

# Can Artificial Intelligence Secure your Infrastructure ‘?’

**A. S. M. Shamim Reza**

**Deputy Manager, NOC  
Link3 Technologies Limited**

**2020  
APRICOT  
APNIC 49**

**Melbourne, Australia**

“

---

**I propose to consider the question,  
"Can machines think?"**

– Alan Turing 1950

---

”

[~] \$whoami

- 10+ years, working for *Link3 Technologies Limited*
- InfoSec Professional
- Passionate about Artificial Intelligence

*EC-Council Certified Security Analyst*

*shamimreza@link3.net*

*sohag.shamim@gmail.com*

*@asmshamimreza* on **LinkedIn**

*@shamimrezasohag* on **Twitter**

# The Motivation

The research work was basically motivated to detect *Anomaly* in *DNS traffic*, from *NetFlow* data, incorporating Different types of *Machine Learning* models.

- What type of attacks that use DNS protocol or target DNS server?
- What are the characteristics of these attacks?
- How NetFlow conversation tells a story?
- Which Machine Learning model helps detect these attacks and how ?
- Which Machine Learning model can detect strange activities ?
- How the data has to be processed ?
- How the ML Model can be build ?

# Agenda

- What is Artificial Intelligence? And it's branches, difference in-between !
- When Machine Learning is required ?
- Classification of ML algorithm and real world example for security analysis !
- Machine learning pipelines! And the dependencies of it.
- What is DNS? And how it works !
- What is NetFlow data? And how it can help us get insights of Network Infrastructure !
- Use-Case: Anomaly detection methods to use !
- OpenSource Software & Libraries for this project! And how they works!
- Get in to the Project to detect anomalies in DNS traffic from NetFlow data with the help Machine Learning algorithm.

\*\*\* This tutorial will not make you an expert on Machine Learning, rather it will provide a hands-on knowledge to start your first project.

# What is Artificial Intelligence?

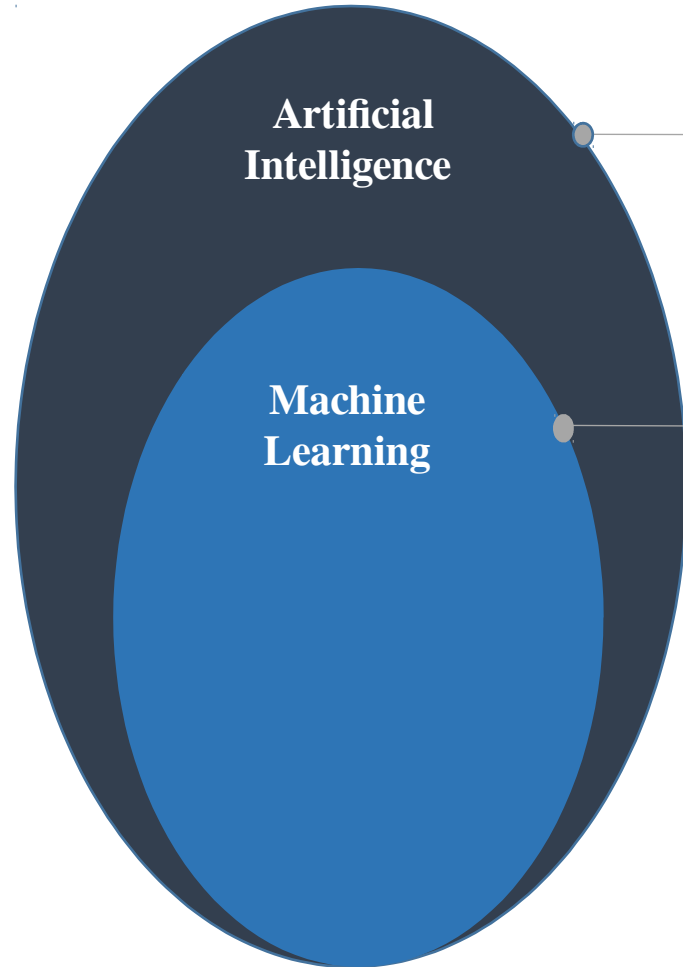


**Artificial  
Intelligence**

Predicting the future isn't magic,  
it's artificial intelligence.

- Dave Waters, Department of Earth Sciences,  
University of Oxford

# What is Artificial Intelligence?



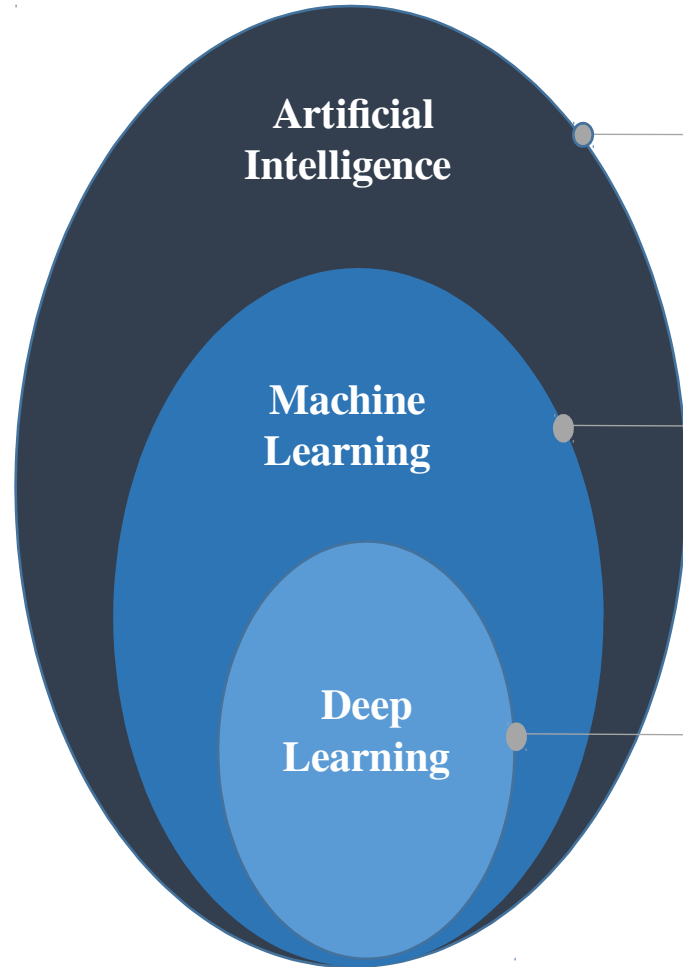
Predicting the future isn't magic,  
it's artificial intelligence.

- Dave Waters, Department of Earth Sciences,  
University of Oxford

Machine Learning is the field of  
study that gives computers the  
ability to learn without being  
explicitly programmed.

- Authur I. Samuel, Stanford University

# What is Artificial Intelligence?



Predicting the future isn't magic,  
it's artificial intelligence.

- Dave Waters, Department of Earth Sciences,  
University of Oxford

Machine Learning is the field of  
study that gives computers the  
ability to learn without being  
explicitly programmed.

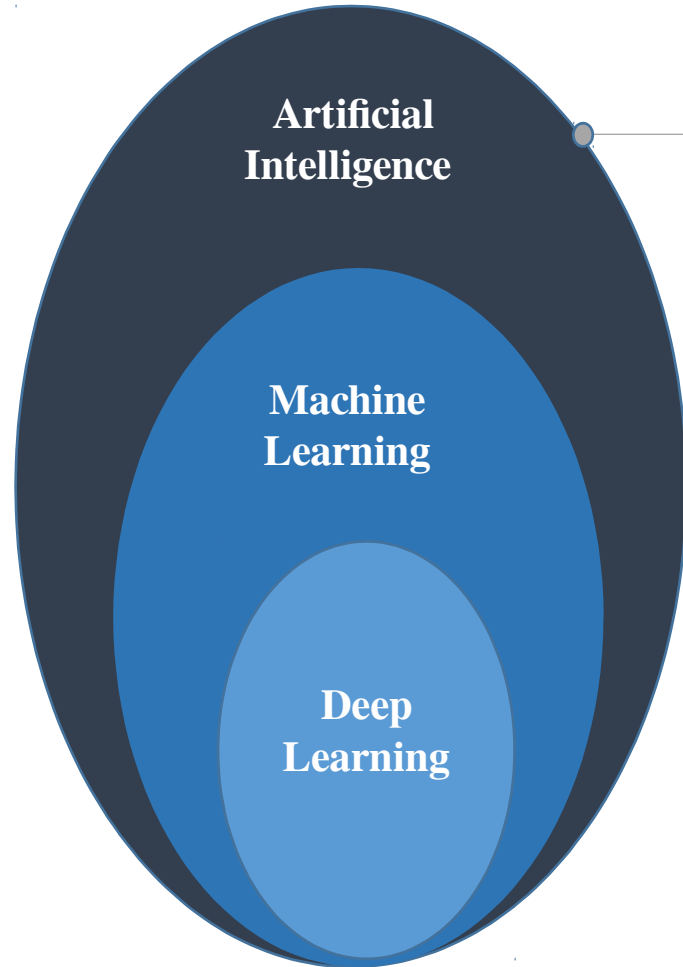
- Authur I. Samuel, Stanford University

I have worked all my life in  
Machine Learning, and I've never  
seen one algorithm knock over  
benchmarks like Deep Learning.

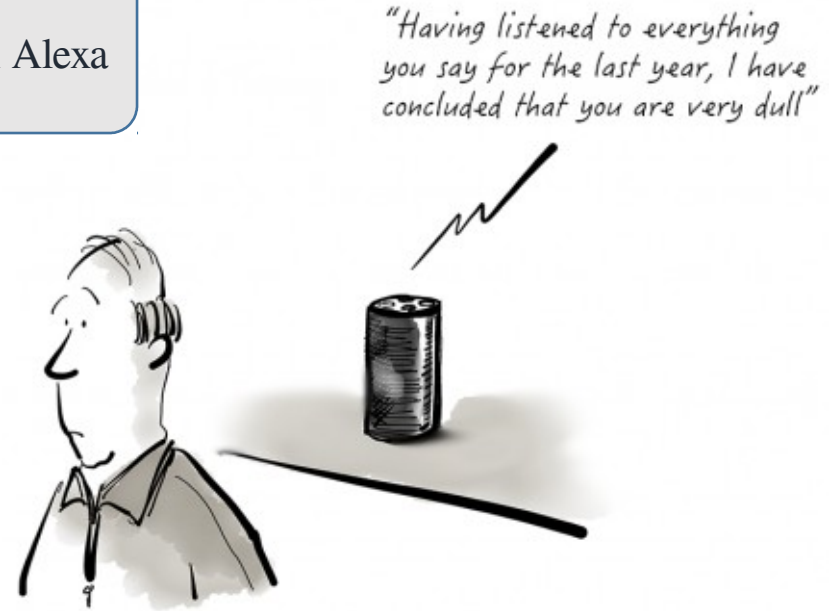
- Andrew NG, Stanford University



# Use-cases of Artificial Intelligence?

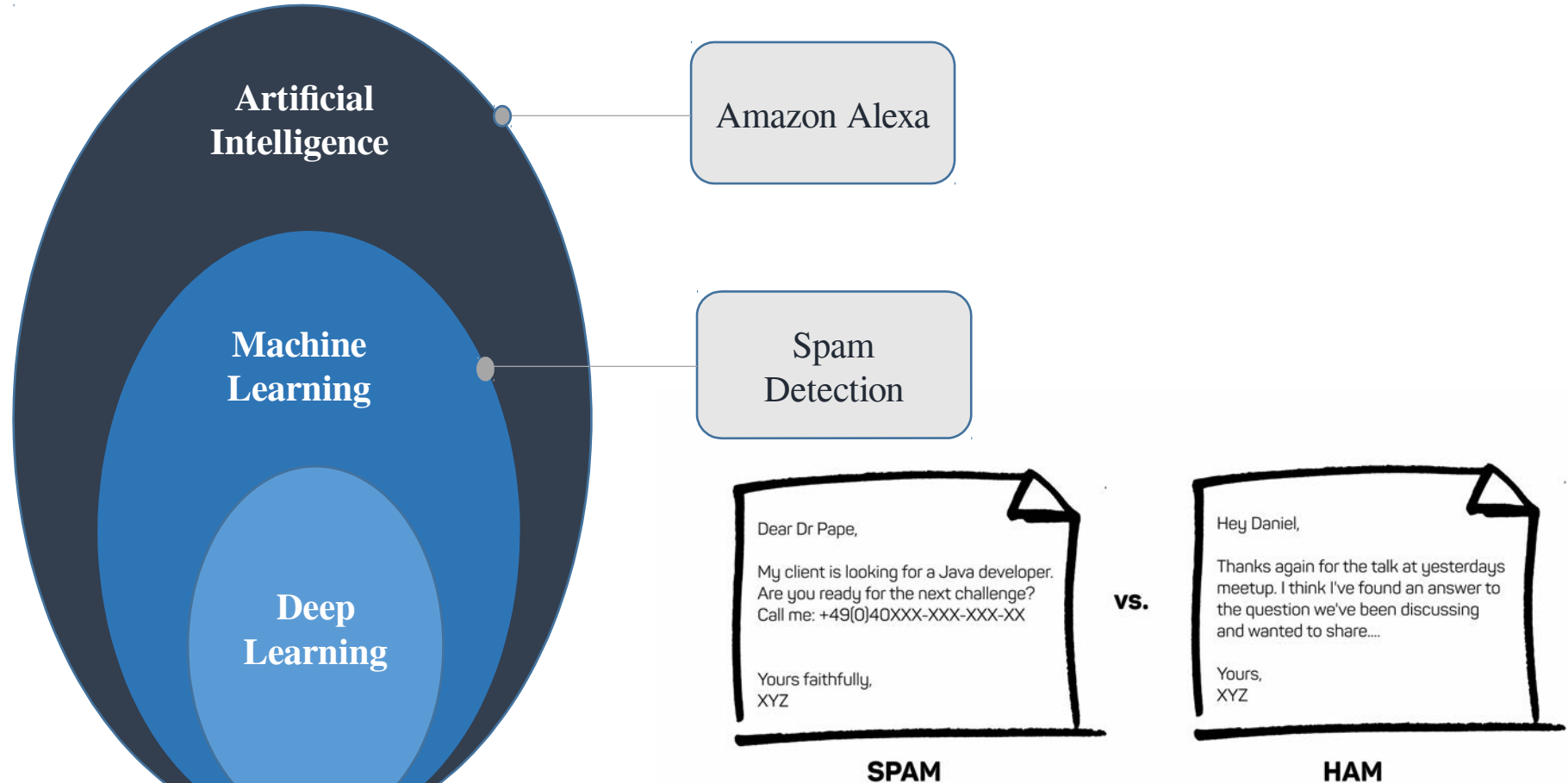


Amazon Alexa

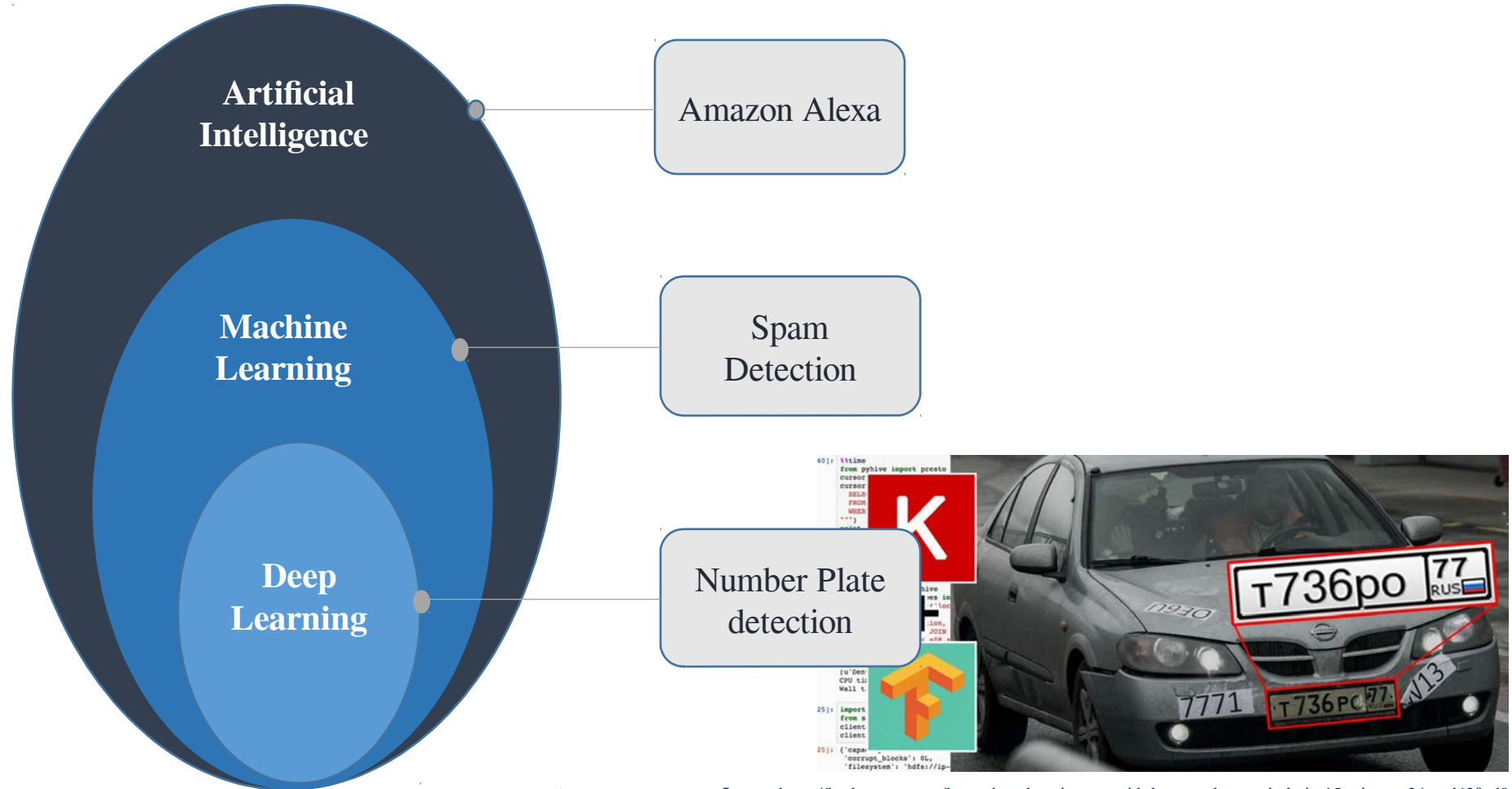


Alexa's new "honesty" skill

# Use-cases of Artificial Intelligence?



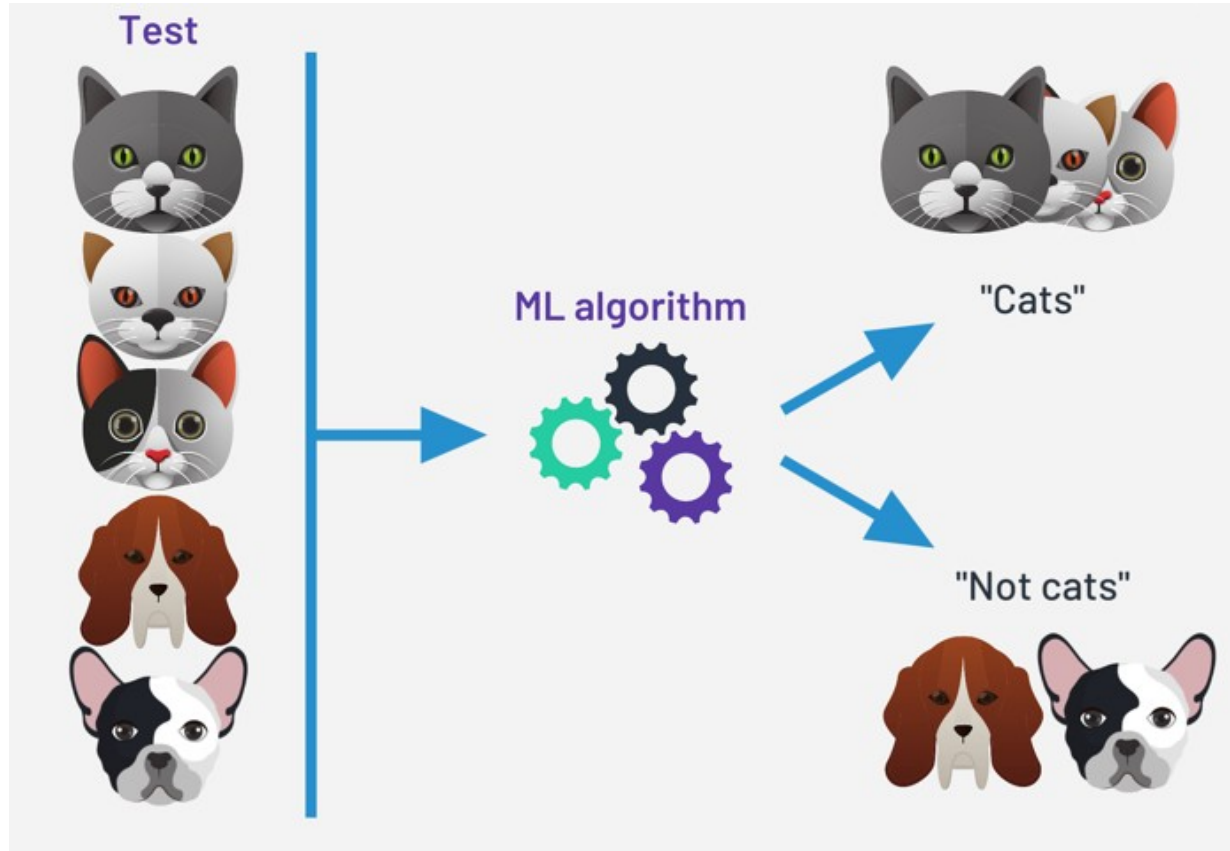
# Use-cases of Artificial Intelligence?



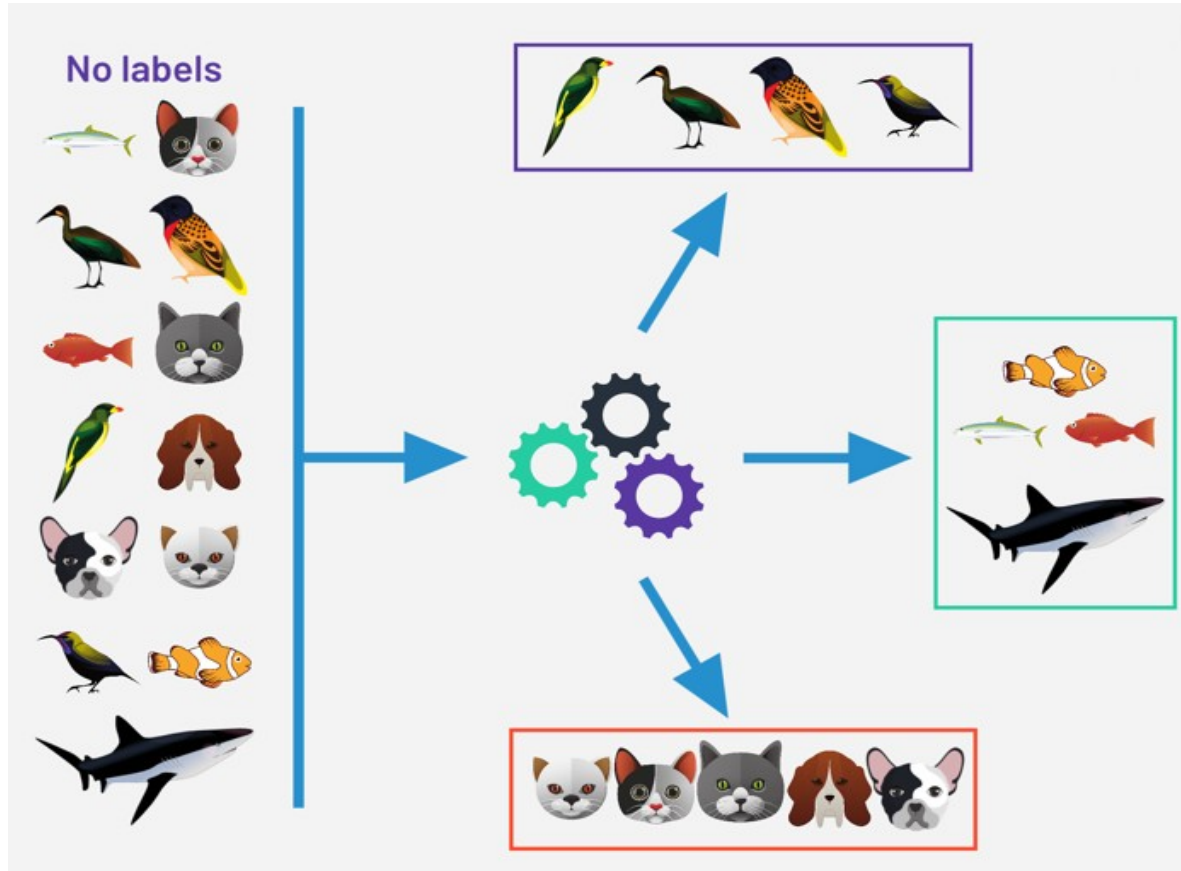
# Classification of Machine Learning

Machine Learning		
Supervised Learning	Unsupervised Learning	Re-enforcement Learning
<p>Supervised learning as the name indicates the presence of a supervisor as a teacher. Basically supervised learning is a learning in which we teach or train the machine using data which is well labeled that means some data is already tagged with the correct answer.</p> <p>For instance, suppose you are given a basket filled with different kinds of fruits. Now the first step is to train the machine with all different fruits one by one like this: If shape of object is rounded and depression at top having color Red then it will be labelled as –Apple. If shape of object is long curving cylinder having color Green-Yellow then it will be labelled as –Banana.</p>	<p>Unsupervised learning is the training of machine using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance. Here the task of machine is to group unsorted information according to similarities, patterns and differences without any prior training of data.</p> <p>Thus the machine has no idea about the features of dogs and cat so we can't categorize it in dogs and cats. But it can categorize them according to their similarities, patterns, and differences.</p>	<p>Reinforcement learning differs from the supervised learning in a way that in supervised learning the training data has the answer key with it so the model is trained with the correct answer itself whereas in reinforcement learning, there is no answer but the reinforcement agent decides what to do to perform the given task. In the absence of training dataset, it is bound to learn from its experience.</p>

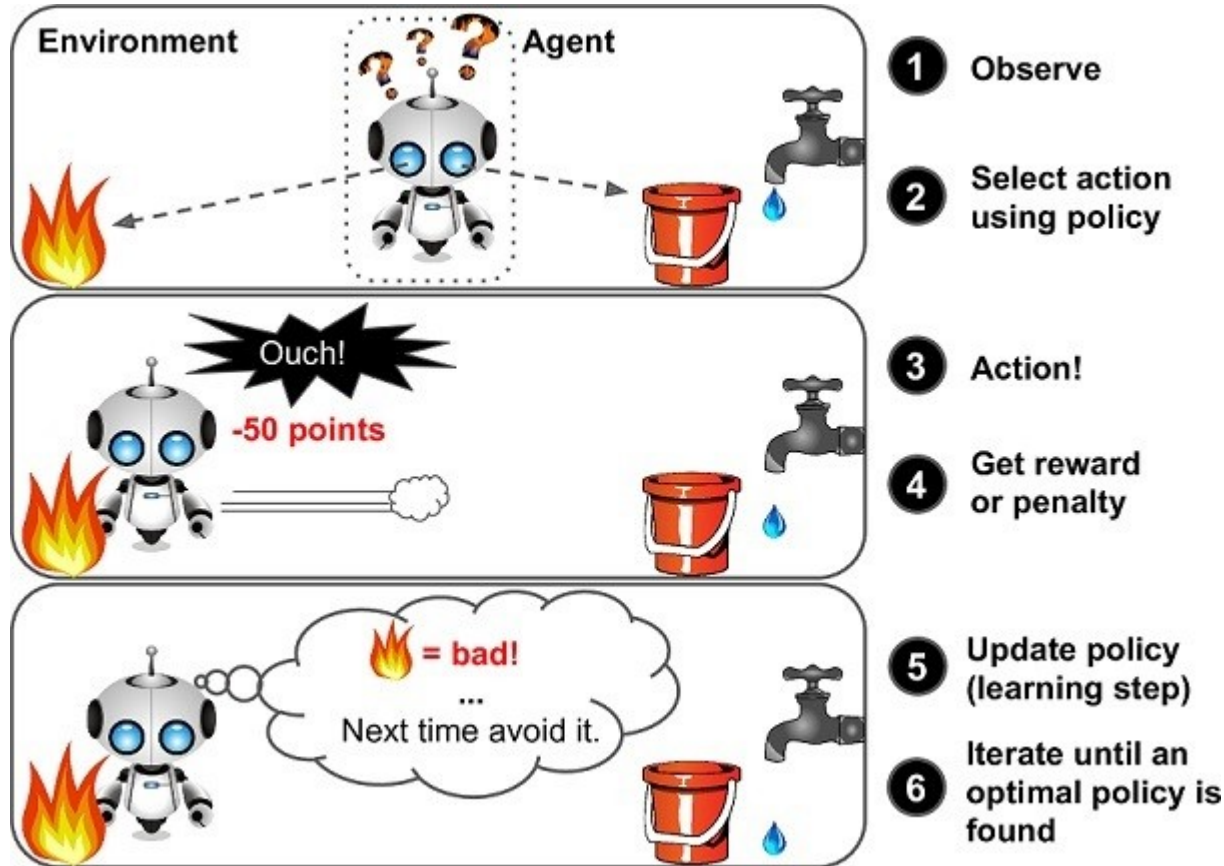
# Classification of Machine Learning



# Classification of Machine Learning



# Classification of Machine Learning



# ML Algorithm Use-cases

## Supervised Learning

### Regression

Regression is used to predict the outcome of a given sample when the output variable is in the form of real values

Example – What is the temperature tomorrow?

Algorithm –

- Linear Regression
- Non-Linear Regression
- Simple Regression
- Multiple Regression

### Classification

Classification is used to predict the outcome of a given sample when the output variable is in the form of categories.

Example – Is it Hot or Cold tomorrow?

Algorithm –

- KNN – K Nearest Neighbor
- SVM – Support Vector Machine
- Logistic Regression
- Decision Tress



# ML Algorithm Use-cases

## Unsupervised Learning

### Clustering

Clustering is sometimes called “unsupervised classification”.

Example – Say we are trying to cluster customers at a grocery store into similar groups so we can target our advertisements. It will use distance method to evaluate each cluster

Algorithm –

- K-means cluster

- Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

- Gaussian Mixture Models (GMMs)

### Dimensionality Reduction

It's a process of reducing the number of random variables under consideration, by obtaining a set of principal variables.

Example – Facial recognition

The various methods used for dimensionality reduction include –

- Principal Component Analysis (PCA)

- Linear Discriminant Analysis (LDA)

- Generalized Discriminant Analysis (GDA)

# ML Algorithm Use-cases

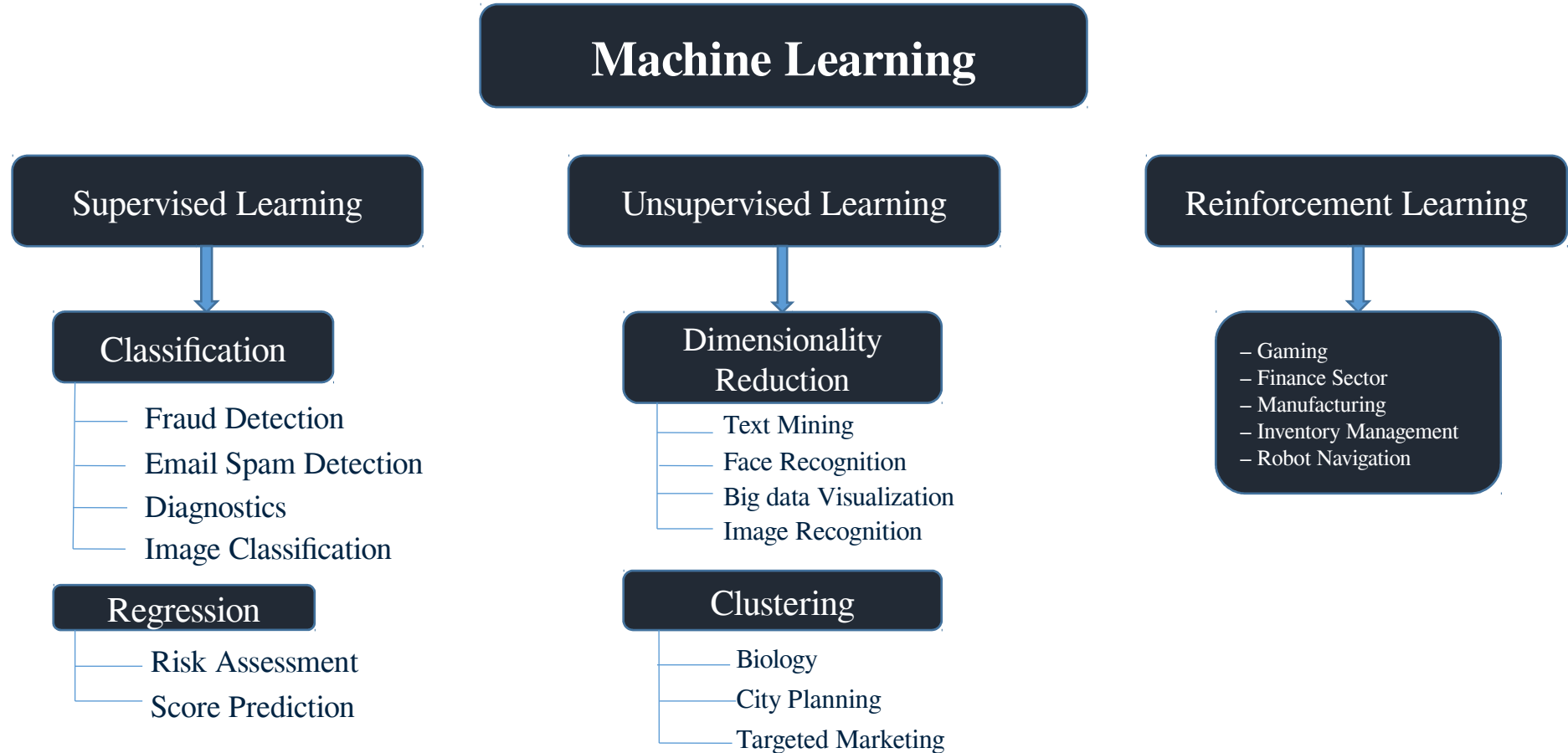
## Reinforcement Learning

In the process of learning, there is still a teacher and learner element. And we still learn the logic of our problem through feedback.

Reinforcement learning is a great example of how broad the overall “learning” approach can be.

Example – Playing chess in Computer

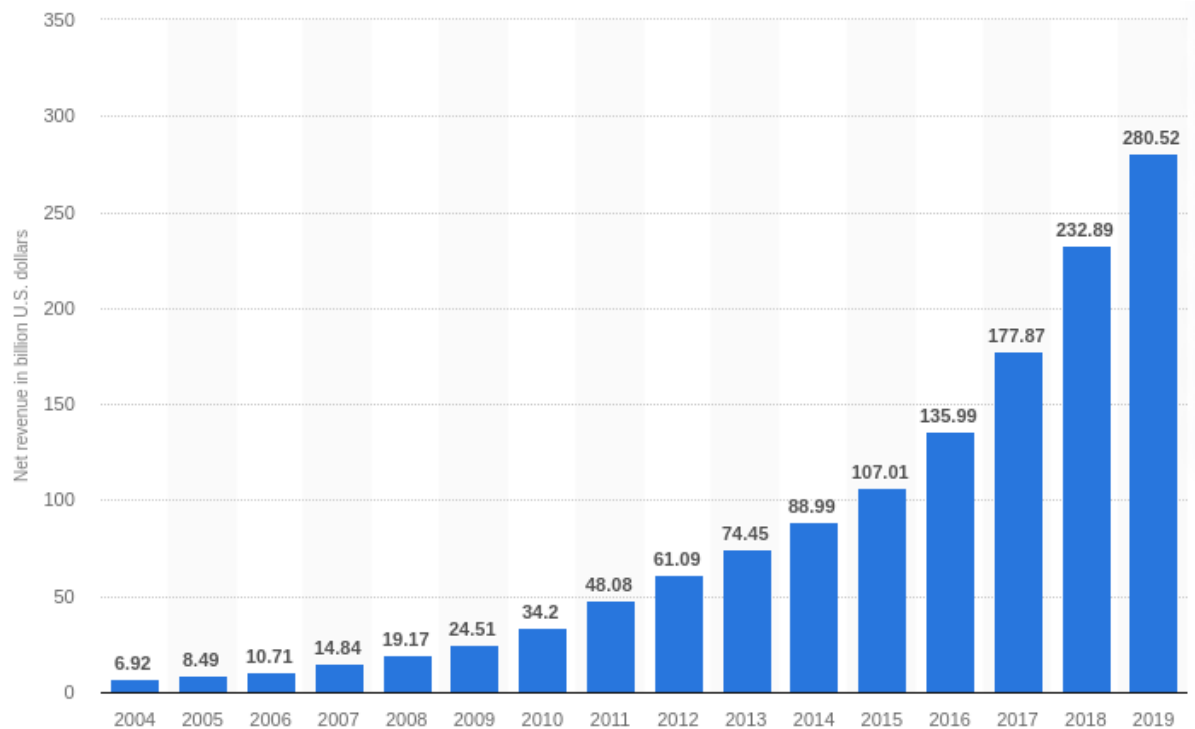
# Classification of Machine Learning



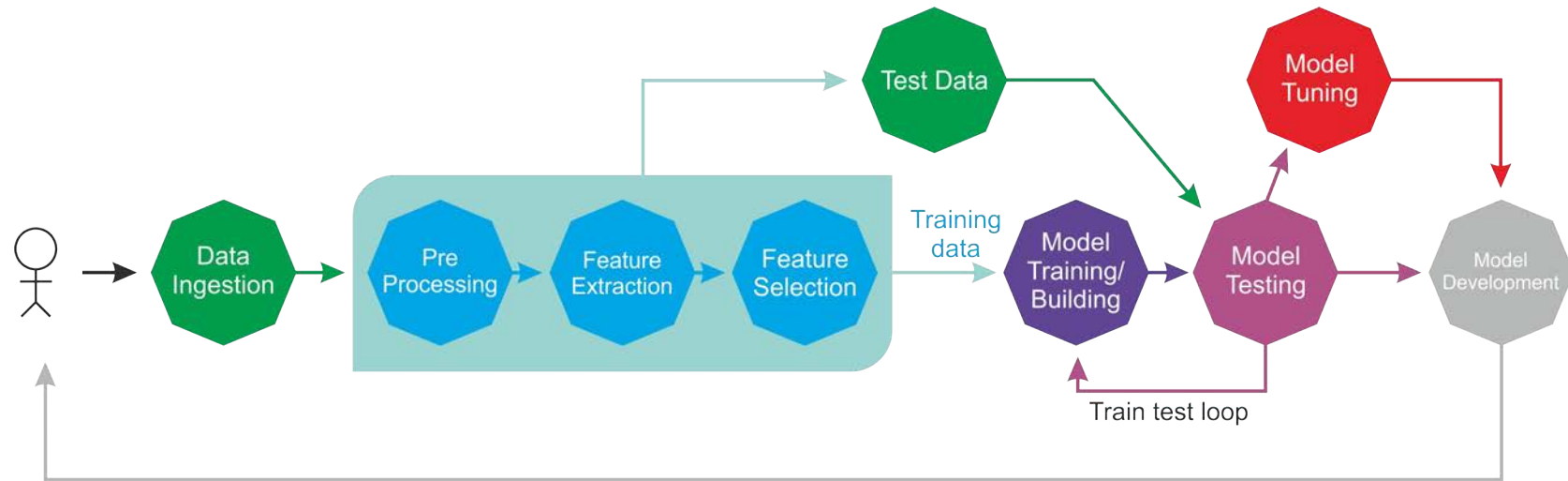
# Now Predict the Image !!!

# Why Machine Learning is required

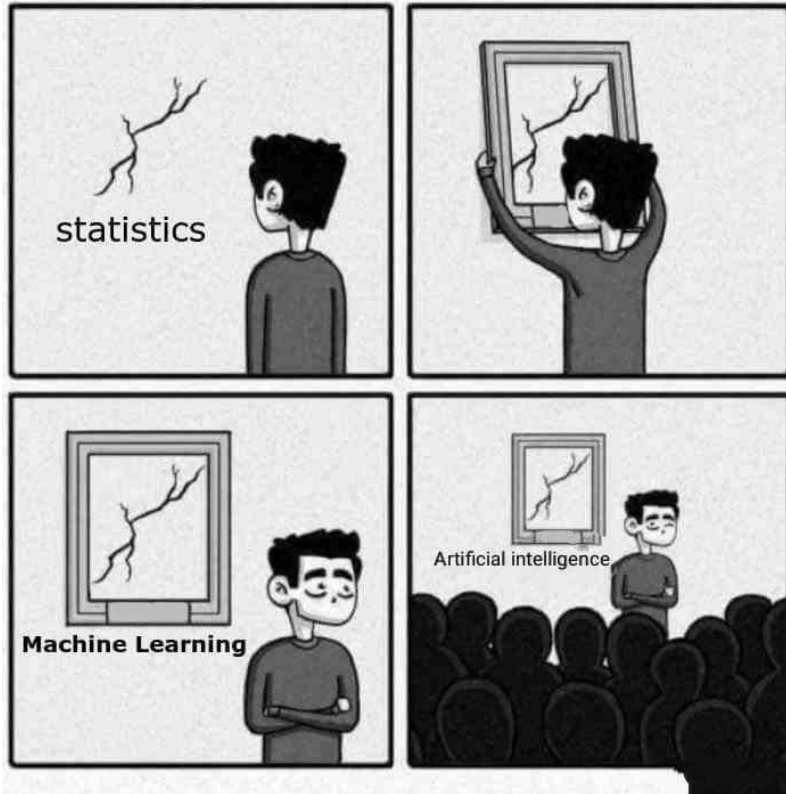
With the recommendation engine,  
They managed to increase 29%  
of annual sales.



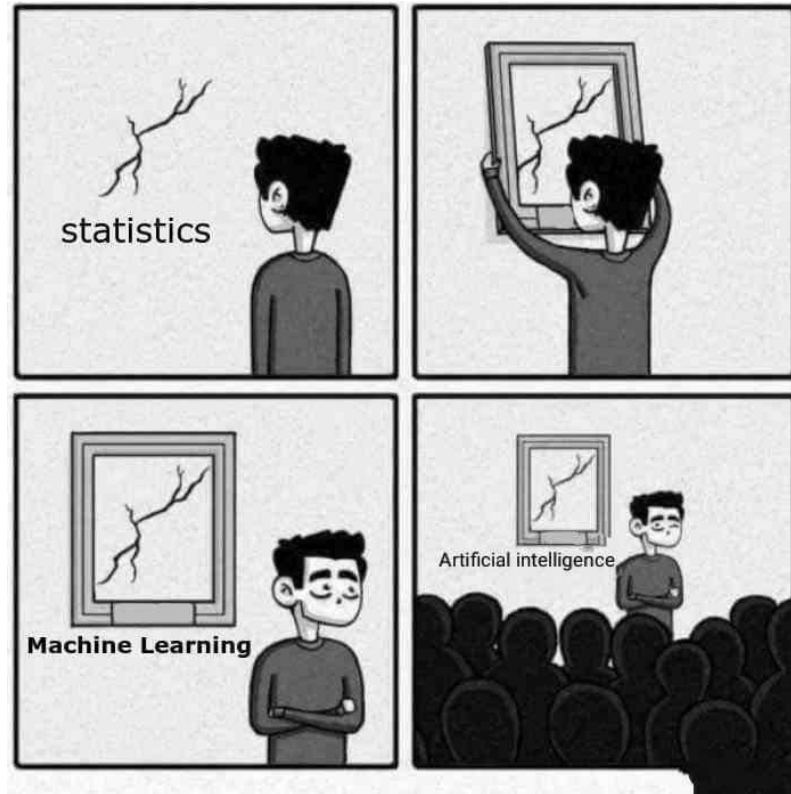
# Machine Learning Workflow



# Now What next ?



# Now What next ?





# What is DNS ?

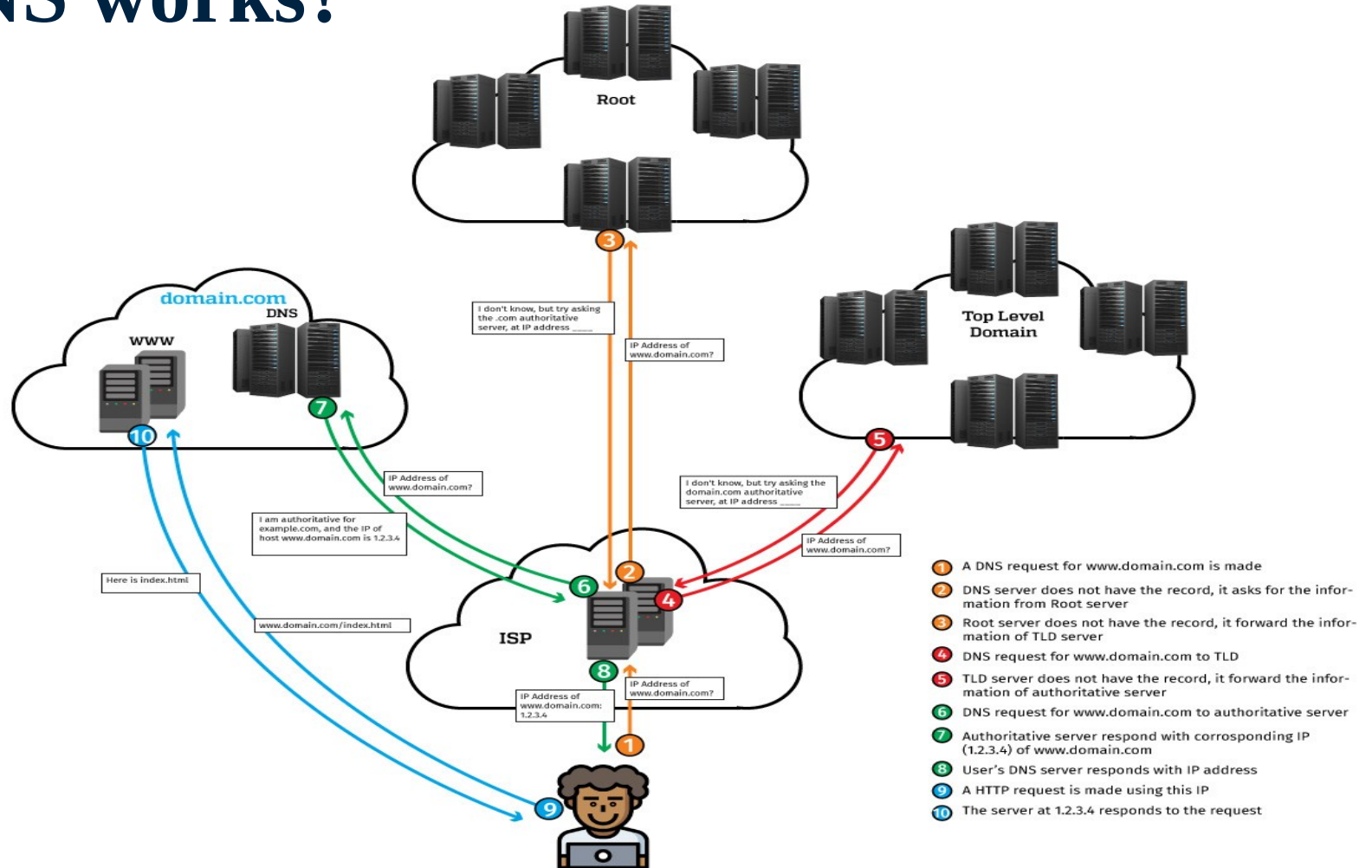
“The Domain Name Server (***DNS***) is the Achilles heel of the Web. The important thing is that it's managed responsibly.”

— Tim Berners-Lee

“ **92% of malware** uses **DNS** in one of three ways: to gain command and control, to ex-filtrate data or redirect traffic.”

Cisco Annual Security Report

# How DNS works?



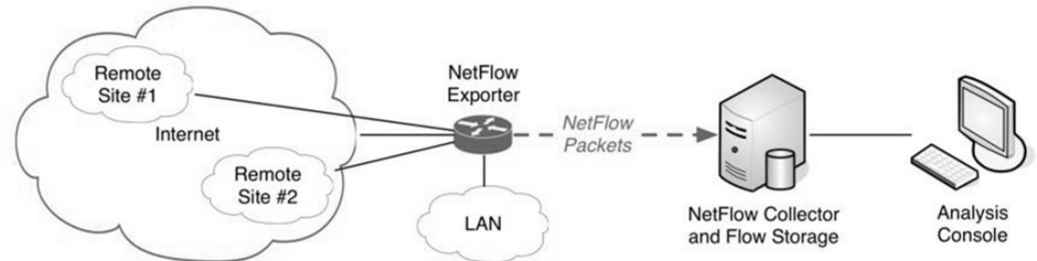
# NetFlow & Its Core Component

“It is network protocol developed by Cisco to collect and monitor network traffic flow data.”

- Wikipedia

## The Core Components of a NetFlow

- Exporter (Device with NetFlow collection enable)
- Collector (Where the NetFlow messages are sent)
- Storage
- Analysis Console



# Attack Definition in-terms of NetFlow

## Attacks that Target DNS Server

Recursive Query Attacks

Cache Poisoning Attacks

Buffer overflow

Port Scan

## Attacks using DNS Server

Reflection Attack

DNS Tunneling

- It is very hard, to give an conclusive answer for a particular attack, if the amount of captured packet is low.
- For **DoS**, the amount of traffic can be high, but if the DNS package is vulnerable, only few packets can perform a successful attack.
- Increase amount of query request to a DNS server can be counted as, **Cache-Poisoning**
- **Buffer overflow**, packet size have to be large enough and it has to be manipulated
- But, a small amount of packet info can show that **port scan** is running
- **DNS tunneling**, could be the same as DoS, due to increase of traffic, larger packets, and using of BASE64 & BASE32 character encoding, generate the queries which is strange.

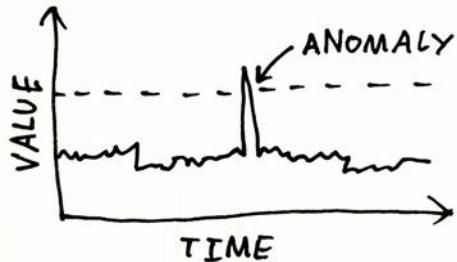
# The NetFlow Data Conversation

Date first seen	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Flags	Tos	Packets	Bytes	pps	bps	Bpp	Flows
2019-08-20 11:57:20.410	7.040	UDP	10.160.252.254:55427	->	192.168.0.254:53	.....	0	4	280	0	318	70	1
2019-08-20 11:57:20.430	7.040	UDP	10.160.252.254:55429	->	192.168.0.254:53	.....	0	4	280	0	318	70	1
2019-08-20 11:57:21.500	7.050	UDP	10.160.252.254:51601	->	192.168.0.254:53	.....	0	4	276	0	313	69	1
2019-08-20 11:57:21.500	7.050	UDP	10.160.252.254:56961	->	192.168.0.254:53	.....	0	4	276	0	313	69	1
2019-07-10 22:52:31.540	158.330	UDP	10.175.8.138:16664	->	192.168.0.254:53	.....	0	111	7509	0	379	67	1
2019-07-10 22:52:26.570	199.400	UDP	10.175.8.78:1031	->	192.168.0.254:53	.....	0	90	6123	0	245	68	1
2019-07-10 22:54:01.770	147.120	UDP	10.175.21.234:7623	->	192.168.0.254:53	.....	0	104	7526	0	409	72	1
2019-07-10 22:54:25.950	133.830	UDP	10.175.24.160:41530	->	192.168.0.254:53	.....	0	117	8018	0	479	68	1
2019-07-10 22:52:24.280	280.700	UDP	10.175.22.226:1029	->	192.168.0.254:53	.....	0	117	8836	0	251	75	1
2019-09-08 19:06:41.210	0.000	UDP	10.160.252.188:61902	->	192.168.0.254:53	.....	0	1	45	0	0	45	1
2019-09-08 21:08:20.570	0.000	UDP	10.160.252.212:40960	->	192.168.0.254:53	.....	0	1	45	0	0	45	1
2019-08-26 21:44:18.720	0.000	UDP	142.93.132.42:55433	->	192.16.204.217:53	.....	72	1	28	0	0	28	1
2019-08-26 21:47:33.480	0.000	TCP	88.6.232.5:42671	->	192.16.204.219:53	....S.	40	1	40	0	0	40	1
2019-08-26 22:18:58.290	0.000	TCP	88.6.232.5:52561	->	192.16.204.213:53	....S.	40	1	40	0	0	40	1
2019-09-12 19:54:04.130	52.830	TCP	10.160.68.26:44628	->	192.168.0.254:53	.AP...	0	319	349200	6	52879	1094	1
2019-09-12 20:04:27.090	1426.890	TCP	10.160.68.26:44628	->	192.168.0.254:53	.AP...	0	245007	349.3 M	171	2.0 M	1425	1
2019-08-20 12:07:59.650	0.000	UDP	10.160.252.254:62415	->	192.168.0.254:53	.....	0	2	202	0	0	101	1
2019-08-20 12:07:59.650	0.000	UDP	10.160.252.254:64165	->	192.168.0.254:53	.....	0	2	202	0	0	101	1
2019-08-20 12:07:59.660	0.000	UDP	10.160.252.254:63760	->	192.168.0.254:53	.....	0	2	116	0	0	58	1
2019-08-20 12:07:59.660	0.000	UDP	10.160.252.254:49521	->	192.168.0.254:53	.....	0	2	198	0	0	99	1
2019-08-20 12:07:59.660	0.000	UDP	10.160.252.254:61534	->	192.168.0.254:53	.....	0	2	198	0	0	99	1
2019-08-20 12:07:59.670	0.000	UDP	10.160.252.254:50694	->	192.168.0.254:53	.....	0	2	192	0	0	96	1
2019-08-20 12:07:59.670	0.000	UDP	10.160.252.254:50557	->	192.168.0.254:53	.....	0	2	192	0	0	96	1
2019-09-01 00:38:35.710	5.000	UDP	10.160.252.205:21343	->	192.168.0.254:53	.....	0	4	1256	0	2009	314	1
2019-09-01 00:38:40.660	5.010	UDP	10.160.252.205:2500	->	192.168.0.254:53	.....	0	4	976	0	1558	244	1
2019-09-01 00:38:45.710	7.960	UDP	10.160.252.205:29718	->	192.168.0.254:53	.....	0	4	1256	0	1262	314	1
2019-09-01 00:38:50.670	6.440	UDP	10.160.252.205:5733	->	192.168.0.254:53	.....	0	4	960	0	1192	240	1
2019-10-19 00:09:22.800	0.000	UDP	10.111.186.85:42531	->	192.168.0.254:53	.....	192	1	160	0	0	160	1
2019-10-19 00:09:22.800	0.000	UDP	10.111.186.85:49651	->	192.168.0.254:53	.....	192	1	160	0	0	160	1
2019-10-19 00:09:22.800	0.000	UDP	10.111.186.85:35340	->	192.168.0.254:53	.....	192	1	160	0	0	160	1

# What is an Anomaly?

“Anomaly detection, is the process of finding data objects with behaviors that are very different from expectation. Such objects are called **anomalies**.”

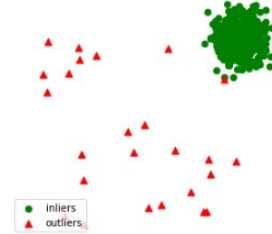
– *Data Mining. Concepts and Techniques* by **Jiawei Han**



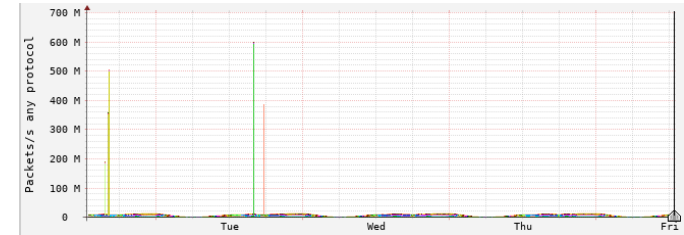
**Noise and Anomalies are not the same !**

# Classification of Anomaly in Netflow

**Point Anomaly** – A data point that differ from the rest of the data set



**Contextual Anomaly** – If an observation is strange because of the context of the observation.



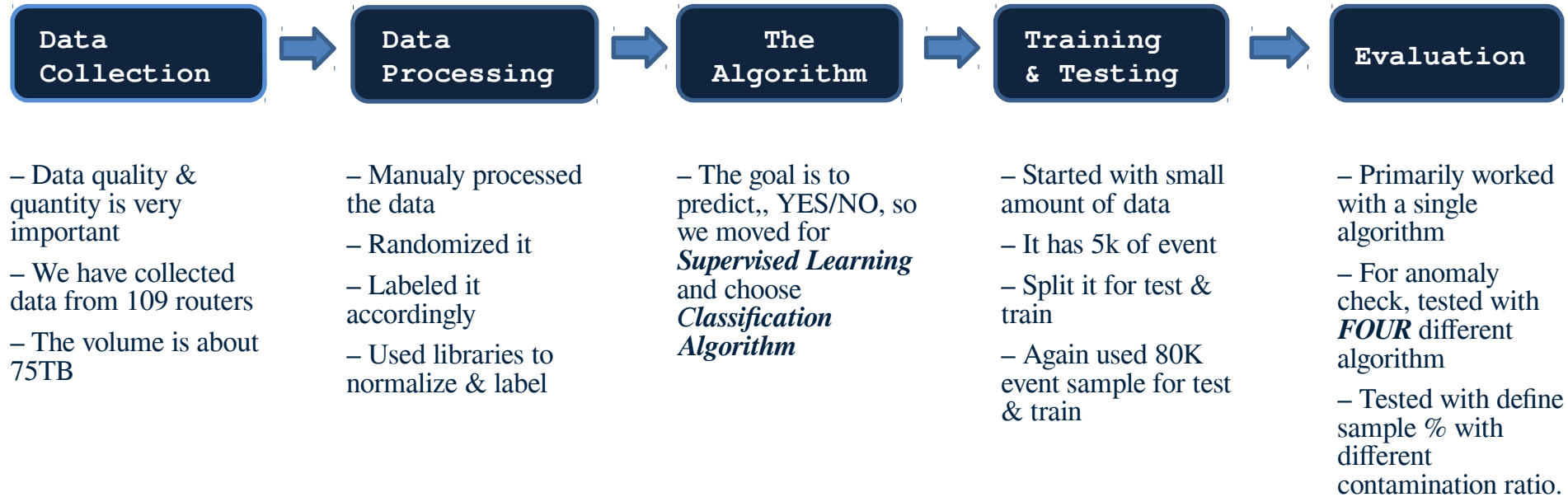
**Collective Anomaly** – A collection of data instances are strange in observation.



**Let us build a  
Classification Model**



# The Process for AI – Machine Learning Pipelines



*(though we have worked only on few selected events)*

# The Process for AI – standard procedure, “*strategies to collect data*”

## How is the data ?

- The data has multiple features
- Too much noise is there

## So how we start working on the DATA ?

- Separated Open DNS Resolver to reduce data noise
- Collected the data that are directed to Anycast DNS server
- Collected the data that are directed to DNS server outside of link3 cloud
- Started with “GO with the Book” formula

# The Process for AI – raw data of attacks, sample

## Attacks that Target DNS Server

### Port Scan

## Attacks using DNS Server

### DNS Tunneling

Date first seen	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Flags	Tos	Packets	Bytes	pps	bps	Bpp	Flows
2019-09-08 19:06:41.210	0.000	UDP	10.160.252.188:61902	->	192.168.0.254:53	.....	0	1	45	0	0	45	1
2019-09-08 21:08:20.570	0.000	UDP	10.160.252.212:40960	->	192.168.0.254:53	.....	0	1	45	0	0	45	1
2019-08-26 21:45:11.320	0.000	UDP	142.93.132.42:55433	->	192.16.204.217:53	.....	72	1	40	0	0	40	1
2019-08-26 21:47:33.480	0.000	TCP	88.6.232.5:42671	->	192.16.204.219:53	....S.	40	1	40	0	0	40	1
2019-08-26 22:18:58.290	0.000	TCP	88.6.232.5:52561	->	192.16.204.213:53	....S.	40	1	40	0	0	40	1
2019-12-03 05:56:07.160	23.780	UDP	170.0.103.254:22	->	192.16.222.196:53	.....	72	60	4140	2	1392	69	1
2019-12-03 05:56:47.470	26.500	UDP	170.0.103.254:22	->	192.16.222.196:53	.....	72	60	4140	2	1249	69	1
2019-12-03 05:57:44.650	19.260	UDP	170.0.103.254:22	->	192.16.222.196:53	.....	72	73	5037	3	2092	69	1
2019-12-03 05:58:35.970	17.010	UDP	170.0.103.254:22	->	192.16.222.196:53	.....	72	36	2484	2	1168	69	1
2019-12-03 05:59:53.340	1.570	UDP	170.0.103.254:22	->	192.16.222.196:53	.....	72	30	2070	19	10547	69	1
2019-12-03 06:00:43.320	2.640	UDP	170.0.103.254:22	->	192.16.222.196:53	.....	72	30	2070	11	6272	69	1

# The Process for AI – standard procedure. *“formatted data”*

```
0.04,2,140,0,70,70,1,nrmlDNSFlow
0.04,2,140,0,70,70,1,nrmlDNSFlow
2.04,3,180,0,128,80,1,nrmlDNSFlow
7.04,4,280,0,318,70,1,nrmlDNSFlow
7.04,4,280,0,318,70,1,nrmlDNSFlow
7.05,4,276,0,313,69,1,nrmlDNSFlow
7.05,4,276,0,313,69,1,nrmlDNSFlow
7.04,4,280,0,318,70,1,nrmlDNSFlow
7.04,4,280,0,318,70,1,nrmlDNSFlow
7.05,4,304,0,344,76,1,nrmlDNSFlow
7.05,4,304,0,344,76,1,nrmlDNSFlow
99.86,73,4560,0,365,62,1,dnsflood
214.29,98,6475,0,241,66,1,dnsflood
63.67,129,8750,2,1099,67,1,dnsflood
222,18661,1.4M,84,50696,75,1,dnsflood
959.21,66129,5.8M,68,48762,88,1,dnsflood
437.5,84479,71.1M,193,1.3M,841,1,dnsflood
556.03,81319,81.7M,146,1.2M,1004,1,dnsflood
1712.01,99985,73.8M,58,344723,737,1,dnsflood
1799.38,250593,258.3M,139,1.1M,1030,1,dnsflood
1899.38,280502,358.3M,140,1.1M,1080,1,dnsflood
1899.38,290711,398.3M,160,1.1M,2010,1,dnsflood
481.86,90507,93.0M,187,1.5M,1027,1,dnsflood
226.31,173,10892,0,385,62,1,dnsflood
```

- Data cleansing is done manually
- Out of 15 features, we have selected only 7 of them, to minimize the dominance on ML model
- String based labeling has been done to confirm the output

# The Process for AI – classification algorithm *kNN*

**What is kNN?** K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions).

`"If you are similar to your neighbors then you are one of them"`

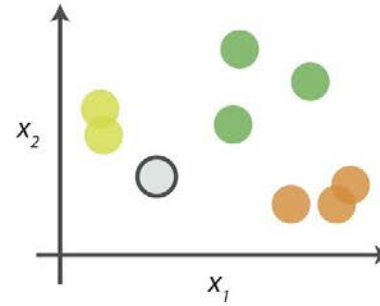
**Significance of K in KNN?** How many points will be considered in the classification, depends on the value of K parameter.

If  $k=5$ , then 5 nearest points will be considered to calculate the result.

# The Process for AI

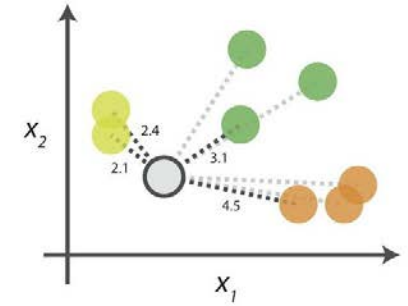
How it works – *kNN*

## 0. Look at the data











Say you want to classify the grey point into a class. Here, there are three potential classes - lime green, green and orange.

## 1. Calculate distances









Start by calculating the distances between the grey point and all other points.

## 2. Find neighbours

Point Distance			
	...		2.1 → 1st NN
	...		2.4 → 2nd NN
	...		3.1 → 3rd NN
	...		4.5 → 4th NN

Next, find the nearest neighbours by ranking points by increasing distance. The nearest neighbours (NNs) of the grey point are the ones closest in dataspace.

## 3. Vote on labels

Class	# of votes	
	2	➔ Class  wins the vote! Point  is therefore predicted to be of class  .
	1	
	1	

Vote on the predicted class labels based on the classes of the  $k$  nearest neighbours. Here, the labels were predicted based on the  $k=3$  nearest neighbours.

# The Process for AI – distance metrics of *kNN*

## Euclidean Distance –

Euclidean Distance is the least possible distance between two points or straight-line distance between two points.

## Manhattan Distance –

Manhattan Distance is the distance between two points measured along the axis at right angles, So it may not be the least distance between the points.





# The Process for AI – distance metrics of *kNN*

## Which one to work with ?

- if the data has a large number of features then the Manhattan Distance will be a better fit.
- if the data has less number of features then the Euclidean Distance will be a better fit.

## Best way to decide the value of K?

- The best way to do so is to run through all the different values of K and select the value for which the error value is the least.



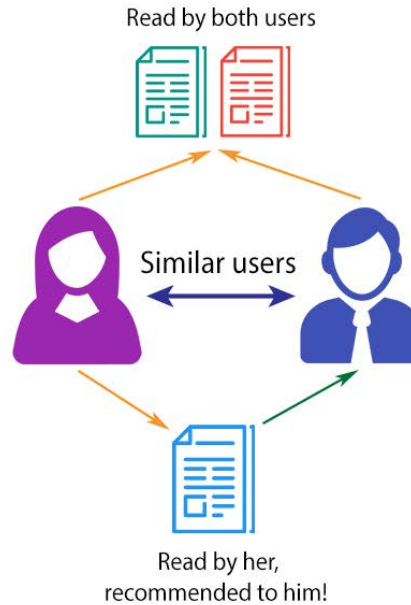
# The Process for AI – use cases of $kNN$

## Recommendation System

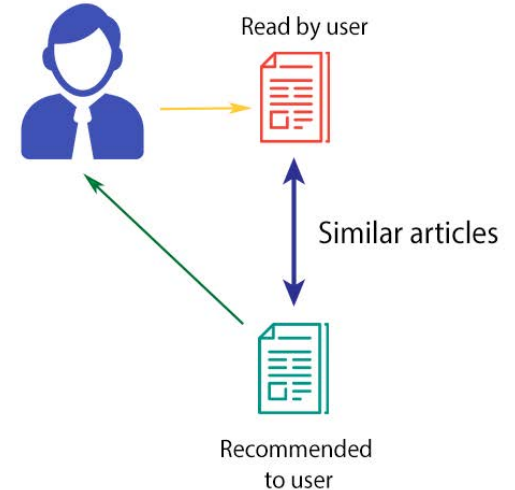
KNN algorithm is used in recommendation systems. Like, first users have to be classified on the basis of their searching behavior and if any user searches for something then we can recommend a similar type of item to all the other users of the same class.

- Movie recommendation in **NetFlix**
- Books/Items recommendation in **Amazon**

### COLLABORATIVE FILTERING



### CONTENT-BASED FILTERING



# The Process for AI – use cases of *kNN*

- Threat Detection
- Predicting Attacks
- Detecting Anomalies

Anomalies are strange observation; so for an observation, its distance to its  $k$ th nearest neighbor could be viewed as the outlying score. It could be viewed as a way to measure the density.

# The Process for AI

Phase One – Working with the Machine Learning Model

*Finding the Known Attacks*

# The Process for AI – import required libraries, *KNN*

```
import pandas as pd
import numpy as np

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier

from math import sqrt
from sklearn import neighbors
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt

from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

**scikit-learn** is an open source simple and efficient tool for data mining and data analysis.

- pandas is used to import the dataset from a file into data frame.
- numpy is used to work on array.
- train\_test\_split is used to split the model into testing and training data
- standard scaler to normalize the data
- matplotlib is used to visualize the dataset
- math is used to import the square root function which we are going to use in finding the Euclidean Distance
- mean\_squared\_error is used to find the error between the actual output and the predicted output
- neighbors is used to implement the knn regression model, from which we are deciding the best fit value of k

# The Process for AI – load the dataset, *KNN*

```
ColNames = ['duration', 'packets', 'bytes', 'pps', 'bps', 'bpp', 'flows', 'class']  
FilePath = "/home/shamim/Documents/International Conference/2020 Apricot/tutorial/Data/DNSPortScanTunnelAtckCheckFlow.csv"
```

```
rows = pd.read_csv(FilePath, names=ColNames)  
netflow_label = rows
```

```
print (len(rows))  
rows.head(10)
```

2176

	duration	packets	bytes	pps	bps	bpp	flows	class
0	0.00	1	68	0	0	68	1	nrmlDNSflow
1	0.00	1	64	0	0	64	1	nrmlDNSflow
2	214.29	98	6475	0	241	66	1	dnsampattack
3	226.31	173	10892	0	385	62	1	dnsampattack
4	4.30	25	1725	5	3209	69	1	dnstunneling
5	1.29	13	897	10	5562	69	1	dnstunneling
6	0.00	1	40	0	0	40	1	portscanatck
7	0.00	1	40	0	0	40	1	portscanatck
8	0.00	1	57	0	0	57	1	nrmlDNSflow
9	0.00	1	77	0	0	77	1	nrmlDNSflow

Using *panda* library to load the CSV file and running the *head* command to see the content of the file.

# The Process for AI – assigning features and encode the label, *KNN*

```
X = rows.iloc[:, 0:7]
y = rows.iloc[:, 7]
```

```
le = LabelEncoder()
y = le.fit_transform(y)
```

```
list(y)
```

```
[2,
 2,
 0,
 0,
 1,
 1,
 3,
 3,
 2,
 2,
 1,
 1,
 3,
 0,
 2,
 2,
 2,
 1,
 1,
 2,
 3,
 2,
 1,
.]
```

**LabelEncoder** is used to normalize labels.

- It can also be used to transform non-numerical labels to numerical labels, which means it encode target labels with value between 0 and `n_classes-1`.
- Transform labels back to original encoding. Transform labels to normalized encoding.

It is not possible to implement regression on strings data, so the above command will encode the data on which we can perform the regression operation.

# The Process for AI – splitting and feature scaling, *KNN*

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0, test_size=0.2)

scaler = StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

The above script splits the dataset into 80% train data and 20% test data.

So now we are going to train the model from the training data and after that we will test it on the testing data.

**“any algorithm that computes distance or assumes normality, scale your features”**

StandardScaler() will normalize the features (each column of X, INDIVIDUALLY !!!) so that each column/feature/variable will have mean = 0 and standard deviation = 1.

Basically it performs the task of Standardization. Usually a dataset contains variables that are different in scale. For e.g. our dataset contains DURATION column with values on scale 0-600.00, Flows column with values on only 1, PPS column has the values with scale 0-400,etc.

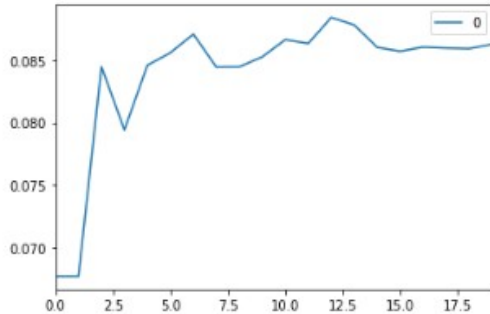
As these columns are different in scale, they are Standardized to have common scale while building machine learning model.

# The Process for AI – finding the value of K, *KNN*

```
rmse_value = []  
for K in range(20):  
    K = K+1  
    model = neighbors.KNeighborsRegressor(n_neighbors = K)  
    model.fit(X_train, y_train)  
    pred=model.predict(X_test)  
    error = sqrt(mean_squared_error(y_test,pred))  
    rmse_value.append(error)
```

```
curve = pd.DataFrame(rmse_value)  
curve.plot()
```

<matplotlib.axes.\_subplots.AxesSubplot at 0x7ff6f0f67128>



This script executes a loop from 1 to 20. And in every iteration, the mean error for predicted values of the test set is calculated and the result is appended to the error list as *rmse\_value*.

The plot the error values against K values to find the best value of k.



# The Process for AI – implementing the classifier, *KNN*

```
: classifier = KNeighborsClassifier(n_neighbors=3, p=2, metric='euclidean')  
classifier.fit(X_train, y_train)  
y_pred = classifier.predict(X_test)
```

- In first line, the *classifier class* is initialized with one parameter as the value of k, that one found in the in last steps.
- In second line, we have trained our model on the training data, that is 80% of the total dataset we split earlier.
- The final step is to make predictions on the dataset using testing data, that is 20% of the total dataset.

# The Process for AI – evaluate the algorithm, *KNN*

```
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
print(accuracy_score(y_test, y_pred.round()))
```

```
[[ 13  2  0  0]
 [  0 25  0  0]
 [  0  0 237 1]
 [  0  0  0 158]]
```

	precision	recall	f1-score	support
0	1.00	0.87	0.93	15
1	0.93	1.00	0.96	25
2	1.00	1.00	1.00	238
3	0.99	1.00	1.00	158
accuracy			0.99	436
macro avg	0.98	0.97	0.97	436
weighted avg	0.99	0.99	0.99	436

0.9931192660550459

A confusion matrix is a performance measurement technique for Machine learning classification.

Above output shows the evaluation metrics confusion matrix, classification matrix, and accuracy score respectively.

**Accuracy of KNN Algorithm is 0.99311**

**You have done nothing,  
but your test accuracy goes from 99.012% to 99.31%**



**One should look for a possible alternative, and  
provide against it.  
It is the first rule of criminal investigation.**

Sherlock Holmes, the adventure of black peter

# The Process for AI

Phase Two – Working with the Machine Learning Model

*Working with Anomaly Detection*

# The Process for AI – “*working with PyOD*”

“**PyOD** (*Python Outlier Detection*) is a comprehensive and scalable Python toolkit for detecting outlying objects in multivariate data.”

– Zhao, Yue and Nasrullah, Zain and Li, Zheng

\*\* *For Anomaly Detection, data was not manually labeled, rather the raw data has been used.*

# The Process for AI – “*working with PyOD, kNN*”

```
import numpy as np
import pandas as pd
from sklearn import preprocessing
```

```
from pyod.models.knn import KNN
from pyod.utils.data import generate_data
from pyod.utils.data import evaluate_print
from pyod.utils.example import visualize
```

```
filePath = "/home/shamim/Documents/International Conference/2020 Apricot/tutorial/Data/AnomalyDetectionWithPyOD-FlowData.csv"
rows=pd.read_csv(filePath)
```

– Import the algorithm

# The Process for AI – “working with PyOD, kNN”

```
if __name__ == "__main__":
    contamination = 0.01 # percentage of outliers
    n_train = 10000 # number of training points
    n_test = 1000 # number of testing points

    # Generate sample data
    X_train, y_train, X_test, y_test = \
        generate_data(n_train=n_train,
                      n_test=n_test,
                      n_features=2,
                      contamination=contamination,
                      random_state=None)

    # train kNN detector
    clf_name = 'KNN'
    clf = KNN()
    clf.fit(X_train)

    # get the prediction labels and outlier scores of the training data
    y_train_pred = clf.labels_ # binary labels (0: inliers, 1: outliers)
    y_train_scores = clf.decision_scores_ # raw outlier scores

    # get the prediction on the test data
    y_test_pred = clf.predict(X_test) # outlier labels (0 or 1)
    y_test_scores = clf.decision_function(X_test) # outlier scores

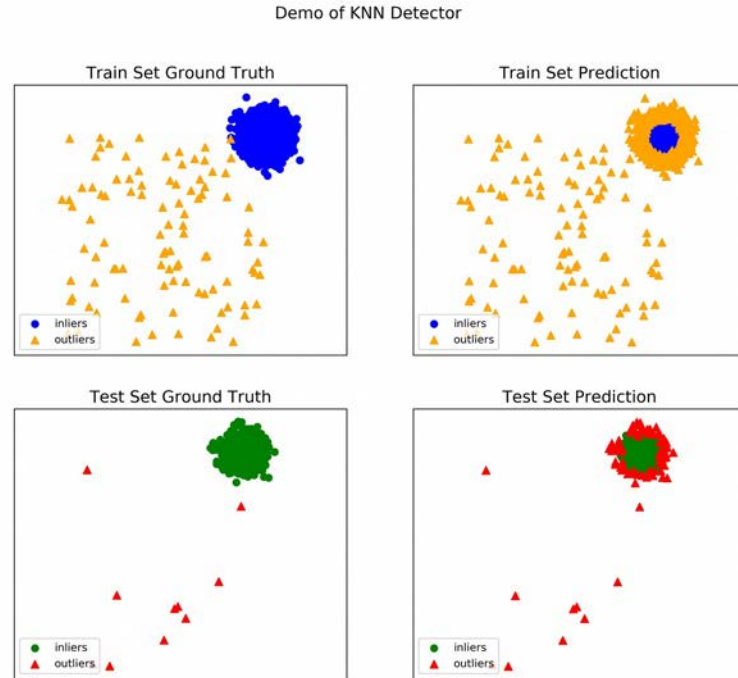
    # evaluate and print the results
    print("\nOn Training Data:")
    evaluate_print(clf_name, y_train, y_train_scores)
    print("\nOn Test Data:")
    evaluate_print(clf_name, y_test, y_test_scores)

    # visualize the results
    visualize(clf_name, X_train, y_train, X_test, y_test, y_train_pred,
             y_test_pred, show_figure=True, save_figure=True)
```

- Import the algorithm
- Set the test and train data ratio
- Set the contamination ratio
- Call the prediction



# The Process for AI – “*working with PyOD, kNN*”



- Show the train and test score
- Visualize the output

On Training Data:

KNN ROC:1.0, precision @ rank n:0.98

On Test Data:

KNN ROC:1.0, precision @ rank n:1.0

# The Process for AI – “*working with PyOD, kNN*”

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

```
print(confusion_matrix(y_test, y_test_pred))
print(classification_report(y_test, y_test_pred))
print(accuracy_score(y_test, y_test_pred.round()))
```

```
[[915  75]
 [  0  10]]

```

	precision	recall	f1-score	support
0.0	1.00	0.92	0.96	990
1.0	0.12	1.00	0.21	10
accuracy			0.93	1000
macro avg	0.56	0.96	0.59	1000
weighted avg	0.99	0.93	0.95	1000

```
0.925
```

So, now let us evaluate the model.

And this is necessary.

# The Process for AI – “What are the lessons?”

- The difference between noise and anomalies
  - Noise often outnumber anomalies
- How to make use of label information/domain knowledge when available
- Fast runtime: can scale up to large datasets and high dimensional datasets
- Known behaviors under different data properties
- Can deal with different types of anomalies
- Its ability to deal with high dimensional problems

# Reference

- Hoai-Vu Nguyen and Yongsun Choi, “Proactive Detection of DDoS Attacks Utilizing k-NN Classifier in an Anti-DDos Framework ” (World Academy of Science, Engineering and Technology, International Journal of Computer and Information Engineering, Vol:4, No:3, 2010 )
- Przemysław Berezinski, Bartosz Jasiul, and Marcin Szpyrka; “ An Entropy-Based Network Anomaly Detection Method”, Entropy 2015, 17, 2367-2408 (<https://www.mdpi.com/journal/entropy>)
- Das, S., Islam, M.R., Jayakodi, N.K. and Doppa, J.R., “Active Anomaly Detection via Ensembles: Insights, Algorithms, and Interpretability.” arXiv preprint arXiv:1901.08930. 2019.
- Ro, K., Zou, C., Wang, Z. and Yin, G., “Outlier detection for high-dimensional data. Biometrika” 2015.
- Suri, N.R. and Athithan, G., “Research Issues in Outlier Detection. In Outlier Detection: Techniques and Applications”, 2019.
- Milan Cermak, Pavel Celeda, Jan Vykopal, “Detection of DNS Traffic Anomalies in Large Networks”, Masaryk University, 2014 ( <https://www.researchgate.net/publication/290882059>)
- Zdrnja, B., Brownlee, N., Wessels, D., “Passive Monitoring of DNS Anomalies. In:Detection of Intrusions and Malware, and Vulnerability Assessment”, 2007.
- Manasrah, A.M., Hasan, A., Abouabdalla, O.A., Ramadass, S., “Detecting Bot-net Activities Based on Abnormal DNS Traffic”, (IJCSIS) International Journal of Computer Science & Information Security, 2009.
- Zhao, Y., Nasrullah, Z. and Li, Z., 2019. PyOD: A Python Toolbox for Scalable Outlier Detection. Journal of machine learning research (JMLR), 20(96), pp.1-7.
- Takashi Washio, Osaka University; IEEE International Conference on Data Mining, Singapore "Which Anomaly Detector should I use?", 2018
- Hodge, V.J. and Austin, J. (2004) A survey of outlier detection methodologies. Artificial Intelligence Review, 22 (2). pp. 85-126.

# Thank You

for your attention