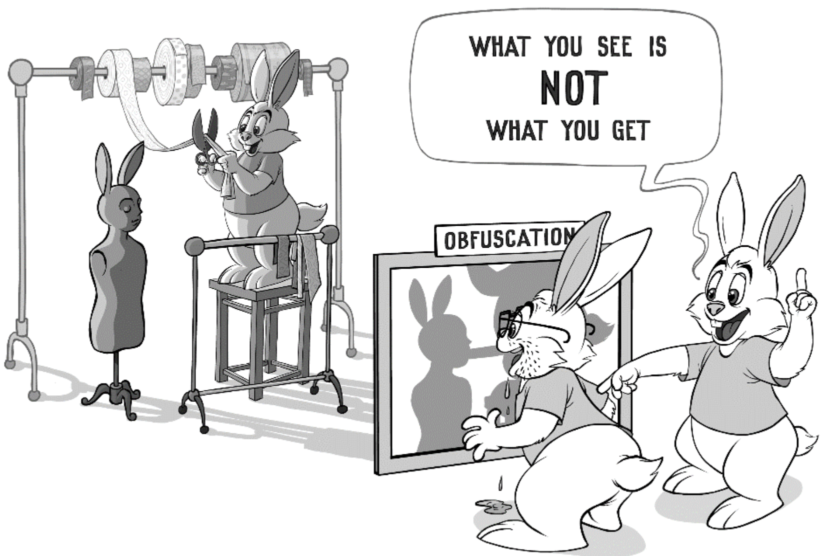
- ***Lack of Interpretability:*** this is a general problem of ML models (e.g., DL provides *beautiful* black-boxes)…but the issue is even more complex in AI4NETS

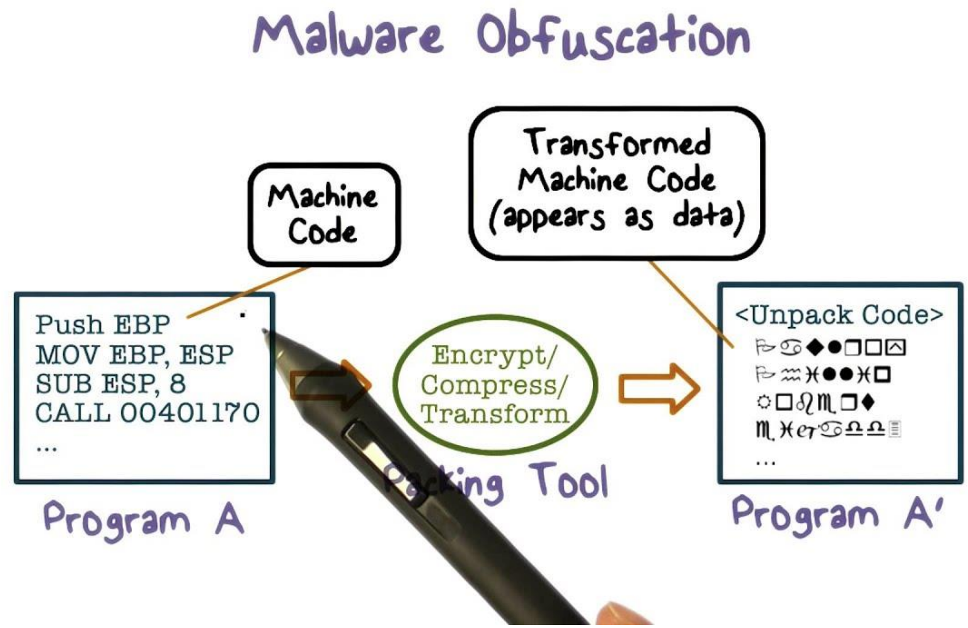
- To **improve trust**, the end-user (**humans**) has to **trust model predictions**, for example, by **understanding which inputs lead to a specific output**, but generally difficult to interpret networking features


- The **lack of interpretability and trust stops AI deployments**:
  - Network security – AI4NETSEC
  - Dynamic Traffic Engineering – AI4NETTE
  - Dynamic network instantiation (NFV) and (re)-configuration (SDN) – AI4SELFNET

# What is Blocking AI Success in Networking?

- ***Learning occurs in an Adversarial Setting:*** services obfuscate and modify their functioning to bypass monitoring and avoid traffic engineering policies

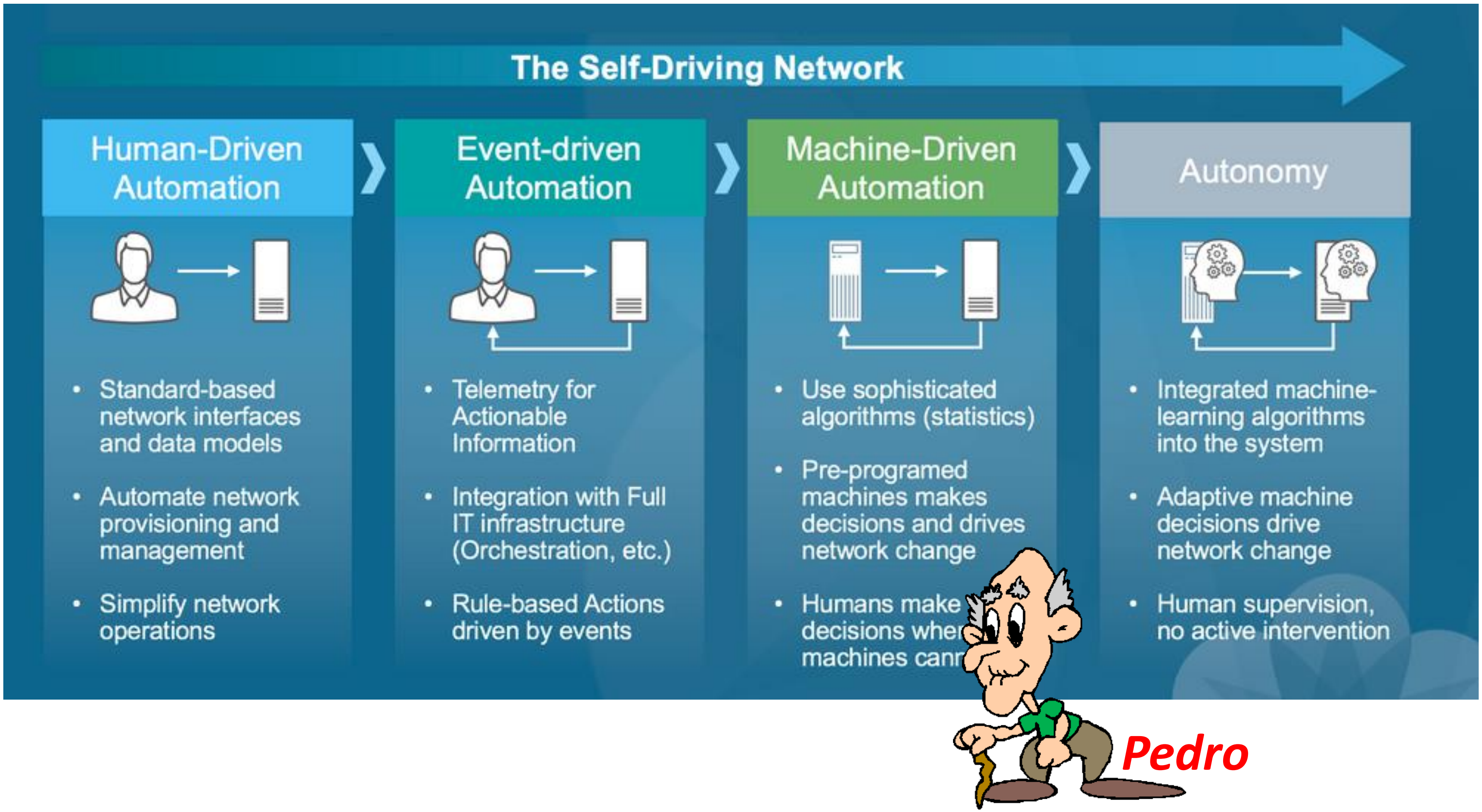



- It becomes even **more trickier to learn**, when the ***adversary*** constantly tries to fool the learner
- **Not only malign actors**, but standard *services*: Skype, QUIC, etc.

# What is Blocking AI Success in Networking?

- ***Lack of Learning Generalization:*** as a consequence of previous issues, **it becomes extremely difficult in the networking practice to learn models which can generalize to operational environments**

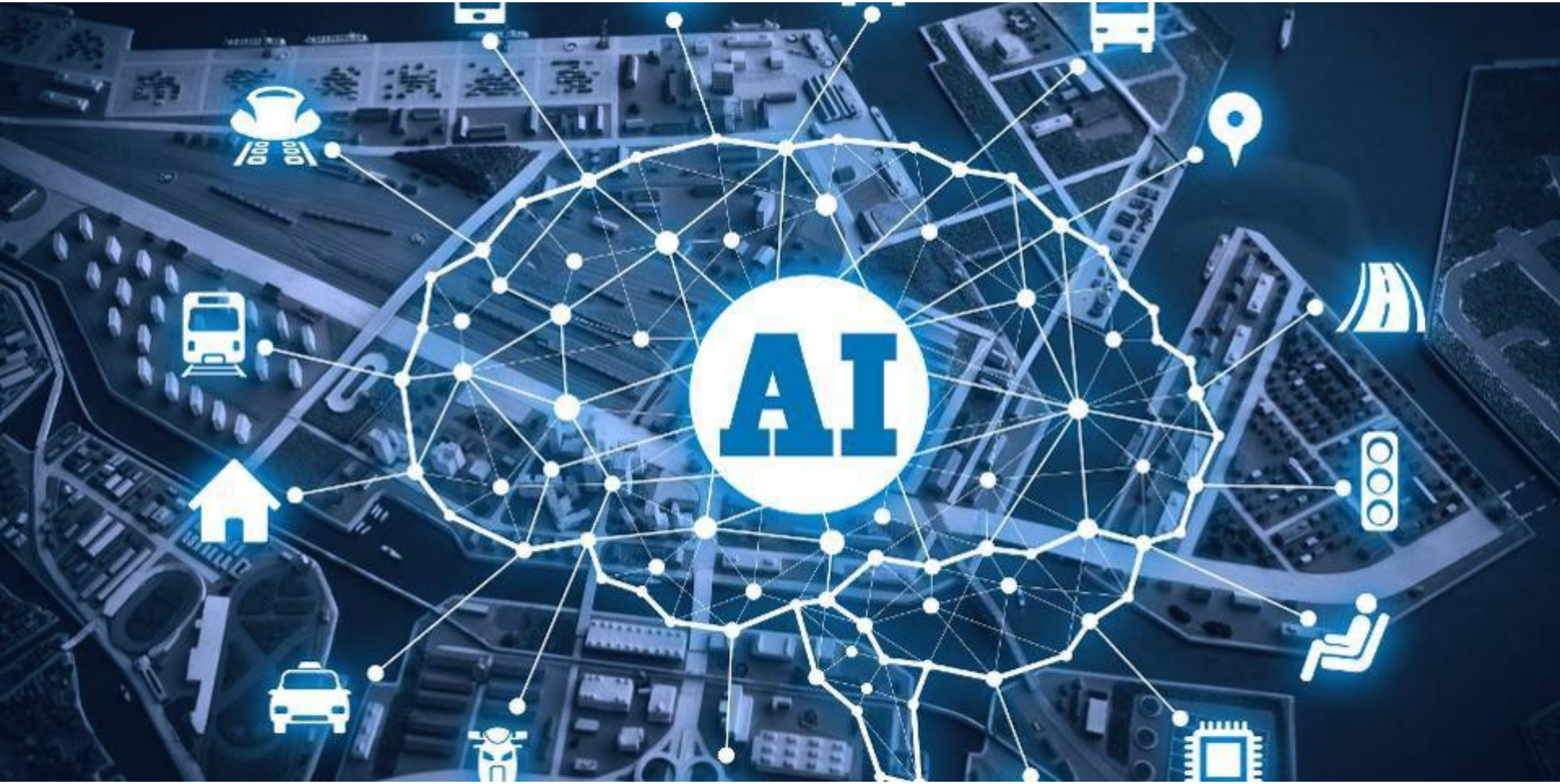



***Pedro***

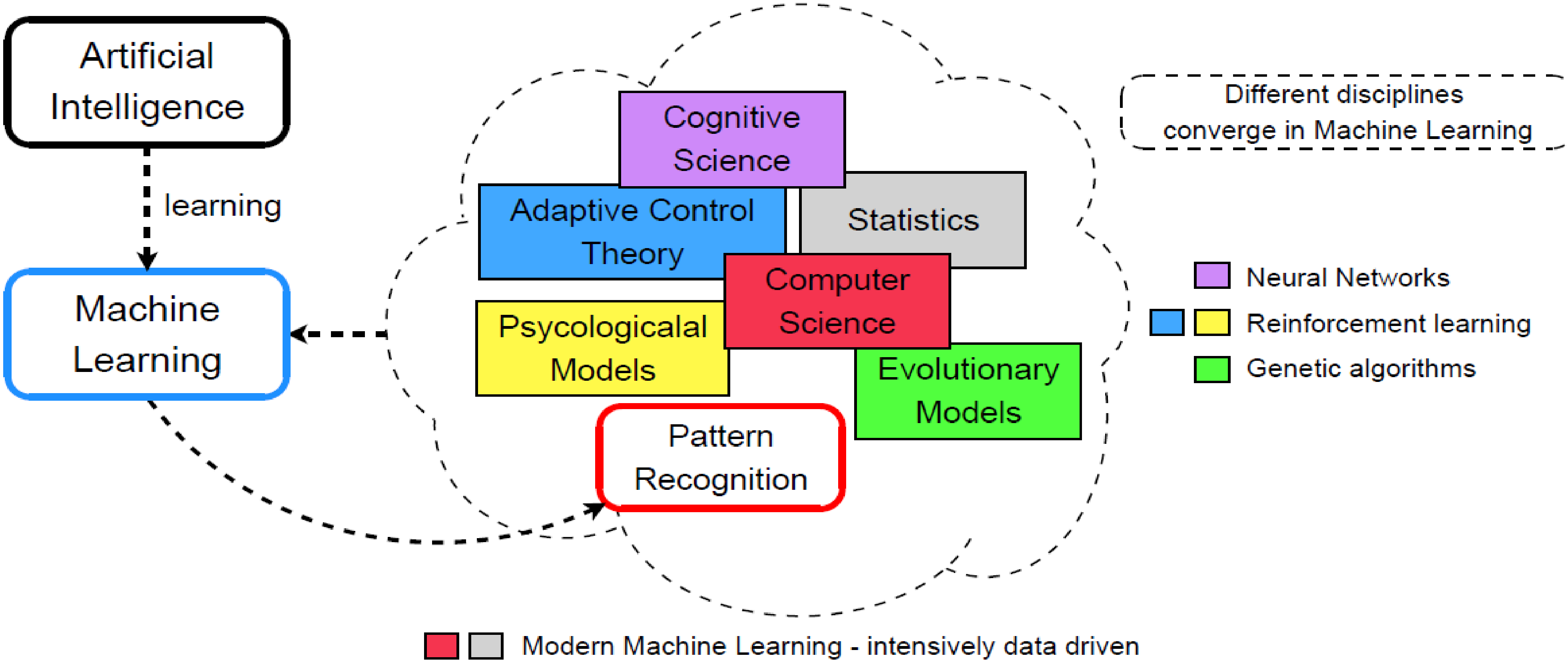
# Organization of the Talk
## Dealing with Some of these Challenges


- Deep Learning for Malware Detection – ***Avoid Feature Engineering***
- Explainable Artificial Intelligence (XAI) **–** ***Interpret Model Decisions***
- Super Learning for Network Security – ***Avoid Model Decision***
- Adaptive/Stream Learning for NetSec – ***Deal with Concept Drifts***

# Let’s take a step back and set a common ground ☺



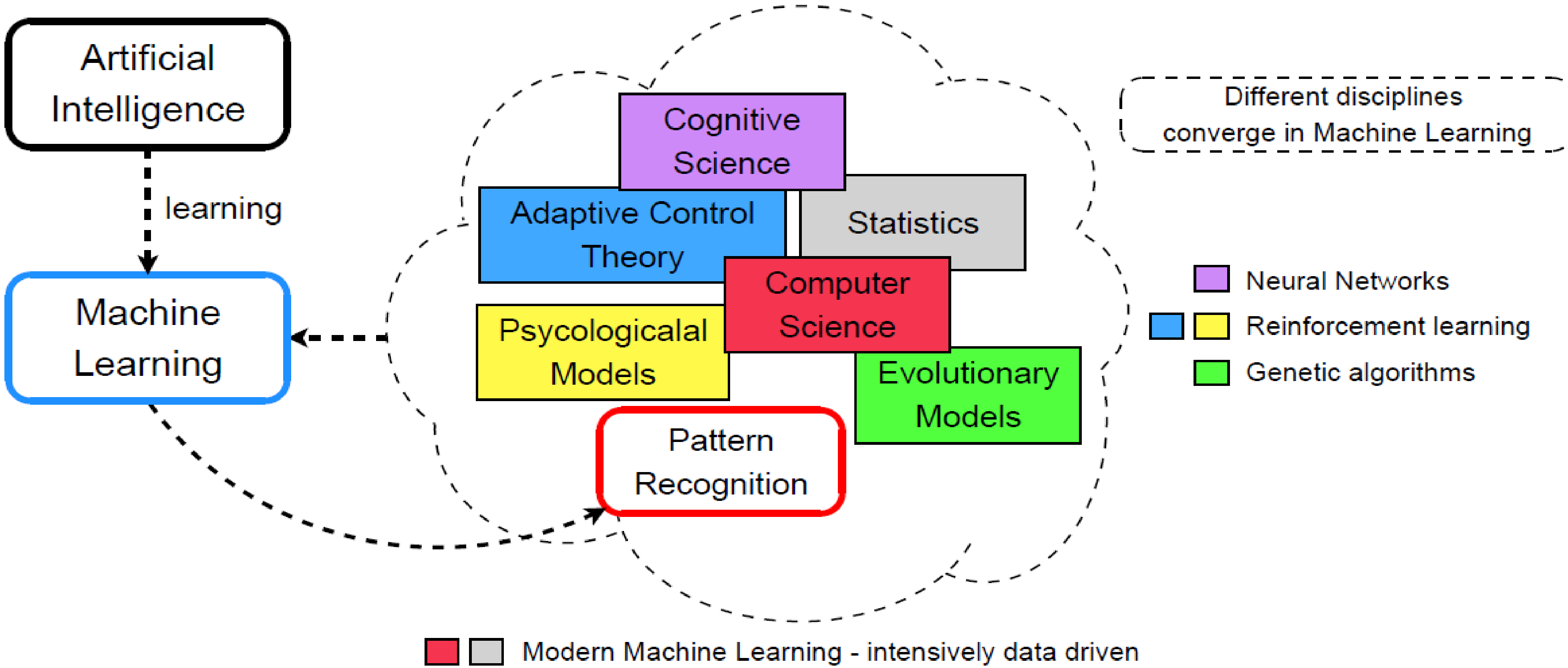
# A little bit of history...




- Minsky (TA'69), McCarthy (TA'71) & Solomonoff – AI/ML ~ Hinton, LeCun & Bengio (TA'18) – Deep Learning

1956 McCarthy (Stanford): "**Artificial Intelligence** is the science and engineering of making intelligent machines, which can perceive their environment and take actions to maximize their chances of success".

# A little bit of history...

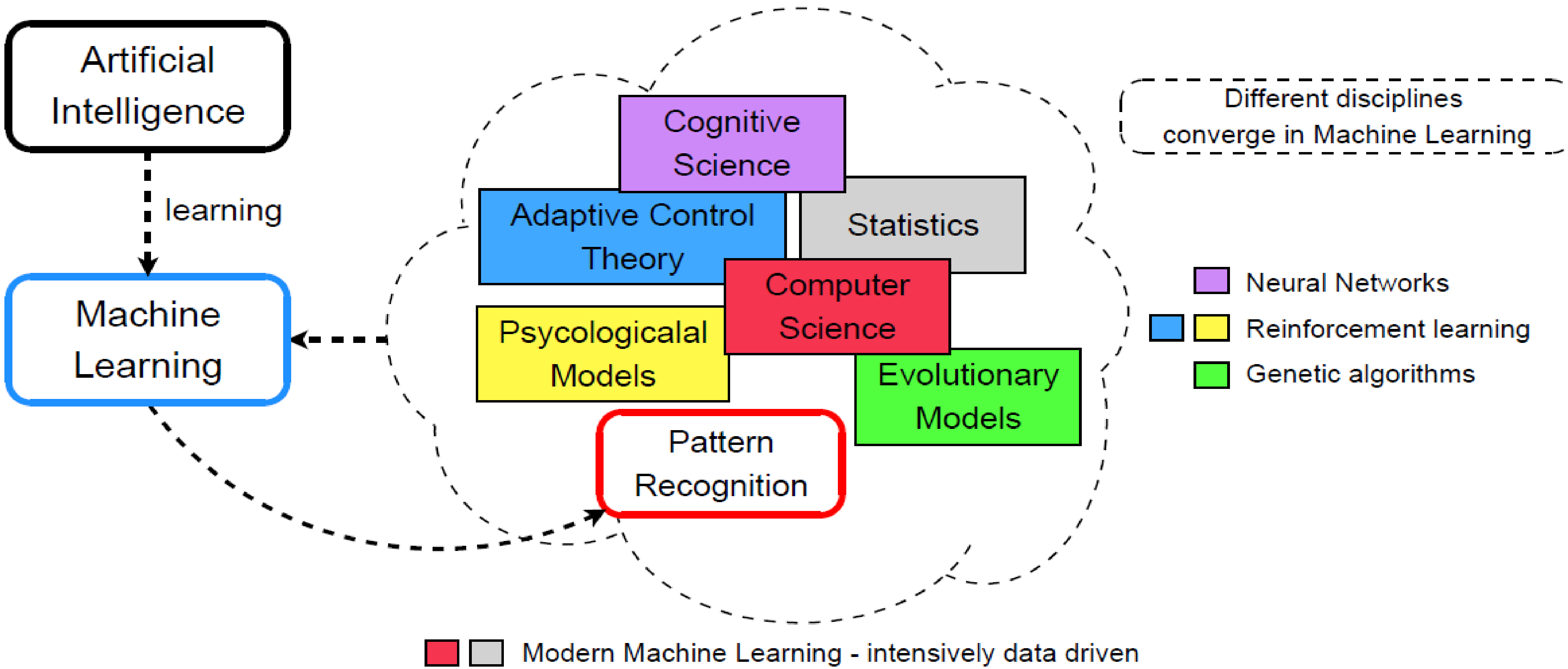



1956 Dartmouth Summer Research Workshop on AI - founding event of AI as a field.

1956 Ray Solomonoff first mentioning the term "Learning Machines"...

1980 ...but the first International Workshop on Machine Learning (currently ICML) appears almost 25 years later.

# A little bit of history...




**Machine Learning (ML)** is about computational approaches to learning: ML aims to understand **computational mechanisms** by which **experience can lead to improved performance**, traducing these into computer algorithms.

# ML: discipline vs tool to solve complex problems

ML in TMA **IS NOT** about trying different algorithms to obtain better results.

To build a solid house on your own, you need to know about architecture, as well as about the intrinsic characteristics of the construction toolbox...

Two commonly arising problems when coupling ML and Networking:

## (I) You have to understand the problem:

- Even a ML expert fails to achieve a good networking solution if he neither knows the good descriptors nor understands the problem (e.g., try to classify flows using only port numbers).
- Keep the scope narrow, to better understand the overall process (i.e., from selecting features to evaluation and conclusions).
- The solution must be meaningful in practical terms (e.g., predicting QoE from descriptors that can't be controlled is pretty useless for QoE management).

# ML: discipline vs tool to solve complex problems

ML in TMA **IS NOT** about trying different algorithms to obtain better results.

To build a solid house on your own, you need to know about architecture, as well as about the intrinsic characteristics of the construction toolbox…

Two commonly arising problems when coupling ML and Networking:

## (II) You have to understand the tool:

- The broader overview you have about the particularities of each ML approach, the better chances to apply the correct one (e.g., avoid killing mosquitos with a hammer).
- The research community does not benefit any further from yet another untried ML approach (e.g., IDS based on KDD'99 dataset).
- A good grasp of calculus, linear algebra, and probability is essential for a clear understanding of ML and PR in TMA and Networking.

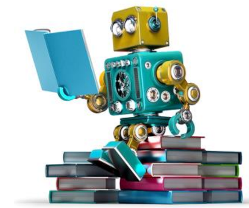
# Organization of the Talk

## Dealing with Some of these Challenges



- **Deep Learning for Malware Detection – *Avoid Feature Engineering***
- Explainable Artificial Intelligence (XAI) **– *Interpret Model Decisions***
- Super Learning for Network Security – ***Avoid Model Decision***
- Adaptive/Stream Learning for NetSec – ***Deal with Concept Drifts***

# Artificial Intelligence – As Smart as a Donut!



- **Machine Learning** is still **very stupid** – the **big revolution** is on **big data processing** and **data availability**/accessibility

- **Current ML benefits** are fundamentally due to machines ability to **blindly**:
  - compute lots of math operations per second
  - handle large amounts of data
  - deal with data in high-dimensional spaces

- **A lot of data required** to “learn“ simple logical inter-relations

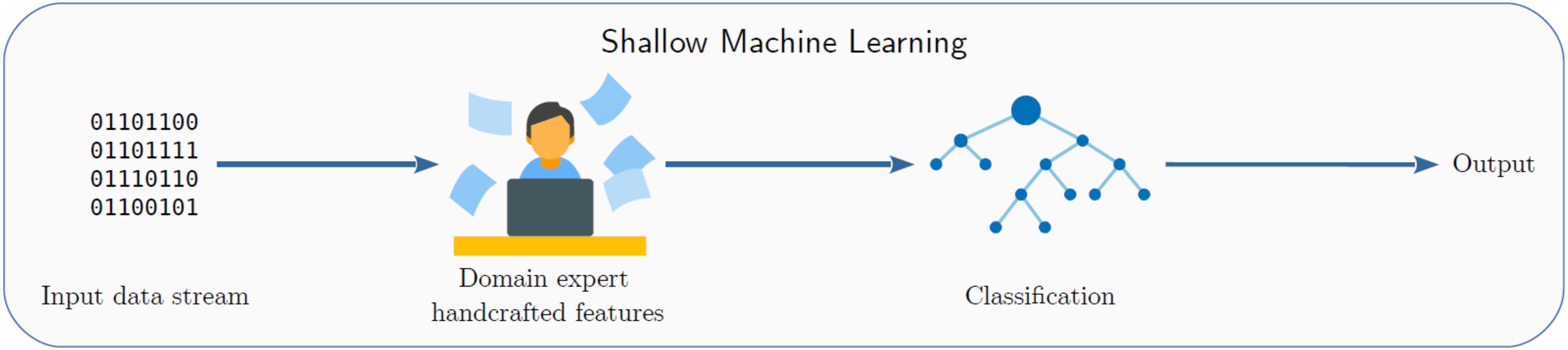
- **Shallow Learning**: less data but ***human expert knowledge required***, to properly guide the ***feature engineering*** process

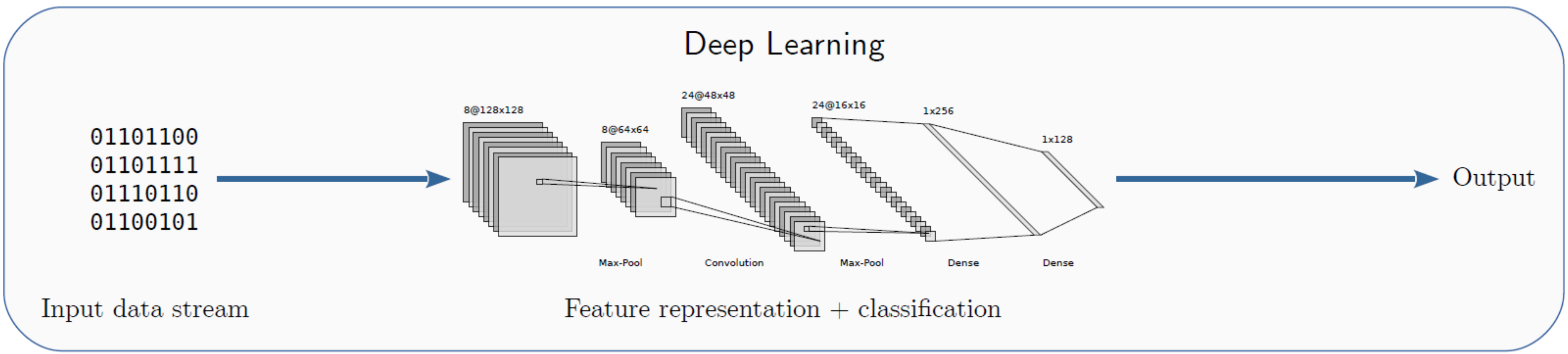
- **Deep Learning**: ***automated feature engineering*** (representation learning) but ***needs much more data***

**RawPower**

we explore **deep learning** for ***blind*** malware detection in network traffic

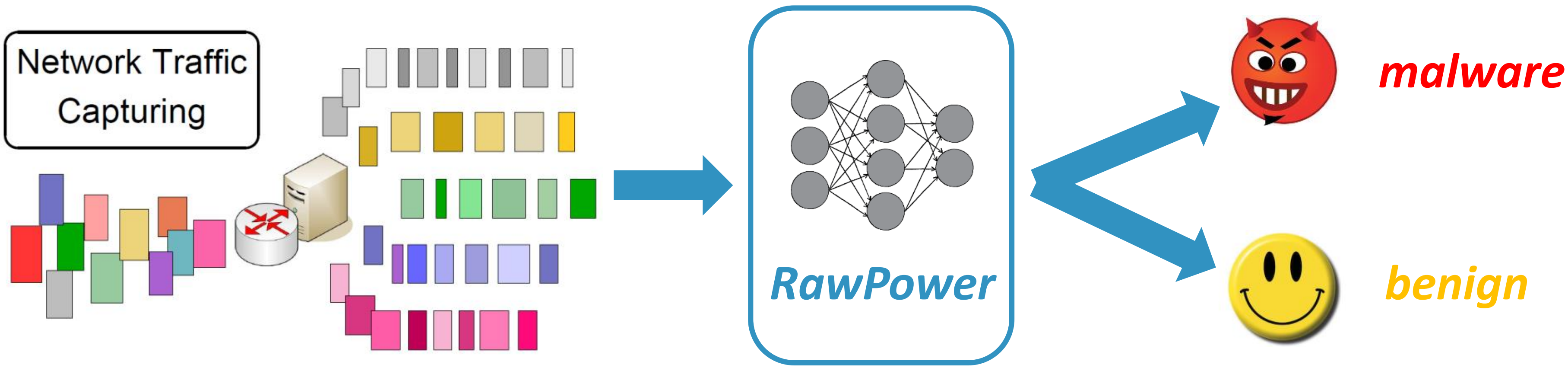
# Shallow Learning vs Deep Learning



# Basic Concepts of RawPower

- The ***input to the Deep Learning*** model is **RAW** – ***only byte-streams***
- **No need to define** tailored, domain-knowledge-based **input features**



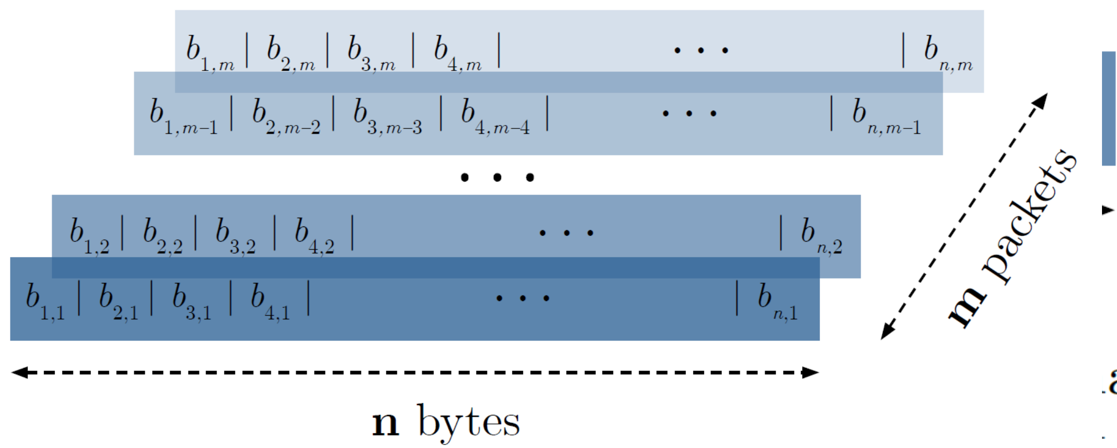

- Different architectures to **analyze both packet-based** and **flow-based byte aggregations**
- Models for **binary malware detection** – fully **supervised-based training**

# Raw Input Representations

- **Input representation of the data**, as well as **network architecture**, are both **key elements** to consider when **building a DL model**

- We take **two types of raw input representations:** ***packets*** and ***flows***. Decimal normalized representation of every byte of every packet is a different input

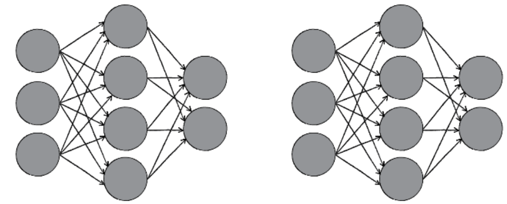
- **Flow representation:** matrix-like input, first **m** packets x first **n** bytes




(a) Pac ... ape of the
input da ... and $n$ the
number

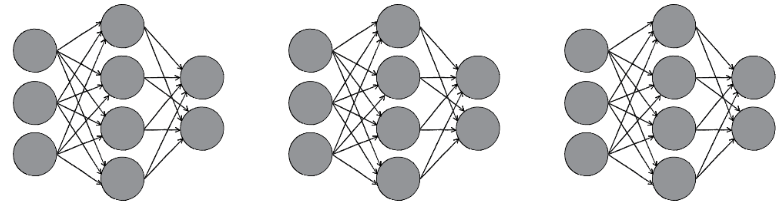
(b) Flow representation for the input data. A tensor of size $(N, m, n)$ where $N$ represents the number of instances –flows–, $m$ the number of channels –packets– and $n$ the number of steps –bytes–.

# Deep Learning – Architectural Principles

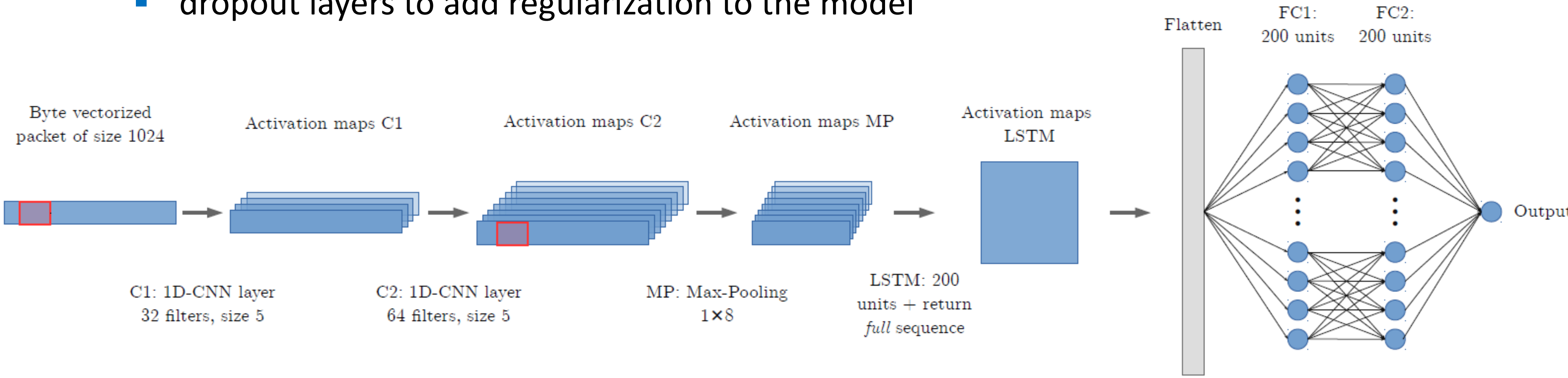


- The **core layers** used for both models are basically two: **convolutional and recurrent**
- Convolutional, to build the feature representation of the spatial data inside the packets and flows
- Recurrent layers are used together with the convolutional ones to allow the model keeping track of temporal information
- **Fully-connected layers** to deal with the different feature combinations
- **Batch Normalization**: layer inputs are normalized for each mini-batch. As a result: higher learning rates can be used, model less sensitive to initialization and also adds regularization
- **Dropout**: randomly drop units (along with their connections) from the neural network during training. A very efficient way to perform **model averaging**

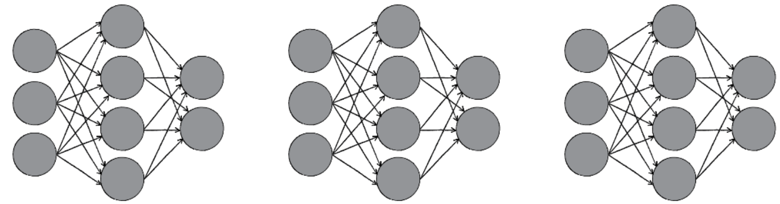
# DL Architectures – Packets



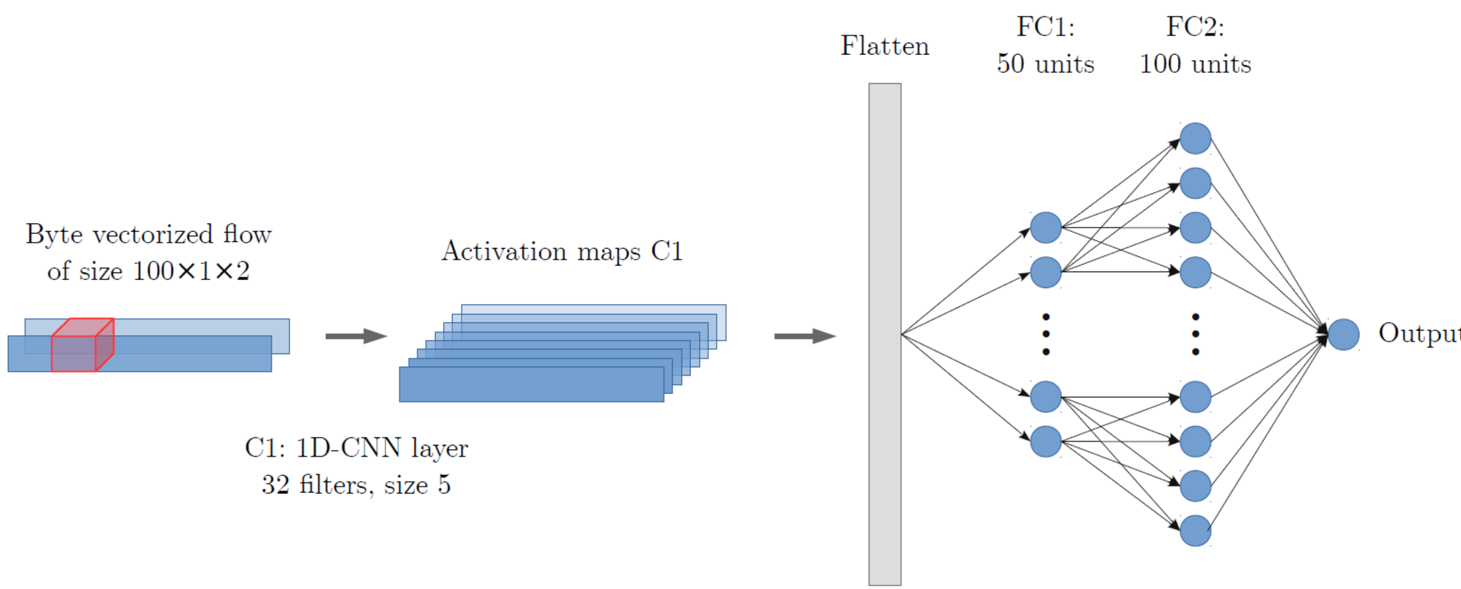
- **Raw Packets Architecture:**
  - **n** is set to first **1024 bytes**
  - two 1D-CNN layers of 32 and 64 filters (size 5) respectively
  - MP - max pooling layer (size 8)
  - LSTM layer with 200 neurons
  - two fully-connected layers of 200 neurons each
  - **binary cross-entropy as loss function**
  - spatial and normal batch normalization layers after each 1D-CNN and FC layers to ease training
  - dropout layers to add regularization to the model



# DL Architectures – Flows

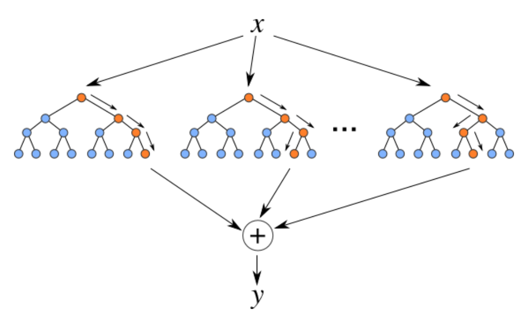


- **Raw Flows Architecture:** we go for a simpler model, with ***less features***
  - **n** is set to first **100 bytes**, and **m** to first **2 packets**
  - one 1D-CNN layers of 32 filter (size 5)
  - two fully-connected layers of 50 and 100 neurons each
  - **binary cross-entropy as loss function**
  - spatial and normal batch normalization layers
  - dropout layers to add regularization to the model



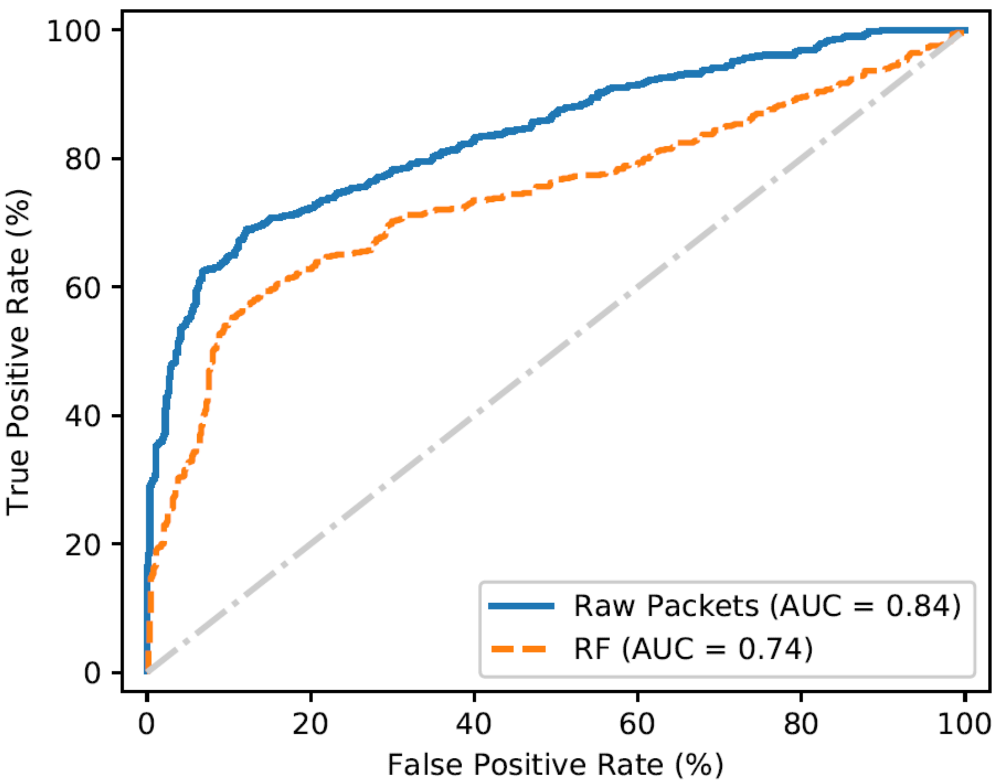
# Evaluations

- All evaluations run on top of **Big-DAMA cluster** (distributed CPU)
- **Keras framework** running **on top of TensorFlow**
- **Dataset**: ***malware*** and ***normal*** traffic captures (pcap) performed by the Stratosphere IPS Project of the CTU University of Prague
- **250.000 raw packet** instances, **70.000 raw flow** instances
- **80%** of the samples for **training**, **10%** for **validation** and **10%** for **testing**
- **Compare** performance to **highly expressive Random Forest**:
  - same raw inputs
  - 100 trees
  - max depth and instances per leaf set for high expression
  - **selected based on great outperformance in state of the art**



# RawPower – Packet Representation

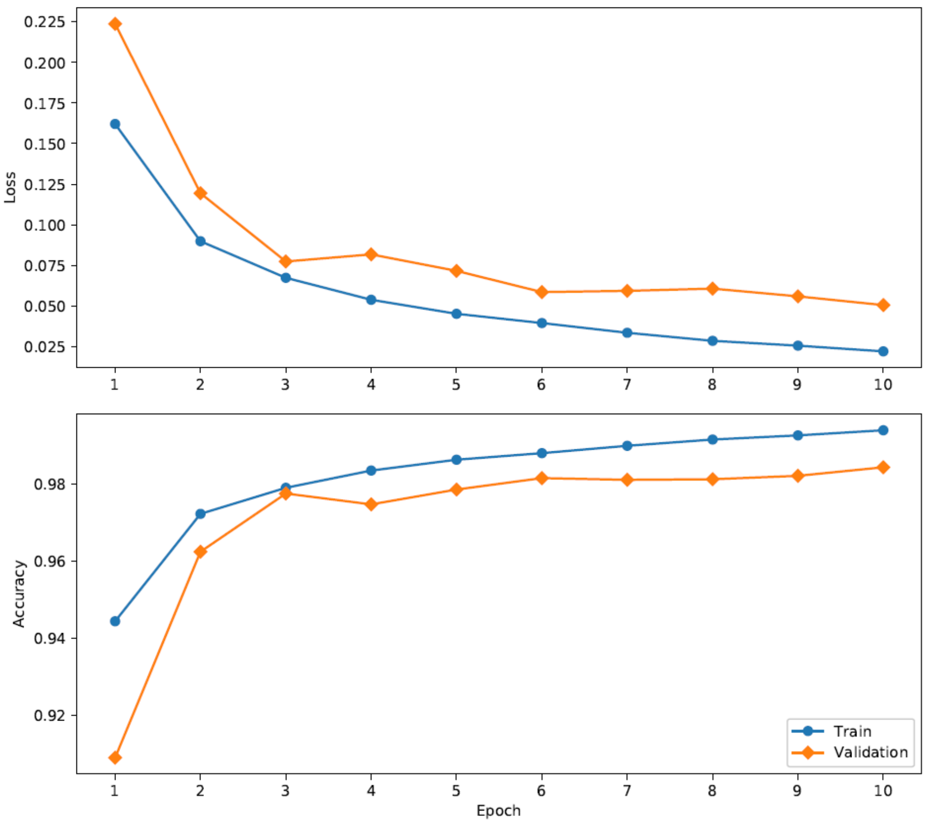
- Malware consists of **10 different malware types**, collected at controlled environment

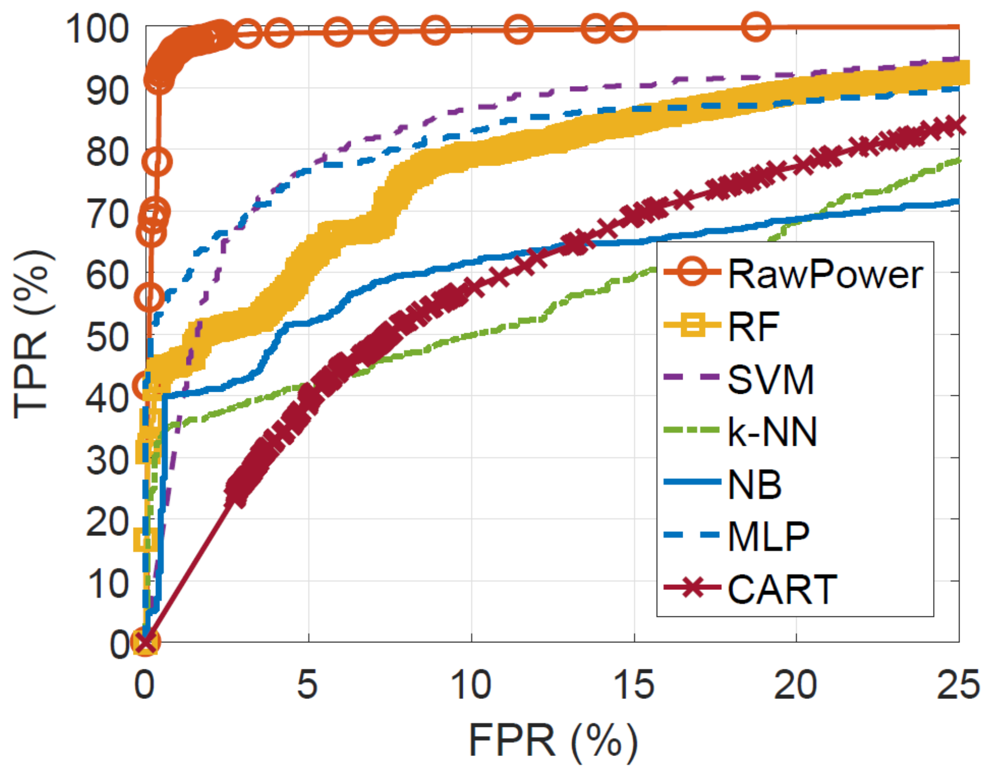



- **ROC curves** for both RawPower and RF
- Both models using the same **raw packet inputs**
- **Performance is not good at the packet-level**
- **Little gain w.r.t. a simple RF model**

# RawPower – Flow Representation vs Shallow ML

- **Training and validation** evolution over **10 epochs**
- **Much better performance** at the **flow level**
- RawPower can **detect almost 98% of the malware** flows with a FPR < 0.5%
- **Shallow models not able to capture the underlying relations**




*RawPower* vs. shallow models.

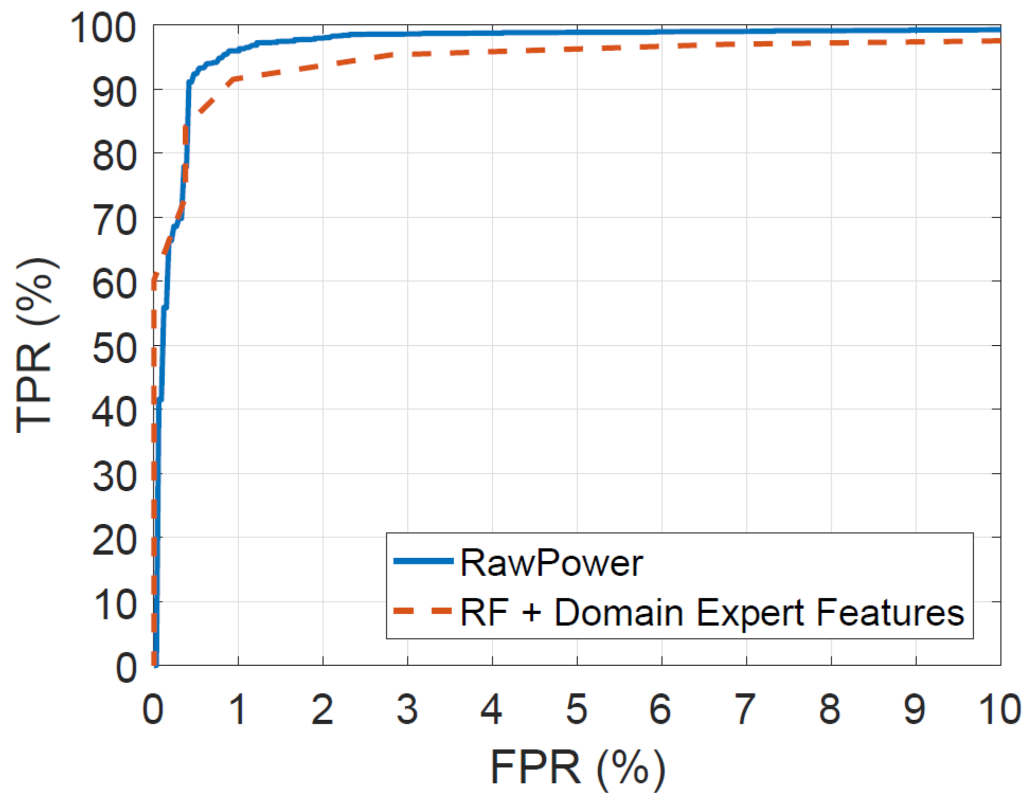
# RawPower – Flow Representation vs Expert Features

- Comparison **against** traditional **RF-based model**, which uses highly engineered **input features**, extracted from **domain knowledge**

- Both models provide **comparable results**

- The **key advantage of RawPower** is to rely directly on the usage of **bytestream raw data as input**

- **Input representation learning**: no the need for feature engineering




*RawPower* vs. knowledge-based inputs.

# Organization of the Talk
## Dealing with Some of these Challenges



- Deep Learning for Malware Detection – ***Avoid Feature Engineering***

- **Explainable Artificial Intelligence (XAI) –** ***Interpret Model Decisions***

- Super Learning for Network Security – ***Avoid Model Decision***

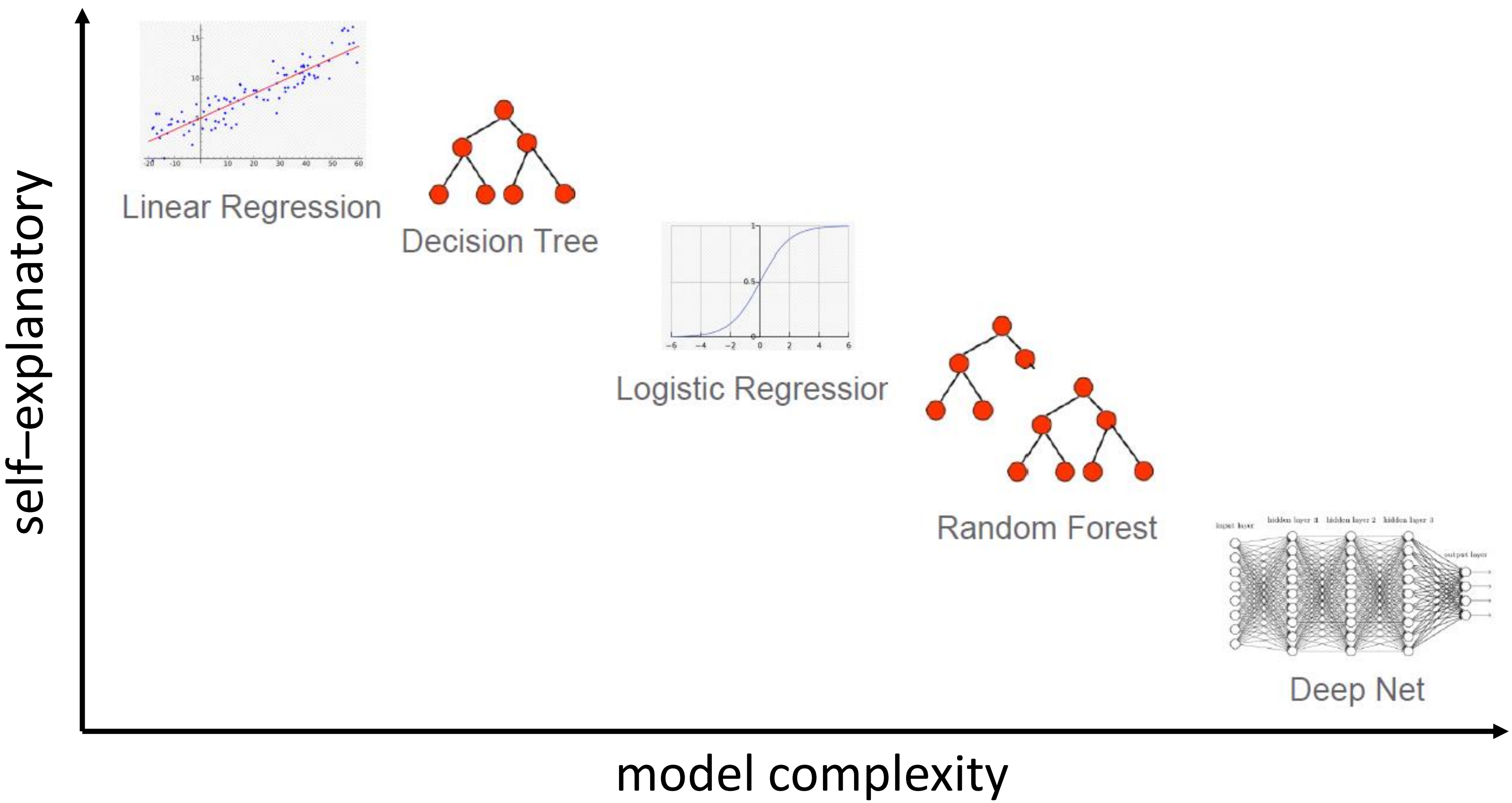
- Adaptive/Stream Learning for NetSec – ***Deal with Concept Drifts***

# EXplainable AI (XAI) – Why Should I Trust You?

- **ML models →** mostly are **black boxes** (exceptions: linear models, decision trees, etc.) – e.g.: some popular ML models have 10s of millions of parameters!
- Models are evaluated off-line before deployment on available test datasets – **data @runtime might change** (concept drift)
- Humans want to **understand model's behavior to gain trust** (applicability in the practice)
  - trusting an individual model's prediction
  - trusting a model (inspect a set of representative individual predictions)
- **Explainable AI:** approaches capable to explain models and individual predictions, by tracking back to the inputs leading to a certain output

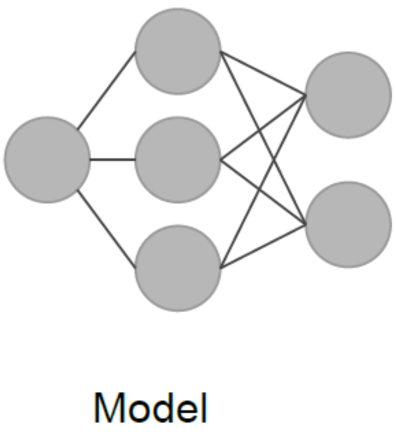
# Why XAI?

- **Ideally**, **ML models** should be ***self-explanatory:*** **improve end-user understanding** and **trust**, by offering simple **explanations** of the ***"whys" of certain decision***
- **Only few models are self-explanatory:**



# A Simple XAI Example

- **Application Example:** AI-supported disease diagnosis




- Explainer: **LIME** - **Local** Interpretable **Model-agnostic** Explanations
- ***LIME approach***: builds an **interpretable model** that is **locally** faithful to the classifier under analysis
- Other approaches: **SHAP**, LRP (NNs), PDP, etc.

# Desired Properties of an *Explainer*

- ***Interpretable*** → qualitative understanding between input features and response
  - user's limitation taken into account, e.g. no hundreds of features presented to the user

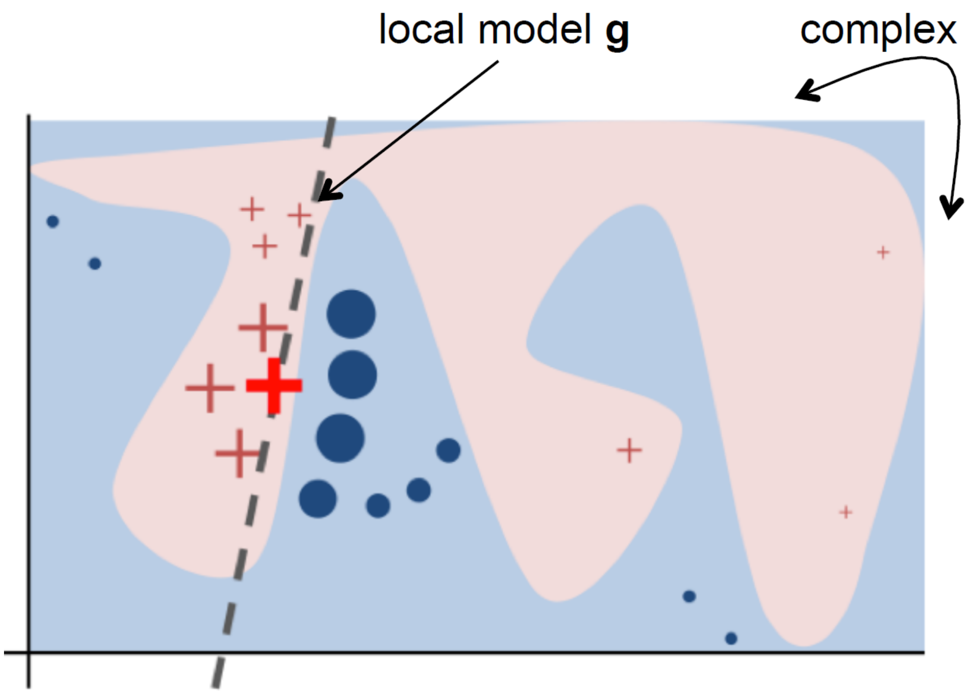
- ***Locally faithful*** → must correspond to original model behavior in the vicinity of the explained prediction
  - globally faithful and interpretable explanations are hard to achieve for complex models

- ***Model-agnostic*** → flexibility; can be used with any model
  - treat the original model as a black box, no reference to any specific property of the original model in the explanation

- ***Global perspective*** → trust in the model; present (few) representative explanations to the user
  - build upon explanations for individual predictions; helpful for model selection

# LIME in a Nutshell – Sampling for Local Exploration

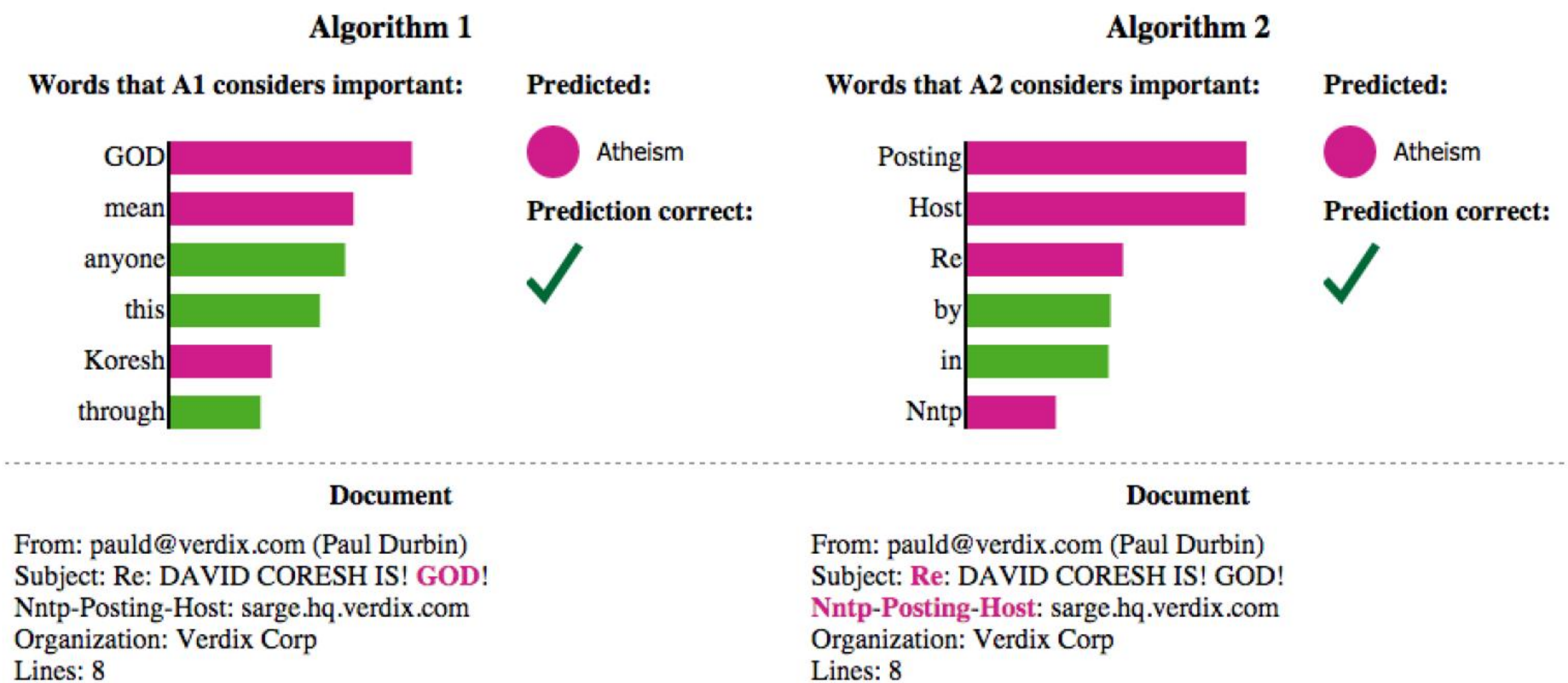
- Let **f** be an unknown complex decision function (blue/pink background)




- The **bold red-cross (x)** is the instance we want to explain
- LIME samples instances **z** around **x**, weighted by some **similarity measure** $D_x \rightarrow D_x(z)$ is higher for instances closer to x
- Using model **f**, gets the corresponding predictions **f(z)**
- Finally, it uses **z** and **f(z)** to **build an interpretable model g** (e.g, linear) around **x**
- **g** is *interpretable*, *locally faithful to* **f** (captured by $D_x$), and *model agnostic* (uses **f(z)** as labels)
- *robust to sampling noise*, thanks to $D_x$

# LIME Examples (I) – Model Comparison/Selection

- Task: word-based **email classification**, Christianity or Atheism
- 2 models (Algorithm 1 vs Algorithm 2), **which one is better?**




- Algorithm 2 is better than Algorithm 1 in terms of **accuracy in validation…**
- …but Algorithm 2 makes predictions for arbitrary reasons…Algorithm 1 is better
- **Performance metrics should be carefully considered**

# LIME Examples (II) – Model Performance Evaluation

- Task: **image classification**, using Google's pre-trained Inception CNN architecture

(a) Original Image

- Figs. (b,c,d) report super-pixel explanations provided by LIME
- Top 3 classes: ***Electric Guitar*** ($p$ = 0.32), ***Acoustic Guitar*** ($p$ = 0.24), and ***Labrador*** ($p$ = 0.21)
- The image is wrongly classified, but **explanations provide trust** in the model, as they are reasonable

# LIME Examples (III) – Discover Biased Data

- Task: train a classifier to distinguish between Wolves and Huskies
- Biased data (e.g., **undesirable strong correlations**) → wrong classifier
- **Hard to identify** by looking at the raw data and predictions

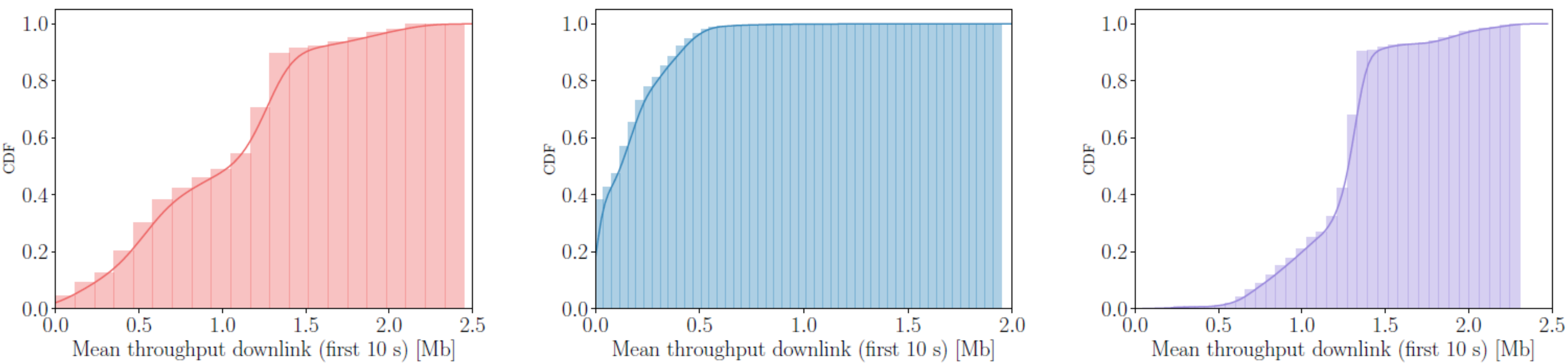
(a) Husky classified as wolf (b) Explanation

- Bias@training: all pictures of Wolves had **snow in background**
- The **classifier performs well** according to **cross-validation** in this **biased dataset...**

- ...but **explanations** of individual predictions show that **the model learnt a biased pattern**: if snow → wolf, else → Husky

# EXPLAIN-IT: Steps to XAI for Unsupervised Learning

- **Task:** unsupervised learning through **Clustering**
- **Challenge:** automatically **interpret clustering results**, beyond traditional structural clustering validation approaches (e.g., silhouette, completeness, homogeneity, etc.)




(a) YouTube sessions in Cluster 0 (b) YouTube sessions in Cluster 1 (c) YouTube sessions in Cluster 2

**Figure 3: Average downlink throughput (ADT) distribution on the first 10s slot, per cluster. While there is strong variance within each cluster, ADT distributions are ordered, with ADT($C_1$) < ADT($C_0$) < ADT($C_2$).**

**Figure 1: The EXPLAIN-IT system. Data is firstly embedded into the *exploration space*, relying on expert knowledge when available. The *summary space* is the result obtained by clustering the exploration space. Next, a supervised, data splitting model is built out of the clustering results. Finally, an XAI approach (LIME) is applied to this splitting model, interpreting the contents of the clusters by adding local interpretations.**

# Organization of the Talk

## Dealing with Some of these Challenges



- Deep Learning for Malware Detection – ***Avoid Feature Engineering***
- Explainable Artificial Intelligence (XAI) – ***Interpret Model Decisions***
- **Super Learning for Network Security –** ***Avoid Model Decision***
- Adaptive/Stream Learning for NetSec – ***Deal with Concept Drifts***

# Ensemble Learning for Network Security

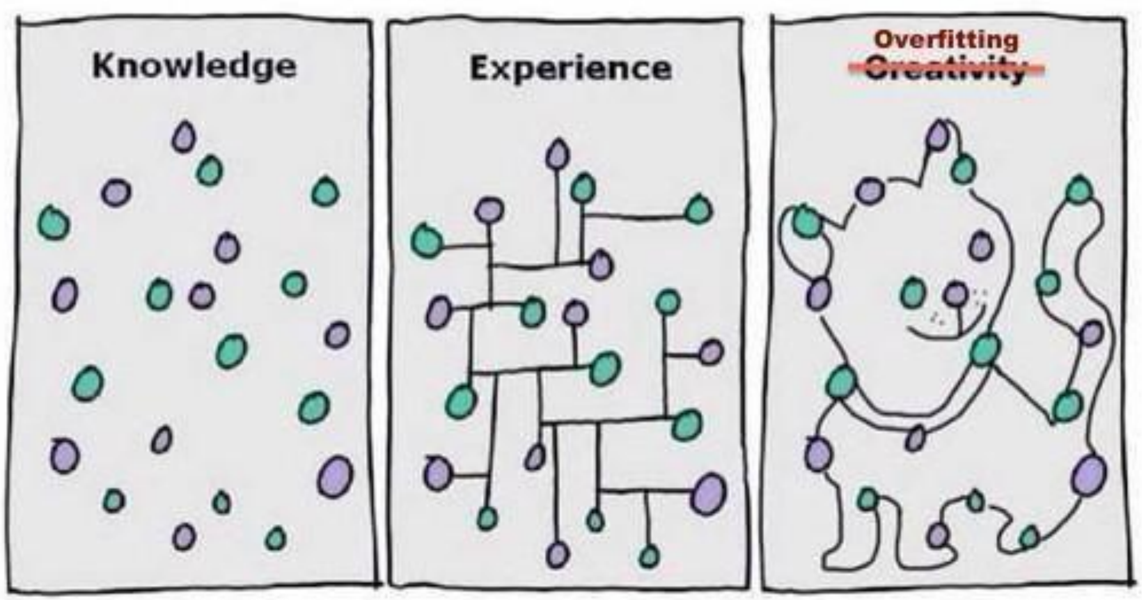
- Which is the **best model** or category of models for a **specific learning task**?
- Deep Learning? Not obvious in the context of Network traffic Monitoring and Analysis (NMA)
- Our claim: "***multiple-eyes principle***" → **ensemble learning** models
- We explore the application of **ensemble learning models** to multiple NMA problems…
- …following a particularly promising model known as the **Super Learner**

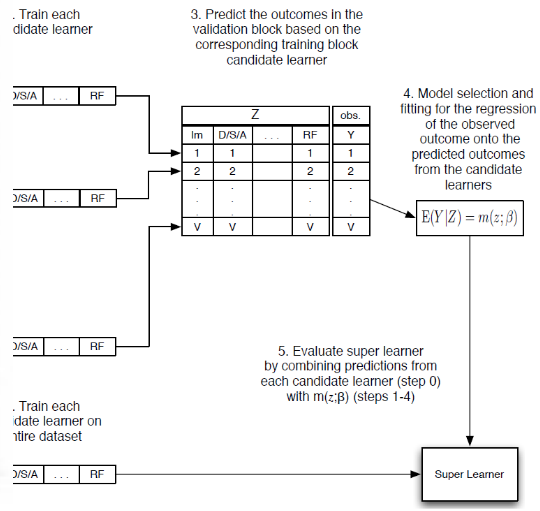
# Ensemble Learning for Network Security

**ensemble learning**: combine multiple (base) learning models to obtain better performance.

- If a set of base learners do not capture the true prediction function (the oracle), **ensembles** can give a **good approximation** to that **oracle function**.
- **Ensembles perform better** than the individual **base algorithms**.
- **Multiple approaches** to ensemble learning, including **bagging** (decrease variance), **boosting** (decrease bias), and **stacking** (improve predictive performance)

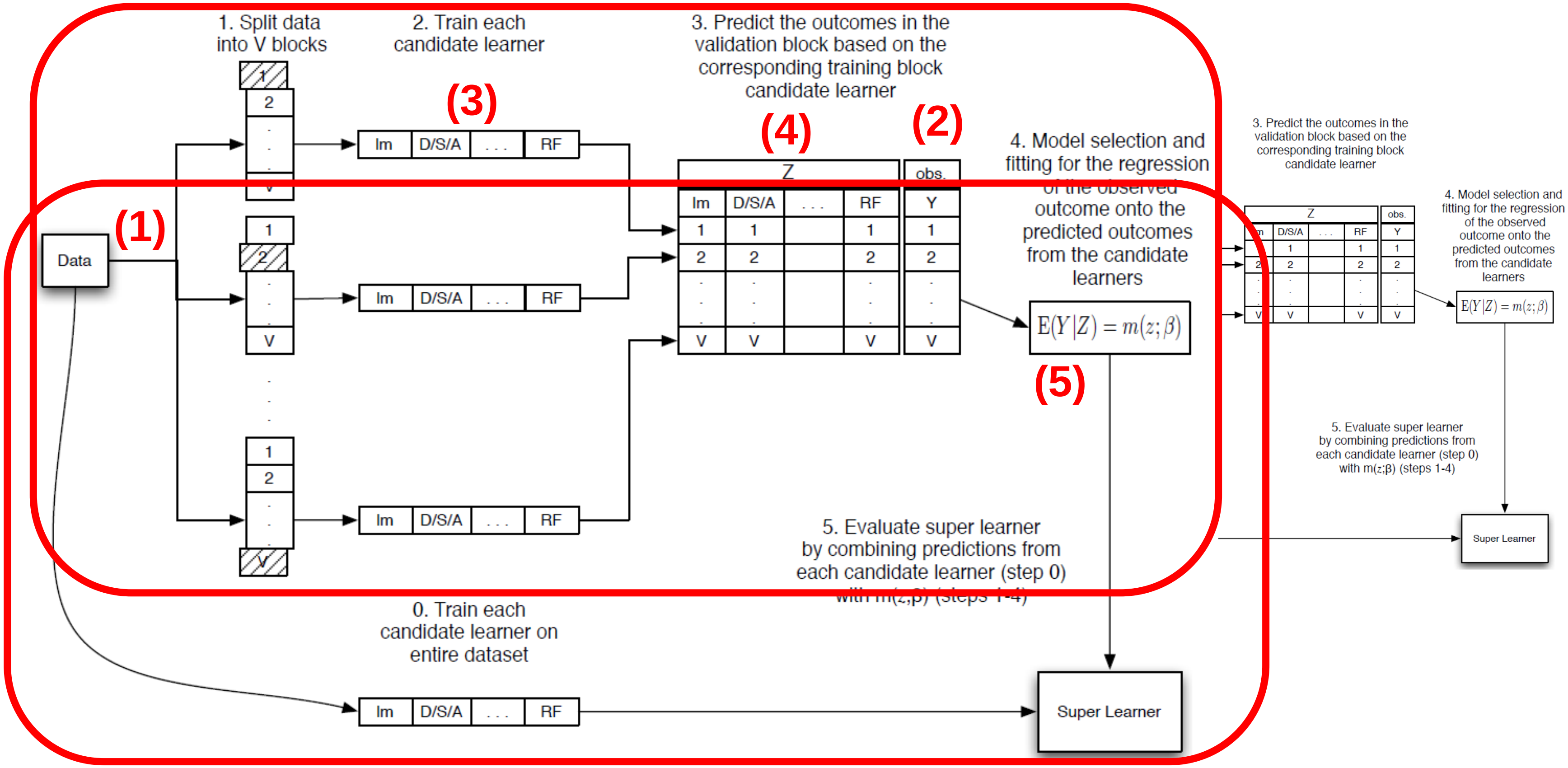
# Super Learner




- **General ensemble learning** approaches might be **prone to over-fitting**.
- **Super Learner** [Van der Laan'07]: **stacking** ensemble learning **meta-model** that **minimizes over-fitting likelihood** using a **variant of cross-validation**.
- Finds the **optimal combination of a collection of prediction algorithms** → performs asymptotically as well – or better, than any of the *base learners*.

# Super Learner – How Does it Work?




- **2-steps approach (training and validation of Super Learner):**

  1st ➔ *given a* ***(1) dataset X(k,l) with labels Y(k)*** ***(2)****, and a set of* ***(3) n base learners*** *(e.g., DTs, ANNs, SVM, etc.), build a* ***(4) new dataset S(Z(k,n), Y(k)) (by cross validation)*** *to* ***(5)*** *train the* ***Super Learner model m(z,β)***

  2nd ➔ *train each of the* ***n base learners*** *using training/validation split of (X,Y), and* ***compute predictions*** *(on top of validation set)* ***using meta-model m(z,β) (trained in step 1)***

# GML Learning for NMA

- The **Super Learner meta-model** could be **whatever algorithm**
- The **original work** [Van der Laan'07] uses a **simple minimum square linear regression model** as the example **Super Learner**.
- ***Problem***: how to **define weights** to **perform** properly in **every dataset**?
- ***GML Learning***: computes **weights** with an **exponential probability of success, reducing** the **influence** of **poor base learning models**.

$$m_{GML}(X) = \sum_{j=1}^{J} w_j h_j(X)$$

***base learners***

$$w_j = \frac{e^{\lambda \alpha_j}}{\sum_{i=1}^{J} e^{\lambda \alpha_i}}$$

***base learner accuracy***

***control variable:*** reduces weight for low accuracy predictors

# Models Benchmarking

**We compare several models for NMA:**

- We take **5 standard base learning models**: linear **SVM, CART, *k*-NN, ANN (MLP)** and **Naïve Bayes**

 We **build 4 different Super Learners:**

1. **Logistic regression** (binary output 0/1)
2. **Weighted Majority Voting (MV):**
    - **MVuniform:** same weight to each base learner
    - **MVaccuracy:** weights are computed using base learner accuracy
3. **Decision Tree meta-learner** (CART)

- **Boosting** (ensemble learning)**: AdaBoost tree**
- **Bagging** (ensemble learning): **Bagging tree** and **Random Forest**
- ***GML Learning***

# Multiple NMA Problems

**Five network measurement problems for model benchmarking:**

1. **NS –** ***detection of network attacks*** in **WIDE/MAWI traffic** (transpacific links)

2. **AD –** ***detection of smartphone-apps anomalies*** in cellular networks (data captured at core cellular network)

3. **QoE-P –** ***QoE prediction*** in cellular networks (data captured at smartphones)

4. **QoE-M –** ***QoE-modeling for video streaming*** (smartphones public datasets)

5. **PPC –** ***Internet-paths*** dynamics tracking – ***prediction of path changes*** (M-Lab traceroute measurements)

# (some) Evaluation Datasets

- We focus on two NMA problems:
  - Detection of **Network Attacks** in **WIDE/MAWI** network traffic
  - Detection of **App-related Anomalies** in an **Operational Cellular Network**

- **Synthetically generated dataset for AD in cellular networks**
  - **derived from real cellular ISP measurements** (traffic measurements collected during 6-months in 2014)
  - **Anomaly Templates**: derived ... the cellular traffic → in this pa...
  - **Evaluation labelled dataset**: 1... different anomaly instances o... intensity (number of involved...
  - **36 features** describing 10' tim...
  - These include FQDNs, DNS err... manufacturer (empirical distri...

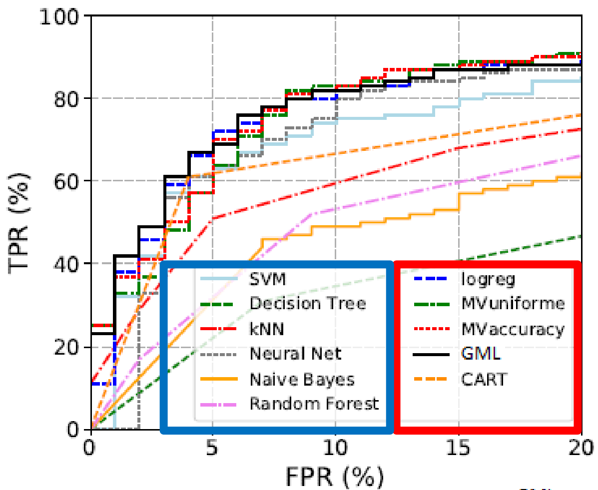
- **WIDE Network traffic using MAWI labels**
  - traffic traces captured daily on backbone link be...
  - **MAWI labels**: uses a combination of four traditi... label the collected traffic by ...
  - 5 attack classes: **DDoS, flashc...**
  - The dataset spans a **full week**... traces are split in **consecutive**...
  - **245 features** describe the traf...
  - These include throughput, pac... addresses and ports, transpor... (empirical distributions, samp... and more

| Field | Feature | Description |
|---|---|---|
| Tot. volume | # pkts | num. packets |
| | # bytes | num. bytes |
| PKT size | pkt_h | $H$(PKT) |
| | pkt_{min,avg,max,std} | min/max/std, $\overline{\text{PKT}}$ |
| | pkt_p{1,2,5,...95,97,99} | percentiles |
| IP Proto | # ip_protocols | num. diff. IP protocols |
| | ipp_h | $H$(IPP) |
| | ipp_{min,avg,max,std} | min/max/std, $\overline{\text{IPP}}$ |

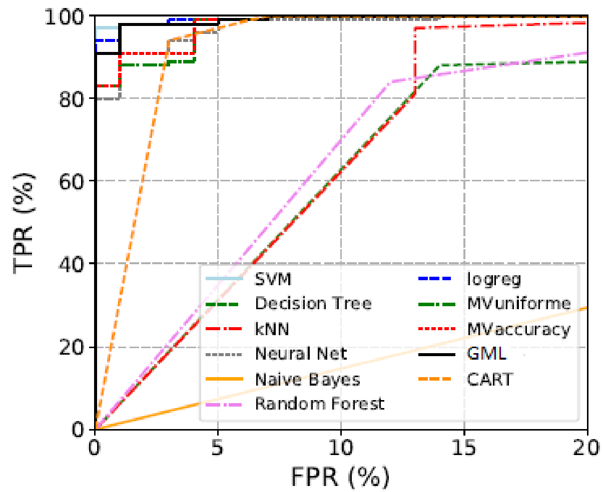
| Type | $E_1$ | $E_2$ | $E_3$ |
|---|---|---|---|
| Start time $t_1$ | 9:00 | 13:00 | 18:00 |
| Duration $d$ | 2h | 1 day | 1h |
| Involved devices $D$ | 10% | 5% | 3% |
| Back-off time | 5 sec | 180 sec | 20 sec |
| Manufacturer | single popular | multiple | multiple |
| OS | single | single | multiple |
| Error flag | +5% timeout | — | — |
| FQDN | top-2LD | top-2LD | top-2LD |

Table III
ANOMALOUS DNS TRAFFIC FEATURES FOR TYPES $E_1$, $E_2$, $E_3$.

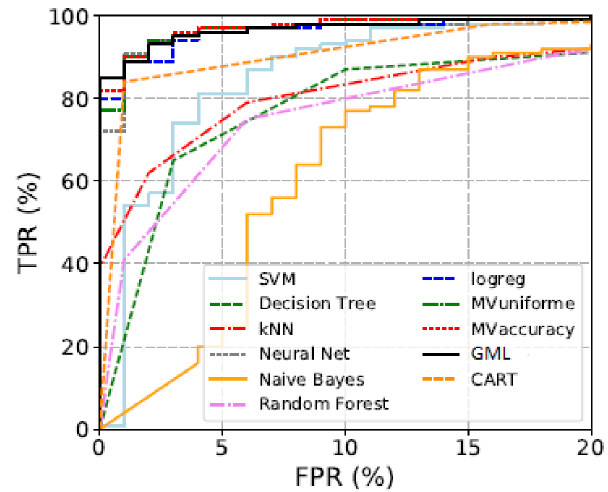
# Benchmark for Network Security




(a) DDoS.



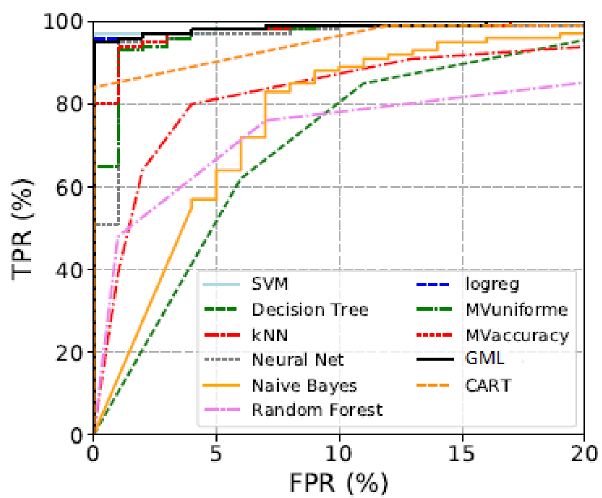

(b) HTTP Flashcrowd (MPTP-la).



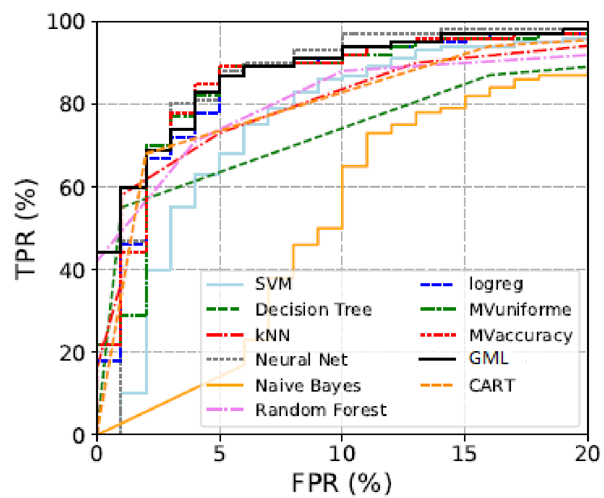

(c) Ping Flood.

***Base Learners*** ***Super Learners***

GML

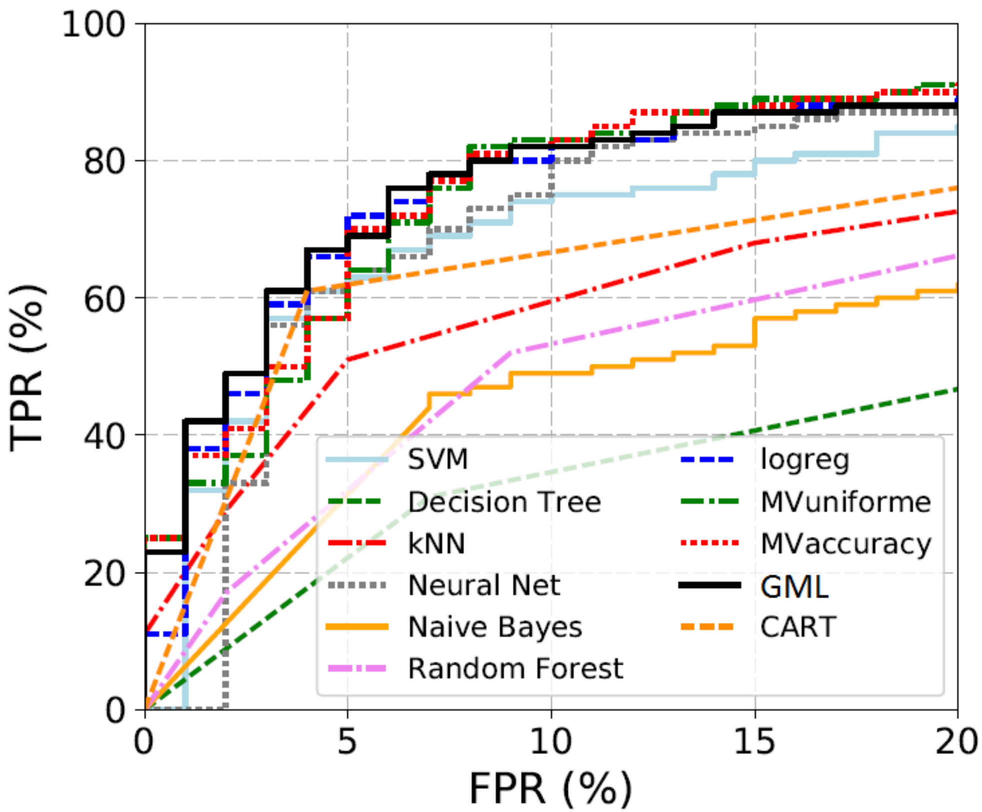



(d) Netscan UDP.




(e) Netscan TCP-ACK.

# Benchmark for Network Security




(a) DDoS.

- **Super Learners (SLs) outperform** both **base learners**, as well as the **RF model**
- The CART SL performs the worst → **regression-based models are more accurate for SL**
- **GML slightly outperforms other SLs**

# Benchmark for Network Security

| | DDoS |
|---|---|
| CART | 0.745 |
| Naïve Bayes | 0.730 |
| MLP | 0.907 |
| SVM | 0.883 |
| kNN | 0.720 |
| Random Forest | 0.827 |
| Bagging Tree | 0.823 |
| AdaBoost Tree | 0.892 |
| logreg | 0.926 |
| MVaccuracy | 0.924 |
| MVuniform | 0.923 |
| CART | 0.867 |
| GML | **0.935** |

- We take the **Area Under the ROC Curve** (AUC) as **benchmarking metric**
- **SLs performance increase is higher when base learners perform worse**
- Even if slightly, the **GML model systematically outperforms other models**

# Benchmark for Anomaly Detection

| | E1 | E2 | E3 |
|---|---|---|---|
| CART | 0.993 | 0.873 | 0.978 |
| Naïve Bayes | 0.956 | 0.861 | 0.959 |
| MLP | 0.997 | 0.944 | 0.996 |
| SVM | 0.996 | 0.944 | 0.995 |
| kNN | 0.995 | 0.859 | 0.963 |
| Random Forest | **0.999** | 0.876 | 0.993 |
| Bagging Tree | 0.996 | 0.885 | 0.983 |
| AdaBoost Tree | 0.998 | 0.945 | 0.995 |
| logreg | **0.999** | 0.952 | 0.996 |
| MVaccuracy | **0.999** | 0.948 | 0.996 |
| MVuniform | **0.999** | 0.945 | 0.996 |
| CART | 0.997 | 0.924 | 0.994 |
| GML | **0.999** | **0.963** | **0.997** |

- Similar observations are drawn from the AD benchmark
- **Anomalies E1 and E3 are easier to detect,** and **base learners** provide already **very accurate** results
- **E2 anomalies are stealthier** (long duration, small volume), and ***GML provides a clear performance increase***

# Full Benchmark in multiple NMA Problems

| | AD | NS | QoE-P | QoE-M | PPC | ALL |
|---|---|---|---|---|---|---|
| CART | 0.948 (**3.9%**) | 0.872 (**11.1%**) | 0.956 (**3.7%**) | 0.952 (**4.4%**) | 0.966 (**1.9%**) | 0.935 (**5.4%**) |
| Naïve Bayes | 0.925 (**6.2%**) | 0.826 (**15.8%**) | 0.752 (**24.2%**) | 0.754 (**24.3%**) | 0.924 (**6.3%**) | 0.819 (**17.1%**) |
| MLP | 0.979 (**0.7%**) | 0.970 (**1.1%**) | 0.887 (**10.7%**) | 0.882 (**11.5%**) | 0.964 (**2.1%**) | 0.929 (**6.0%**) |
| SVM | 0.978 (**0.8%**) | 0.955 (**2.6%**) | 0.786 (**20.8%**) | 0.790 (**20.7%**) | 0.886 (**10.1%**) | 0.869 (**12.1%**) |
| kNN | 0.939 (**4.8%**) | 0.892 (**9.1%**) | 0.788 (**20.6%**) | 0.793 (**20.4%**) | 0.920 (**6.7%**) | 0.854 (**13.6%**) |
| Random Forest | 0.956 (**3.1%**) | 0.903 (**7.9%**) | 0.983 (**1%**) | 0.978 (**1.8%**) | 0.969 (**1.6%**) | 0.957 (**3.2%**) |
| Bagging Tree | 0.954 (**3.2%**) | 0.895 (**8.7%**) | 0.976 (**1.7%**) | 0.975 (**2.1%**) | 0.973 (**1.3%**) | 0.953 (**3.6%**) |
| AdaBoost Tree | 0.979 (**0.7%**) | 0.930 (**5.2%**) | 0.982 (**1.1%**) | 0.984 (**1.2%**) | 0.875 (**11.2%**) | 0.954 (**3.5%**) |
| logreg | 0.982 (**0.4%**) | 0.960 (**2.1%**) | 0.981 (**1.1%**) | 0.978 (**1.9%**) | 0.941 (**4.5%**) | 0.970 (**1.9%**) |
| MVaccuracy | 0.981 (**0.5%**) | 0.974 (**0.7%**) | 0.984 (**0.9%**) | 0.991 (**0.6%**) | 0.972 (**1.3%**) | 0.981 (**0.8%**) |
| MVuniform | 0.980 (**0.6%**) | 0.973 (**0.8%**) | 0.980 (**1.3%**) | 0.984 (**1.2%**) | 0.980 (**0.5%**) | 0.979 (**1.0%**) |
| CART | 0.971 (**1.5%**) | 0.946 (**3.6%**) | 0.956 (**3.6%**) | 0.960 (**3.6%**) | 0.968 (**1.8%**) | 0.959 (**3.0%**) |
| GML | **0.986** | **0.981** | **0.993** | **0.996** | **0.985** | **0.989** |

- ***GML*** does not only **outperforms the most accurate first level learners**…
- …but **also outperforms other ensemble-learning models** based on bagging, boosting and stacking
- The **GML model performs the best for all scenarios**, suggesting a potentially good approach to go for by default in similar NMA problems

# Organization of the Talk

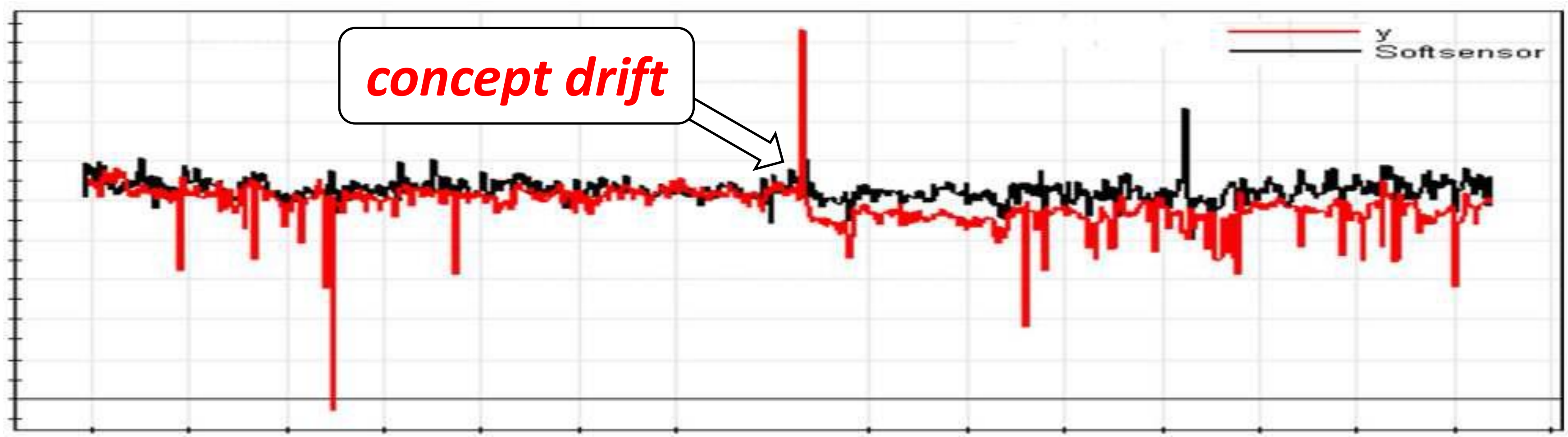
## Dealing with Some of these Challenges



- Deep Learning for Malware Detection – ***Avoid Feature Engineering***
- Explainable Artificial Intelligence (XAI) – ***Interpret Model Decisions***
- Super Learning for Network Security – ***Avoid Model Decision***
- **Adaptive/Stream Learning for NetSec – *Deal with Concept Drifts***

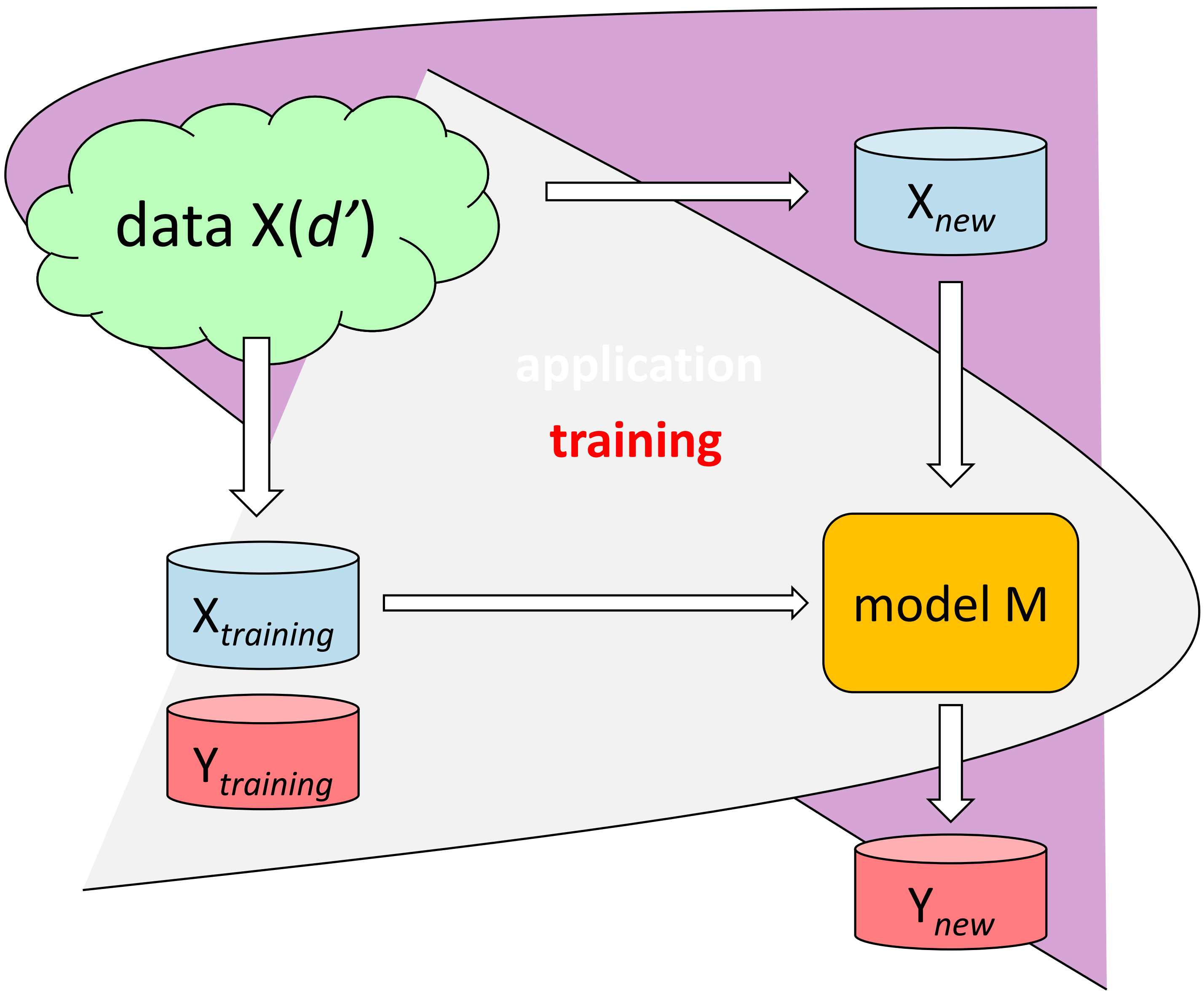
# Adaptive or Stream-based Learning (*credits to Albert Bifet*)

- Let us go a bit deeper into the problem of **concept drift in supervised learning**
- And overview the main principles **how to deal with concept drift**




- ***Concept Drift (non-stationarity)***: the **statistical properties** defining the relationships between **input data and output target** **change over time**.
- This causes problems because the **predictions become less accurate as time passes**.

# Concept Drift: a Trap for (off-line) Supervised Learning




**training**: learn a mapping function
$\mathbf{Y_{training} = M\,(X_{training})}$

**application**: use learnt function/model on newly, unseen data
$\mathbf{Y_{new} = M\,(X_{new})}$

…but what happens if/when $\mathbf{X_{new}}$ is **derived from a different distribution** $\boldsymbol{d' \neq d}$?

# (off-line) Supervised Learning under Concept Drifts

- Detection of **network attacks** in MAWI – WIDE network
- 10-fold cross-validation, **high detection performance with low FPR**…

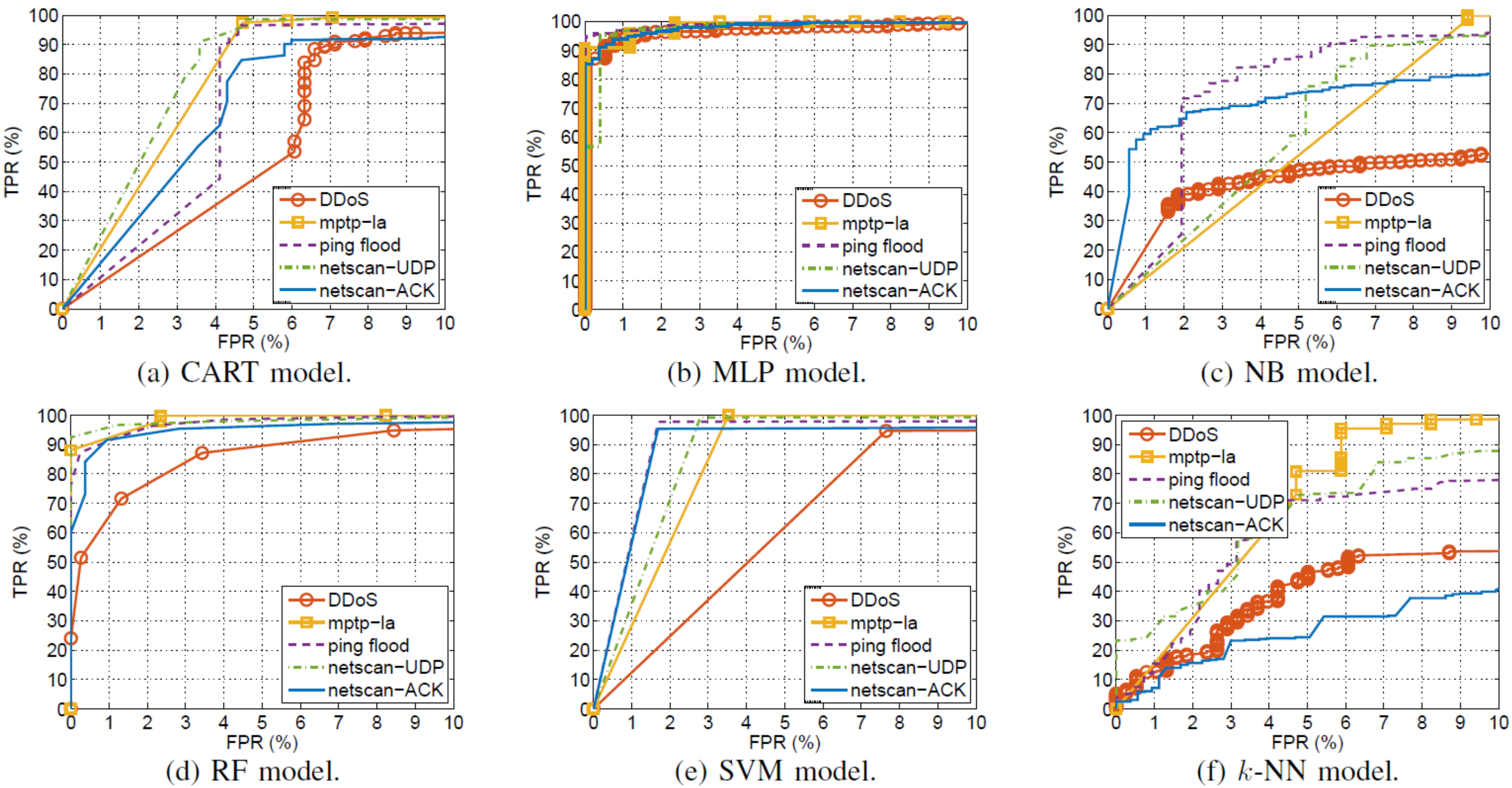
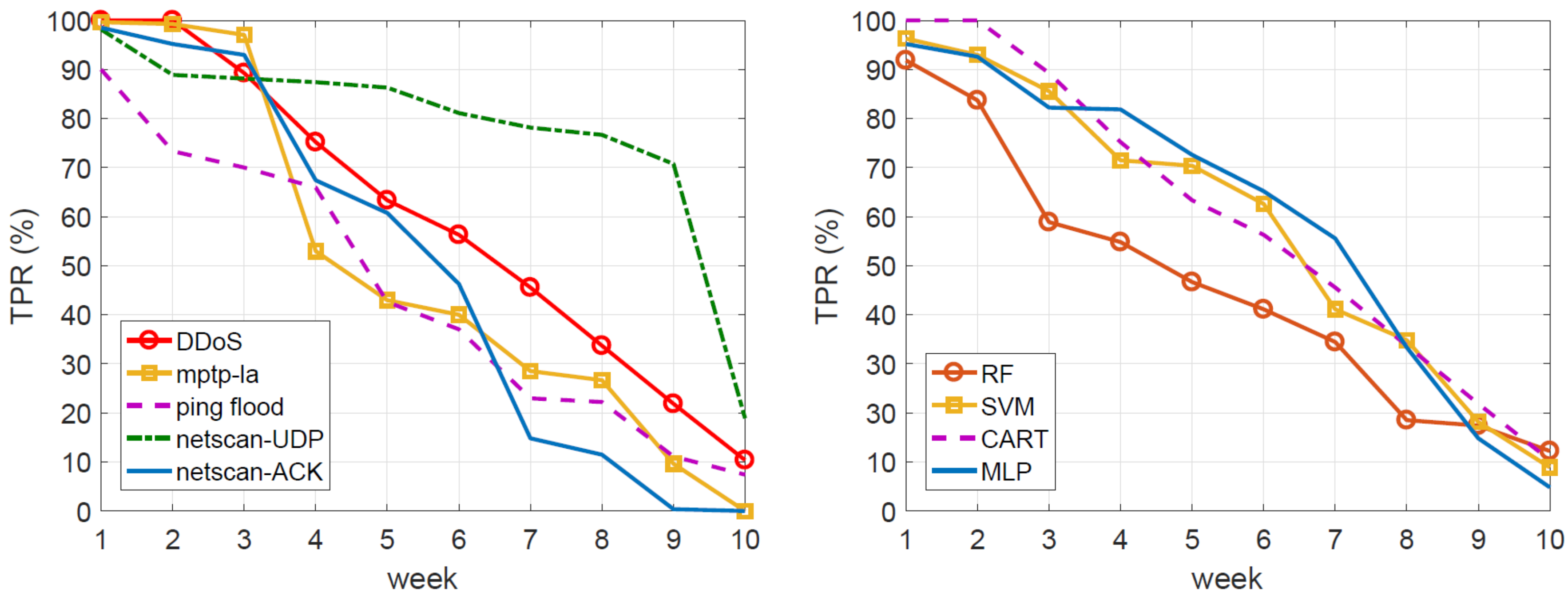



Figure 1: Detection performance (ROC curves) achieved by the different models for detection of network attacks.

# (off-line) Supervised Learning under Concept Drifts

- …accuracy remains high for the first 3 weeks (**training on first 3 days**)…
- …but **models accuracy start to rapidly degrade over time**

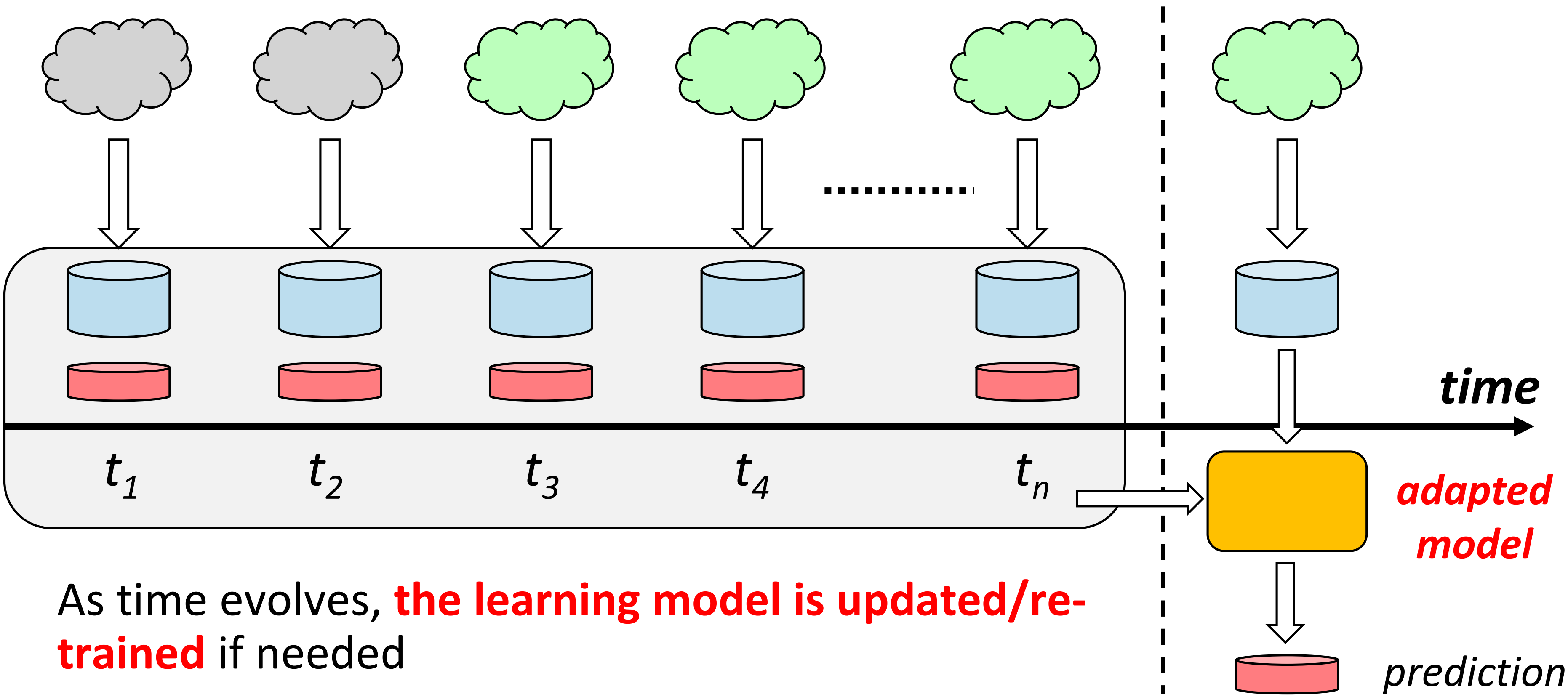



(a) CART model. (b) Detection of DDoS attacks.

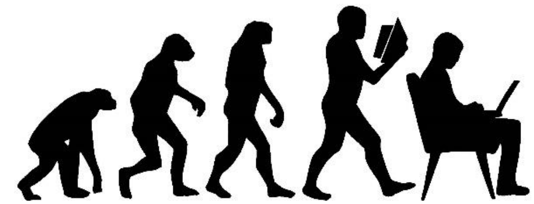
Figure 2: Performance drift for the off-line trained models along time. Training is done on the first 3 days of data.

# Learning in an Online Setting – Stream/Adaptive Learning

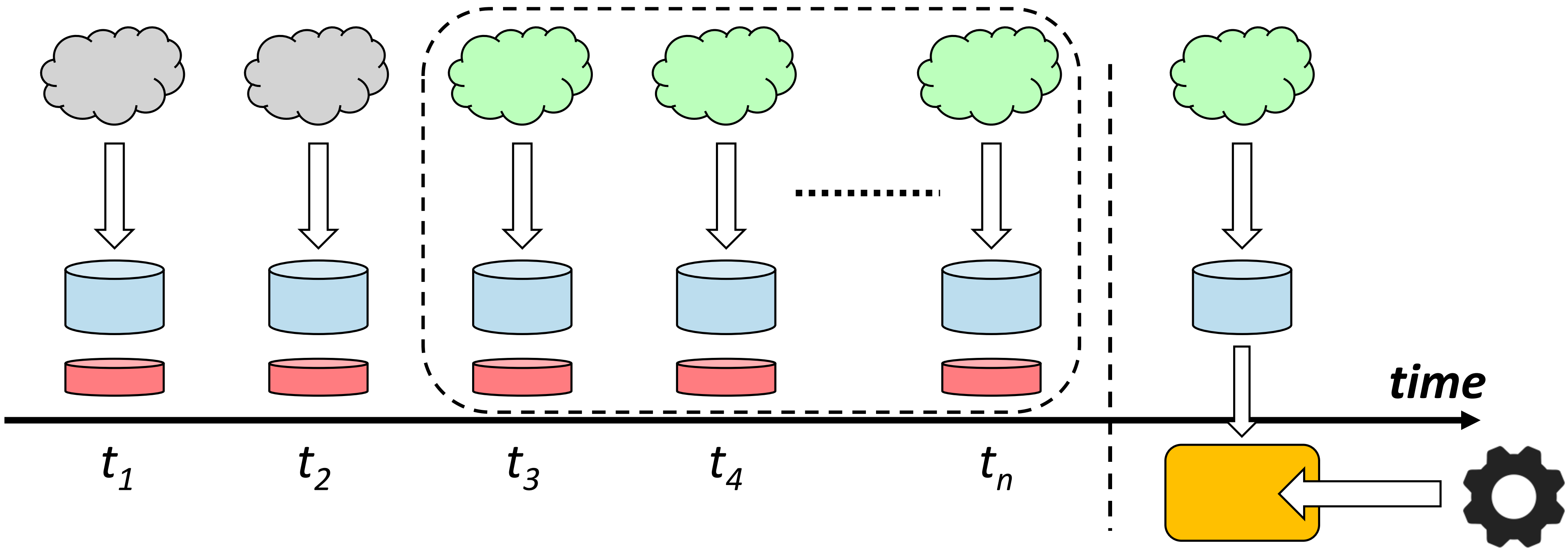
- In an **online setting**, **data** arrives continuously, as a **stream of samples**
- **Adaptive learning** consists of **learning from continuous data** in efficient way, using a **limited amount of memory**
- Adaptive learning approaches work in a **limited amount of time**



# Adaptation Strategies



- Two main approaches for adaptation:
  - **re-train the model** by carefully selecting the ***best*** data
  - **adjust** the previously learnt **model incrementally**



# Desired Properties of a System to Handle Concept Drift

- Adapt **fast** to concept drift

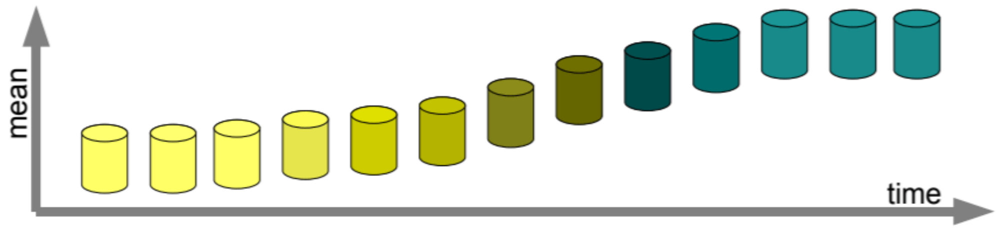
- **Robust** to noise, but **adaptive to changes**

- Capable to **deal with reoccurring contexts** (avoid catastrophic forgetting)

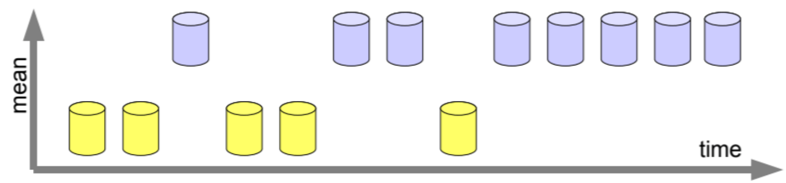
- **Use limited resources in terms of time and memory**

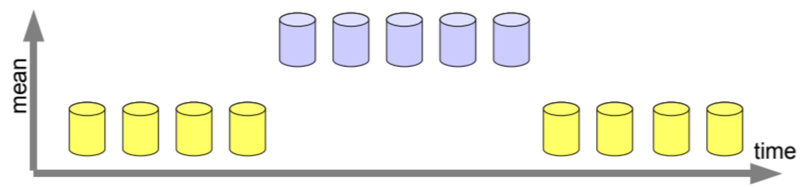
# What types of Concept Drift can we get?

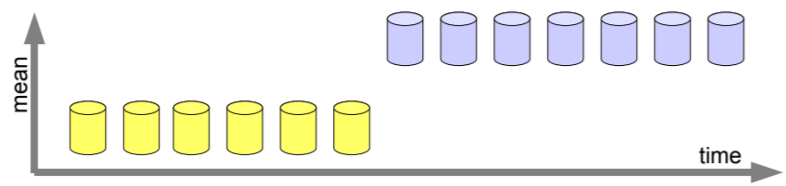
- The **change** to the data **could take any form**
- It is **conceptually easier** to consider the case where there is some **temporal consistency** to the change
- **Incremental drift:** one could assume that data collected within a specific time period show the same relationship and that this **changes smoothly over time**




- But of course, other types of changes may include:
  - A **gradual drift** over time
  - A **recurring** or cyclical **drift**
  - A sudden or **abrupt drift**



# Adaptation Strategies to Concept Drift

- A taxonomy of approaches (***A. Bifet, J. Gama***)

strategy

memory

change detection and follow up
triggering

adapt at every step
evolving

reactive forgetting
single model

ensemble
maintain memory

# Adaptation Strategies to Concept Drift

- A taxonomy of approaches (***A. Bifet, J. Gama***)

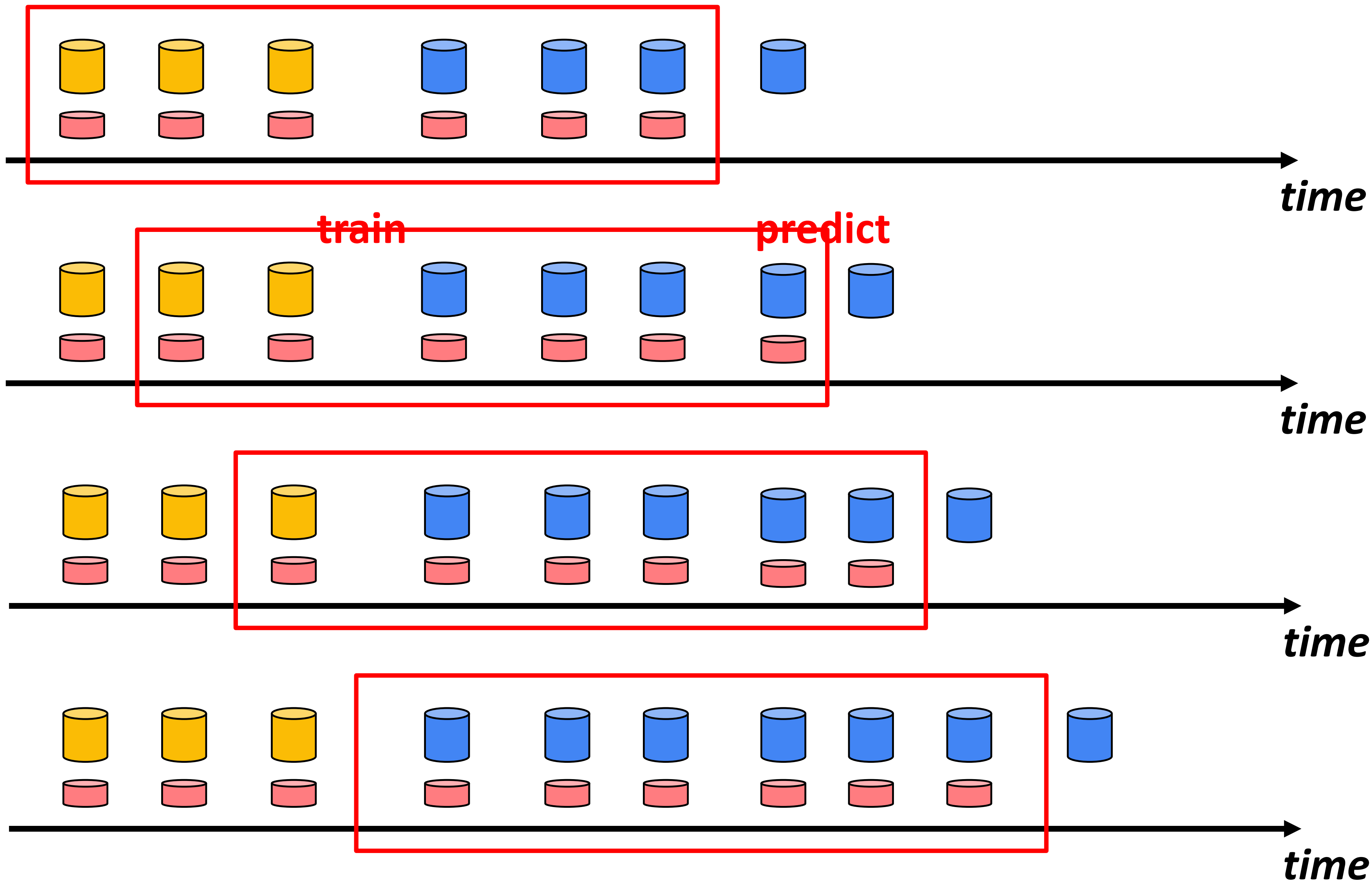
| memory \ strategy | triggering | evolving |
|---|---|---|
| single model | | |
| ensemble | | |

# Adaptation Strategies to Concept Drift

- A taxonomy of approaches (***A. Bifet, J. Gama***)

| memory \ strategy | triggering | evolving |
| --- | --- | --- |
| single model | | **forgetting**<br>• **forget old data**<br>• **re-train at fixed rate**<br>• **fixed windows**<br>• **instance weighting** |
| ensemble | | |

# Fixed-size Training Window

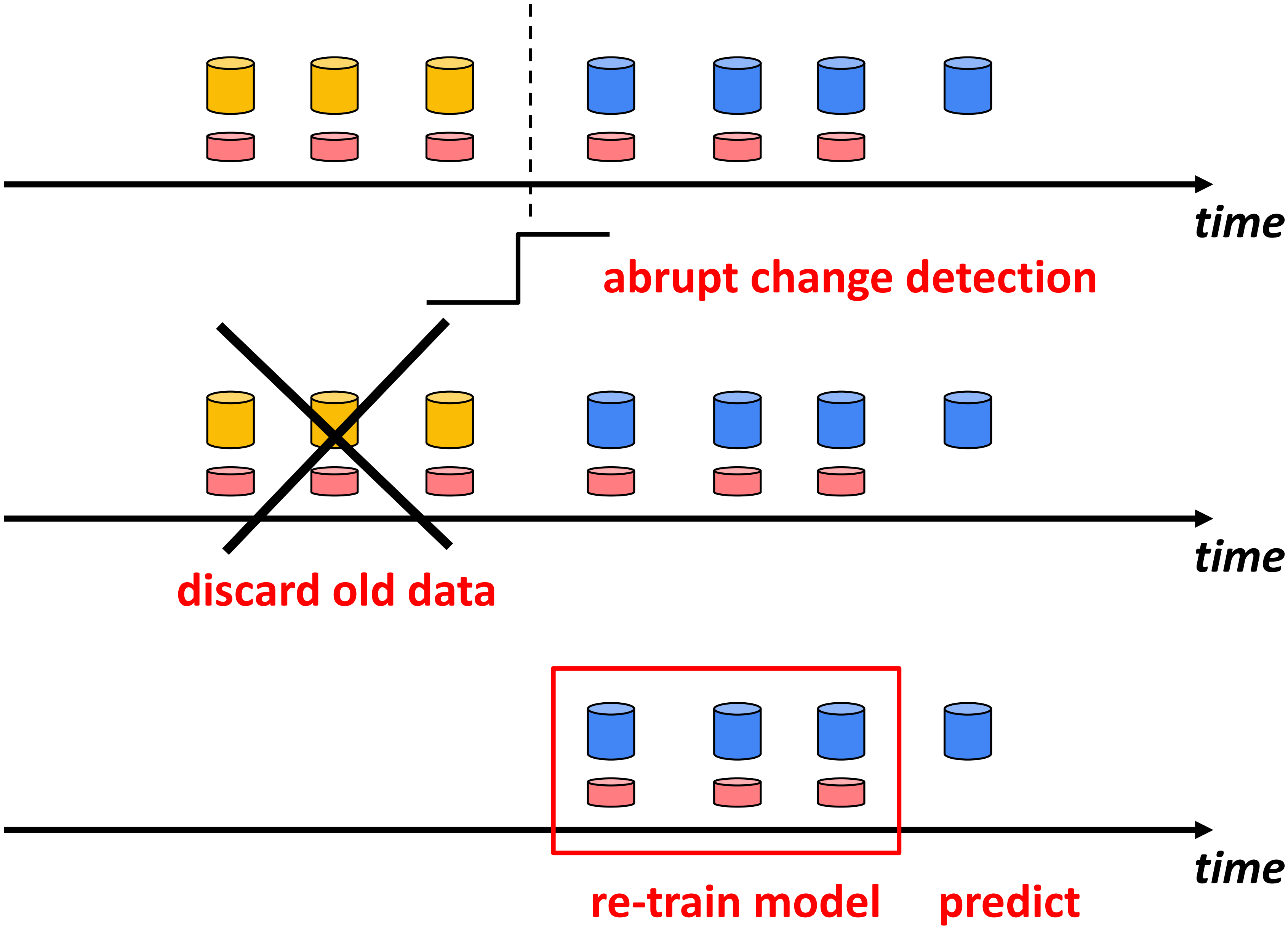


## Adaptation Strategies to Concept Drift

- A taxonomy of approaches (***A. Bifet, J. Gama***)

| memory \ strategy | triggering | evolving |
|---|---|---|
| single model | **detectors**<br>• **detect a change and discard the past**<br>• **variable windows** | |
| ensemble | | |

# Variable Training Window, Change Detection and Cut

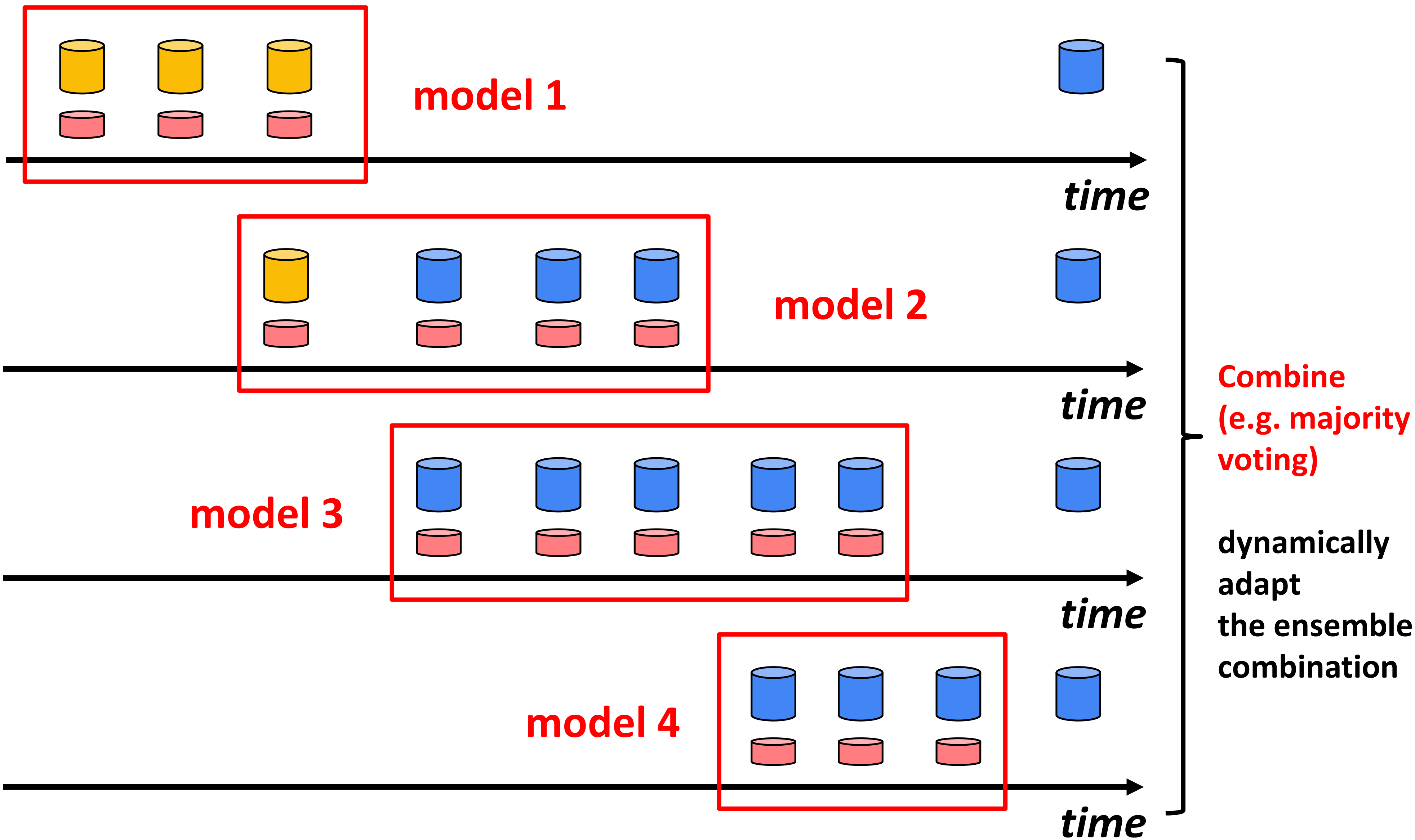


# Adaptation Strategies to Concept Drift

- A taxonomy of approaches (***A. Bifet, J. Gama***)

| strategy / memory | triggering | evolving |
| --- | --- | --- |
| single model | | |
| ensemble | | **dynamic ensemble**<br>• build many models<br>• dynamically combine<br>• dynamic combination rules |

# Dynamic Ensemble Learning

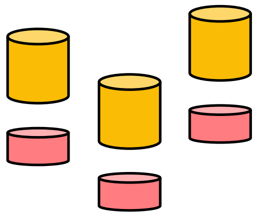


# Adaptation Strategies to Concept Drift

- A taxonomy of approaches (***A. Bifet, J. Gama***)

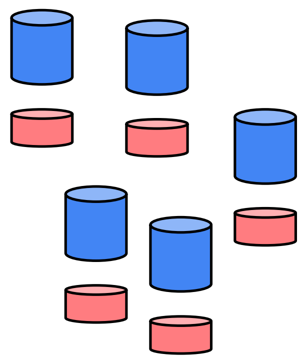
| memory \ strategy | triggering | evolving |
|---|---|---|
| single model | | |
| ensemble | **contextual**<br>• **build many models**<br>• **switch among them based on input**<br>• **meta–learning** | |

# Contextual (Meta) Approaches

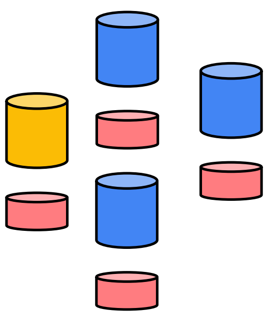


*set 1 → model 1*

**partition training data to build multiple models**

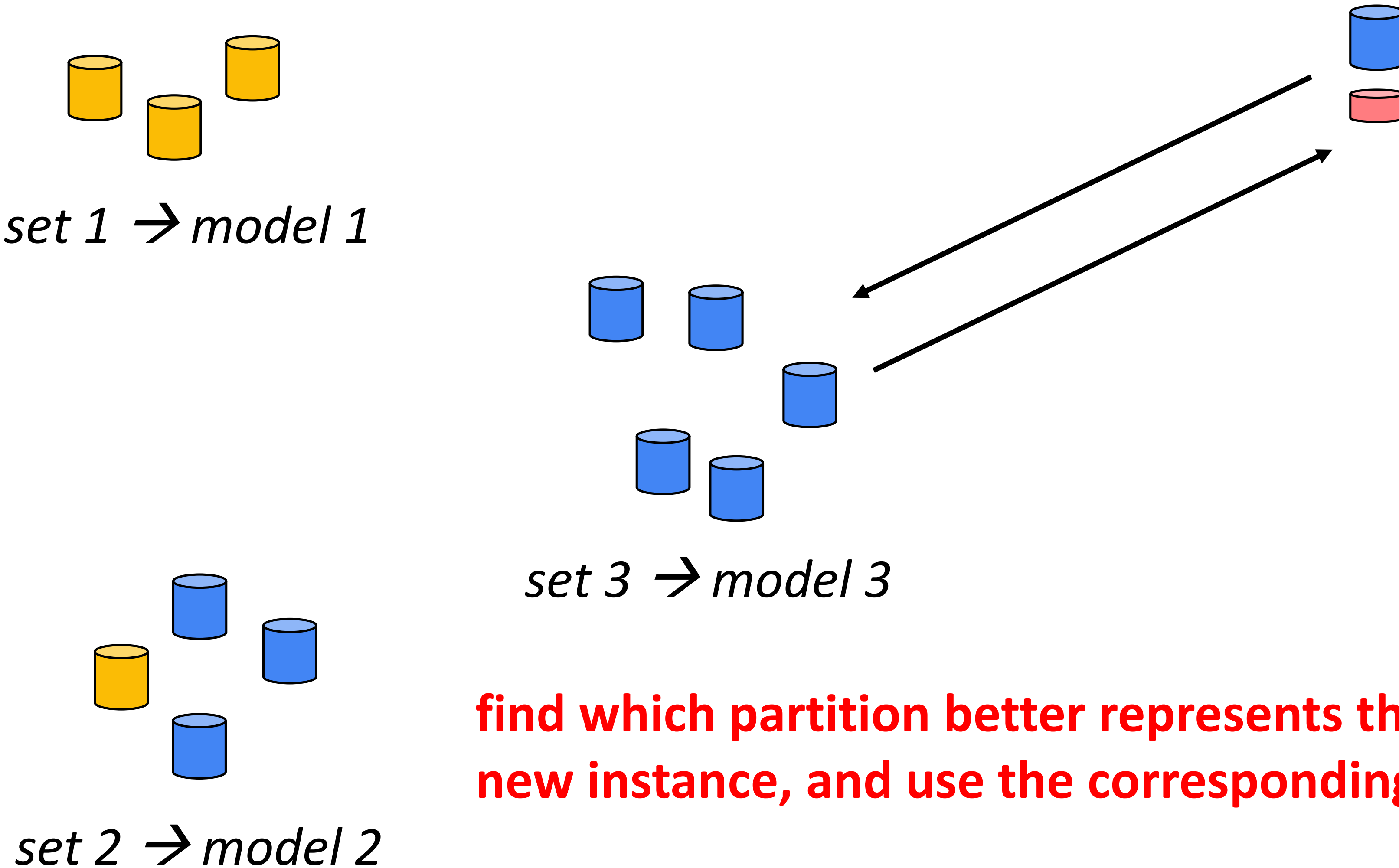


*set 3 → model 3*



*set 2 → model 2*

# Contextual (Meta) Approaches




**find which partition better represents the new instance, and use the corresponding model**

# Adaptation Strategies to Concept Drift

- A taxonomy of approaches (***A. Bifet, J. Gama***)

| memory \ strategy | triggering | evolving |
| --- | --- | --- |
| single model | **detectors**<br>• detect a change and discard the past<br>• variable windows | **forgetting**<br>• forget old data<br>• re-train at fixed rate<br>• fixed windows<br>• instance weighting |
| ensemble | **contextual**<br>• build many models<br>• switch among them based on input<br>• meta–learning | **dynamic ensemble**<br>• build many models<br>• dynamically combine<br>• dynamic combination rules |

# Adaptation Strategies to Concept Drift

- A taxonomy of approaches (***A. Bifet, J. Gama***)

| strategy / memory | triggering | evolving |
| --- | --- | --- |
| single model | **detectors**<br>**abrupt drift** | **forgetting**<br>**abrupt drift** |
| ensemble | **contextual**<br>**recurring drift** | **dynamic ensemble**<br>**gradual drift** |

# Adaptive/Stream Learning Models for NetSec

- Implement an adaptive approach using ***single models*** and a ***change-detection*** algorithm to detect concept drifts

- Take ADWIN (**Adaptive WINdowing**) to detect changes

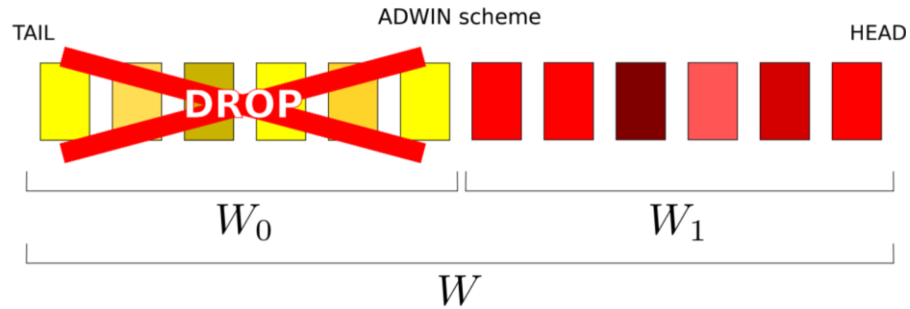
- ADWIN automatically **grows the learning window** when **no change** is apparent, and ***shrinks it when concept drifts*** are detected

- **Properties:** automatically adjusts its **window size** to the **optimum** balance point **between reaction time** and **small variance**

# Adaptive WINdowing algorithm

The idea of ADWIN is straightforward:

- it keeps a sliding window W with the most recently observed data
- whenever two ***large enough*** sub-windows of W exhibit ***distinct enough*** averages, **the older portion of the window is dropped**.




initialize window $W$

**for** each $t > 0$ **do**

  $W \leftarrow W \cup \{x_t\}$ (add $x_t$ to the head of $W$)

  **repeat**drop instances from the tail of $W$

  **until** $\|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}\| \geq \epsilon$ for every split of $W = W_0 \cdot W_1$

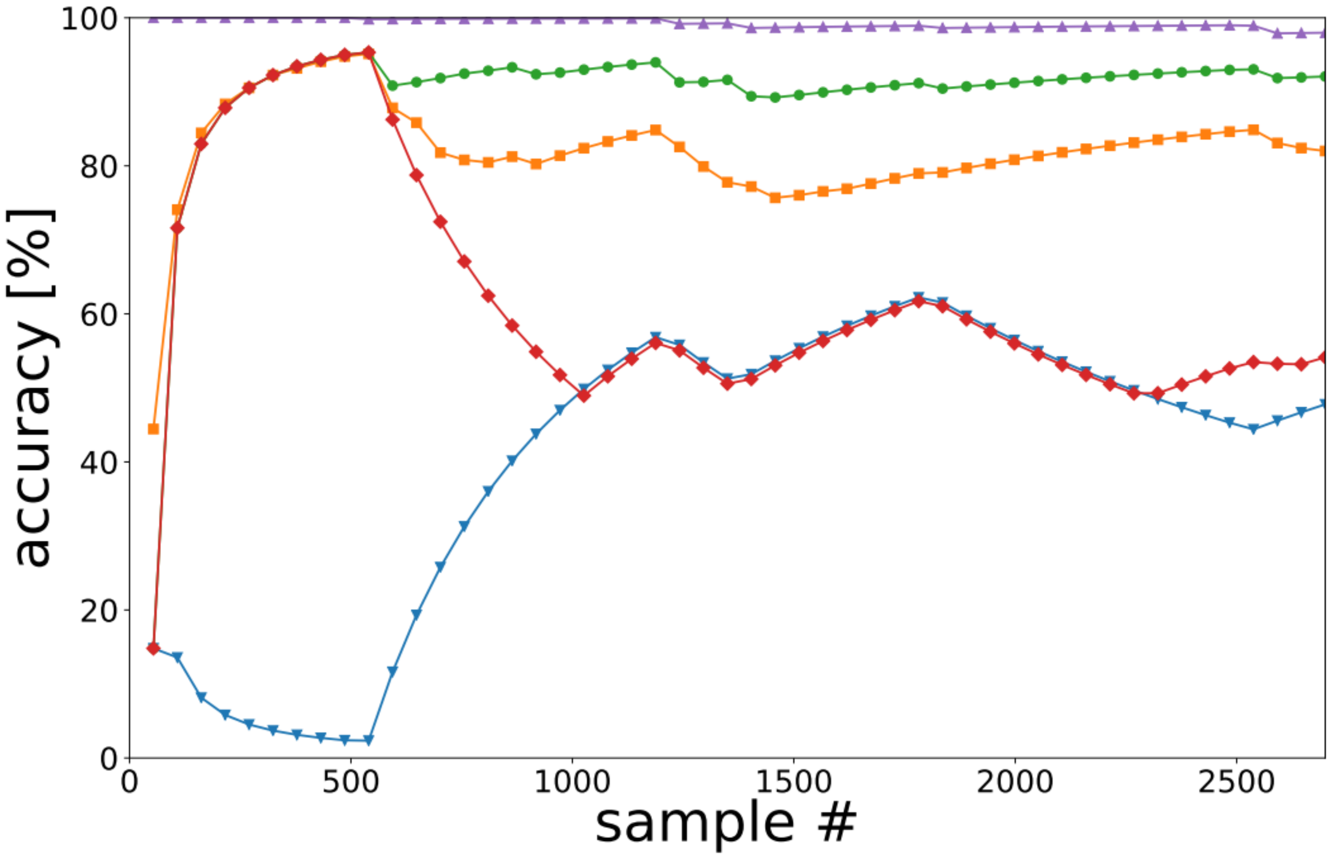
  return $\hat{\mu}_W$

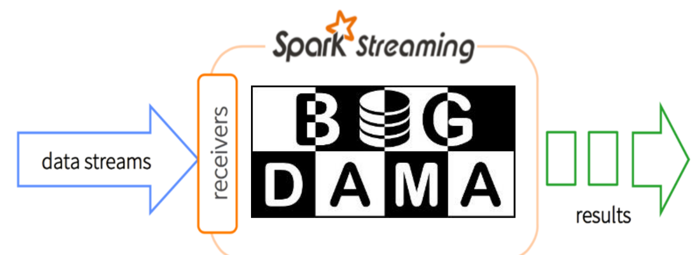
**end for**

where $\hat{\mu}_{W_0}$ and $\hat{\mu}_{W_1}$ are the averages of the instances in $W_0$ and $W_1$ respectively.

# Adaptive/Stream Learning Models for NetSec

- Adaptive learning algorithms **trained on labelled data, using ADWIN**



# Stream-based Learning Models Performance

- Multiple stream machine learning models, using ADWIN
- Detection accuracy, ***normalized to batch-based algorithms*** performance

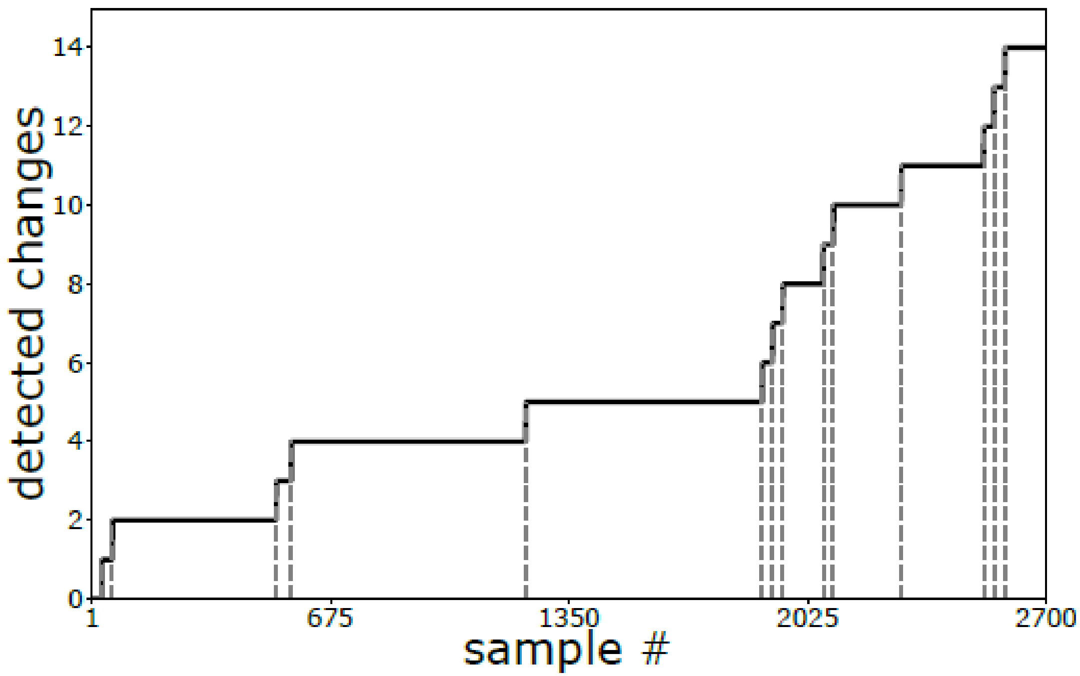
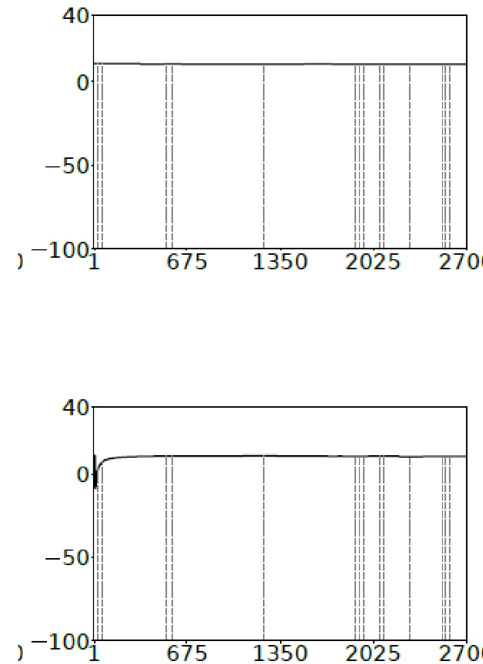


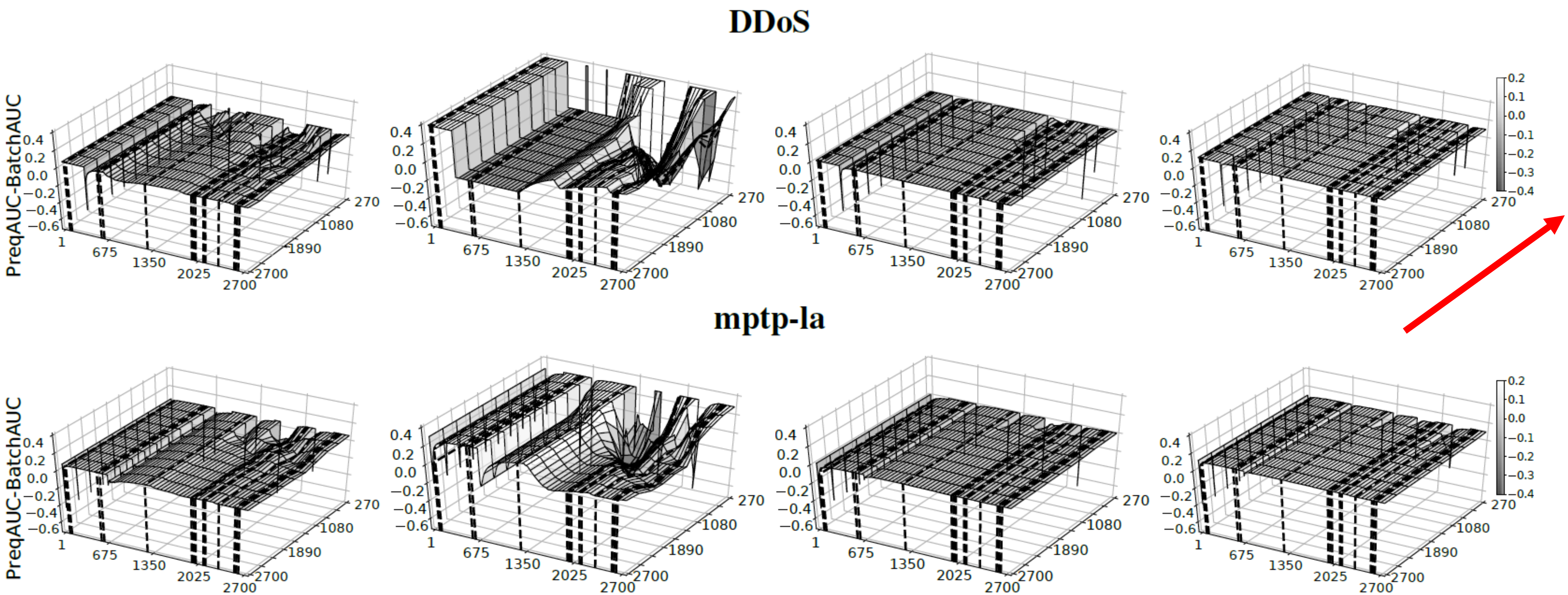

**Figure 1:** ***Page-Hinkley Concept Drift Detection.*** **Changes in the dataset distribution detected by the Page-Hinkley test. Detected changes are marked with dashed lines.**




(d) *SGD*

# Stream-based Learning Models Performance

- Multiple stream machine learning models, using ***fixed windowing***
- AUC (ROC curve), ***normalized to batch-based algorithms*** performance
- **Different window sizes tested**



(a) *k-NN* (b) *HAT* (c) *ARF* (d) *SGD*

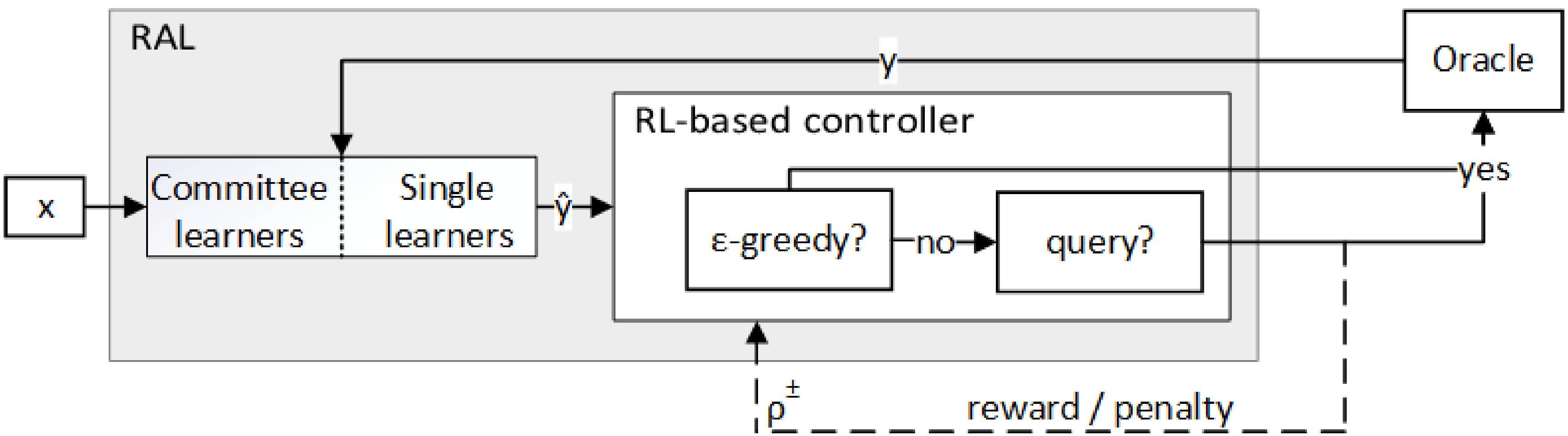
# Improving Stream-based Active Learning by Reinforcement (RAL)

- How do we deal with the **limited amount of labeled data?**
- **Active Learning (AL):** aims at labelling only the most informative samples
- **AL** can be applied to the **streaming scenario**, to complement previous approaches and **reduce the amount of labeled data**


- RAL – improves stream-based AL by Reinforcement Learning (RL)
  - AL bases its decisions based EXCLUSUVELY on model uncertainty
  - **RAL permits to additionally learn in a feedback loop**, based on the **effectiveness of the requested labels**
  - **Reward** in case asking oracle was informative (models would have predicted wrong label)
  - **Penalty** otherwise

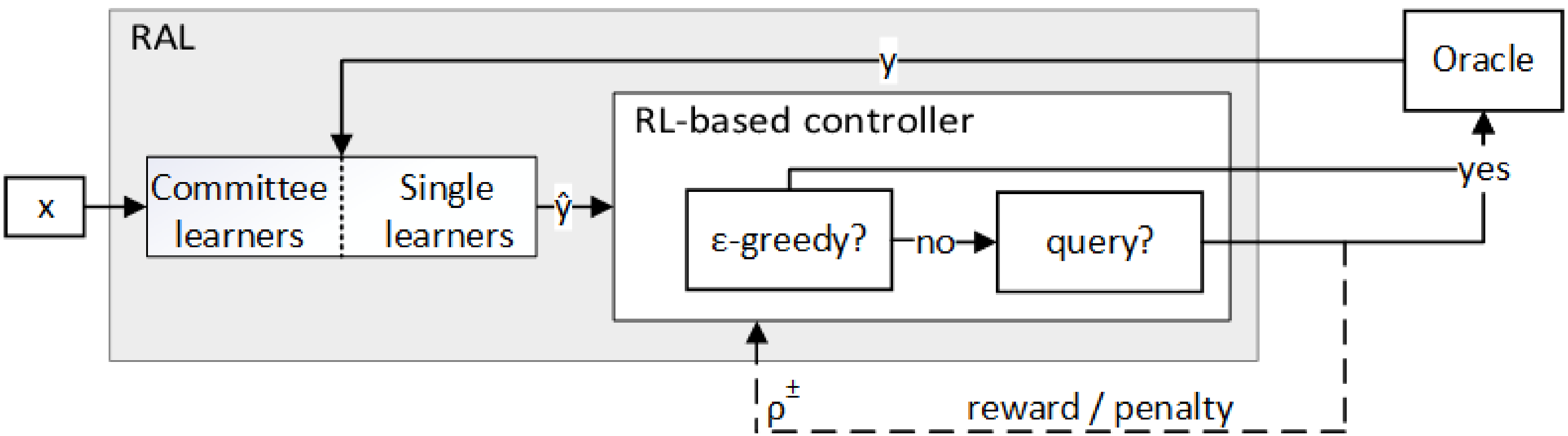
# RAL Principles and Components

- RAL is based on an ensemble of models
- RAL makes use of **contextual-bandit algorithms** (EXP4) to tune the decision powers of the different models depending on their behavior
- RAL uses a **ε-greedy approach** to handle concept drift and **improve the exploration/exploitation trade-off**



# RAL Principles and Components

- The **querying decision** (ask or not for a label) is taken **based on model prediction uncertainty** and a **threshold**
- Each algorithm in the ensemble (committee) gives its advice, based on its prediction uncertainty
- RAL takes into account the decisions of the members + their decision power
- Obtained **feedback influences the querying threshold**:
  - In case of **penalty**, the threshold decreases…..otherwise, it **slightly increases**



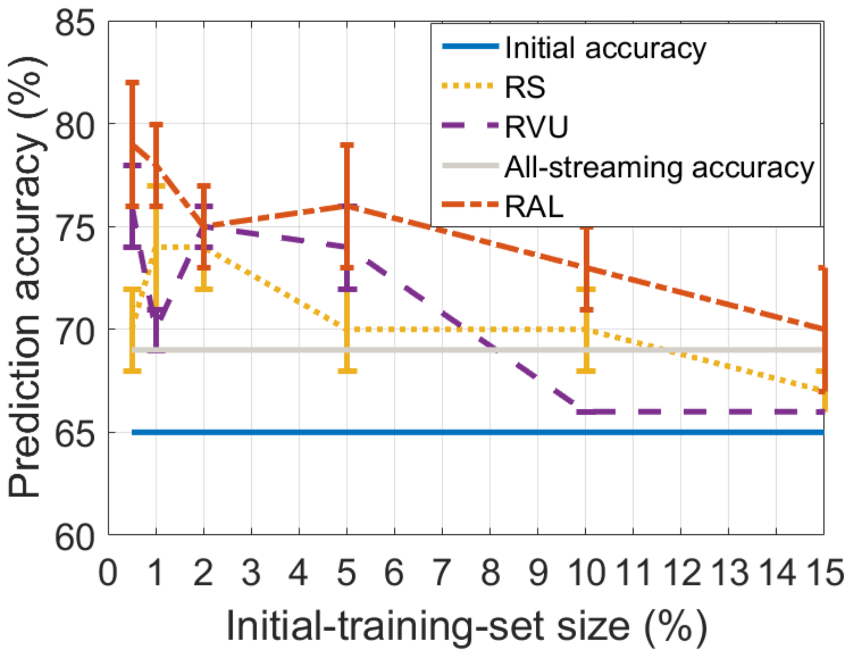
# RAL Evaluation vs. State of the Art

- **RAL vs** RVU (**Randomized Variable Uncertainty**) and simple **random sampling** (RS)

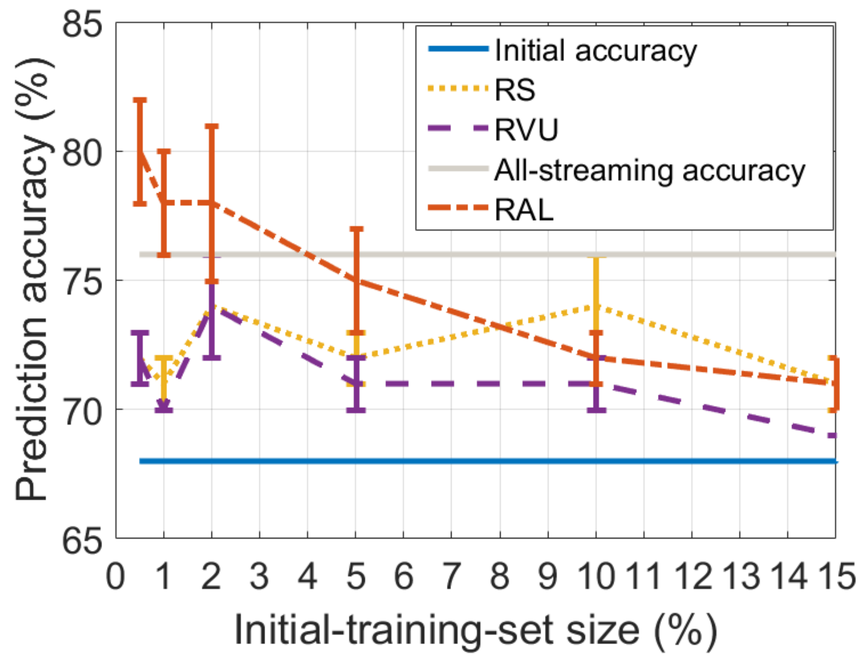
- Evaluation on data extracted from **MAWILab – in the wild network security**

- We divide each dataset into three consecutive parts:
  - **Initial training set** (variable size)
  - **Validation set** (last 30%), to evaluate the classifiers
  - **Streaming set** (remaining part of the dataset), for picking samples to learn from

# RAL Evaluation vs. State of the Art – Prediction Accuracy




Flood attack

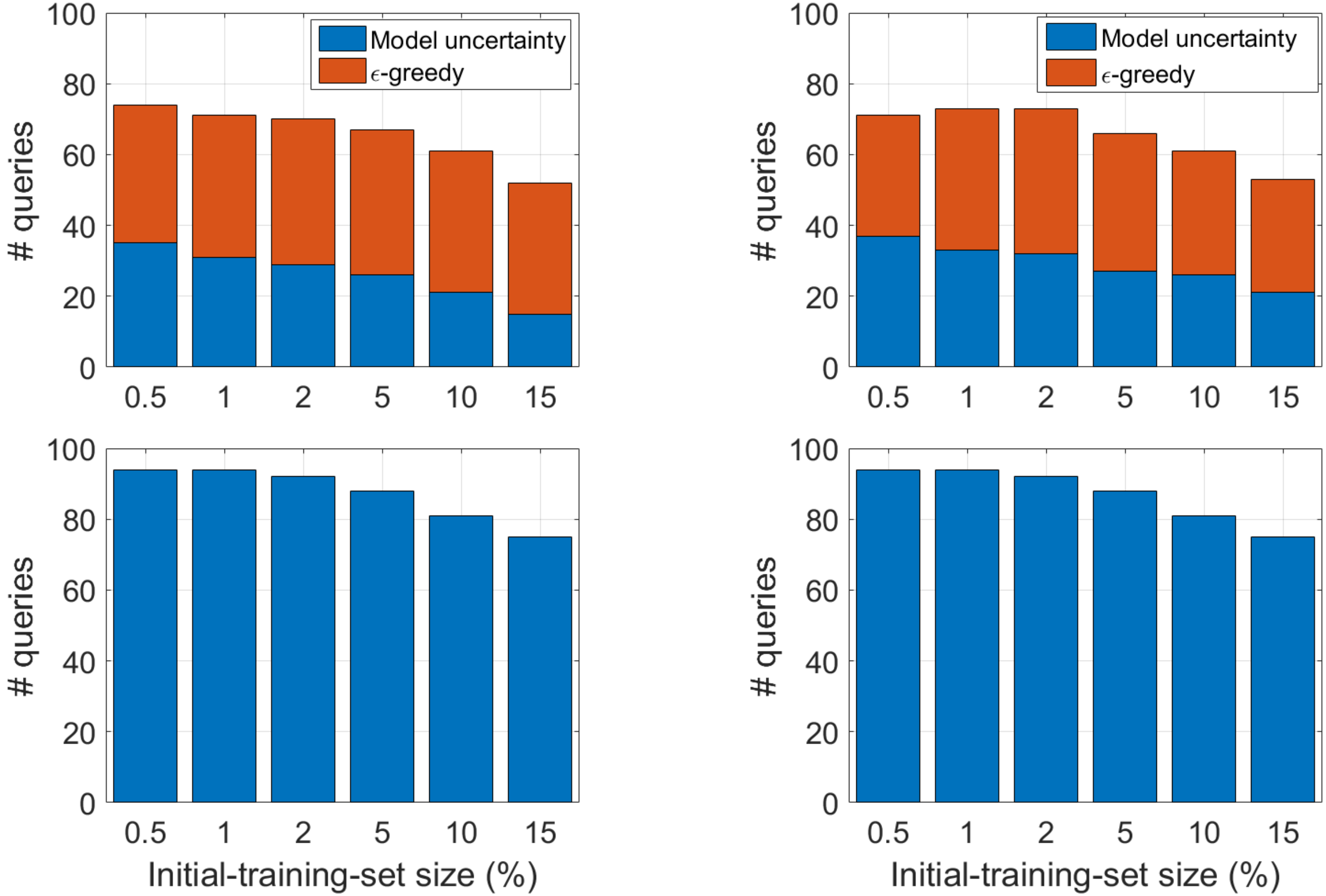



Netscan attack

# RAL Evaluation vs. State of the Art – Querying Cost

RAL

RVU




Flood attack

Netscan attack

# So What's Next?

- We're still far from ***making AI immediately applicable***
  - Limitations of learning process, data, models
  - Lack of generalization
  - Continual learning challenges – catastrophic forgetting and transfer
  - Lack of real knowledge generation – building simple mappings is *easy*
  - Portability of models to real deployments – *plug & play?*
- ***Effective Machine Learning*** – a mix of interesting challenges:
  - Transfer learning
  - Explainable AI (XAI)
  - Multi-task learning
  - Meta learning
  - Hierarchical learning
- And back right to the start: **the successful application of AI to network measurement problems is still on an early stage**

https://bigdama.ait.ac.at/

http://mobiqoe.ait.ac.at/

# Thanks

**Dr. Pedro Casas**

**Data Science & Artificial Intelligence**

**AIT Austrian Institute of Technology @Vienna**

*pedro.casas@ait.ac.at*

*http://pcasas.info*