Développement et amélioration de la qualité d'un service web contenant une fonctionnalité d'intelligence artificielle
-
Rapport Intermédiaire

Carrio Florian

[1] Université Grenoble-Alpes, France
[2] **L**aboratoire **I**nformatique de **G**renoble, France
[3] Graduate School of System Informatics - Kobe University, Japan

Abstract. Durant les dernières décennies, les techniques intelligence artificielle et plus particulièrement le deep-learning ont rencontré un succès sans précédent, en découlent une forte utilisation et popularité dans de nombreux domaines très variés. Malheureusement, persiste une "boîte noire magique" nécessaire se cachant derrière ces systèmes, notamment dans les techniques d'apprentissage dit "profond", conséquence de cela : un cruel problème de compréhension et d'interprétation de ces systèmes. Plus globalement, le manque de qualité dans ces applications se fait clairement ressentir, notamment en termes de sécurité, en découle ainsi une demande de plus en plus urgente pour l'innovation de nouvelles techniques rigoureuses pour améliorer la qualité logicielle de ces systèmes. Dans ce rapport nous n'aurons pas la prétention de résoudre tous ces problèmes, mais nous allons réaliser une ébauche non terminée à l'heure actuelle pour essayer d'apporter un début de solution à l'aide d'une passerelle formée par le travail sur la validation de ces systèmes intelligents par des chercheurs du LIG et le développement de ces systèmes et l'implantation de ces méthodologies ici au Japon.

Keywords: Intelligence Artificielle · Machine Learning · Génie Logiciel · Qualité .

1 Introduction

Afin de toujours mieux satisfaire les utilisateurs, de réduire les coûts et surtout de mieux maîtriser nos systèmes il n'est plus négligeable de ne pas valider nos applications avant que celles-ci soient mises en production. Raison de plus si elles utilisent de l'intelligence artificielle, fonctionnalité très versatile durant l'exécution dont il est difficile de prédire l'évolution. Il est donc fondamental de prouver de manière cohérente que ce genre de système réponde de manière cohérente à un ensemble de conditions prédéfinies. Ce qui garantira l'exactitude, la fiabilité, la performance attendue, la capacité à discerner des enregistrements

altérés et donc la validité du système.
Pour cela nous allons évaluer une approche de validation basée sur une émission de traces par l'application même pour nous appuyer sur un ensemble de règles et propriétés à valider sur ses traces même.
De plus, la sécurité étant bien trop souvent négligée, nous nous pencherons également sur celle-ci dans nos travaux.
C'est ainsi que dans la suite de cet article, nous présenterons le principe de validation tout juste énoncé à l'aide de ParTrap.
Puis nous aborderons le développement d'une application utilisant le machine learning pour améliorer sa qualité, par la suite nous aborderons le thème de sécurité pour enfin aboutir sur la validation de notre système avant de conclure sur l'ensemble de nos recherches.

2 PartTraP

De par la thèse de Yoann Blein, créateur de ce langage PartTraP, et l'article [8] décrivant ce même langage et sa prise en main nous pouvons définir Partrap comme étant un langage formel déclaratif très accessible nous permettant d'exprimer aisément des propriétés et règles paramétriques, cela à partir de traces finies fournies par l'application elle-même au format JSON et donc à l'exécution de celle-ci.
Cela nous permet ainsi de vérifier au fur et à mesure de l'exécution d'une application si celle-ci évolue ou pas dans le sens souhaité. Ces traces, peuvent être obtenues par exemple, dans le cadre d'une application développée en C, par une librairie très utile nommée Log4C [4], copie presque parfaite de Log4J pour le langage C. Ainsi grâce à celles-ci nous pouvons rédiger à l'aide de Partrap des règles de sémantiques à respecter afin de toujours avoir une intelligence cohérente, et dans le cas contraire, plus facilement comprendre d'où peuvent provenir les problèmes en retraçant l'évolution de celle-ci par exemple. Pour illustrer le fonctionnement de Partrap, prenons l'exemple d'une application de SAV pour une entreprise de mobilier. Cette application consiste à la mise en relation de clients avec des agents du service après-vente pour une réclamation ou toutes autres requêtes.
Ainsi, les utilisateurs récupèrent un ticket via cette application en précisant le but de leur visite. Puis l'application va appeler les utilisateurs au fur et à mesure, en fonction de leurs places dans la file d'attente.
Très naturellement, on attend certaines propriétés de cette application :

- Chaque client doit passer dans l'ordre de la file d'attente
- Une requête doit être formulée
- La requête formulée doit pouvoir être gérer par les agents
- Le produit doit se conformer au service demandé

Pour vérifier si l'application respecte ces propriétés informelles, on utilise la librairie log4J pour tracer les comportements importants. On choisit en particulier de tracer l'identifiant de la fonction appelée, le timestamp à ce moment

là, son résultat correspondant au service du SAV choisit mais aussi l'identifiant du produit défaillant. On exécute l'application et on obtient alors une trace qui retrace le comportement de l'application.

{"id": "ticket", "method":"takeTicket", "productNumber":0018, "timestamp":1555082490, "result":technical }

{"id": "ticket", "method":"takeTicket", "productNumber":0004, "timestamp":1555087897, "result":financial }

{"state": "startHandleClient", "productNumber" : "0018", "id":"Florian", "timestamp":"1555082492"}

{"state": "finishHandleClient", "productNumber" : "0018", "id":"Florian", "timestamp":"1555082910"}

{"state": "startHandleClient", "productNumber" : "0004", "id":"Lucas", "timestamp":"1555082915"}

{"state": "finishHandleClient", "productNumber" : "0004","id":"Lucas", "timestamp":"1555083790"}

Ensuite, on exprime en Partrap les propriétés informelles décrites ci-dessus.

```
1 GoodTicket: absence_of Ticket t where t.productNumbler == "" && result == "";
2
3 OneByOneClient: startHandleClient s followed_by finishHandleClient f where s.productNumber == f.productNumber;
```

Fig. 1. Propriétés ParTraP écrites pour l'exemple du SAV

Nous pouvons voir ici 2 propriétés illustrant nos propos récupérer dans le code de notre programme ParTraP qui se vérifie au cours de l'analyse des traces au format JSON fournie par l'application.
La première assez triviale, vérifiant simplement que le ticket de SAV pris par le client contient bel et bien des caractères, nous aurions pu vérifier également que la requête est bel et bien formulée en français pour une entreprise française par exemple.
Concernant la deuxième, le SAV ne pouvant s'occuper que d'un client à la fois, dans le cas d'une application multi-threadée par exemple, cette propriété vérifie ainsi que tous les clients sont traités l'un après l'autre. Il est également possible ici de vérifier à l'aide du timestamp lors de l'émission du ticket, que le client traité est bien celui ayant le plus petit timestamp.

Enfin, on évalue les propriétés avec l'environnement attaché à Partrap.

Maintenant, il est de notre devoir de mettre en place cet outil à travers nos travaux réalisés au Japon.

3 Développement d'une application utilisant Microsoft Azure

Microsoft Azure est une plate-forme applicative (Platform as a Service ou PaaS) en cloud-computing, offrant à travers une interface très simplifiée et facile d'utilisation de nombreux services classiques mais aussi l'opportunité de créer des solutions analytiques prédictives grâce au Azure Machine Learning Studio qui est le service qui nous intéresse.

Nous allons ainsi illustrer le principe de Microsoft Azure en ayant au préalable choisit un ensemble de données probant, par la suite sera déployer l'application utilisant le web-service d'Azure.

3.1 India's Student Performance Prediction

À partir des données récupérées sur le site de l'UCI [1] très utilisé ici au Japon, le but de notre application est de prédire si un nouvel étudiant souhaitant rentrer dans une université indienne sera capable de réussir son année.
Pour cela il est nécessaire de fournir de nombreuses informations.
Ces données sont toutes celles utilisées par notre set de données, c'est de cette manière que notre futur modèle pourra faire sa prédiction.
Cette base de données récupérée est décomposée en 22 attributs concernant l'élève et son entourage, ainsi que son passé scolaire sous le format ARFF[2], accepté par Azure et utilisant le modèle Attribut-Type pour définir les différentes variables censées composer les différentes colonnes d'un tuple.

S.No	Attribute	Description	Values
1	GE	Gender	Male, Female
2	CST	Caste	General,SC,ST,OBC,MOBC
3	TNP	Class X Percentage	Best, Very Good, Good, Pass, Fail
4	TWP	Class XII Percentage	Best, Very Good, Good, Pass, Fail
5	IAP	Internal Assessment Percentage	Best, Very Good, Good, Pass, Fail
6	ESP	End Semester Percentage	Best, Very Good, Good, Pass, Fail
7	ARR	Whether the student has back or arrear papers	Yes, No
8	MS	Marital Status	Married, Unmarried
9	LS	Lived in Town or Village	Town, Village
10	AS	Admission Category	Free, Paid
11	FMI	Family Monthly Income in INR	Very High, High, Above Medium, Medium, Low
12	FS	Family Size	Large, Average, Small
13	FQ	Father Qualification	IL, UM, 10, 12 , Degree, PG (IL= Illiterate UM= Under Class X)
14	MQ	Mother Qualification	IL, UM, 10, 12 , Degree, PG
15	FO	Father Occupation	Service, Business, Retired, Farmer, Others
16	MO	Mother Occupation	Service, Business, Retired, Farmer, Others
17	NF	Number of Friends	Large, Average, Small
18	SH	Study Hours	Good, Average, Poor
19	SS	Student School attended at Class X level	Govt., Private
20	ME	Medium	Eng,Asm,Hin,Ben
21	TT	Home to College Travel Time	Large, Average, Small
22	ATD	Class Attendance Percentage	Good, Average, Poor

Fig. 2. Attributs du Dataset des étudiants Indiens

L'étape fondamentale ici fut de choisir le bon algorithme de Machine Learning. De nombreux algorithmes pour le Machine Learning existent en fonction des valeurs traitées et du résultat voulu, mais 2 concepts principaux en sont à l'origine comme expliqué ci-après.

1. Concept de Classification
   Pour attribuer une classe ou une catégorie parmi un ensemble fini d'éléments.
2. Concept de régression
   Ici la prédiction sera directement une valeur numérique.

L'apprentissage quant à lui peut s'effectuer de manière supervisée, grâce à des données étiquetées afin de déterminer la classe des futures données injectées dans l'application.

À l'inverse, existe aussi l'apprentissage non supervisé où l'algorithme devra repérer des motifs communs afin de constituer des groupes homogènes pour attribuer aux nouvelles données entrantes un groupe labellisé par nos propres observations.

De nombreux algorithmes découlent de ces 2 concepts, ici nous nous focaliseront sur les algorithmes traitant de données d'ores et déjà classées comme c'est le cas pour notre ensemble.

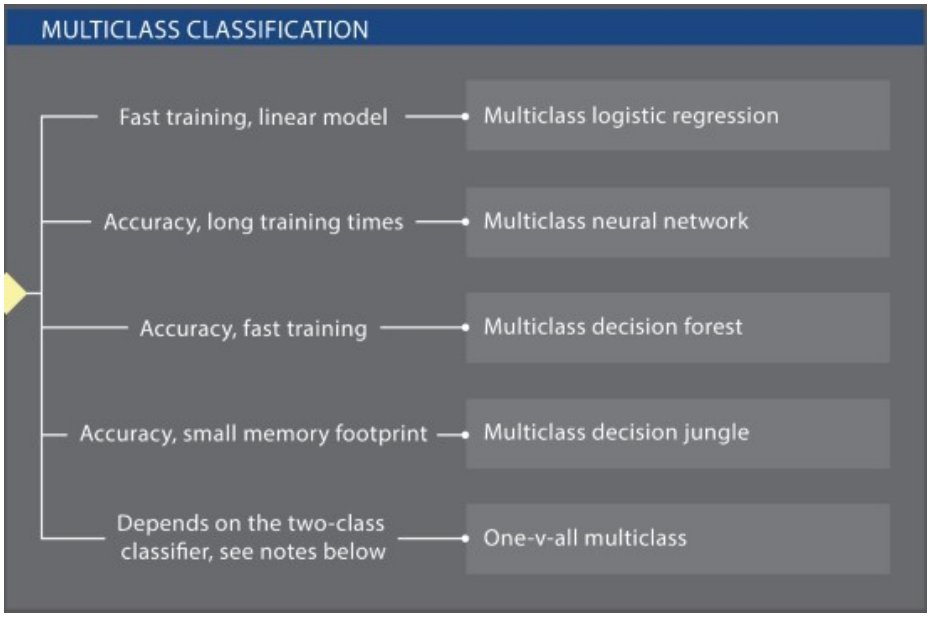

Fig. 3. Principaux algorithmes de Machine Learning de Classification

Parmi ces algorithmes figure l'algorithme dit de "L'arbre de décision", servant à classifier plusieurs observations à partir d'un corpus d'observations déjà étiqueté. Nous parcourons ainsi l'arbre, où, à chaque nœud correspond à une propriété à vérifier(exemple : l'élève a eu une bonne moyenne générale) ce qui aura pour effet sans doutes d'augmenter les chances de l'étudiant de réussir sa future année.

Existe aussi l'algorithme dit des "Forêts aléatoires", très proche de l'algorithme précédent, sauf qu'il réserve un traitement beaucoup plus particulier aux tuples présents dans le dataset qui se rapproche le plus des informations fournies pour votre prédiction désirée.

C'est en essayant plusieurs de ces algorithmes comme illustrer ci-dessous à travers le graphique joint, que nous avons obtenu les meilleurs résultats avec ce dernier.

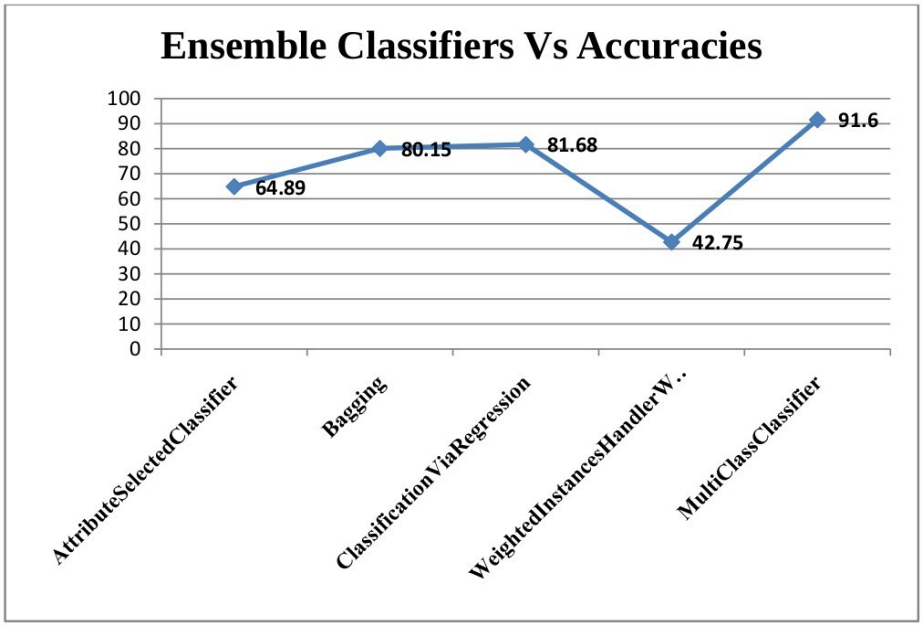

Fig. 4. Comparaison de la précision des algorithmes de Machine Learning de Classification

C'est donc cet algorithme que nous allons utiliser, mais il existe de nombreux autres algorithmes pour des cas plus spécifiques, comme le "Grandient Boosting" très utile lorsque beaucoup de données sont manquantes.

3.2 Modèle Azure

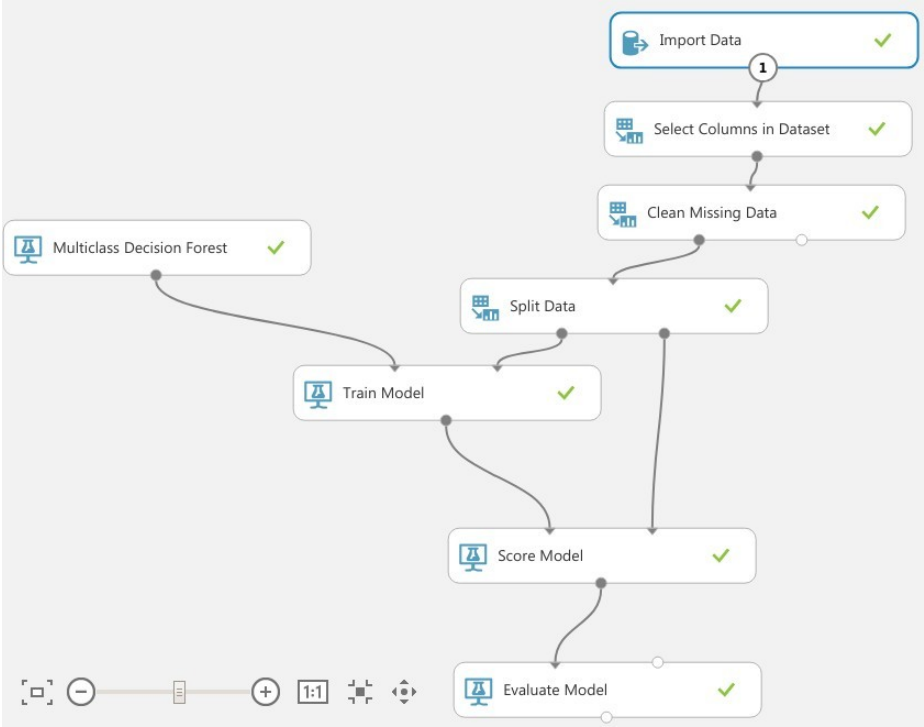

Fig. 5. Modèle Microsoft Azure Studio Machine Learning

Nous pouvons observer ci-dessus les différentes étapes nécessaires à la création d'un modèle prédictif pour notre application. Premièrement l'importation des données s'effectue grâce au fichier ARFF, puis il est important de sélectionner uniquement les attributs pertinents pour notre prédiction, par exemple, un attribut spécifie si l'étudiant est marié ou pas, ce qui n'est jamais le cas, la non-sélection de cet attribut est alors de mise. Enfin s'effectue le nettoyage des rares données manquantes altérant notre prédiction. Deuxièmement nous avons séparé les données à hauteur de 75%-25% uniquement pour produire le modèle de score qui nous permet de prédire à l'aide du modèle évaluant, l'attribut voulu pour les 25% de données uniquement dans le but de vérifier que l'algorithme choisi obtient un bon indice de confiance et donc prédit la valeur exacte.
En dernier lieu nous créons le web service nous fournissant ainsi un code squelette en python pour effectuer la prédiction à l'aide d'une clef API nous permettant d'utiliser notre modèle à travers le service fourni par Microsoft Azure.

3.3 Interface Python

De par le programme précédent fourni par Azure nous avons créé une application Python afin d'envoyer les données nécessaires pour la prédiction et surtout pour émettre des traces à analyser plus tard avec ParTraP. Nous avons opté pour une interface graphique avec d'augmenter la facilité d'utilisation du programme avec une liste de choix à envoyer pour que le modèle traite la demande, afin d'éviter de nombreux problèmes, de sécurité notamment que nous allons aborder.

4 Sécurité

On s'intéresse ici aux problèmes de sécurité qui peuvent se poser lors du développement et du déploiement d'une application sous la forme d'un Web Service.

4.1 Investigation des différents risques et menaces

Afin d'être les plus exhaustifs possible dans nos recherches, nous avons recherché tous les types de menaces envisageables pour nos applications à travers les différents modules qu'elles proposent. Dans un premier lieu, en dehors de toutes attaques possibles, il faut s'assurer de la compatibilité des données, qui diffèrent en fonction de nos datasets. Tout en restant du coté utilisateur et son interaction avec notre application, il faut que celui-ci envoie le bon nombre de valeurs, pour les bonnes colonnes, ce qui reste trivial.
De plus il faut bien évidemment vérifiée que la connexion internet peut bel et bien s'établir entre l'utilisateur et l'application.

Viens alors les nombreux types d'attaques pouvant survenir :

1. Attaque dite par substitutions de l'utilisateur, un autre utilisateur malintentionné se faisant passé pour un élève par exemple envoie de fausses informations afin de faire échouer la prédiction de la victime
2. Attaque de modification des données envoyées par l'utilisateur, par une attaque du milieu [3] par exemple
3. Attaque directe de la base de donnée du dataset afin de fausser toutes les prédictions
4. Attaque par déni de service sur l'application dans le cas d'une application distribuée, ou encore directement sur Azure ou même sur l'utilisateur

La liste est très sûrement non exhaustive mais regroupe les principaux risques que nous allons maintenant essayer de contrer.

4.2 Minimisation des risques

Concernant les données pour l'interaction homme-machine, l'interface graphique déjà programmée me permit d'ores et déjà d'avoir des données correctes lors de l'envoi de sa requête au modèle Azure, en effet chacune des listes box créée à l'aide de la librairie Tkinter [5] de Python m'a permis de proposer à l'utilisateur

le choix limité d'un certain type de variable, et l'obligation d'en choisir une pour un envoi complet et correct de celles-ci. Pour la connexion Internet, une simple requête au site de Microsoft suffit pour vérifier le bon état de marche de celle-ci.

Quant aux différents types d'attaques voici les différentes solutions proposées. Pour l'attaque dite par substitution : l'utilisation d'une API Key personnelle par utilisateur cryptée et connue que par lui-même. Un système d'authentification aurait aussi pu convenir pour cette situation. Pour l'attaque par "Man in the middle", l'utilisation du service proposé par Microsft, Azure Keyvault [7], se révèle très pertinent, il permet en plus de crypter l'api key, de signer et vérifier les messages transmis, octroyant ainsi la confidentialité et l'intégrité recherchée. Nous sommes actuellement en train de travailler dessus.
À propos des attaques par déni de service, pour protéger notre application si celle-ci est distribuée, nous pourrions mettre en place un système de pare-feu identifiant les ip suspectes essayant d'impacter notre service en mettant donc un filtre sur celles-ci, cela demande une analyse en temps réel en amont et des données pour la traçabilité.
Enfin pour éviter toute fuite de données provenant du dataset initialement fourni, en sortant du cadre de microsoft azure qui semble sécurisé, une solution pourrait être le cryptage symétrique de celles-ci.

4.3 Common Weaknesses Enumeration for Python

Dans le but de pousser plus loin nos travaux, j'ai fait le parallèle avec notre module de cybersécurité du premier semestre en recherchant les potentielles failles existantes (CWE[6]), pour le langage Python. J'ai ainsi proposé une liste exhaustive présente en annexes des failles potentiellement présentes dans un programme python, notamment lorsque celui-ci est distribué sur un serveur web.

5 Testabilité et Validation

L'intégralité de ces implémentations de sécurité peut, elles aussi, faire l'objet de traces après l'utilisation de ses différents moyens afin de s'assurer du bon fonctionnement de ces protections mises en place à l'aide de ParTraP.

Ainsi après chaque rajout de données dans le dataset, il peut être possible de vérifier premièrement que comme voulu ci-dessus, les données envoyées soient bien les données reçues. Deuxièmement après une analyse effectuée sur l'algorithme de machine learning choisi en amont, celles-ci aient bien fait évoluer l'intelligence artificielle dans le sens voulu. Et troisièmement que personne n'ait rajouté de données entre-temps dessus faisant évoluer l'algorithme dans un sens contraire à nos attentes.

6 Conclusion

Un des objectifs étant de produire une évaluation d'une approche de validation par traces pour des applications ayant une fonctionnalité d'intelligence artificielle, nous ne sommes pas encore parvenu à cette étape du stage, ce que nous ambitionnons d'effectuer durant la deuxième moitié de notre stage. En revanche, pour ce qui est de la sécurité et du développement d'une application utilisant l'intelligence artificielles, nous avons pu remplir ce devoir, notamment du coté de la thématique de la sécurité qui a été très approfondie ces jusqu'à présent.

Nos perspectives actuelles sont donc de finir cette partie traitant de sécurité pour vite rebondir sur notre but initial étant la validation et la testabilité de fonctionnalités liées au machine learning. En essayant de s'immiscer dans les travaux de Masahide pour essayer d'exprimer des propriétés formelles ou directement en faisant évoluer notre application pour les exprimer sur celles-ci.

7 Remerciements

Je tiens à remercier très particulièrement Lydie Du Bousquet et Philippe Lalanda pour nous avoir accompagné tout le long de notre stage, pour nous avoir aidé autant sur le plan personnel que scolaire ce qui a grandement facilité notre insertion dans ce pays.
Plus globalement, je remercie la région Auvergne Rhône-Alpes, l'université Grenoble-Alpes et le LIG de nous avoir offert l'opportunité de réaliser ce rêve qui est d'aller au pays du soleil levant dans de telles conditions. Je remercie également Latifa Boudiba d'avoir été aussi compréhensive et aimable à nos égards pour la réservation de nos billets d'avion.
Et pour terminer, je remercie notre professeur Masahide Nakamura et toute son équipe pour sa disponibilité, son amabilité et son accompagnement tout du long de notre séjour, qui a aussi grandement facilité notre insertion dans son équipe et son pays.

Merci.

References

1. UCI Machine Learning Database, https://archive.ics.uci.edu/ml/datasets/Student+Academics+Performance#.
2. ARFF File Description, https://www.cs.waikato.ac.nz/ ml/weka/arff.html
3. Man in the middle attaque, Wikipedia https://en.wikipedia.org/wiki/Man-in-the-middle_attack
4. Log4C, Github https://github.com/bmanojlovic/log4c
5. Tkinter Librairie, Python Wiki https://wiki.python.org/moin/TkInter
6. Common Weaknesses Enumeration, CWE https://cwe.mitre.org/
7. Azure Key-Vault for Security, Microsoft Azure https://azure.microsoft.com/fr-fr/services/key-vault/
8. Ansem Ben Cheikh et al. “An Environment for the ParTraP Trace Property-Language (Tool Demonstration)”. In:Runtime Verification - 18th Inter-national Conference, RV 2018, Limassol, Cyprus, November 10-13, 2018,Proceedings. Ed. by Christian Colombo and Martin Leucker. Vol. 11237.Lecture Notes in Computer Science. Springer, 2018, pp. 437–446.doi:10.1007/978-3-030-03769-7_26. https://doi.org/10.1007/978-3-030-03769-7%5C$_2$6

Coverity Coverage for CWE: Python Coverity Software Testing Platform version 2018.12		
CWE	Name	Coverity checker
20	Improper Input Validation	• XSS • PATH_MANIPULATION
21	Pathname Traversal and Equivalence Errors	• PATH_MANIPULATION
22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	• PATH_MANIPULATION
23	Relative Path Traversal	• PATH_MANIPULATION
36	Absolute Path Traversal	• PATH_MANIPULATION
74	Improper Neutralization of Special Elements in Output Used by a Downstream Component ('Injection')	• XSS
78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	• OS_CMD_INJECTION
79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	• XSS
80	Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)	• XSS
82	Improper Neutralization of Script in Attributes of IMG Tags in a Web Page	• XSS
83	Improper Neutralization of Script in Attributes in a Web Page	• XSS
85	Doubled Character XSS Manipulations	• XSS
86	Improper Neutralization of Invalid Characters in Identifiers in Web Pages	• XSS
87	Improper Neutralization of Alternate XSS Syntax	• XSS
89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	• SQLI
94	Improper Control of Generation of Code ('Code Injection')	• NOSQL_QUERY_INJECTION
95	Improper Neutralization of Directives in Dynamically Evaluated Code ('Eval Injection')	• SCRIPT_CODE_INJECTION
209	Information Exposure Through an Error Message	• SENSITIVE_DATA_LEAK
313	Cleartext Sensitive Data in a Database	• SENSITIVE_DATA_LEAK
313	Cleartext Storage in a File or on Disk	• SENSITIVE_DATA_LEAK

Coverity Coverage for CWE: Python Coverity Software Testing Platform version 2018.12		
CWE	Name	Coverity checker
314	Cleartext Storage in the Registry	• SENSITIVE_DATA_LEAK
315	Cleartext Storage of Sensitive Information in a Cookie	• SENSITIVE_DATA_LEAK
317	Cleartext Storage of Sensitive Information in GUI	• SENSITIVE_DATA_LEAK
319	Cleartext Transmission of Sensitive Information	• SENSITIVE_DATA_LEAK
398	Indicator of Poor Code Quality	• COPY_PASTE_ERROR • IDENTICAL_BRANCHES
476	NULL Pointer Dereference	• FORWARD_NULL • REVERSE_INULL
480	Use of Incorrect Operator	• CONSTANT_EXPRESSION_RESULT
502	Deserialization of Untrusted Data	• UNSAFE_DESERIALIZATION
532	Information Exposure Through Log Files	• SENSITIVE_DATA_LEAK
561	Dead Code	• UNREACHABLE • DEADCODE
569	Expression Issues	• CONSTANT_EXPRESSION_RESULT
601	URL Redirection to Untrusted Site ('Open Redirect')	• OPEN_REDIRECT
611	Improper Restriction of XML External Entity Reference ('XXE')	• XML_EXTERNAL_ENTITY
688	Function Call With Incorrect Variable or Reference as Argument	• IDENTIFIER_TYPO
783	Operator Precedence Logic Error	• CONSTANT_EXPRESSION_RESULT

Fig. 6. Faiblesses connues lors de la programmation en Python