Data Stream Mining Techniques for Security Applications

YUN SING KOH

ykoh@cs.auckland.ac.nz

https://www.cs.auckland.ac.nz/~yunsing/

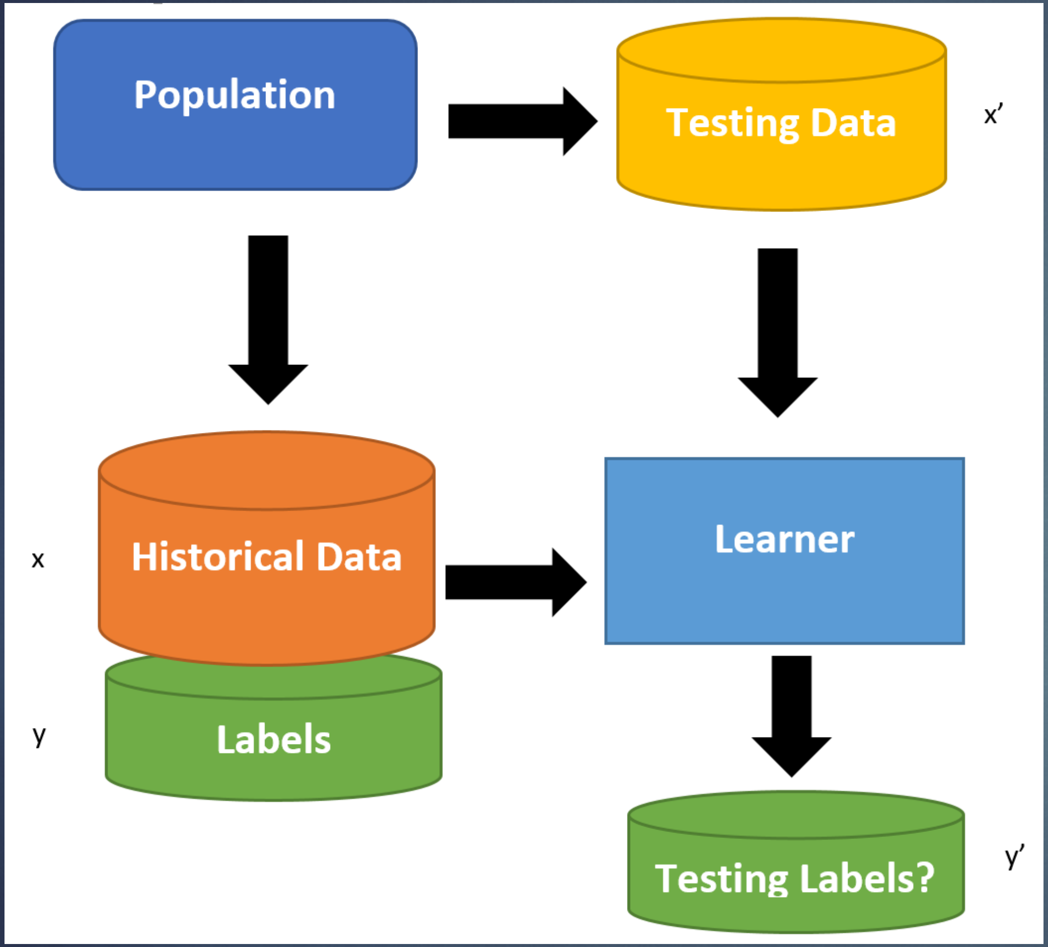
THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

Agenda

- Introduction to data streams
- Problem of Concept Drift
- Some of our solutions
- Concept Drift in Security Applications

Data Streams

- **Data Stream Mining** is the process of extracting knowledge structures from continuous data records.
- A data stream is an ordered sequence of instances that in many applications of data stream mining can be read only once or a small number of times using limited computing and storage capabilities.



Training:
Learning a mapping function

$y = \boldsymbol{f}\,(x)$

Application:
Applying ***f*** to unseen data

$y' = \boldsymbol{f}\,(x')$

Supervised Learning

Problem Area...

- Many data-stream analyses are dependent on speed for their value
  - For instance, in automated stock trading, slower decisions are very likely to gain less value than faster decisions
- Many data-stream analyses run in resource-constrained situations
  - For instance, smart sensors or weather monitoring stations may be reliant on battery power. Time- and memory-hungry analyses may be infeasible in these environments

Data Stream Properties

Properties

1. At high speed
2. Infinite
3. Can't store them all

4. Can't go back; or too slow
5. Evolving, non-stationary reality

What this means in an algorithmic sense?

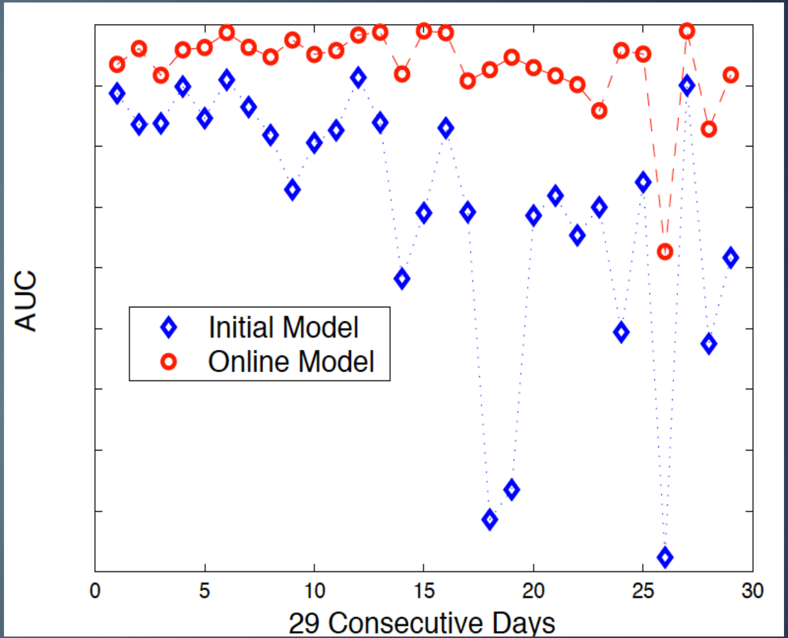
1. One pass
2. Low time per item - read, process, discard
3. Sublinear memory - only summaries or sketches
4. Anytime, real-time answers
5. The stream evolves over time

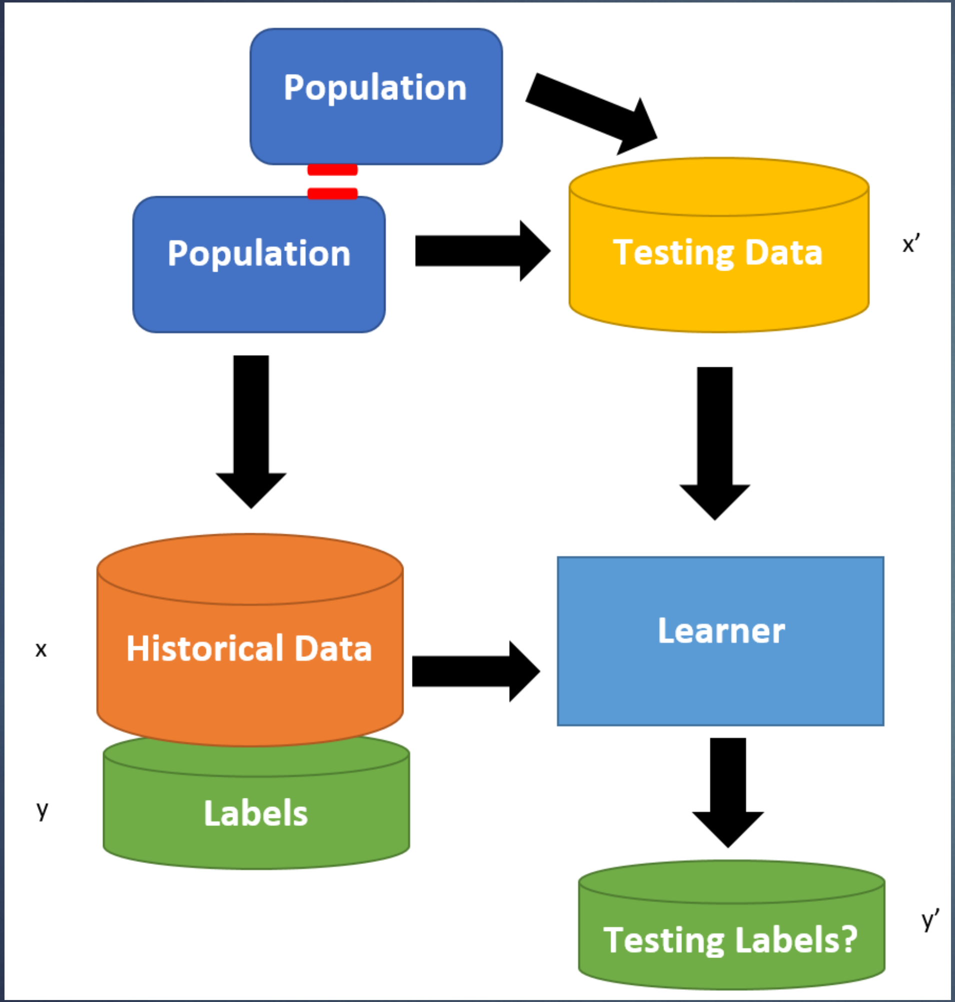
Predictive – Classification

x							
f(x)	zebra	penguin	zebra	penguin	zebra	zebra	?

Initial Model vs Online Model

- Need to retrain!
  - Things change over time
  - How often?
- Data unused until next update!
  - Value of data wasted





Training:

$y = f(x)$

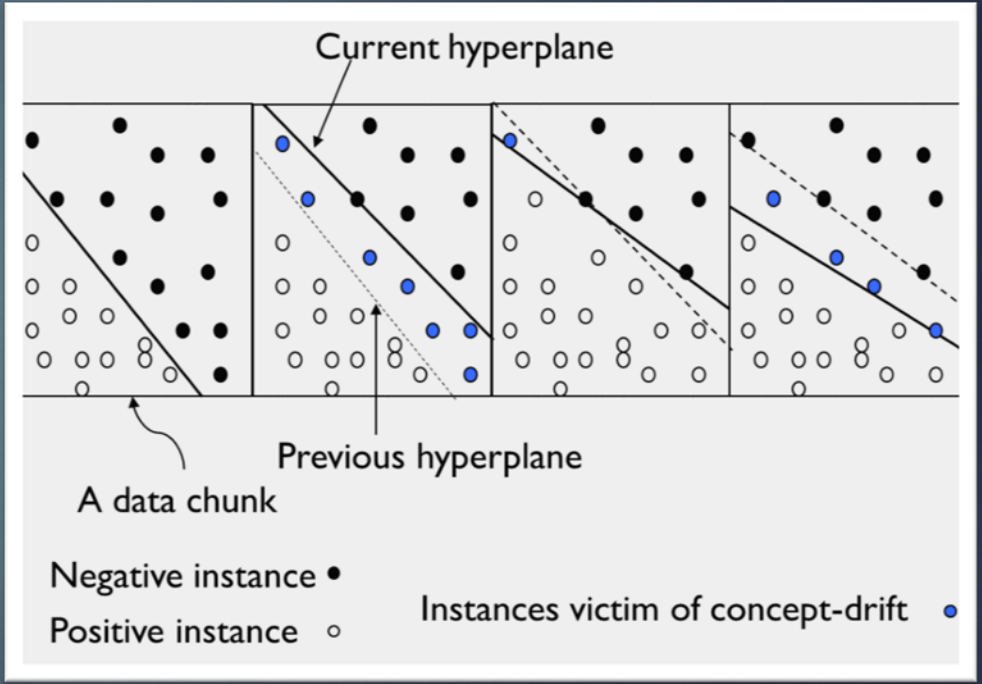
Application:

$y' = f??(x')$

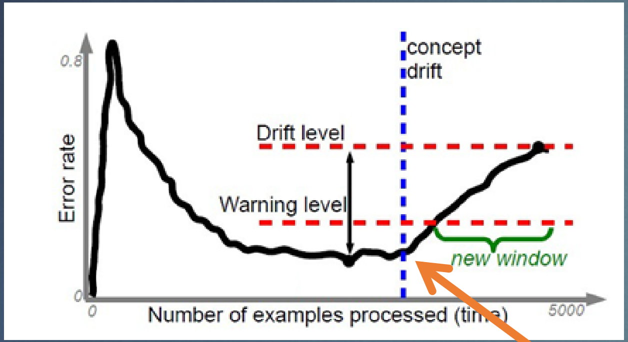
Learning with concept drift

Volume, Velocity, Variety & **Variability**

- data comes from complex environment, and it evolves over time.
- concept drift = underlying distribution of data is changing



Concept Drift & Error rates



- When there is a change in the class-distribution of the examples:
  - The actual model does not correspond any more to the actual distribution.
  - The error-rate increases
- Basic Idea:
  - Learning is a process.
  - Monitor the quality of the learning process:
    - Monitor the evolution of the error rate.

Adaptation Methods

- The ***adaptation model*** characterizes the changes in the decision model do adapt to the most recent examples.
- Blind Methods:
  - Methods that adapt the learner at regular intervals without considering whether changes have really occurred.
- Informed Methods:
  - Methods that only change the decision model after a change was detected. They are used in conjunction with a detection model.

Desired Properties of a System To Handle Concept Drift

- Adapt to concept drift asap
- Distinguish noise from changes
- Recognizing and reacting to reoccurring contexts
- Adapting with limited resources

Background - Concept Drift

Types of drift

1. Abrupt

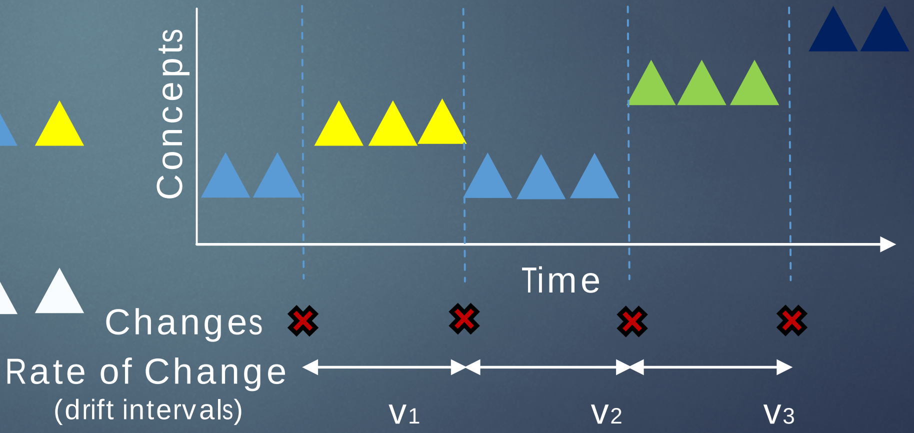
2. Gradual

3. Incremental

Drift Volatility

- Rate of concept change

Example



Concept Drift Detection Mechansim

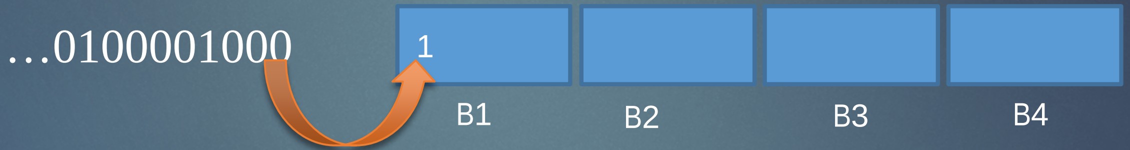
- Seed (Reactive Technique)
- ProSeed (Proactive Technique)
- CPF (Recurrent Concepts)

SEED Detector – Change Detector

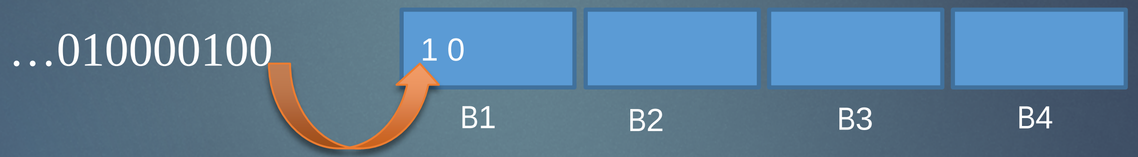
David Tse Jung Huang, Yun Sing Koh, Gillian Dobbie, Russel Pears: Detecting Volatility Shift in Data Streams. ICDM 2014
Part of MOA system:
https://github.com/Waikato/moa/blob/master/moa/src/main/java/moa/classifiers/core/driftdetection/SEEDChangeDetector.java

As each instance of the data (predictive error rates) arrives it is stored in a block B_i each block can store up to x number of instances.

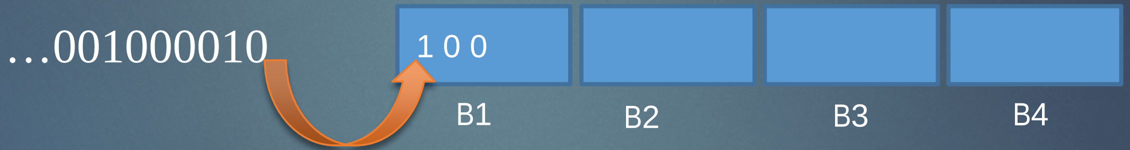
SEED Detector – Change Detector



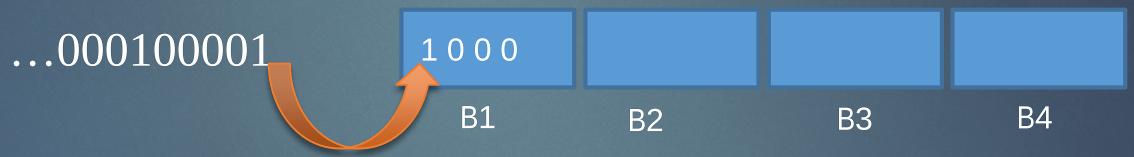
SEED Detector – Change Detector



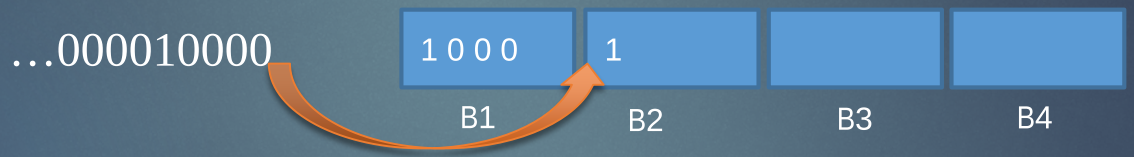
SEED Detector – Change Detector



SEED Detector – Change Detector



SEED Detector – Change Detector



SEED Detector – Change Detector

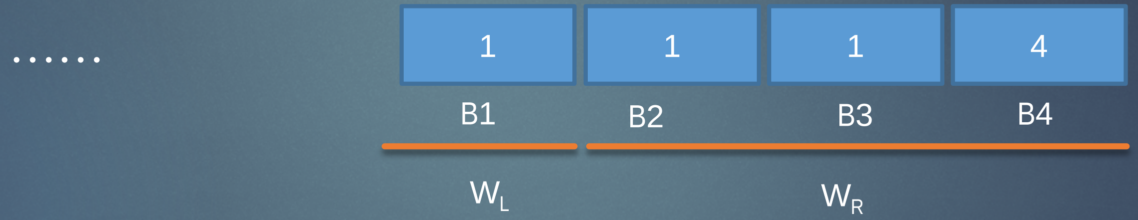
……

B1	B2	B3	B4
1 0 0 0	1 0 0 0	0 1 0 0	1 1 1 1

SEED Detector – Change Detector

......	1	1	1	4
	B1	B2	B3	B4

SEED Detector – Change Detector

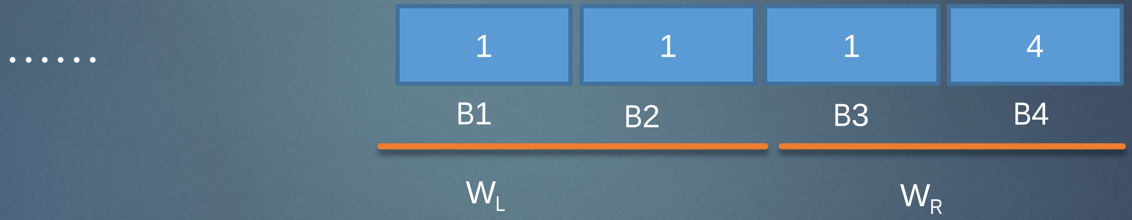


To check for drift, the window W is split into two sub-windows W_L and W_R and each of the boundaries between the blocks is considered as a potential drift.

$\mu W_L - \mu W_R \geq \varepsilon$

ε is the drift threshold

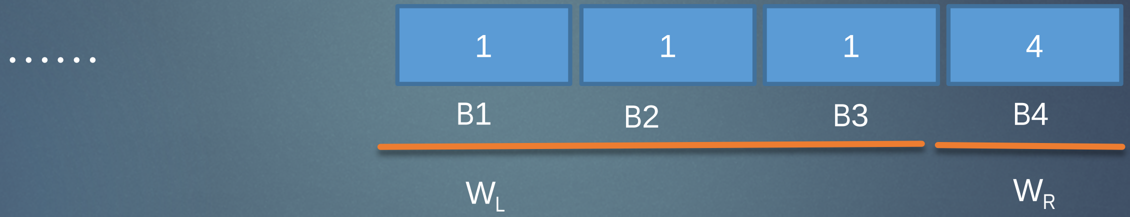
SEED Detector – Change Detector



$\mu W_L - \mu W_R \geq \varepsilon$

ε is the drift threshold

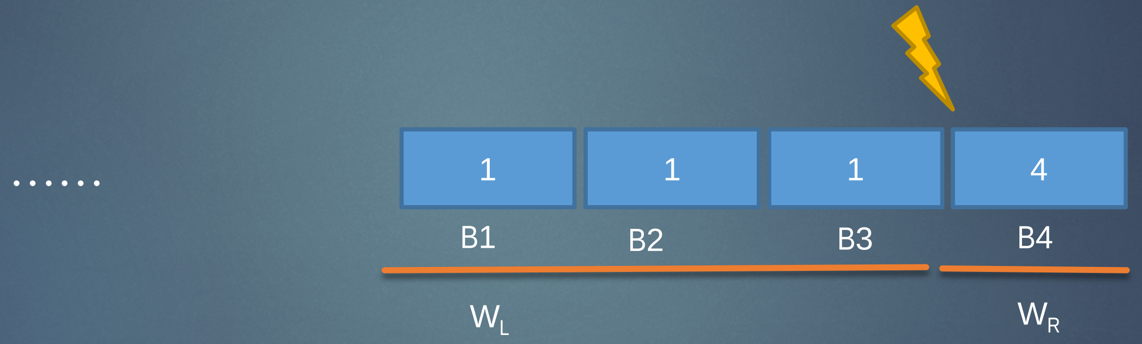
SEED Detector – Change Detector



$\mu W_L - \mu W_R \geq \varepsilon$

ε is the drift threshold

SEED Detector – Change Detector



$\mu W_L - \mu W_R \geq \varepsilon$

ε is the drift threshold

ε normally set using a statistical bound *i.e. Hoeffding bound*

SEED Detector – Change Detector

......	1 0 0 0	1 0 0 0	0 1 0 0	1 1 1 1
	B1	B2	B3	B4

Using every boundary as potential drift point is excessive. SEED performs block compressions to merge consecutive blocks that are homogeneous in nature.

$\mu W_L - \mu W_R \leq \varepsilon_{Homogeneous}$

$\varepsilon_{Homogeneous}$ is the merge threshold

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

SEED Detector – Change Detector

……	1 0 0 0	1 0 0 0	0 1 0 0	1 1 1 1
	B1	B2	B3	B4

Using every boundary as potential drift point is excessive. SEED performs block compressions to merge consecutive blocks that are homogeneous in nature.

$\mu W_L - \mu W_R \leq \varepsilon_{Homogeneous}$

$\varepsilon_{Homogeneous}$ is the merge threshold

SEED Detector – Change Detector

……

1000 1000 0100	1111
B1 – B3	B4

$\mu W_L - \mu W_R \leq \varepsilon_{Homogeneous}$

$\varepsilon_{Homogeneous}$ is the merge threshold

SEED Detector – Change Detector

……

1000 1000 0100	1111
B1 – B3	B4
W_L	W_R

$\mu W_L - \mu W_R \leq \varepsilon_{Homogeneous}$

$\varepsilon_{Homogeneous}$ is the merge threshold

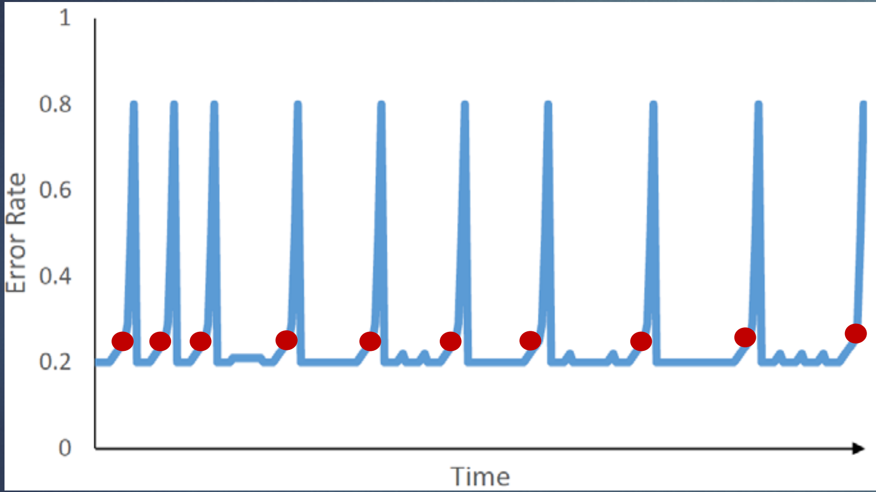
Volatility Shift in Data Streams

David Tse Jung Huang, Yun Sing Koh, Gillian Dobbie, Russel Pears: Detecting Volatility Shift in Data Streams. ICDM 2014

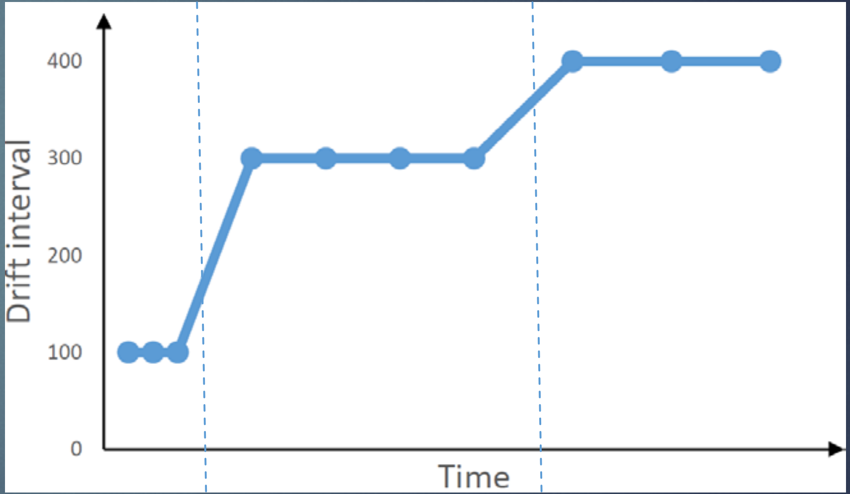
- It is useful to understand characteristics of a stream, such as volatility.
- Example: Machine performance and maintenance
  - Drift: Deviations in machine performance.
  - Volatility: Monitoring the deviations.

Example of Drift Volatility

- Error rate stream showing drift points



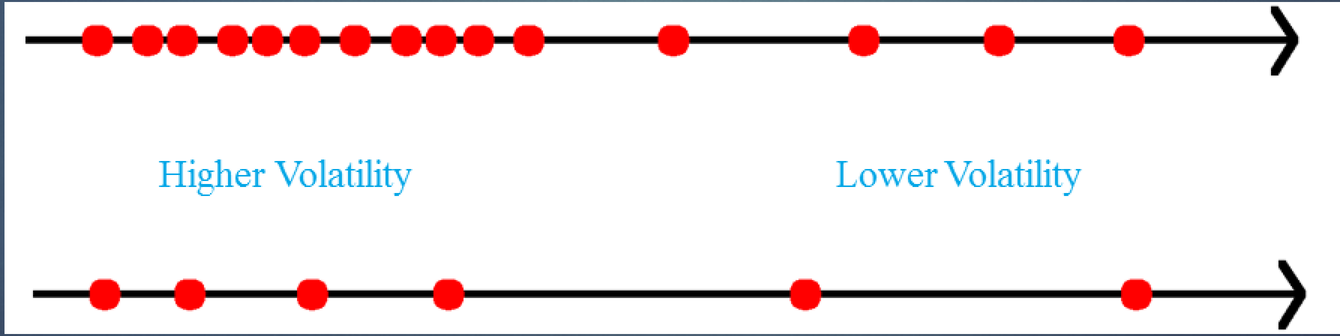
- Drift volatility (rate of change)



Volatility Shift in Data Streams

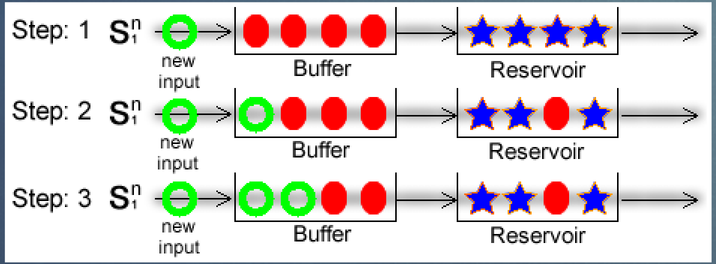
Input Stream → Drift Detector → Drift Points → Volatility Detector → Volatility Shifts

- A stream has a high volatility if drifts are detected frequently and has a low volatility if drifts are detected infrequently.
- Streams can have similar characteristics but be characterized as stable and non-volatile in one field of application and extremely volatile in another.



Volatility Detector Example

- There are two main components in our volatility detector: a buffer and a reservoir.
- The buffer is a sliding window that keeps the most recent samples of drift intervals acquired from a drift detection technique.
- The reservoir is a pool that stores previous samples which ideally represent the overall state of the stream.



$$Relative\ Volatility = \frac{\sigma_{BUFFER}}{\sigma_{RESERVOIR}}$$

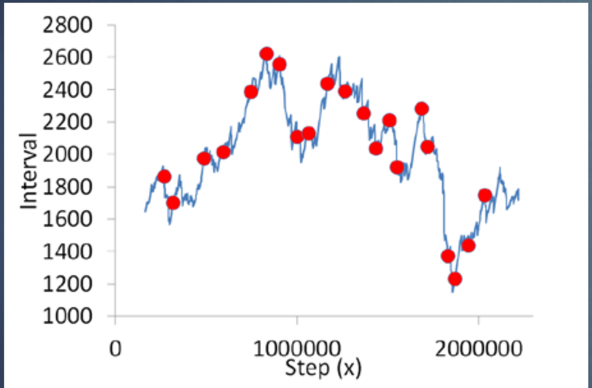
Shift in Relative Variance:

Given a user defined confidence threshold β ϵ [0,1], a shift in relative variance occurs when

$$Relative\ Variance > 1.0 + \beta$$
$$Relative\ Variance < 1.0 - \beta$$

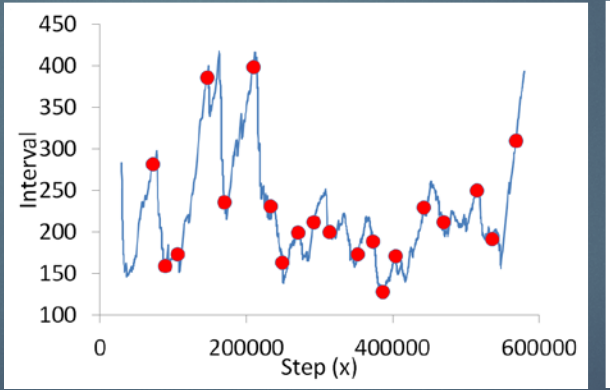
Real World Results

Each stream was evaluated using a Hoeffding Tree to produce the binary stream that represents the classification errors then passed to our drift detector.



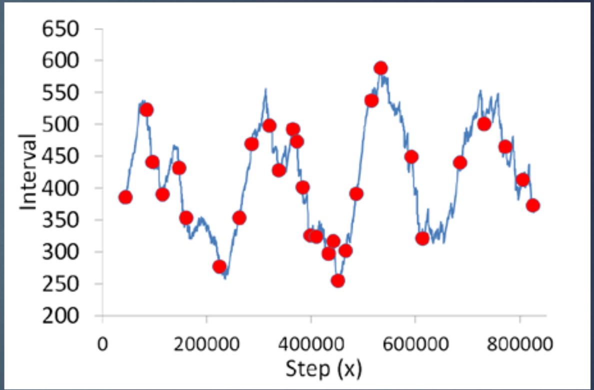
Sensor Stream

- 1,150 change points found
- 21 volatility shifts
- intervals between 1500 to 2500



Forest Covertype

- 2,611 change points found
- 20 volatility shifts
- intervals between 100 to 450



Poker Hand

- 2,059 change points were found
- 30 volatility shifts
- intervals between 150 to 600

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

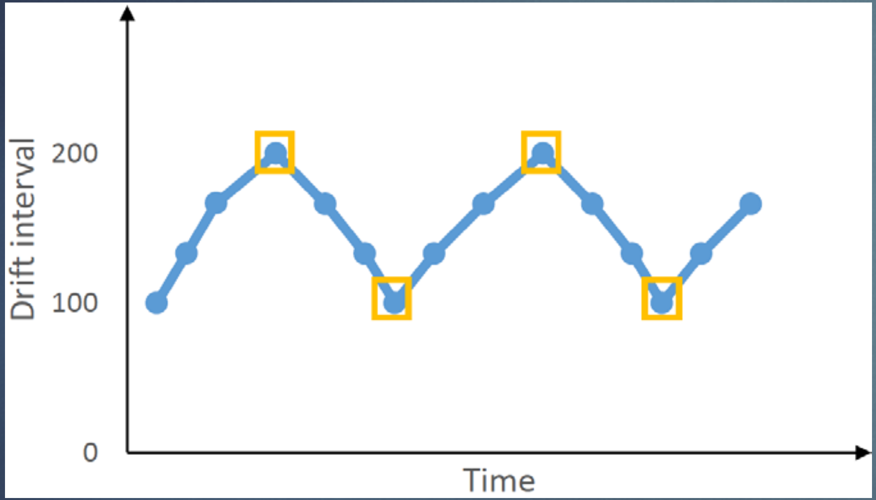
Proactive Drift Detection System

Kylie Chen, Yun Sing Koh, Patricia Riddle: Proactive drift detection: Predicting concept drifts in data streams using probabilistic networks. IJCNN 2016: 780-787

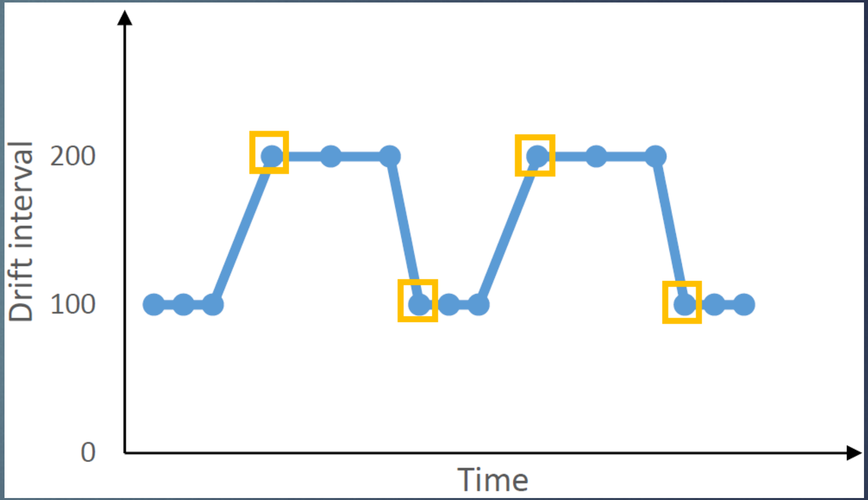
- Modelling Drift Volatility Trends
- Goals:
  - Predict location of next drift
    - Drift Prediction Method using Probabilistic Networks
  - Use predictions to develop proactive drift detection methods
    - Adaptation of Drift Detection Method SEED
    - Adaptation of data structure using compression

Modelling Drift Volatility Trends

- Progressive volatility change



- Rapid volatility change



Example of Drift Prediction Method

Example of drift intervals

$\boxed{100 \;\; 100}$ $\boxed{100 \;\; \mathbf{300}}$ 300 300 300 **400** 400 400

1. Identify volatility change points (Volatility Detector)
2. Outlier removal to construct pattern from drift interval windows

p_2 300 300 300

?

Pattern Reservoir

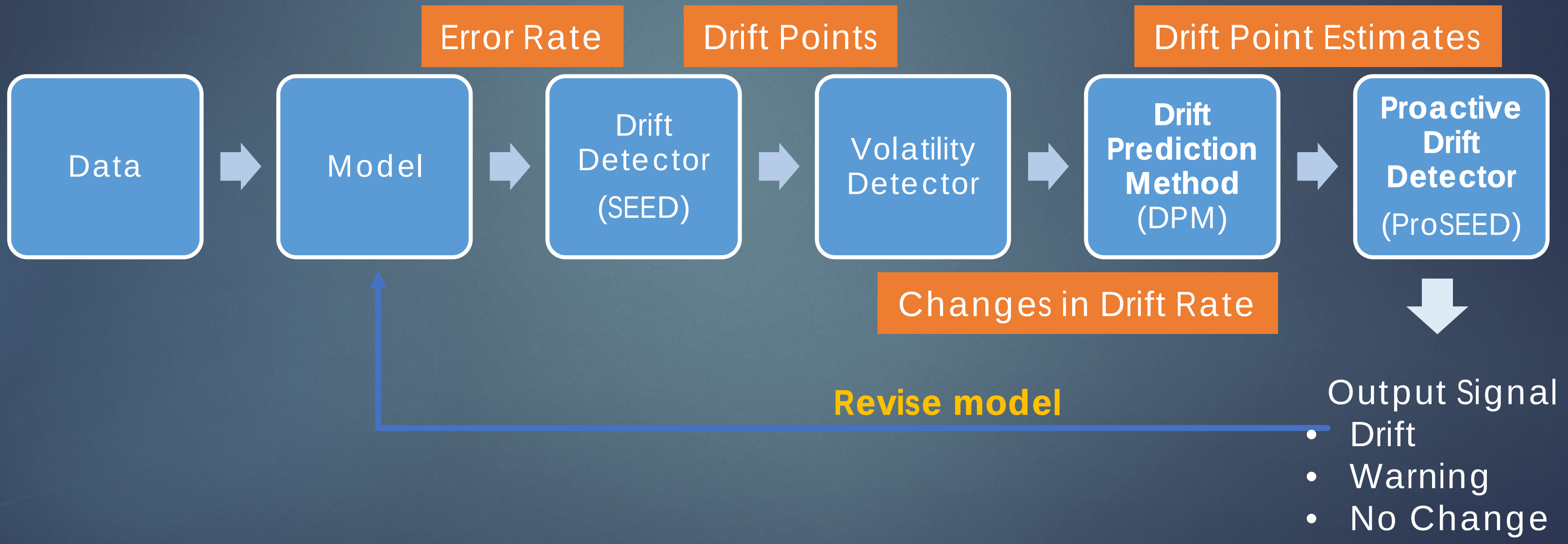
p_1 100 100 100

p_2 300 300 300

3. Match patterns to stored patterns
4. Update probabilistic network

$p_1 \xrightarrow{1.0} p_2$

Proactive Drift Detection System



Adapting the data structure of SEED

Extend the SEED Detector to use predicted drifts from our Drift Prediction Method

Adaptation of data compression of SEED detector

- no compression in blocks where we expect drift

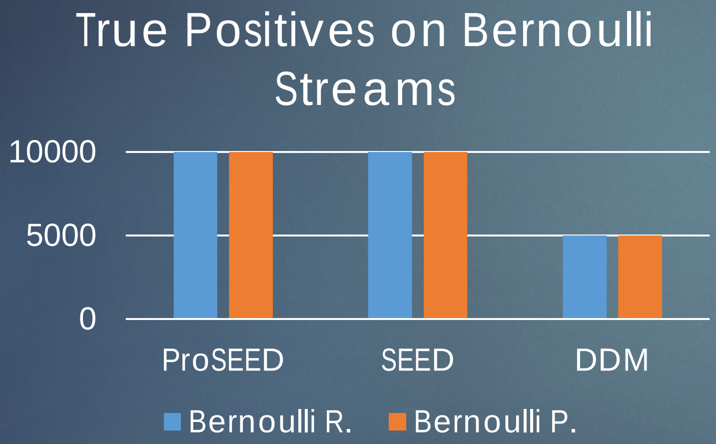
Example of error stream

- 00011000100110110111

Expected Predicted drifts at time steps 6 and 18

- 0001 | 1000 | 1001 | 1011 | 0111
- c1 c2 c3 c4
- 0001 | 1000 | 10011011 | 0111
- c1 c2 c3

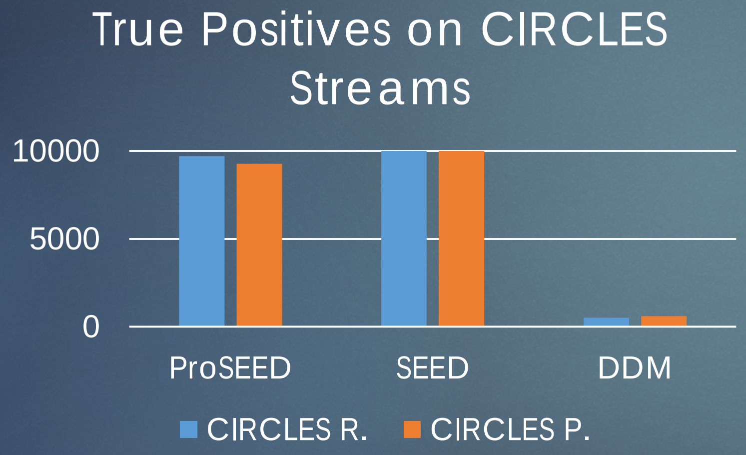
Results - Proactive Drift Detection (Bernoulli)



Average Number of False Positives

Detector	Bernoulli R.	Bernoulli P.
ProSEED	**33.10**	**44.32**
SEED	213.34	210.50
DDM	97.41	100.98

Results - Proactive Drift Detection (CIRCLES)



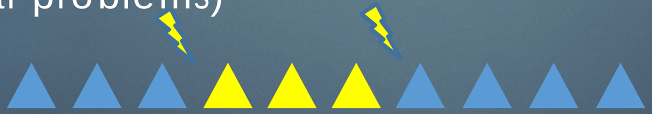
Average Number of False Positives

Detector	CIRCLES R.	CIRCLES P.
ProSEED	**271.44**	**10.05**
SEED	481.77	531.62
DDM	306.94	380.32

Concept Profiling Framework (CPF)

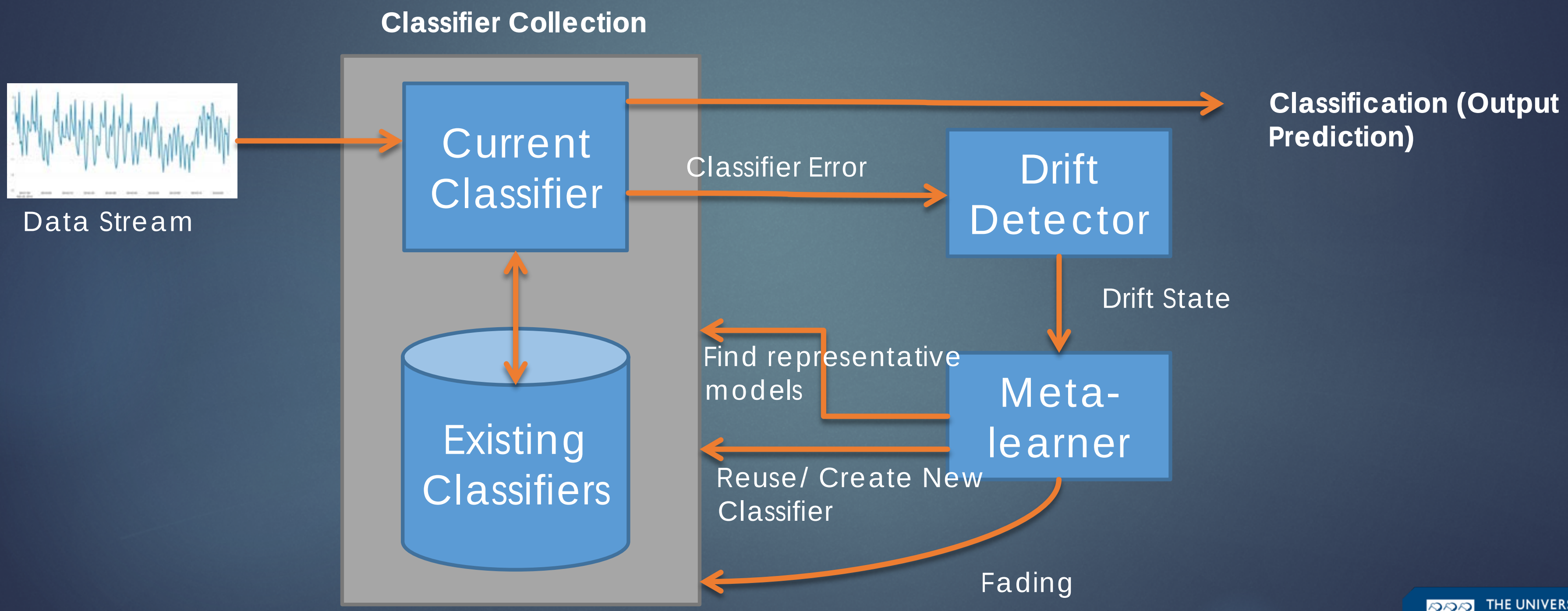
Robert Anderson, Yun Sing Koh, Gillian Dobbie: CPF: Concept Profiling Framework for Recurring Drifts in Data Streams. Australasian Conference on Artificial Intelligence 2016: 203-214 (Best Student Paper Award.)

- Concept Profiling Framework (CPF), a meta-learner that uses a concept drift detector and a collection of classification models to perform effective classification on data streams with recurrent concept, through relating models by similarity of their classifying behaviour.
- Existing state-of-the-art methods for recurrent drift classification often rely on resource-intensive statistical testing or ensembles of classifiers (time and memory overhead that can exclude them from use for particular problems)



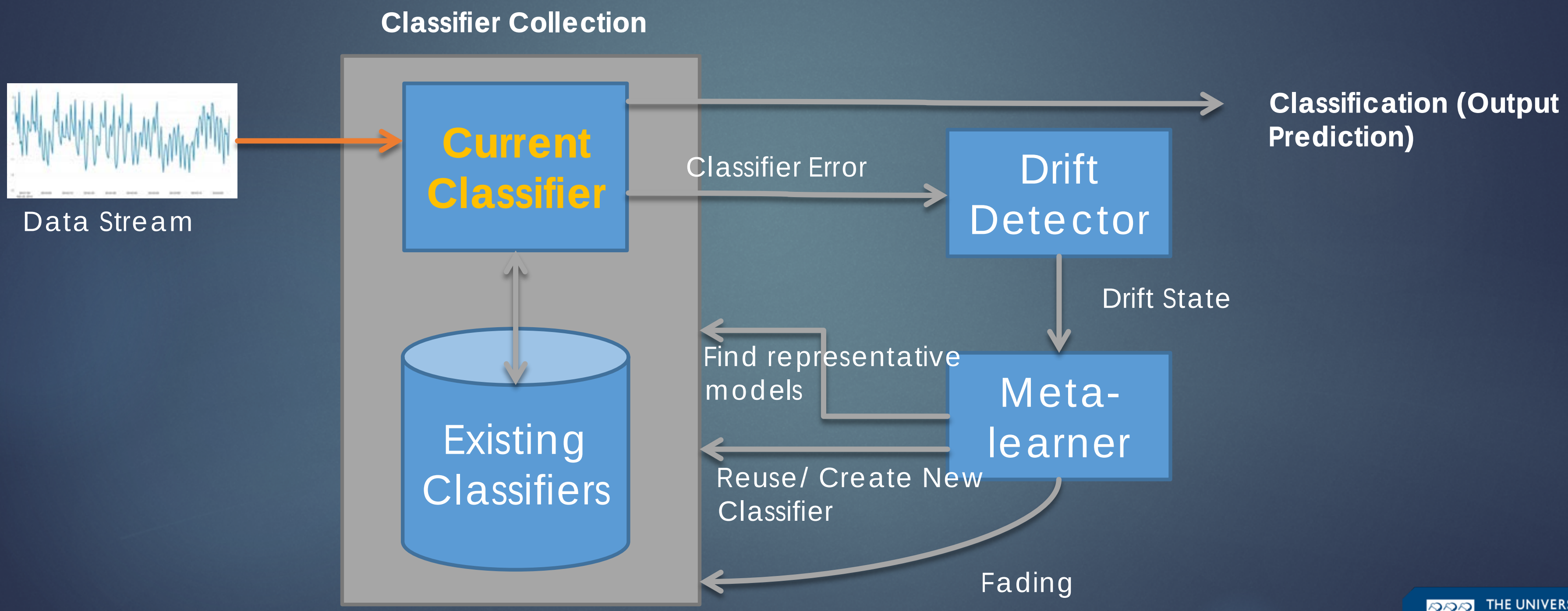
Recurring Concept

Concept Profiling Framework



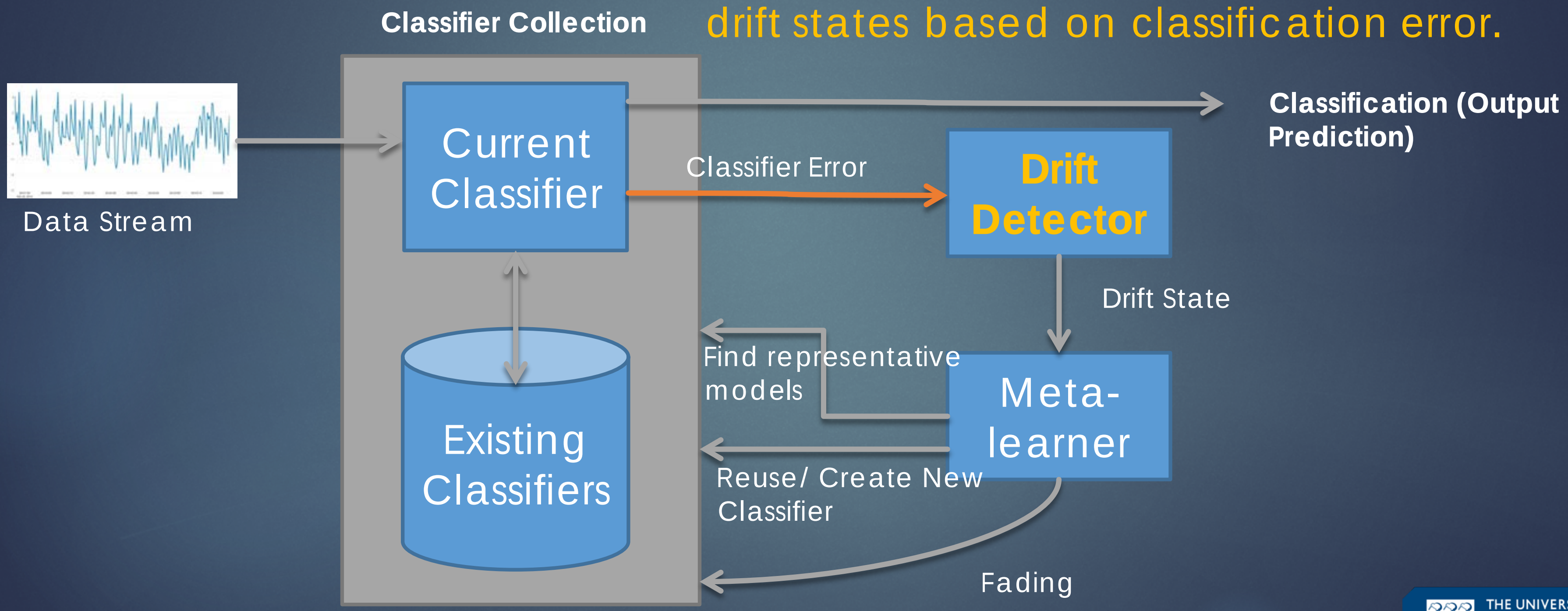
Concept Profiling Framework

Data Stream instances are handled by the current classifier.



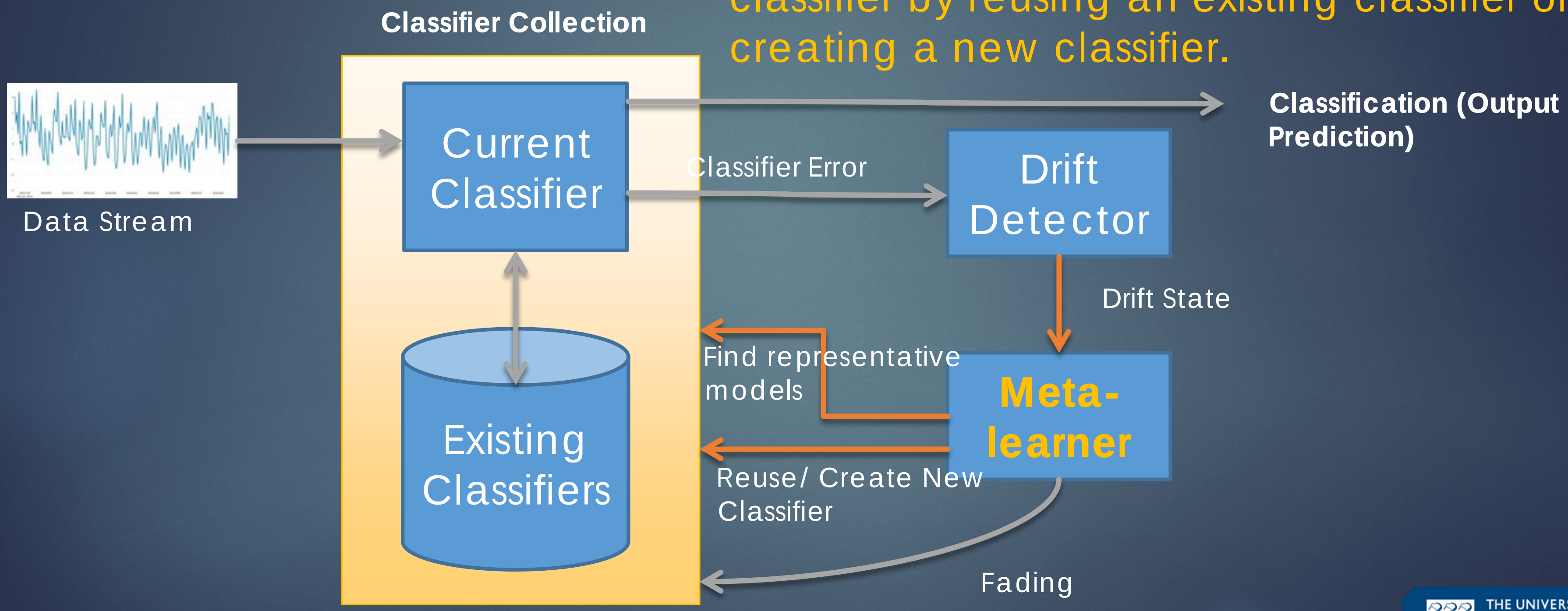
Concept Profiling Framework

The drift detector detects warning and drift states based on classification error.



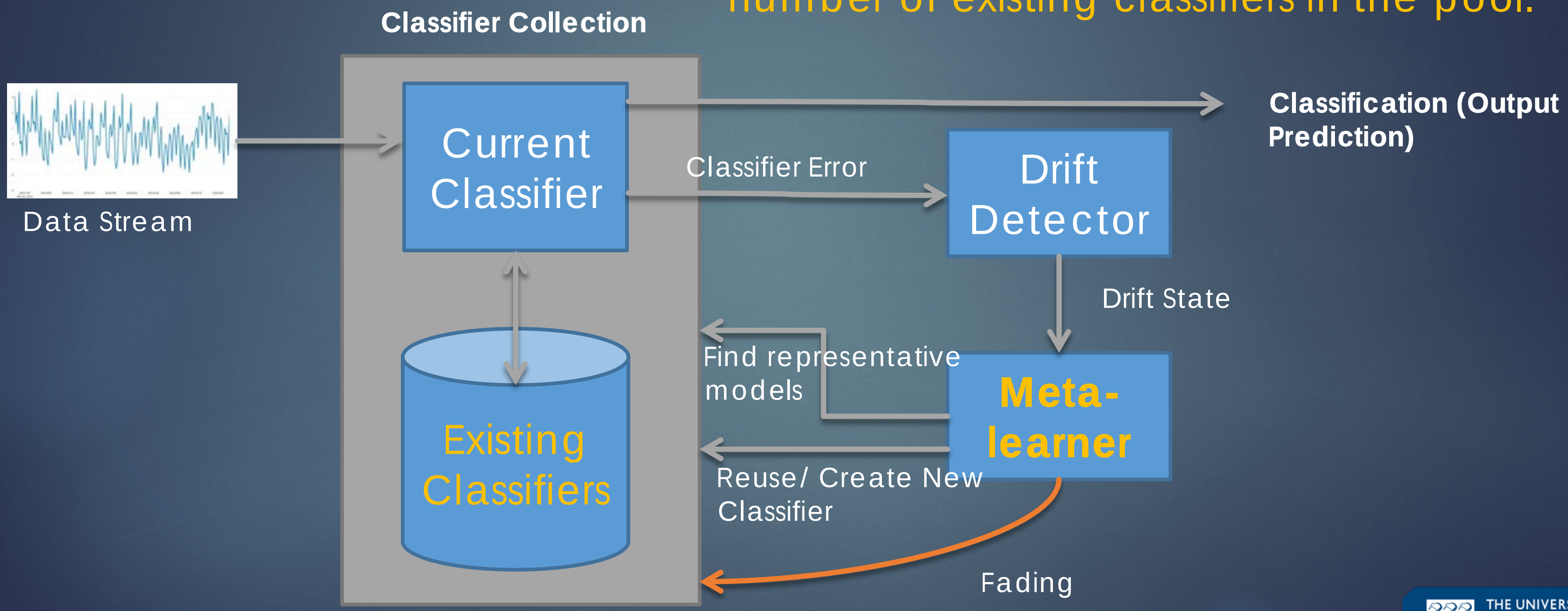
Concept Profiling Framework

On drift the meta-learner chooses the next classifier by reusing an existing classifier or creating a new classifier.



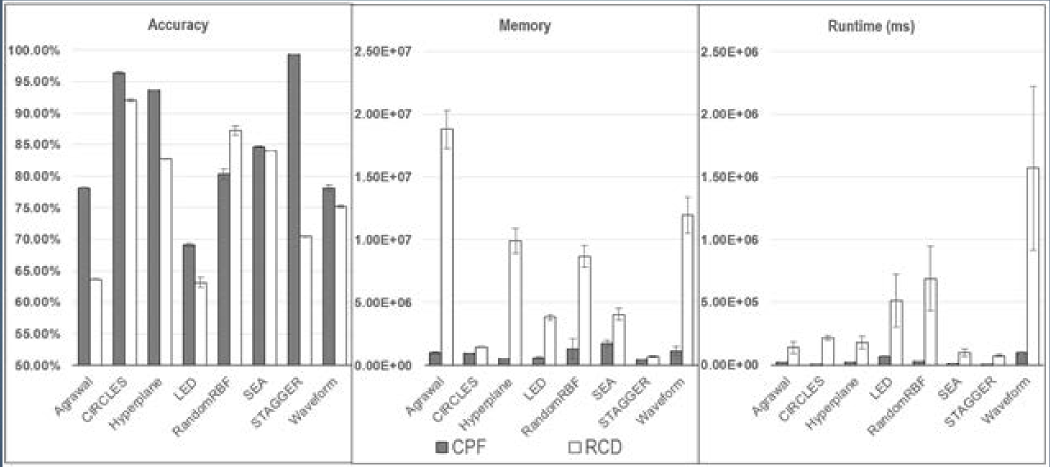
Concept Profiling Framework

Fading mechanism controls the number of existing classifiers in the pool.



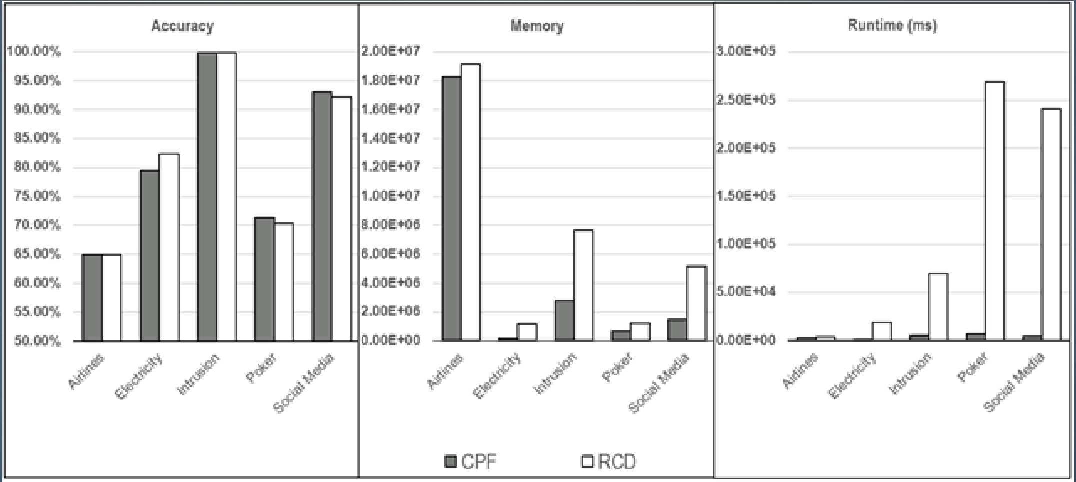
CPF: Synthetic datasets

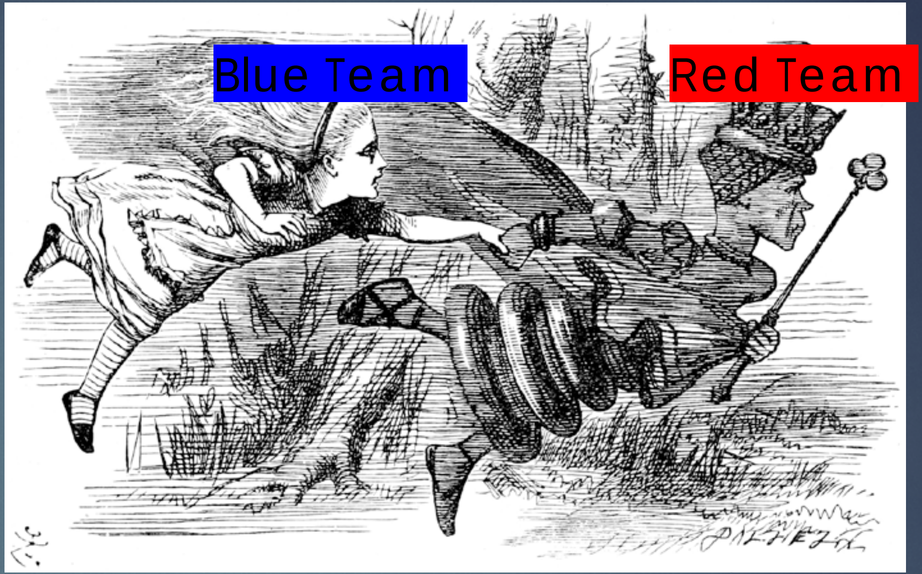
- Our technique generally achieved better accuracy while taking less time and memory than RCD



CPF: Real-world datasets

- Our technique generally maintained similar accuracy while taking less time and memory than RCD





Concept Drift in Security Applications

RED QUEEN'S RACE

Credit Card Fraud Detection

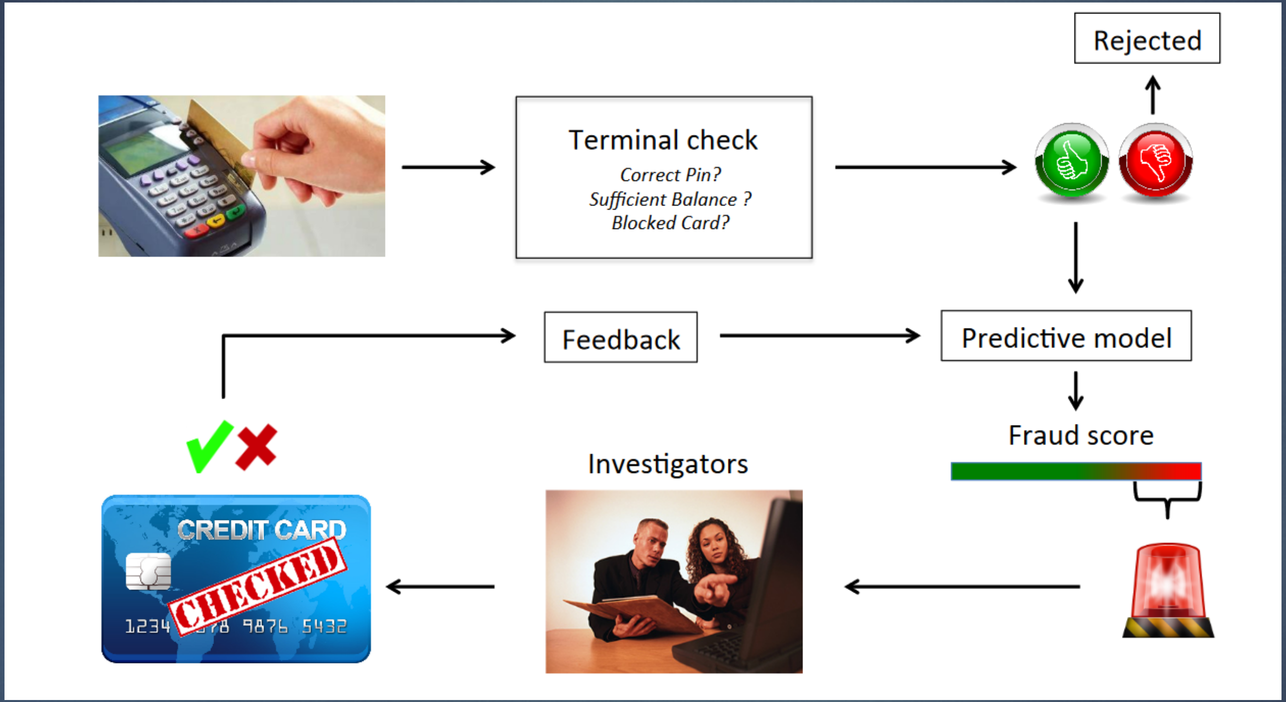


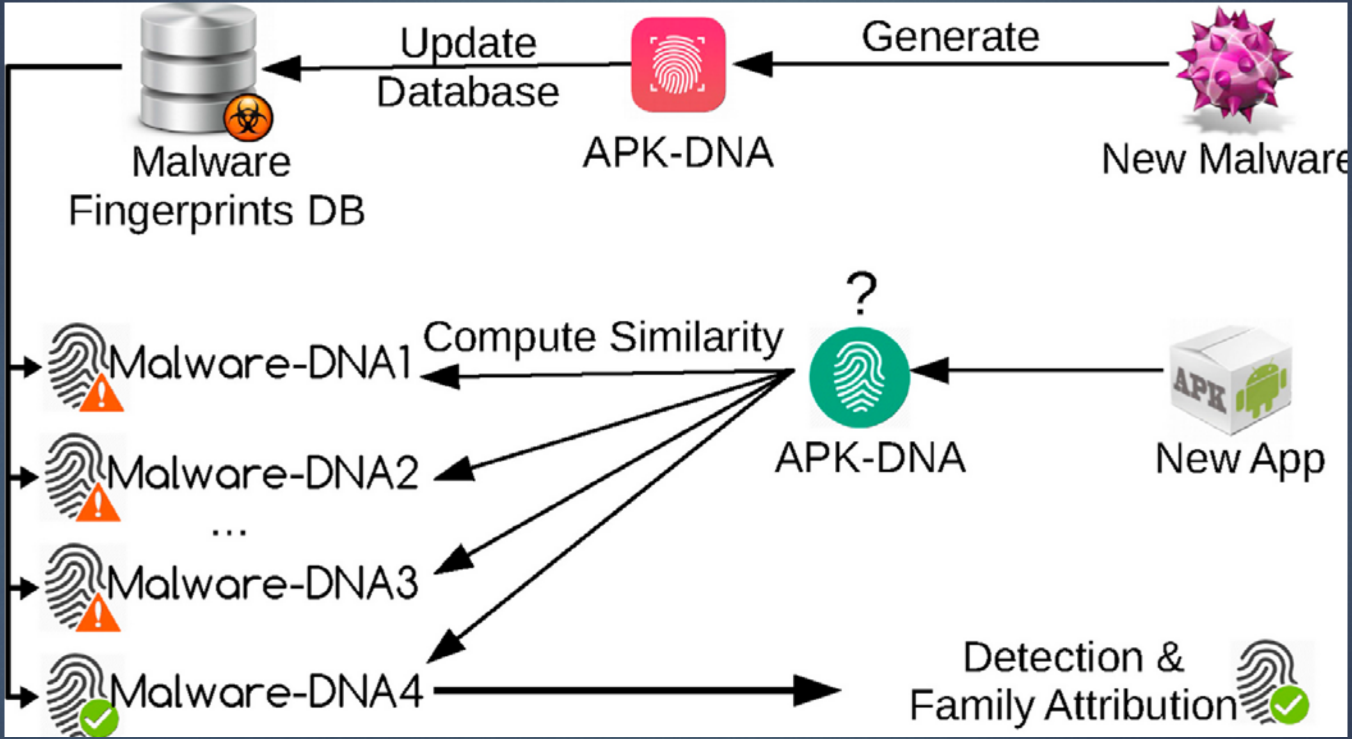
Image credit: Andrea Dal Pozzolo. Adaptive Machine Learning for Credit Card Fraud Detection. 2015

Challenges in Fraud Detection Systems using Machine Learning

- Frauds represent a small fraction of all the daily transactions (*class unbalance*).
- Frauds distribution evolves over time because of seasonality and new attack strategies (*concept drift*) .
- The true nature (class) of the majority of transactions is typically known only several days after the transaction took place, since only few transactions are timely checked by investigators (*uncertain class labels*).

[1] Andrea Dal Pozzolo, Olivier Caelen, Yann-Ael Le Borgne, Serge Waterschoot, and Gianluca Bontempi. Learned lessons in credit card fraud detection from a practitioner perspective. Expert Systems with Applications, 41(10):4915–4928, 2014.
[2] Andrea Dal Pozzolo, Reid A. Johnson, Olivier Caelen, Serge Waterschoot, Nitesh V Chawla, and Gianluca Bontempi. Using HDDT to avoid instances propagation in unbalanced and evolving data streams. In International Joint Conference on Neural Networks (IJCNN), 2014, pages 588–594. IEEE, 2014.
[3] Andrea Dal Pozzolo, Giacomo Boracchi, Olivier Caelen, Cesare Alippi, and Gianluca Bontempi. Credit card fraud detection and concept-drift adaptation with delayed supervised information. In Joint Conference on Neural Networks (IJCNN), 2015 International. IEEE, 2015.

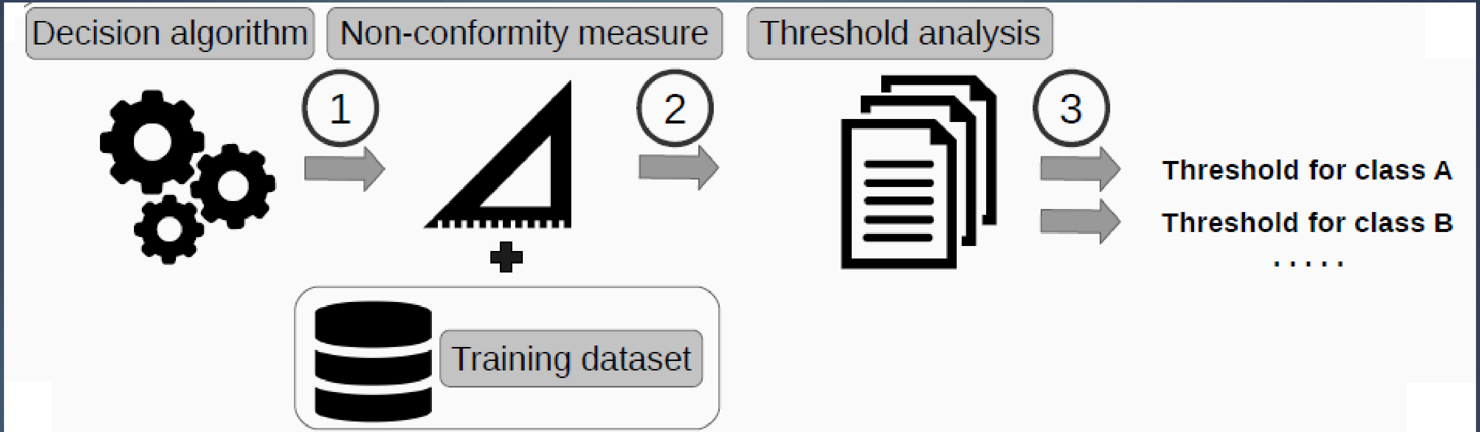
Concept Drift in Malware Families



Challenges in Malware Detection using Machine Learning

- Traditional batch-learning based methods are typically plagued by two challenges that make them unsuitable for real-world large-scale malware detection:
- **Population drift:** the population of malware is constantly evolving e.g. exploiting new vulnerabilities, and evading novel detection techniques.
  - collection of malware identified today unrepresentative of malware generated in the future.
- **Volume:** Batch learners have to be frequently re-trained using huge volumes of data, explains the paper.
  - severe scalability issues when used in the Android malware detection context where we have millions of samples already and thousands streaming in every day. Retraining frequently with such a volume renders them computationally impractical.

Concept Drift in Malware Families



Annamalai Narayanan, Yang Liu, Lihui Chen, Jinliang Liu: Adaptive and scalable Android malware detection through online learning. IJCNN 2016: 2484-2491
A. Narayanan, M. Chandramohan, L. Chen and Y. Liu: Context-Aware, Adaptive, and Scalable Android Malware Detection Through Online Learning, in IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 1, no. 3, pp. 157-175, June 2017.
Roberto Jordaney, Kumar Sharad, Santanu Kumar Dash, Zhi Wang et al.: Transcend: detecting concept drift in malware classification models. Proceedings of the 26th USENIX Security Symposium (USENIX Security 2017). 2017.

Thank you and Questions?