Classifier Robustness in Adversarial Settings

Francesco Bergadano

Department of Computer Science
University of Torino

Presentation plan

1) Basic concepts in **anomaly detection**
   1) Cybersecurity applications of Machine Learning
   2) Formalization and metrics
   3) ROC & PR curve analysis
2) Adversarial **evasion** and defenses
   1) Adversarial examples and evasion
   2) Known evasion resistance measures
   3) Defenses and randomization
   4) Randomization as keyed learning
3) New research*
   1) Adversarial failure curves
   2) New randomization techniques
   3) Evaluation on Intrusion Detection data sets

*joint work with Sandeep Gupta and Bruno Crispo (University of Trento)

Basic Concepts in Anomaly Detection

1) Cybersecurity Applications of Machine Learning
2) Formalization and Metrics
3) ROC & PR curve analysis

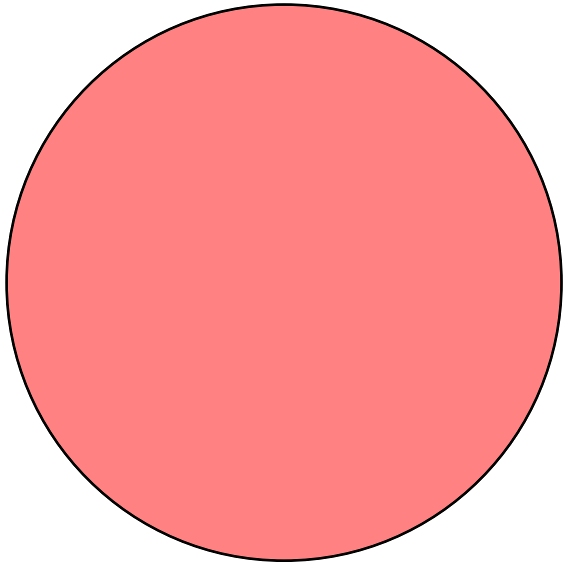
Cybersecurity applications of ML

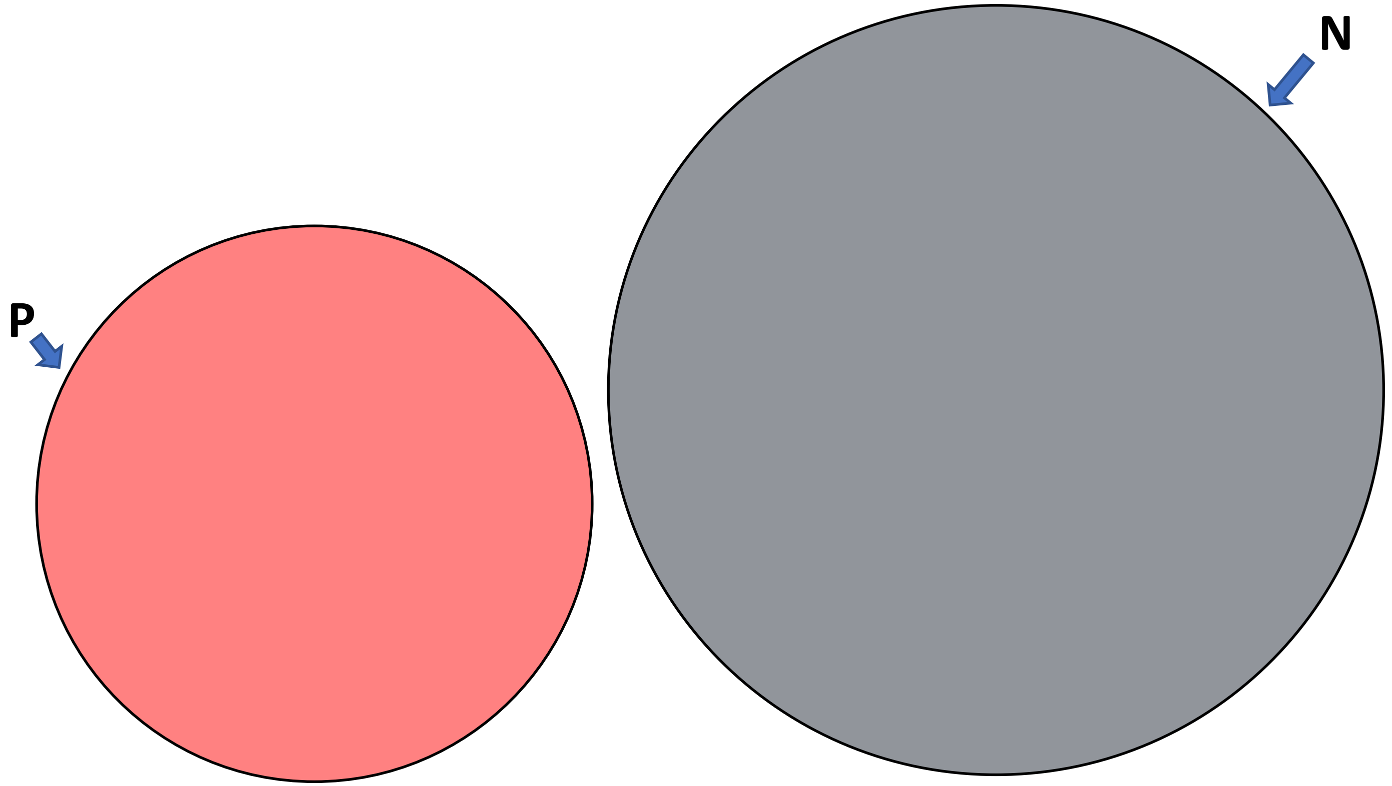
- User authentication
  - Authentication via physical biometrics
  - Authentication from user behaviors
    - Location / device info & type
    - Voice / sound
    - Keystroke / mouse / smarpthone dynamics
- Anomaly detection:
  - Host intrusion detection
  - Network intrusion detection
  - Malware detection
  - Spam filtering
  - Defacement response

Cybersecurity applications of ML

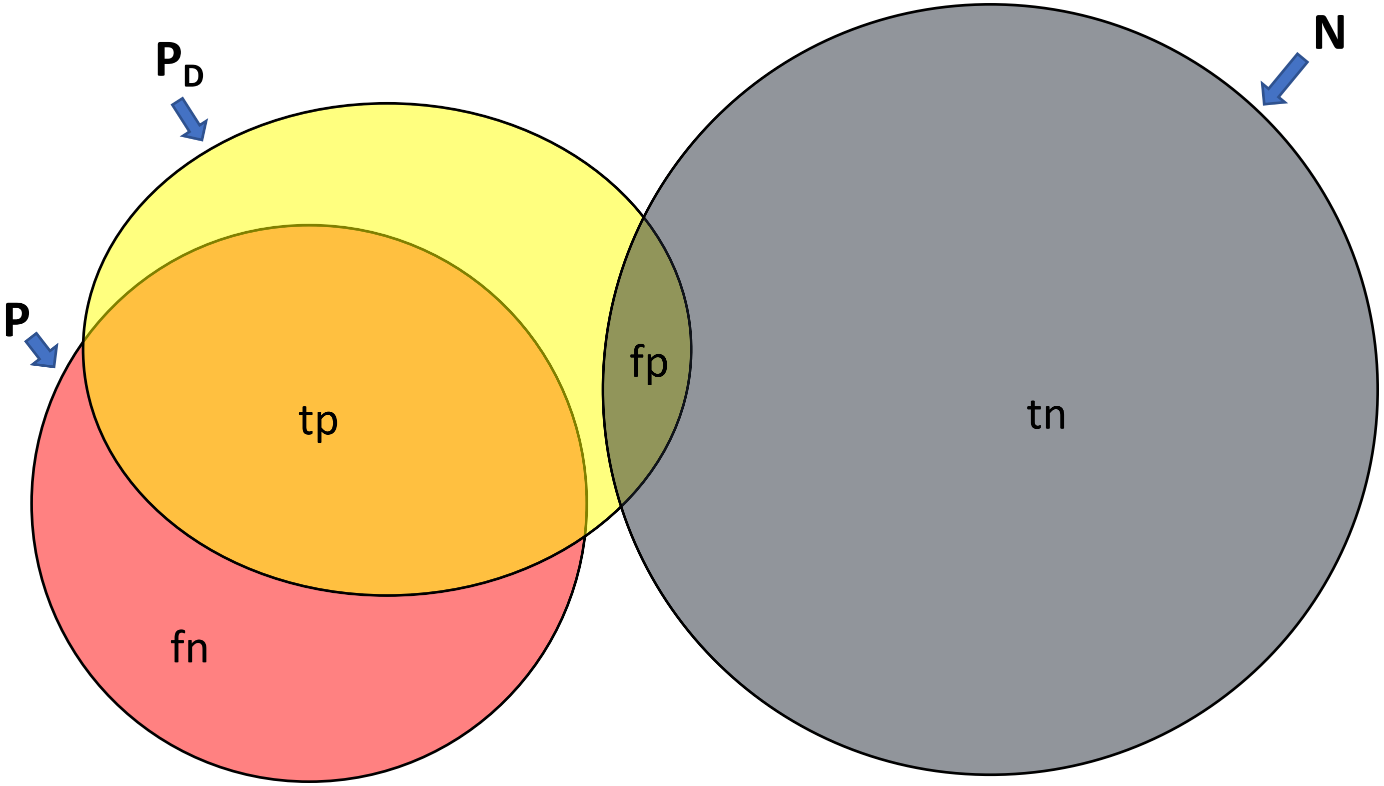
The user authentication / anomaly detection continuum

One step authentication	Multifactor authentication	Risk-based authentication	Continuous authentication	Anomaly detection





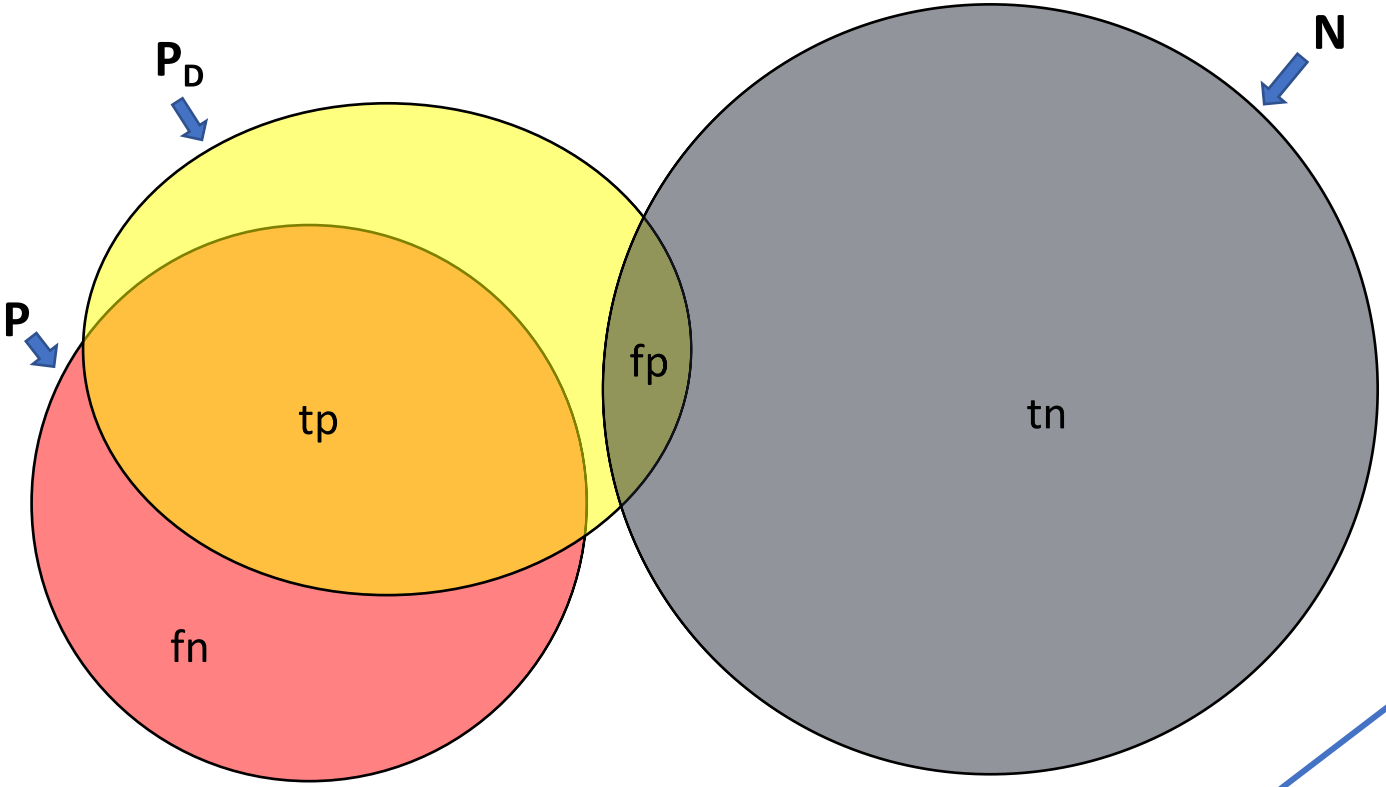
P = positives (anomalies)
N = negatives (normal data)



P = positives (anomalies)
N = negatives (normal data)
P_D = classified as positive by defender

fp = N ∩ P_D = defender false positives
tp = P ∩ P_D = defender true positives
fn = P-tp = defender false negatives
tn = N-fp = defender true negatives

Standard Anomaly Detection Concepts



Well-known metrics

confusion matrix (contingency table) →

|tp|	|fp|
|fn|	|tn|

accuracy = correct/total = (|tp|+|tn|)/(|P|+|N|)

tpr = recall = sensitivity = hit rate = |tp|/|P|

fpr = false alarm rate = |fp|/ |N|

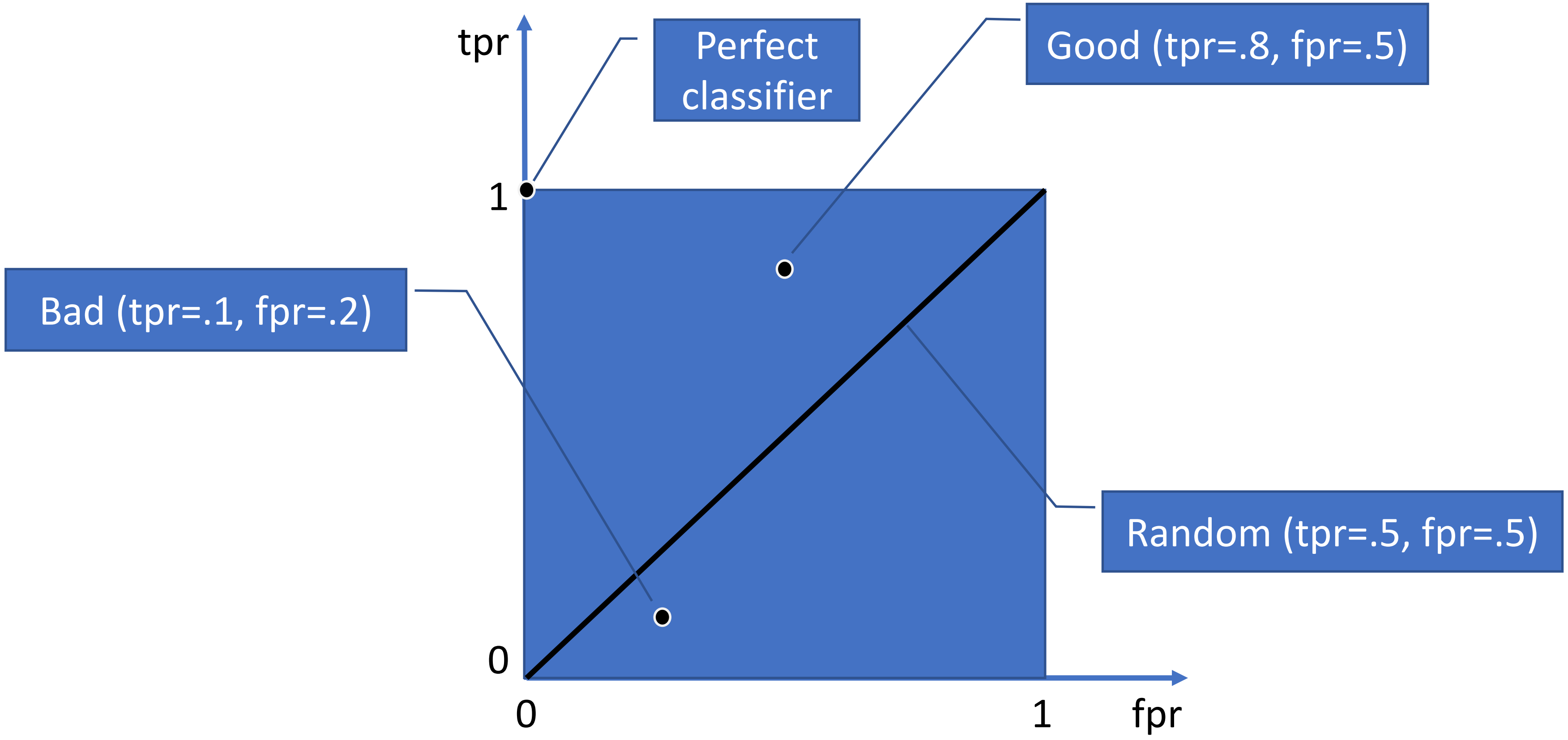
precision = positive predictive value =
= correctness when P_D=anomaly =
= |tp|/(|tp|+|fp|)

specificity = |tn| / (|fp|+|tn|) = 1 – fpr

F-measure = F_1 score = 2/[(1/precision)+(1/recall)]

Defender success = $\begin{cases} \textbf{maximise true positive rate} \\ \textbf{minimise false positive rate} \end{cases}$

ROC space (Receiver Operating Characteristics)[1]



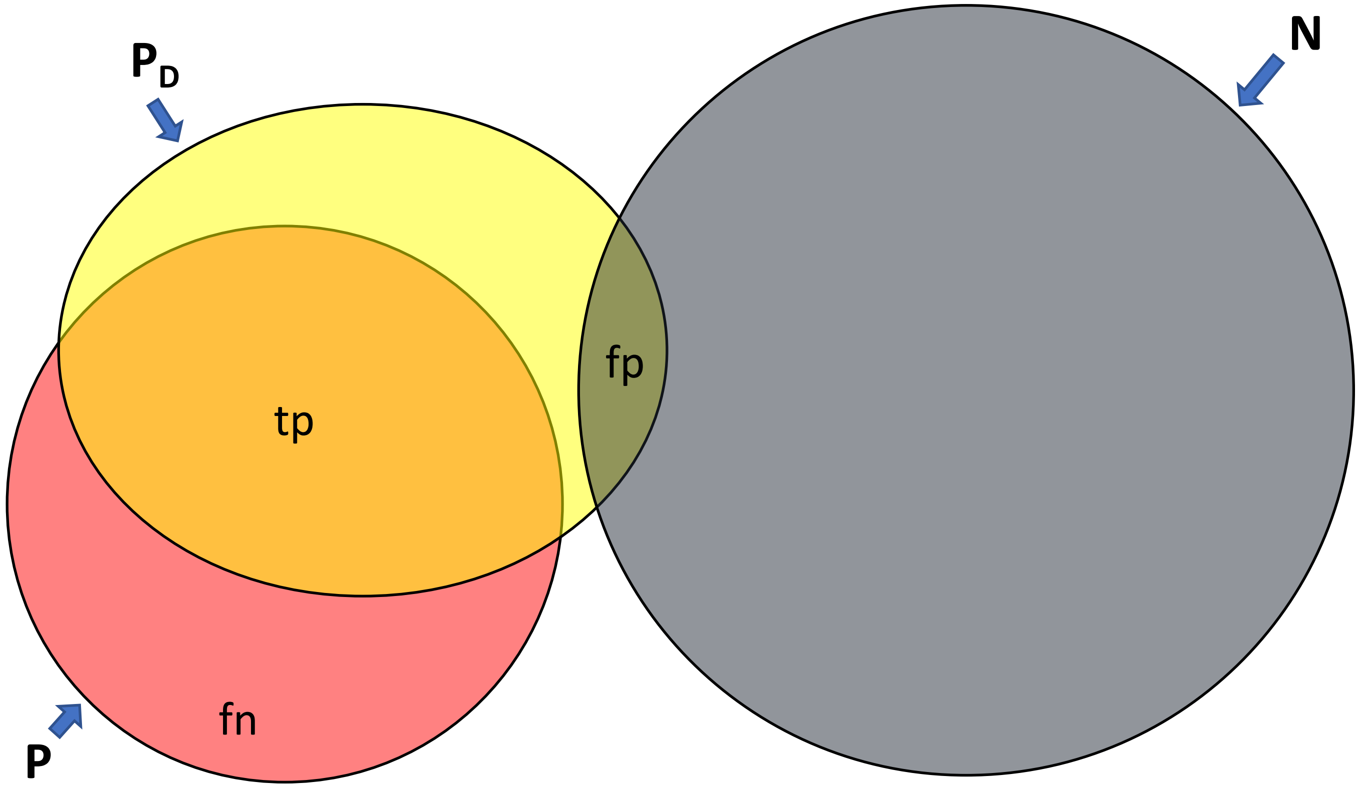
[1] Tom Fawcett, “An introduction to ROC analysis”, Pattern Recognition Letters 27, pp. 861-874, 2006

Threshold (or probabilistic) classifiers

They do not output just 0 or 1 (normal data vs anomaly),
buth rather an arbitrary real number, called a *score.*

We can transform a threshold classifier TC into a
discrete classifier C as follows:

C(e) = 1 (anomaly) if TC(e) ≥ threshold



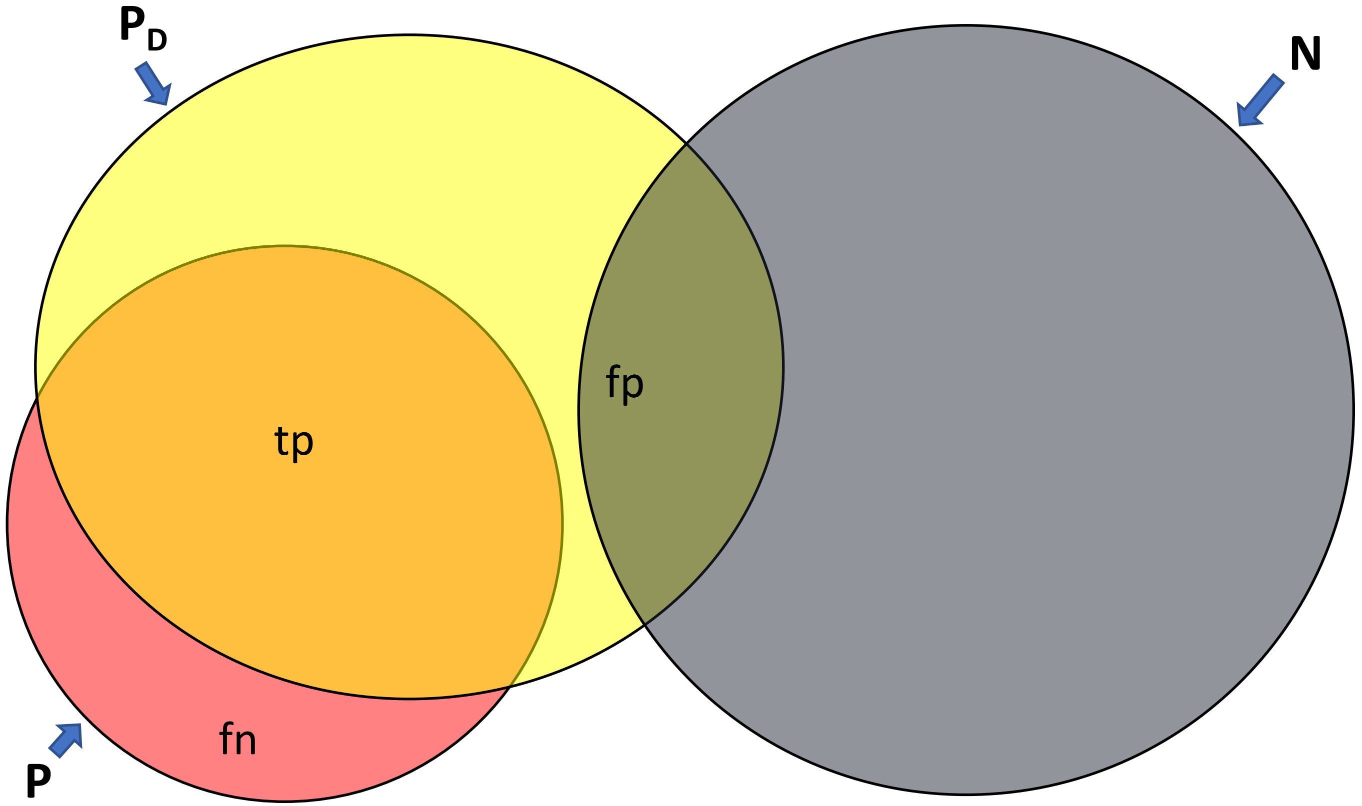
clf_D = defender's classifier

Normally clf_D is a threshold classifier,
i.e. $clf_D(e)$=score and $e \in P_D$ if score ≥ threshold

Defender success = minimize fpr
maximise tpr

$\textbf{fpr} = \frac{|fp|}{|N|}$ $(false\ positive\ rate)$

$\textbf{tpr} = \frac{|tp|}{|P|} = \frac{|P|-|fn|}{|P|} = 1\text{-fnr}$ $(true\ positive\ rate)$



clf_D = defender's classifier

Normally clf_D is a threshold classifier,
i.e. $clf_D(e)$=score and $e \in P_D$ if score $\geq$ threshold

When threshold decreases:
tp grows ← good
fp grows ← bad

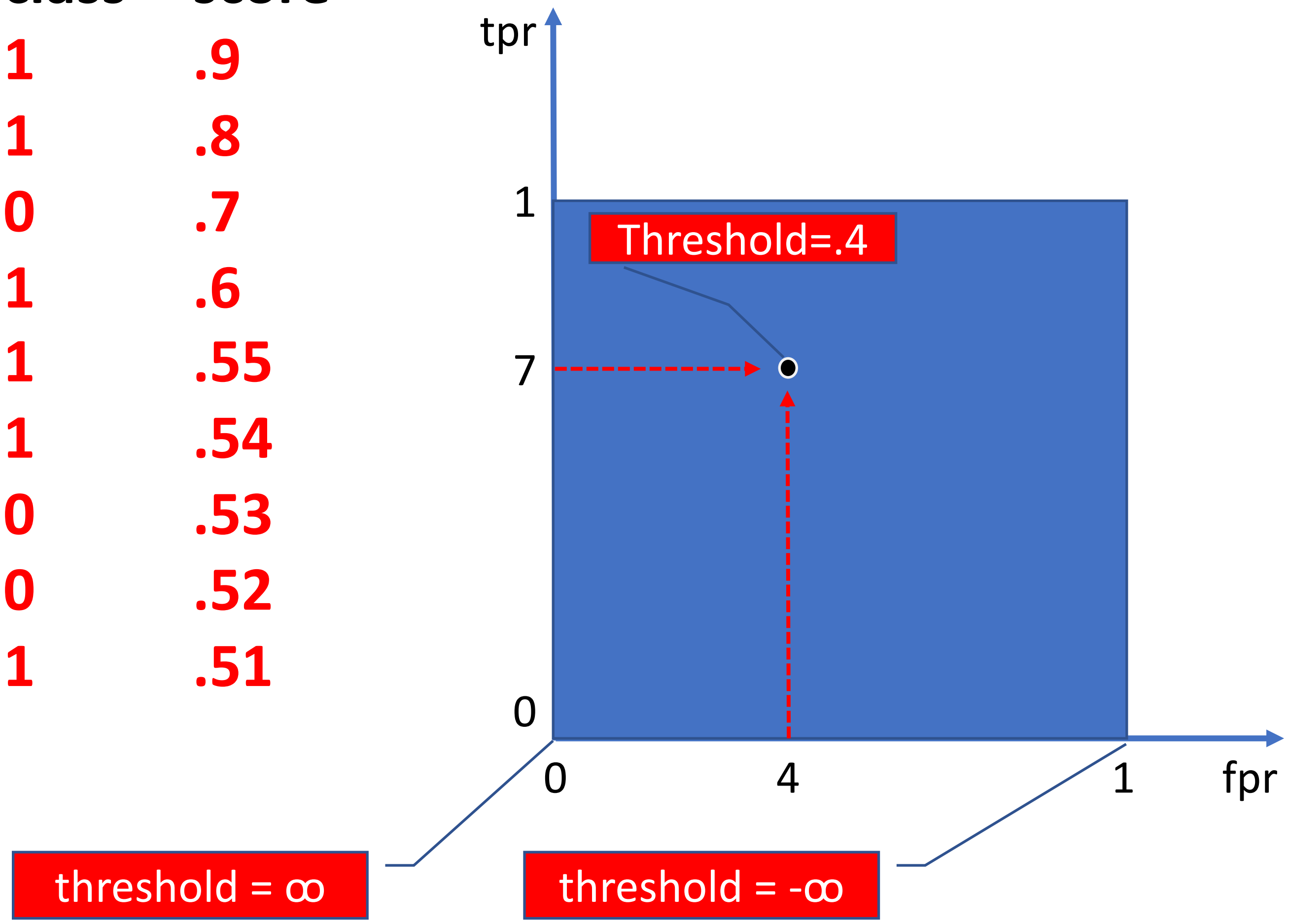
Defender success = minimize fpr → $\text{fpr} = \frac{|\text{fp}|}{|\text{N}|}$ $(\textit{false positive rate})$

maximise tpr → $\text{tpr} = \frac{|\text{tp}|}{|\text{P}|} = \frac{|\text{P}|-|\text{fn}|}{|\text{P}|} = 1\text{-fnr}$ $(\textit{true positive rate})$

Threshold classifiers: ROC curves [1]

if score ≥ threshold then class=1

#n	class	score
1	1	.9
2	1	.8
3	0	.7
4	1	.6
5	1	.55
6	1	.54
7	0	.53
8	0	.52
9	1	.51

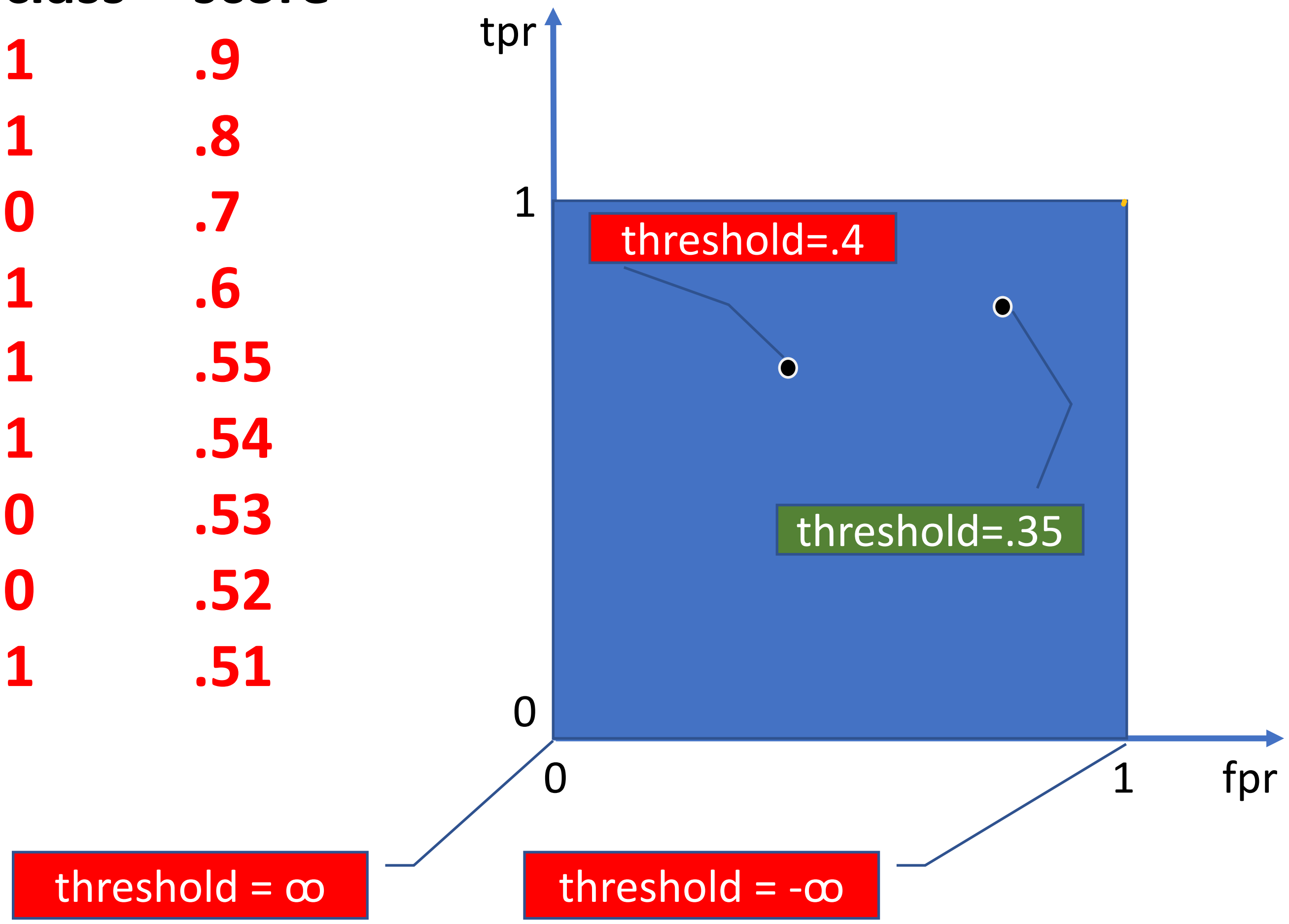


10	0	.505
11	1	.4
12	0	.39
13	1	.38
14	0	.37
15	0	.36
16	0	.35
17	1	.34
18	0	.33
19	1	.3
20	0	.1

Threshold classifiers: ROC curves [1]

if score ≥ threshold then class=1

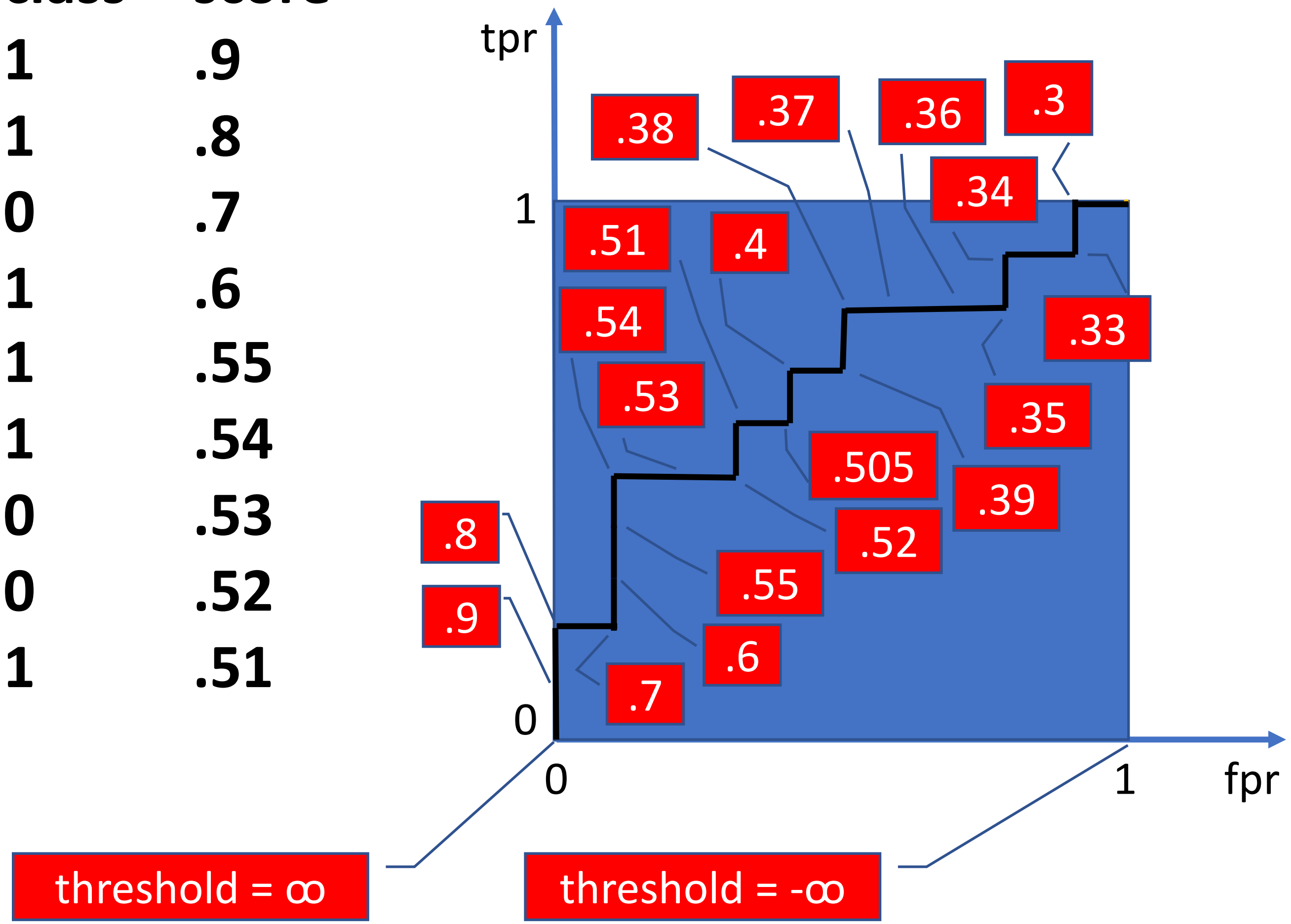
#n	class	score
1	1	.9
2	1	.8
3	0	.7
4	1	.6
5	1	.55
6	1	.54
7	0	.53
8	0	.52
9	1	.51
10	0	.505
11	1	.4
12	0	.39
13	1	.38
14	0	.37
15	0	.36
16	0	.35
17	1	.34
18	0	.33
19	1	.3
20	0	.1



Threshold classifiers: ROC curves [1]

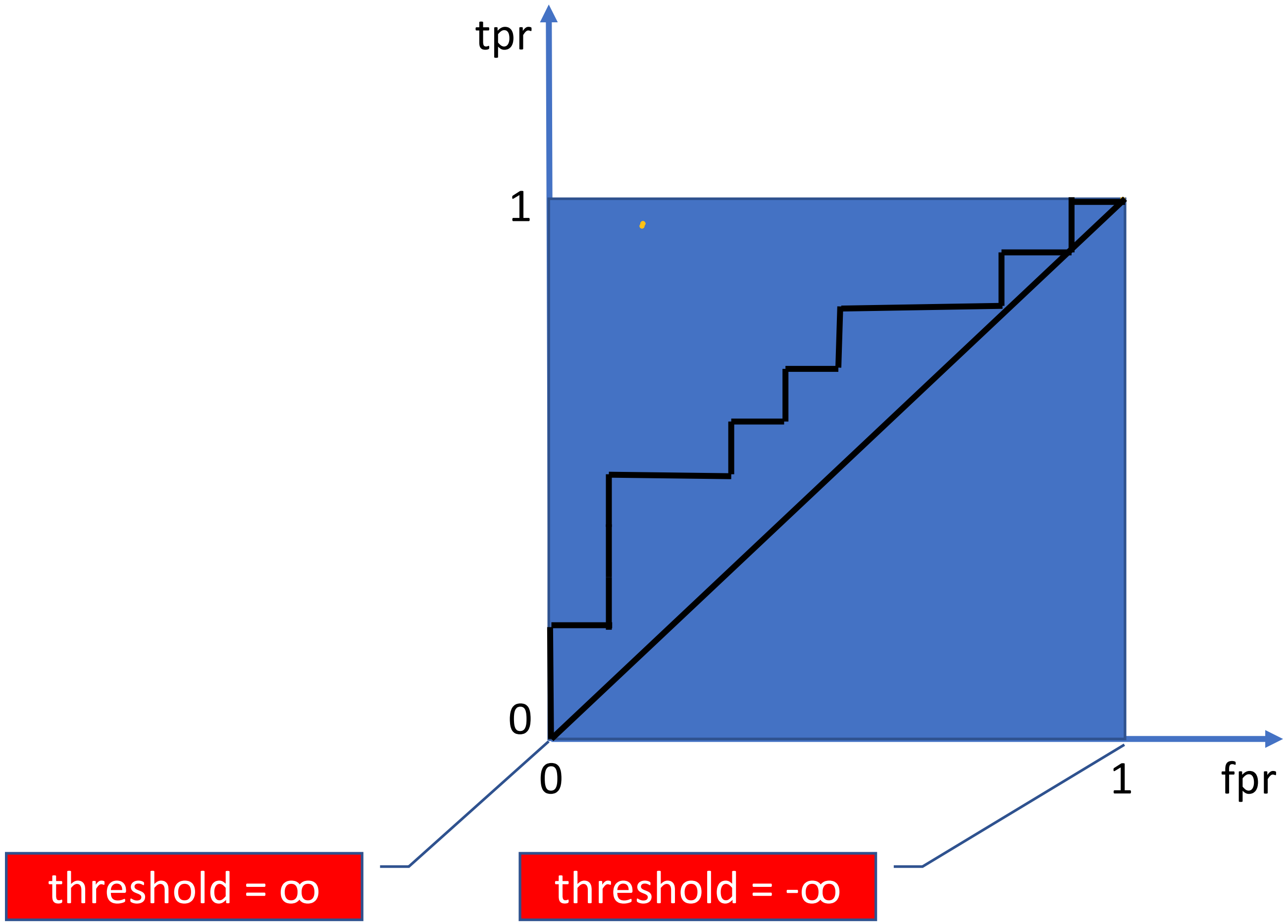
if score ≥ threshold then class=1

#n	class	score
1	1	.9
2	1	.8
3	0	.7
4	1	.6
5	1	.55
6	1	.54
7	0	.53
8	0	.52
9	1	.51
10	0	.505
11	1	.4
12	0	.39
13	1	.38
14	0	.37
15	0	.36
16	0	.35
17	1	.34
18	0	.33
19	1	.3
20	0	.1



Threshold classifiers: ROC curves [1]

if score ≥ threshold then class=1

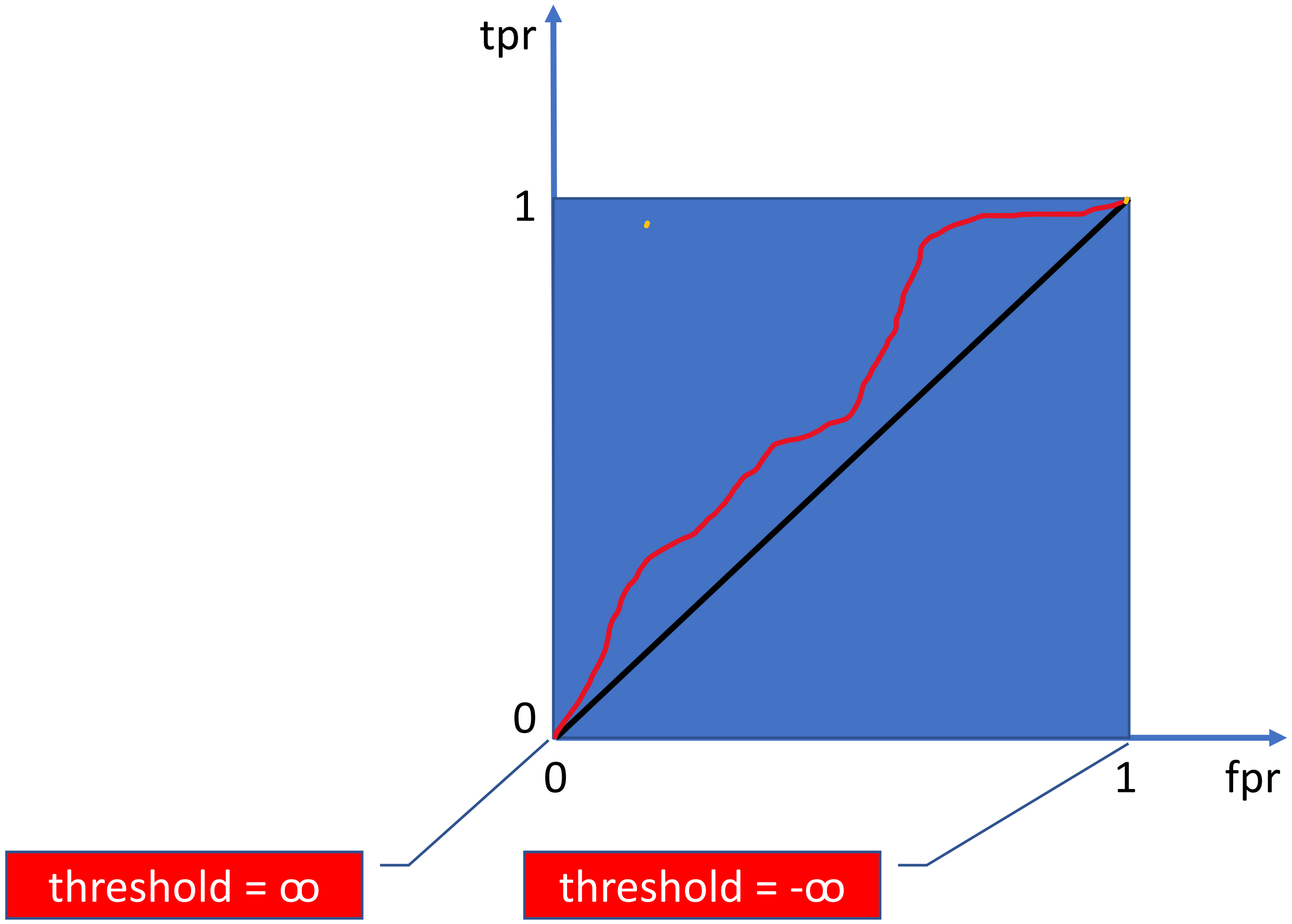


Remark:

ROC curves are monotonic (but not always convex)

because, when thresholds decrease, tpr increases

Threshold classifiers: ROC curves

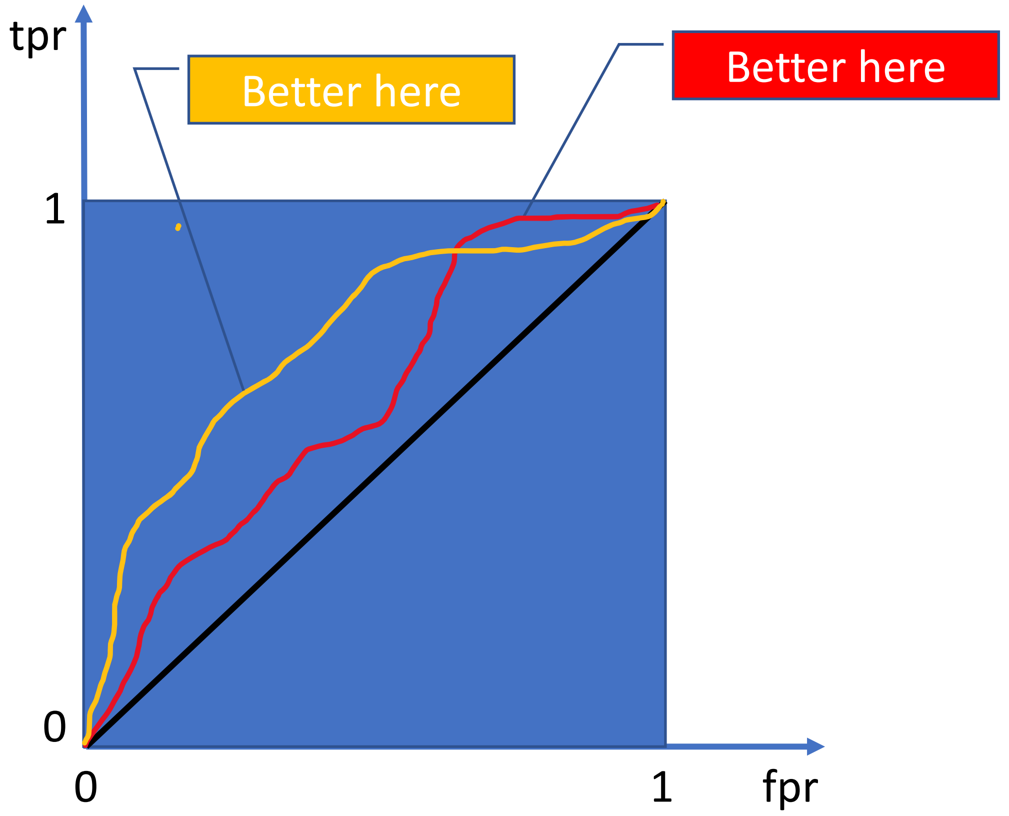


Remark:

ROC curves are monotonic (but not always convex)

because, when thresholds decrease, tpr increases

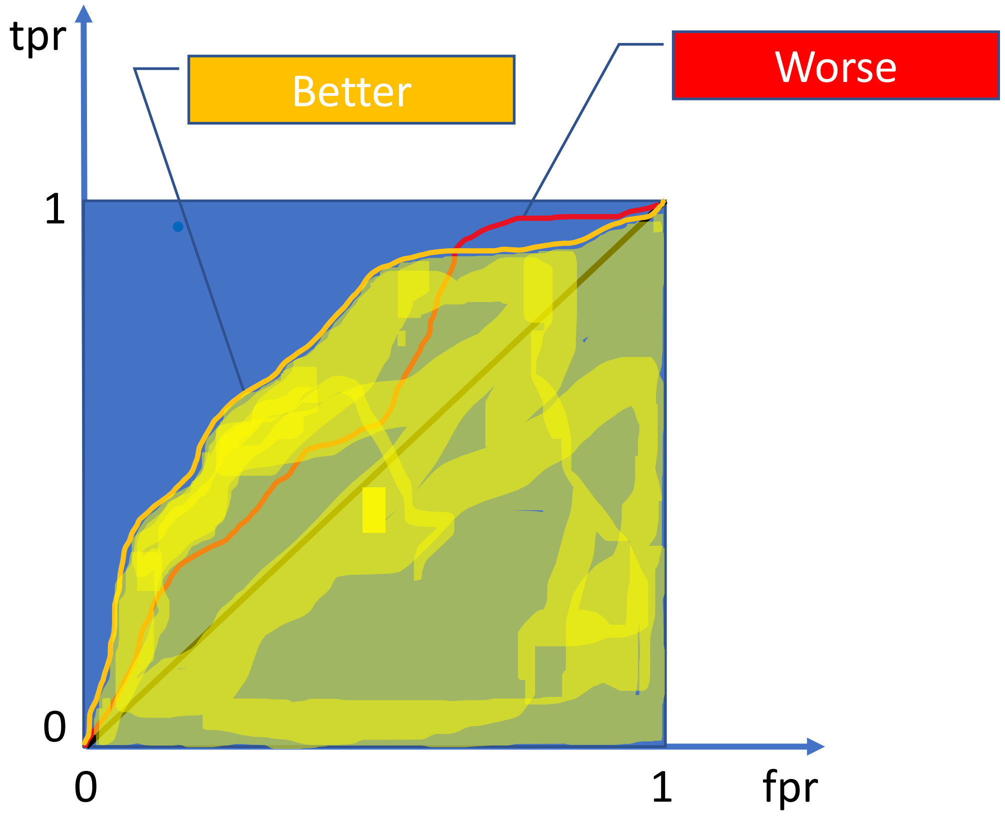
Comparing classifiers with ROC curves



ROC curves can be analyzed to compare classifiers, depending on the accepted fpr range

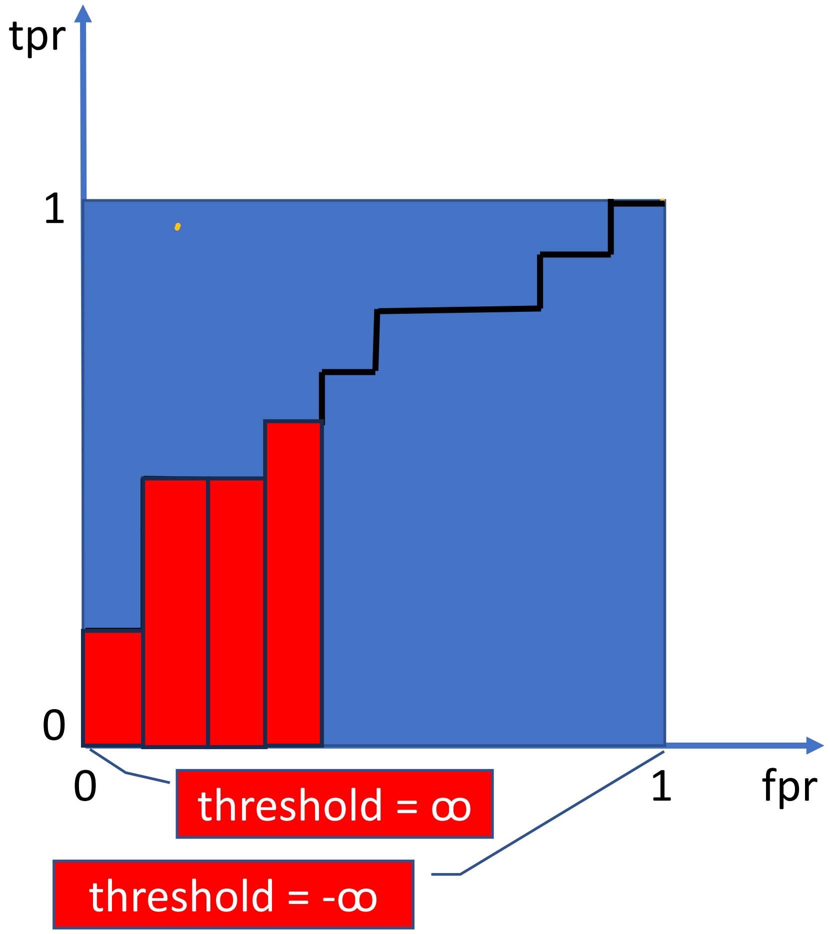
Area Under ROC curve (AUROC)

a single number for comparing different classifiers



Computing ROC-AUC (trapezoid metdhod)[1]

with O(|test set|*log(|test set|)) complexity

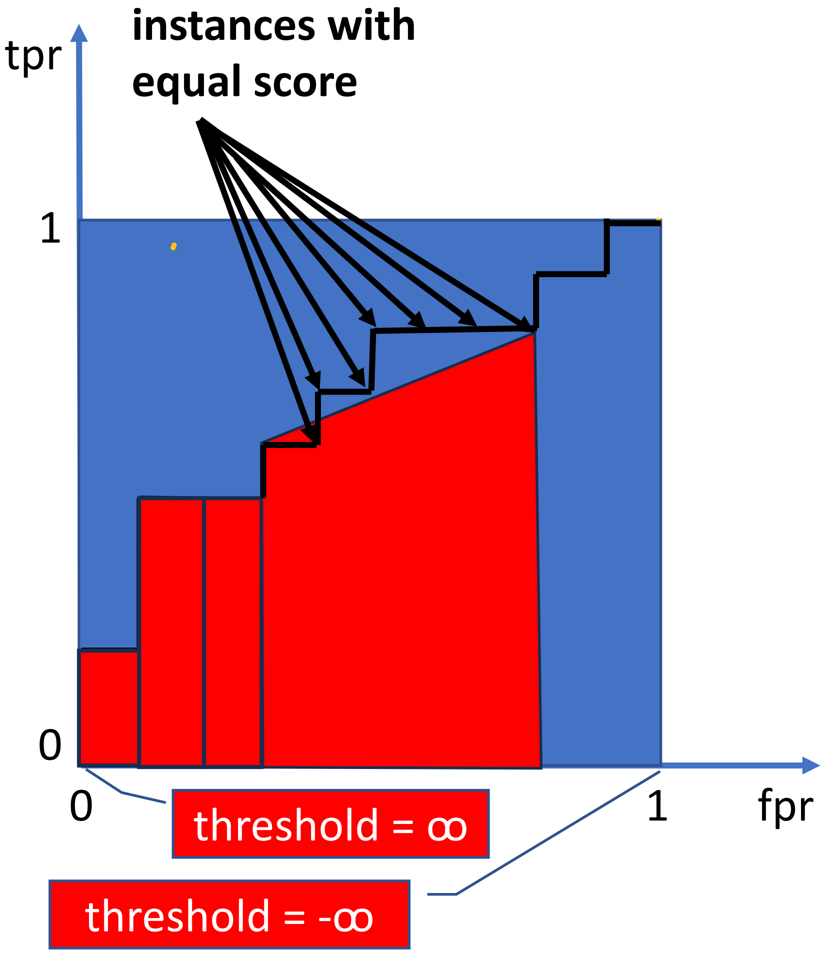


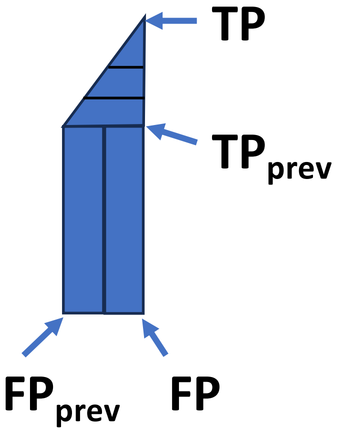
$FP=TP=0$; $FP_{prev}=TP_{prev}=0$;
$A=0$; $score_{prev}=-\infty$; $i=1$
for $i=1$ to |test set| // test set sorted by score
if score(i)≠ $score_{prev}$ then
A+=trapezoid(FP,FP_{prev},TP,TP_{prev})
$score_{prev}$=score(i)
FP_{prev}=FP; TP_{prev}=TP
if class(i)=1 then TP=TP+1
if class(i)=0 then FP=FP+1
A+=trapezoid(N,FP_{prev},P,TP_{prev})
Return A/(P*N)

$(FP-FP_{prev})*(TP+TP_{prev})/2$

[1] Tom Fawcett, "An introduction to ROC analysis", Pattern Recognition Letters 27, 2006

Computing ROC-AUC





$FP=TP=0$; $FP_{prev}=TP_{prev}=0$;
$A=0$; $score_{prev}=-\infty$; $i=1$
for $i=1$ to |test set| // test set sorted by score
if $score(i) \neq score_{prev}$ then
A+=trapezoid($FP, FP_{prev}, TP, TP_{prev}$)
$score_{prev}=score(i)$
$FP_{prev}=FP$; $TP_{prev}=TP$
if $class(i)=1$ then $TP=TP+1$
if $class(i)=0$ then $FP=FP+1$
A+=trapezoid($N, FP_{prev}, P, TP_{prev}$)
Return $A/(P*N)$

$(FP-FP_{prev})*(TP+TP_{prev})/2$

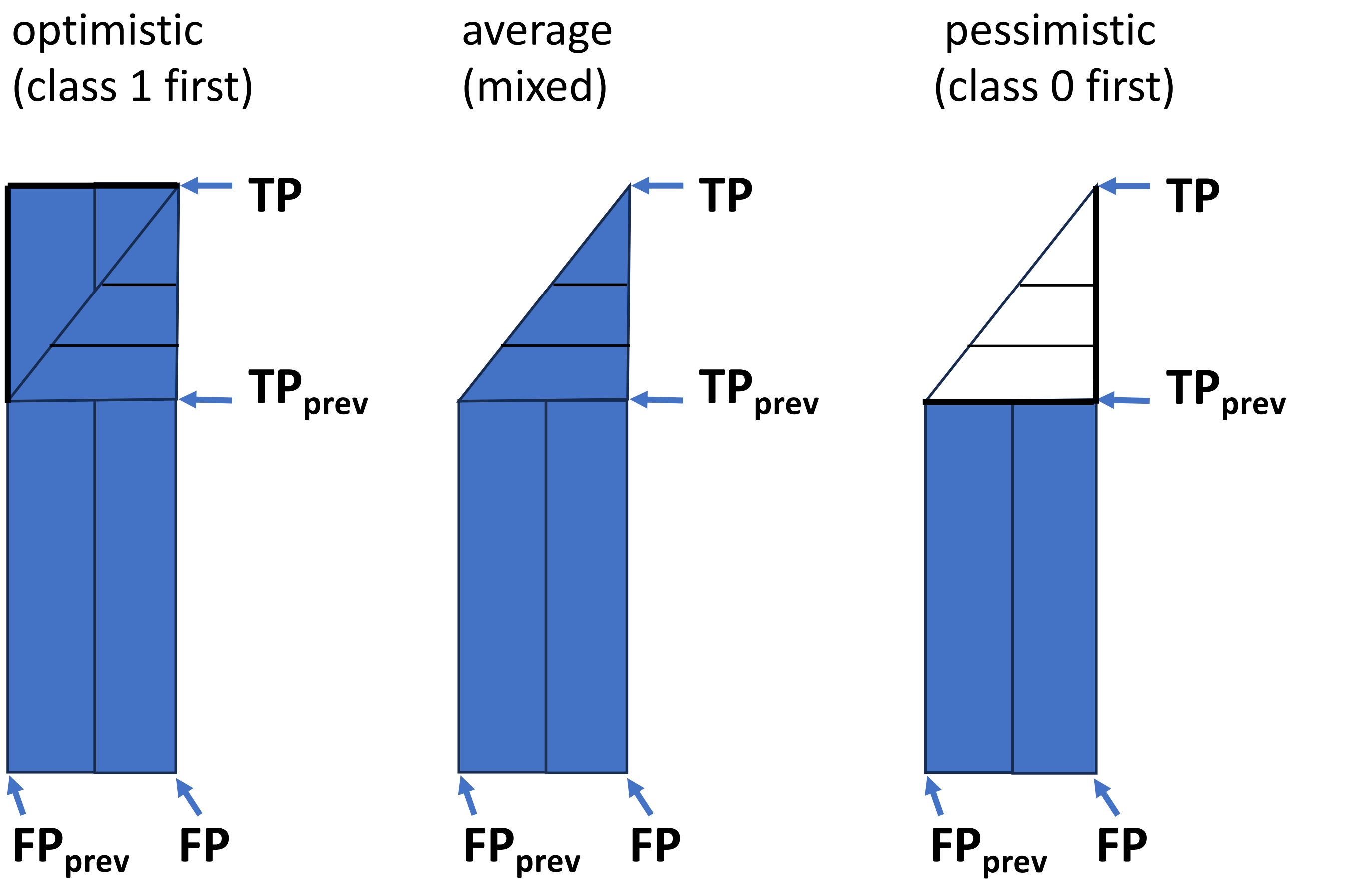
Handling instances with equal score

optimistic

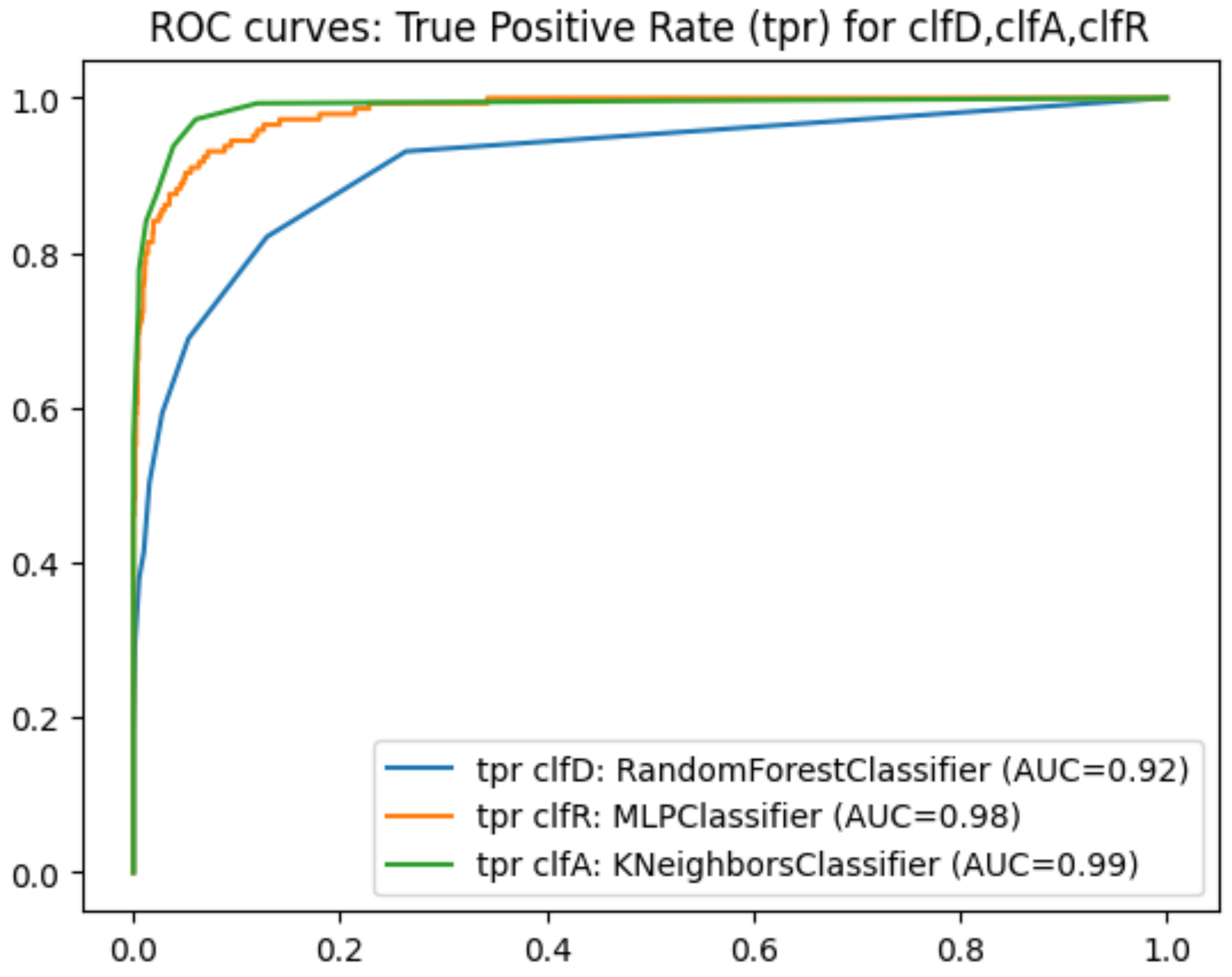
#n	class	score
i	1	.7
i+2	1	.7
i+3	1	.7
i+4	0	.7
i+5	0	.7

pessimistic

#n	class	score
i+4	0	.7
i+5	0	.7
i	1	.7
i+2	1	.7
i+3	1	.7



ROC curves obtained with scikit-learn and the digits dataset^



^ using digit 3 as anomaly, and all other digits as normal data

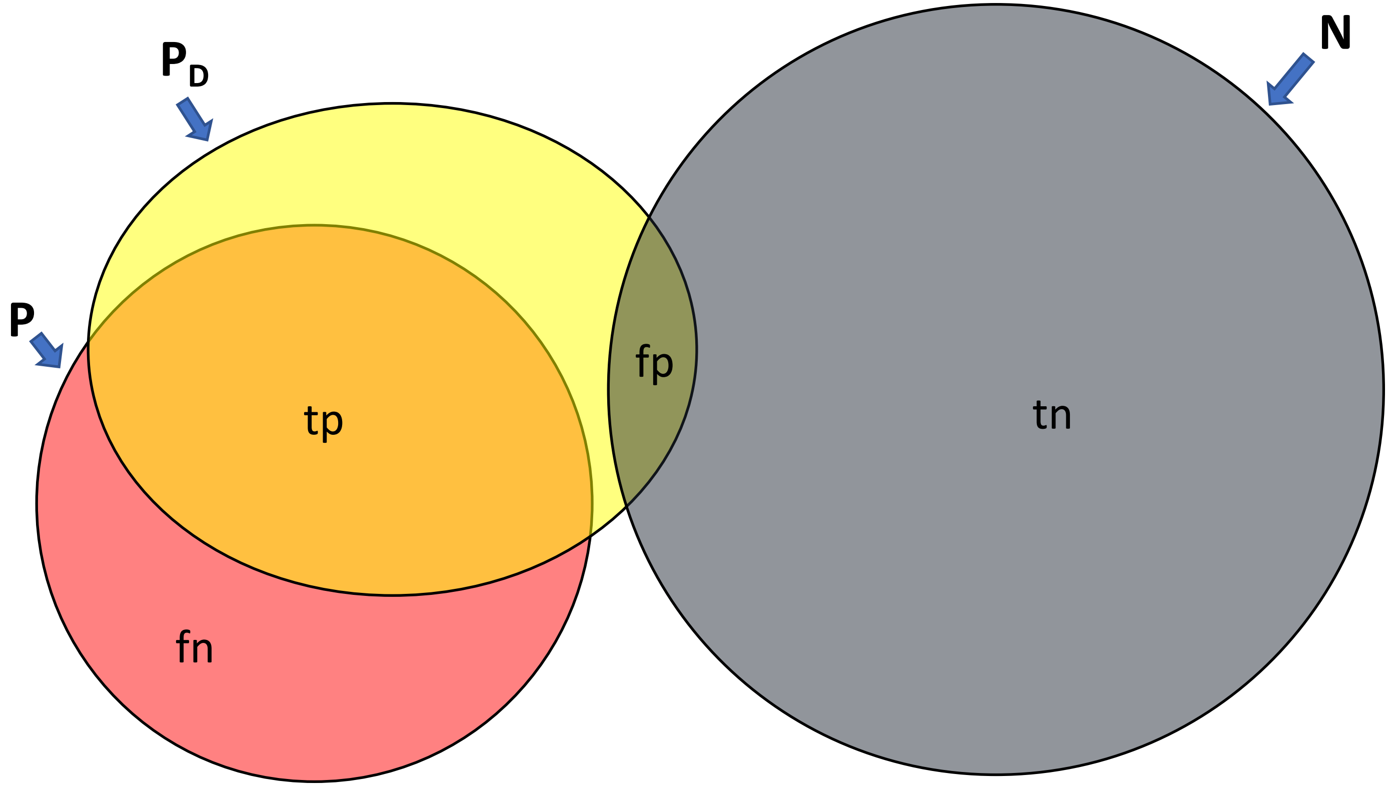
Exercises

1) Learn classifiers using scikit-learn
   - Download the digits data set using Scikit-lean, and re-label digit 3 as *anomaly* and all other digits as *normal*
   - Split the data set into a training set and a test set
   - Learn different models (e.g. random forest, mlp, knn)
2) Use the metrics module to evaluate ROC-AUC on the test set, for the learned models (see 1_ROCexamples.py)
3) Implement your own ROC-AUC computation, using the trapezoid method

Exercises, part 1)

```
clfA=KNeighborsClassifier(10)
clfD=RandomForestClassifier(max_depth=10,
                            n_estimators=10)
clfR=MLPClassifier(alpha=1, max_iter=1000)
#%% dataset preparation & modification
digits = datasets.load_digits()
n_samples = len(digits.images)
data = digits.images.reshape((n_samples, -1))
anomaly=3 #3 = anomaly, others = normal
for i in np.arange(digits.target.size):
    if digits.target[i]==anomaly:
                digits.target[i]=1
    else:       digits.target[i]=0
```

```
#prepare training&test, fit&predict
X_train, X_test, y_train, y_test =
   train_test_split(data, digits.target,
        test_size=0.8, shuffle=False)
clfD.fit(X_train, y_train)
clfA.fit(X_train, y_train)
clfR.fit(X_train, y_train)
predD = clfD.predict_proba(X_test)[:, 1]
predA = clfA.predict_proba(X_test)[:, 1]
predR = clfR.predict_proba(X_test)[:, 1]
```



accuracy = correct/total = (|tp|+|tn|)/(|P|+|N|)

tpr = recall = sensitivity = hit rate = |tp|/|P|

fpr = false alarm rate = |fp|/ |N|

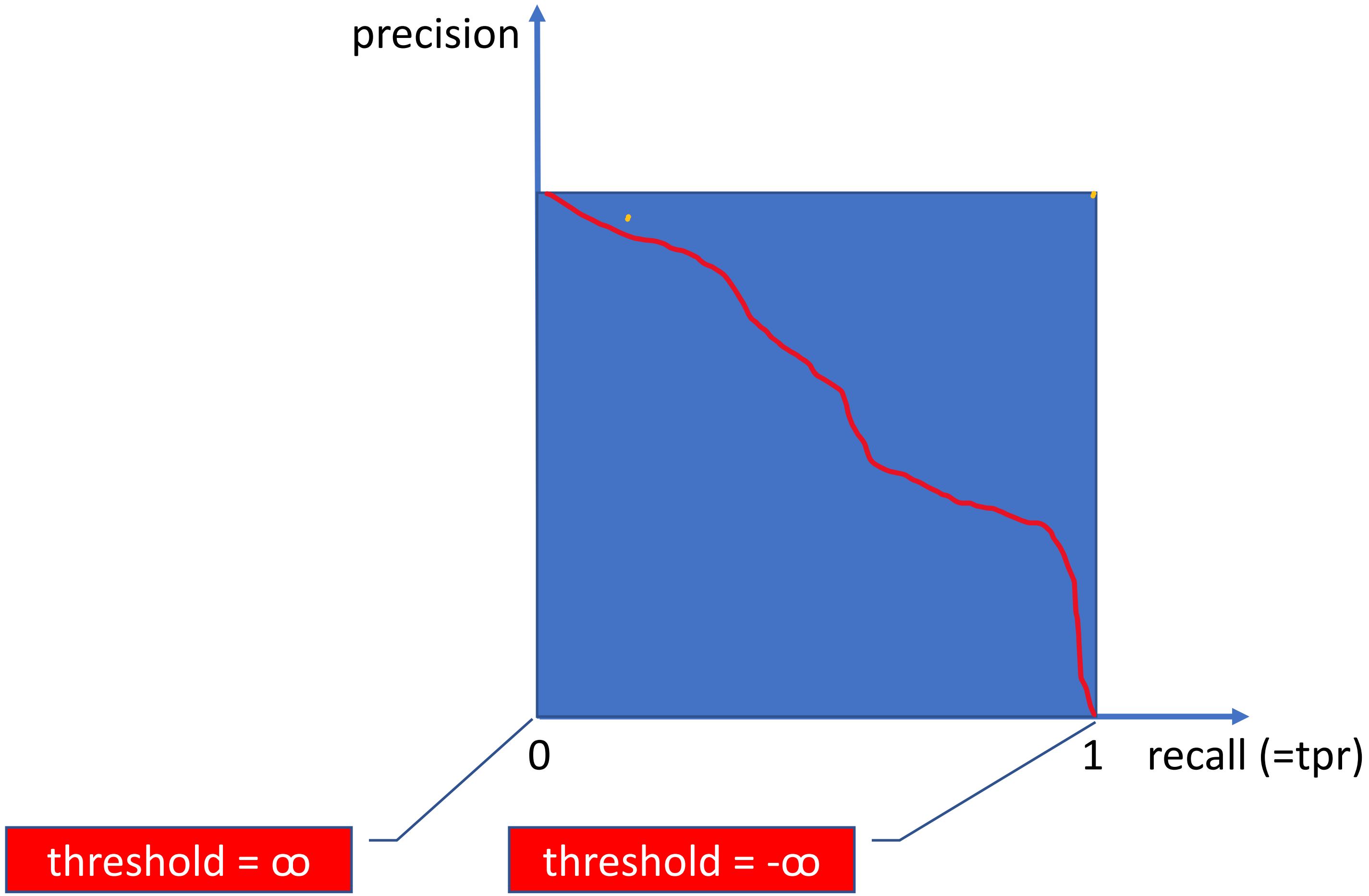
precision = positive predictive value =
= correctness when P_D=anomaly =
= |tp|/(|tp|+|fp|)

F-measure = F_1 score = 2/[(1/precision)+(1/recall)]

Accuracy, fpr and ROC curves may not work well
when classes are highly unbalanced (|N| >> |P|),
because fpr is marginally influenced by even large changes in |fp|

In this case we prefer to evaluate a precision vs recall trade-off, using PR curves

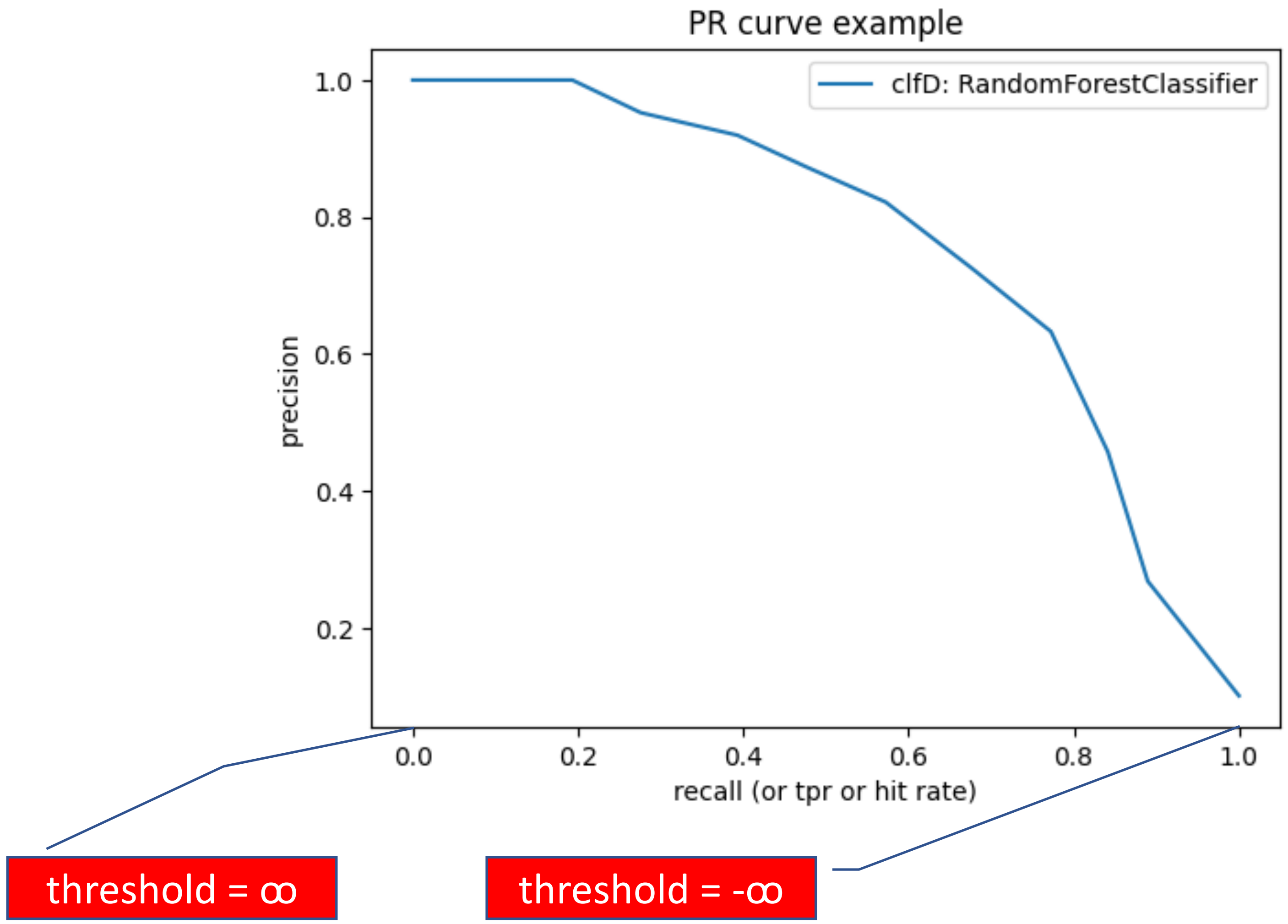
Precision Recall curves



Remark:

when the threshold decreases, recall increases because more anomalies are recognized, but precision tends to decrease as more normal cases may be classified as anomalies

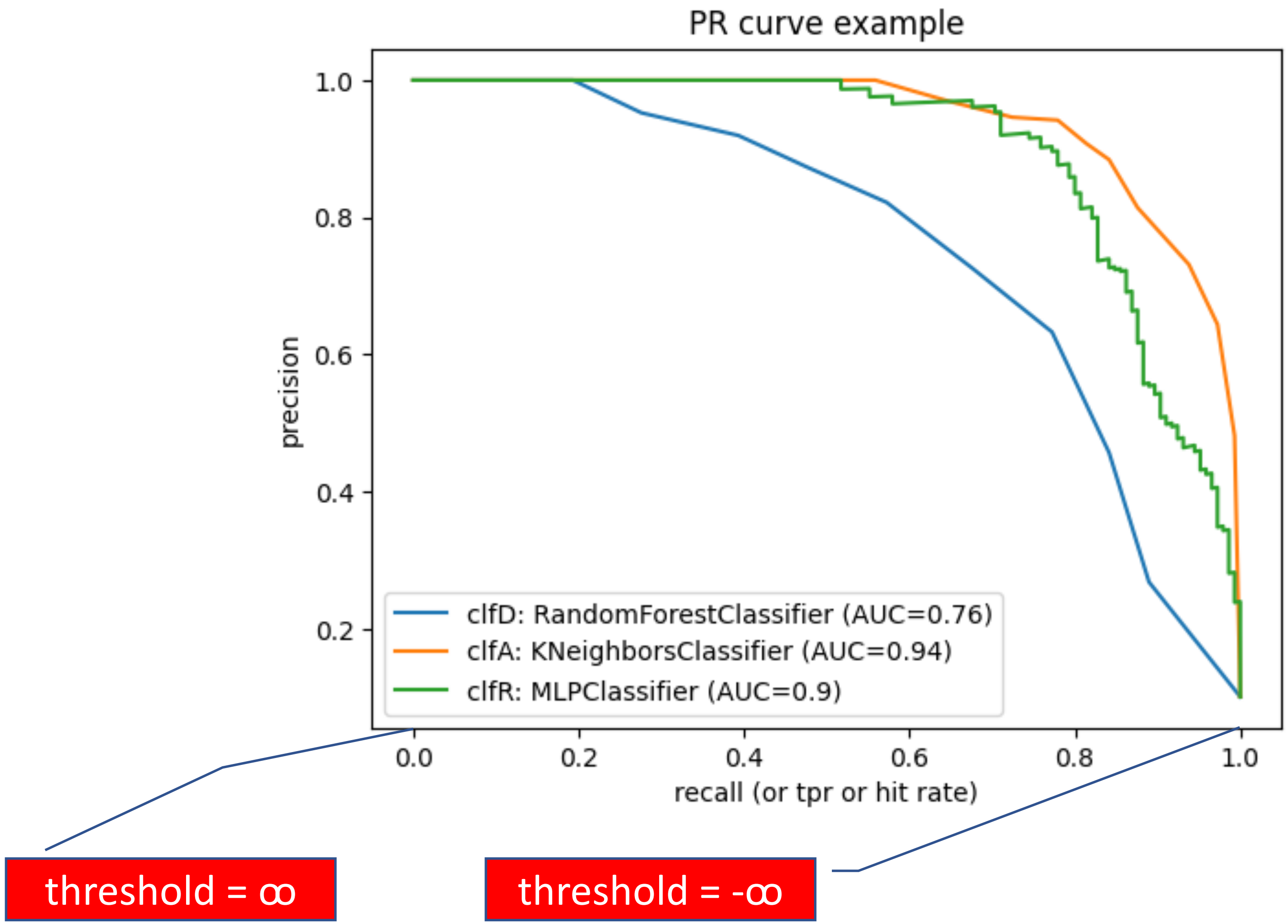
Precision Recall curves: examples



Remark

when the threshold decreases, recall increases because more anomalies are recognized, but precision tends to decrease as more normal cases may be classified as anomalies

Comparing classifiers with PR curves



Remark:

AUC can be computed with the trapezoid method as for ROC curves

Summary

1) Cybersecurity Applications of Machine Learning
2) Formalization and Metrics
3) ROC curve analysis

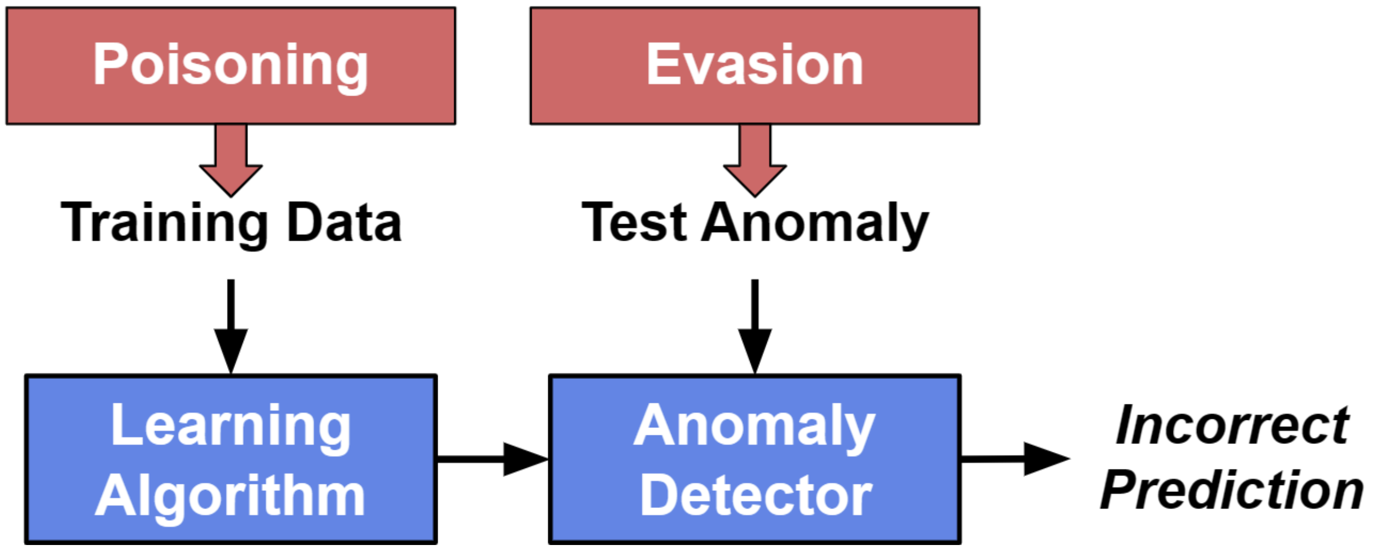
Adversarial evasion and defenses

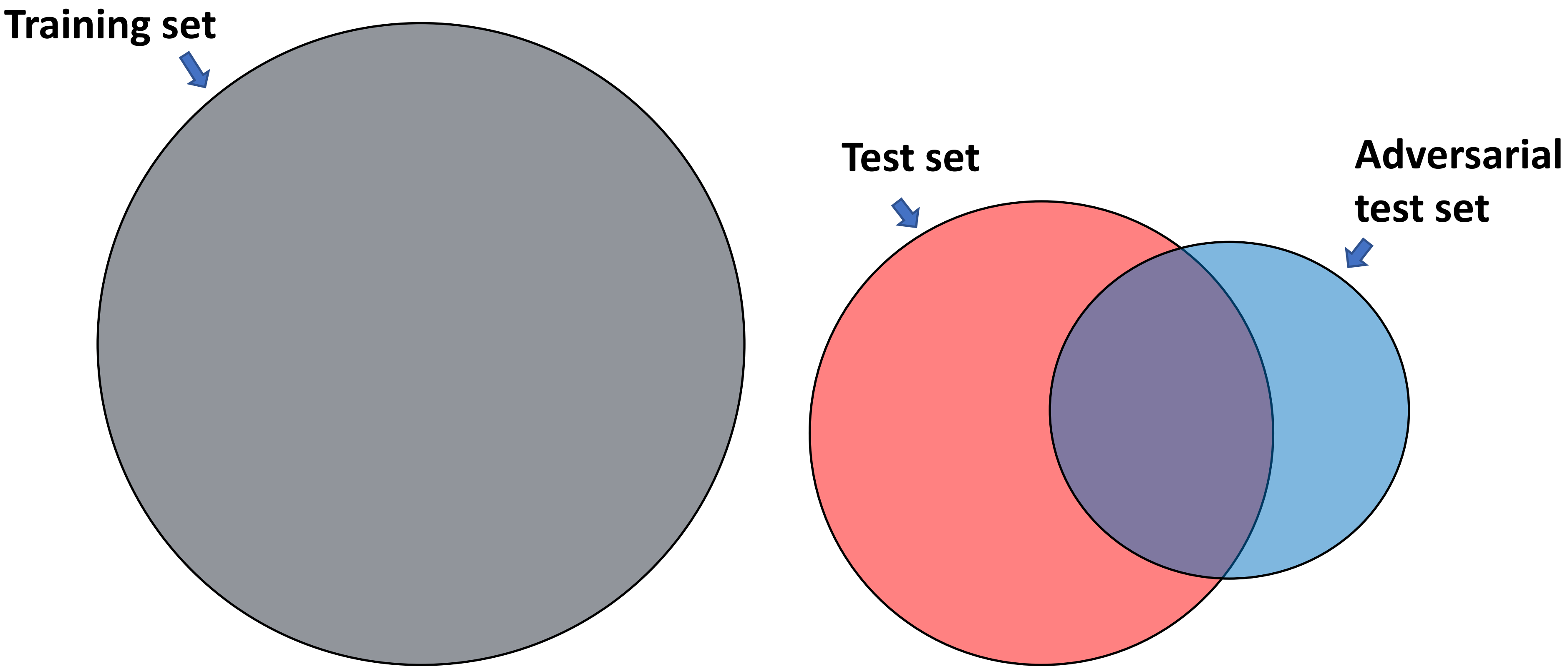
Previous work on:

- Adversarial Examples
- Evasion resistance metrics
- Randomization and keyed learning

Adversarial actions in anomaly detection contexts

- Data poisoning
- Evasion
  - Selection from an anomaly in the test set
  - Modificatoin of an anomaly in the test set

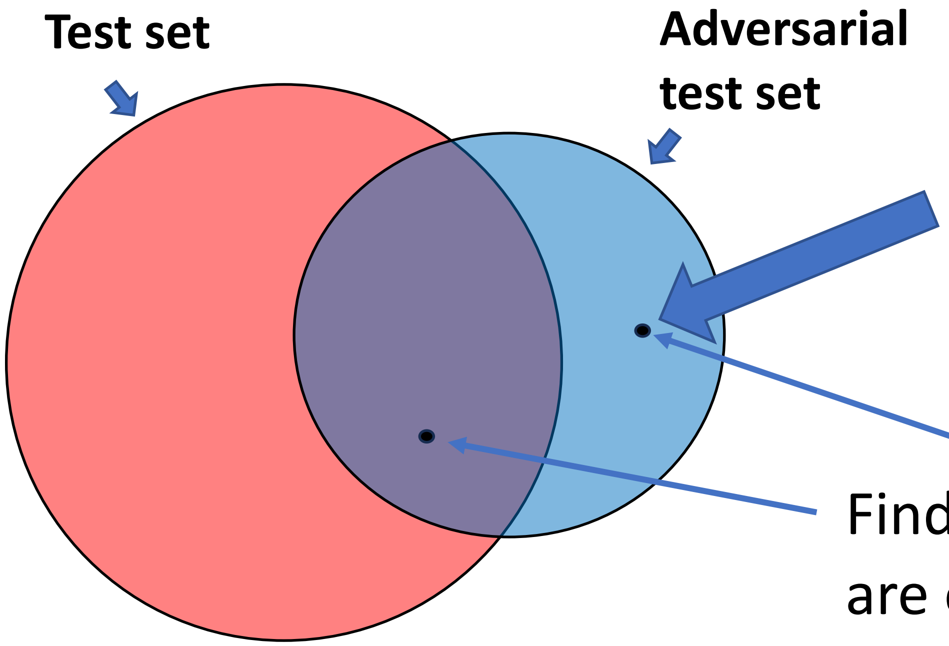




Evasion = adversarial attack at test time:

Training set is not modified
Test set is modified, as the adversary can:
- **select a subset of the test set**
- **modifiy some instances in the test set**

Adversarial objectives

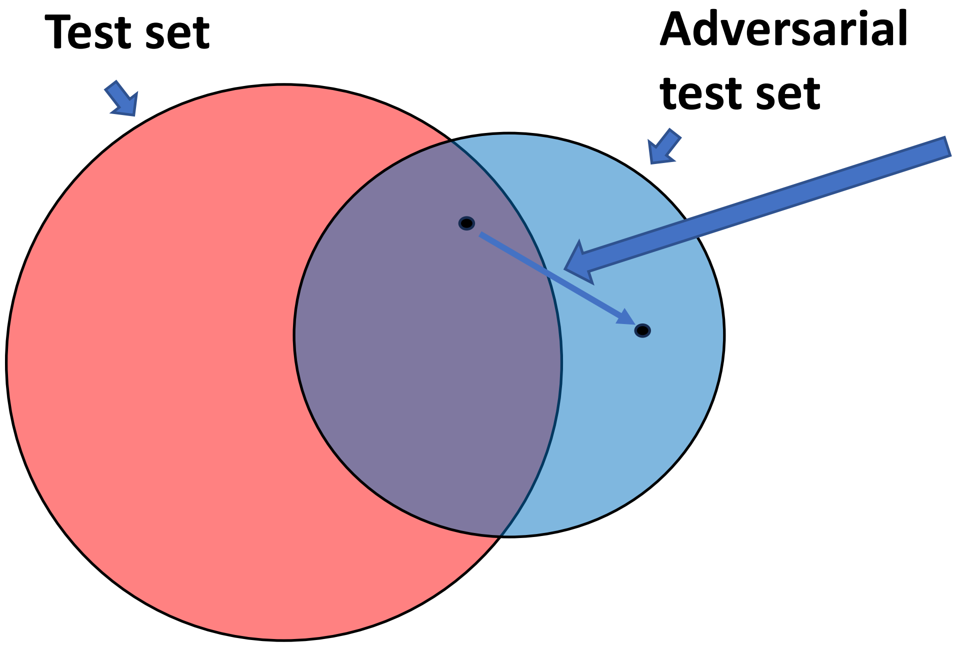


Most frequently investigated in the literature (from the adversary's perspective)

Find or create test instances <x,y> that are effective attacks but go undetected: y=1 (anomaly), but $C_D(x)$=0 (not detected)

Where C_D is the classifier learned by the defender using the training set.

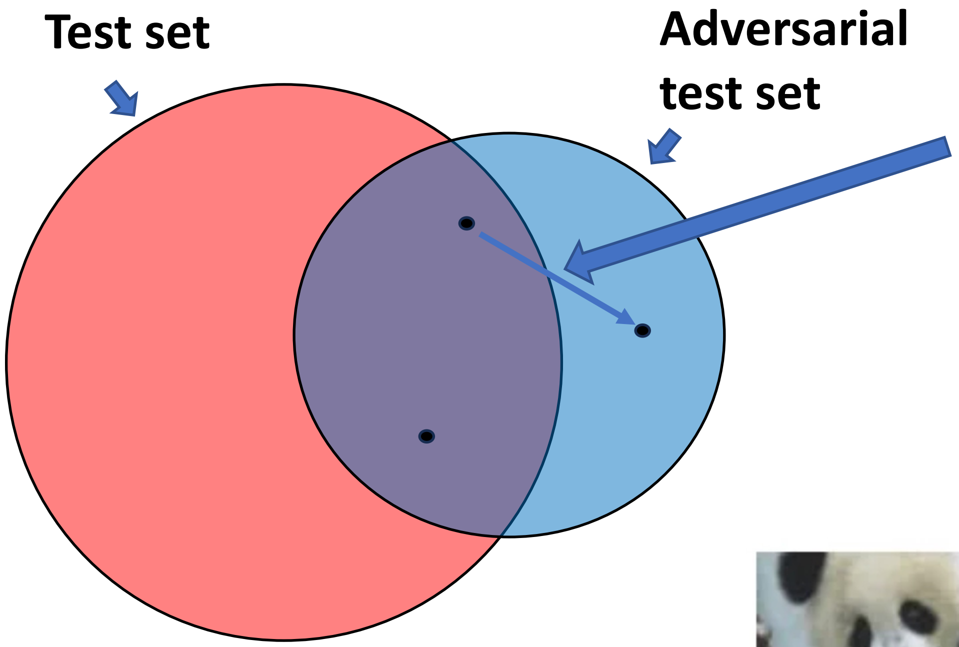
Adversarial examples



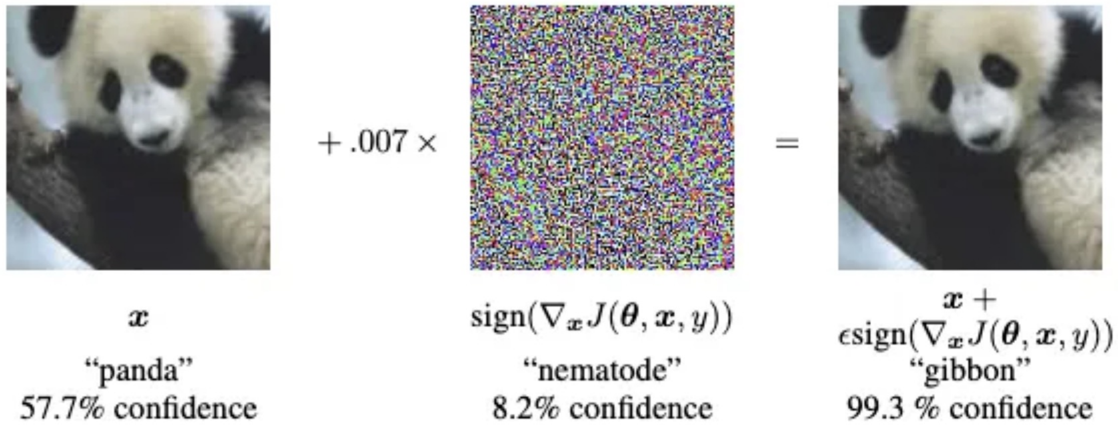
Most frequently investigated in the literature (from the adversary's perspective):

modify a test instance that is correctly classified as anomalous, into an apparently similar one that avoids detection

Adversarial examples

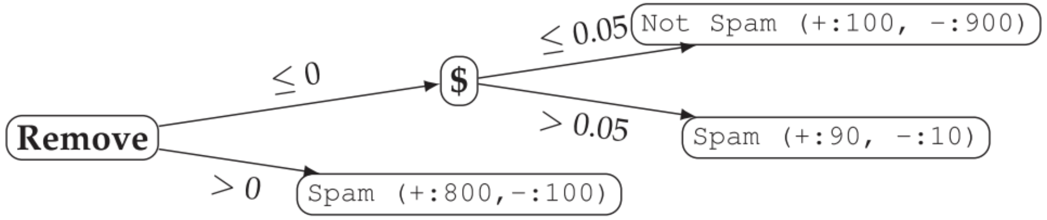


modify a test instance that is correctly classified as anomalous, into an apparently similar one that avoids detection

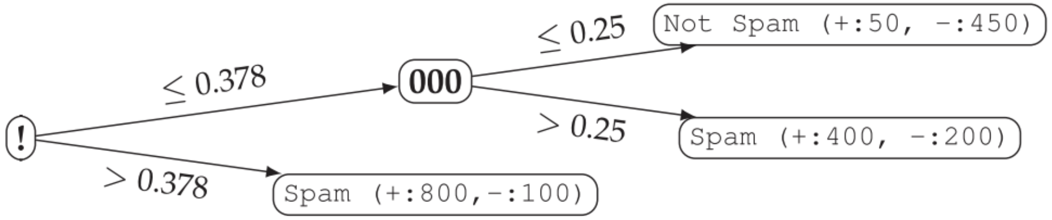


[2] I. J. Goodfellow, J. Shlens & C. Szegedy. Explaining and harnessing adversarial examples, Proc. ICLR 2015.

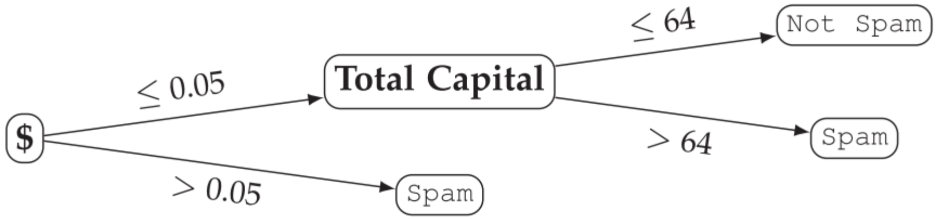
Adversarial examples in discrete domains [3]



(a) Model f_1



(b) Model f_2



(c) Model f_3

Remove	$	!	000	Total Capital
0	0.2	0.4	0.3	100

(a) Attacker's original spam email

Remove	$	!	000	Total Capital
0	**0.05**	0.4	0.3	100

(b) Modified email to trick f_1 (bold as changes)

Remove	$	!	000	Total Capital
0	**0.05**	0.4	0.3	**64**

(c) Modified email to trick at least two models in Figure 1

Remove	$	!	000	Total Capital
0	**0.05**	**0.378**	**0.25**	100

(d) Modified email to trick f_1 and f_2

[3] Fan Yang, Zhiyuan Chen, and Aryya Gangopadhyay
Using Randomness to Improve Robustness of Tree-Based Models Against
Evasion Attacks IEEE Transactions on Knowledge and Data Engineering, 2022

Cost of adversarial examples

We could assign a weigth to each feature, e.g.

w(Remove)=0.2, w($)=0.1, w(!)=0.1,

w(Total Capital)=0.4, w(000)=0.2

When feature F_i of e (e_i) is changed to x, the

cost is $w(F_i)*|x - e_i|/(max(F_i)-min(F_i))$

Suppose max(Total Capital = 200),

max($)=max(!)=max(000)=max(Remove)=1

then

Cost(b)=(.2-.05)*w($)=.015 (tricks only 1 tree)

Cost(c)=(.2-.05)*w($)+

w(Total Capital)*(100-64)/200=.087 (high cost)

Cost(d)=(.2-.05)*w($)+(.4-.378)*w(!)+

(.3-.25)*w(000)=.15*.1+.022*.1+.05*.2=.0272

Remove	$	!	000	Total Capital
0	0.2	0.4	0.3	100

(a) Attacker's original spam email

Remove	$	!	000	Total Capital
0	**0.05**	0.4	0.3	100

(b) Modified email to trick f_1 (bold as changes)

Remove	$	!	000	Total Capital
0	**0.05**	0.4	0.3	**64**

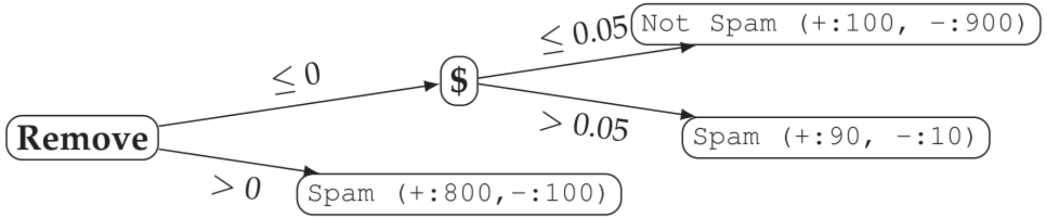
(c) Modified email to trick at least two models in Figure 1

Remove	$	!	000	Total Capital
0	**0.05**	**0.378**	**0.25**	100

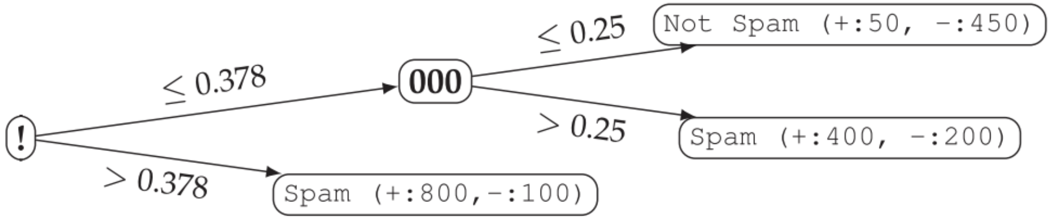
best

(d) Modified email to trick f_1 and f_2

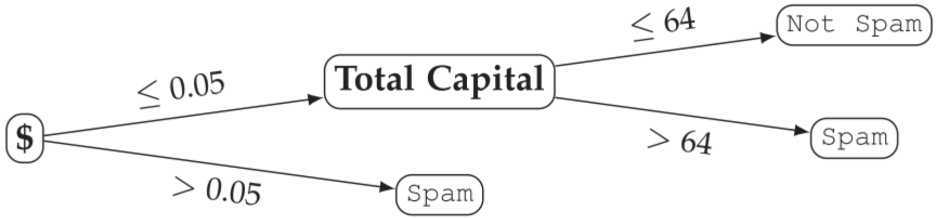
Adversarial example generation (instance based greedy search)



(a) Model f_1



(b) Model f_2



(c) Model f_3

Remove	$	!	000	Total Capital
0	0.2	0.4	0.3	100

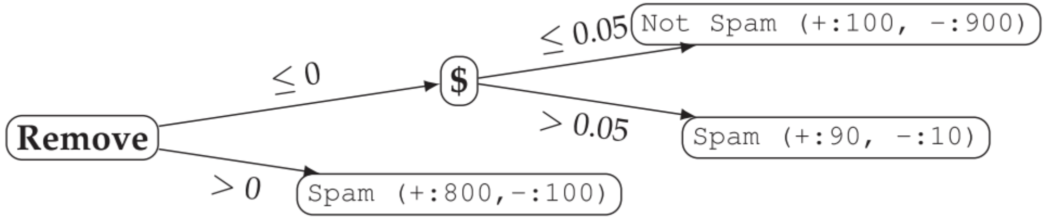
(a) Attacker's original spam email

Input: an anomalous example e and a classification forest F, where each node holds exclusive binary conditions
Output: an adversarial anomaly a(e), misclassified by F, and the cost of transforming e into a(e)

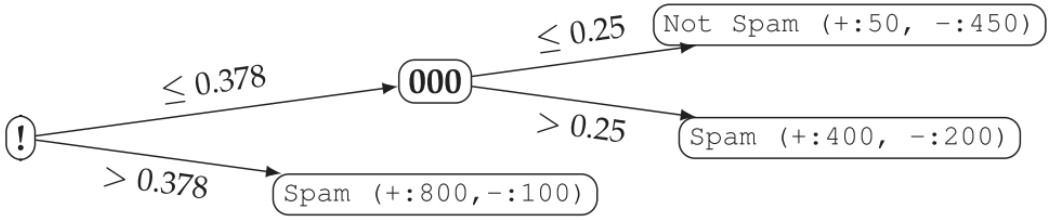
x = instance with the lowest positive score, computed as the number of trees in F classifying x as an anomaly
cost=0;
while more than 50% of trees in F classify e as an nomaly:
  select feature i so that changing e_i to x_i has minimum cost C_i and maximum benefit (number of trees in F that no longer classify e as an anomaly)
  $e_i = x_i$; cost+=C_i
return e,cost

[4] A. Kantchelian, J. D. Tygar, A. Joseph. Evasion and Hardening of Tree Ensemble Classifiers, ICML 2016

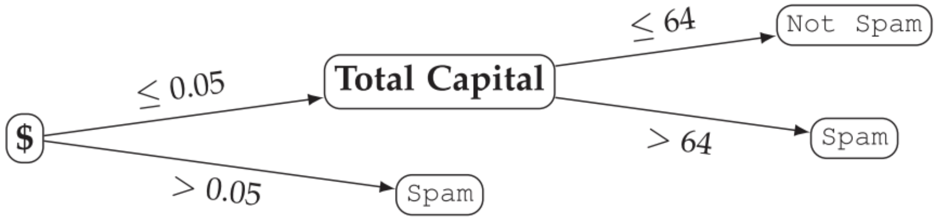
Adversarial example generation (model based greedy search)



(a) Model f_1



(b) Model f_2



(c) Model f_3

Remove	$	!	000	Total Capital
0	0.2	0.4	0.3	100

(a) Attacker's original spam email

Input: an anomalous example e and a classification forest F, where each node holds exclusive binary conditions

Output: an adversarial anomaly a(e), misclassified by F, and the cost of transforming e into a(e)

{T1, …, Tn} = random permutation of trees in F

e_1=e; cost=0; cond=true;

for j=1 to $\lceil n/2 \rceil$:

  for each path P_i in T_j from root to a 'non-anomaly' leaf:

    $cond_i$=the conditions in the path P_i

    C_i = cost for tranforming e_j into $a_{i,j}$ so that cond&$cond_i$ = true

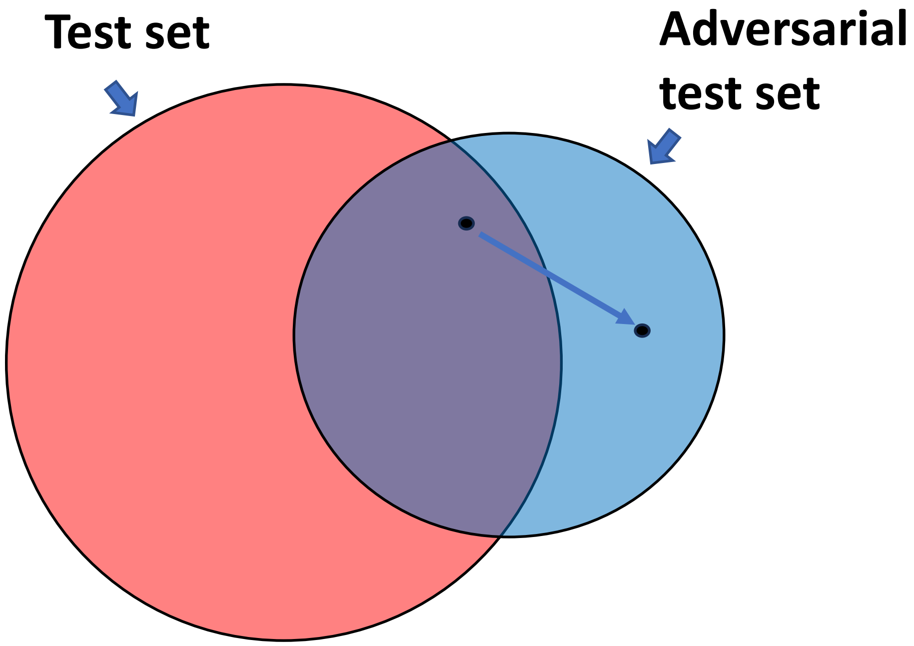
  i_min = $argmin_i(C_i)$; cond = cond and $cond_{i\_min}$; cost += C_{i_min}

  $e_{j+1} = a_{i\_min,j}$

return $e_{\lceil n/2 \rceil+1}$, cost

Exercise: simulate the 2 algorithms on the above forest and spam email, generating an adversarial spam email

Evasion resistance metrics

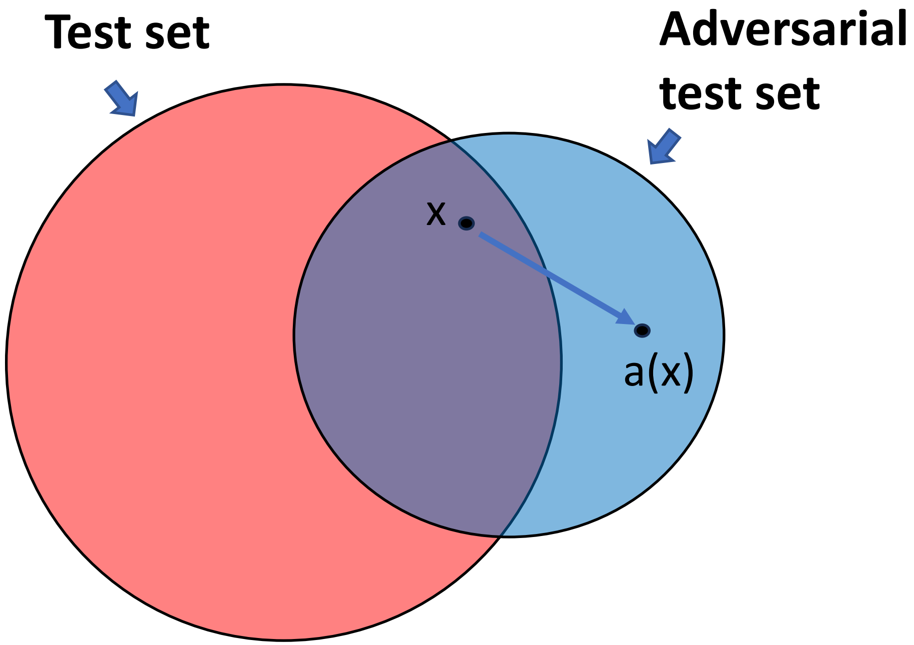


Learn evasion resistant classifiers, that make it diffult for the adversary to evade detection

Research question:

How do we measure evasion resistance?

Evasion resistance metrics (A)



[4] Dalvi, N., Domingos, P., Mausam, Sanghai, S., Verma, D.: Adversarial classification. In Proc. ACM Int. Conf. Knowledge Discovery Data Mining, 2004

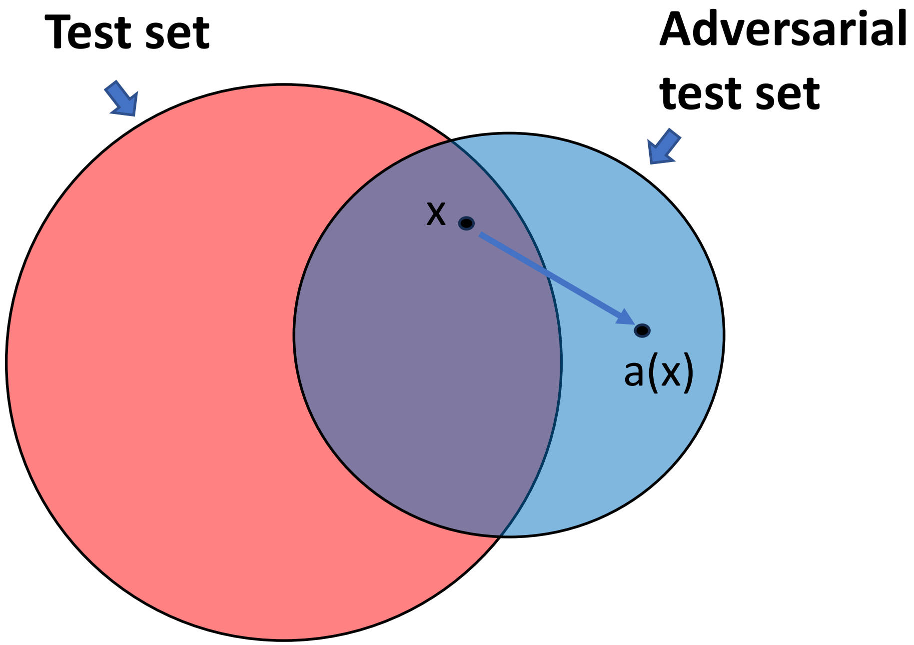
[5] Biggio, B., Fumera, G., Roli, F. (2008). Adversarial Pattern Classification Using Multiple Classifiers and Randomisation. Springer LNCS 5342, 2008

[6] Biggio, B., Corona, I., Maiorca, D., Nelson, B., Srndic, N., laskov, P., Giacinto, G., Roli, F., Evasion Attacks against Machine Learning at Test Time, arxiv.org/abs/1708.06131v1, 2017

Adversarial Utility $U_A(x,a(x))$ = gain – cost
where gain = advantage gained by transforming anomaly x into a(x), maximum if $C_D(x)=1$ and $C_D(a(x))=0$
and cost = cost of the transformation of x into y (e.g., computational, or loss of anomaly's effectiveness)

Defender's goals: (1) robustness (minimize adversarial utility) and (2) accuracy

Evasion resistance metrics (B)



[3] Fan Yang, Zhiyuan Chen, and Aryya Gangopadhyay Using Randomness to Improve Robustness of Tree-Based Models Against Evasion Attacks, IEEE TKDE, 2022

[7] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein. Square attack: a query-efficient black-box adversarial attack via random search. Euro Conf. on Computer Vision, 2020.

[8] J. Chen and Q. Gu. Rays: A ray searching method for hard-label adversarial attack. In ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2020.

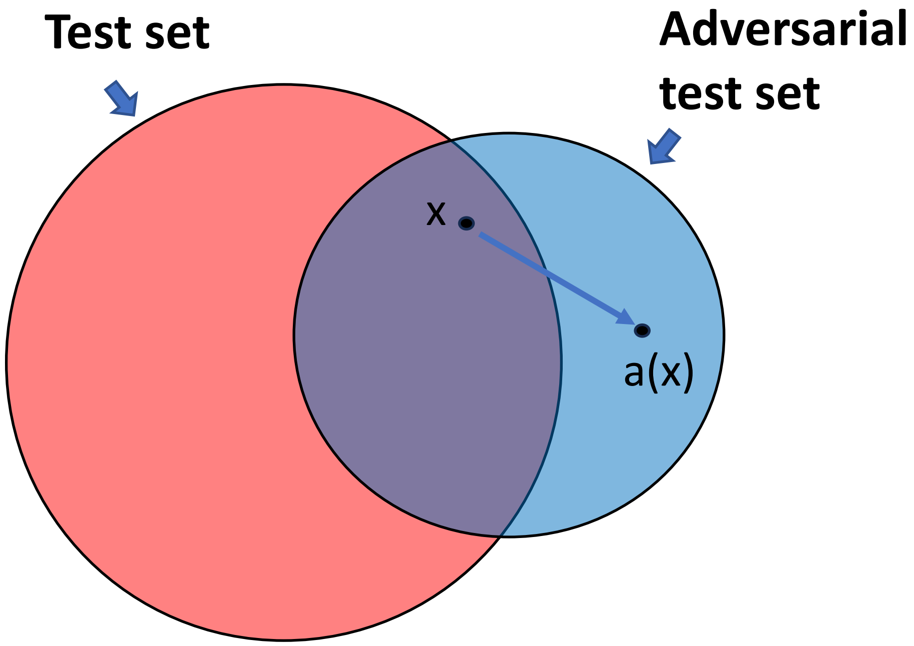
Measure the cost needed for evasion in terms on an L_p distance d: $d(x, a(x)) = \sum_{j=1}^{m} c_j |x_j - a(x)_j|^{1/p}$

where c_j is the cost of the j-th feature and

x_j is the value of the j-th feature for test instance x

Defender's goals: (1) robustness (maximize adversary's cost) and (2) accuracy

Evasion resistance metrics (C)



[3] Fan Yang, Zhiyuan Chen, and Aryya Gangopadhyay Using Randomness to Improve Robustness of Tree-Based Models Against Evasion Attacks, IEEE TKDE, 2022

[9] Jing Wu, Mingyi Zhou, Ce Zhu, Yipeng Liu, Mehrtash Harandi, and Li Li. Performance evaluation of adversarial attacks: Discrepancies and solutions. ArXiv 2104.11103, 2021.

Measure adversarial evasion success rate when a maximum cost budget B is allowed

Defender’s goals: (1) robustness (minimize adversarial success rate) and (2) accuracy

Evasion resistance methods

1) Randomization: make the learned classifier unpredictable for the adversary
2) Retrain after adding adversarial examples [10]
3) Defensive distillation [11]

- During learning
  - Random training subset
  - Random feature selection
  - Ensemble learning
- Post-learning
  - Add random noise
  - Select a random sub-ensemble

[10] H. Lee et al., "Generative adversarial trainer: defense to adversarial perturbations with GAN", ArXiv 1705.03387, 2017
[11] N. Papernot et al., "Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks", IEEE Symposium on Security & Privacy, IEEE 2016

Learning time randomization

- Random training subset [5]
- Random features / parameters
  - Secret feature set [12]
  - Initial weights of a neural network [5]
- Multiple classifiers & Ensemble Learning
  - Random weight sets for SPAM assassin filters, generated via SVMs [5]
  - Weighted random forests [3]

[12] M. Barreno et al., “Can Machine Learning be Secure?”, ASIA CCS 2006

Post-learning (test time) randomization

- Add bounded random noise [12]
  - Binary classifiers (randomly flip classification)
  - Threshold classifiers (add random value to score)
- Select a random sub-ensemble
  - Randomly select some trees in the learned random forest [3]

Randomization using “keys”

Keyed Intrusion Detection

[1] J. E. Tapiador et al., Key-recovery attacks on KIDS, a keyed anomaly detection system, IEEE Trans. Dependable Secure Comput., 2015

[2] R. Bendale et al., KIDS: Keyed Anomaly Detection System, Int. J. Adv. Eng. Res. Dev., 2017

Keyed Learning

[3] F. Bergadano. “Keyed learning: An adversarial learning framework”, ETRI Journal 41 (5), 608-618, 2019

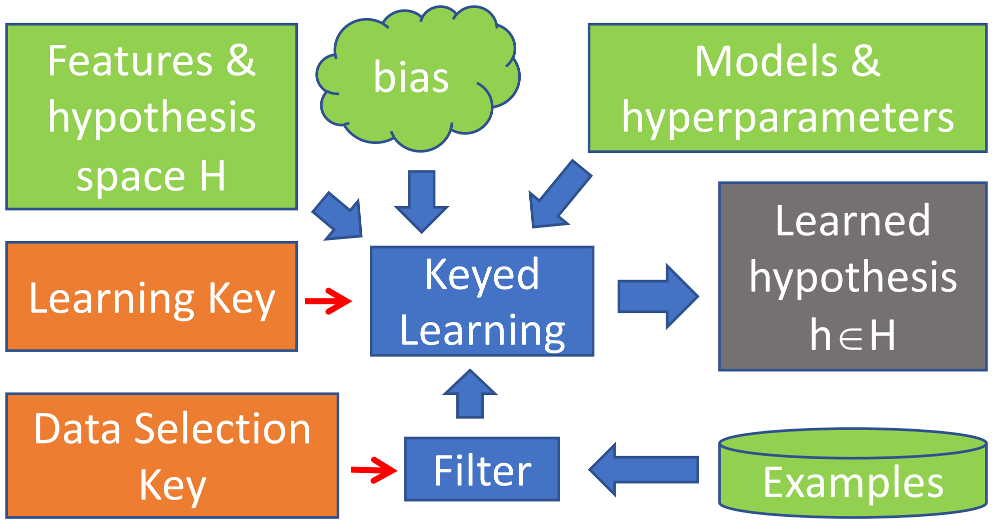
Keyed learning

- Keyed Learning = Machine Learning with a (secret) key

- Why: because we do not want the adversary to replicate learning and predict our decisions

- How: use the key to generate secrets, and use them every time some decision is needed during the learning process

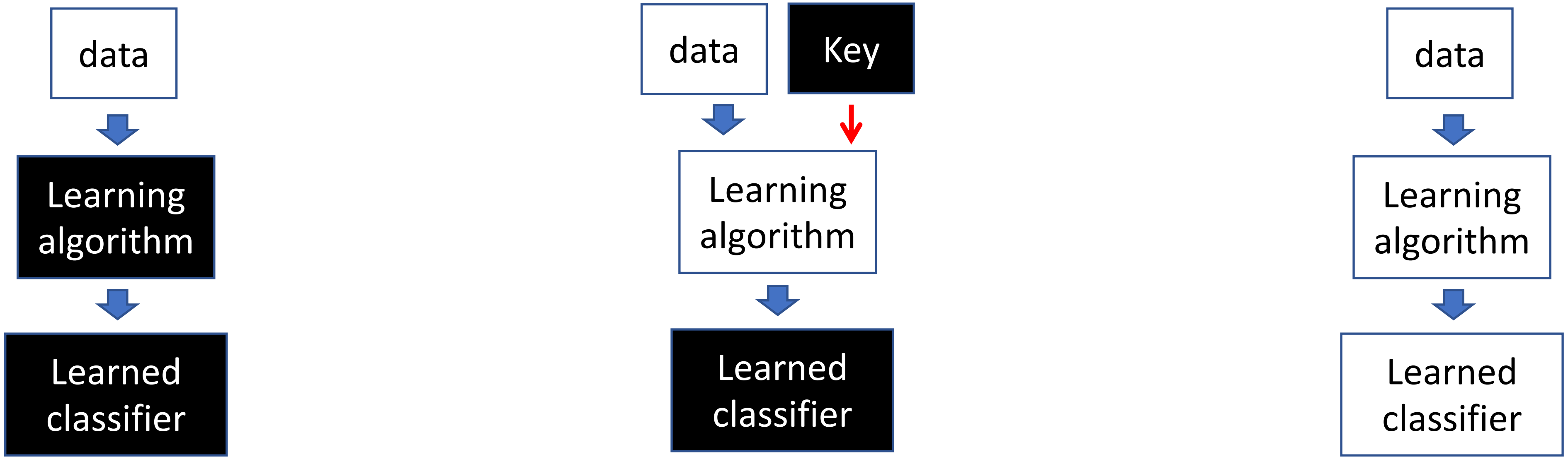
Using a key while learning



1) Selection of the training examples
2) Selection of models & parameters
3) Selection of features & H

Keyed learning in anomaly detection and Kerchoff’s principle

Black box | **Keyed box** | **White box**



security through obscurity (sto)

secret limited to key

vulnerable to learning replay

Summary

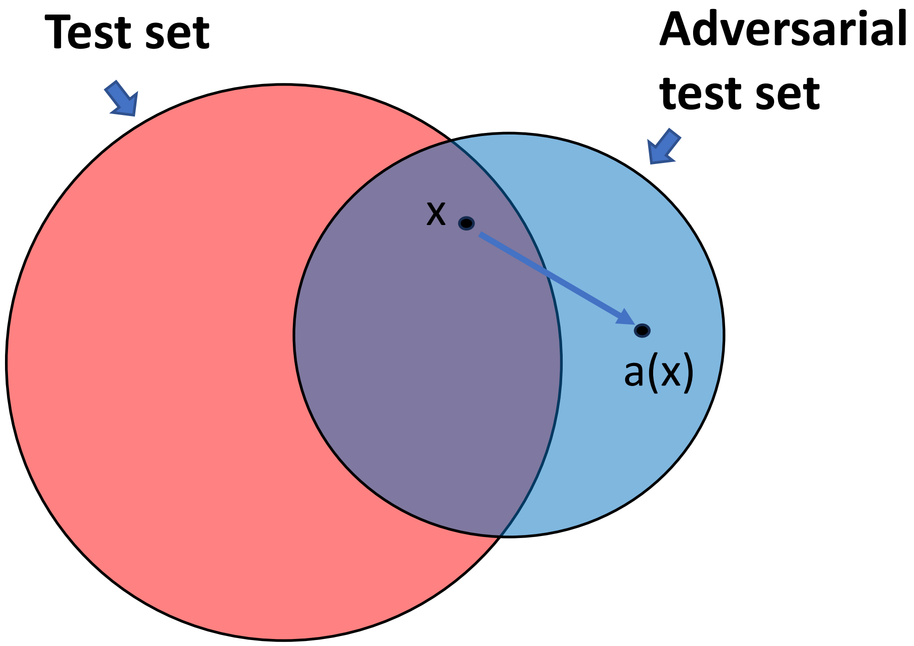
- types of adversarial attacks, and evasion & adversarial examples
- evasion resistance metrics
- evasion resistance methods, esp. randomization
- general notion of keyed learning
- «keyed box» threat model

New research on evasion resistance

F. Bergadano, S. Gupta, B. Crispo

1) Metrics:
   - adversarial failure rate (afr)
   - adversarial failure curves
   - AFR-AUC (area under the curve)
2) Randomization
   - Trainset size pinning
   - Model matrix
3) IDS application

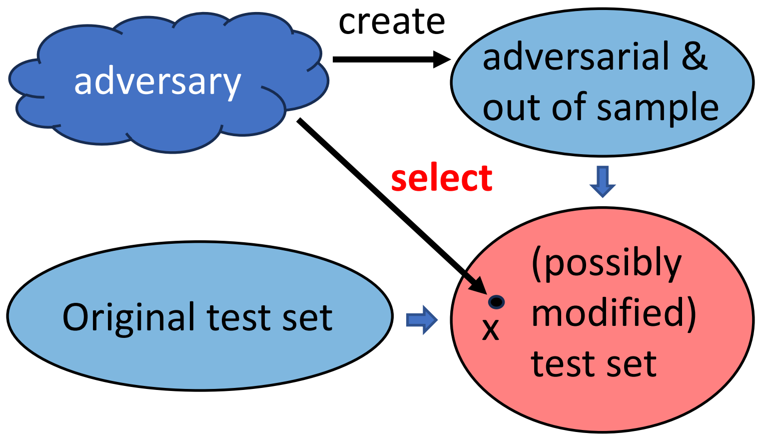
Shortcomings of known evasion resistance measures



(1) We do not know, in practice, how an adversary will behave, and producing an adversarial test set often requires arbitrary and artificial assumptions:
real-world adversarial test sets do not exist

(2) Previous studies do not consider the fact that, in most anomaly detection applications, C_D is a threshold function. Hence some form of **ROC-curve analysis would be appropriate**

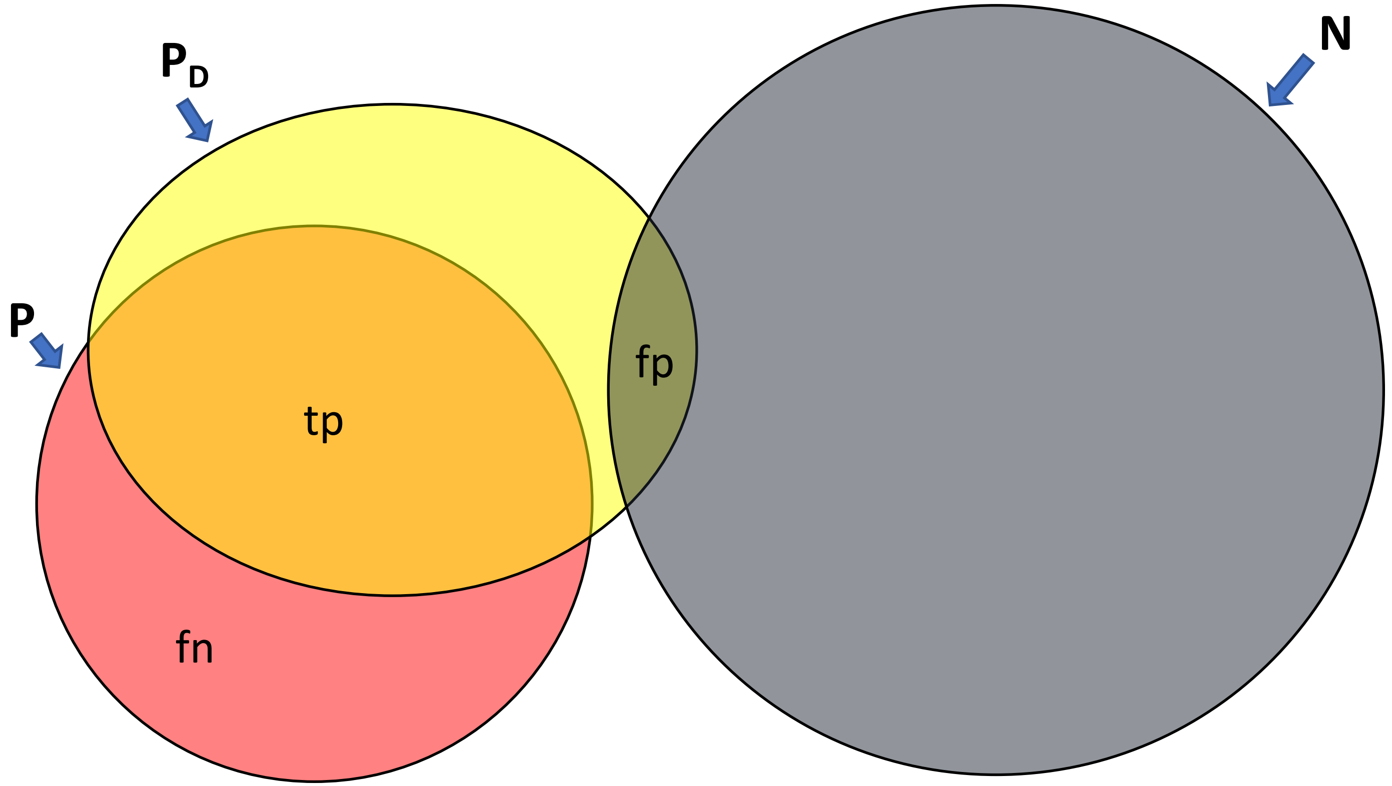
Adversarial test set generation: a different perspective



Evaluate, on the possibly modified test set:

(1) ROC-AUC
and
(2) afr-AUC*

*adversarial failure rate



P = positives (anomalies)
N = negatives (normal data)
P_D = classified as positive by defender

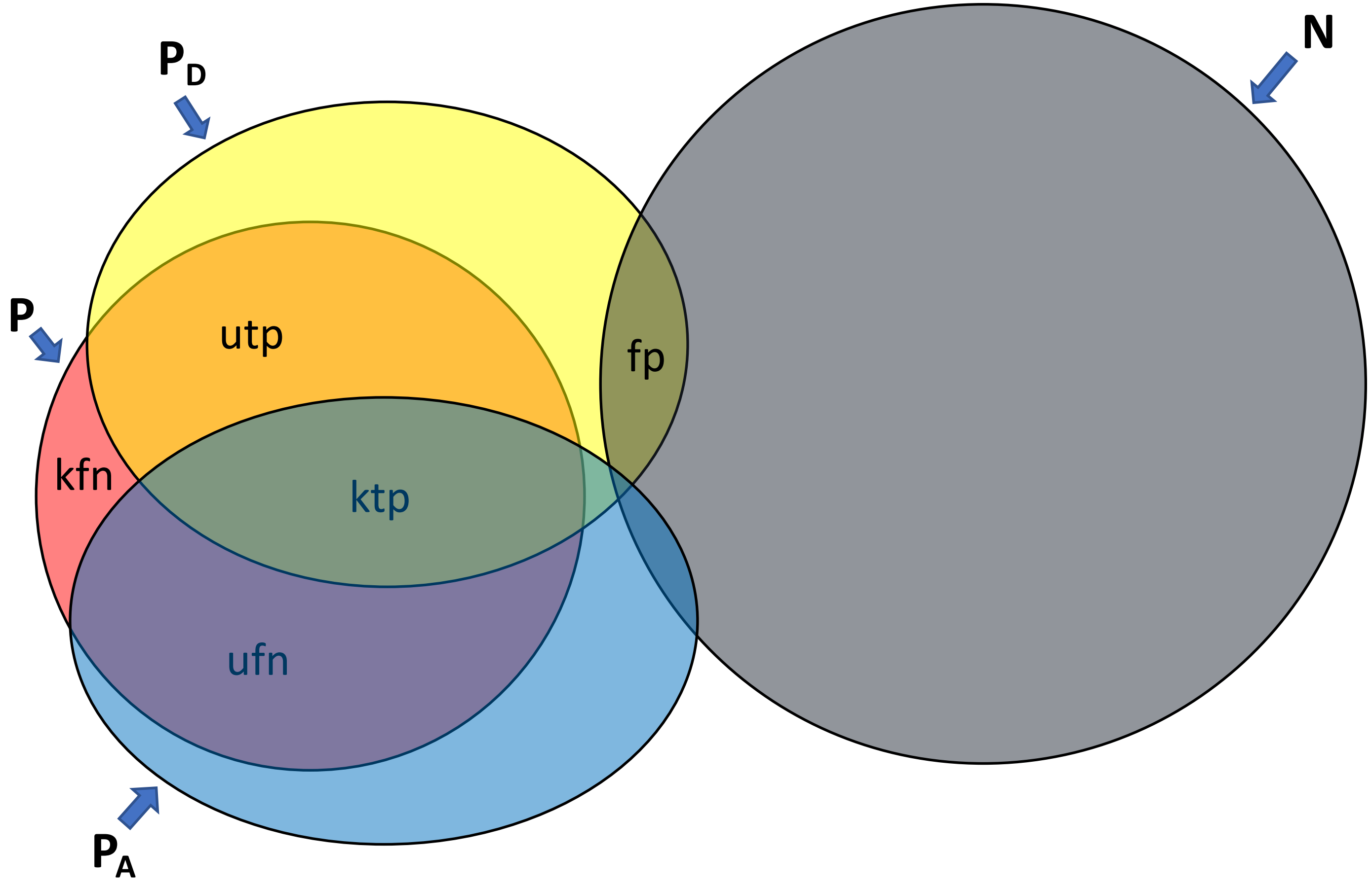
fp = N ∩ P_D = defender false positives
tp = P ∩ P_D = defender true positives
fn = P-tp = defender false negatives

Defender success = minimize fpr
maximise tpr

fpr = $\frac{|fp|}{|N|}$ $(false\ positive\ rate)$

tpr = $\frac{|tp|}{|P|} = (true\ positive\ rate)$

Standard Anomaly Detection Concepts

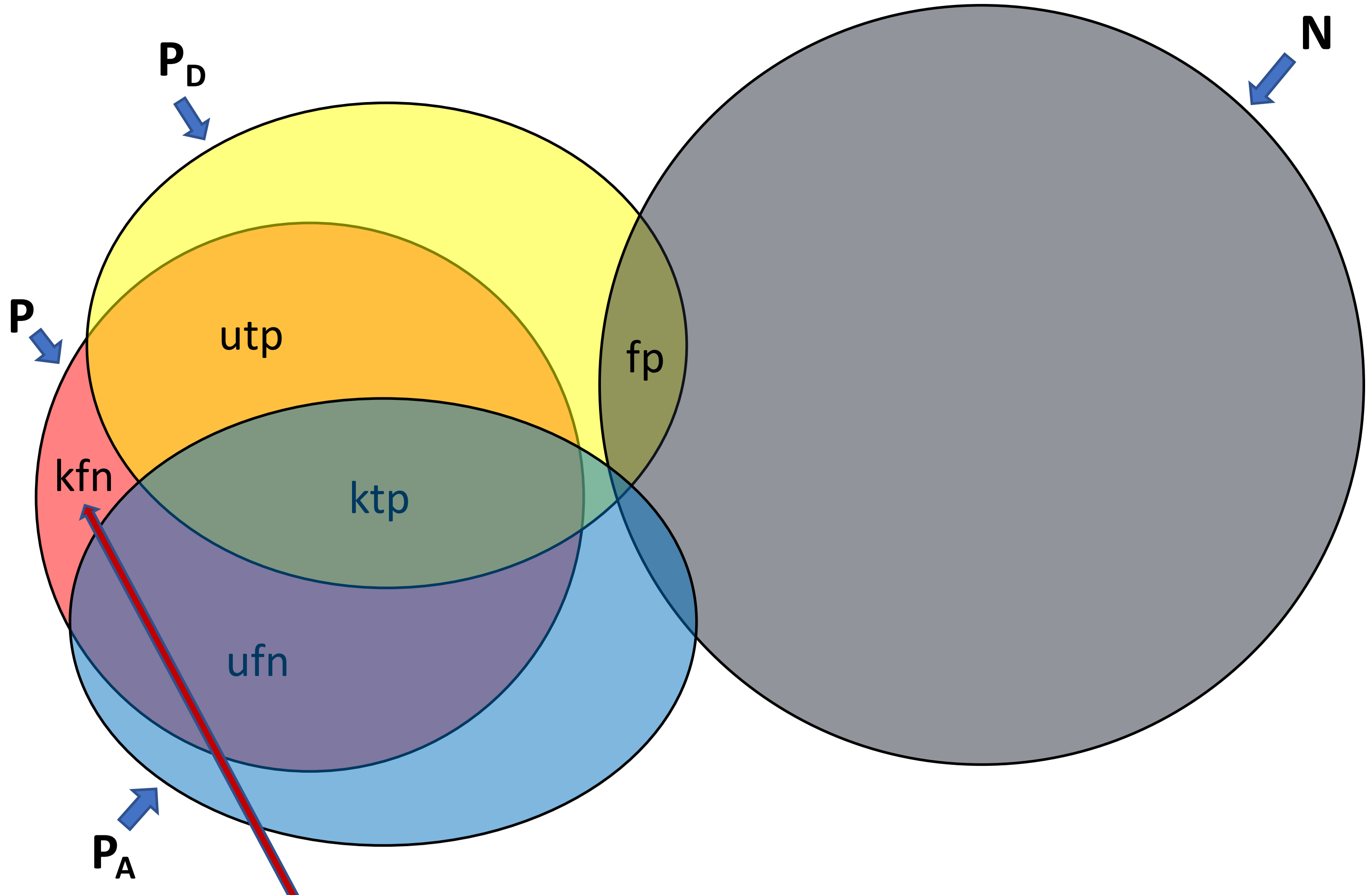


P = positives (anomalies)
N = negatives (normal data)
P_D = classified as positive by defender
P_A = classified as positive by adversary

$fp = N \cap P_D$ = defender false positives
$tp = P \cap P_D$ = defender true positives
$fn = P - tp$ = defender false negatives

kfn = known false negatives
ufn = unknown false negatives
ktp = known true positives
utp = unknown true positives

Extension to Adversarial Context



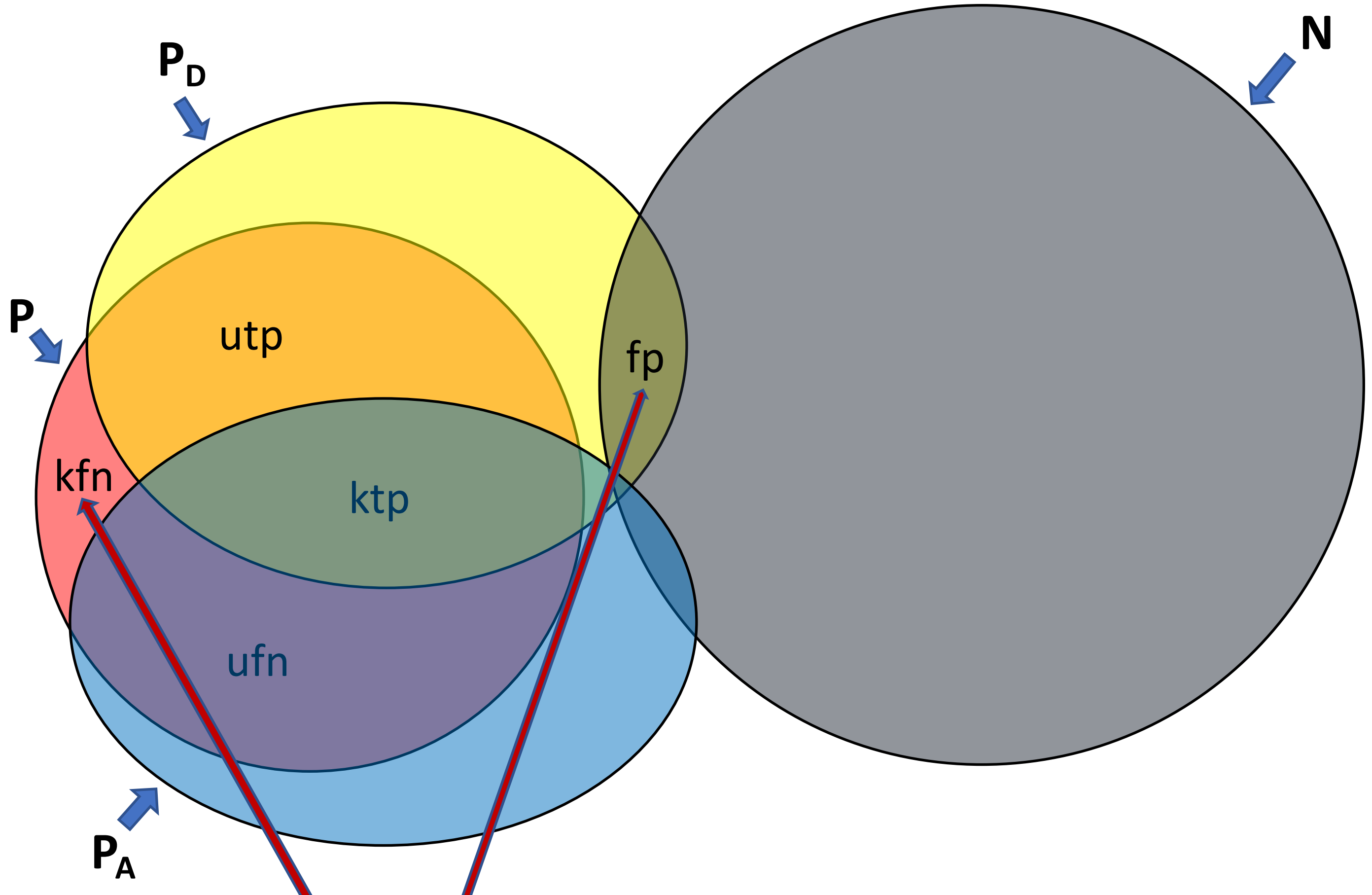
P = positives (anomalies)
N = negatives (normal data)
P_D = classified as positive by defender
P_A = classified as positive by adversary

fp = N ∩ P_D = defender false positives
tp = P ∩ P_D = defender true positives
fn = P-tp = defender false negatives

kfn = known false negatives
ufn = unknown false negatives
ktp = known true positives
utp = unknown true positives

Evasive adversarial hypothesis: P_D and P_A are similar
Evasive adversarial strategy: select an anomaly in kfn ∪ utp
Adversarial success: the selected anomaly belongs to kfn

kfn = known to be fn by adversary = critical area (defender wants to minimize this)



Performance measures for an evasive adversary, who selects e in kfn U utp

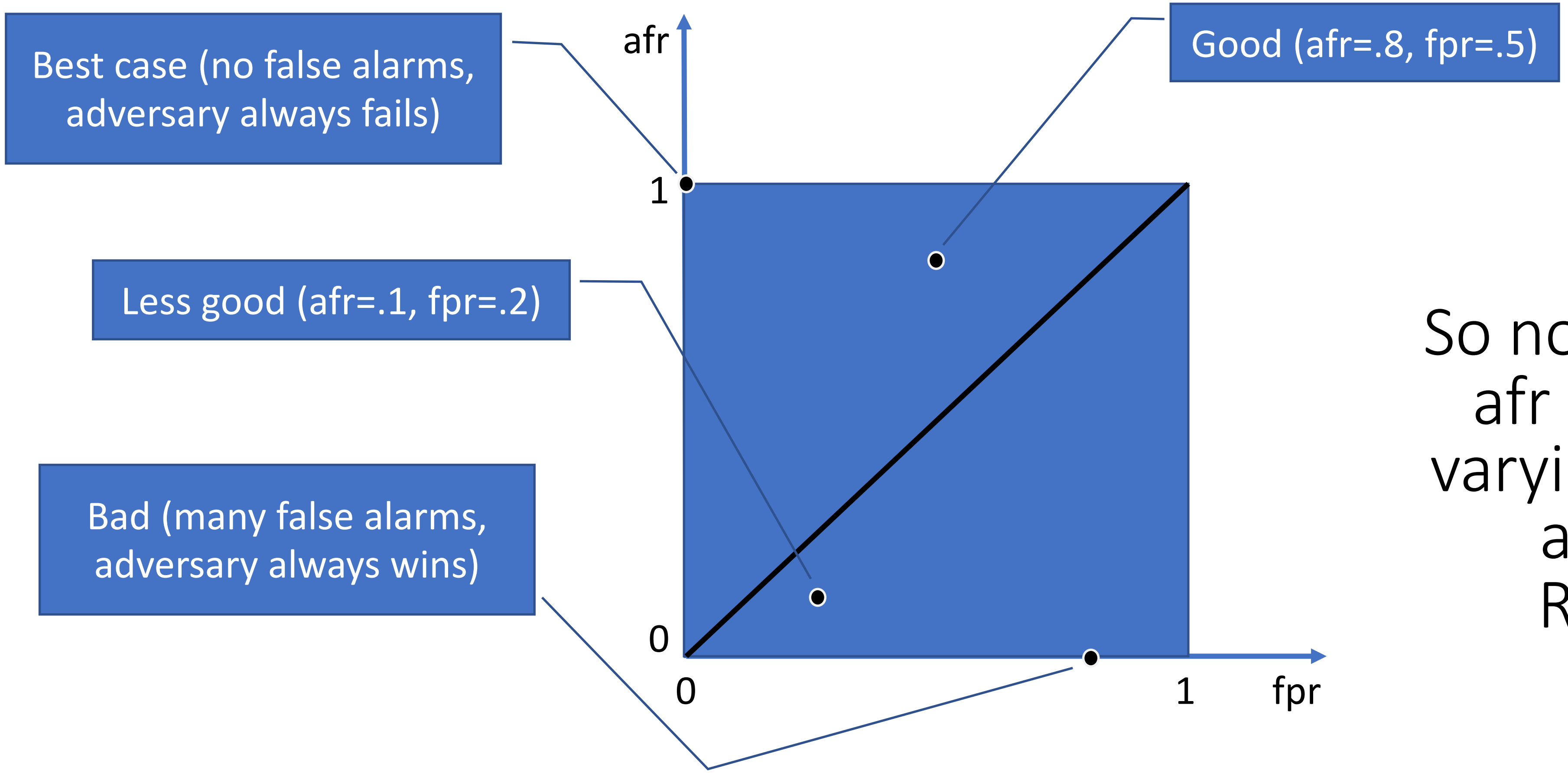
1) Minimize **fpr** = |fp|/|N|
2) Maximize the «adversarial failure rate» (afr), where

$$afr = \begin{cases} 1 & if \quad kfn \cup utp = \emptyset \\ \frac{|utp|}{|kfn|+|utp|} & otherwise \end{cases}$$

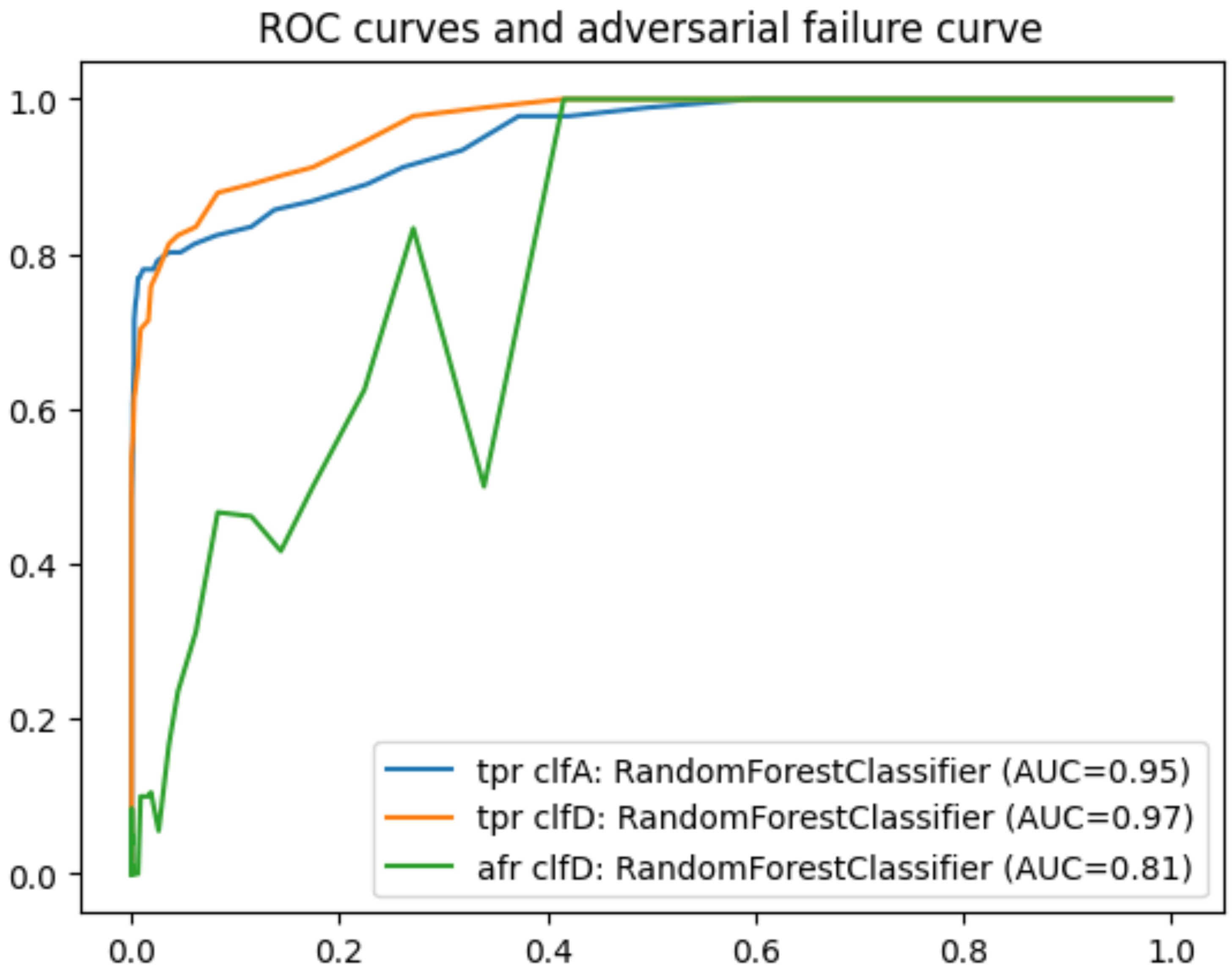
→ there is no e such that e∈P and e∉P_A

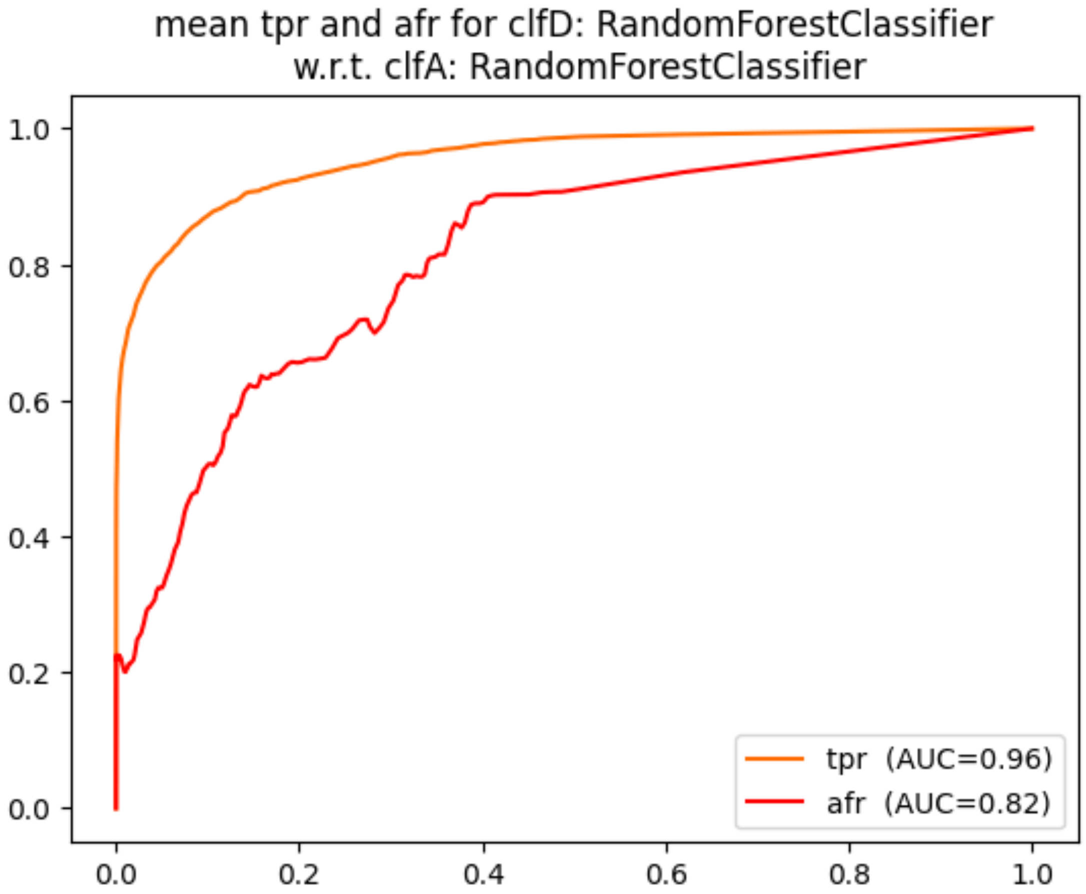
→ example e chosen by the adversary is in utp

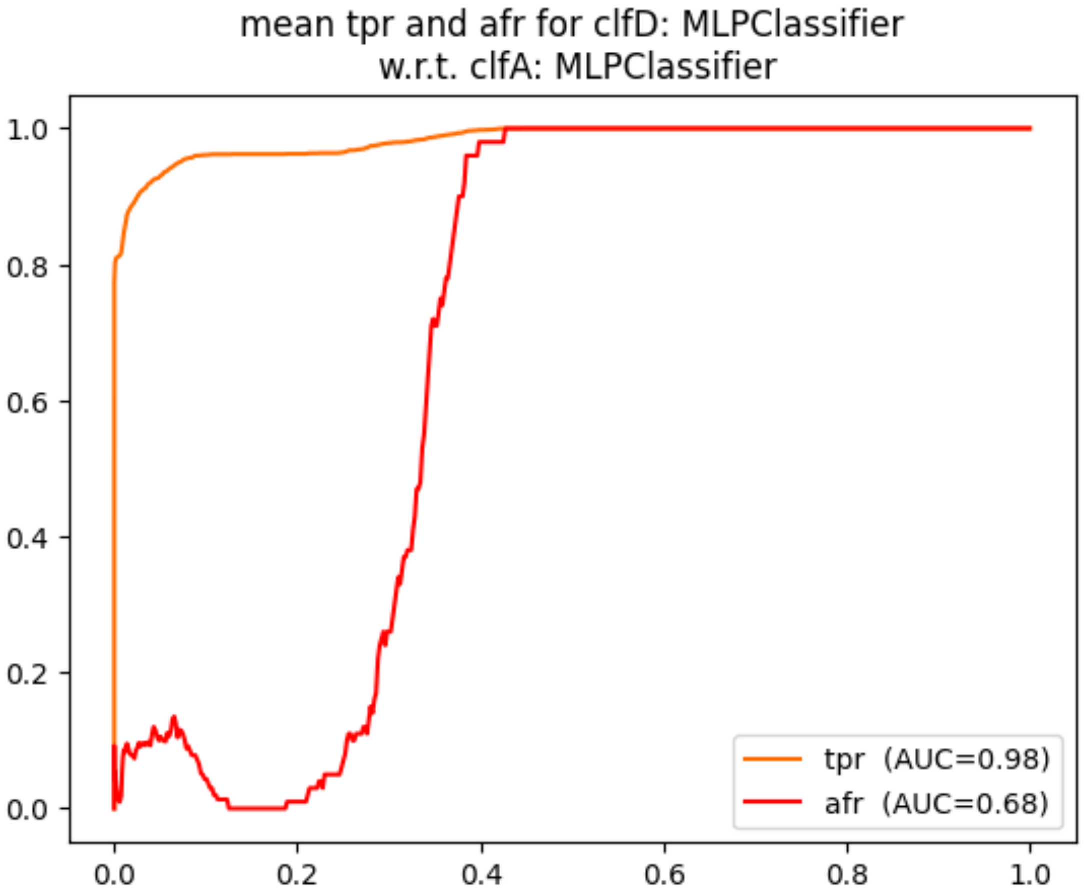
New notion: afr space



So now we can plot afr curves when varying thresholds, as done for ROC curves


ROC curves and adversarial failure curve
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
tpr clfA: RandomForestClassifier (AUC=0.95)
tpr clfD: RandomForestClassifier (AUC=0.97)
afr clfD: RandomForestClassifier (AUC=0.81)


mean tpr and afr for clfD: RandomForestClassifier
w.r.t. clfA: RandomForestClassifier
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
tpr (AUC=0.96)
afr (AUC=0.82)


mean tpr and afr for clfD: MLPClassifier
w.r.t. clfA: MLPClassifier
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
tpr (AUC=0.98)
afr (AUC=0.68)

Summary

- Evasive adversary threat model:
  - tries to replicate the defender's learning step
  - wants to evade detection by selecting test examples that are false negatives
- Definition of adversarial failure rate (afr)
  - adversarial failure = attack detected or evasion impossible
- Adversarial failure curves (based on afr)
  - afr_AUC as a good measure of evasion resistance

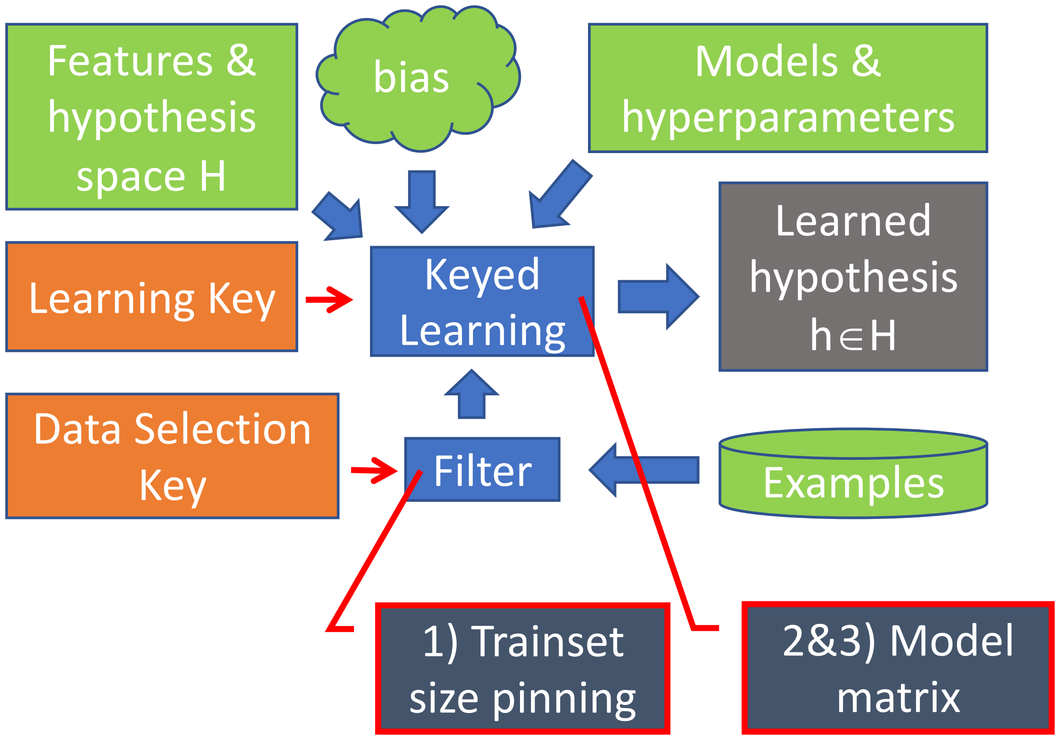
New Randomization Techniques

New randomization techniques, targeting evasion resistance as measured by AFR-AUC:

1) Training set size pinning

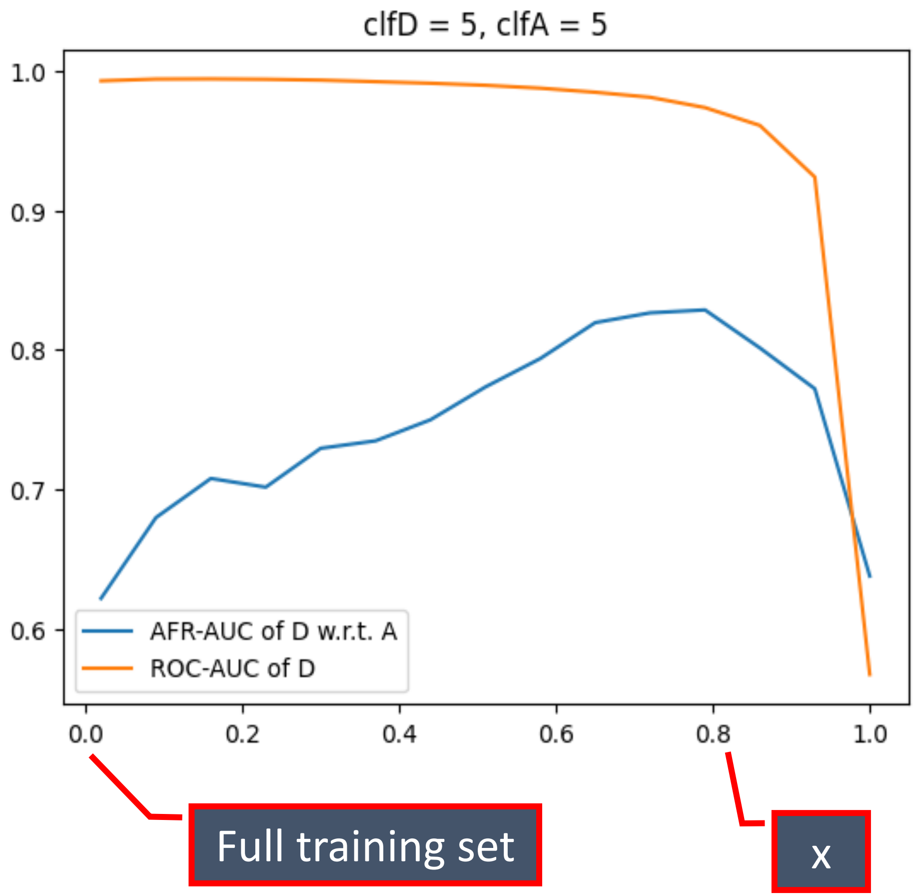
2) Model Matrix

How can we randomize the learning process?



1) Selection of the training examples
2) Selection of models & parameters
3) Selection of features & H

1) Randomize by selecting training examples: training set size pinning



Assumption: the adversary knows all training data T

Idea:

1) Use a secret and random training subset $T_i \subseteq T$
2) Pin the optimal size x of T_i by induction, i.e. the size of T_i that will maximise AFR-AUC on a validation set

2) Randomize with model matrix

afrAUC for different combinations of clfA and clfD (trainSize=1.0)*

clfD \ clfA	knn	random forest	adaboost
knn	0.46	0.91	0.99
random forest	0.72	0.80	0.92
adaboost	0.89	0.92	0.62

Defender randomly picks a row
Adversary randomly picks a comumn
(each combination has equal probability)

combined afr = average afr* = 0.8

***data obtained with the scikit-learn digits dataset**

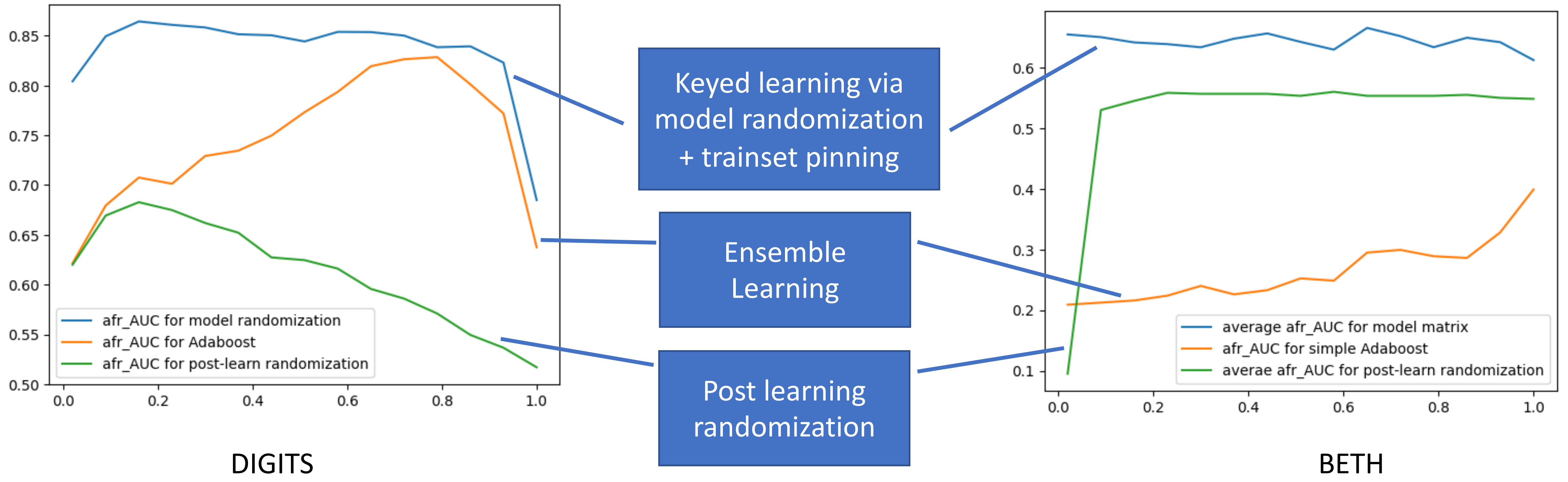
Application to IDS (with the Beth dataset)

Comparing well-known randomization methods to our combined techniques (training set size pinning and model matrix) w.r.t. AFR-AUC

The Beth[1] data set

- Context and data sources
    - more than 8 million total labeled data points, tracking 23 honeypots for 9 hours
    - working subset of 1,141,078 data points as suggested in [1]
- Features and classes
    - 14 numeric and discrete features, plus 2 binary class labels (‘sus’, ‘evil’)
- Preprocessing
    - we implemented a preprocessing phase as suggested in appendix A of [1]
    - after checking with the authors, we removed one additional feature (userId), that would have otherwise made the problem too easy

[1] Kate Highnam, Kai Arulkumaran, Zachary Hanif, and Nicholas R. Jennings. “Beth Dataset: real cybersecurity data for anomaly detection research”, Conf. Applied ML for Inf. Security (CAMLIS 2021).


0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.0
0.2
0.4
0.6
0.8
1.0
afr_AUC for model randomization
afr_AUC for Adaboost
afr_AUC for post-learn randomization
DIGITS
Keyed learning via model randomization + trainset pinning
Ensemble Learning
Post learning randomization
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.2
0.4
0.6
0.8
1.0
average afr_AUC for model matrix
afr_AUC for simple Adaboost
averae afr_AUC for post-learn randomization
BETH

Conclusions

- New performance measure for evasion avoidance:
  - afr (adversarial failure rate) curves and AFR-AUC
- New randomization schemes:
  - Training set randomization via trainset size pinning
  - Model matrix
- Experimental comparison using two different data sets (digits, Beth) + work in progress with Kyoto IDS:
  - Combination of model matrix & training set size pinning
  - Post-learning randomization
  - Randomization intrinsic in ensemble learning

Consistently superior

References

[1] Kate Highnam, Kai Arulkumaran, Zachary Hanif, and Nicholas R. Jennings. “Beth Dataset: real cybersecurity data for anomaly detection research”, Conf. Applied ML for Inf. Security (CAMLIS 2021).

[2] F. Bergadano. “Keyed learning: An adversarial learning framework”, ETRI Journal 41 (5), 608-618, 2019

[3] T. Fawcett. “An introduction to ROC analysis”, Pattern Recognition Letters 27, pp. 861-874, 2006

[4] S. Rota Bulò et al., “Randomized prediction games for adversarial machine, learning”, IEEE Trans. Neural Netw. Learn. Syst. 28 (2017), no. 11, 2466–2478

[5] O. Taran, S. Rezaeifar, T. Holotyak, S. Voloshynovskiy. “Machine learning through cryptographic glasses: combating adversarial attacks by key-based diversified aggregation”, EURASIP J. Inf. Secur. 2020

[6] B. Biggio, F. Roli, “Wild patterns: Ten years after the rise of adversarial machine learning”, in Proc. ACM SIGSAC Conference on Computer and Communications Security, CCS ’18, New York, NY, USA, 2018

[7] F. Yang et al., “Using Randomness to Improve Robustness of Tree-based Models against Evasion Attacks”, IEEE Trans. KDE, 34(2), pages 969-982, 2022

[8] R. S. Mrdovic and B. Drazenovic, “ KIDS: a Keyed Intrusion Detection System”, Proc. DIMVA 2010.