MINISTÈRE
DES ARMÉES
Liberté
Égalité
Fraternité

Hackaton - IA vs AI

– Rapport Final –

Clovis **Lechien**
Léa **Margery**
Loris **LIN**
Robin **LEVASSEUR**
Alexandre **MOUTON-BISTONDI**

25 Avril 2024

Table des matières

1 Projet & Outils

1.1 Découverte du sujet

Cette phase du projet nous a offert une précieuse opportunité de mieux nous connaître et de renforcer nos liens d'équipe. L'amélioration de notre cohésion a directement contribué à une répartition des tâches plus efficace et harmonieuse.
En nous appuyant sur nos compétences individuelles, nous avons formé différents groupes de travail, permettant ainsi à chacun de valoriser au mieux son potentiel dans la réalisation de nos objectifs communs.

1.2 Métriques de performances

Nous avons choisi d'utiliser différentes métriques de performances afin de mesurer la qualité de nos modèles.
Le **F1-Score** nous a permis de comprendre et mesurer les performances de nos différents modèles en machine learning. Ce score permet de quantifier la proportion de réponses correctes avancées par le modèle pendant son entraînement.
Pour notre modèle de deep-learning, nous avons choisi de vérifier ses performances à l'aide de la **train accuracy** et de la **validation accuracy**. Ils permettent tous deux de donner un aperçu de la validité des réponses fournies par le modèle, que ce soit pendant l'entraînement ou pendant la phase de validation.
Enfin, pour tous nos modèles nous avons aussi utilisé le **R2-Score** afin de mesurer cette fois-ci le sur-apprentissage de nos modèles. Un score élevé de ce-dernier nous permet de vérifier que notre modèle généralise bien et ne s'est pas sur-entraîné sur notre jeu de données.
Un modèle sur-entraîné est un modèle qui réussit bien la tâche qui lui est attribué sur les données du dataset d'entraînement mais qui ne généralise pas bien sur d'autres données.

1.3 Onyxia

L'équipe a fait le choix stratégique d'intégrer Onyxia, une plateforme de collaboration de données de l'INSEE, afin de simplifier le partage et la manipulation des données.
Le choix de rester sur cet environnement a été alimenté par la disponibilité des données dans un premier temps et à l'accès à un environnement de développement capable de répondre à nos besoins.
Cette décision a donc été suivie d'une rapide prise en main collective de la plateforme pour assurer une transition fluide vers son utilisation optimale.

1.4 Python

Nous avons utilisé de nombreuses librairies intégrées directement à Onyxia pour la plupart. En voici quelques unes :

— pandas
— numpy
— scikit-learn
— tiktoken
— spacy
— dash

L'utilisation de python en général était naturelle étant donnée l'abondance de ressources liées à l'intelligence artificielle sur ce langage.

2 Analyse des jeux de données

2.1 HC3

	question	human_answers	chatgpt_answers
0	Historical P/E ratios of small-cap vs. large-c...	[There is most likely an error in the WSJ's da...	[Historical price-to-earnings (P/E) ratios for...
1	Should you co-sign a personal loan for a frien...	[I know this question has a lot of answers alr...	[Co-signing a personal loan for a friend or fa...
2	Should I avoid credit card use to improve our ...	[If you pay it off before the cycle closes it ...	[It can be a good idea to avoid using credit c...
3	Difference between 'split and redemption' of s...	[It is the first time I encounter redemption p...	[Share split and redemption are two different ...
4	Pros & cons of investing in gold vs. platinum?	[Why Investors Buy Platinum is an old (1995) a...	[Gold and platinum are both precious metals th...

FIGURE 1 – Premier jeu de données HC3

Le premier jeu de données qui nous a été fourni comportait un faible nombre de données (7210 lignes) et était incomplet.
Les questions et réponses issues de la source 'reddit_eli5ónt été retirées, car les données avaient été préalablement rendues illisibles par le créateur du jeu de données, ce qui nous a amené à supposer qu'elles pouvaient être faussées.
Pour utiliser ce jeu de données nous avons du mettre en place quelques étapes de data processing afin de le rendre utilisable par notre premier modèle au début du hackaton.

2.2 DeepfakeTextDetect

Le second jeu de données, sur lequel nous avons travaillé au cours du reste de la compétition, était bien plus grand (56819 lignes) et possédait une répartition des sources nettement plus importante (322 sources différentes).
Désormais, au lieu d'avoir 3 colonnes avec la question et les deux réponses nous n'avons plus qu'un seule colonne qui comporte le texte à classifier.

	text	label	src
0	Little disclaimer: this deals with US laws and...	1	cmv_human
1	Read: Mentally Retarded Downs. See, we've got ...	1	cmv_human
2	If any of you frequent rbadhistory, there is a...	1	cmv_human
3	I believe in a flat tax system, where everyone...	1	cmv_human
4	Edit: Ok guy's, my views have been changed on ...	1	cmv_human
...	...	...	...
56814	We consider the recovery of a source term f (x...	1	sci_gen_human
56815	Self-supervised learning (SlfSL), aiming at le...	1	sci_gen_human
56816	Recurrent neural networks (RNNs) have achieved...	1	sci_gen_human
56817	Deep reinforcement learning (DRL) is a booming...	1	sci_gen_human
56818	As part of Smart Cities initiatives, national,...	1	sci_gen_human

56819 rows × 3 columns

FIGURE 2 – Nouveau jeu de données hack train

Cela facilite les traitements à appliquer. De manière générale ce jeu de données était quasiment utilisable pour la création de nos modèles.
Nous avons vite remarqué que ce-dernier était presque parfaitement équilibré en nombre de source entre les humains et les IAs, donc encore une fois aucun rééquilibrage n'a été fait.
Par contre, les sources étaient bien plus nombreuses et très fortement déséquilibrées (3,1% des sources sont sur des textes humains contre 96,9% pour les IAs). Ce n'est pas un problème tant que le nombre total de données et bien équilibré entre les humains et les IAs, et c'est le cas.
La distribution sur la longueur des textes révèle une concentration des données principalement entre 0 et 1000 caractères.
Enfin, on remarque que les 3 mots les plus courants pour les IAs et les Humains sont relativement proches :

Principaux mots :

— said => (humains & IAs)
— one => (humains & IAs)
— people / time => (humains / IAs)

3 Pré-processing et formatage des données

3.1 HC3

	question	answers	target
0	Dow Jones Industrial Average (DJIA), NASDAQ 10...	The Dow Jones Industrial Average (DJIA) is a s...	1
1	Can a husband and wife who are both members of...	Since from the question it seems that you're t...	0
2	how is jerky made	Jerky is a type of dried, cured meat that is u...	1
3	How can OTC scams affect you?	Am I being absurd? No. Should I be worrying? Y...	0
4	Uncashed paycheck 13 years old	It is not uncommon for people to have uncashed...	1
...	...	...	...
14415	Can one use Google Finance to backtest (i.e. s...	Yes, you can use Google Finance to backtest tr...	1
14416	How to determine the metal used during an earl...	It is generally safe to undergo an MRI after o...	1
14417	Who are the sellers for the new public stocks?	When a company goes public, it issues shares o...	1
14418	What could cause recurred cough with sensation...	There are several possible causes of a recurre...	1
14419	What are the treatment options for having ston...	There are several treatment options for a kidn...	1

14420 rows × 3 columns

FIGURE 3 – Pré-traitement du premier jeu de données

Nous avons commencé par analyser ce premier jeu de données avant de faire un quelconque traitement pour comprendre sa structure et la répartition des données.
Cette phase nous a permis de comprendre les différents traitements à effectuer.
Nous avons donc commencé par ajouter une colone "target" qui permet au modèle de vérifier la prédiction choisie (1 ou 0 pour les humains et IAs respectivement).
Pour ce faire nous avons dédoublé les données des réponses et les avons regroupé sous une seule colonne "answers".
Nous avions choisi de garder la colonne "question" pour faire de l'aug-

mentation de données (feature engineering) dessus et enrichir notre jeu de données d'entraînement.
Sans les réponses provenant de la source "reddit_eli5", le jeu de données était très bien équilibré et ne nécessitait pas de suppression de données ou de rééquilibrage.
Enfin, avons tokenisé et vectorisé les textes présents dans la colonne "answers".

3.2 DeepfakeTextDetect

Après analyse du second jeu de données, nous avons remarqué que peu de changements étaient à faire sur cette nouvelle source.
Nous avons simplement tokenisé et vectorisé les textes présents dans la colonne "text", comme nous avons pu le faire pour le jeu de données antérieur.

4 Modèle Optimal

4.1 Régression Logistique

La régression logistique est un algorithme d'apprentissage supervisé pour la classification et la prédictions de variables binaires ou catégorielles. Il s'agit d'un des modèles les plus simples de classification d'apprentissage supervisé.

Elle repose sur des principes mathématiques pour établir des liens entre un ensemble de données X et les différentes catégories de classe Y.

Elle est particulièment adaptée à notre cas d'usage en raison de sa capacité à effectuer une classification binaire : attribuer la valeur 1 si la réponse est humaine, ou 0 si la réponse est générée par GPT.

Concernant l'implémentation du modèle, nous avons choisi d'utiliser la fonction "CountVectorizer" de la bibliothèque sklearn pour transformer nos données en vecteurs. En effet la fréquence d'apparition de certains mots présente une différence significative entre les réponses humaines et celles générées par GPT, ainsi que des variations syntaxiques.

Avec le contexte du projet, nous avons donc choisi de prendre le "gpt_tokenizer" de la bibliothèque sklearn et en utilisant la recherche sur grille, nous avons exploré différentes valeurs pour le paramètre ngram_range, car ce paramètre permet de capturer le contexte d'une séquence de mots.

4.2 Hyperparamètres

Pour obtenir des performances optimales en utilisant le modèle de la régression logistique, il est primordial de passer par une étape cruciale : l'optimisation des hyperparamètres.

Dans l'apprentissage automatique, un hyperparamètre est un paramètre dont la valeur est utilisée pour contrôler le processus d'apprentissage.

Pour ce, nous avons tenté d'utiliser la méthode bien connue de la recherche sur grille (GridSearch) pour ajuster les hyperparamètres de la régression logistique tels que le "penalty" et le "solver".

Cependant, ces hyperparamètres interagissent de manière complexes, par conséquent, nous avons été contraints d'explorer manuellement toutes les combinaisons possibles pour avoir la configurations optimales.

Résultats obtenus :

— penalty : l2
— solver : liblinear
— ngram_range : (3, 3)

4.3 Résultats

Ce modèle a été le premier que nous avons implémenté sur le premier jeu de données et les résultats obtenus étaient très satisfaisants. C'est pourquoi, après avoir reçu le second jeu de données nous avons tout d'abord essayer de réutiliser ce modèle. Nous voulions déterminer si ce modèle là fonctionnerait tout aussi bien sur le nouveau jeu de données.

Il s'est avéré que cela a été le cas, nous avons alors concentré nos efforts sur l'amélioration de ses performances, tout en essayant en parallèle de nouveaux modèles afin de les comparer.

Premier jeu de données :

— F1 Score : 0,98
— R2 Score : 0,91
— Temps d'entraînement : 30 secondes

Second jeu de données :
— F1 Score : 0,95
— R2 Score : 0,79
— Temps d'entraînement : 3 minutes

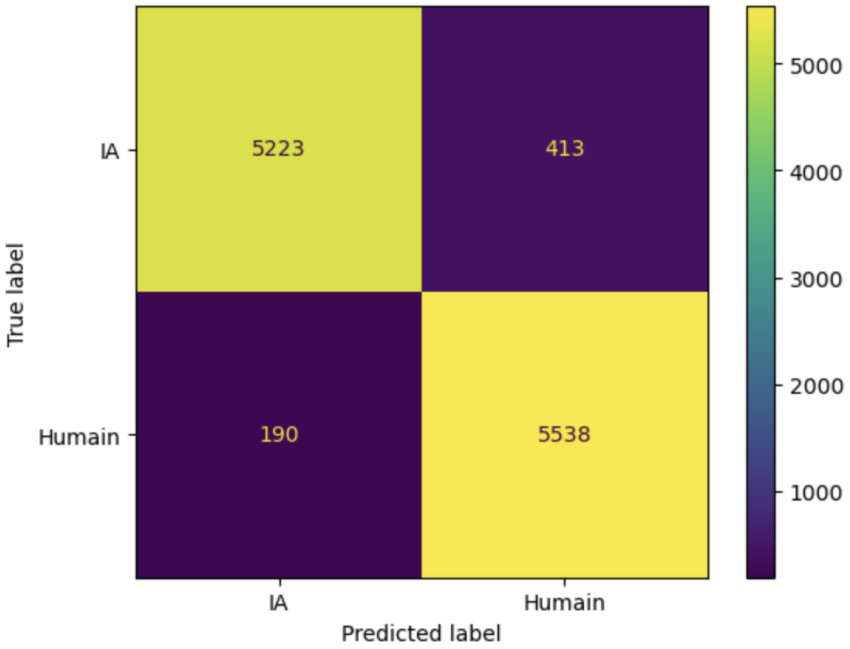

FIGURE 4 – Matrice de confusion Logistique

5 Autres pistes explorées

Après avoir exploré ce qu'il était possible de réaliser avec la régression logistique, nous nous sommes penchés sur différents modèles en espérant trouver une meilleure solution.

5.1 Apprentissage Automatisé

5.1.1 K-voisins les Plus Proches

Ce modèle, également connu sous le nom de KNN ou k-NN, repose sur la proximité des données pour effectuer une classification ou des prédictions sur le regroupement d'un point de données individuel.

En pratique, cela signifie que les données sont placées dans un espace à plusieurs dimensions, dépendant du nombre de caractéristiques, et que les étiquettes sont attribuées à un ensemble de points qui sont ensuite regroupés en fonction de leur proximité dans cet espace.

Résultats obtenus :
— F1 Score : 0,91
— R2 Score : 0,64
— Temps d'entraînement : 1.5 minutes

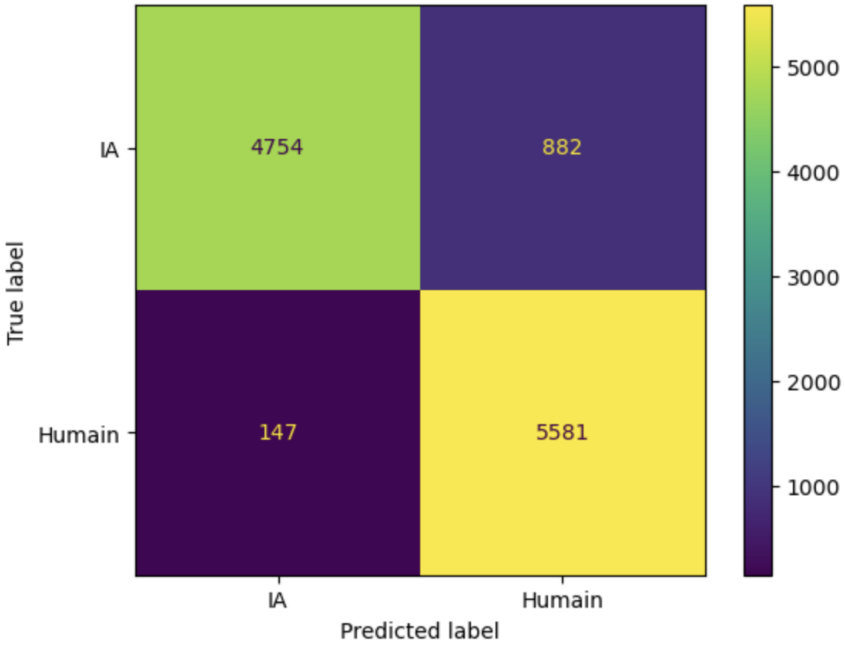

FIGURE 5 – Matrice de confusion Knn

5.1.2 Classificateur Naïf Bayésien

Ce modèle se base sur le théorème de Bayes avec une hypothèse naïve selon laquelle les caractéristiques sont indépendantes les une des autres. L'objectif est d'estimer la probabilité qu'une donnée appartienne à une certaine classe en se basant sur des caractéristiques données.

Ainsi, en disposant de certaines caractéristiques, le modèle choisira la classe ayant la probabilité la plus élevée en appliquant les probabilités conditionnelles.

Résultats obtenus :
— F1 Score : 0,83
— R2 Score : 0,30

— Temps d'entraînement : 1 minutes

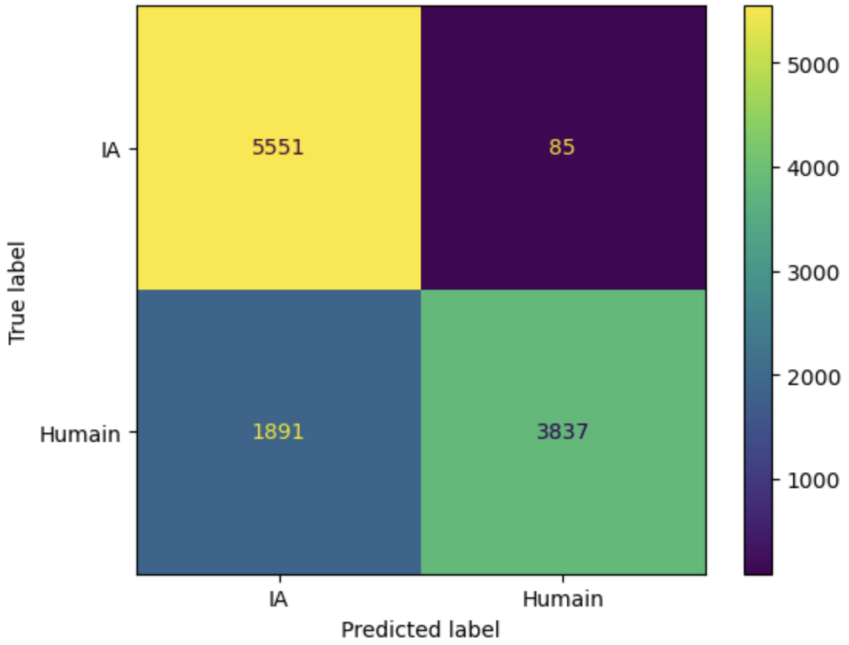

FIGURE 6 – Matrice de confusion Naïf Bayésien

5.1.3 Forêt d'arbres décisionnels

Ce modèle repose sur des arbres que l'on appelle arbre de décision ou arbre décisionnel. Ces arbres permettre de prendre une décision grâce à une série de questions (aussi appelées tests) dont la réponse (oui/non) mènera à la décision finale.

Tout d'abord, nous nous sommes tournés vers un algorithme d'arbres décisionnels mais nous nous sommes rapidement rendus compte qu'il existait un autre modèle dans le même genre (la forêt d'arbres décisionnels), qui en était une version améliorée.

En effet, la forêt d'abres décisionnels est plus rapide, plus précise et

résiste au problème de sur-apprentissage, contrairement à l'algorithme d'arbres décisionnels.

La première version de ce modèle avait une profondeur de 20 et était constituée de 100 arbres décisionnels. Cependant, les résultats attendus étaient de piètre qualité. Nous avons donc petit à petit augmenté la profondeur du modèle jusqu'à arriver à une profondeur de 100. En revanche cette augmentation de la profondeur du modèle causait une augmentation de la durée nécessaire à l'entraînement de celui-ci.

Résultats obtenus :

— F1 Score : 0,88
— R2 Score : 0,53
— Temps d'entraînement : 4 minutes

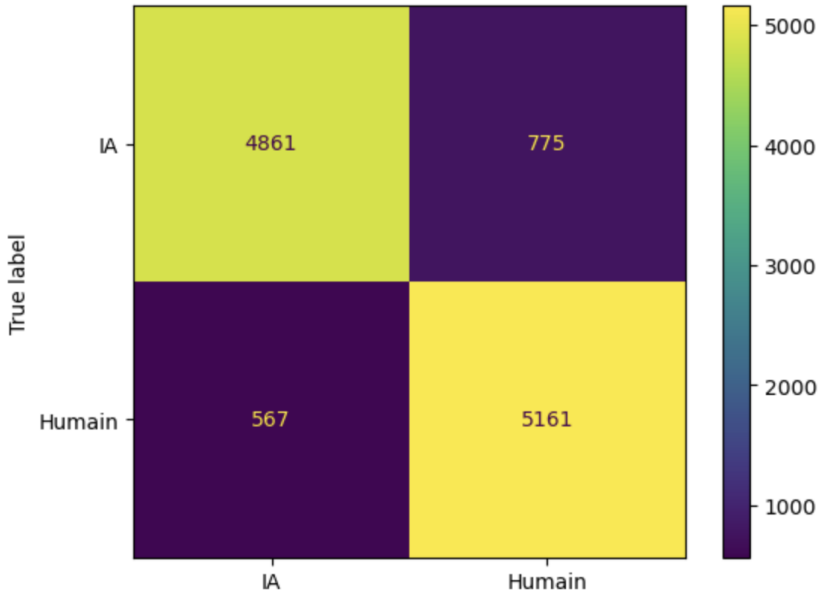

FIGURE 7 – Matrice de confusion Forêt d'arbres décisionnels

5.1.4 Machine à vecteurs de support (SVM)

Ensuite, nous avons essayé d'entraîner un SVM. C'est un modèle qui permet de gérer des données complexes (non linéaires), qui a une bonne généralisation sur de nouvelles données et réduit le risque de sur-apprentissage.

L'objectif principal des SVM est de trouver un hyperplan dans un espace qui sépare au mieux les exemples des différentes classes. Celui-ci est choisi de manière à maximiser la distance entre l'hyperplan et les exemples les plus proches de chaque classe.

En revanche, ce modèle est très long à entraîner. En effet, nous avons attendu la fin de l'entraînement pendant plus de 30 min sans parvenir à un résultat. Puisque l'entraînement du modèle prenait beaucoup de temps, nous ne nous sommes pas attardés sur l'entraînement de SVM et avons préféré explorer de nouvelles pistes.

Résultats obtenus :
— F1 Score : pas de résultats obtenus
— R2 Score : pas de résultats obtenus
— Temps d'entraînement : supérieur à 30 minutes

5.2 Apprentissage Profond

5.2.1 Bert fine-tuning

Nous avons pu étendre l'horizon des solutions testées jusqu'au deep learning, en faisant un fine-tuning du "LLM" (Large Language Model) Bert sur nos données.
Celui-ci nous a donné des résultats très satisfaisants, cependant nous avons choisi de ne pas prendre cette solution pour répondre à la problématique.
En effet, comparé à la solution retenue, les coûts pour mettre en place

ce fine-tuning sont très conséquents pour des améliorations quasi insignifiantes.
L'utilisation de GPUs (plus chers et donc moins facilement scalable) et le temps d'éxecution important sont des facteurs majeurs ayant facilités la décision.

6 Benchmark

Benchmark			
	PRÉCISION (ACCURACY)	TEMPS D'ÉXECUTION	SCORE R2
K VOISINS LES PLUS PROCHES	0.91	1.5 minutes	0.64
RÉGRESSION LOGISTIQUE	0.95	3 minutes	0.79
CLASSIFICATEUR NAÏF BAYÉSIEN	0.83	1 minutes	0.30
FORÊT D'ARBRES DÉCISIONNELS	0.88	4 minutes	0.53
MACHINE À VECTEURS DE SUPPORT (SVM)	X	> 30 minutes	X
BERT FINE-TUNING	0.94	40 minutes	0.80

FIGURE 8 – Tableau comparatif des performances

7 Difficultées rencontrées

Au cours du projet, nous avons rencontré certaines difficultés à commencer par la prise en main de la plateforme Onyxia, et l'ouverture du premier dataset qui présentait des problèmes de formatage. Par la suite, nous avons été ralenti par une stagnation des performances de notre modèle que nous estimions insatisfaisantes (0,86). Enfin, l'élaboration de la solution de deeplearning a également été compliquée notamment à cause d'Onyxia, raison pour laquelle nous l'avons finalement implantée en local depuis Kaggle, tout comme le frontend de notre modèle.

8 Conclusion

Force est de constater que ces trois jours de hackathon furent une occasion unique d'agrandir nos compétences dans les secteurs de l'IA, de la data mais également en management de projet.
En effet, après avoir fait connaissance de notre équipe, nous avons rapidement commencé l'analyse du dataset ainsi que le pré-processing des données. Puis, l'élaboration de différents modèles de machine et deep learning nous a permis de déterminer le meilleur rapport entre la performance et les couts : une technologie de régression logistique.
Avec un taux de précision de 95% notre modèle est donc capable de faire la différence entre un text rédigé par un humain et un autre généré par une intelligence artificielle.

9 Ouverture

Au cours de notre analyse, nous avons constamment évalué l'utilité potentielle de notre modèle et son adéquation avec les intérêts du ministère des armées. Cette réflexion nous a menés à deux conclusions principales :
Pour la partie recrutement d'une part, notre modèle peut être utilisé pour authentifier les lettres de motivation dans les processus de recrutement, en identifiant celles générées artificiellement. De plus, il permet de détecter les candidatures automatisées, évitant ainsi une sur-représentation de certains profils et garantissant une sélection de candidats plus équitable et transparente.
Nous avons aussi estimé que notre modèle pourrait apporter un soutien essentiel dans le filtrage des informations entrantes pour prévenir la désinformation.

10 Sources

Le lien github du projet :
`https://github.com/ClovisDyArx/hackaton-ai_vs_ia`
Le lien des slides du pitch technique :
`https://www.canva.com/design/DAGDZ_3-c04/QhwaodEiSZDES6WuvMp3PQ/edit?utm_content=DAGDZ_3-c04&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton`
Le lien des slides du pitch final :
`https://www.canva.com/design/DAGDPPO4NG0/AfXMtszmcckzOUvezkNyjw/edit?utm_content=DAGDPPO4NG0&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton`
Scikit-learn :
`https://scikit-learn.org/stable/`
Dash Plotly :
`https://dash.plotly.com/`
Onyxia :
`https://datalab.sspcloud.fr/`