



Université
Paris Cité

Détection d'intrusion réseau

UE Fouille de Données et Aide à la Décision

Leonard NAMOLARU · Abdennour MESSIKH

+++

Introduction

Le moyen le plus probable pour des attaquants d'accéder à une infrastructure est de passer par le réseau.

Sécurité réseau

Protéger les réseaux informatiques contre la malveillance, l'utilisation abusive et le déni de service (DoS).

Objectifs du projet

- Utiliser l'apprentissage automatique pour découvrir des corrélations entre les données du réseau.
- Classification du trafic réseau.
- Découvrir les attaquants à l'intérieur du réseau.



Jeu de données



Exploration des données

Premier diagnostic

seul le fichier *Train_data.csv* contient la colonne *class* qui permet de déterminer s'il s'agit d'une anomalie ou d'un trafic réseau normal.

- Ignorer le fichier *Test_data.csv* et utiliser (après le *pre-processing*) que le fichier *Train_data.csv* pour la division en 3 ensembles : entraînement, validation, test.
- Par conséquent, par commodité, tout au long de la présentation, l'utilisation du terme « jeu de données » ou « *Dataset* » fera référence uniquement au fichier *Train_data.csv*.

Exploration des données

42 colonnes

- duration
- protocol_type
- Service
- Flag
- src_bytes
- dst_bytes
- Land
- wrong_fragment
- Urgent
- Hot
- num_failed_logins
- logged_in
- num_compromised
- root_shell
- su_attempted
- num_root
- num_file_creations
- num_shells
- num_access_files
- num_outbound_cmds
- is_host_login
- is_guest_login
- Count
- srv_count
- serror_rate
- srv_serror_rate
- rerror_rate
- srv_rerror_rate
- same_srv_rate
- diff_srv_rate
- srv_diff_host_rate
- dst_host_count
- dst_host_srv_count
- dst_host_same_srv_rate
- dst_host_diff_srv_rate
- dst_host_same_src_port_rat
- dst_host_srv_diff_host_rat
- dst_host_serror_rate
- dst_host_srv_serror_rate
- dst_host_rerror_rate
- dst_host_srv_rerror_rate
- class

Comment explorer les données avec autant de colonnes ?

Nous allons diviser les colonnes en 3 groupes : nominal, binaire, numérique

Exploration des données

exploration_des_donnees.py

```
obtenir_noms_colonnes_csv(jeu_de_donnees : str) -> list  
chaine_caracteres_est_elle_nombre(nombre : str) -> bool  
convertir_chaine_caracteres_int_float(nombre : str)
```

```
repartition_colonnes_selon_type_donnees_dans_colonne(jeu_de_donnees : str,  
noms_colonnes : list) -> dict :
```

La fonction renvoie un dictionnaire dans lequel les noms des colonnes sont divisés selon le type de données dans chaque colonne : binaire, numérique ou nominal.

```
impression_statistiques(jeu_de_donnees : str, colonnes_binaires : list,  
colonnes_nominales : list, colonnes_numeriques : list) -> None
```

Exploration des données

Vue sous forme de tableau de 5 lignes aléatoires :

	duration	protocol_type	service	flag	...	dst_host_srv_serror_rate	dst_host_rerror_rate	dst_host_srv_rerror_rate	class
15180	0	tcp	gopher	S0	...	1.00	0.00	0.00	anomaly
7541	0	tcp	http	SF	...	0.00	0.00	0.00	normal
14653	0	udp	domain_u	SF	...	0.00	0.00	0.00	normal
16863	0	tcp	private	S0	...	1.00	0.00	0.00	anomaly
9094	0	tcp	http	REJ	...	0.01	0.33	0.37	normal

[5 rows x 42 columns]

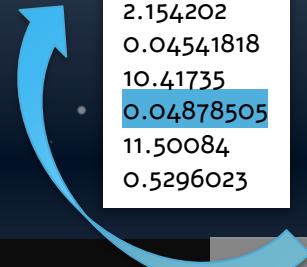
Colonnes binaires

Colonnes binaires :	count	mean	std	min	25%	50%	75%	max
land	25192.0	0.000079	0.008910	0.0	0.0	0.0	0.0	1.0
urgent	25192.0	0.000040	0.006300	0.0	0.0	0.0	0.0	1.0
logged_in	25192.0	0.394768	0.488811	0.0	0.0	0.0	1.0	1.0
root_shell	25192.0	0.001548	0.039316	0.0	0.0	0.0	0.0	1.0
num_shells	25192.0	0.000357	0.018898	0.0	0.0	0.0	0.0	1.0
num_outbound_cmds	25192.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0
is_host_login	25192.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0
is_guest_login	25192.0	0.009130	0.095115	0.0	0.0	0.0	0.0	1.0

2 colonnes qui ne contiennent que la valeur 0

Exploration des données

Colonnes numériques 10 premières colonnes



Sans effectuer une normalisation,
src_bytes dominerait, ce qui ferait manquer
au modèle, par exemple, des informations
potentiellement importantes dans la colonne
su_attempted.

Colonnes numériques :

	count	mean	std	min	25%	50%	75%	max
duration	25192.0	305.054104	2.686556e+03	0.0	0.00	0.00	0.00	42862.0
src_bytes	25192.0	24330.628215	2.410805e+06	0.0	0.00	44.00	279.00	381709090.0
dst_bytes	25192.0	3491.847174	8.883072e+04	0.0	0.00	0.00	530.25	5151385.0
wrong_fragment	25192.0	0.023738	2.602208e-01	0.0	0.00	0.00	0.00	3.0
hot	25192.0	0.198039	2.154202e+00	0.0	0.00	0.00	0.00	77.0
num_failed_logins	25192.0	0.001191	4.541818e-02	0.0	0.00	0.00	0.00	4.0
num_compromised	25192.0	0.227850	1.041735e+01	0.0	0.00	0.00	0.00	884.0
su_attempted	25192.0	0.001350	4.878505e-02	0.0	0.00	0.00	0.00	2.0
num_root	25192.0	0.249841	1.150084e+01	0.0	0.00	0.00	0.00	975.0
num_file_creations	25192.0	0.014727	5.296023e-01	0.0	0.00	0.00	0.00	40.0

Exploration des données

```
flag  
OTH      5  
REJ     2216  
RSTO    304  
RSTOS0   21  
RSTR    497  
S0      7009  
S1       88  
S2       21  
S3       15  
SF     14973  
SH       43  
dtype: int64
```

```
class  
anomaly 11743  
normal   13449  
dtype: int64
```

Colonnes nominales

Nous devrons effectuer du
One-hot encoding



Label Encoding

```
protocol_type  
icmp      1655  
tcp       20526  
udp       3011  
dtype: int64  
  
service  
IRC        40  
X11        22  
Z39_50    172  
auth       189  
bgp        146  
...  
urp_i      124  
uucp      157  
uucp_path  133  
vmnet      107  
whois      145  
Length: 66, dtype: int64
```

Prétraitement des données

□ Suppression des colonnes inutiles

```
def suppression_colonnes_inutiles(data_frame : DataFrame, noms_colonnes : list) -> DataFrame :  
    for nom_colonne in noms_colonnes :  
        if len( data_frame.groupby([nom_colonne]).size() ) == 1:  
            data_frame.drop(nom_colonne, axis = 1, inplace = True)  
  
    return data_frame
```

□ Conversion des colonnes nominales en colonnes binaires

```
def conversion_colonnes_nominales_en_colonnes_binaires(data_frame : DataFrame, noms_colonnes_nominales : list) -> DataFrame :  
    for nom_colonne in noms_colonnes_nominales :  
        if len( data_frame.groupby([nom_colonne]).size() ) == 2:  
            data_frame[[nom_colonne]] = data_frame[[nom_colonne]].apply( LabelEncoder().fit_transform )  
        else :  
            data_frame = get_dummies(data_frame, columns=[nom_colonne])  
  
    return data_frame
```

Prétraitement des données

□ Normalisation

Le score standard d'un échantillon x est calculé comme suit :

$$Z = \frac{x - \mu}{\sigma}$$

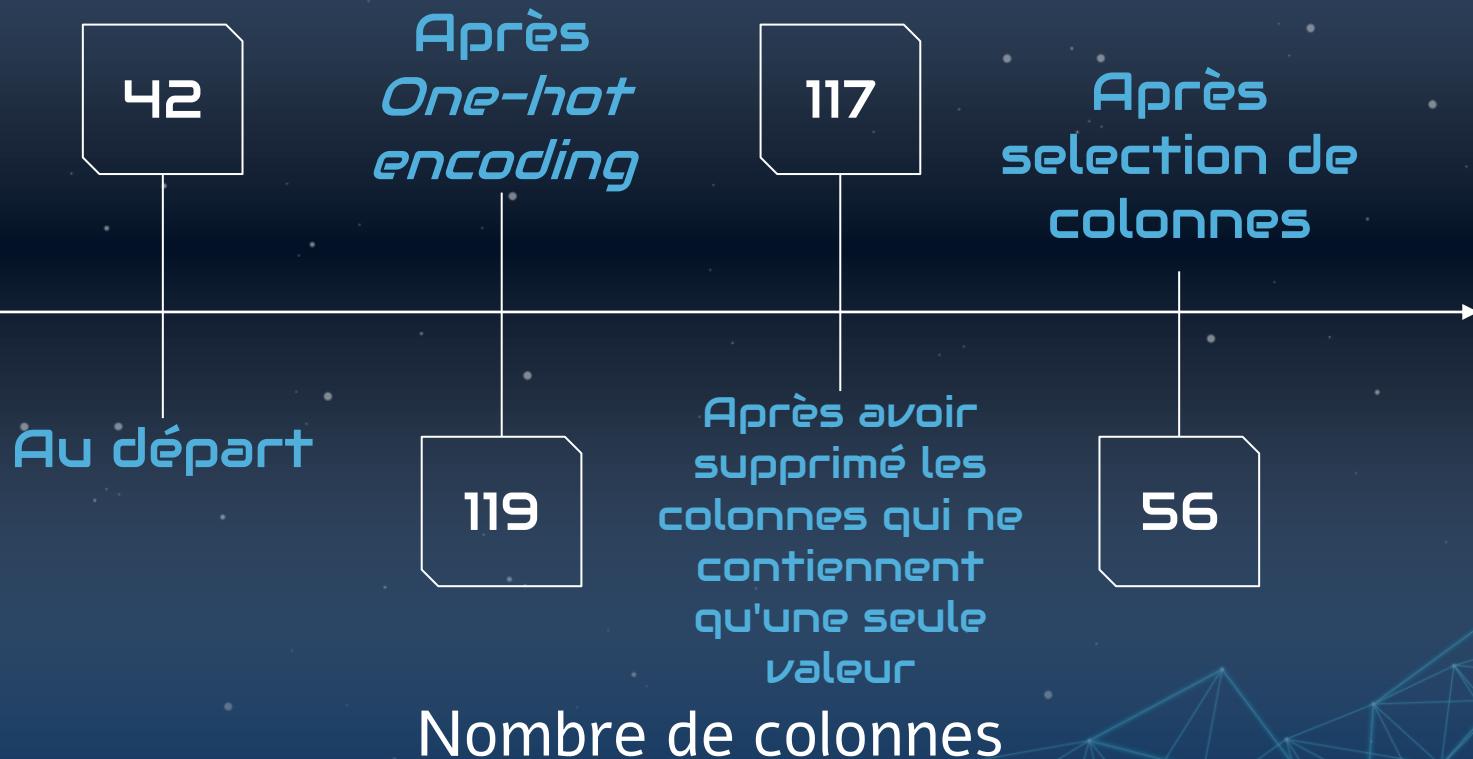
Moyenne

$$\mu = \frac{1}{N} \sum_{i=1}^N x_i$$

Ecart-type

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2}$$

Sélection de colonnes



```

def attribute_ratio(data_frame : DataFrame, colonnes_numeriques : list) -> DataFrame :
    classe = data_frame['class']

    moyennes = data_frame[colonnes_numeriques].mean()
    moyenne_par_classe = data_frame[ colonnes_numeriques + ['class']].groupby('class').mean()

    dictionnaire_attribute_ratio = dict()
    for colonne in colonnes_numeriques :
        dictionnaire_attribute_ratio[colonne] = max(moyenne_par_classe[colonne]) / moyennes[colonne]

    colonnes_binaires = []
    for colonne in data_frame.columns :
        if colonne not in colonnes_numeriques and colonne != 'class' :
            colonnes_binaires.append( colonne )

    dictionnaire = dict()
    for colonne in colonnes_binaires :
        serie_0 = data_frame[ data_frame[colonne] == 0 ].groupby('class').size()
        serie_1 = data_frame[ data_frame[colonne] == 1 ].groupby('class').size()

        if max(serie_1 / serie_0) :
            dictionnaire_attribute_ratio[colonne] = max(serie_1 / serie_0)

    dictionnaire_attribute_ratio = dict((cle, valeur) for cle,valeur in dictionnaire_attribute_ratio.items() if not isnan(valeur))
    dictionnaire = sorted(dictionnaire_attribute_ratio.items(), key = lambda x : x[1], reverse = True)

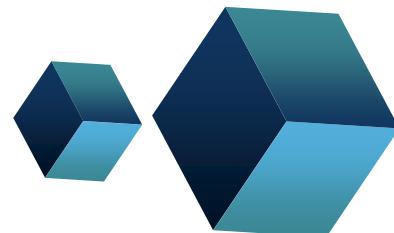
    colonnes = []
    for cle, valeur in dictionnaire :
        if valeur >= 0.01 :
            colonnes.append(cle)

    data_frame = data_frame[colonnes]
    colonnes_numeriques_mis_a_jour = list(set(colonnes_numeriques).intersection(colonnes))

    # .....

    data_frame['class'] = classe
    return data_frame

```



Sélection de colonnes

56 colonnes

- rerror_rate
- dst_host_srv_serror_rate
- srv_serror_rate
- same_srv_rate
- dst_host_same_srv_rate
- dst_host_srv_rerror_rate
- dst_host_rerror_rate
- duration
- srv_rerror_rate
- su_attempted
- num_compromised
- num_root
- dst_bytes
- logged_in
- srv_count
- src_bytes
- num_file_creations
- is_guest_login
- class
- service_http
- service_private
- service_domain_u
- service_smtp
- service_ftp_data
- service_eco_i
- service_ecr_i
- service_other
- service_finger
- service_telnet
- service_ftp
- service_Z39_50
- service_efs
- service_whois
- service_uucp
- service_auth
- service_courier
- service_nnsp
- service_iso_tsap
- service_time
- service_uucp_path
- service_imap4
- service_bgp
- service_ctf
- service_hostnames
- flag_SF
- flag_S0
- flag_REJ
- flag_RSTR
- flag_RST0
- protocol_type_tcp
- protocol_type_udp
- protocol_type_icmp

Division du jeu de données en trois

```
def division(jeu_de_donnees : str, graine : int, entrainement : str, validation : str, test : str) -> None
    seed(graine)

    i = 0
    for ligne in fichier_jeu_de_donnees :
        valeur_aleatoire = random() # random() -> x est dans l'intervalle [0, 1).

            if i != 0 :
                fichier = None
                if(valeur_aleatoire < (1 / 3)) :
                    fichier = fichier_entrainement
                elif (valeur_aleatoire < (2 / 3)) :
                    fichier = fichier_validation
                else :
                    fichier = fichier_test

                fichier.write(ligne)

            else : # i == 0 => C'est la premiere ligne avec les noms des colonnes, donc cette ligne sera copiee dans les 3 fichiers
                fichier_entrainement.write(ligne)
                fichier_validation.write(ligne)
                fichier_test.write(ligne)

        i += 1

division('jeu_de_donnees_apres_pretreatment_selection_colonnes.csv', time(), 'entraiment.csv', 'validation.csv', 'test.csv')
```

Naïve bayes

- Un classificateur basé sur l'application du théorème de Bayes.
- Hypothèse d'indépendance entre les caractéristiques.

Théorème de Bayes

La probabilité qu'un paquet réseau soit une anomalie, sachant, par exemple, que le protocole de couche transport utilisé est TCP :

$$P(ANOMALIE|TCP) = \frac{\frac{\% \text{ d'anomalies}}{\text{avec le protocole TCP}} \times \frac{\% \text{ d'anomalies}}{P(ANOMALIE)}}{\underbrace{P(TCP)}_{\% \text{ de paquets réseau avec le protocole TCP}}}$$

Naïve bayes

Score D'entraînement 0.9333413259800983 || Score de validation 0.930936276254895

	precision	recall	f1-score	support
0	0.97	0.88	0.92	3860
1	0.90	0.98	0.94	4567
accuracy			0.93	8427
macro avg	0.94	0.93	0.93	8427
weighted avg	0.93	0.93	0.93	8427

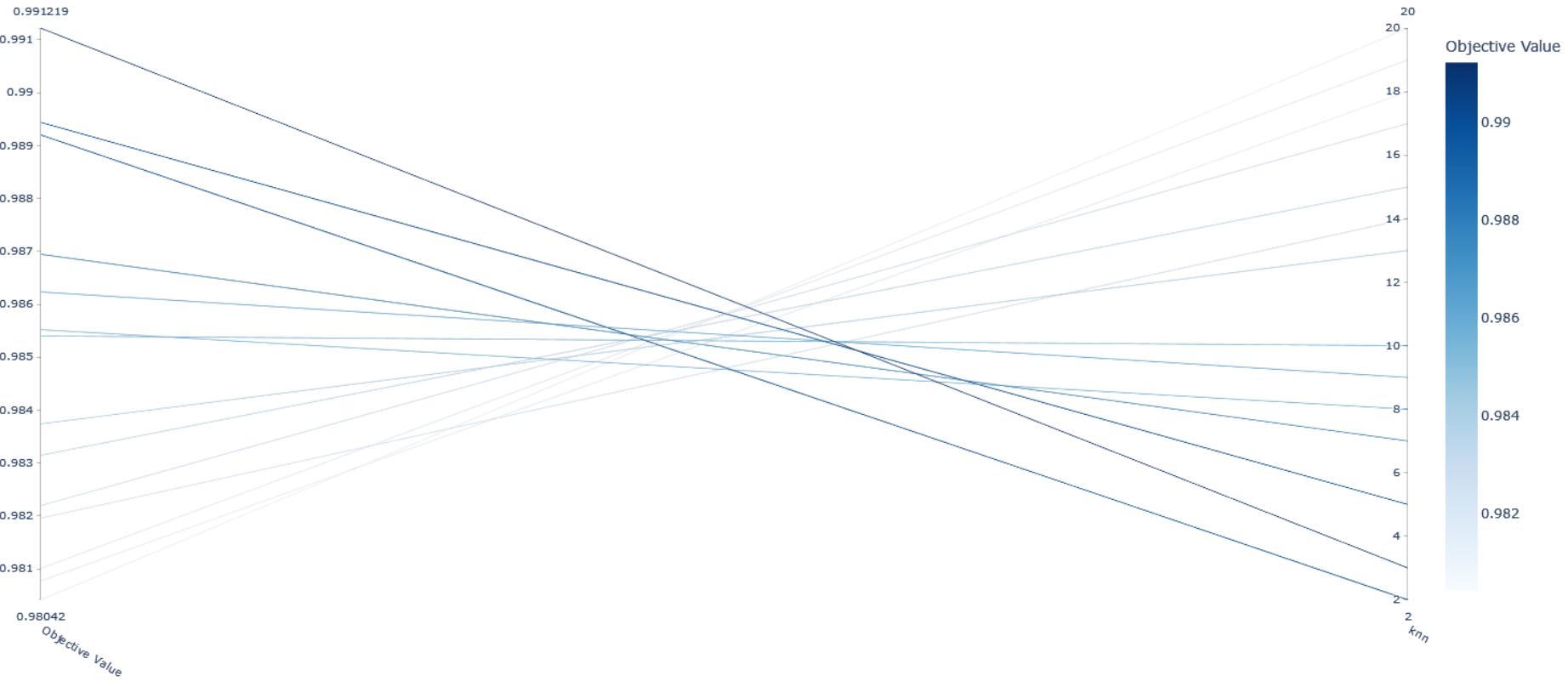
Les K plus proches voisins

- Entrée : les k exemples d'apprentissage les plus proches dans le jeu de données.

Score D'entraînement 0.9928066179115214 || Score de validation 0.9912187017918594

	precision	recall	f1-score	support
0	0.99	0.99	0.99	3860
1	0.99	0.99	0.99	4567
accuracy			0.99	8427
macro avg	0.99	0.99	0.99	8427
weighted avg	0.99	0.99	0.99	8427

Parallel Coordinate Plot

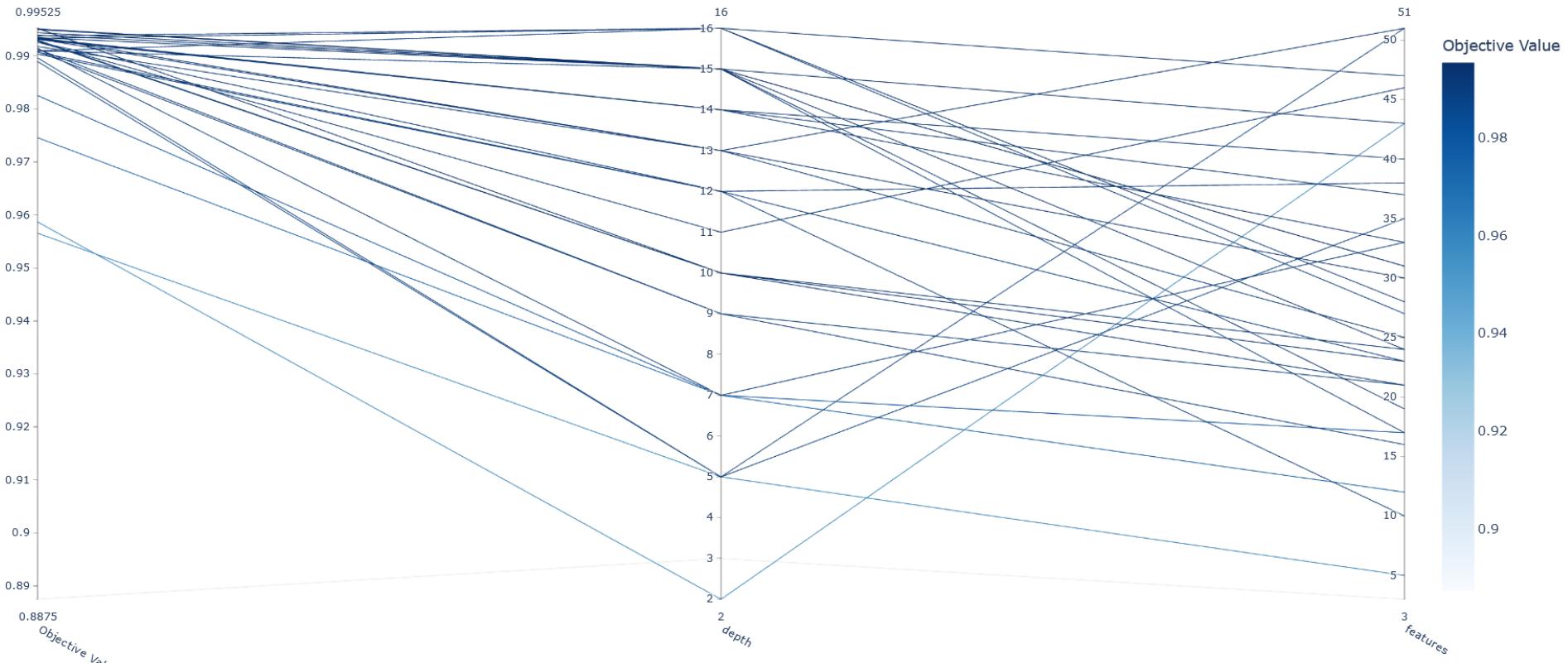


Arbres de décision

- Un arbre de décision est un outil qui utilise un modèle arborescent de décisions et leurs conséquences possibles.
- Principe : réfléchir dimension par dimension, probablement le plus proche d'une décision humaine. (Source : cours 7)

Score D'entraînement 1.0 Score de validation 0.9911000355998576				
	precision	recall	f1-score	support
0	0.99	0.99	0.99	3860
1	0.99	0.99	0.99	4567
accuracy			0.99	8427
macro avg	0.99	0.99	0.99	8427
weighted avg	0.99	0.99	0.99	8427

Parallel Coordinate Plot

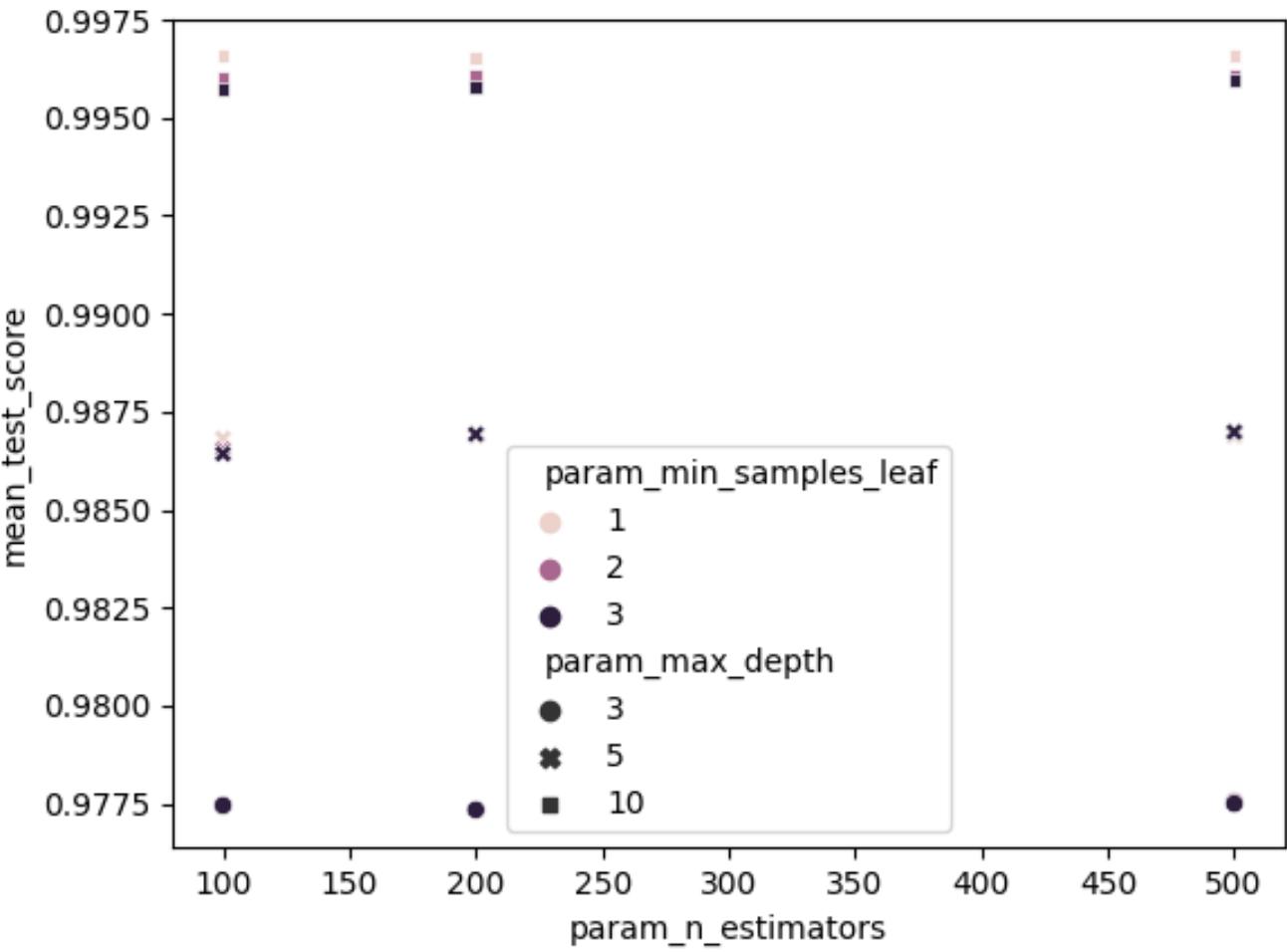


Forêts aléatoires

- Construction d'une multitude d'arbres de décision.

```
Score D'entraînement 0.9977220956719818 || Score de validation 0.9983386733119735
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	3860
1	1.00	1.00	1.00	4567
accuracy			1.00	8427
macro avg	1.00	1.00	1.00	8427
weighted avg	1.00	1.00	1.00	8427



```
def forets_aleatoires(x_entrainement : pd.DataFrame, y_entrainement : pd.DataFrame, x_validation : pd.DataFrame, y_validation : pd.DataFrame) -> tuple :
    rfc_modele = RandomForestClassifier(random_state = 0)
    params = {'n_estimators': [100, 200, 500], 'max_depth': [3, 5, 10], "min_samples_leaf": [1, 2, 3]}

    # Utilisation de GridSearchCV pour trouver le meilleur ensemble d'hyperparametres
    gs = GridSearchCV(rfc_modele, params, cv=4, scoring='f1')
    gs.fit(pd.concat([x_entrainement, x_validation]), pd.concat([y_entrainement, y_validation]))

    df = pd.DataFrame(gs.cv_results_)
    sns.scatterplot(data=df, x='param_n_estimators', y='mean_test_score', style='param_max_depth', hue='param_min_samples_leaf')
    plt.show()
    plt.savefig("forets_aleatoires.png")

    rfc_modele = gs.best_estimator_

    # Score du modele
    y_pred = rfc_modele.predict(x_validation)

    random_forest_score_entrainement = rfc_modele.score(x_entrainement, y_entrainement)
    random_forest_score_validation = rfc_modele.score(x_validation, y_validation)

    return (random_forest_score_entrainement, random_forest_score_validation, classification_report(y_validation, y_pred))
```

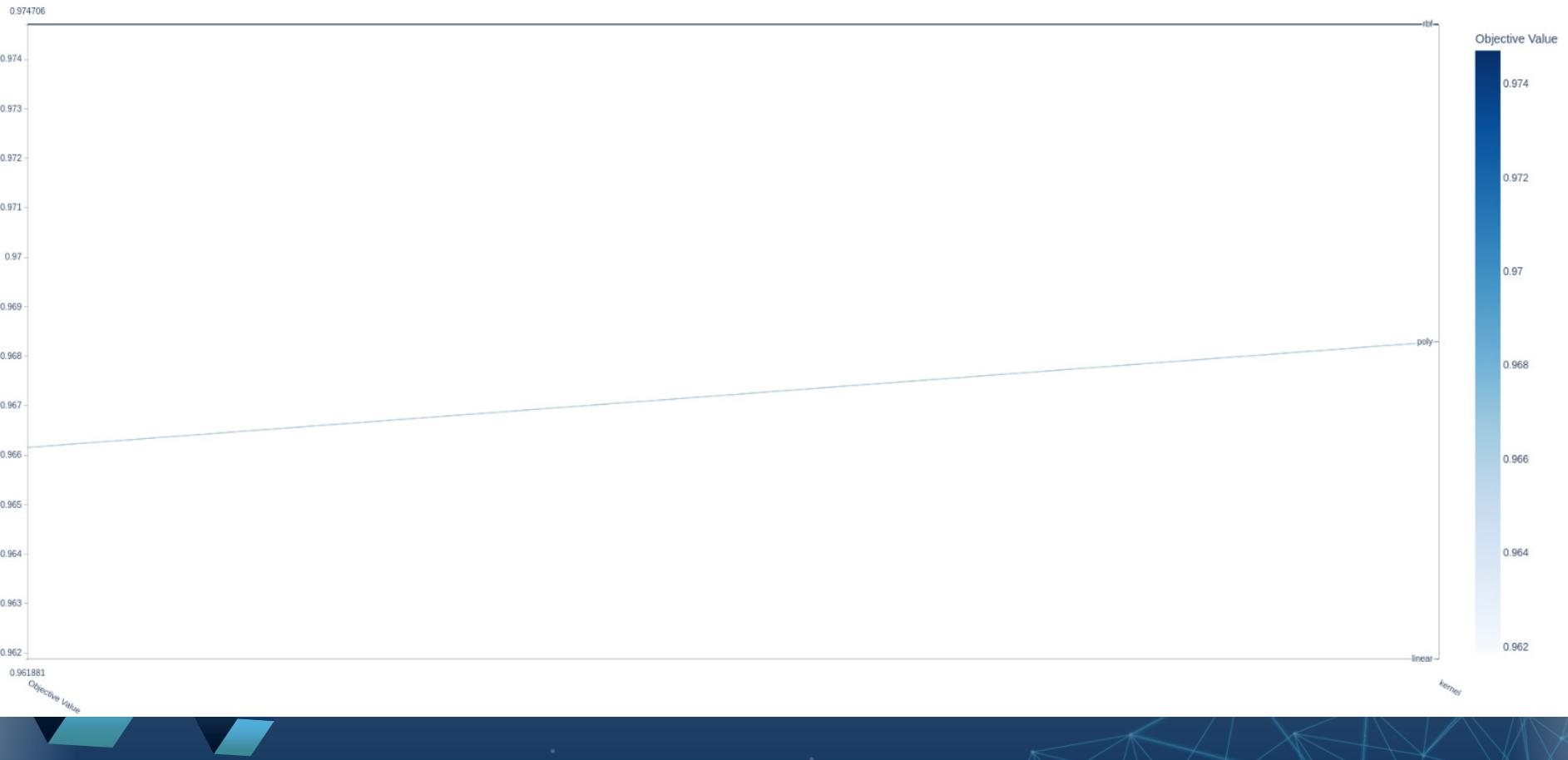
Support vector machine

- SVM mappe les exemples d'entraînement sur des points dans l'espace afin de maximiser la largeur de l'écart entre les deux catégories. De nouveaux exemples sont ensuite cartographiés dans ce même espace et prédits comme appartenant à une catégorie en fonction de quel côté ils se situent. (source : en.wikipedia.org)

Score D'entraînement 0.9857331255245174 || Score de validation 0.9869467188797911

	precision	recall	f1-score	support
0	0.99	0.98	0.98	3900
1	0.98	0.99	0.99	4441
accuracy			0.99	8341
macro avg	0.99	0.99	0.99	8341
weighted avg	0.99	0.99	0.99	8341

Parallel Coordinate Plot

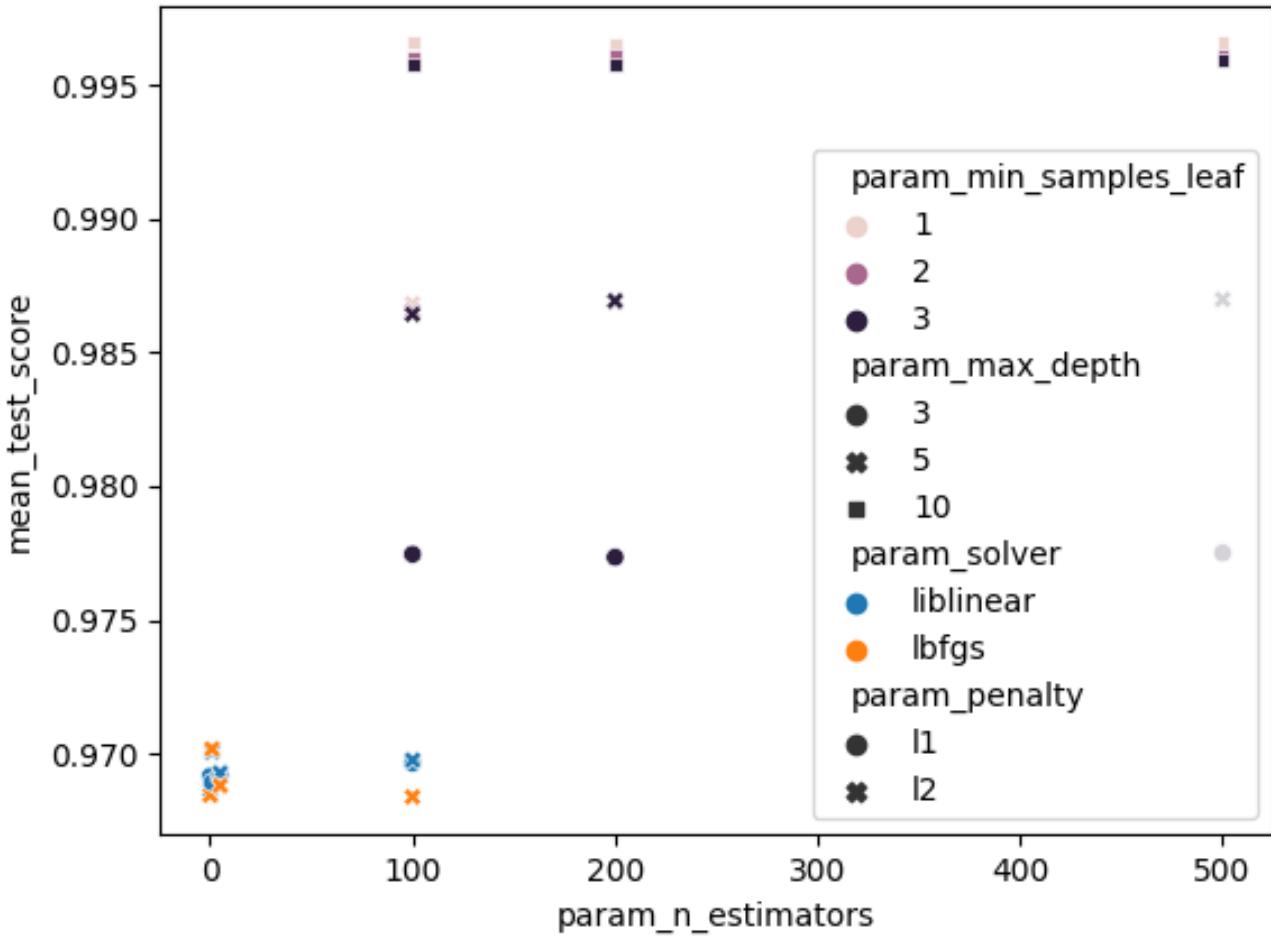


Régression logistique

- La probabilité qu'un événement se produise en faisant en sorte que la probabilité logarithmique de l'événement soit une combinaison linéaire d'une ou plusieurs variables indépendantes. (Source : en.wikipedia.org)

Score D'entraînement 0.9857331255245174 || Score de validation 0.9869467188797911

	precision	recall	f1-score	support
0	0.98	0.96	0.97	3900
1	0.96	0.98	0.97	4441
accuracy			0.97	8341
macro avg	0.97	0.97	0.97	8341
weighted avg	0.97	0.97	0.97	8341



Code - récapitulatif

- exploration_des_donnees.py
- pretraitement_des_donnees.py
- selection_colonnes.py
- diviser_jeu_de_donnees_en_trois.py
- modeles.py

Résultats

Naïve bayes

• .

Score D'entraînement 0.9333413259800983 || Score de test 0.9293684710351378

	precision	recall	f1-score	support
0	0.97	0.88	0.92	3983
1	0.90	0.98	0.94	4441
accuracy			0.93	8424
macro avg	0.93	0.93	0.93	8424
weighted avg	0.93	0.93	0.93	8424

Régression logistique



Score D'entraînement 0.9857331255245174 || Score de test 0.9863485280151947

	precision	recall	f1-score	support
0	0.98	0.95	0.97	3900
1	0.96	0.98	0.97	4441
accuracy			0.97	8341
macro avg	0.97	0.97	0.97	8341
weighted avg	0.97	0.97	0.97	8341

Résultats

Les K
plus proches
voisins

Score D'entraînement 0.9956839707469128 || Score de test 0.988960113960114

	precision	recall	f1-score	support
0	0.98	1.00	0.99	3983
1	1.00	0.98	0.99	4441
accuracy			0.99	8424
macro avg	0.99	0.99	0.99	8424
weighted avg	0.99	0.99	0.99	8424

Support
vector
machine

Score D'entraînement 0.9857331255245174 || Score de test 0.9863485280151947

	precision	recall	f1-score	support
0	0.99	0.98	0.98	3900
1	0.98	0.99	0.99	4441
accuracy			0.99	8341
macro avg	0.99	0.99	0.99	8341
weighted avg	0.99	0.99	0.99	8341

Résultats

Arbres de décision

Score D'entraînement 0.9956839707469128 || Score de test 0.988960113960114

	precision	recall	f1-score	support
0	0.98	1.00	0.99	3983
1	1.00	0.98	0.99	4441
accuracy			0.99	8424
macro avg	0.99	0.99	0.99	8424
weighted avg	0.99	0.99	0.99	8424

Forêts aléatoires

Score D'entraînement 0.9984414338808296 || Score de test 0.9990503323836657

	precision	recall	f1-score	support
0	1.00	1.00	1.00	3983
1	1.00	1.00	1.00	4441
accuracy			1.00	8424
macro avg	1.00	1.00	1.00	8424
weighted avg	1.00	1.00	1.00	8424

Discussion

- Un projet enrichissant qui nous a permis d'expérimenter la réalisation d'un projet de machine learning.
- Comprendre l'importance de la procédure de prétraitement.
- Une combinaison de 2 grands domaines de l'informatique aujourd'hui : l'apprentissage automatique et la cybersécurité.

Contributions

- **Idée du projet:** Leonard Namolaru
- **Naive bayes, les K plus proches voisins, arbres de décision, forêts aléatoires, SVM et régression logistique :** Abdennour Messikh
- **Exploration des données, prétraitement des données, sélection des colonnes et division du jeu de données en trois, organisation du code sous forme de fonctions génériques :** Leonard Namolaru
- **Graphes:** Abdennour Messikh
- **Slides (à part les graphes):** Leonard Namolaru

Merci!

CREDITS: This presentation template was created by [Slidesgo](#), and includes icons by [Flaticon](#) and infographics & images by [Freepik](#)

