BANQUE DE FRANCE
EUROSYSTÈME

ANALYSE STATISTIQUE DE FILTRAGES RÉSEAUX ET APPRENTISSAGE AUTOMATIQUE

- JSSI 2024 -

DGSI

DIRECTION GÉNÉRALE DU SYSTÈME D'INFORMATION

FRANÇOIS VALLEY
francois.valley@banque-france.fr
INGÉNIEUR CYBERSÉCURITÉ

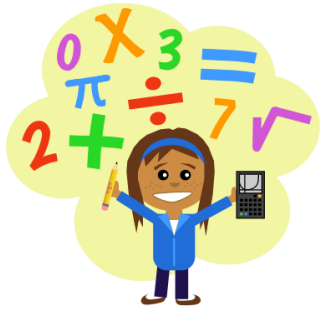
12/03/2024

CONTEXTE

- REX sur une expérimentation interne Banque de France
- Plus de détails techniques dans l'article « Analyse statistique de filtrages réseaux et apprentissage automatique » paru dans MISC – Cybersécurité offensive & défensive n°131 (janvier/février 2024)

DISCLAIMERS

- « aucun algorithme à paillettes »
- XP menée par un DataScientist « amateur »
- La démarche expérimentale est aussi (plus ?) importante que le résultat
- Il y aura des notions mathématiques (un peu)



OBJECTIF

- Analyse de sécurité « hors audit formel » des filtrages appliqués par les DevOps projets sur 2 périmètres IAAS internes
  - Périmètre 1 : ~2000 serveurs, exposés sur le réseau interne Banque
  - Périmètre 2 : ~150 serveurs, exposés sur Internet (presque)
- Analyse effectuée en 2022, les infrastructures sont maintenant dé-commissionnées.

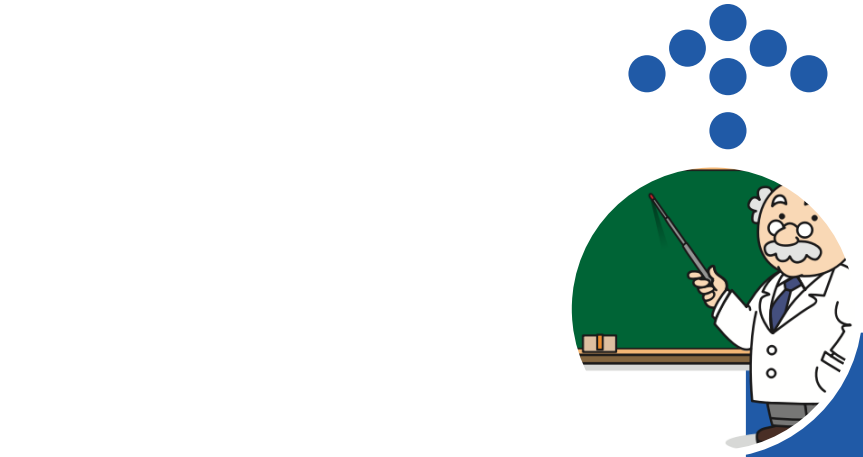
LES MÉTHODES HABITUELLES

- Prendre un expert réseau et lui faire analyser environ 100k lignes de filtrage
  - Pbm 1 : Trouver le « volontaire motivé »
  - Pbm 2 : Éviter les erreurs sur les 100k filtrages
  - Pbm 3 : Garder le « volontaire » motivé pour le prochain audit
- Automatiser au travers de règles « expertes »
  - Par ex : pas de flux SSH depuis Internet
  - Pbm 1 : Être restrictif et obtenir de nombreuses exceptions
  - Pbm 2 : Être « large » dans nos règles et ne détecter pas grand chose

L'IDÉE

- Utiliser les outils « modernes » d'analyse statistique (ML, IA & co) pour :
  - Avoir une vue globale de la sécurité des filtrages en place
    - Au sens : quel est le risque de sécurité pris globalement sur les serveurs d'une telle infrastructure au niveau de la surface d'attaque réseau ?
  - Concevoir un outil de détection de filtrages réseaux à risques non acceptables.

LA DÉMARCHE



Apprentissage supervisé

Clustering / Visualisation

Construction du jeux de données

CONSTRUCTION DU JEUX DE DONNÉES (CDJDD)

- Quelques principes
  - Analyse statistique = représentation sous forme de matrice, donc tout doit être « nombre »
    - Allow tcp 22 to 10.0.0.0/27 ⇔[1.4, -3.9, … , 5.9]
  - Et idéalement, des mesures qui reflètent des caractéristiques en lien avec notre objectif, donc discriminantes pour la sécurité

CDJDD : EXTRACTION DES RÈGLES DE FILTRAGE 1/2

- Utilisation des API OpenStack pour extraire :
  - Security group (filtrages appliqués au niveau VM)
  - Firewall group (filtrages appliqués au niveau du réseau privé du tenant/projet)
- Compilation de ces extractions pour obtenir un tableau recensant l'ensemble des flux ouverts :

Serveur	Réseau distant	Port destination	Direction
Serveur A	192.168.1.20/32	22	ingress
Serveur A	0.0.0.0/0	443	ingress
Serveur B	192.168.1.0/24	443	egress

CDJDD : EXTRACTION DES RÈGLES DE FILTRAGE 2/2

- Remarque 1 : c'est pas si simple
- Remarque 2 : l'imbrication entre security group et firewall group rend l'extraction des flux effectifs ouverts pas simple à calculer
- Remarque 3 : de nombreuses interprétations à faire (-1 en n° de port, protocole à null, etc.)
- Remarque 4 : c'est long

CDJDD : EXTRACTION DES RÈGLES DE FILTRAGE 2/2

- [illegible]i
- [illegible]a [illegible]roup et firewall group rend
- [illegible]ec [illegible]e à [illegible]
- [illegible]lk [illegible]s à [illegible]
- [illegible]

```
#Function that generate rules_df
def gen_rules_df(data_path):

    #load servers dataframe
    servers_list = list()
    for f_name in os.listdir(data_path):
        if f_name.startswith("storm_server__"):
            storm_env, project_name, secgroup_id = re.search(r'storm_server__([^_]+)_.([^.]+)-.([^.]+).json',f_name).groups()
            json_s = ""
            with open(os.path.join(data_path,f_name)) as f:
                json_s = f.read()
            if json_s != "":
                d = json.loads(json_s)
                d["storm_env"] = storm_env
                d["project_name"] = project_name
                if ("addresses" in d.keys()) and ("=" in d["addresses"]):
                    addr = d["addresses"].split("=")[1].split(", ")
                    if len(addr) > 1:
                        d["ext_ip"] = addr[1]
                if "security_groups" in d.keys():
                    sec_groups = d["security_groups"].split("\n")
                    del d["security_groups"]
                    for sec_group in sec_groups:
                        d["security_group"] = sec_group.split("=")[1].strip("'")
                        servers_list.append(d.copy())
    servers_df = pd.DataFrame.from_dict(servers_list)
    del servers_list

#   #load project table
#   projects_df = pd.DataFrame()
#   with open(os.path.join(DATA_PATH,"atom_projects.json")) as f:
#       json_s = f.read()
#   if json_s != "":
#       a = json.loads(json_s)
#       for d in a:
#           for c in d.keys():
#               if not c in projects_df.columns.to_list():
#                   projects_df[c] = np.nan
#           projects_df.loc[len(projects_df)] = d


    #load security group dataframe
    sec_groups_list = list()
    for f_name in os.listdir(data_path):
        if f_name.startswith("storm_security_group__"):
            storm_env, project_name, secgroup_id = re.search(r'storm_security_group__([^_]+)_.([^.]+)-.([^.]+).json',f_name).groups()
            json_s = ""
            with open(os.path.join(data_path,f_name)) as f:
                json_s = f.read()
            if json_s != "":
                d = json.loads(json_s)
```

```
#Tools to categorize network stream type by port
PORT_TYPE = {
        "admin" : [ 22, 3389, ],
        "vpn": [ 1080, 22, 1337,  ],
        "system": [ 111, 123, 2377, 25, 4789, 53, 7946, 9100, ],
        "fsbdd": [ 20048, 2049, 3306, 5432, ],
        "http": [ 443, 80, 8080, 8081, 8082, 8083, 8084, 8085, 8086, 8087, 8088, 8089, 8090, ]
        }

def get_port_type(port_min, port_max):
    out = set()
    for port in range(port_min,port_max+1):
        for t, r in PORT_TYPE.items():
            if port in r:
                out.add(t)
            else:
                out.add("other")
    return out


#Tools to load IP -> AFS mapping table
def load_subnet_mapping(gain_file,sheet_name="Extraction_GAIN_20210218"):
    #load subnet mapping
    gain = pd.read_excel(gain_file, sheet_name, engine='openpyxl')
    #gain.groupby("Zone AFS").size()

    network_list = list()
    for index, row in gain.iterrows():
        r = dict()
        r["net"] = ipaddress.IPv4Network(row["Start address"]+"/"+str(row['Subnet prefix']))
        r["name"] = row["Subnet name"]
        r["afs"] = row["Zone AFS"]
        network_list.append(r)
    return network_list

def get_afs(ip_net):
    out = dict()
    #test if internal tenant network
    if ipaddress.IPv4Network("192.168.0.0/16").overlaps(ip_net):
        out["TENANT"] = ip_net.num_addresses
    for net in NETWORK_LIST:
        if net["afs"] in ["RESEAU",]:
            continue
        if net["net"].overlaps(ipaddress.IPv4Network("192.168.0.0/16")):
```

```
############

############

utilisés
rt"] = sec_group_rules["Port Range"].apply(lambda x : x.split(":")[0])
es[~(sec_group_rules["port"].isnull())]
es.groupby("port")["port"].count()


(one-row == one rule)
s_df.merge(servers_df.add_prefix("server_"),left_on=["project_id","name"],right_on=["server_project_id
.drop_duplicates(subset=['server_id','port_range_max','port_range_min','protocol","remote_ip_prefix","r
#remove ipv6
rules_df = rules_df.loc[rules_df["ethertype"]!="IPv6"]
rules_df.reset_index(inplace=True)

#compute afs from rule
def get_afs_from_row(x):
    cols = x.index.to_list()
    if "remote_ip_prefix" in cols and type(x["remote_ip_prefix"]) != float:
        network = ipaddress.IPv4Network(x["remote_ip_prefix"],strict=False)
        afs = get_afs(network)
    elif "remote_group_id" in cols  and type(x["remote_group_id"]) != float:
        afs = { "TENANT": 1 }
    elif "remote_ip_prefix" in cols and "remote_group_id" in cols and type(x["remote_ip_prefix"]) == float and type(x["rem
        network = ipaddress.IPv4Network("0.0.0.0/0",strict=False)
        afs = get_afs(network)
    else:
        afs = dict()
    return afs
afs_compute_df = rules_df.drop_duplicates(subset=["remote_group_id","remote_ip_prefix"])[["remote_group_id","remote_ip_pre
afs_compute_df["afs"] = afs_compute_df.apply(get_afs_from_row,axis=1)
rules_df["afs"] = rules_df.merge(afs_compute_df,on=["remote_group_id","remote_ip_prefix"],how="left")["afs"]

#compute stream type
def get_port_type_from_row(x):
    if type(x["port_range_min"])!=float:
        if type(x["port_range_max"])!=float:
            return list(get_port_type(int(x["port_range_min"]),int(x["port_range_max"])))
        else:
            return list(get_port_type(int(x["port_range_min"]),int(x["port_range_min"])))
    else:
        if type(x["port_range_max"])==float:
            return list(get_port_type(1,65535))
        else:
            return list()
port_type_compute_df = rules_df.drop_duplicates(subset=["port_range_min","port_range_max"])[["port_range_min","port_range_
port_type_compute_df["port_types"] = port_type_compute_df.apply(get_port_type_from_row,axis=1)
rules_df["port_types"] = rules_df.merge(port_type_compute_df,left_on=["port_range_min","port_range_max"],right_on=["port_r
```

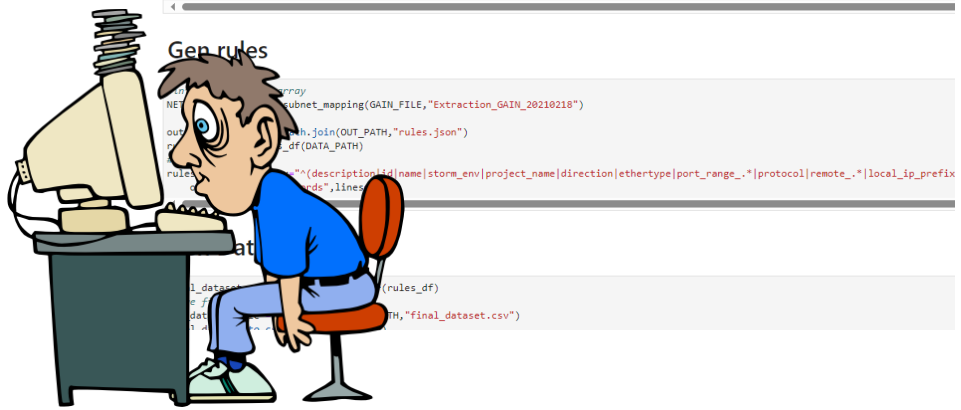
```
    for afs in list(set([ d["afs"] for d in NETWORK_LIST if type(d["afs"])!=float])):
        for t in PORT_TYPE.keys():
            for direction in ["ingress","egress"]:
                colname = "n_" + direction + "_" + afs + "_" + t
                if afs in afs_temp.columns.values:
                    rules_df[colname] = (rules_df["direction"] == direction) * (t_temp == t).any(axis=1) * afs_temp[afs]
    rules_df.drop(["port_types","afs"],axis=1,inplace=True)

    #create a human readable representation of a rule
    remote_df = rules_df["remote_ip_prefix"].astype(str)+rules_df["remote_group_id"].astype(str)
    remote_df.replace("nan","",regex=True,inplace=True)
    rules_df["secgroup_summary"] = rules_df["direction"].astype(str).apply(lambda x : x[0]) + "|" + \
                                   remote_df + "|" + \
                                   rules_df["port_range_min"].astype(str)+":"+rules_df["port_range_max"].astype(str)

    return rules_df

def gen_dataset_by_server(rules_df):
    #group by server_id
    final_dataset = rules_df.filter(regex=r"^(server_id|n_.+)").groupby("server_id").sum()
    temp_server_df = rules_df.drop_duplicates(subset=['server_id'], keep='first')[["server_id","server_name","storm_env","project_name",
    temp_server_df = temp_server_df.merge(rules_df.groupby("server_id").agg({ "secgroup_summary": lambda x : "\n".join(x.to_list())}),
                                          on="server_id")
    temp_server_df.drop(["secgroup_summary_x"],axis=1,inplace=True)
    final_dataset = final_dataset.merge(temp_server_df,on="server_id",how="left")

    return final_dataset
```



CDJDD : REPRÉSENTER LES IP

- IP -> un nombre en base 256 ? -> oui, mais quelle signification ?
- L'IP exacte en soit ne porte pas vraiment d'information, c'est plutôt son rattachement à un réseau/périmètre
  - IP -> zones réseaux de la PSR et du KM réseau ARSEC (ZAC-I, ZS, ZIC-E, etc.)
    - Oui, ce n'est pas un nombre, mais une catégorie, on y reviendra après…
  - On garde le nb d'IP lié à la règle

Serveur	Réseau distant	Port destination	Direction
Serveur A	192.168.1.20/32	22	ingress
Serveur A	0.0.0.0/0	443	ingress
Serveur B	192.168.1.0/24	443	egress

Serveur	Zone distante	Port destination	Direction	Nb_ip
Serveur A	DMZ	22	ingress	1
Serveur A	Internet	443	ingress	~4.10^9
Serveur A	DMZ	443	ingress	256
Serveur B	DMZ	443	egress	256

CDJDD : REPRÉSENTER LES PORTS

- Le n° de port TCP/UDP est un nombre, cool ! Mais…
  - TCP 22(SSH)/25(SMTP) proches, mais rien à voir dans leurs utilisations et l'implication sécurité sous jacente.
  - TCP 22(SSH)/3389(RDP) éloignés, mais souvent traités de pair (administration)
- Choix de représentation similaire aux IPs
  - On regroupe les services associés aux ports en catégories :
    - administration (SSH, RDP)
    - VPN (SOCKS, OpenVPN, etc.)
    - services systèmes (DNS, NTP, RPCBBIND, MICROSOFT-DS, etc.)
    - bases de données / stockage (SQL, NFS, etc.)
    - WEB (HTTP, HTTPS, sur ports standards ou alternatifs comme 8080, 8443, etc.)

Serveur	Zone distante	Port destination	Direction	Nb_ip
Serveur A	DMZ	22	ingress	1
Serveur A	Internet	443	ingress	~4.10^9
Serveur A	DMZ	443	ingress	256
Serveur B	DMZ	443	egress	256

Serveur	Zone distante	Service	Direction	Nb_ip
Serveur A	DMZ	ADMIN	ingress	1
Serveur A	Internet	WEB	ingress	~4.10^9
Serveur A	DMZ	WEB	ingress	256
Serveur B	DMZ	WEB	egress	256

CDJDD : QUELQUES TOURS DE TOURNEVIS EN PLUS 1/3

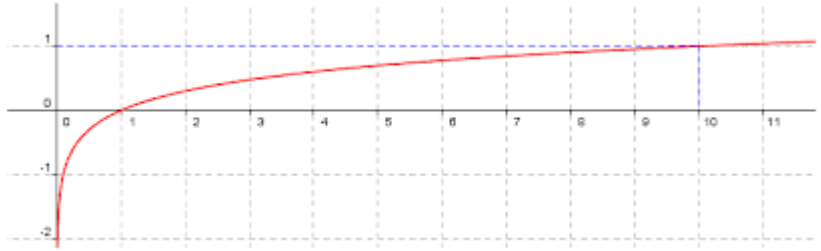
- Transformer les catégories en colonne
  - Petit nom : One-hot encoding
    - Chaque combinatoire (zone, service, direction) est transformé en une colonne
  - Attention, cela fait beaucoup de colonnes/dimensions vu les combinatoires (~500)

Serveur	Zone distante	Service	Direction	Nb_ip
Serveur A	DMZ	ADMIN	ingress	1
Serveur A	Internet	WEB	ingress	~4.10^9
Serveur A	DMZ	WEB	ingress	256
Serveur B	DMZ	WEB	egress	256

Serveur	Ingress_ DMZ_ADMIN	Ingress_ Internet_WEB	Ingress_ DMZ_WEB	Egress_ DMZ_WEB
Serveur A	0.3	0	0	0
Serveur A	0	~9	0	0
Serveur A	0	0	2.4	0
Serveur B	0	0	0	2,4

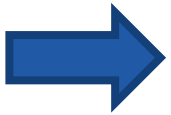
CDJDD : QUELQUES TOURS DE TOURNEVIS EN PLUS 2/3

- Application d'un logarithme sur toutes les valeurs
  - Atténue les extrêmes positifs
  - Plus conforme à une appréciation de la sécurité
  - Plus efficace pour le clustering



- Agrégation des lignes pour n'avoir plus qu'une ligne par serveur
  - On s'approche de notre tableau de données gérable pour de la statistique : chaque vecteur ligne de notre matrice représente les filtrages appliqués à une machine

Serveur	Ingress_ DMZ_ADMIN	Ingress_ Internet_WEB	Ingress_ DMZ_WEB	Egress_ DMZ_WEB
Serveur A	0.3	0	0	0
Serveur A	0	~9	0	0
Serveur A	0	0	2.4	0
Serveur B	0	0	0	2,4



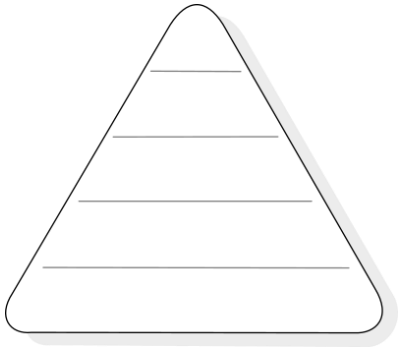
Serveur	Ingress_ DMZ_ADMIN	Ingress_ Internet_WEB	Ingress_ DMZ_WEB	Egress_ DMZ_WEB
Serveur A	0.3	~9	2.4	0
Serveur B	0	0	0	2,4

CDJDD : QUELQUES TOURS DE TOURNEVIS EN PLUS 3/3

- Last but not least : réduction de dimensions/colonnes
  - Suppression des dimensions avec une corrélation linéaire parfaite
  - Au final il ne reste plus que ~180 dimensions
    - Dû aux filtrages larges 0.0.0.0/0 qui « activent » toutes les zones de manière uniforme.

- Fin de la partie de préparation

CLUSTERING / VISUALISATION / APPRENTISSAGE NON SUPERVISÉ

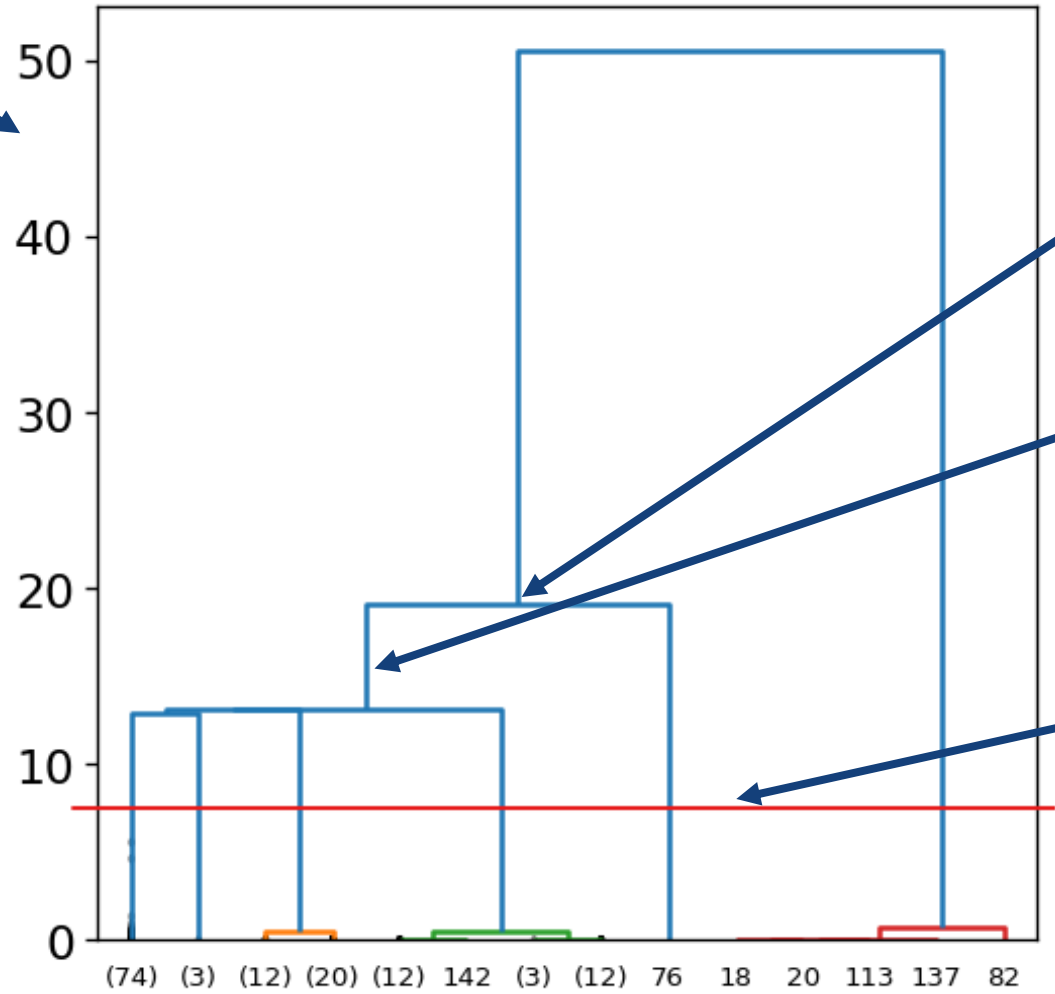


- On va utiliser une Classification Ascendante Hiérarchique (CAH)
  - Regroupement des individus (serveurs) de proche en proche en priorisant les plus proches
  - A l'avantage de visuellement « voir » ce qui se passe dans les regroupements pour déterminer :
    - Si on a des regroupements pertinents
    - Déterminer un nombre de groupes (cluster) adapté pour analyser nos données
    - Ça fait un graphique avec un nom sympa à placer dans un repas de famille : un dendrogramme.

```
clustering =  AgglomerativeClustering(distance_threshold=0, n_clusters=None, linkage="single").fit(X)
```

CLUSTERING / VISUALISATION / APPRENTISSAGE NON SUPERVISÉ

Dendrogramme résultat pour le périmètre 2



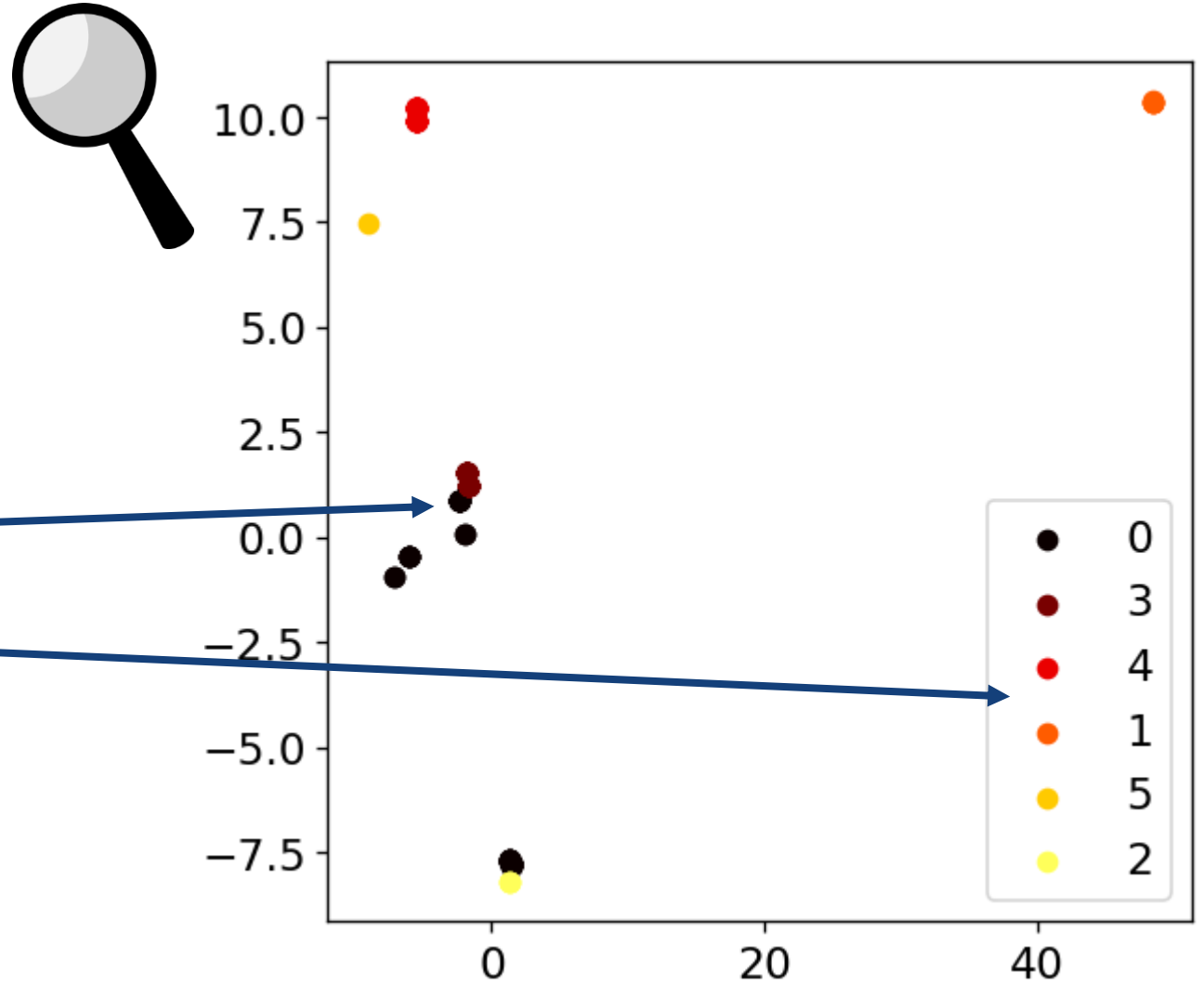
Un regroupement entre un groupe et un individu

La hauteur des « étages » entre les deux individus/groupes : plus les groupes d'individus sont distants, plus la hauteur avant regroupement est importante

On décide de fixer le regroupement à cet étage, soit 6 groupes/clusters. (Explications en séance)

CLUSTERING / VISUALISATION / APPRENTISSAGE NON SUPERVISÉ

- Représentation en ACP (analyse en composantes principales) sur les 2 dimensions principales
  - Cela donne une projection imparfaite en 2D des données
  - Chaque point est un serveur
  - Les couleurs correspondent aux regroupements issus de la CAH

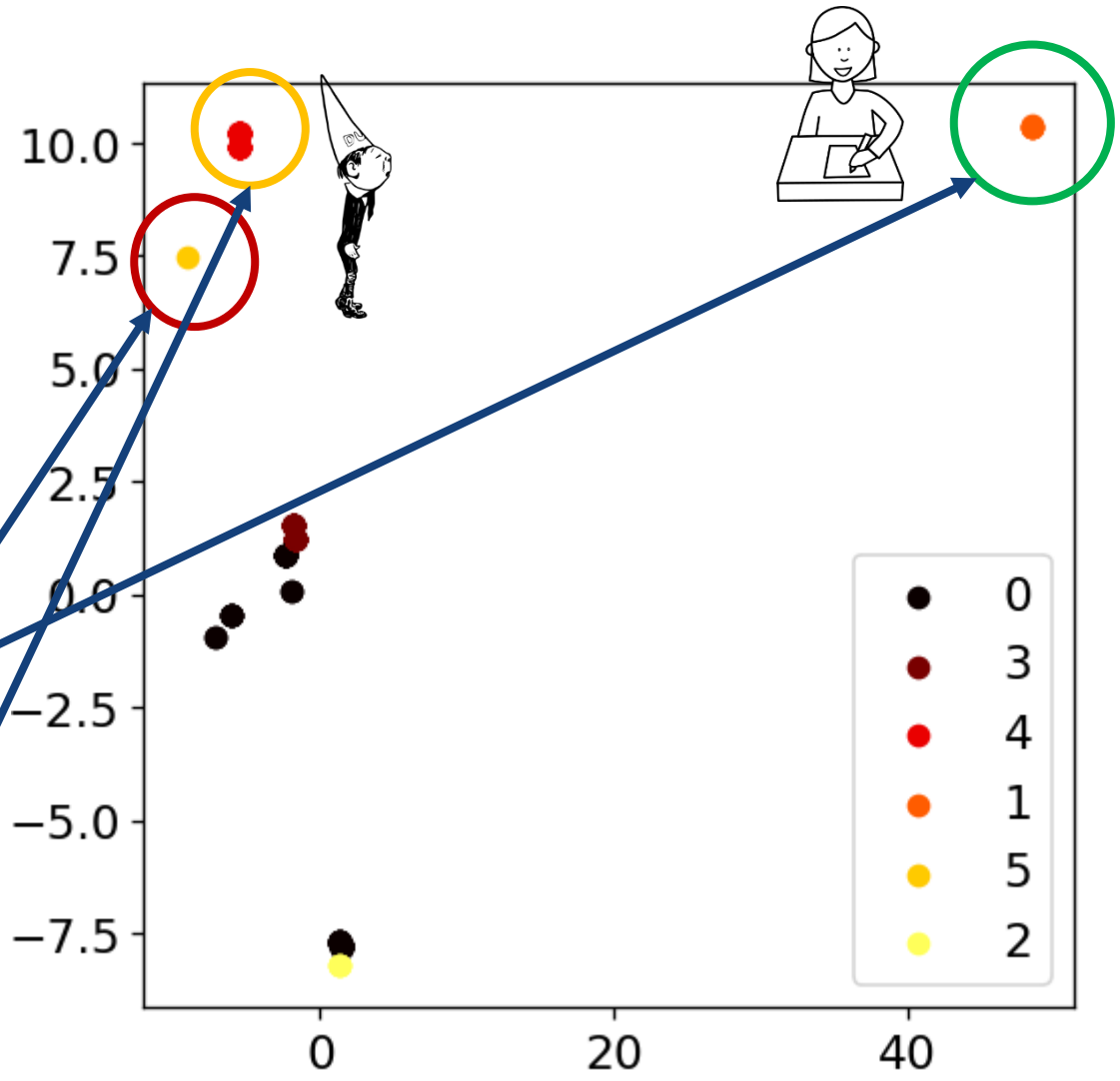



```
pca = decomposition.PCA(n_components=2)
X_transform = pca.fit_transform(X)

#print ind graph
unique_cluster = pd.Series(clusters).unique()
for i,c in enumerate(unique_cluster):
    plt.scatter(X_transform[clusters == c,0],X_transform[clusters == c,1], c=cm.hot(i/len(unique_cluster)), label=c)
plt.legend()
plt.show()
```

CLUSTERING / VISUALISATION / APPRENTISSAGE NON SUPERVISÉ

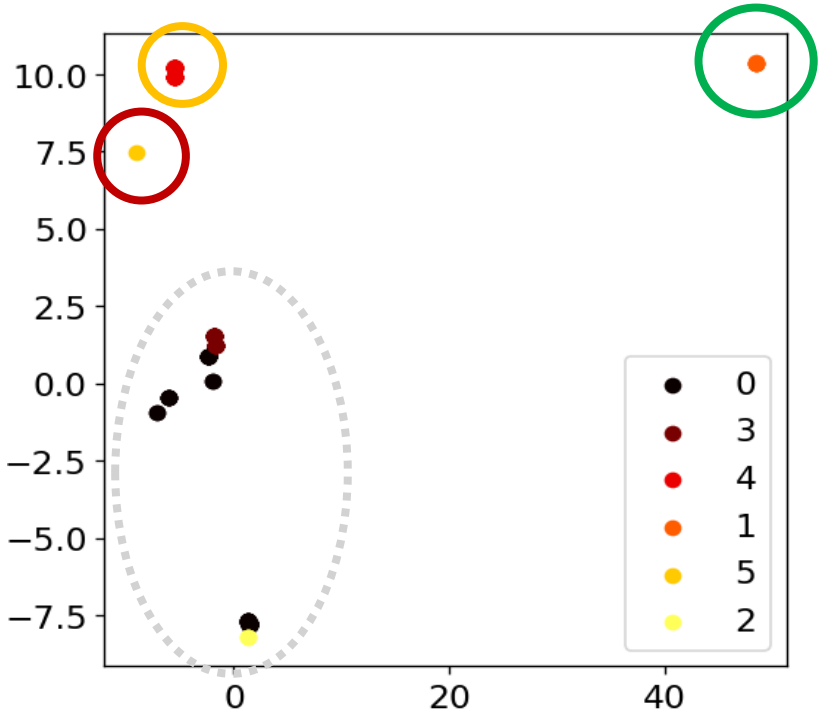
- Résultats pour le périmètre 2 (143 serveurs)
  - 6 groupes identifiés, quelques échantillons dans chaque groupe ont été analysés manuellement (analyse des règles filtrages humainement lisibles) :
    - Groupe 1 : les bons élèves (5)
    - Groupe 5 : retrait de permis (1)
    - Groupe 4 : ouvertures trop larges en entrée (33)



CLUSTERING / VISUALISATION / APPRENTISSAGE NON SUPERVISÉ

« Avoir une vue globale de la sécurité des filtrages en place »

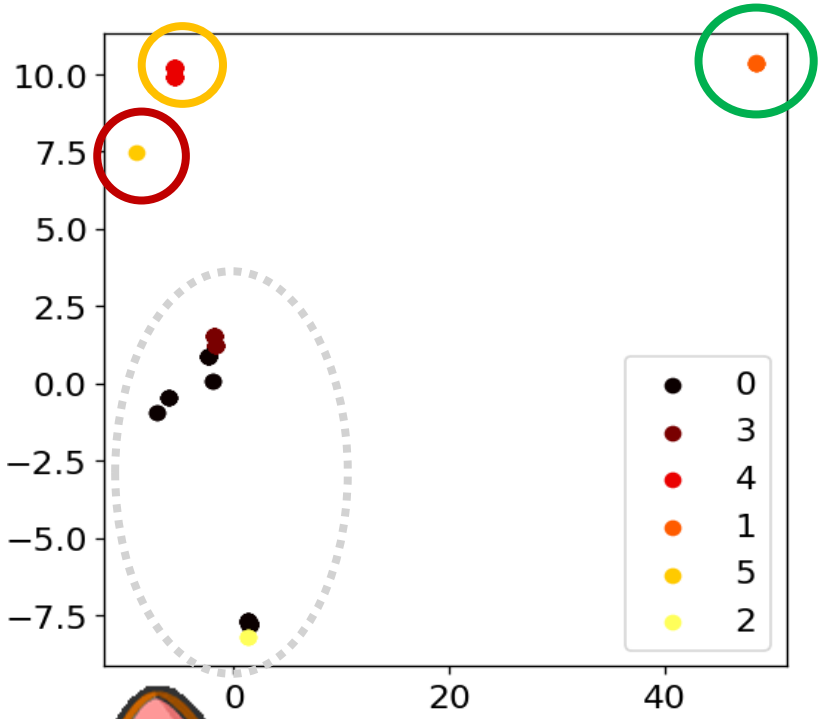
- Le clustering a permis de regrouper les meilleurs filtrages (pour la sécurité) et les plus mauvais
- Reste un groupe majoritaire « d'indéterminées »
- Filtrages analysés sur 6 machines = 34 machines à risque identifiées
  - Rendement plus important à l'échelle.
- Ça tourne sur un poste de travail



CLUSTERING / VISUALISATION / APPRENTISSAGE NON SUPERVISÉ

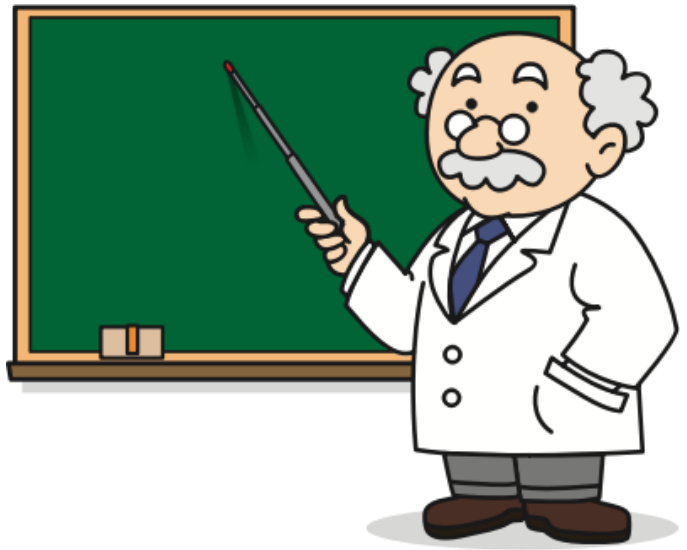
« Avoir une vue globale de la sécurité des filtrages en place »

- Le clustering a permis de regrouper les meilleurs filtrages (pour la sécurité) et les plus mauvais
- Reste un groupe majoritaire « d'indéterminées »
- Filtrages analysés sur 6 machines = 34 machines à risque identifiées
  - Rendement plus important à l'échelle.
- Ça tourne sur un poste de travail



APPRENTISSAGE SUPERVISÉ

- Apprentissage supervisé = besoin de données labellisées: chaque machine doit avoir un label en lien avec l'objectif : bon filtrage ou mauvais filtrage.
- On pourrait passer en revue les 2000 ou 150 machines précédentes…
- … ou prendre le résultat de notre clustering précédent pour entrainer un modèle d'apprentissage



APPRENTISSAGE SUPERVISÉ

- Quel modèle ? Tous !
- Pycaret : « low-code machine learning » (l'apprentissage automatique pour les nuls)
  - On reprend nos données précédentes, et on met le label « mauvais » aux groupes 4 et 5, et « bon » pour le reste.
  - Et :

```
from pycaret.classification import *
s = setup(X, target = 'label', fold = 5)
best = compare_models()
```

  - Résultats :

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lr	Logistic Regression	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	3.4060
knn	K Neighbors Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	2.3400
dt	Decision Tree Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0740
svm	SVM - Linear Kernel	1.0000	0.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0560
rf	Random Forest Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.3140
ada	Ada Boost Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.2720
gbc	Gradient Boosting Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.2420
et	Extra Trees Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.3260
lightgbm	Light Gradient Boosting Machine	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.2320

APPRENTISSAGE SUPERVISÉ

- Quel modèle ? Tous !
- Pycaret : « low-code machine learning » (l'apprentissage automatique pour les nuls)
  - On reprend nos données précédentes, et on met le label « mauvais » aux groupes 4 et 5, et « bon » pour le reste.
  - Et :

```
from pycaret.classification import *
s = setup(X, target = 'label', fold = 5)
best = compare_models()
```

  - Résultats :

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lr	Logistic Regression	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	3.4060
knn	K Neighbors Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	2.3400
dt	Decision Tree Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0740
svm	SVM - Linear Kernel	1.0000	0.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0560
rf	Random Forest Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.3140
ada	Ada Boost Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.2720
gbc	Gradient Boosting Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.2420
et	Extra Trees Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.3260
lightgbm	Light Gradient Boosting Machine	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.2320
nb	Naive Bayes	0.0000	0.0750	0.0500	1.0000	0.0714	0.0655	0.0680	0.0000

APPRENTISSAGE SUPERVISÉ

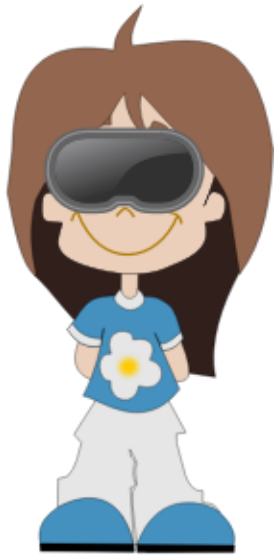
- Rien de magique : on apprend sur un résultat de clustering : les groupes sont déjà bien formés…
- Le but est de « bootstrapper » un modèle avec des données initiales pour ensuite le mettre en production et commencer à noter les faux positifs/faux négatifs
- Et relancer l'évaluation des modèles pour voir celui qui se débrouille le mieux avec les nouvelles données
- Easy ?

APPRENTISSAGE SUPERVISÉ

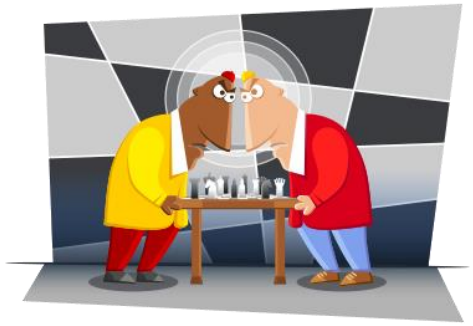
- Quelques points à avoir en tête sur la mise en production dans un monde réel :
  - Conserver un jeu de données d'apprentissage dans le temps ;
  - Mettre en œuvre une boucle d'enrichissement des données : labellisation par des analystes (i.e faux positifs et faux négatifs), résultats de futures revues de configuration;
  - Mettre à jour régulièrement le modèle en relançant l'apprentissage ;
  - Vérifier une éventuelle dérive du modèle et éventuellement changer d'algorithme ou les hyper-paramètres (paramètres utilisés pour contrôler le processus d'apprentissage) ;
  - Mettre à disposition le modèle de classification (API) pour les applications clientes ;
  - Intégrer les appels au modèle depuis les applications clientes (outils de ticketing, SIEM, etc.).

APPRENTISSAGE SUPERVISÉ

- « Concevoir un outil de détection de filtrages réseaux à risques non acceptables »
  - En théorie, il y a du potentiel…
  - … si on investit.
  - Pas d'action concrète effectuée au-delà du PoC dans le cadre de l'expérimentation

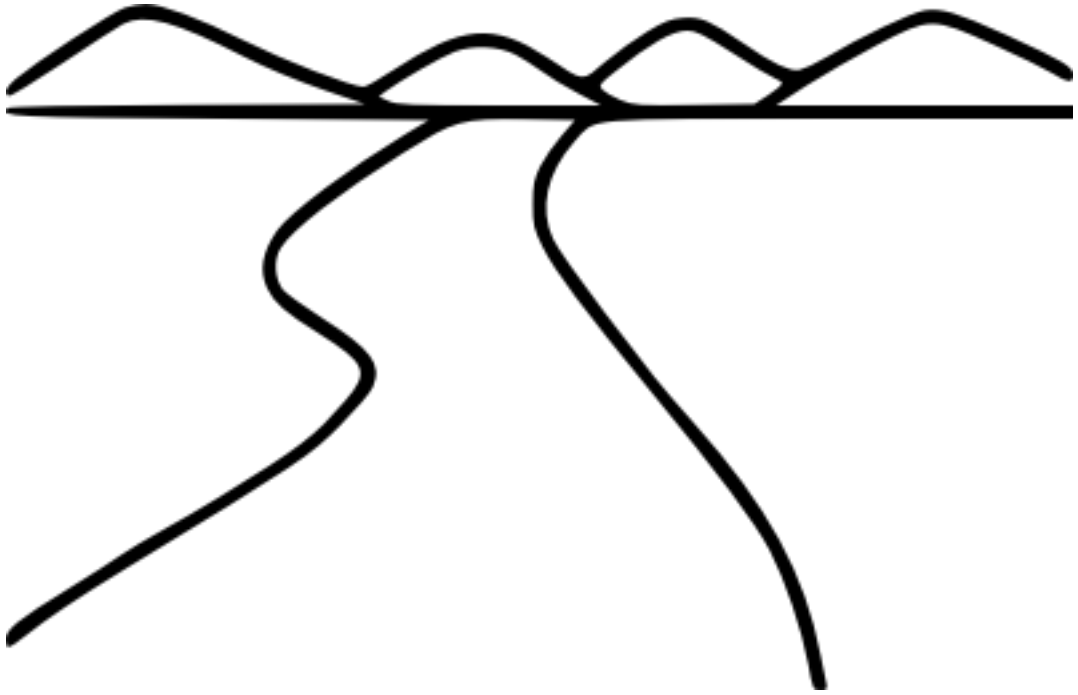


UN PEU DE RECUL (SUBJECTIF)



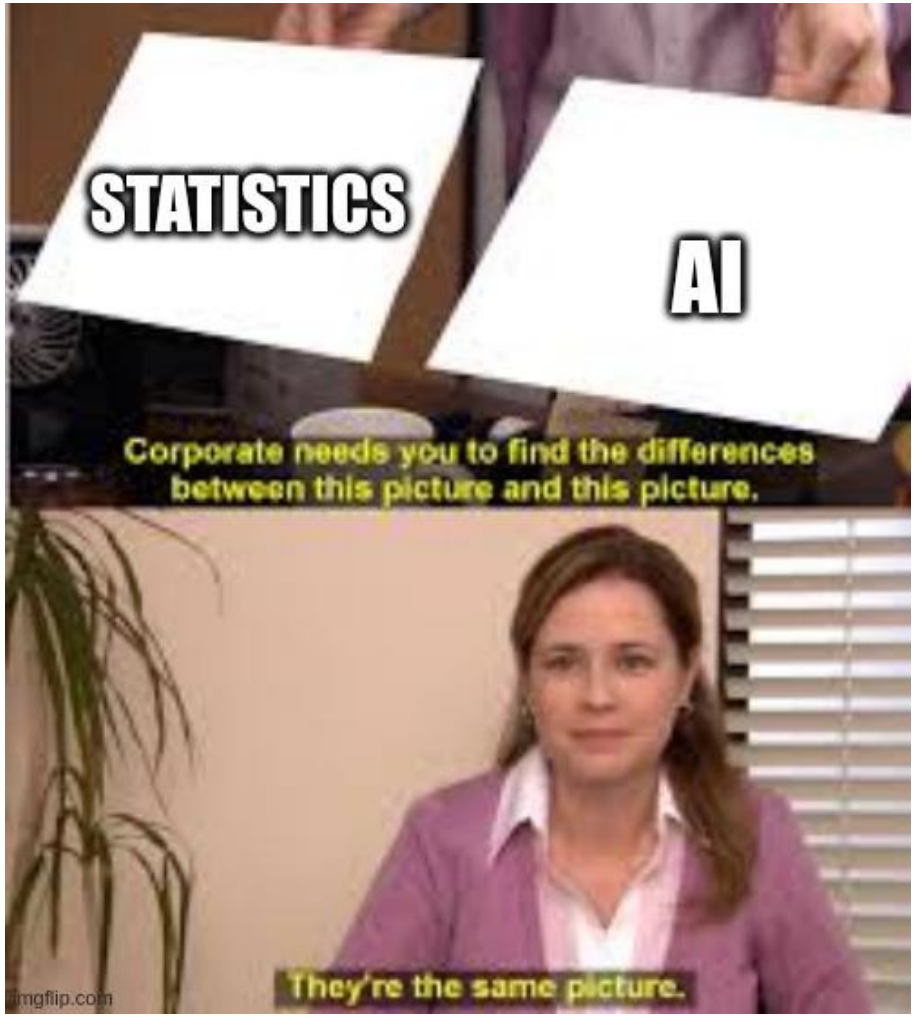
Analyse de filtrage classique	Analyse de filtrage par analyse statistique
Analyse de filtrages sur 150 machines	Analyse de filtrage sur 6 machines + extrapolation sur les groupes ✓
✓ Erreurs humaines	Erreurs de classification
✓ Un fichier excel	Infrastructure « usine à gaz »
Expert réseau	Machine (à supprimer des experts ☺… ou pas) ✓
✓ Analyse « justifiable » par un expert	Règles de décision opaques
Rédaction de règles « expertes »	Règles générées implicitement par l'apprentissage
La réalité : règle de détection sur l'ouverture des port 22 et 80 depuis des réseaux >/32	La réalité : on n'aurait pas fait mieux avec le résultat clustering…

DU COUP, À REFAIRE OU PAS ? QUEL CAS D'USAGE ?



- Analyse d'un périmètre large
  - Mode « quick and dirty »
  - Hors objectif de conformité
- Conception d'un modèle de décision à destination des projets
  - Permettrait au projet de s'auto-évaluer sans (avant de) monopoliser un expert

QUESTIONS ?


STATISTICS
AI
Corporate needs you to find the differences between this picture and this picture.
They're the same picture.
imgflip.com