Leibniz Supercomputing Centre
of the Bavarian Academy of Sciences and Humanities

AI for Anomaly Detection

Prerequisites

Professional Data Science Background with Python

Basics of Deep Learning – Have trained a DNN

Agenda

- Introduction to Anomaly Detection
- Supervised Learning with XGBoost
- Break
- Unsupervised Learning with Autoencoders
- Unsupervised Learning with GANs
- Assessment: Apply one technique to a new dataset

Introduction to Anomaly Detection

WHAT IS AN ANOMALY?

A ***data point*** which differs significantly from other ***data points***

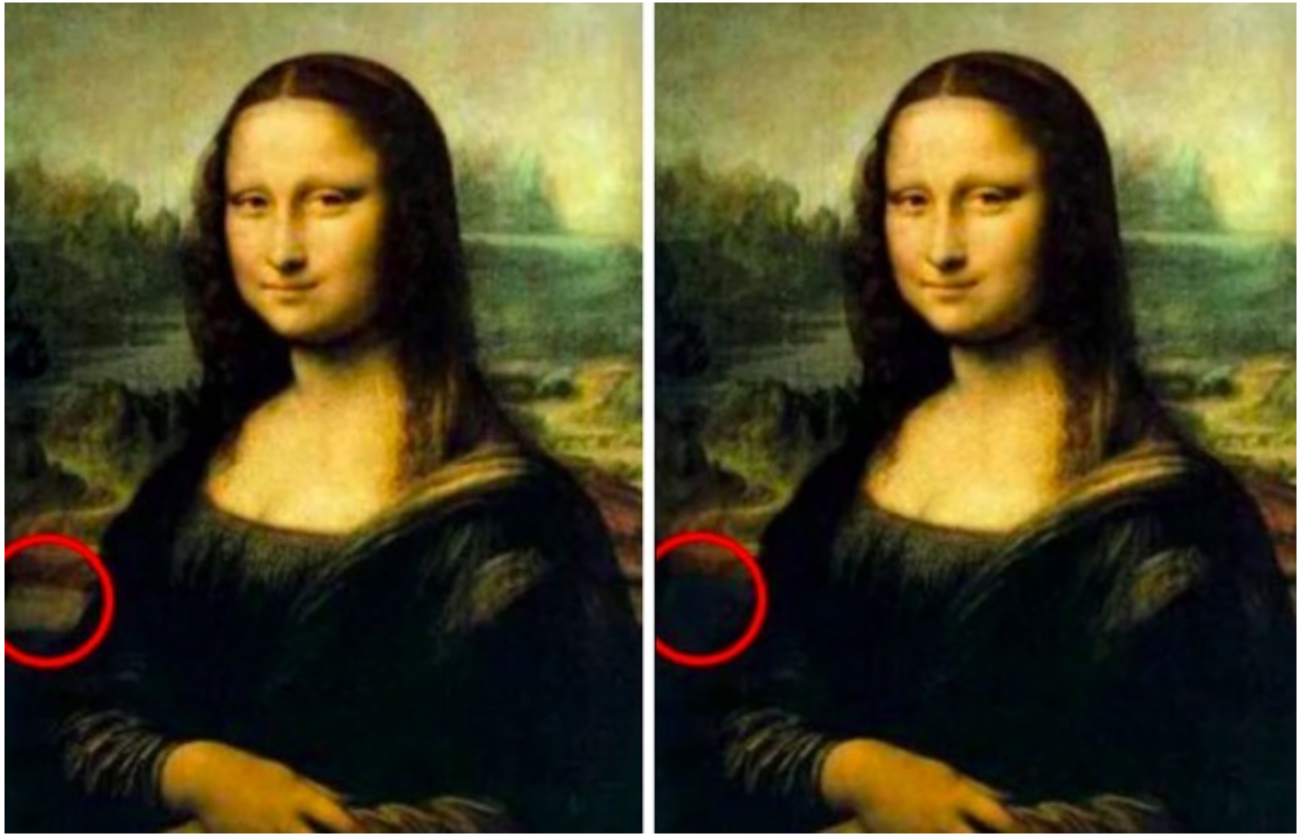
- An observation that is likely generated by a different mechanism

- Finding anomalies can be useful in telecom/sp networks, cyber security, finance, industry, IOT, healthcare, autonomous driving, video surveillance, robotics.

- Many other problems can be framed as anomaly detection: customer retention, targeted advertising.

SPOT THE ANOMALY

SPOT THE ANOMALY



EXERCISE

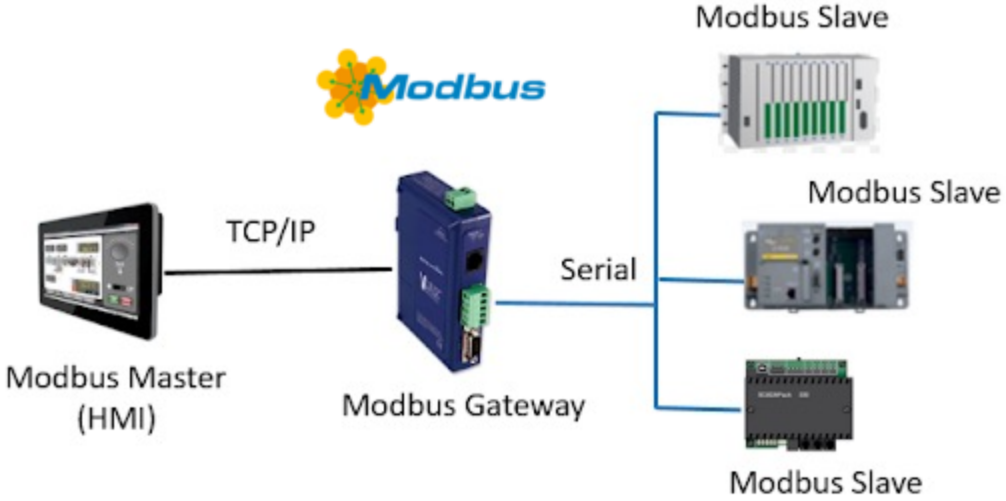
- What are some of the scenarios that produce anomalies in your organization/domain?
- What data sources might affect or record those anomalous activities?
- What kind of data analytics techniques could be applied or have been applied to detect those events?

Why is Anomaly Detection Important?

Case Study

Programmable Logic Controllers (PLCs)

Supervisory control and data acquisition (**SCADA**)



The Stuxnet Worm

Case Study

- A 500-kilobyte malicious computer worm that targets SCADA systems.
- Spread:
  - Through infected removable drives such as USB flash drives.
- Operation:
  - Analyzed and targeted Windows networks and computer systems.
  - Compromised the Step7 software, the worm gained access to 45 S7 to the PLCs.
  - Virus modified project communication configurations for the PLC’s Ethernet ports
- Result:
  - Infected over 100,000 computers & 22 Manufacturing sites
  - Appears to have impacted Natanz nuclear facility destroying 984 uranium enriching centrifuges.

DATASET

At a glance!

Name	KDD99 Intrusion Detection Dataset Publicly available at http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
Size	743 Mb
No. of Features	Numeric = 22 ; Categorical = 9
No. of Rows	18 Million
No. of Classes	23 (Including the Normal category)
Variable Types	Numeric & Categorical
Goal	**Detect Anomalies by studying Network Packet logs**

DATASET

Basic Features	Content Features	Traffic Features
duration	hot	count
protocol_type	num_failed_logins	serror_rate
service	logged_in	rerror_rate
src_bytes	num_compromised	same_srv_rate
dst_bytes	root_shell	diff_srv_rate
flag	su_attempted	srv_count
land	num_root	srv_serror_rate
wrong_fragment	num_file_creations	srv_rerror_rate
urgent	num_shells	srv_diff_host_rate
	num_access_files	
	num_outbound_cm	
	is_hot_login	
	is_guest_login	

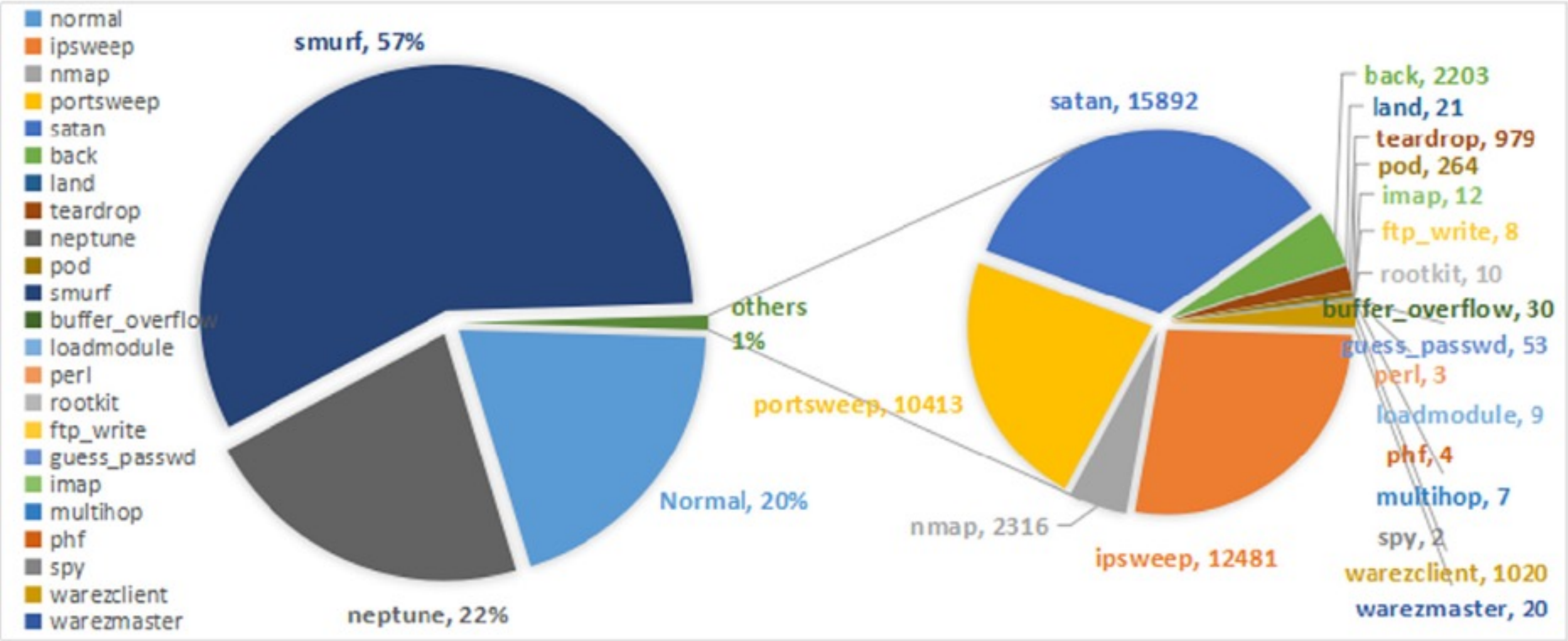
■ Numerical

■ Categorical

Detailed Description @
https://kdd.ics.uci.edu/databases/kddcup99/task.html

DATASET

Visualization by class



Handling Time Series Data

For Classification

Time	Feature 1	Feature 2	Feature 3
00:00:00	*Val_1*	*Val_2*	*Val_3*
00:00:01	*Val_4*	*Val_5*	*Val_6*
00:00:02	*Val_7*	*Val_8*	*Val_9*
00:00:03	*Val_10*	*Val_11*	*Val_12*

Averaging Features

Duration	Feature 1	Feature 2	Feature 3
1	*Avg(Val_1,Val_4)*	*Avg(Val_2,Val_5)*	*Avg(Val_3,Val_6)*
1	*Avg(Val_7,Val_10)*	*Avg(Val_8,Val_11)*	*Avg(Val_9,Val_12)*

Sampling Features

Duration	Feature 1	Feature 2	Feature 3
1	*Val_4*	*Val_5*	*Val_6*
1	*Val_10*	*Val_11*	*Val_12*

IN THE NEWS

Telecom

Operators beware: DDoS attacks—large and small—keep increasing

by Brian Santo | Jun 6, 2017 12:19pm

Telecoms industry and DNS attacks: attacked the most, slowest to fix

Networks are a prized target for hackers, as each attack costs £460,000 on average to remediate

https://www.information-age.com/telecoms-industry-dns-attacks-attacked-slowest-fix-123469037/

Telecom operators are not properly prepared for cyber-attacks: A10 Networks

Mobile network operators are not properly prepared for cyber attacks, and the core of 3G and 4G networks is generally not protected.

ETTelecom | Updated: January 15, 2018, 13:41 IST

https://telecom.economictimes.indiatimes.com/news/telecom-operators-are-not-properly-prepared-for-cyber-attacks-a10-networks/62504221

Hackers Are Tapping Into Mobile Networks' Backbone, New Research Shows

Parmy Olson Forbes Staff

AI, robotics and the digital transformation of European business.

https://www.forbes.com/sites/parmyolson/2015/10/14/hackers-mobile-network-backbone-ss7/#59d777f85142

Hack Attack: Sony Confirms PlayStation Network Outage Caused By 'External Intrusion'

Rip Empson @ripemp / 8 years ago

Comment

https://techcrunch.com/2011/04/23/hack-attack-sony-confirms-playstation-network-outage-caused-by-external-intrusion/

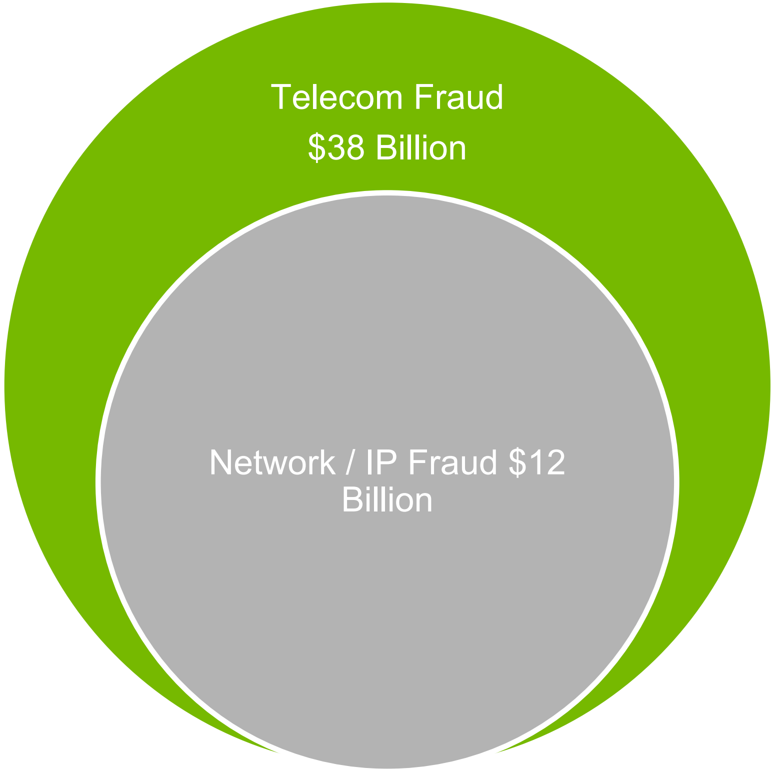
ANDY GREENBERG SECURITY 04.16.18 07:52 PM

THE WHITE HOUSE WARNS ON RUSSIAN ROUTER HACKING, BUT MUDDLES THE MESSAGE

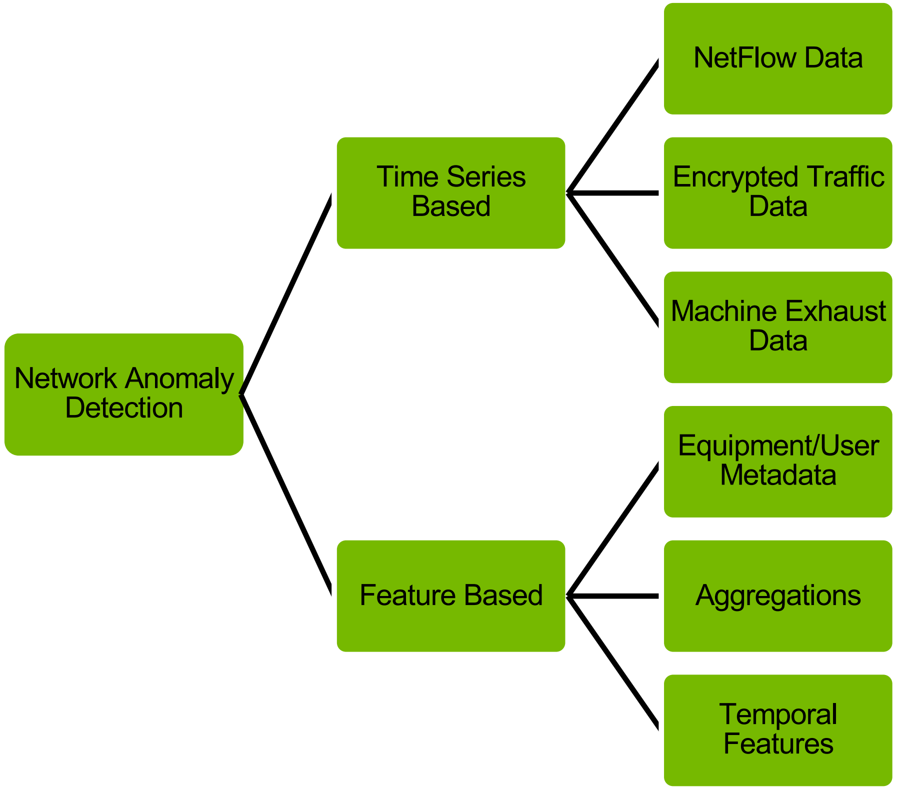
https://www.wired.com/story/white-house-warns-russian-router-hacking-muddles-message/

ANOMALY DETECTION IN NETWORKS

Why do we need it in Telecom ?



What sort of data can we leverage?



DETECTION METHODS IN THIS COURSE

Anomaly Detection		
Supervised (When you have Labels)	Unsupervised (When you don't have labels for your data)	
XGBoost	Autoencoders	Generative Adversarial Networks
RAPIDS		

GPU ACCELERATED XGBOOST

XGBOOST

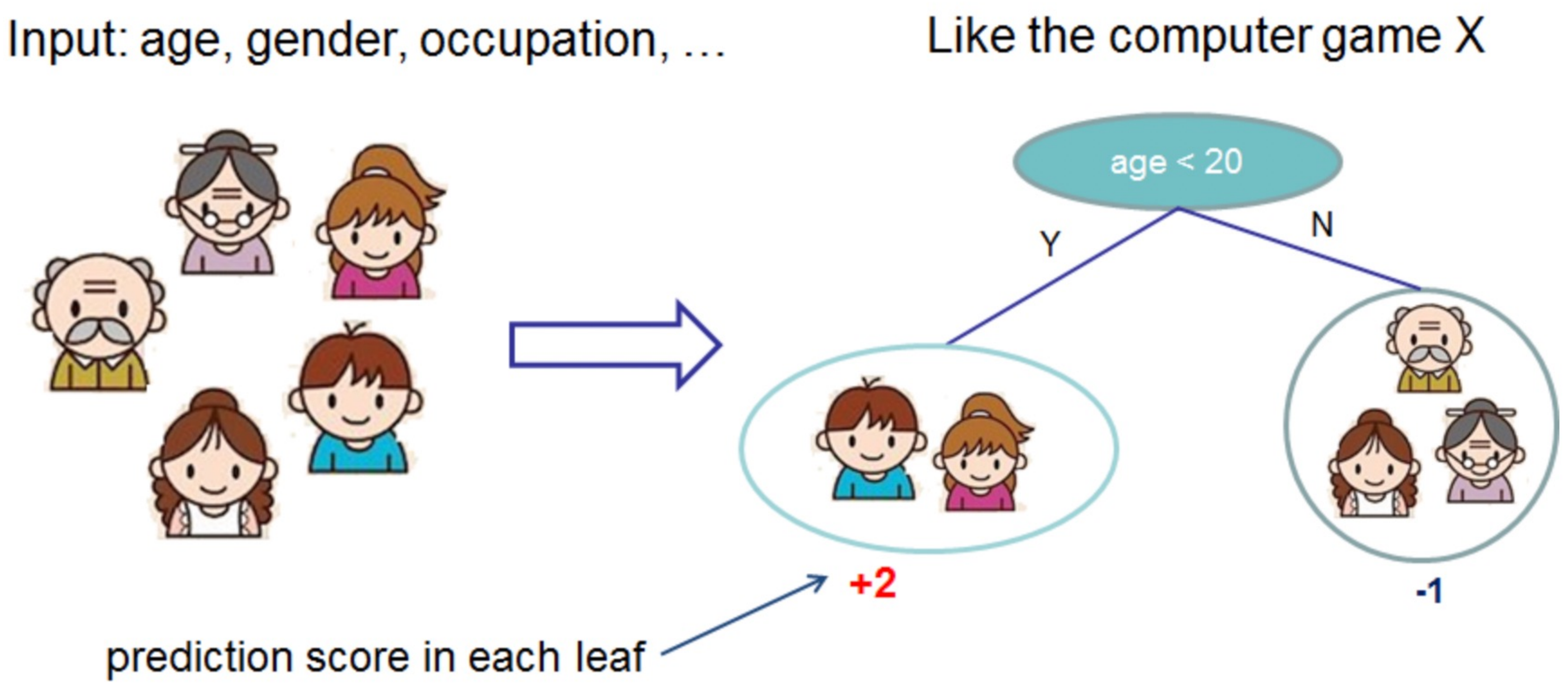
Definition

XGBoost is an implementation of gradient boosted decision trees designed for speed and performance.

It is a powerful tool for solving classification and regression problems in a supervised learning setting.

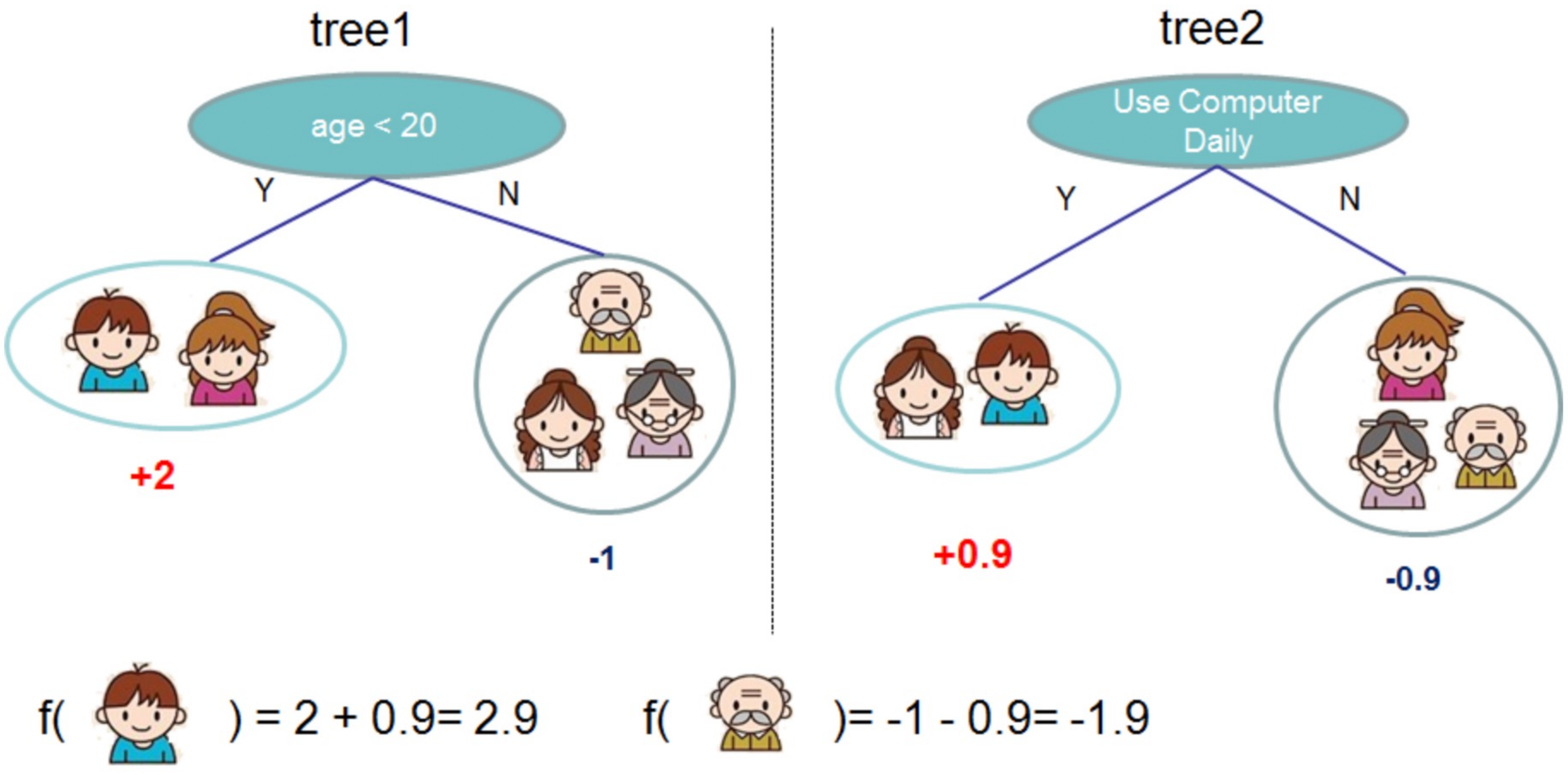
PREDICT: WHO LIKES COMPUTER GAME X

Example of Decision Tree



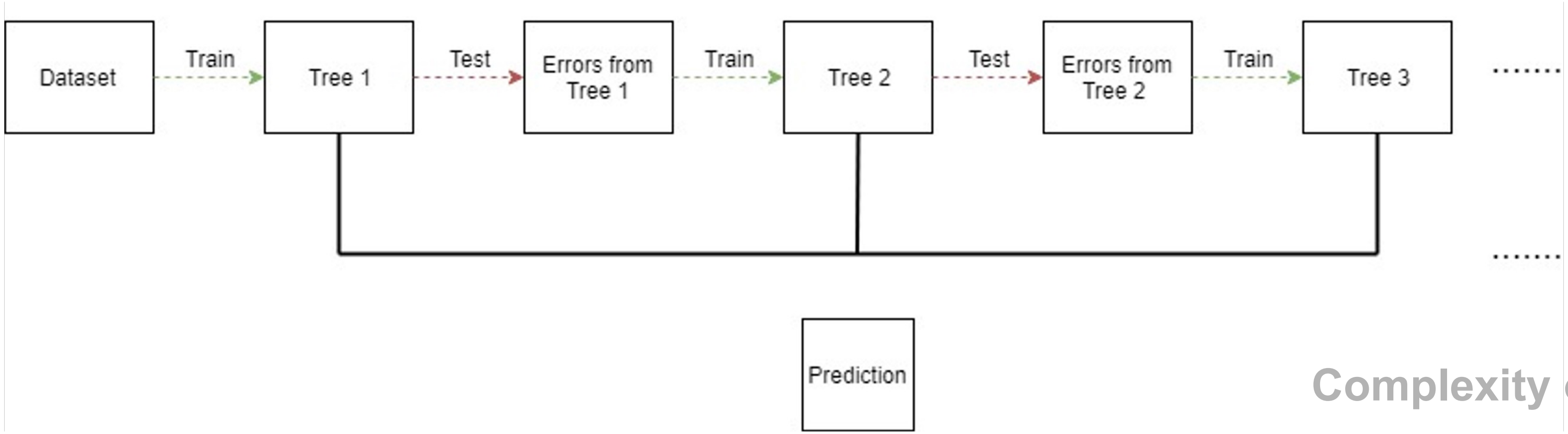
Source: https://goo.gl/eTxVtA

ENSEMBLED DECISION TREES



Source: https://goo.gl/eTxVtA

GRADIENT BOOSTED TREES FOR STRONGER PREDICTIONS



Build trees one at a time, where each new tree helps to correct errors made by previously trained tree.

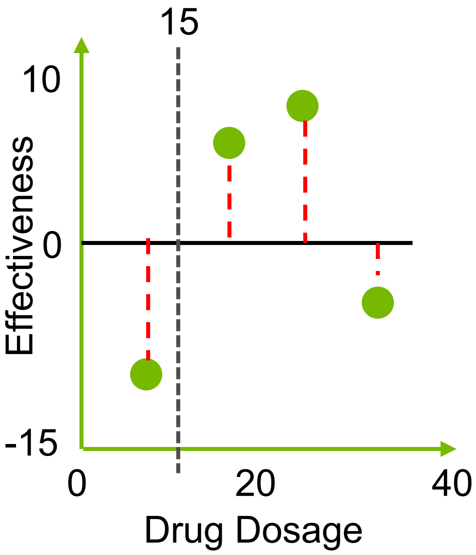
Complexity of the Trees

$$Obj = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

Training Loss

XgBoost

Intuitive Example for Tree Construction



Step 1: Start as a single leaf
Input all residuals

-10.5, 6.5, 7.5, -7.5

Step 2: Calculate similarity score
For all residuals

$$\frac{\text{Sum of residuals squared}}{\text{No. of residuals} + \text{Regularization}}$$

Set Threshold @ Arbitrary Drug Dosage 15

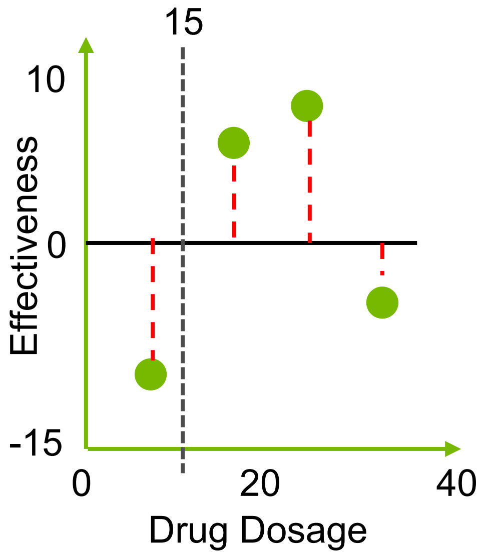
-10.5, 6.5, 7.5, -7.5

S0 = 4
Setting Reg = 0

https://www.youtube.com/watch?v=OtD8wVaFm6E

XgBoost

Intuitive Example for Tree Construction



Step 3: Calculate Similarity Score
For Left and Right Sub Trees

-10.5, 6.5, 7.5, -7.5

S10 = 110.25
Setting Reg = 0

-10.5

6.5, 7.5, -7.5

S11 = 14.08
Setting Reg = 0

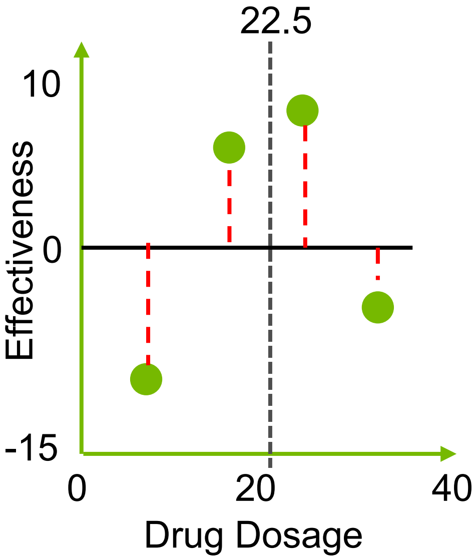
Step 4: Calculate Gain
Lower Leaves cluster similar residuals than root ?

Gain = Left Similarity + Right Similarity – Root Similarity

G1 = 120.33

XgBoost

Intuitive Example for Tree Construction



Step 5: Calculate Similarity Score
For Left and Right Sub Trees

Dosage < 22.5

S10 = 8
Setting Reg = 0

-10.5 , 6.5

7.5, -7.5

S11 = 0
Setting Reg = 0

Step 6: Calculate Gain
Lower Leaves cluster similar residuals than root ?

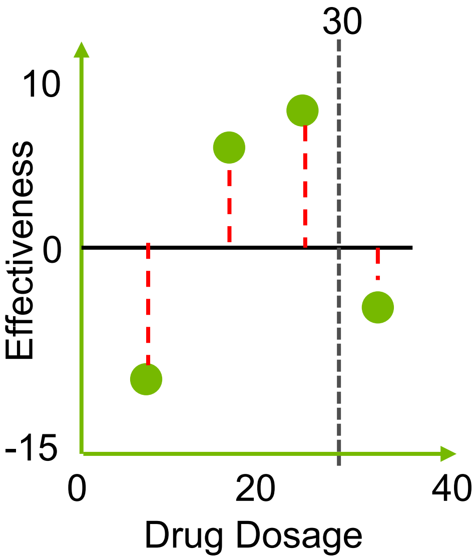
Gain = Left Similarity + Right Similarity – Root Similarity

G2 = 4

Since G2 = 4 < G1 = 120.33
Tree 1 had better split

XgBoost

Intuitive Example for Tree Construction



Step 7: Calculate Similarity Score
For Left and Right Sub Trees

Dosage < 30

S10 = 4.08
Setting Reg = 0

-10.5 , 6.5, 7.5

-7.5

S11 = 56.25
Setting Reg = 0

Step 6: Calculate Gain
Lower Leaves cluster similar residuals than root ?

Gain = Left Similarity + Right Similarity – Root Similarity

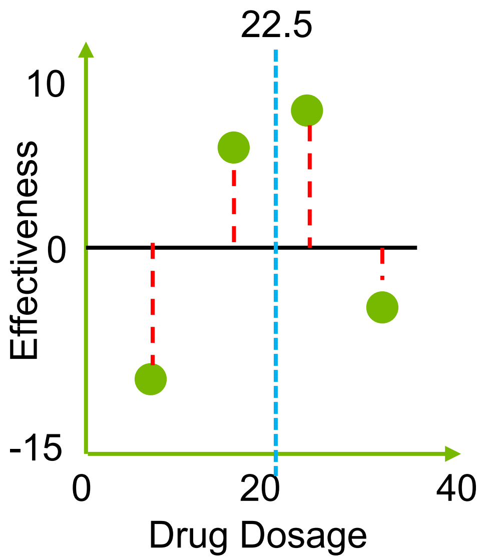
G3 = 56.33

Since G3 = 56.33 < G1 = 120.33

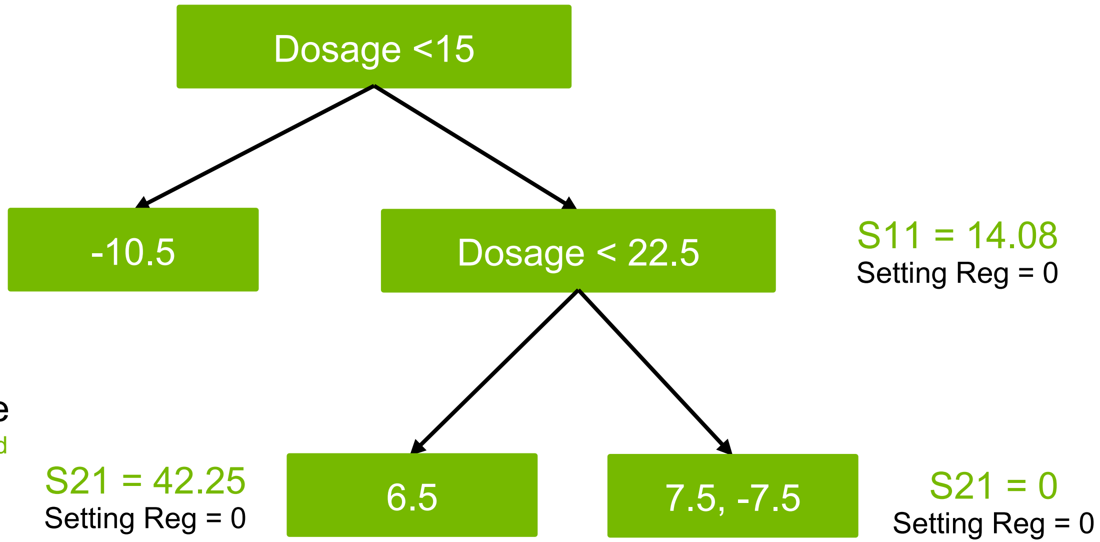
Tree 1 had better split

XgBoost

Intuitive Example for Tree Construction



S10 = 110.25
Setting Reg = 0



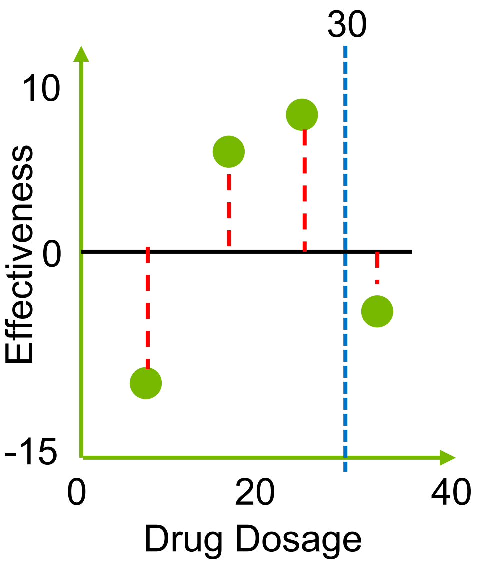
Step 8: Calculate Similarity Score
For Left and Right Sub Trees (2nd Level)

Step 9: Calculate Gain
Lower Leaves cluster similar residuals than root ?

G12 = 42.25 – 14.0 = 28.17

XgBoost

Intuitive Example for Tree Construction



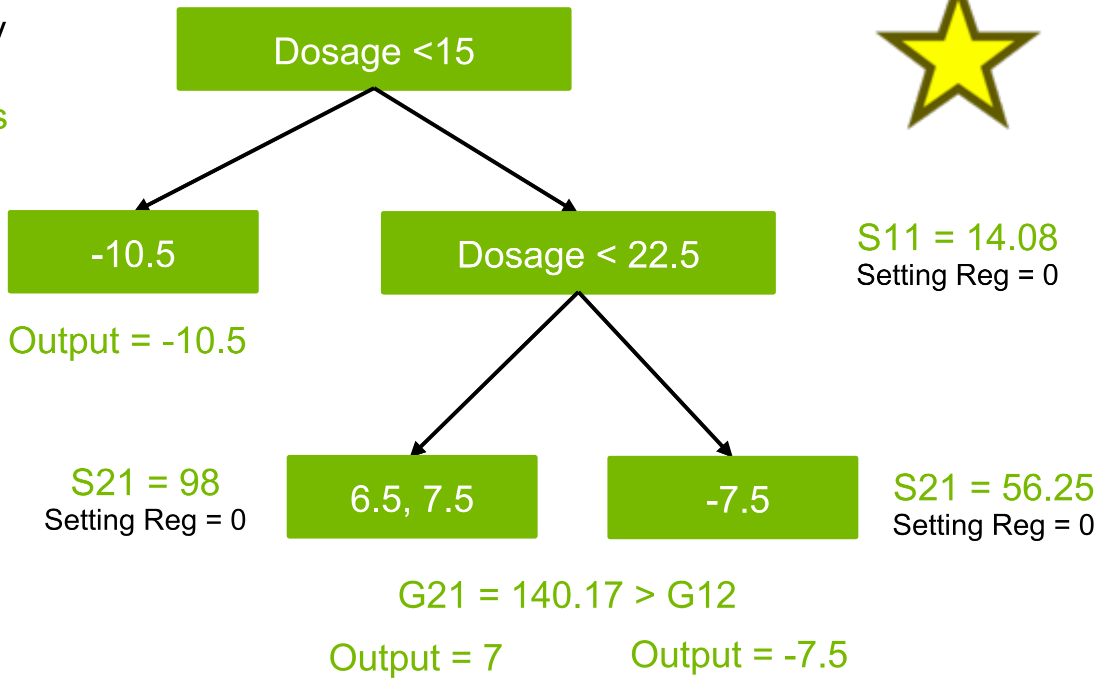
Step 10: Calculate Similarity Score

For Left and Right Sub Trees

S10 = 110.25
Setting Reg = 0

Step 11: Calculate Gain

Lower Leaves cluster similar residuals than root ?



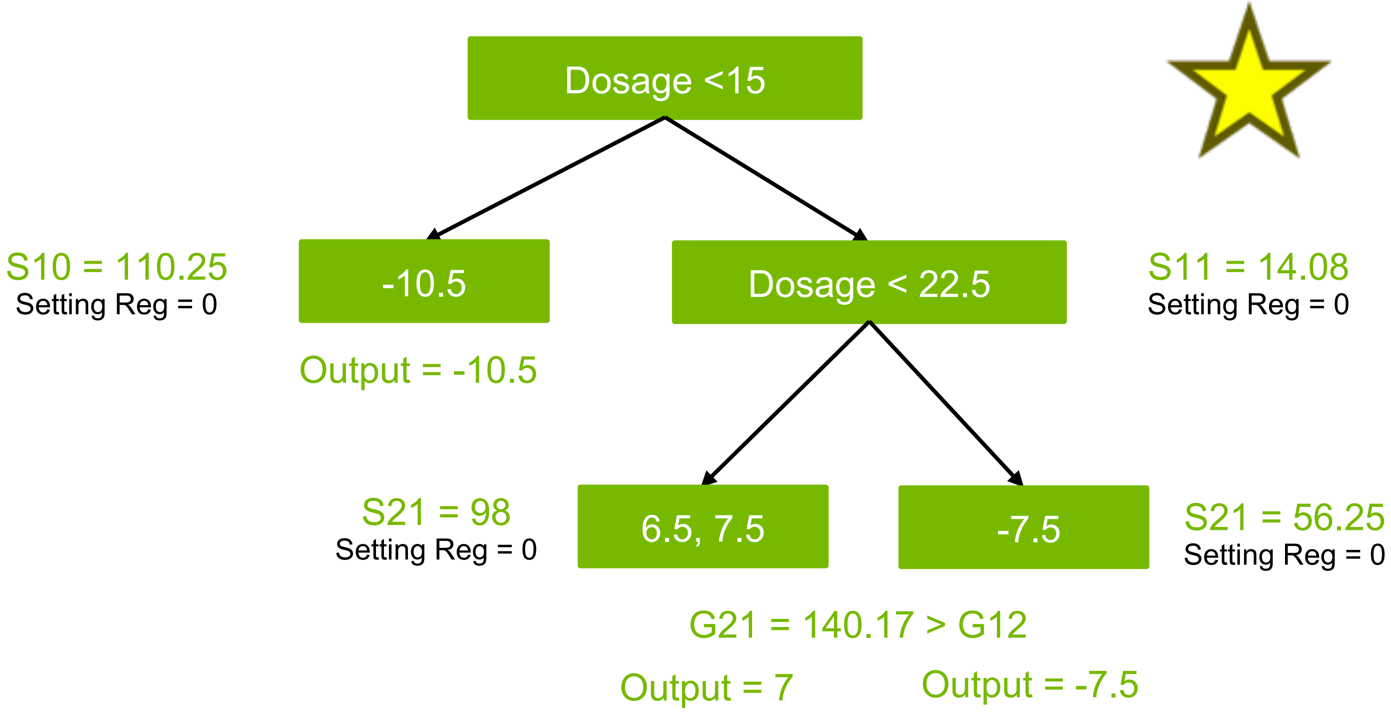
Step 12:
Calculate Output Value $= \dfrac{\text{Sum of residuals}}{\text{No. of residuals} + \text{Regularization}}$

XgBoost

Intuitive Example for Tree Construction

Re - Calculate Residuals :
Assuming Gradient Multiplier = 0.3

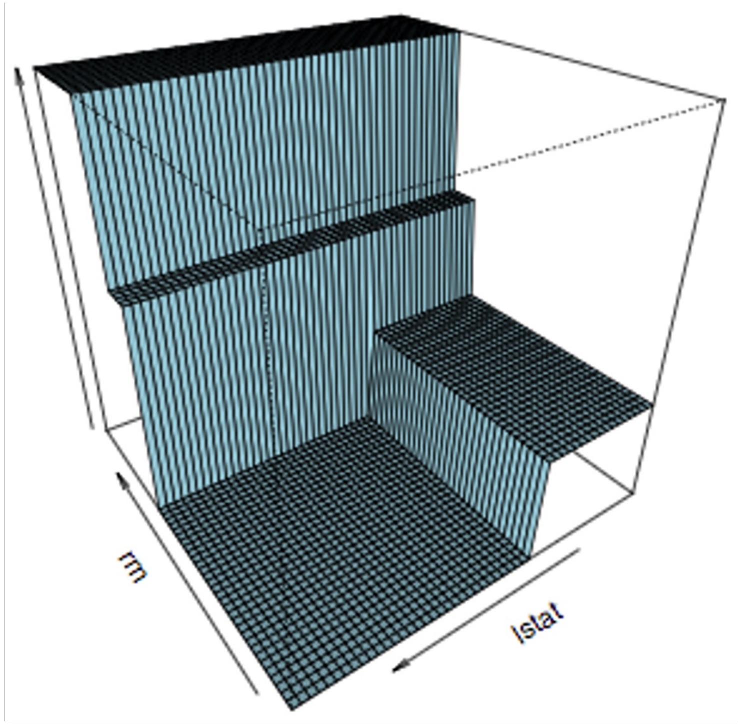
- R1 : 0.5 + 0.3 (-10.5) = - 2.65
- R2 : 0.5 + 0.3(7) = 2.6
- R3 : 0.5 + 0.3(7) = 2.6
- R4 : 0.5 + 0.3(-7.5) = -1.75

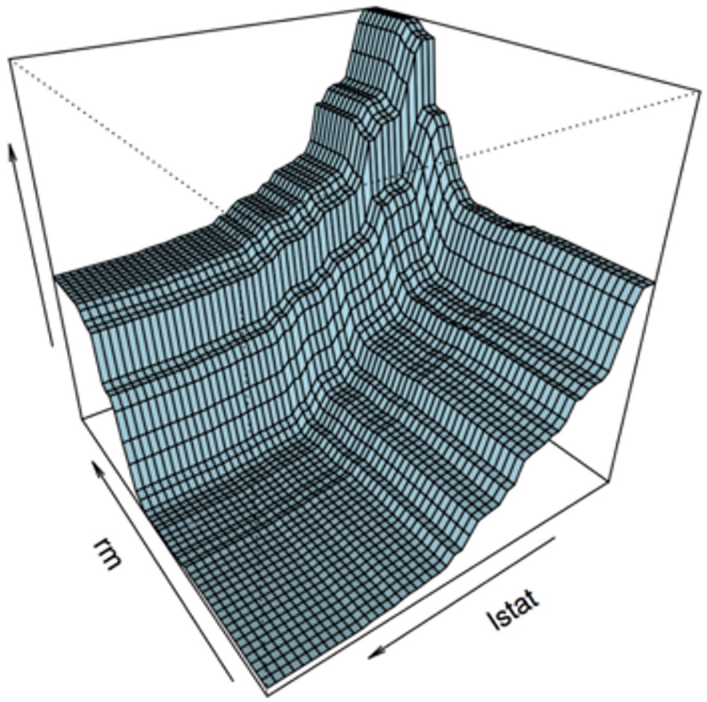


Step 13:
Construct new tree with updated Residuals

TRAINED MODELS VISUALIZATION

Single Decision Tree vs Ensembled Decision Trees





Models fit to the *Boston Housing* Dataset

Source: https://goo.gl/GWNdEm

XgBoost

Building up from a Decision Tree

Optimized version of GBT incorporating parallelism, tree pruning and regularization.

Utilize Gradient Descent to minimize errors in the sequentially built trees.

Trees built sequentially minimizing errors from previous trees and weighing better performing ones more.

Utilize random subsets of a dataset to build multiple decision trees

Ensemble of multiple decision trees to arrive at decision through majority voting

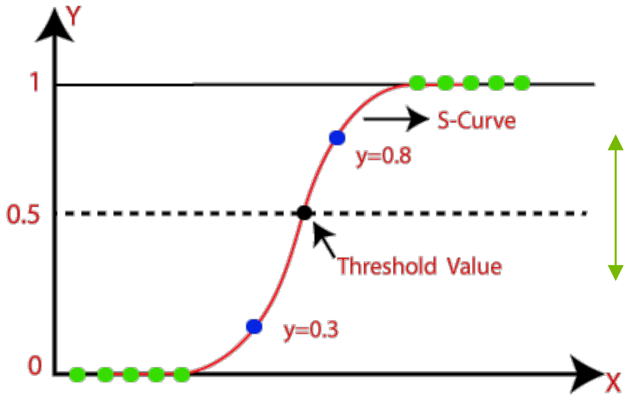
Tree based algorithm that outputs decisions based on certain conditions.

WHY XGBOOST?

ROC CURVE

Construction

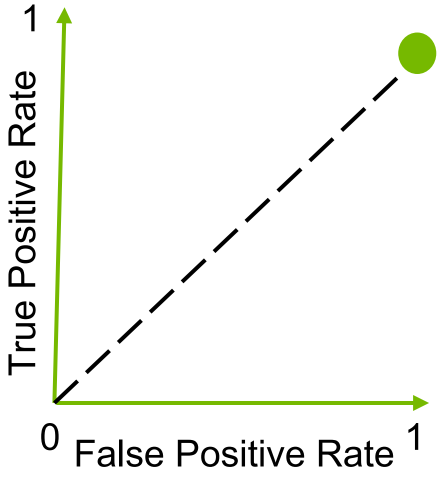
Move
Threshold



		Actual	
		Anomaly	Not Anomaly
Predicted	Anomaly	TP	FP
	Not Anomaly	FN	TN

True Positive Rate
TP / (TP+FN) = Sensitivity

False Positive Rate
FP / (FP +TN)

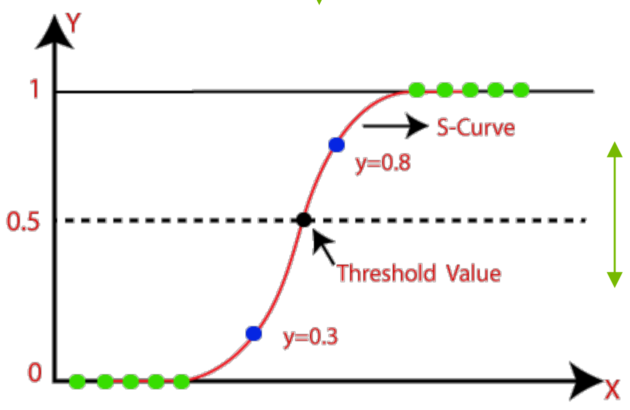


https://www.youtube.com/watch?v=4jRBRDbJemM

ROC CURVE

Construction

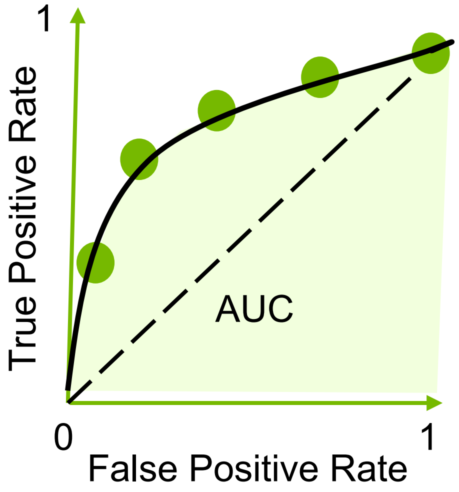
Move
Threshold



		Actual	
		Anomaly	Not Anomaly
Predicted	Anomaly	TP	FP
	Not Anomaly	FN	TN

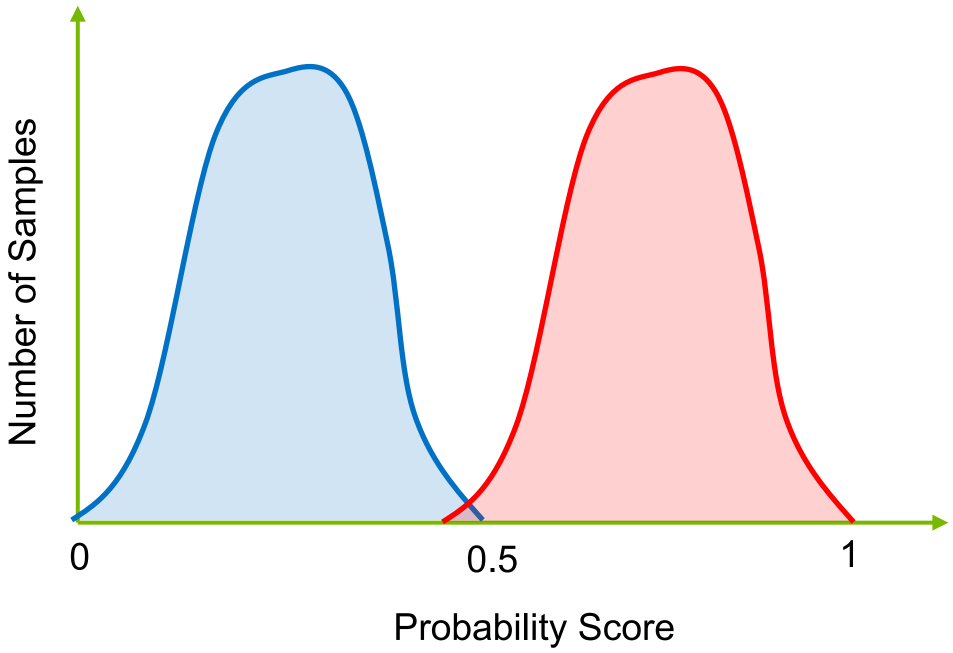
True Positive Rate
TP / (TP+FN) = Sensitivity

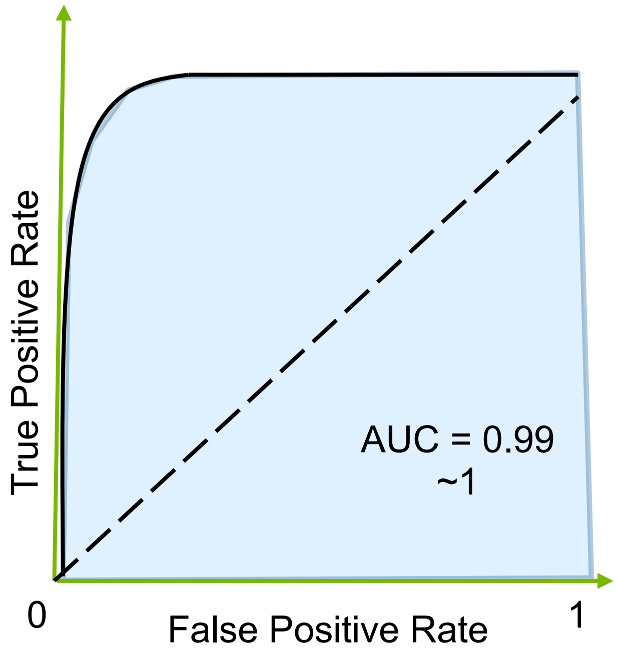
False Positive Rate
FP / (FP +TN)



ROC CURVE

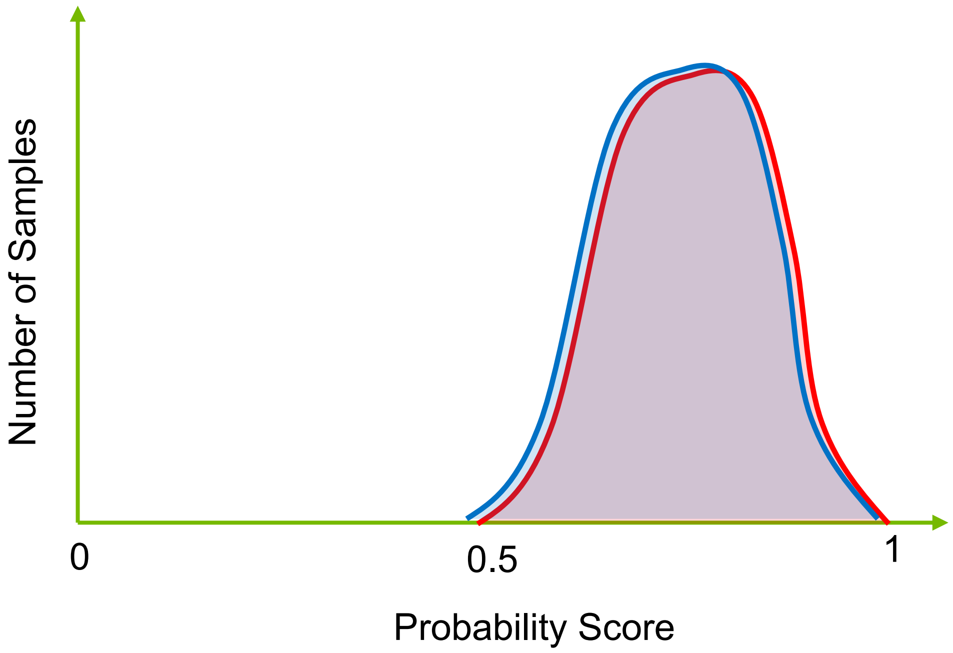
Interpretation

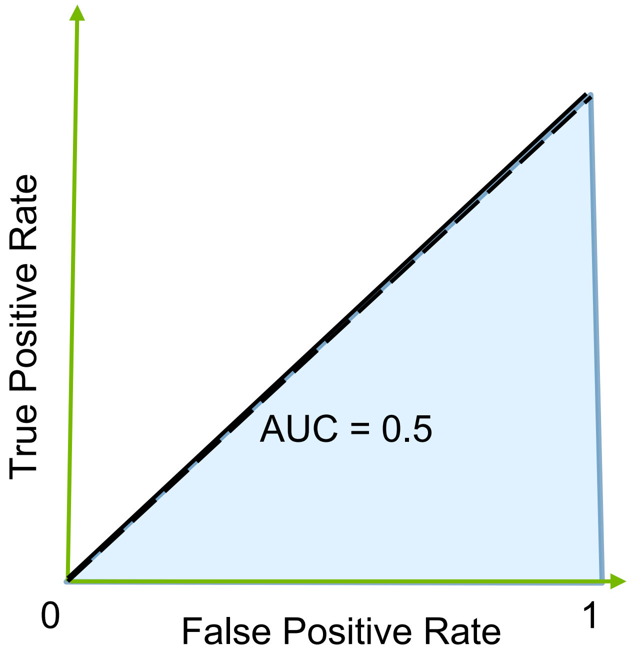




ROC CURVE

Interpretation



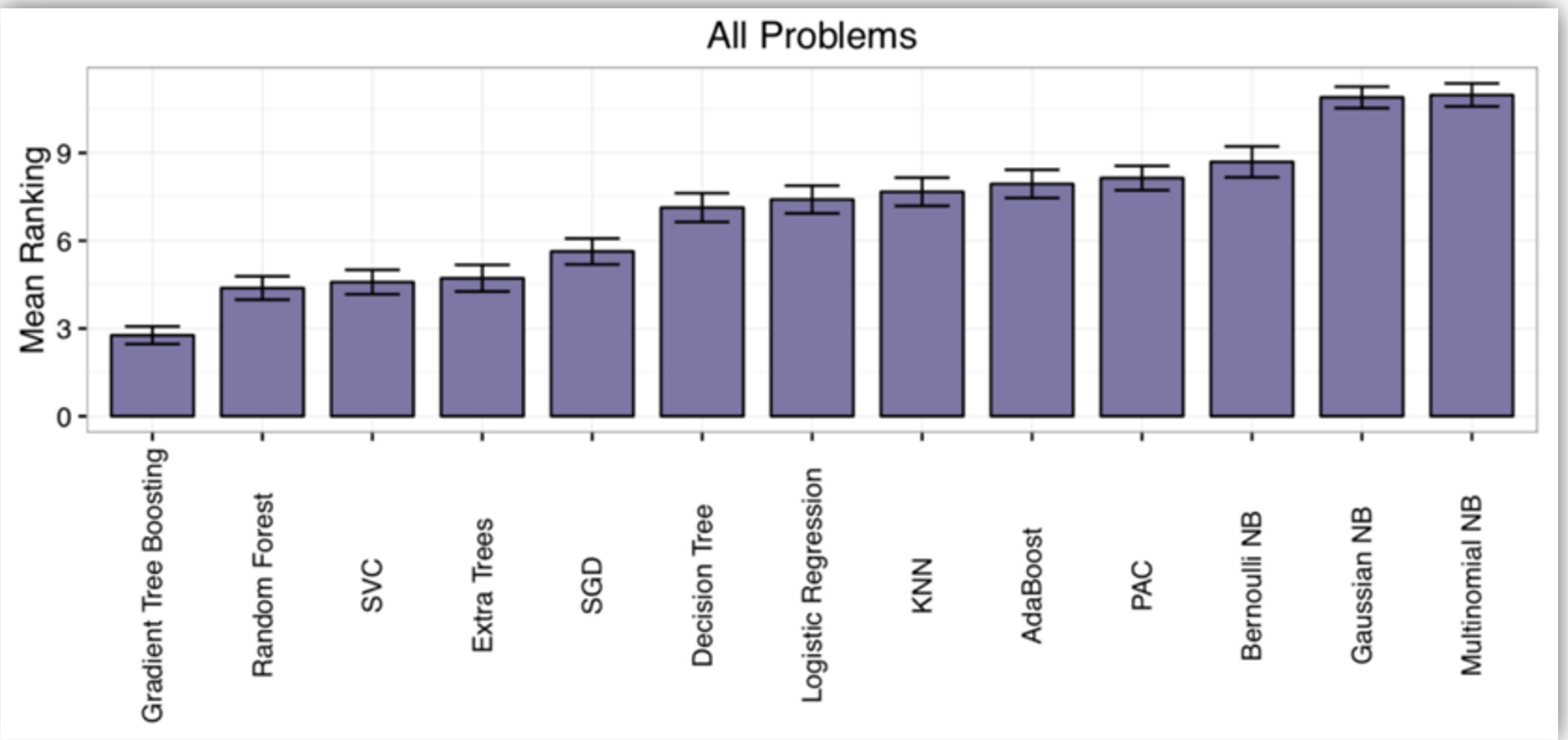


Better the separation between the classes = Better Model / Classifier

WHICH ML ALGORITHM PERFORMED BEST

Average rank across 165 ML datasets

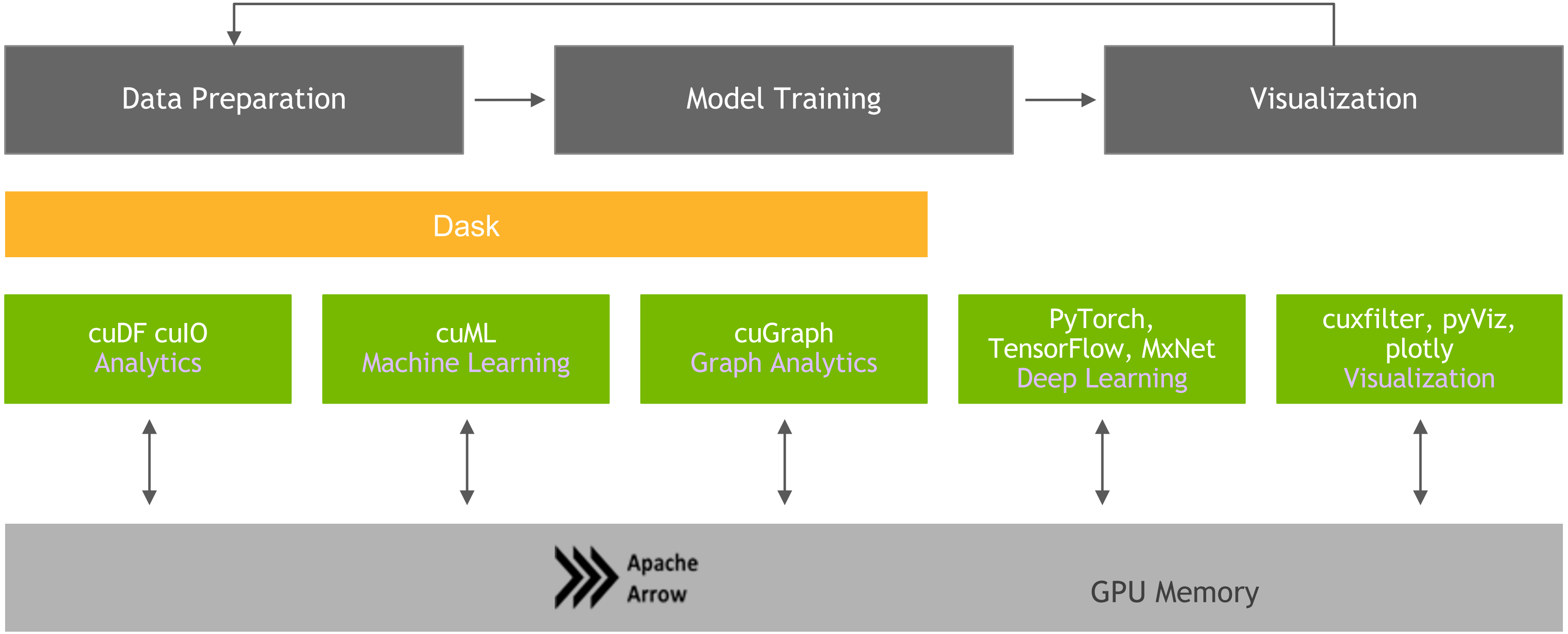
Lower
Is
Better



Source: https://goo.gl/R8Y8Pp

RAPIDS

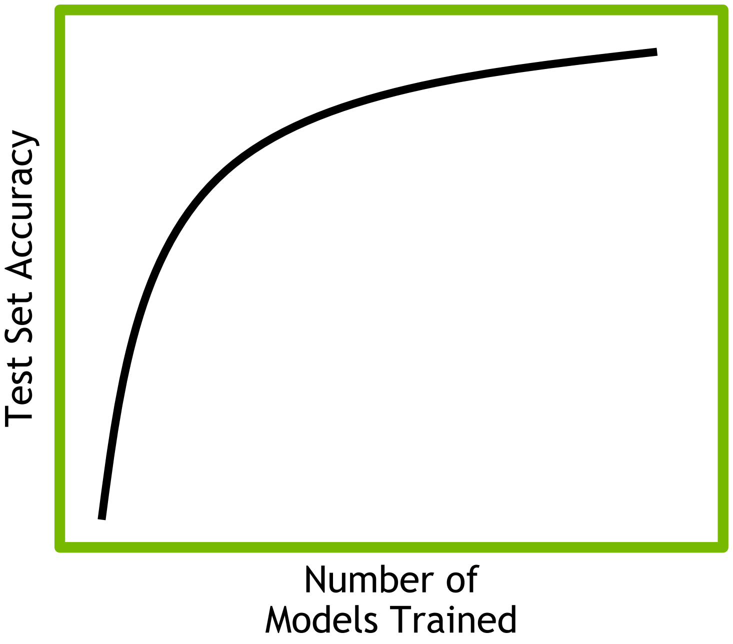
End-to-End Accelerated GPU Data Science



WHY RAPIDS + XGBOOST?

TIME TO TRAIN

Rapid Data Science



Model Selection and Hyper-Parameter Tuning

```
best_model = init_model

for (m,h) in zip(models, hyperparams):

    my_model = train(m,h)

    if acc(my_model) > acc(best_model):

        best_model = my_model
```

RAPIDS WITH XGBOOST

Multi-GPU, Multi-Node, Scalability

- XGBoost:
  - Algorithm tuned for eXtreme performance and high efficiency
  - Multi-GPU and Multi-Node Support
- RAPIDS:
  - End-to-end data science & analytics pipeline entirely on GPU
  - User-friendly Python interfaces
  - Relies on CUDA primitives, exposes parallelism and high-memory bandwidth
  - Benefits from DGX system designs (NVLINK, NVSWITCH, dense compute platform)
  - Dask integration for managing workers & data in distributed environments

Work through the first reflection

1.2 Dataset Modification

Notice that the dataset has more anomalies than normal data. Reflect for a moment about the implications of having more anomalies might be. Reflect either here in the notebook, on a piece of paper, or with a peer sitting next to you.

```
Reflection:
```

We'll come back to test your hypothesis shortly.

Section 3: Impact of Skewed Data

As we prepared our data, we pointed out that there were more anomalies than normal data and considered the implications of this dataset skew that doesn't match the real world. Take a moment now see how adjusting our dataset impacts performance.

In [2]:
```
def reduce_anomalies(df, pct_anomalies=.01):
    labels = df['label'].copy()
    is_anomaly = labels != 'normal.'
    num_normal = np.sum(~is_anomaly)
    num_anomalies = int(pct_anomalies * num_normal)
    all_anomalies = labels[labels != 'normal.']
    anomalies_to_keep = np.random.choice(all_anomalies.index, size=num_anomalies, replace=False)
    anomalous_data = df.iloc[anomalies_to_keep].copy()
    normal_data = df[~is_anomaly].copy()
    new_df = pd.concat([normal_data, anomalous_data], axis=0)
    return new_df
```

In [ ]:
```
df = reduce_anomalies(df)
```

Let's see what anomalies we have after the reduction.

In [ ]:
```
pd.DataFrame(df['label'].value_counts())
```

Return to data preprocessing and rerun cells to this point, comparing and contrasting performance. Again, reflect below, on paper, or with a peer. Reflect on *why* the reduction of anomalies had the impact that it did.

What was the impact of reducing anomalies in the dataset and why do you think that is?

```
Answer:
```

Multi-Class Classifier Challenge

In the field below, set up `dtrain`, `dtest`, `evals`, and `model` as exemplified when we trained our binary classifier.

Note: Multiclass labels are in y_train and y_test. Hint: Control F will help you find `dtrain`, `dtest`, `evals` and `model`.

You can see how adding multiple classes doesn't increase the complexity in training this type of model.

In [ ]:
```
%%time


dtrain = ##SEE BINARY CLASSIFIER FOR HINT##
dtest = ##SEE BINARY CLASSIFIER FOR HINT##
evals = ##SEE BINARY CLASSIFIER FOR HINT##
model = ##SEE BINARY CLASSIFIER FOR HINT##
```

CATEGORICAL FEATURES

One-hot encoding vs Entity Embedding

	X1	C1	C2
1	Xa	C1a	C2a
2	Xb	C1b	C2a
3	Xc	C1b	C2c

One-hot encoding

	X1	C1a	C1b	C1c	C2a	C2b	C2c
1	Xa	1	0	0	1	0	0
2	Xb	0	1	0	1	0	0
3	Xc	0	1	0	0	0	1

C1	Embedding
C1a	[0.1,0.2,0]
C1b	[0.5,0.7,0.1]

,

C2	Embedding
C2a	[-0.5,0.2,0.4]
C2c	[0.3,0.7,0.8]

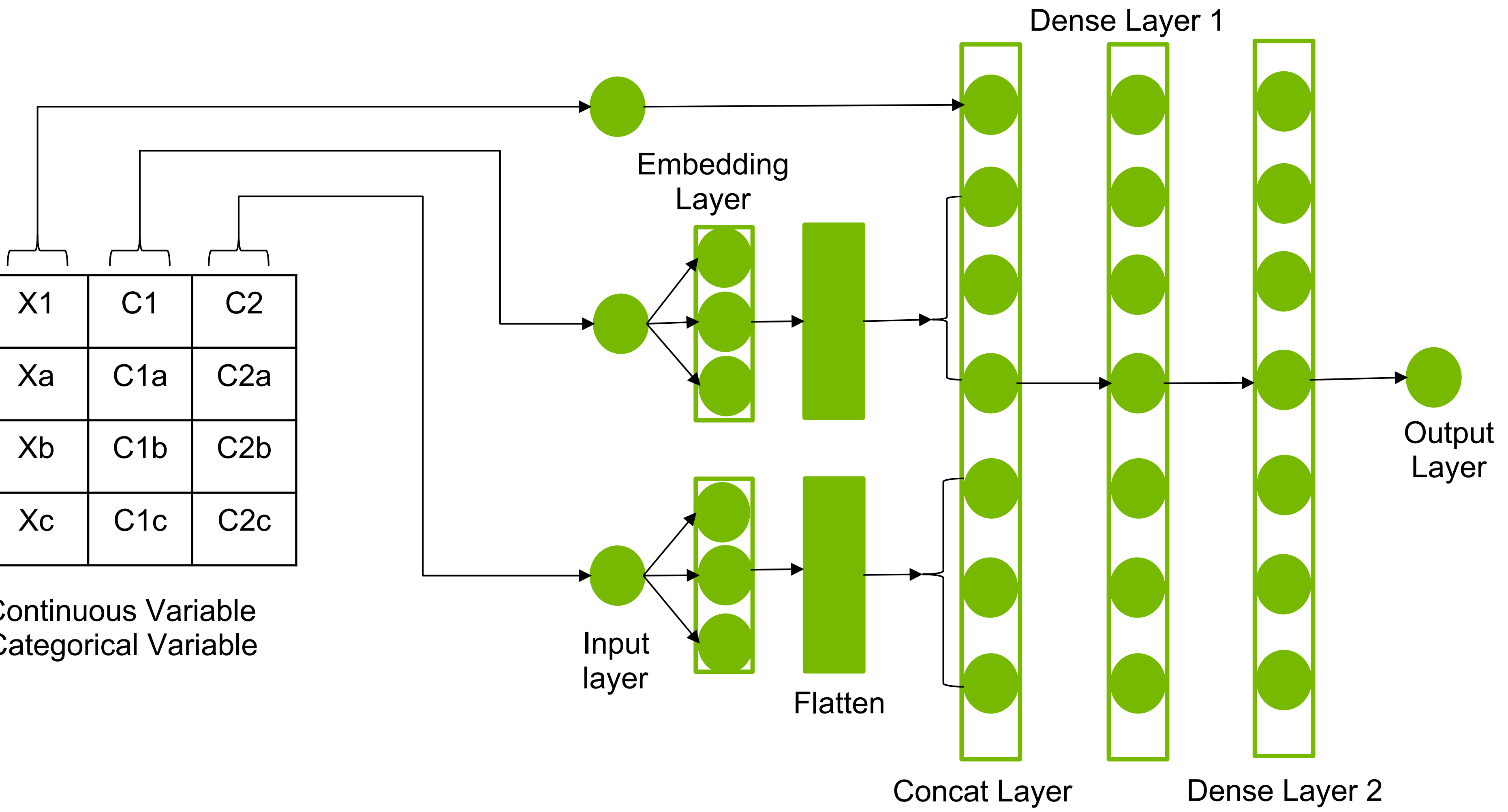
Entity Embedding

	X1	C11	C12	C13	C21	C22	C23
1	Xa	0.1	0.2	0	-0.5	0.2	0.4
2	Xb	0.5	0.7	0.1	-0.5	0.2	0.4
3	Xc	0.5	0.7	0.1	0.3	0.7	0.8

EMBEDDING CATEGORICAL FEATURES

	X1	C1	C2
1	Xa	C1a	C2a
2	Xb	C1b	C2b
3	Xc	C1c	C2c

X = Continuous Variable
C = Categorical Variable



ERROR METRICS

Comparing the different Metrics

Mean Absolute Error (MAE): This measures the absolute average distance between the real data and the predicted data, but it fails to punish large errors in prediction.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}|$$

Mean Square Error (MSE): This measures the squared average distance between the real data and the predicted data. Here, larger errors are well noted (better than MAE). But the disadvantage is that it also squares up the units of data as well. So, evaluation with different units is not at all justified.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$$

Root Mean Squared Error (RMSE): This is actually the square root of MSE. Also, this metrics solves the problem of squaring the units.

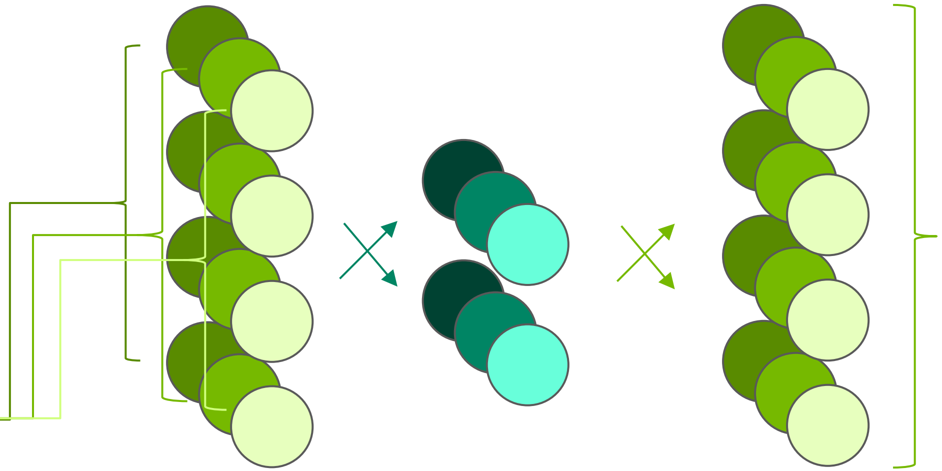
$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2}$$

Auto-Encoder on Time-series Data

Bonus

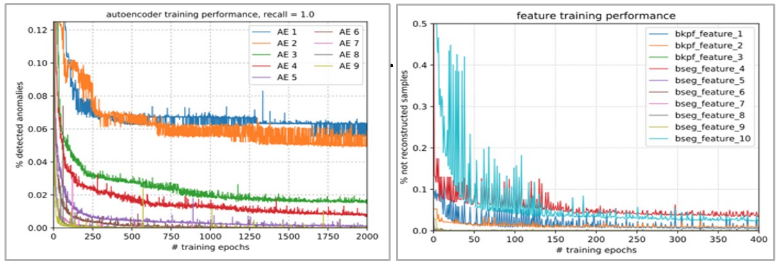
x N

Time	Feature 1	Feature 2	Feature 3
00:00:00	*Val_1*	*Val_2*	*Val_3*
00:00:01	*Val_4*	*Val_5*	*Val_6*
00:00:02	*Val_7*	*Val_8*	*Val_9*
00:00:03	*Val_10*	*Val_11*	*Val_12*



Time	Feature 1	Feature 2	Feature 3
00:00:00	*Val_1_1*	*Val_2_2*	*Val_3_3*
00:00:01	*Val_4_4*	*Val_5_5*	*Val_6_6*
00:00:02	*Val_7_7*	*Val_8_8*	*Val_9_9*
00:00:03	*Val_10_10*	*Val_11_11*	*Val_12_12*

Reconstruction Error – Training Performance



LAB 2 HANDS-ON

Lab2.ipynb

Markdown Python 3

Network Anomaly Detection using Autoencoders ¶

Welcome to the second lab of this series!

In the previous lab we used XGBoost, a powerful and efficient tree based algorithm for classification of anomalies. We were able to near perfectly identify the anomalous data in the KDD99 dataset and which type of anomaly occurred. However, in the real-world labeled data can be expensive and hard to come by. Especially with network security, zero-day attacks can be the the most challenging and also the most important attacks to detect.

So how do we approach this problem?

For starters, we could have security analysts investigate the network packets and label anomalous ones. But that solution doesn't scale well and our models might have difficulty identifying attacks that haven't occurred before.

Our solution will be to use unsupervised learning. Unsupervised learning is the class of machine learning and deep learning algorithms that enable us to draw inferences from our dataset without labels.

In this lab we will use autoencoders (AEs) to detect anomalies in the KDD99 dataset. There are a lot of advantages to using autoencoders for detecting anomalies. One main advantage is the that AEs can learn non-linear relationships in the data.

While we will not be using the labels in the KDD99 dataset explicitly for model training, we will be using them to evaluate how well our model is doing at detecting the anomalies. We will also use the labels to see if the AE is embedding the anomalies in latent space according to the type anomaly.

[1]:
```
# Import libraries that will be needed for the lab
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from IPython.display import Image

from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import roc_curve, auc, confusion_matrix
```

Choose one

`How does the ratio of anomalies to normal data impact results and why?`

Recall that when using XG Boost, the ratio didn't impact training meaningfully. Anomalies were simply *a class* of our dataset, not made special in any way by their rare nature. Using AutoEncoders, you'll see that that's no longer true. We'll explore the questions of `how rare is rare enough?` and `how does that impact our ability to identify multiple classes of anomalies?`.

In the cell below, choose to either use 1% or 5% anomaly in your dataset by setting the `pct_anomalies` parameter to .01 or .05 respectively. If you are taking this in an in-person workshop, choose a partner and do both so you can compare and contrast.

In [ ]:
```
pct_anomalies = ##.01 or .05##
```

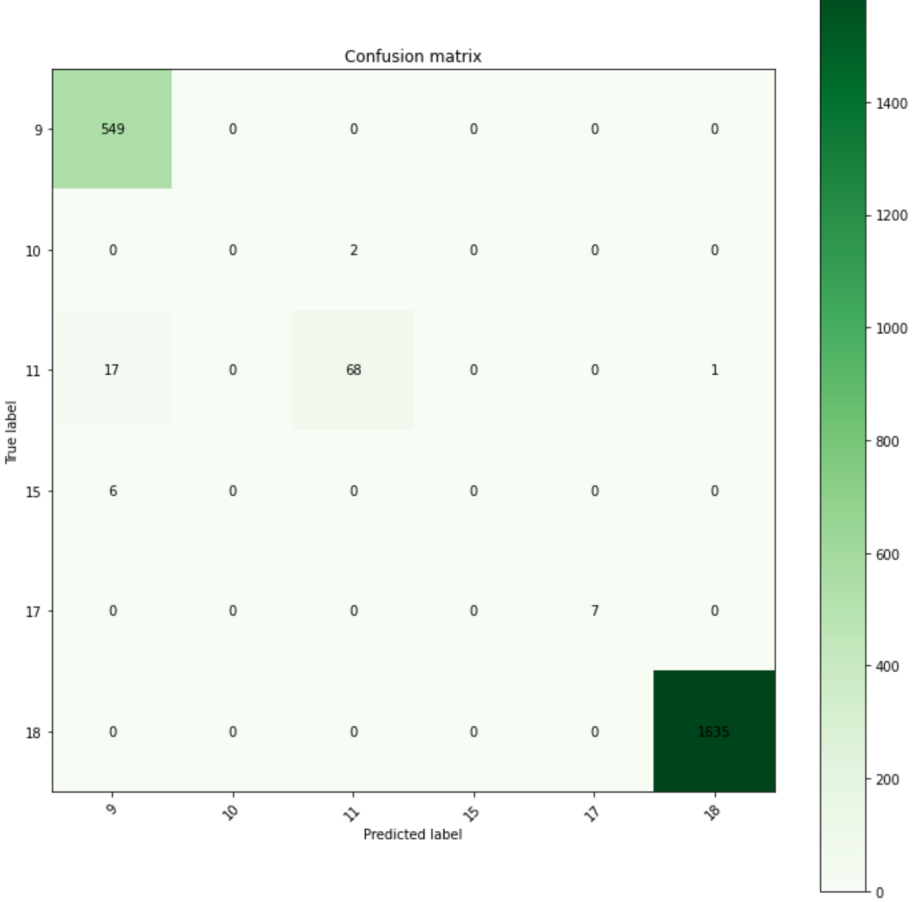
In [ ]:
```
!python preprocess_data.py --pct_anomalies $pct_anomalies
```

THANK YOU!

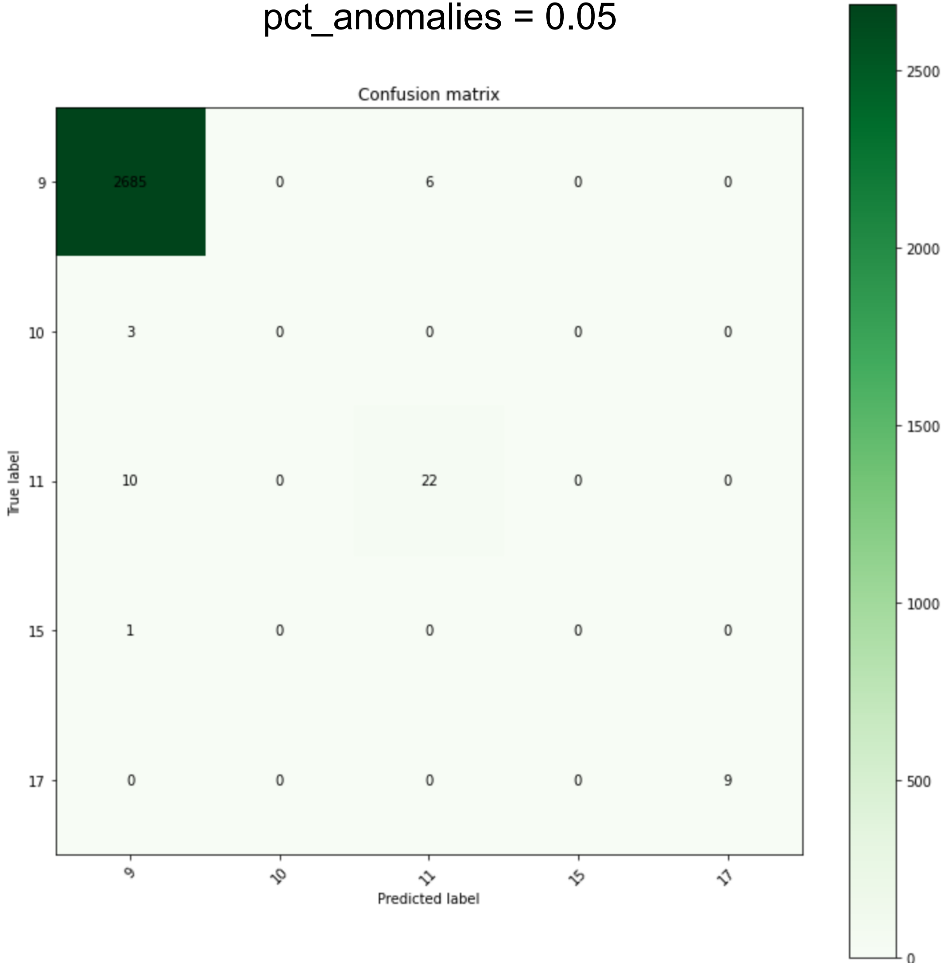
NVIDIA®

SHORT RECAP

pct_anomalies = 0.01



pct_anomalies = 0.05



LAB 3

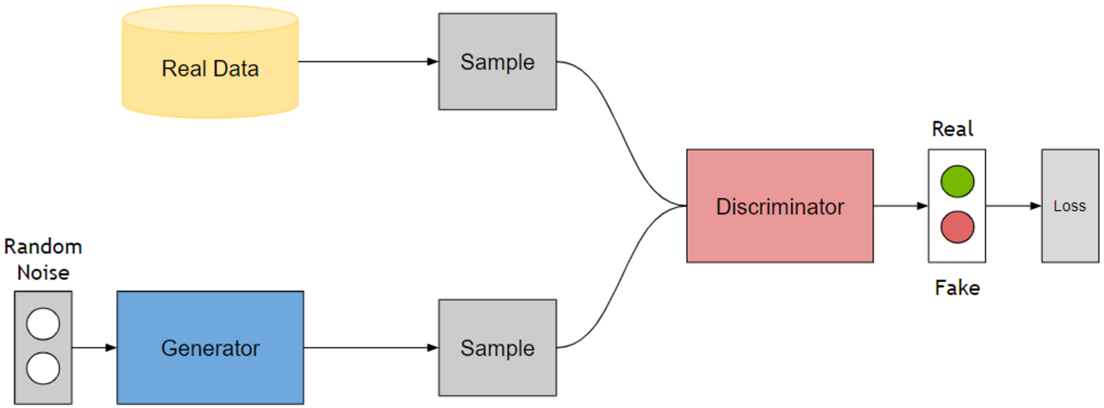
WHAT IF YOU HAD NO IDEA WHAT YOUR ANOMALIES ARE GOING TO LOOK LIKE?

LAB 3

(OR)
YOUR DATA DOES NOT FOLLOW A GAUSSIAN DISTRIBUTION?

GENERATIVE ADVERSARIAL NETWORKS

- A generative model that learns to generate samples that have the same characteristics as the samples in the dataset.
- The Generator, `G`, produces fake samples
- The discriminator, ‘D’, receives samples from both G and the dataset.
- During Training: The generator tries to fool the discriminator by outputting values that resemble real data and the discriminator tries to become better at distinguishing between the real and fake data.



GAN APPLICATIONS

Figure 5: 1024 × 1024 images generated using the CELEBA-HQ dataset. See Appendix F for a larger set of results, and the accompanying video for latent space interpolations.

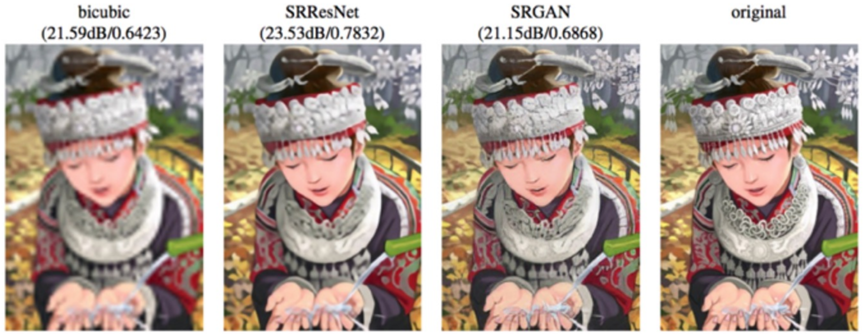
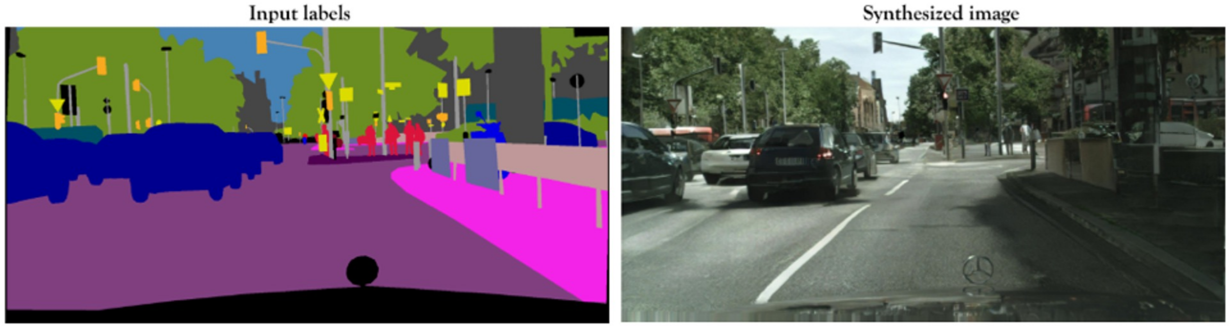


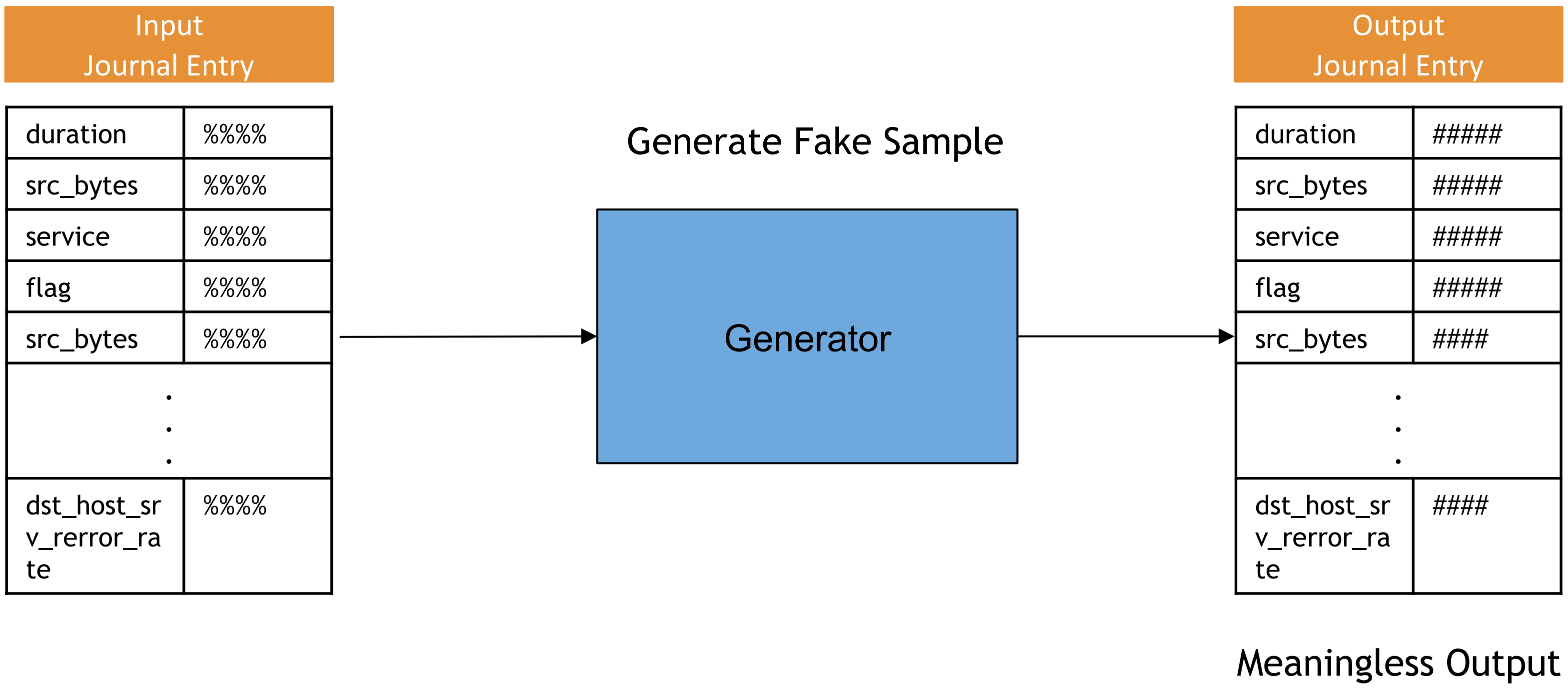
Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]



pix2pixHD

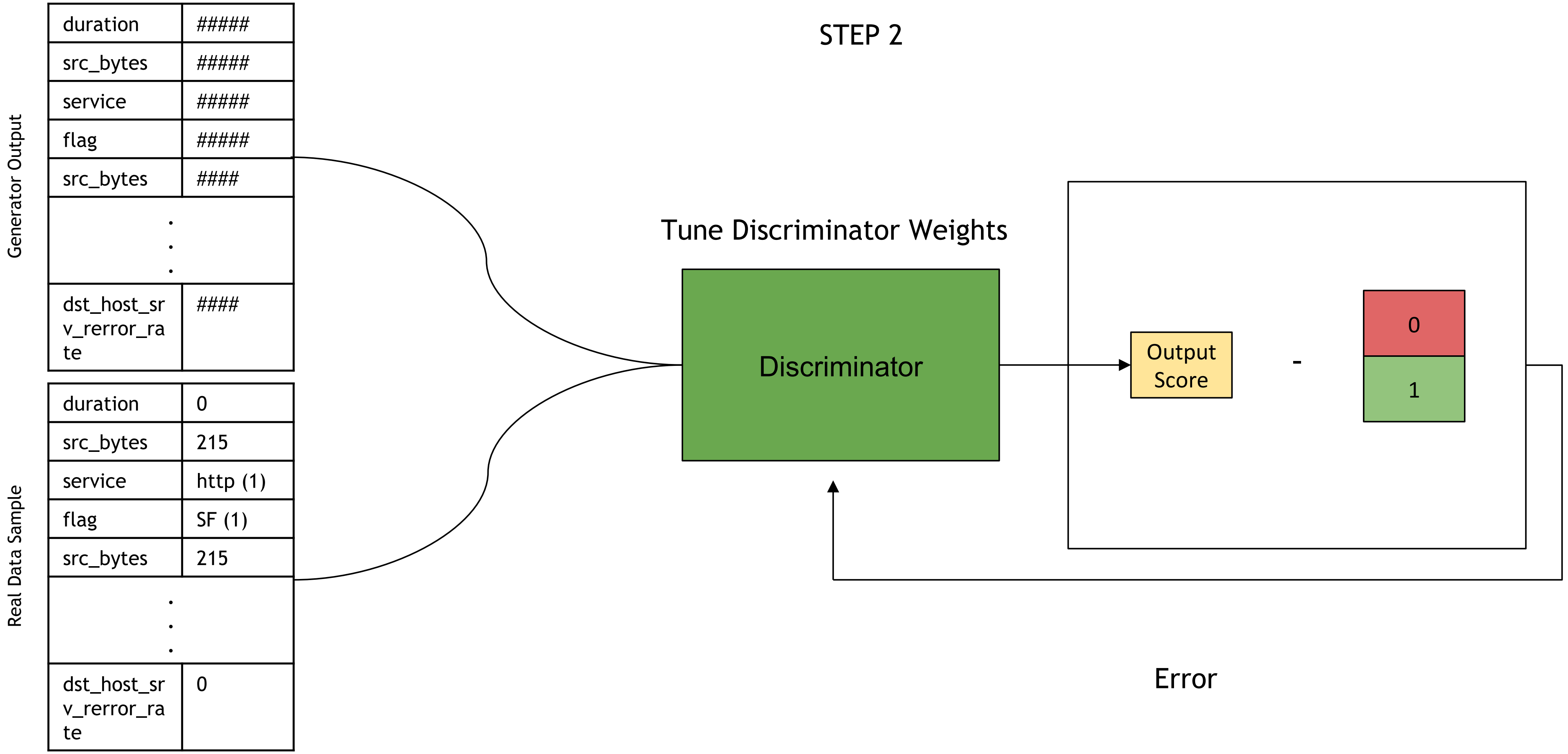
TRAINING THE GAN

STEP 1



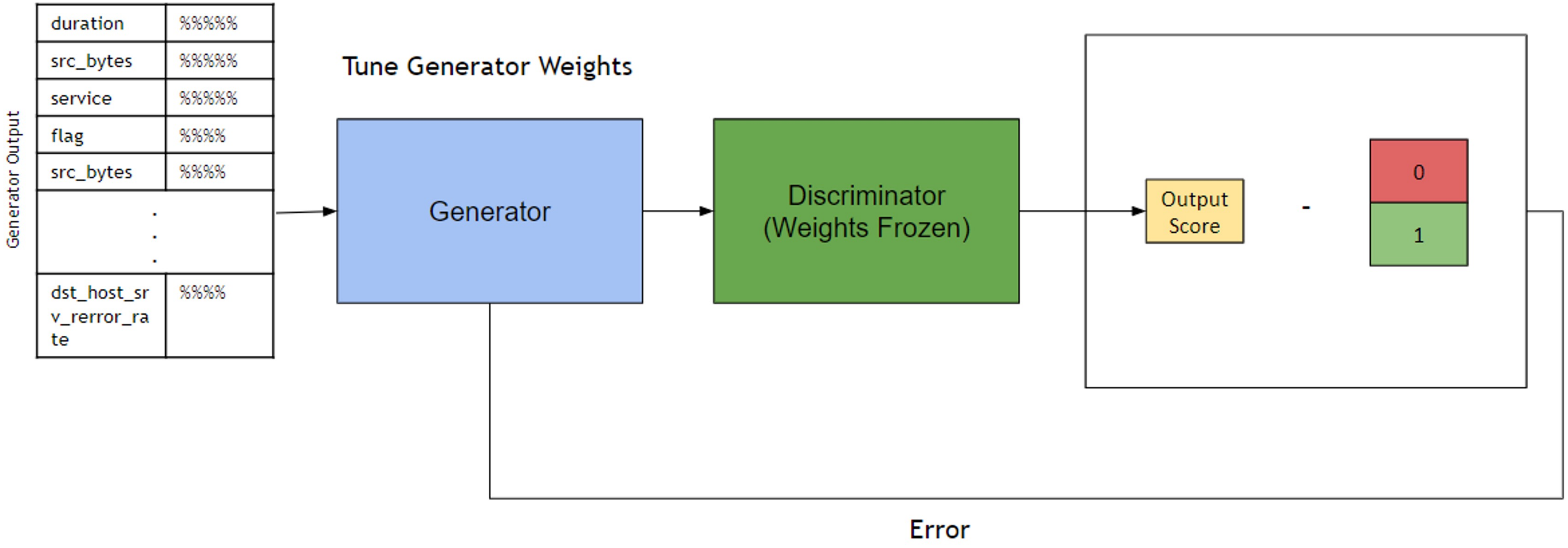
TRAINING THE GAN

STEP 2



TRAINING THE GAN

STEP 3

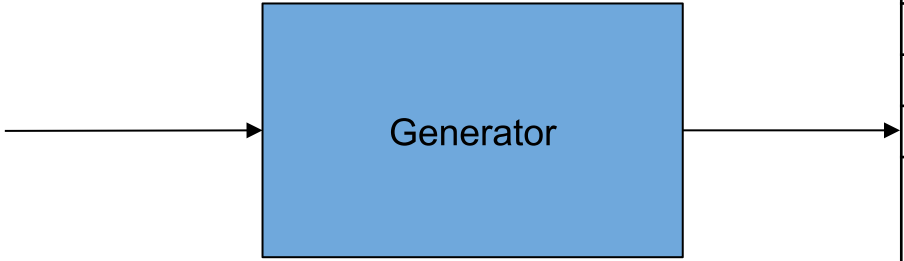


TRAINING THE GAN

BACK TO STEP 1

Input Journal Entry	
duration	%%%%
src_bytes	%%%%
service	%%%%
flag	%%%%
src_bytes	%%%%
. . .	
dst_host_srv_rerror_rate	%%%%

Generate Fake Sample

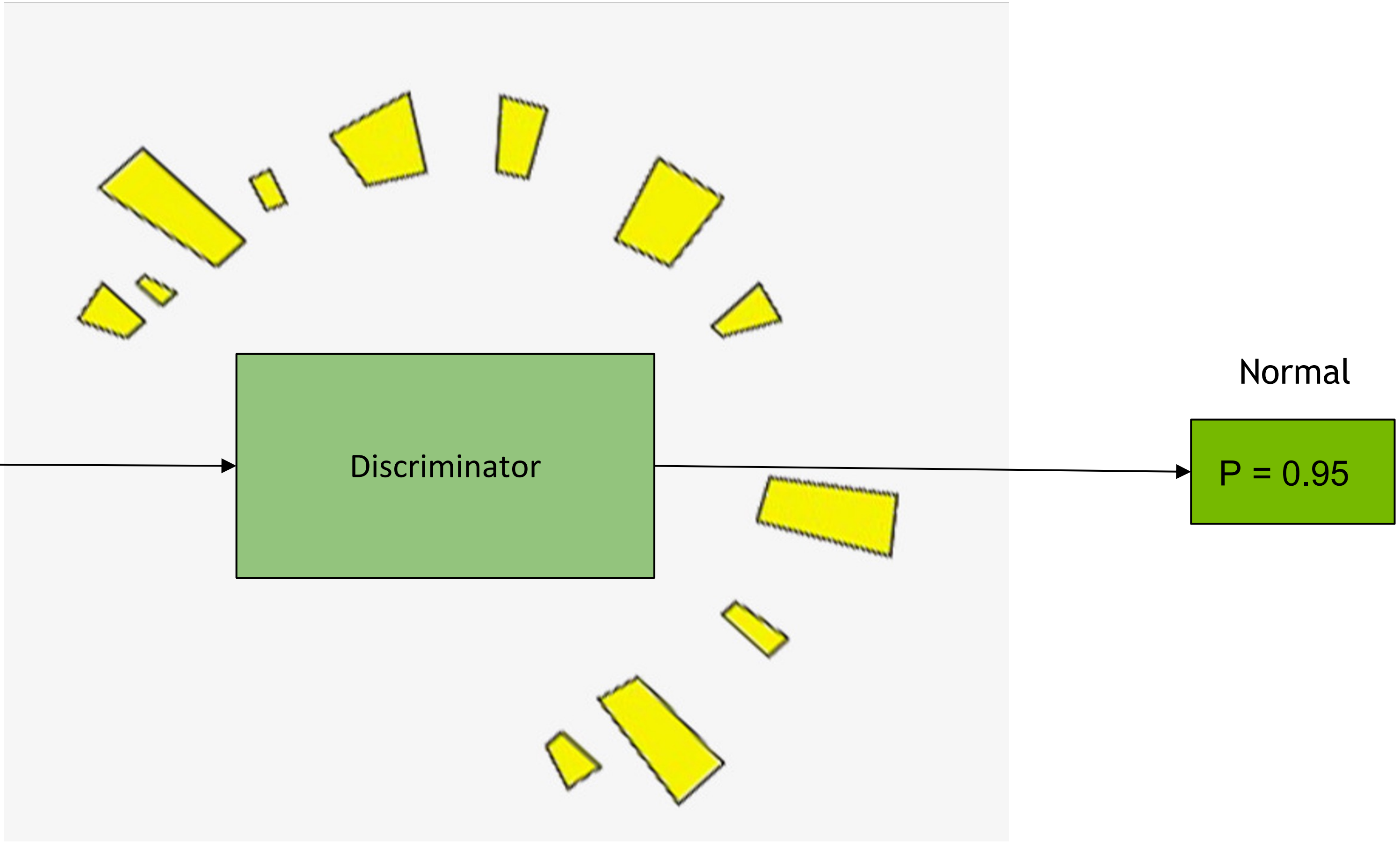


Output Journal Entry	
duration	0
src_bytes	215
service	http (1)
flag	SF (1)
src_bytes	215
. . .	
dst_host_srv_rerror_rate	0

Meaningful Output

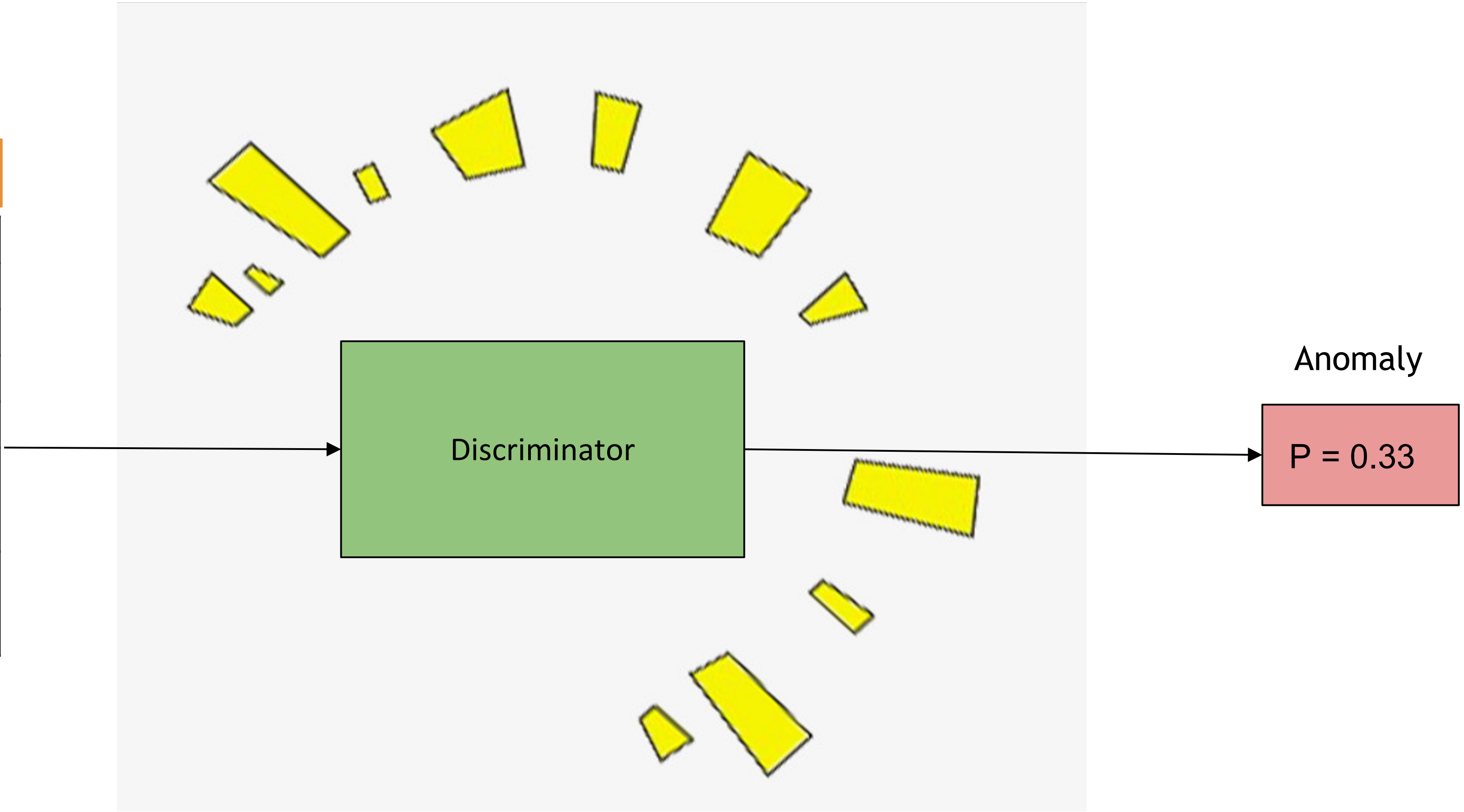
ANOMALY DETECTION

Input Test Point	
duration	0
src_bytes	215
service	http (1)
flag	SF (1)
src_bytes	215
. . .	
dst_host_srv_rerror_rate	0



ANOMALY DETECTION

Input Test point	
duration	99999
src_bytes	3239862
service	5454
flag	SF (1)
src_bytes	215
. . .	
dst_host_srv_rerror_rate	0



LAB 3 WALKTHROUGH

Lab3 (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Python 3

Run Markdown

Anomaly Detection in Network Data using GANs

- Ananth Sankar , Solutions Architect at NVIDIA.
- Eric Harper , Solutions Architect, Global Telecoms at NVIDIA.

Welcome to the third lab of this series!

In the Previous Labs we tried our hand at supervised and unsupervised anomaly detection using XgBoost and Deep Autoencoders on the KDD-99 Network Intrusion Dataset.

We addressed the issue of unlabelled training data through the use of Deep Autoencoders in the second lab. However, unsupervised methods such as PCA and Autoencoders tend to be effective only on highly correlated data such as the kdd dataset and these algorithms might also require the data to follow a Gaussian Distribution.

> "Adversarial training (also called GAN for Generative Adversarial Networks), and the variations that are now being proposed, is the most interesting idea in the last 10 years in ML, in my opinion.". Yann LeCun, 2016.

So what do GANs bring to the table and how are they different from Deep Autoencoders?

GANs are generative models that generate samples similar to the training dataset by learning the true data distribution. So instead of compressing the input into a latent space and classifying the test samples based on the reconstruction error, we actually train a classifier that outputs a probability score of a sample being Normal or Anamolous as demonstrated by Zenati, H., Foo, C., Lecouat, B., Manek, G. and Chandrasekhar, V. in their paper Efficient GAN-Based Anomaly Detection . As we will see later in the lab, this has positioned GANs as very attaractive unsupervised learning techniques.

GANs can be pretty tough to train and improving their stability is an active area of research today.

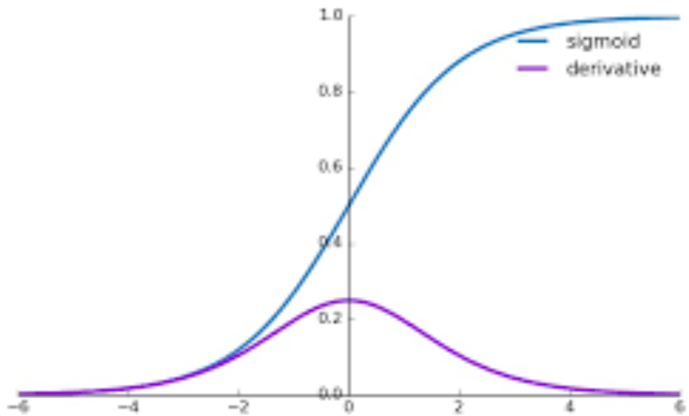
In [1]:

```
# Import libraries that will be needed for the lab
import os
import sys
import time
import logging
```

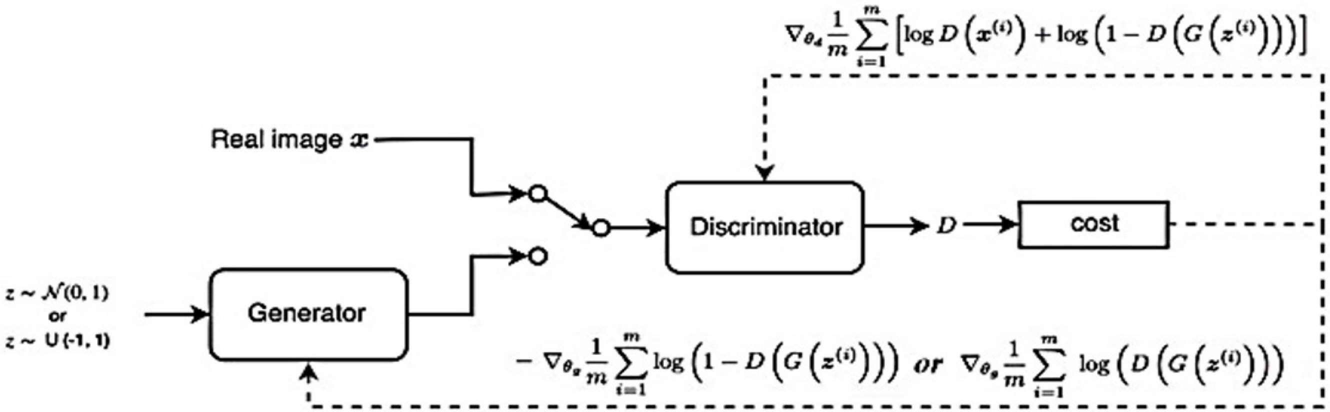
CONCLUSIONS

RECAP OF GANs

Problem of Vanishing Gradients



As hidden layers increase the partial derivative terms starts becoming smaller and smaller.



The discriminator doesn't provide enough information for the generator to make progress.

Weak Classifier
Weak Generator

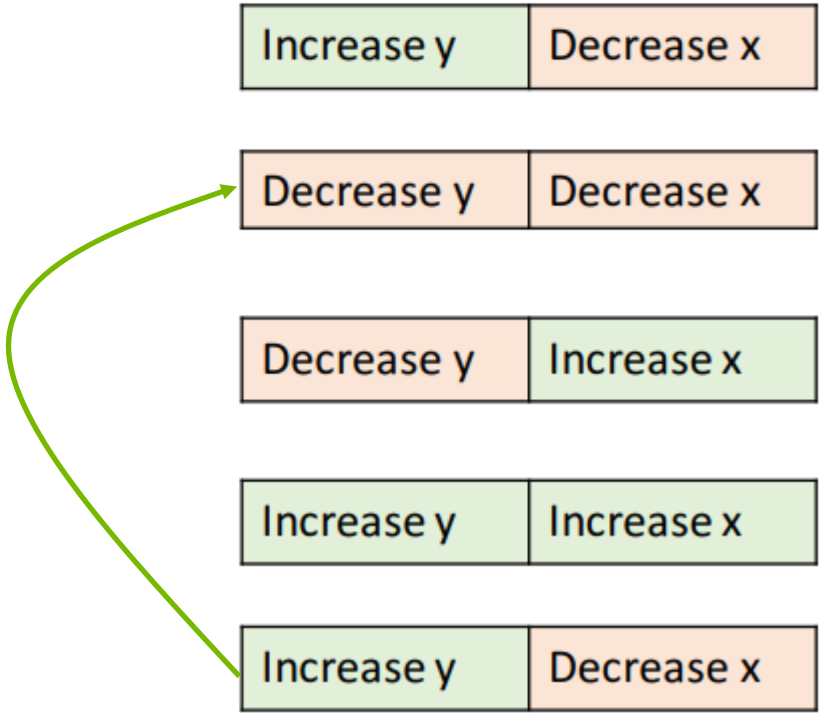
https://medium.com/analytics-vidhya/the-problems-of-generative-adversarial-networks-gans-3d887efa578e

RECAP OF GANs

Problem of Non-Convergence

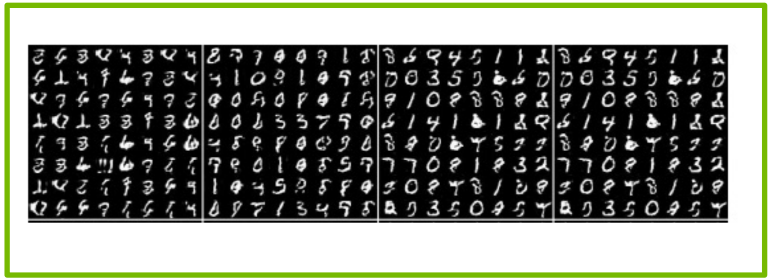
GANs involve two players

- Discriminator is trying to maximize its reward.
- Generator is trying to minimize Discriminator's reward.
- SGD was not designed to find the Nash equilibrium of a game.
- Problem: We might not converge to the Nash equilibrium at all

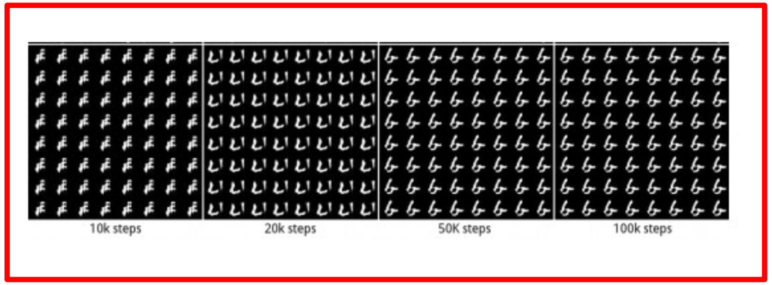


RECAP OF GANs

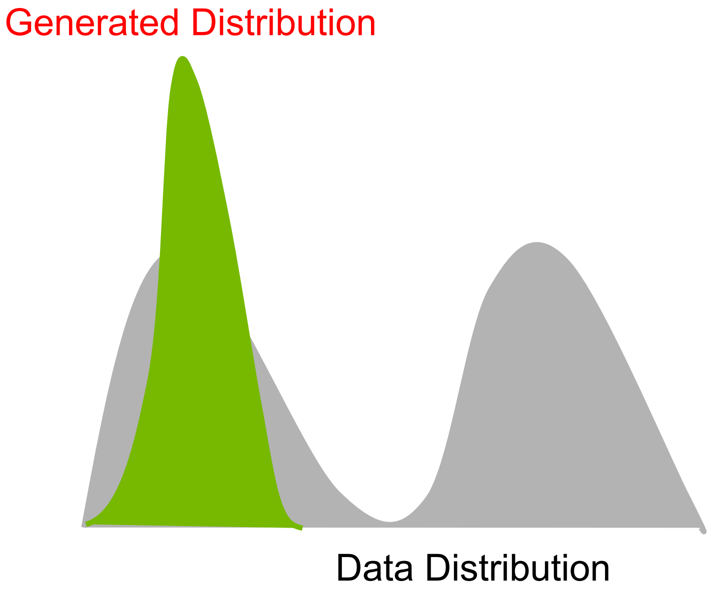
Problem of Mode Collapse



No Mode Collapse



Mode Collapse



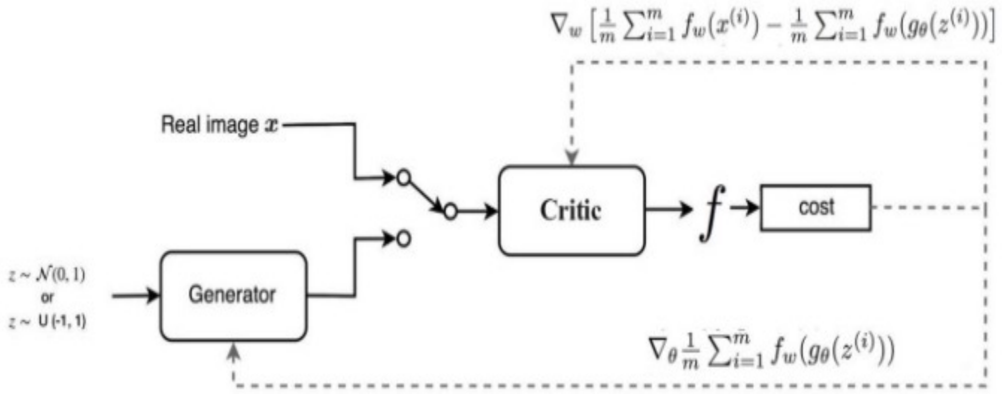
- Generated images converge to x^ that fool D the most -- most realistic from the D perspective
- Discriminator gets stuck in a local minimum and doesn't find the best strategy.
- Generator keeps producing small set of modes or output types.

RECAP OF GANS

Some Solutions - WGAN

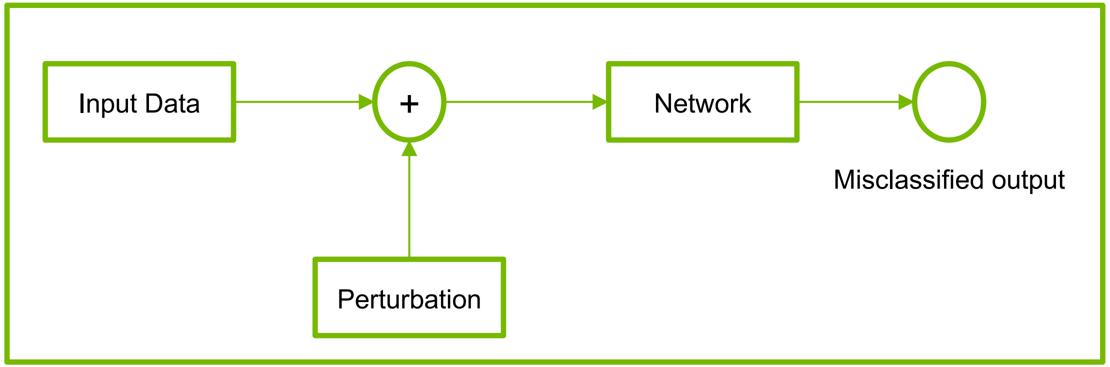
The major difference is due to the cost function:

- Discriminator does not actually classify instances rather outputs a number.
- Discriminator training just tries to make the output bigger for real instances than for fake instances => Called a “ critic” than a discriminator.
- If the discriminator gets stuck in local minima, it learns to reject the outputs that the generator stabilizes on. So the generator must try something new.
- Helps avoid problems with vanishing gradients & model collapse.



RECAP OF GANs

Adversarial Attacks



White – Box attacks :

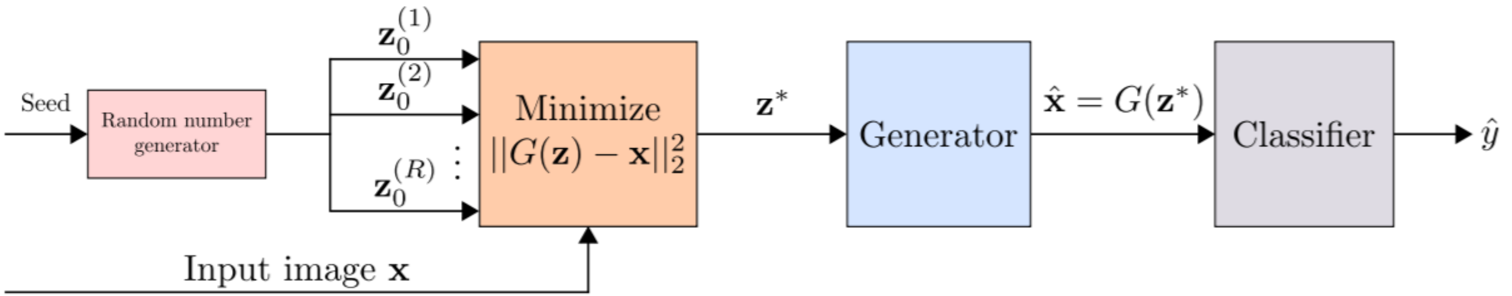
- Attackers have access to Model architecture, weights.
- Calculate the perturbation δ based on loss function.
- Attackers push the perturbed image to be misclassified to a specific target class.

Black - Box attacks :

- Attackers do not have access to the classifier or defense parameters.
- Trains a substitute model using a very small dataset augmented by synthetic images labeled by querying the classifier.
- Examples that fool the substitute end up being misclassified by the targeted classifier.

DEFENCE GAN

Pipelining a GAN with Anomaly Detection Classifier

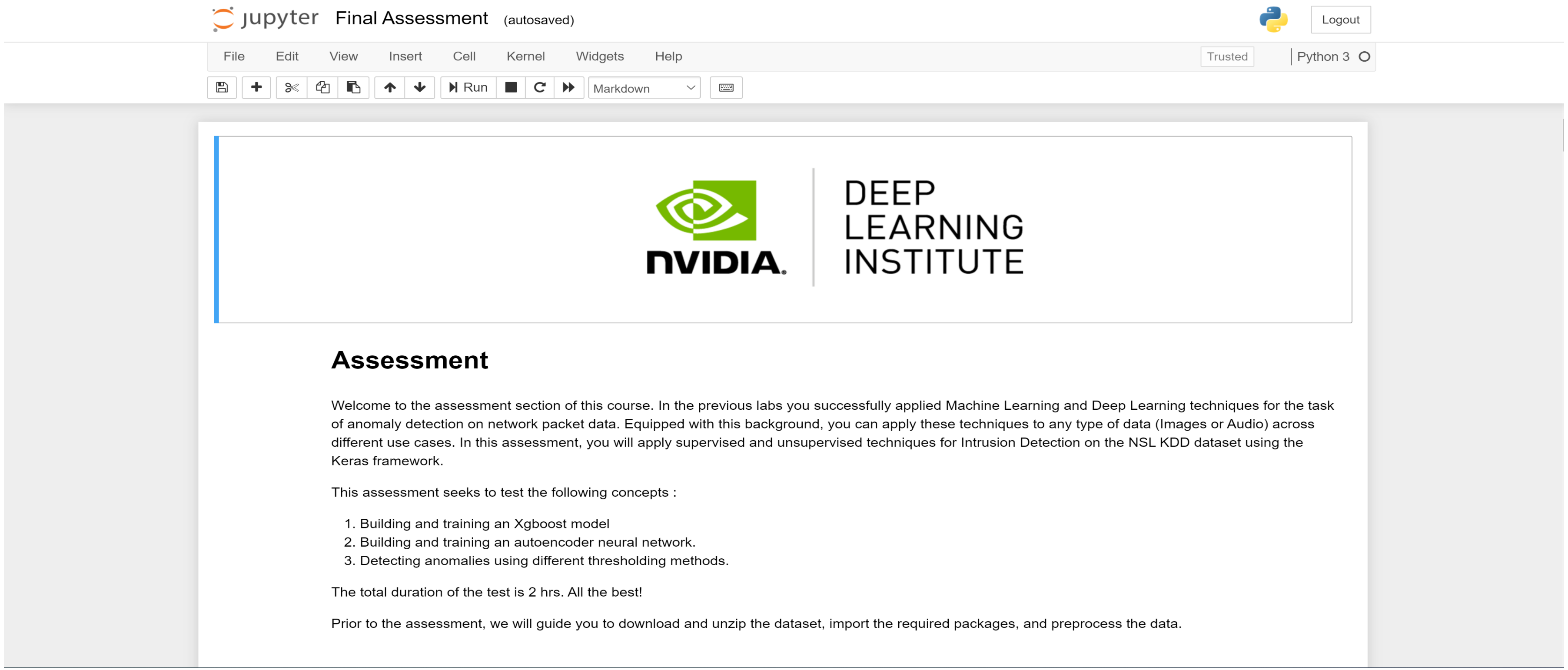


- WGAN trained on legitimate (un-perturbed) training samples to “denoise” adversarial examples.
- At test time, prior to feeding an image x to the classifier, x is projected onto the range of the generator by minimizing the reconstruction error $||G(z) - x||_2^2$ and produce output to a given image which does not contain the adversarial changes.
- The resulting reconstruction G(z) is then given to the classifier. Results in a substantial reduction of any potential adversarial noise.

https://arxiv.org/pdf/1805.06605.pdf

All the Best!

Assessment



Survey

https://survey.lrz.de/index.php/841192?lang=en

NVIDIA

Thanks!