



SAINT- JÉRÔME POLYTECHNIQUE

SAINT- JÉRÔME POLYTECHNIC

(SJP5 – Génie Infotronique)

Mémoire de fin d'études en vue de l'obtention du
Diplôme d'ingénieur de Conception

**CONCEPTION ET MISE EN PLACE D'UNE API DE PREDICTION DU RISQUE DE CREDIT BANCAIRE BASEE
SUR LE MACHINE LEARNING**

Par
TENE TALLA CHRIS JORDAN

Stage Numéro :
21-SJP5-GIT-LT-06-02

Effectué au sein de l'entreprise :



Encadré par :

M. EMINI LUCAS
(Tuteur Académique)

M. FOMBA COLLINS
(Tuteur Entreprise)

Année Académique 2020/2021

DEDICACE



**A toute ma famille, la
famille TALLA**

REMERCIEMENT

L'accomplissement effectif de ce travail, ainsi que toutes les leçons que nous en avons tirées n'auraient pu se faire sans la participation de plusieurs personnes. Nous tenons donc à remercier tous ceux qui ont participé de près ou de loin à la rédaction de ce mémoire. Ces remerciements vont particulièrement à l'endroit de :

- L'Administrateur Général M. **Emmanuel POHOWE**, pour sa sagesse.
- L'Institut Universitaire Catholique Saint Jérôme de Douala pour notre formation ;
- Mes mentors, M. **FABO Keven Arcel**, M. **POKA TOUKAM Fritz**, ingénieurs datascientist pour leurs conseils, leurs transferts de connaissances et leur disponibilité.
- M. **EMINI Lucas** notre encadreur académique pour sa disponibilité et ses moyens mis à notre disposition pour une meilleure consistance du travail ;
- Notre encadreur professionnel, M. **FOMBA Collins Ken** pour sa disponibilité et pour tous ses conseils au cours de notre séjour dans son entreprise ;
- A tout le personnel de IWOMI Technologies qui a rendu notre séjour agréable au sein de l'entreprise ;
- Nos parents pour tous les efforts et sacrifices qu'ils ont dû faire pour notre éducation ;
- Nos frères, nos sœurs, nos amis et amies pour leur soutien inconditionnel ;
- Tous nos aînées académiques pour leurs conseils, leur confiance et leur disponibilité ;
- Toute la 5^{ème} promotion de Saint-Jérôme Polytechnique ;
- Toutes les personnes qui de près ou de loin ont contribué à la rédaction de ce mémoire.

LISTE DES TABLEAUX

Tableau 1 Fiche signalétique de l'entreprise IWOMI Technologies	3
Tableau 2 Typologie de problème et objectifs fixés	37
Tableau 3 Type des données	42

LISTE DES FIGURES

Figure 1 Organigramme de IWOMI Technologies [10].....	4
Figure 2 Régression linéaire	13
Figure 3 API REST source : www.redhat.com	25
Figure 4 Logo Langage Python [3]	26
Figure 5 Jupyter Notebook [20]	27
Figure 6 Interface serveur Jupyter	28
Figure 7 Interface Jupyter.....	28
Figure 8 Interface Visual Studio Code	29
Figure 9 Pandas[5].....	29
Figure 10 Numpy[5]	30
Figure 11 Fractionnement des données [20]	34
Figure 12 Diagramme bête à corne	36
Figure 13 Notre dataset	38
Figure 14 Importation de librairie pour l'exploration des données	40
Figure 15 Importation du dataset.....	40
Figure 16 inventaire des données	41
Figure 17 Dictionnaire des valeurs manquantes	42
Figure 18 Détection des valeurs manquantes	43
Figure 19 Dataframe Cat et Num	44
Figure 20 valeurs aberrantes de la variable Credit_amount	44
Figure 21 Valeurs aberrantes de la variable duration	45
Figure 22 Valeurs aberrantes de la variable âge.....	45
Figure 23 histogramme de la variable credit_history	46
Figure 24 histogramme de la variable checking status	46
Figure 25 Histogramme de la variable credit_amount	47
Figure 26 Histogramme de la variable class	47
Figure 27 Distribution de la variable credit_amount en fonction de la variable class	48
Figure 28 Distribution âge en fonction de class	49
Figure 29 Distribution duration en fonction de class	49
Figure 30 Corrélation entre les variables numériques	50
Figure 31 Imputation des variables numériques	52
Figure 32 Imputation des variables catégorielles	52
Figure 33 Suppression des valeurs aberrantes.....	53
Figure 35 encodage variable ordinales	54
Figure 34 Résultat encodage à la main	54
Figure 36 One Hot encoding	55
Figure 37 RobustScaler	56
Figure 38 Suréchantillonnage	57
Figure 39 Split des dataset.....	58
Figure 40 entraînement random Forest	59
Figure 41 Random Forest bonne généralisation.....	61
Figure 42 régression logistique avant de grid search	62
Figure 43 régression logistique après le grid search	63
Figure 44 Matrice de confusion.....	65

Figure 45 Benchmark et comparaison des modèles	66
Figure 46 Page d'accueil de l'API.....	68
Figure 47 Méthode GET /healthcheck	69
Figure 48 Méthode GET/model_performances.....	70
Figure 49 Méthode POST/predict	71
Figure 50 Architecture de communication de l'API.....	72

LISTE DES ABREVIATIONS

API: Application Programming Interface

HTTP: HyperText Transfer Protocol

IA: Intelligence Artificielle

KNN: KNeighbors

ML: Machine Learning

PME : Petite et Moyenne Entreprise

SVC: Support Vector Classifier

SVM : Support Vector Machine

Table des matières

DEDICACE	i
REMERCIEMENT	ii
LISTE DES TABLEAUX	iii
LISTE DES FIGURES	iv
LISTE DES ABREVIATIONS	vi
RESUME	x
ABSTRACT	xi
INTRODUCTION	1
CHAPITRE 1: CONTEXTE DU STAGE AU SEIN DE L'ENTREPRISE IWOMI TECHNOLOGIES	3
I. Présentation de l'entreprise IWOMI Technologies.	3
1. Présentation et historique	3
2. ORGANIGRAMME	4
II. Les produits et services de IWOMI Technologies.	4
1. Présentation des services.....	4
2. Présentation des produits	5
3. Cadre de travail	8
CHAPITRE 2 : REVUE BIBLIOGRAPHIQUE.....	10
I. Généralités sur le Machine Learning.....	10
1. L'intelligence artificielle.....	10
2. Le Machine Learning	10
II. Etude de l'existant dans la banque en générale	16
1. Sollicitation	16
2. Cueillette d'information	16
3. La constitution du dossier	17
4. L'analyse du dossier.....	17
5. Décision de prêter ou non	18
6. Elaboration de l'offre de crédit.....	18
III. Le machine Learning dans la finance.	19
1. Le trading algorithmique.....	19
2. La souscription d'assurance	19
3. La détection de fraude.	19
4. La prédiction du risque de crédit (score de crédit).....	20
5. L'automatisation de processus	21

6.	Avantages de l'apprentissage automatique dans la finance	21
7.	Amélioration de l'évaluation du risque de crédit par Machine Learning	21
8.	Les avancées de l'IA dans la notion du risque de crédit bancaire	22
IV.	Les logiciels pour la prédiction du risque de crédit	23
1.	Les logiciels de prédiction du risque de crédit.....	23
2.	Les API	23
CHAPITRE 3 : MATERIELS ET METHODES		26
I.	Matériels.....	26
1.	Langage de programmation	26
2.	Environnement de programmation.....	27
3.	Bibliothèques et Framework utilisés.....	29
4.	Hardware utilisé.....	31
II.	Méthode	31
1.	La définition des objectifs et la typologie de problème	32
2.	L'acquisition et la description du dataset.....	32
3.	L'analyse exploratoire des données.	32
4.	La préparation des données	32
5.	La modélisation des données	35
6.	Le choix du modèle et les benchmarks	35
7.	Le déploiement.....	35
CHAPITRE 4 : CONCEPTION ET IMPLEMENTATION		36
I.	L'analyse fonctionnelle du besoin de notre système	36
II.	La conception du système	36
1.	Objectifs et typologie de problème	36
2.	Acquisition et description des données	38
3.	Analyse Exploratoire des Données (EDA)	40
4.	Le prétraitement des données ou preprocessing	51
5.	Modélisation des données	58
6.	Evaluation des modèles et comparaison.	63
III.	Implémentation sous forme d'API.....	66
1.	Raison de la mise sur pied d'une API	66
2.	Les conteneurs.....	67
3.	Méthode utilisée.....	67
4.	Résultats finaux.....	68
5.	Discussion	71

CONCLUSION ET PERSPECTIVES	73
REFERENCES BIBLIOGRAPHIQUES	74

RESUME

Le but principal de ce mémoire est de concevoir et d'implémenter une API de prédiction du risque de crédit bancaire basée sur le machine Learning. Pour parvenir à concevoir cette API, nous sommes passés par plusieurs étapes clés : La fixation de l'objectif (avoir une accuracy supérieure ou égale à 80% et une bonne matrice de confusion) et l'identification de la typologie de problème, l'acquisition du dataset(base de connaissances), l'analyse exploratoire des données acquises pour comprendre les interactions entre les données, le prétraitement des données, la modélisation des données, le benchmarks et la comparaison des modèles et enfin le déploiement du pipeline¹ sous forme d'API. Après avoir suivi la méthodologie, nous avons obtenu de nombreux modèles mais nous avons retenus la régression logistique, la forêt aléatoire, l'adaboosting comme modèles finaux pour notre API. Ces modèles ont produit de bonnes performances car ils ont une accuracy supérieure à 80% et une bonne matrice de confusion. Notre API réussit à bien prédire un bon client et un mauvais client en fonction des données qu'on lui fournit.

Mots clés : **Accuracy, matrice de confusion, risque de crédit, API, Big data**

¹ Pipeline : c'est un moyen d'automatiser le flux de travail d'apprentissage automatique en permettant aux données d'être transformées et corrélées dans un modèle qui peut ensuite être analysé pour obtenir des sorties.

ABSTRACT

The main goal of this thesis is to design and implement an API for prediction of bank credit risk based on machine learning. To achieve the design of this API, we went through several key stages: Setting the objectives (having an accuracy greater than or equal to 80% and a good confusion matrix) and identifying the type of problem, the acquisition of the dataset (knowledge base), exploratory analysis of the data acquired to understand the interactions between the data, the preprocessing of the data, the modeling of the data, the benchmarks and the comparison of the models and finally the deployment of the pipeline in the form of 'API. After following the methodology, we obtained many models but we retained logistic regression, random forest, and adaboosting as final models for our API. These models produced good performance because they have an accuracy greater than 80% and a good confusion matrix. Our API is able to correctly predict a good customer and a bad customer based on the new data we provide to it.

Keywords: **Accuracy, confusion matrix, credit risk, API, big data**

INTRODUCTION

Avec l'évolution des besoins du monde, de nouveaux projets naissent et nécessitent des fonds pour leur réalisation. Le ministère camerounais des PME, dans son rapport du 3 septembre 2020 portant sur l'économie sociale et l'artisanat, a annoncé que 14229 nouvelles PME ont été créées dans le pays au cours de l'année 2019[12]. Parmi ces nombreuses PME créées, beaucoup ont eu besoin de financement pour être mises sur pied. Pour pouvoir lancer leur activité, les personnes n'ayant pas assez de fond se rapprochent de plus en plus des institutions financières (banques, micro finances) pour demander un financement. En réponse à cette demande, les institutions financières leur proposent une aide sous forme de prêt bancaire. Mais avant de se voir accorder ou refuser un prêt, le demandeur de prêt est soumis à de nombreuses analyses (son revenu par mois, le motif de son prêt, le montant de son prêt, etc...) et traitement de leurs informations. Le processus actuel de traitement et d'analyse des informations des clients désirant un crédit dans certaines institutions financières, se fait encore manuellement et ce travail est très fastidieux. Les problèmes qui se posent avec la méthode traditionnelle est que, premièrement le processus de traitement des informations est très lent à cause de la quantité énorme de données à traiter par les hommes ; ensuite, le second problème posé est celui de la décision finale par le banquier car elle peut être prise avec les émotions ce qui peut être fatale à la banque (faillite) si le demandeur à qui est destiné le crédit ne rembourse pas son crédit. La prédiction faite par la banque peut être fausse si certains détails sont omis par un manque de vigilance, d'attention ou d'objectivité.

Face à la complexité du traitement et la difficulté de la manipulation des masses de données essentiellement par l'homme, les informaticiens ne sont pas restés indifférents et ont mis sur pied l'intelligence artificielle² qui se divise en deux branches : l'apprentissage automatique (machine Learning) et l'apprentissage profond (deep Learning). Ces notions ont été mise sur pied pour résoudre les problèmes de masse de données, de rapidité de traitement, de prédiction à l'aide des outils informatiques et mathématiques (probabilité, statistique, etc...). Ces outils sont insensibles aux émotions, ont une vision plus large d'un problème qu'un humain et prennent en compte tous les détails d'un problème pour prédire une meilleure réponse qu'un humain. De même, nous assistons à l'expansion et la popularité des API dans le monde, de par la facilité de leur intégration dans tout système et de par l'évolution de l'informatique. Nous pouvons donc nous pencher sur la résolution du problème de lenteur dans le traitement des

² Intelligence artificielle : ensemble des théories et des techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence humaine [17]

données bancaires tout en prédisant le risque de crédit bancaire (savoir si un client fera défaut dans le remboursement de son crédit) ceci en utilisant l'apprentissage automatique. C'est dans cette optique que notre travail sera de concevoir et d'implémenter une API de prédiction du risque de crédit bancaire basée sur le machine Learning.

Pour mener à bien notre travail, nous avons subdiviser notre travail en quatre (4) grands chapitres : le chapitre un fait office de présentation de l'entreprise d'accueil, le chapitre deux, dans lequel nous présentons tous les concepts gravitant autour de notre sujet sous formes de revue bibliographique, dans le chapitre trois nous présentons les méthodes et les matériels utilisés pour aboutir à notre solution finale et enfin le chapitre quatre qui présente les résultats de la réalisation de notre système.

CHAPITRE 1: CONTEXTE DU STAGE AU SEIN DE L'ENTREPRISE IWOMI TECHNOLOGIES

Ce chapitre a pour objectif de présenter l'environnement de stage. Dans cette optique, nous allons commencer par présenter l'entreprise, ensuite ses différents produits et services, et enfin le cadre dans lequel nous avons travaillé.


I. Présentation de l'entreprise IWOMI Technologies.

1. Présentation et historique

IWOMI Technologies Ltd est une Fintech (entreprise alliant technologie et finance) Camerounaise, et acteur majeur dans la lutte pour l'inclusion financière à travers la transformation digitale du secteur bancaire et financier en Afrique.

Elle a été créée en 2015 par M. FOMBA Collins dans le but de lutter pour l'augmentation du taux de bancarisation et de l'inclusion financière. Elle est spécialisée dans l'intégration, le développement des solutions révolutionnaires et innovantes, le conseil en stratégie, formation, organisation et transformation d'entreprises, et dans les prestations en systèmes d'information. Elle met à la disposition de ses partenaires/clients des solutions et services de qualité, révolutionnaires et pointus ; qui allient vision stratégique, connaissance métier et maîtrise technique. Ceci leur permet d'acquérir un avantage concurrentiel. Ci-dessous est présentée la fiche signalétique de l'entreprise IWOMI Technologies qui est la carte d'identité de celle-ci et permet de mieux la connaître.

Tableau 1 Fiche signalétique de l'entreprise IWOMI Technologies

Logo	
Raison sociale	IWOMI Technologies Ltd
Secteur d'activité	Tertiaire
Adresse	Bonamoussadi, Carrefour Eto'o
Ville	Douala, Cameroun
Contact	+237 677859147
Objet social	Fintech
Date de création	2015
Site web	iwomitechnologies.com
Nombre d'employés	16

2. ORGANIGRAMME

Un organigramme est une représentation de la structure interne d'une entreprise. L'organigramme de IWOMI Technologies est présenté comme suit :

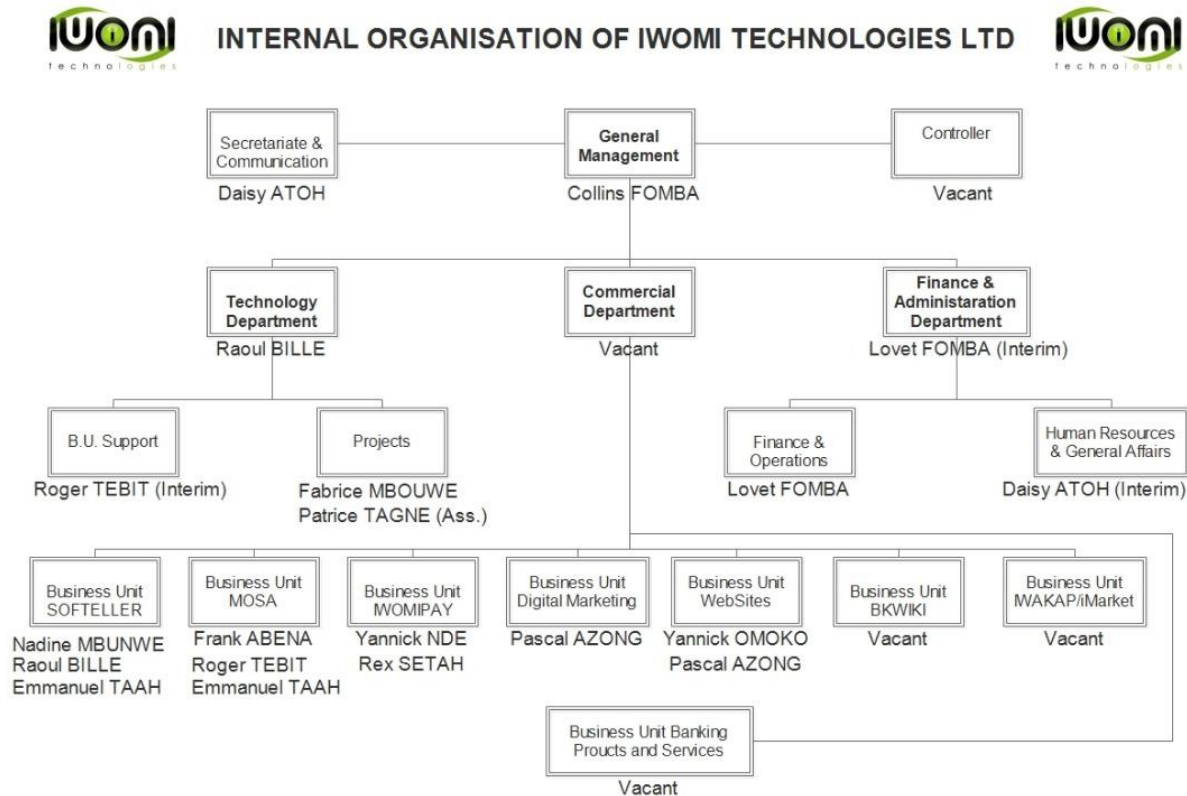


Figure 1 Organigramme de IWOMI Technologies [10]

IWOMI Technologies comprend plusieurs postes. Ainsi nous allons faire la description de quelques postes :

- General Management : A sa tête nous avons le promoteur de l'entreprise IWOMI Technologies
- Human ressources & affaires : Ici, nous avons un responsable qui est chargé de s'occuper des affaires humaines et relationnels au sein d'IWOMI Technologies.

II. Les produits et services de IWOMI Technologies.

1. Présentation des services

IWOMI Technologies Ltd a une large gamme de services développés jusqu'ici. Nous pouvons citer :

- Intégration système

- + Management de projet
- + Conseil en stratégie et organisation IT
- + Implémentation, maintenance et mise à jour du système bancaire de base
- + Développement d'application Web et mobile
- + Formation
- + Assistance technique
- + Design et hébergement web

2. Présentation des produits

IWOMI possède à son actif de nombreux produits touchant le domaine bancaire principalement mais aussi le domaine commercial. Nous distinguons deux catégories de produits à savoir:

2.1.Les Produits B2C (Business to consumer)

➤ Softeller

C'est une plate-forme accessible via son pc, mobile, tablette qui permet d'effectuer de nombreuses transactions partout dans le monde et de façon simple

Deux de ses services majeurs sont:

- Le transfert d'argent de l'étranger via des plateformes de paiement telles que PayPal, MasterCard, Visa vers le Cameroun via Orange Money, MTN Mobile Money, Express Union
- Le paiement de factures ENEO, CamWater, Canal+, etc...

➤ BKWIKI

C'est une application web et mobile qui permet en temps réel d'avoir accès à des informations sur les banques en Afrique (conditions générales, différentes agences, différents GAB, offres de crédit, offre de crédit, etc...) mais encore sur les différentes terminologies du domaine. Il s'agit en fait d'un moteur de recherche bancaire avec une localisation sur la carte en temps réel de celle-ci.

➤ IWAKAP

C'est une plate-forme de crowdfunding, qui permet de collecter des fonds du monde entier grâce à plusieurs moyens de paiement tels que les cartes de crédit, PayPal, et les services de Mobile Money dans son compte.

➤ **IMARKET**

C'est une plate-forme web et mobile qui a pour principale fonction la gestion des activités commerciales (suivi de commandes, mise à jour des catalogues), elle a également un module qui permet de gérer les paiements grâce à son système fractionné.

2.2.Produits B2B (Business to business)

a. Digital Banking Suite

C'est une suite qui permet de digitaliser les opérations bancaires (demande de crédit, facturation, transfert d'argent, consultation d'historique, etc...). Cette suite comprend:

- **Mobile Banking** : C'est l'ensemble des techniques qui permettent d'effectuer des opérations bancaires à partir d'un téléphone portable, d'un mobile, d'une tablette ou d'un Smartphone.
- **Web Banking**: C'est l'accès aux services bancaires via le web. Il propose les mêmes services que le mobile Banking, cependant il dispose de beaucoup plus d'options (plafond pour un retrait d'argent plus élevé, etc...).
- **SMS Banking**: Avec cette solution, le client pourra obtenir ses informations bancaires où qu'il soit, quand il le souhaite et quel que soit son opérateur de téléphonie mobile par des SMS.
- **USSD Banking**: Le client pourra avoir ses informations bancaires grâce à des codes USSD suivant l'opérateur choisi.
- **WhatsApp Banking**: Grâce au WhatsApp Banking, le client a accès à ses informations bancaires grâce à l'application WhatsApp et aussi il a aussi droit à un assistant virtuel.

b. IWOMI Core

C'est une solution d'interfaçage qui permet de relier plusieurs systèmes afin qu'ils puissent communiquer. Il s'agit de l'ensemble des systèmes dans lesquels la banque enregistre les clients, produits, transactions, comptes ainsi que l'architecture d'intégration entre ces systèmes.

c. BILLERS-PRO

C'est une suite sécurisée qui permet aux banques de digitaliser les règlements de factures afin d'aider les sociétés dans la collecte de leur argent pour éviter tous les risques liés à la manipulation du cash et elle permet de gérer et optimiser les flux d'argent entrant et sortant

dans une banque ou dans une structure commerciale. Cette application agit sur 3 couches de personnes:

- La couche Administrateur : Il s'agit du gestionnaire qui se charge de la gestion de ses clients.
- La couche Facturier : Il s'agit d'une société tierce désirant de collecter et emmètré ses factures via la plate-forme.
- La couche Client : Il s'agit de l'utilisateur final qui a la vue sur toutes ses factures et peut les payer sur la plate-forme via des moyens de paiement divers.

d. Loan 360

C'est une plate-forme conçue pour les institutions financières (banques, microfinances, etc...) qui leur permet de gérer les crédits (demandes, suivis, remboursement) de bout en bout en s'appuyant sur des technologies telles que l'intelligence artificielle, le machine Learning, le big data afin d'aider le gestionnaire dans sa prise de décision.

e. ARES-Reporting

C'est une solution de reporting qui permet d'avoir en temps réel des informations sur les états d'une banque et ainsi limiter les risques et les incidents encourus par la banque. Entre autre grâce à ARES-Reporting nous pouvons avoir comme services:

- ✓ Centraliser les risques
- ✓ Le contrôle du reporting
- ✓ Tableau de bord qui permet d'avoir une vue globale sur ses finances
- ✓ Le soldes des paiements et prévoir les incidents qui peuvent survenir lors des paiements.

f. M.O.S.A (Mobile Sales App)

C'est une application qui permet la collecte et le suivi des différents paiements via plusieurs moyens (mobile money, PayPal, Visa, Cash, QR Code, etc...). De façon simple, avec M.O.S.A ont peut avoir les traces de toutes ses activités c'est-à-dire ce qu'on vend (extrant), ce qui entre (intrants) et aussi nous permet de gérer le stock. Grace à M.O.S.A la traçabilité et le suivi de son historique dans ses différentes activités commerciales et autre est plus visible. Elle est subdivisée en :

- **L'administrateur** : Il s'agit du super utilisateur, il peut éditer tout ce qui a trait à ses produits, ses agents, son historique de paiement qui récapitule toutes les entrées et les sorties d'argent, il peut consulter son stock pour voir s'il faut réapprovisionner son stock. L'administrateur de MOSA a accès à son compte essentiellement via l'application Web. Seul lui peut valider un retrait d'argent de la plate-forme. Il a à sa disposition un tableau de bord qui lui permet de connaître en temps réel à quel niveau évolue son activité.
- **L'agent** : Il s'agit de l'utilisateur du terminal (TPEs, Ordinateurs). Il peut quant à lui : ajouter une catégorie pour rendre plus simple le classement des différents produits qu'il doit écouler. Il peut aussi consulter son historique et aussi émettre des factures grâce au terminal présent devant lui.

g. IWOMI Pay

C'est une plate-forme, qui permet à un système tierce, par différents moyens de paiement, d'accepter (MTN Mobile Money, Orange Money, Carte bancaire, PayPal) et effectuer des paiements (MTN Mobile Money, Orange Money) ; payer ses factures ; faire des achats de crédit grâce à ses différents APIs (Application Programming Interface) intégrés à des sites et autres applications.

Après avoir présenté les différents produits et services offerts par IWOMI Technologies, nous pouvons à présent présenter ce dans quoi nous avons travaillé.

3. Cadre de travail

Parmi ces nombreux produits présentés plus haut, celui qui nous porte le plus d'intérêt est le produit Loan 360.

Nous avons effectué notre stage dans la réalisation d'une partie importante de ce projet. Loan 360, comme décrit plus haut est une plateforme permettant de gérer les prêts bancaires. A notre arrivée en entreprise, le projet était en arrêt car il n'y avait pas des personnes qualifiées sur la main pour l'améliorer et corriger ses manquements. Les banques accusaient un ralentissement dans le traitement de leurs données avec l'utilisation de ce produit initial et ayant vu ce problème nous avons donc proposé une solution de traitement d'information et de prédiction basé sur le Machine Learning qui résoudra le problème de lenteur et de prédiction dans le processus de traitement des informations.

➤ **Objectif du projet**

L'objectif principal de l'entreprise à travers ce projet est de simplifier le processus d'octroi de crédit bancaire. Nous avons effectué essentiellement notre travail sur la partie traitement des informations car cette partie accusait au départ des problèmes particuliers. Cela s'expliquait par la lenteur en utilisant les méthodes de traitement traditionnel et la non objectivité de la décision d'octroi finale. C'est dans cette lancée qu'il sera question pour nous d'optimiser le processus de traitement des informations lors d'une demande de crédit, pour ainsi prédire un risque de crédit et prendre une décision finale fiable et plus précise.

Ainsi s'achève ce chapitre dans lequel nous avons présenté l'entreprise d'accueil dans sa généralité mais aussi notre cadre de travail. Il est donc intéressant pour nous de présenter maintenant la littérature gravitant autour de notre thème dans le chapitre suivant.

CHAPITRE 2 : REVUE BIBLIOGRAPHIQUE

Dans ce chapitre nous présenterons le machine Learning dans sa généralité, ensuite nous présenterons l'état actuel du processus d'octroi de crédit après nous présenterons la notion de crédit scoring, par la suite nous présenterons le machine Learning dans le domaine de des finances, et nous finirons par présenter les API de machine Learning qui existent sur le marché

I. Généralités sur le Machine Learning

Nous ne saurons parler de machine Learning sans parler du grand ensemble qui est l'intelligence artificielle.

1. L'intelligence artificielle

L'intelligence artificielle (IA) est l'ensemble des théories et des techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence humaine [17]. Elle est aussi définie par l'un de ses créateurs, **Marvin Lee Minsky**, comme : " la construction de programmes informatiques qui s'adonnent à des tâches qui sont pour l'instant, accomplies de façon plus satisfaisante par des êtres humains car elles demandent des processus mentaux de haut niveau tels que : l'apprentissage perceptuel, l'organisation de la mémoire et le raisonnement critique. ".

Il existe plusieurs domaines dans l'intelligence artificielle dont le machine Learning qui sera le prochain point à aborder.

2. Le Machine Learning

2.1.Définition du Machine Learning

Le machine Learning est un sous-ensemble de l'intelligence artificielle (IA) qui est axé sur la création de systèmes qui apprennent ou améliorent les performances en fonction des données qu'ils traitent [7].

2.2.Domaine d'application du Machine Learning

Aujourd'hui, nous utilisons le machine Learning dans tous les domaines. Il est utilisé dans :

- Le commerce en ligne permettant de faire des recommandations client, les prédictions de ventes etc...
- Internet dans les moteurs de recherche, les assistants intelligents, etc...
- La médecine pour le diagnostic médical des maladies, la prescription des médicaments en fonction des symptômes, etc...

- L'imagerie et la vidéographie pour la reconnaissance des formes, des images, la synthèse d'images etc...
- Dans le traitement des sons pour la reconnaissance vocale.
- Dans la finance pour la détection des fraudes, la prédiction des risques d'octroi de crédit, etc...
- Et aussi dans d'autres domaines tels que les télécommunications, la robotique, etc...

2.3.Objectifs du Machine Learning

Le Machine Learning est massivement utilisé pour la Data Science et l'analyse de données. Il permet de **développer**, de **tester** et d'**appliquer** des algorithmes d'analyse prédictive sur différents types de données afin de **prédire** le futur [16]. En automatisant le développement de modèle analytique, le Machine Learning permet d'accélérer l'analyse de données et de la rendre plus précise. Il permet d'assigner aux machines des tâches au cœur de l'analyse de données comme la classification, le clustering ou la détection d'anomalie.

Pour atteindre ces objectifs, le machine Learning se divise en deux phases principales: la phase d'apprentissage et la phase de prédiction.

2.4.Les phases du machine Learning

2.4.1. La phase d'apprentissage

L'objectif du machine Learning est de laisser la machine construire son système de raisonnement sans avoir à imposer un programme au préalable [18]. Pour cette phase d'apprentissage, la machine se base sur plusieurs exemples afin de comprendre la logique du modèle qu'elle doit intégrer. Pour se former, le machine Learning va s'intéresser aux masses de données à analyser afin d'en déterminer l'algorithme de transformation. A partir des exemples étudiés, le machine Learning commence ainsi sa phase d'apprentissage en toute autonomie.

2.4.2. La phase de prédiction

Après avoir intégré le raisonnement et l'algorithme du problème en question, le machine Learning doit être en capacité de déterminer la finalité d'une situation donnée [18]. Plus l'apprentissage du machine Learning est complet, plus les prédictions obtenues par cet outil seront précises.

2.5.Les types d'apprentissage automatique.

Le fonctionnement du machine Learning se base sur 3 types d'apprentissage: l'apprentissage supervisé, l'apprentissage non-supervisé, et l'apprentissage par renforcement.

Ces 3 types s'inscrivent dans les phases d'apprentissage et de prédiction qui caractérisent le fonctionnement du machine Learning.

2.5.1 L'apprentissage supervisé

Ce type d'apprentissage s'appuie sur un algorithme d'apprentissage supervisé, c'est-à-dire que la machine s'intéresse à des données d'entrée et de réponses aux données déjà connues. Ce traitement des données connues, va permettre à l'algorithme de se former et d'élaborer un modèle de prévisions adapté pour les données à traiter par la suite [7].

Pour développer ces modèles prédictifs, l'apprentissage supervisé va notamment se baser sur des techniques de **classification**. La mise en pratique des techniques de classification implique le fait que les données peuvent être identifiées et catégorisées en fonction de leurs caractéristiques. L'application des techniques de classification se retrouvent dans différents domaines : digital, médical, banque.

L'apprentissage supervisé se retrouve également dans les techniques de **régression**. Cette technique est utilisée pour prévoir des variables continues. A partir de l'apprentissage supervisé d'un modèle de données variables, la machine est censée prévoir les différentes variations et fluctuations de données réelles comme les variations de température ou la demande en énergie selon les périodes de l'année.

2.5.2 L'apprentissage non supervisé

A l'inverse de l'apprentissage supervisé, celui non supervisé va s'appuyer sur des données d'entrée dont les réponses ne sont pas identifiées. L'objectif de ce type de technique est de mettre en exergue des modèles intrinsèques aux données traitées [7]. Parmi l'apprentissage non supervisé, on retrouve la technique de clustering. Ce modèle d'apprentissage non supervisé est le plus courant et permet d'identifier des points communs entre certaines données et de les regrouper clairement par groupe.

La machine se contente d'explorer les données à la recherche d'éventuelles patterns. Elle ingère de vastes quantités de données, et utilise des algorithmes pour en extraire des caractéristiques pertinentes requises pour étiqueter, trier et classer les données en temps réel sans intervention humaine.

2.5.3 Apprentissage par renforcement

L'apprentissage par renforcement consiste à laisser un algorithme apprendre de ses erreurs pour atteindre un objectif. L'algorithme essayera de nombreuses approches différentes pour tenter d'atteindre son but [7].

En fonction de ses performances, il sera récompensé ou pénalisé pour l'inciter à poursuivre dans une voie ou à changer d'approche. Cette technique est notamment utilisée pour permettre à une IA de surpasser les humains dans les jeux.

Par exemple, AlphaGo de Google a battu le champion de Go grâce à l'apprentissage par renforcement. De même, OpenAI a entraîné une IA capable de vaincre les meilleurs joueurs du jeu vidéo Dota 2.

La force du machine Learning repose sur les modèles qui traitent les données qu'on passe à la machine, alors il est important de présenter quelques modèles de Machine Learning.

2.6. Les modèles de l'apprentissage automatique

En machine Learning on distingue des modèles en apprentissage supervisé mais aussi en apprentissage non supervisé :

2.6.1. Les modèles supervisés

a. La régression linéaire:

C'est un modèle avec lequel on essaie de trouver une relation entre la variable à prédire et les variables qui décrivent nos données. Un modèle de régression linéaire est un modèle de machine Learning dont la variable cible (Y) est quantitative tandis que la variable (X) peut être quantitative ou qualitative [6]. L'idée de la régression linéaire est simplement de trouver une droite qui correspond le mieux aux données. La fonction recherchée est de la forme : $Y = f(X)$ avec $f(X)$ une fonction linéaire.

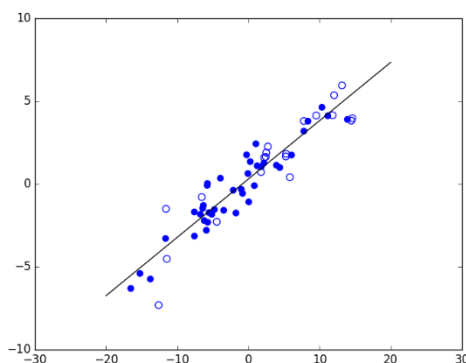


Figure 2 Régression linéaire

b. Arbre de décision

Les **arbres de décision** sont un modèle populaire, utilisé dans la recherche opérationnelle, la planification stratégique et le **Machine Learning**. Chaque rectangle dans un arbre de décision en machine Learning est appelé un **nœud** [6]. Plus il y a des nœuds, plus l'arbre décisionnel sera précis (en général). Les derniers nœuds de l'arbre décisionnel, où une décision est prise, sont appelés les « **feuilles** » de l'arbre.

c. Forêt aléatoire (Random Forest)

La forêt aléatoire est un algorithme d'apprentissage supervisé. La « forêt » qu'il construit, est un ensemble d'arbres de décision, généralement formés avec la méthode de l'ensachage [6]. L'idée générale de la méthode d'ensachage est qu'une combinaison de modèles d'apprentissage augmente le résultat global. Un grand avantage de la forêt aléatoire est qu'elle peut être utilisée à la fois pour les problèmes de classification et de régression, qui forment la majorité des systèmes d'apprentissage automatique actuels.

d. Régression Logistique

La régression logistique est semblable à la régression linéaire, mais elle est utilisée pour modéliser la probabilité d'un nombre fini de résultats, généralement deux. Une équation logistique est créée de telle sorte que les valeurs des résultats ne peuvent être qu'entre 0 et 1.

e. Support Vector Machines (SVMs)

Un **Support Vector Machine** est une technique de classification supervisée qui trouvera un **hyperplan** ou une frontière entre les deux classes de données qui maximisera la marge entre les deux classes [6]. Il y a plusieurs plans qui peuvent séparer les deux classes, mais un seul plan peut maximiser la marge ou la distance entre les classes. Dans l'algorithme SVM, nous recherchons à maximiser la marge entre les points de données et l'hyperplan. La fonction de perte qui aide à maximiser la marge est la perte de charnière.

f. KNeighbors

L'algorithme KNeighbors (KNN) encore appelé algorithme du plus proche voisin suppose que des choses similaires existent à proximité [6]. En d'autres termes, des choses similaires sont proches les unes des autres. L'algorithme KNN saisit l'idée de similitude (parfois appelée distance ou proximité) avec certaines mathématiques que nous aurions pu apprendre dans notre enfance.

g. Naïve Bayes

Un classifieur bayésien naïf suppose que l'existence d'une caractéristique pour une classe, est indépendante de l'existence d'autres caractéristiques [6]. Un fruit peut être considéré comme une pomme s'il est rouge, arrondi, et fait une dizaine de centimètres. Même si ces caractéristiques sont liées dans la réalité, un classifieur bayésien naïf déterminera que le fruit est une pomme en considérant indépendamment ces caractéristiques de couleur, de forme et de taille.

2.6.2. Les modèles non supervisés

a. Regroupement ou clustering

Le regroupement ou Clustering est la technique la plus utilisée pour résoudre les problèmes d'apprentissage non supervisé. La mise en cluster consiste à séparer ou à diviser un ensemble de données en un certain nombre de groupes, de sorte que les ensembles de données appartenant aux mêmes groupes se ressemblent davantage que ceux d'autres groupes.

b. Clustering par décalage moyen

Le clustering par décalage moyen est un algorithme basé sur une fenêtre glissante qui tente de trouver des zones denses de points de données. C'est un algorithme basé sur un centroïde, ce qui signifie que l'objectif est de localiser les points centraux de chaque groupe / classe, ce qui fonctionne en mettant à jour les candidats pour que les points centraux soient la moyenne des points dans la fenêtre glissante.

2.6.3. Les modèles ensemblistes

Ce sont des modèles qui combinent plusieurs modèles pour en faire un super modèle. On en distingue deux sous-familles :

a. Le bagging

Il permet d'entraîner plusieurs modèles en parallèle pour ensuite les regrouper afin de prendre une décision. L'algorithme de bagging le plus connu est le Random Forest dont le principe est d'apprendre plusieurs arbres de manière indépendante sur des échantillons (avec remplacement) des données d'apprentissage. La prédiction finale sera agrégée par un vote majoritaire (ou une moyenne des probabilités) issu des prédictions des différents arbres appris.

b. Le boosting

Il consiste aussi à combiner des classificateurs mais de manière séquentielle. L'idée est que le classificateur numéro N se concentre sur les items mal classés par le classificateur numéro N-1. En effet chaque classificateur est entraîné sur un jeu de données pondéré qui accorde plus

d'importance aux observations qui ont été mal classées par le classificateur précédent. La prédiction finale est aussi une combinaison pondérée des prédictions de tous les classificateurs appris en fonction de leur performance.

II. Etude de l'existant dans la banque en générale

Dans le domaine de la banque et de la finance en général, le processus d'octroi de crédit bancaire est assez long [1]. Pour qu'une banque décide d'octroyer le crédit à un client et en refuser à un autre, un long processus est mis en place. Les principales étapes du processus d'octroi de crédit de façon traditionnelle sont:

- La sollicitation
- La cueillette d'information
- La constitution du dossier
- L'analyse du dossier
- Décision de prêter ou non
- Elaboration de l'offre de crédit

1. Sollicitation

Ici, il s'agit de la demande de crédit proprement dite. Le client posera sa demande de crédit à l'institution financière en face de lui.

2. Cueillette d'information

Lorsque le client potentiel manifeste le désir d'obtenir du crédit, alors commence l'opération de collecte des informations utiles à la banque pour décider d'accorder ou pas un crédit. La première source d'information est le client lui-même, qui apportera son dossier d'entreprise (s'il fait partie d'une entreprise), ses rapports salariaux annuels, une description de son projet et ses propres prévisions financières. La deuxième source d'informations se sont les données que la banque peut déjà avoir sur ce client compte tenu de ses relations antérieures avec celle-ci. Il peut s'agir de prêts antérieurs ou en cours et de l'historique des transactions quotidiennes du client. Enfin, la banque peut obtenir de l'information des tiers à savoir l'agence de crédit, bureau de comptable, fournisseurs ou firme d'experts pouvant donner une opinion sur le projet considéré tel que des études de faisabilité ou des études de marché. En recueillant une information le chargé de compte doit continuellement tenter de juger la fiabilité de la source et la valeur de l'information fournie

3. La constitution du dossier

De façon générale, le dossier de demande de crédit comprendra de nombreux éléments en fonction de l'entité qui demande le prêt (particulier ou entreprise). S'agissant de l'**entreprise** nous avons les éléments tels que :

- L'identification et le bref historique de l'entreprise
- La description de son marketing (ses produits, ses segments de marchés)
- La description de ses opérations d'approvisionnement et de production ainsi que l'appréciation de ses équipements et de sa technologie.
- Evaluation de sa main d'œuvre et de ses relation travail
- Evaluation de sa direction (intégrité, stabilité et compétence)
- Analyse de ses résultats financiers
- Le motif de la demande de crédit et des projections financières.

S'agissant d'un **particulier**, nous pouvons avoir les éléments tels que :

- L'âge du demandeur
- Son sexe
- Ses revenus annuels
- Le montant de sa demande
- Le motif de sa demande
- Ses antécédents bancaires etc...

4. L'analyse du dossier

Une fois l'information obtenue sur chacun des points en question, celle-ci pourra être traitée de manière qualitative par le chargé de compte. L'approche qualitative traditionnelle se base sur cinq critères que l'on appelle les cinq « C » du crédit, ce sont en anglais : capacity, character, capital, collateral et conditions. Examinons tour à tour ces critères.

- **La capacité de remboursement** réfère aux revenus et mieux encore aux flux financiers dont posera probablement l'emprunteur pour rembourser le capital et les intérêts de sa dette
- **Le caractère** a trait à la valeur morale et à l'intention de repayer de l'entreprise ou du particulier.
- **Le capital** mesure le degré d'implication financière de l'emprunteur dans l'entreprise et pour le particulier ses revenus et ses actifs
- Le terme « **collateral** » fait référence aux garanties que peut donner l'emprunteur.

- **Les conditions** signifient les conditions du marché, soit de la conjuncture économique générale et sectorielle et en particulier le niveau des taux d'intérêt.

5. Décision de prêter ou non

Suite à l'analyse, les représentants de la banque décideront d'abord s'ils désirent ou non procéder à une offre de financement. Ils réalisent qu'ils peuvent alors faire deux types d'erreur :

- **Erreur de type 1:** accepter de prêter à un emprunteur qui fera éventuellement défaut
- **Erreur de type 2:** refuser de prêter à un emprunteur qui aurait bien honorer ses engagements.

6. Elaboration de l'offre de crédit

L'offre de crédit comporte plusieurs éléments :

- **Le montant du prêt :** Il doit correspondre à la fois au besoin du projet et à la capacité de remboursement de l'emprunteur.
- **Le taux d'intérêt :** Il reflétera le niveau général des taux dans l'économie, l'échéance du prêt et la prime de risque que requière le prêt.
- **La période d'amortissement :** est la durée totale du remboursement du prêt. Elle correspond le plus souvent à la durée de la vie de l'actif financé.
- **Les garanties exigées :** Elles peuvent prendre plusieurs formes: comptes à recevoir, stocks, équipement, immeubles, valeurs mobilières, cautions personnelles, d'une compagnie-mère ou du gouvernement.
- **Les clauses de gestion imposées**
- **La négociation et la signature du contrat**
- **Le déboursement du montant de prêt au client**
- **Le suivi et l'intervention**
- **La conclusion du prêt**

III. Le machine Learning dans la finance.

L'utilisation du Machine Learning en Finance ne cesse de se développer au cours de ces dernières années. Ce développement se fait ressentir de par l'évolution de certaines branches de la finance. Entre autre, comme application du machine Learning dans le domaine de la finance nous avons : le trading algorithmique, la souscription d'assurance, la détection de fraude, la prédiction du risque de crédit, l'automatisation de processus, etc...

1. Le trading algorithmique

Dans le trading algorithmique, l'apprentissage automatique aide à prendre de meilleures décisions de trading. Un modèle mathématique surveille les nouvelles et les résultats commerciaux en temps réel et détecte les tendances pouvant forcer la hausse ou la baisse des cours des actions [15]. Il peut ensuite agir de manière proactive pour vendre, détenir ou acheter des actions en fonction de ses prévisions. Les algorithmes d'apprentissage automatique peuvent analyser simultanément des milliers de sources de données, ce que les traders humains ne peuvent réaliser. Comme exemple de logiciel de trading algorithmique, nous pouvons citer DUPLITRADE, MQL5 (qui est le plus utilisé par les traders crypto-monnaies)

2. La souscription d'assurance

À court terme, le Machine Learning peut aider à automatiser de grands volumes de souscription d'assurances auto, habitation, etc. À l'avenir, l'intelligence artificielle améliorera la modélisation, en soulignant les considérations clés pour les décideurs humains qui auraient autrement pu passer inaperçues [15]. Il est également possible que le Machine Learning permettra une souscription personnalisée par entreprise ou individu, en tenant compte de comportements et de circonstances uniques. La souscription améliorée peut tirer parti non seulement de l'apprentissage automatique pour l'exploration de données, mais également de la technologie portable et des analyseurs faciaux d'apprentissage en profondeur.

Par exemple, Lapetus, une startup, souhaite utiliser les selfies pour prévoir avec exactitude l'espérance de vie. Ces types d'analyses de risque nuancées et en temps réel permettront non seulement une tarification plus précise des clients, mais également une détection précoce des risques (source : www.analyticsinsights.io)

3. La détection de fraude.

La fraude par carte de crédit en utilisant de nouveaux comptes est le numéro un des cyber crimes liés à l'usurpation d'identité. Les paiements devenant de plus en plus fluides, les banques

ne peuvent plus compter sur la détection régulière des fraudes basée sur des règles. Les systèmes de détection des fraudes basés sur l'apprentissage automatique se généralisent.

Mastercard utilise la solution **Vocalink** pour évaluer toutes les transactions des nouveaux clients au début du cycle de vie de chaque transaction. Cela permet de prévenir la fraude à un stade précoce [15]. Leur solution de prévention de la fraude surveille les transactions et informe les institutions financières des processus suspects qui nécessitent un examen plus approfondi. Outre la prévention de la fraude, elle fournit également des rapports d'analyse et prévient le blanchiment d'argent.

4. La prédiction du risque de crédit (score de crédit)

Lorsqu'une banque prête de l'argent à une personne, elle prend le risque que cette dernière ne rembourse pas cet argent dans le délai convenu. Ce risque est appelé Risque de Crédit. Alors avant d'octroyer un crédit, les banques vérifient si le client (ou la cliente) qui demandent un prêt sera capable ou pas de le rembourser.

Grâce à des modèles de Machine Learning, les banques peuvent modéliser la probabilité de défaut de paiement et ainsi attribuer un score à chaque nouveau demandeur de crédit. Ce score est appelé score de risque de crédit ou credit scoring. Nous distinguons 6 types de score à savoir :

- **Le score d'appétence** : C'est la probabilité pour un client d'être intéressé par un produit ou un service donné.
- **Le score de risque de crédit** : C'est la probabilité qu'un client avec un compte courant ne rembourse pas son crédit.
- **Le score d'acceptation** : C'est la probabilité qu'un nouveau client ne rembourse pas son crédit
- **Le score de recouvrement** : C'est l'évaluation du montant susceptible d'être récupéré sur un compte ou un crédit au contentieux.
- **Le score d'attribution** : C'est la probabilité pour un client de quitter la banque

La société LenddoEFL a développé un logiciel qui pourrait aider les banques à déterminer la solvabilité d'emprunteurs potentiels grâce à l'analyse prédictive et au traitement du langage naturel. Nous avons aussi la société ZestFinance qui propose un logiciel appelé ZAML, qui, selon elle, peut aider les organismes prêteurs à déterminer la solvabilité des emprunteurs potentiels et à réduire les défauts de paiement en utilisant des analyses prédictives et un traitement en langage naturel [8].

5. L'automatisation de processus

L'automatisation des processus est l'une des applications les plus courantes de l'apprentissage automatique en finance. La technologie permet de remplacer le travail manuel, d'automatiser les tâches répétitives et d'augmenter la productivité.

De ce fait, l'apprentissage automatique permet aux entreprises d'optimiser leurs coûts, d'améliorer l'expérience de leurs clients et de développer leurs services. Voici des exemples d'utilisation de l'apprentissage automatique en finance:

- Chatbots (Comme exemple nous avons Watson développé par IBM)
- Automatisation de centre d'appels.

6. Avantages de l'apprentissage automatique dans la finance

Grâce à l'IA, les décisions assistées par ordinateur sont prises beaucoup plus rapidement en raison de la grande vitesse de traitement des données des systèmes informatiques [14]. Ses principaux avantages dans le secteur financier sont les suivants :

- Automatisation des processus de routine;
- Une vitesse de service accrue;
- Réduction des coûts de résolution des tâches standard;
- Une précision accrue du traitement de grandes quantités de données;
- Amélioration de la qualité du système de soutien à la clientèle.

7. Amélioration de l'évaluation du risque de crédit par Machine Learning

Dans la plupart des institutions financières, les modèles de notation de crédit sont toujours fondés sur l'approche du tableau de bord, c'est-à-dire la dynamique caractéristique au moment de leur création. Un emprunteur potentiel doit posséder suffisamment de données historiques sur le comportement d'emprunt antérieur pour être qualifié de « brûlable ». En l'absence de telles informations historiques (ce qui est une situation typique pour les nouveaux clients du secteur bancaire), même les emprunteurs solvables se voient refuser l'accès au crédit.

Contrairement aux méthodes traditionnelles de notation de crédit (la méthode du tableau de bord) axées sur le rendement passé de l'emprunteur, la notation de crédit grâce à l'IA est plus sensible aux indicateurs en temps réel de la solvabilité de l'emprunteur potentiel, comme le niveau actuel de revenu, les possibilités d'emploi et sa capacité potentielle de gagner. Ainsi, les emprunteurs à fort potentiel sont inclus dans les programmes de crédit, tandis que ceux qui réussissent officiellement l'évaluation conventionnelle de notation de crédit en sont exclus. En

d'autres termes, la notation de crédit basée sur l'IA permet des prévisions de bénéfices précises basées sur les modèles intelligents de l'IA.

8. Les avancées de l'IA dans la notion du risque de crédit bancaire

Les solutions de l'IA transforment la notation de crédit de plusieurs façons, signalant un changement de paradigme dans cet aspect. Voici quelques bonus évidents de l'intégration de l'IA dans la notation de crédit [15].

- **Une plus grande orientation client.**

Les méthodes traditionnelles de notation du crédit sont critiquées à maintes reprises comme dépassées, dans une plus large mesure en raison de leur normalisation et de leur manque de sensibilité aux disparités et aux nuances individuelles. Avec l'introduction de l'IA dans les systèmes de notation de crédit, les banques obtiennent des informations uniques sur le comportement financier de leurs clients non seulement sur la base de données historiques, mais aussi sur la base des prévisions de revenus potentiels

- **Une plus grande vitesse.**

Plus précisément dans la notation de crédit, les applications d'IA deviennent de plus en plus populaires en raison de leur capacité à accélérer le processus de prise de décisions de prêt sans aucun compromis de qualité ou de précision. Aujourd'hui, des masses de données superflues, non structurées et partiellement structurées sont incluses dans l'analyse (l'utilisation des médias sociaux, activités de téléphonie mobile, etc.) pour prendre des décisions plus intelligentes en matière de crédit, mais avec l'aide de l'IA, la vitesse du traitement des données reste élevée par rapport aux méthodes traditionnelles.

- **Un meilleur accès au crédit.**

Avec l'aide de la science des données, la notation de crédit est devenue plus orientée vers l'avenir, contrairement aux approches axées sur le passé à la vieille école. De cette façon, un plus grand nombre d'emprunteurs ont accès au crédit aujourd'hui (étudiants, fondateurs d'entreprises prometteuses, résidents étrangers, etc...), ce qui stimule leur entreprise et les aide à relancer leurs idées. L'obtention de son tout premier crédit est également devenue plus simple, ce qui se fait maintenant en fonction des projections financières de l'IA concernant le potentiel de revenu et les possibilités d'emploi du client.

IV. Les logiciels pour la prédiction du risque de crédit

Les logiciels de prédiction de risque de crédit sont assez nombreux sur le marché, mais ces logiciels sont confrontés à une autre catégorie de programme appelé API.

1. Les logiciels de prédiction du risque de crédit

Sur le marché il existe de nombreux logiciels permettant de gérer les risques de crédit bancaire et nous pouvons citer :

- **Obertys** qui est un logiciel permettant aux institutions financières de calculer le score de leur client. Il permet essentiellement de faire les tâches telles que : scorer tous les clients de manière plus intelligente et scientifique, étudier des périodes creuses pour la proposition de crédit, obtenir le score sous forme de pourcentage, offrir la possibilité de proposer un montant acceptable pour les mauvais scores, savoir à quel pourcentage de risque le prêt sera soldé. Mais le problème avec ce système c'est son prix assez élevé et pas personnalisable à sa convenance [19].
- **BEST** développé par PROLOGIA, qui est un outil d'aide à la décision donnant à l'utilisateur tous les éléments nécessaires à la prise de décision finale. Il analyse le risque pour l'octroi d'un crédit, il calcule la tarification pour un dossier de crédit, calcule la notation des clients en se basant sur le comportement bancaire. Le principal inconvénient de ce logiciel est qu'il est privé et on ne peut pas le personnaliser à notre convenance [8].
- Nous avons **Orange Datamining** qui est un outil gratuit de traitement des données qui est basé sur le machine Learning aussi, mais son inconvénient est qu'il n'est pas conseiller pour le déploiement en production car il est beaucoup plus éducatif et instructif au même titre que le logiciel **TANAGRA**.

Le problème majeur avec tous ces logiciels est qu'ils n'offrent pas de possibilités de personnalisation car ce sont des logiciels déjà fixes. Or, les API quant à eux peuvent s'intégrer à n'importe quel système existant et ainsi offrant des possibilités de personnalisation suivant nous goût, d'où nous avons opter pour une API.

2. Les API

2.1.Définition d'API

Une API est un programme permettant à deux ou plusieurs applications distinctes de communiquer entre elles et d'échanger des données. Cela évite notamment de recréer et

redévelopper entièrement une application pour y ajouter ses informations. Par exemple, elle est là pour faire le lien entre des données déjà existantes et un programme indépendant.

A l'ère de l'open data et de la digitalisation de la société, les interfaces de programmation sont au cœur du fonctionnement du web. C'est le cas par exemple pour une application de prédiction. Ce n'est pas elle qui va chercher et analyser les informations des différents clients et nous les transmettre mais bien une API qui va se connecter à la base de données où se trouvent ces informations pour les traiter et les afficher dans notre application.

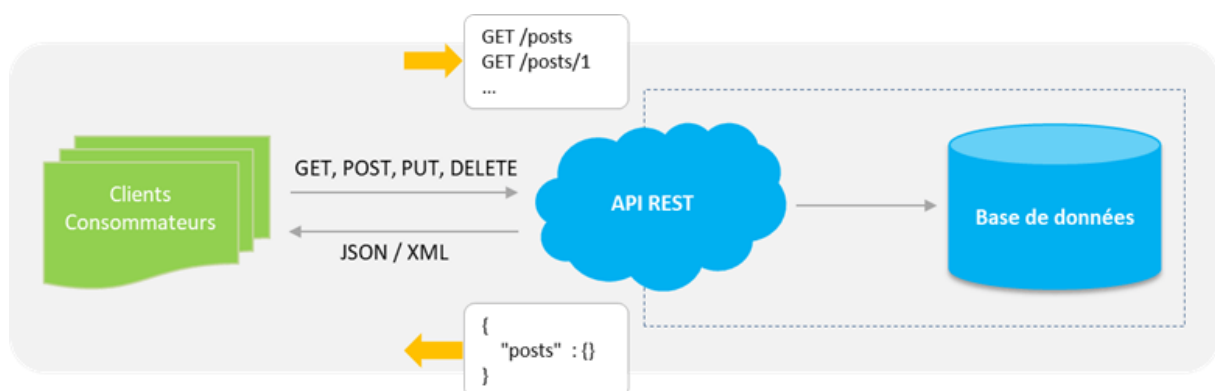
2.2. Les différents types d'API

Il existe deux catégories principales d'API : les API publiques, appelées aussi open API, et les API privées, connues sous le nom de Enterprise API. Les API sécurisées disposent alors d'une clé d'identification, fournie par un service d'authentification et d'autorisation.

Du fait de la grande diversité d'applications client, les API doivent s'appuyer sur un protocole de communication, le SOAP (Simple Object Access Protocol) ou le REST (Representational State Transfer) afin d'être compatibles aux diverses plateformes mobiles, qu'il s'agisse d'une application Windows, Apple ou Android. L'API Rest (ou Restful) est à présent la plus utilisée car elle offre plus de flexibilité et c'est ce type que nous devons mettre en place.

2.3. Le fonctionnement des API REST

L'API REST, Representational State Transfer, parfois appelée Restful, est une API qui permet la communication entre deux applications web de manière légère et efficace. Elle a été définie par Roy Fielding, un contributeur important du protocole HTTP, dans sa thèse de doctorat sur les styles d'architecture logiciels. L'API REST est basée sur le protocole http en permettant l'échange des données, dans la majorité des cas, au format JSON, qui est à la fois efficace et facilement lisible par l'homme. Elle fait particulièrement usage des méthodes GET



(obtenir les données dans une base de donnée), POST (créer un nouvel objet dans la base de donnée) et PUT (modifier / remplacer une ressource).

Ainsi, s'achève ce chapitre dans lequel nous avons présenté la littérature qui gravite autour de notre thème et nous pouvons ainsi passer au chapitre suivant pour décrire et présenter le matériel et les méthodes utilisés pour la résolution de notre problème.

Figure 3 API REST source : www.redhat.com

CHAPITRE 3 : MATERIELS ET METHODES

Dans ce chapitre, il sera question pour nous de présenter le matériel et les méthodes utilisés pour la réalisation de notre API.

I. Matériels

Pour parvenir à la version finale de notre solution, plusieurs procédés ont été fait en amont, et dans ce cas nous aborderons d'abord le matériel utilisé pour notre système ; car pour faire fonctionner un système il faut au préalable du matériel (logiciel et/ou physique).

1. Langage de programmation

Python



Figure 4 Logo Langage Python [3]

En Science de données, le langage le plus utilisé est Python car il présente de nombreux avantages par rapport aux autres langages à savoir :

- Simplicité de syntaxe,
- Adapté à toute sorte de plateforme tel que Windows, MacOS, Linux, etc,
- Supporte le développement fonctionnel, orienté objet, et procédural,
- Temps de développement relativement court par rapport à d'autres langages,
- Grande Communauté,
- Grande bibliothèque de librairies. [3]

Python est un langage de programmation interprété, multi-paradigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Le langage Python est placé sous une licence libre proche de la licence BSD4 et fonctionne sur la plupart des plates-formes informatiques, des smartphones aux ordinateurs centraux5, de Windows à Unix avec notamment GNU/Linux en passant par MacOS, ou encore Android, iOS, et peut aussi

être traduit en Java ou .NET. Il est conçu pour optimiser la productivité des programmeurs en offrant des outils de haut niveau et une syntaxe simple à utiliser.

2. Environnement de programmation

Pour faire tourner un langage de programmation, il faut impérativement un environnement de programmation. Pour la réalisation de notre système nous avons utilisés deux environnement de programmation à savoir Jupyter Notebook et Visual Studio Code.

➤ Jupyter Notebook



Figure 5 Jupyter Notebook [20]

Le Jupyter Notebook est une application Web open source qui permet de créer et de partager des documents contenant du code en direct, des équations, des visualisations et du texte narratif. Les utilisations incluent : nettoyage et transformation des données, simulation numérique, modélisation statistique, visualisation des données, apprentissage automatique, et bien plus encore.

Jupyter prend en charge plus de 40 langages de programmation, dont Python, R, Julia, etc...; et notre code peut produire une sortie riche et interactive.

Grâce à cet environnement, nous avons pu effectuer toutes les étapes de base (du traitement des données jusqu'à la modélisation) d'un projet de ML, et la force de cet environnement est le fait qu'il affiche le résultat sous forme de visuel directement à la suite du code entré et tout cela se fait dans un navigateur en localhost.

Pour l'installer c'est assez simple, tout d'abord il faut ouvrir l'invite de commande sous Windows et insérer **pip install jupyter**. Il sera alors téléchargé et installé sur votre ordinateur et la prochaine étape à suivre est de démarrer son serveur en entrant toujours l'invite de

commande **jupyter notebook**. Après avoir effectué ces étapes, jupyter sera démarré et le navigateur par défaut devrait aussi démarrer à l'URL suivante : <http://localhost:8888/tree>.

Le navigateur devrait ressembler maintenant à ceci :

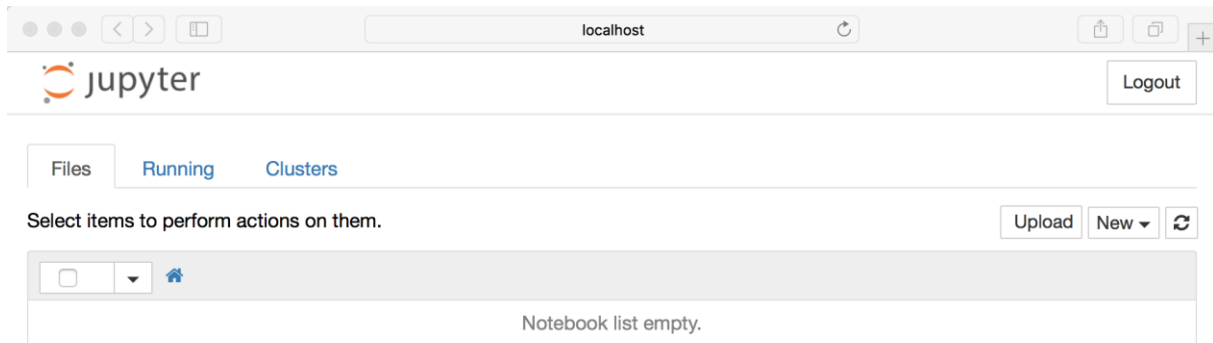


Figure 6 Interface serveur Jupyter

Maintenant que nous avons démarré un serveur Notebook, nous devons créer un document Notebook réel.

Tout ce que nous devons faire est de cliquer sur le nouveau bouton (en haut à droite), et il va ouvrir une liste de choix. Par souci de simplicité, choisissons Python 3.

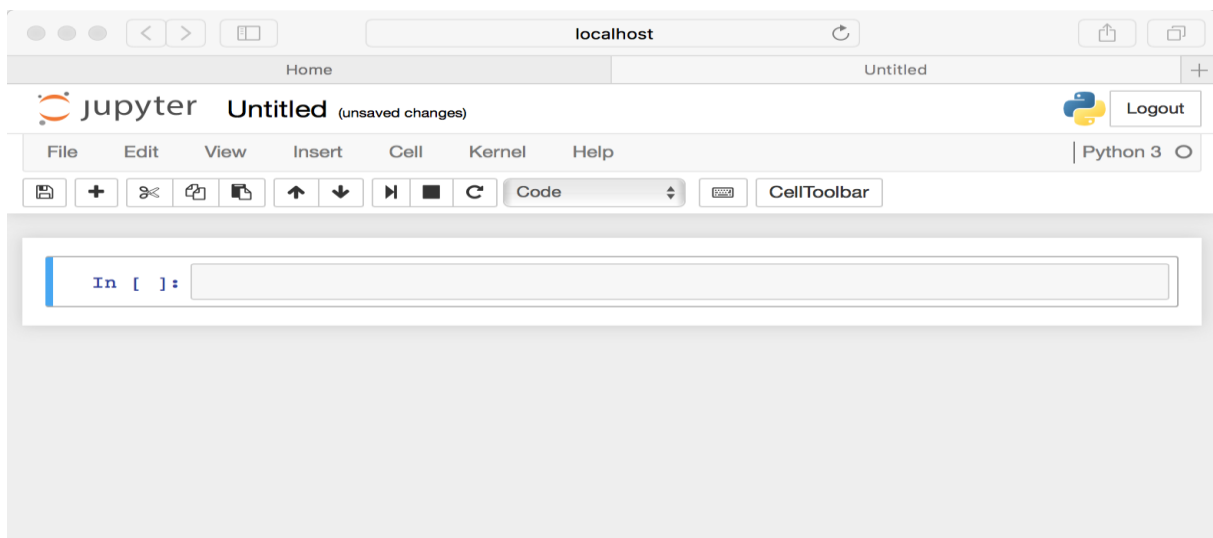
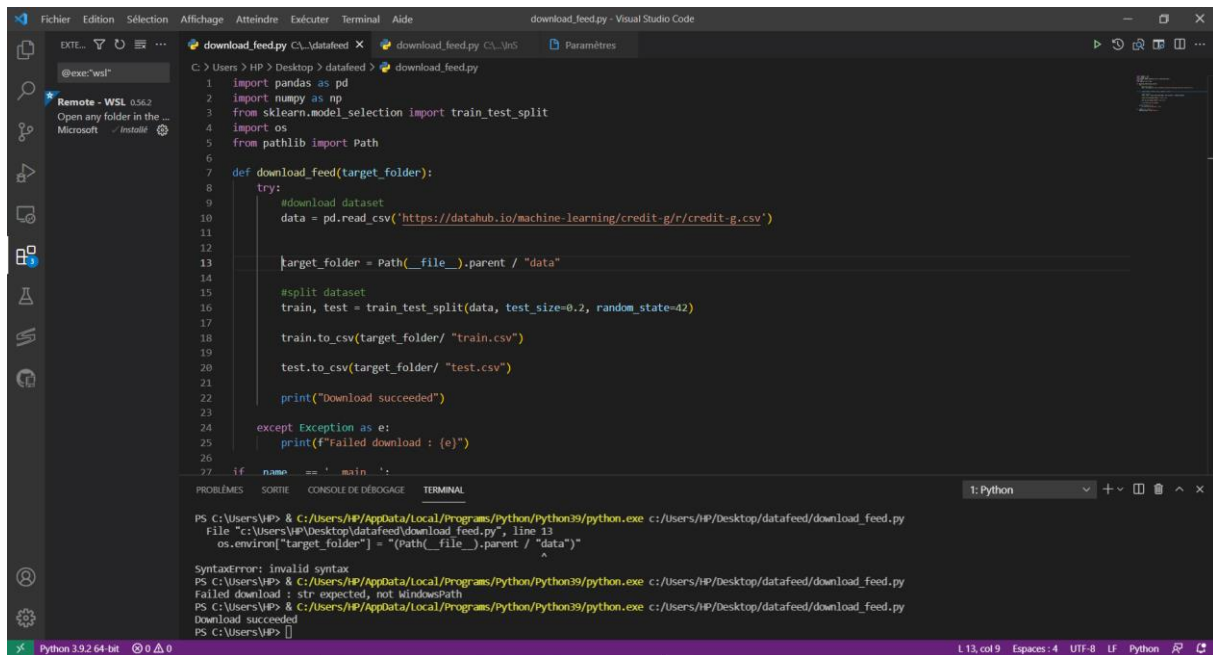


Figure 7 Interface Jupyter

➤ Visual Studio Code

Visual Studio Code est un éditeur de code extensible développé par Microsoft pour Windows, Linux et MacOS. Les fonctionnalités incluent la prise en charge du débogage, la mise en évidence de la syntaxe, la complétion intelligente du code, les snippets, la refactorisation du code et Git intégré.

Nous l'avons utilisé pour éditer le code final de notre application web car il est simple à utiliser et dispose de nombreuses bibliothèques intégrées qui facilite la programmation.



```
download_feed.py
1 import pandas as pd
2 import numpy as np
3 from sklearn.model_selection import train_test_split
4 import os
5 from pathlib import Path
6
7 def download_feed(target_folder):
8     try:
9         #download dataset
10         data = pd.read_csv('https://datahub.io/machine-learning/credit-g/r/credit-g.csv')
11
12         #target_folder = Path(_file_).parent / "data"
13
14         #split dataset
15         train, test = train_test_split(data, test_size=0.2, random_state=42)
16
17         train.to_csv(target_folder/ "train.csv")
18         test.to_csv(target_folder/ "test.csv")
19
20         print("Download succeeded")
21
22     except Exception as e:
23         print(f"Failed download : {e}")
24
25
26
27 if __name__ == '__main__':
28     pass
```

```
PS C:\Users\HP> & C:\Users\HP\AppData\Local\Programs\Python\Python39\python.exe c:\Users\HP\Desktop\datafeed/download_feed.py
File "c:\Users\HP\Desktop\datafeed/download_feed.py", line 13
    os.environ["target_folder"] = "(Path(_file_).parent / "data")"
                                     ^
SyntaxError: invalid syntax
PS C:\Users\HP> & C:\Users\HP\AppData\Local\Programs\Python\Python39\python.exe c:\Users\HP\Desktop\datafeed/download_feed.py
Failed download : str expected, not WindowsPath
PS C:\Users\HP> & C:\Users\HP\AppData\Local\Programs\Python\Python39\python.exe c:\Users\HP\Desktop\datafeed/download_feed.py
Download succeeded
PS C:\Users\HP>
```

Figure 8 Interface Visual Studio Code

3. Bibliothèques et Framework utilisés

Pour notre projet de Machine Learning, nous avons dû utiliser de nombreuses bibliothèques et Framework à savoir :

➤ Pandas

Pandas est un outil rapide, puissant, flexible et facile à utiliser pour l'analyse et la manipulation des données. Il est open source, construit en plus du langage de programmation Python.



Figure 9 Pandas[5]

➤ Numpy

Numpy est une bibliothèque pour langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.



Figure 10 Numpy[5]

➤ Le framework Scikit-learn

Scikit-learn est une framework libre Python destinée à l'apprentissage automatique. Elle est développée par de nombreux contributeurs notamment dans le monde académique par des instituts français d'enseignement supérieur et de recherche comme Inria.

Elle propose dans son framework de nombreuses bibliothèques d'algorithmes à implémenter, clé en main. Ces bibliothèques sont à disposition notamment des data scientist.

Elle comprend notamment des fonctions pour estimer des forêts aléatoires, des régressions logistiques, des algorithmes de classification, et les machines à vecteurs de support

➤ FastAPI

Pour réaliser et construire notre API, nous nous sommes tournés essentiellement vers le framework python FastAPI. FastAPI est un cadre web moderne, rapide (haute performance) pour la construction d'API avec Python 3.6. C'est grâce à ce Framework que nous allons construire notre API final. Ses principales caractéristiques sont les suivantes [2] :

- **Rapide:** Très performant, à égalité avec **NodeJS** et **Go** (grâce à Starlette et Pydantic).
- **Rapide au code:** Augmentez la vitesse de développement des fonctionnalités d'environ 200% à 300%. *
- **Moins de bogues:** Réduisez environ 40 % des erreurs induites par l'homme (développeur).
- **Intuitive:** Grand support de l'éditeur. Achèvement partout. Moins de temps à débogage.
- **Facile:** Conçu pour être facile à utiliser et à apprendre. Moins de temps à lire des documents.
- **Court :** Réduisez au minimum la duplication du code. Fonctionnalités multiples de chaque déclaration de paramètre. Moins de bugs.
- **Robuste:** Obtenez du code prêt pour la production. Avec documentation interactive automatique.

- **Basé sur les normes:** Basé sur (et entièrement compatible avec) les normes ouvertes pour les API : OpenAPI (précédemment connu sous le nom de Swagger) et JSON

Dans un second temps, nous avons utilisé l'environnement linux plus précisément la distribution Ubuntu pour faire tourner notre API, car la majorité des API tournent sur des serveurs linux.

4. Hardware utilisé

Pour exécuter notre travail de sa genèse jusqu'à sa finalisation, nous avons utilisé :

- Un laptop : HP Pavillon Core i7 10th Generation, 16Go de RAM, et 3.6GHZ de fréquence du processeur pour la phase traitement des données, développement de l'API, et la construction des modèles
- Un Asus ROG Zephyrus 15 AMD Ryzen 9, 16 Go de RAM et 3.7GHZ de fréquence de processeur, pour l'entraînement des modèles et la recherche par grille puisque cela demande une énorme puissance de calcul.

II. Méthode

Un projet de Machine Learning ne s'aborde pas comme un projet logiciel classique. En effet la modélisation par apprentissage diffère totalement de la programmation stricte basée sur des règles et exceptions. Il en va donc naturellement de même sur la façon d'aborder ce type de projet. Pour autant une méthode 100% agile ne conviendra pas nécessairement, du moins pas à toutes les étapes. [20]

Ce n'est pas un projet de développement classique et il a donc ses propres contraintes mais surtout il aura besoin d'une grande souplesse et de réajustements réguliers.

Réussir son projet de Machine Learning revient donc à respecter les étapes ci-dessous:

- Définition des objectifs
- Acquisition et description du dataset
- Analyse exploratoire des données
- Préparation des données (preprocessing)
- Modélisation
- Evaluation et scoring (Itération)
- Déploiement

1. La définition des objectifs et la typologie de problème

Avant de s'engager dans ce type de projet, il faut définir ses objectifs dès le début et identifier dans quelle typologie de problème nous nous trouvons (classification ou régression, apprentissage supervisé ou non supervisé).

L'identification de sa typologie de problème revient à analyser l'approche que nous devons employer pour la résolution du problème.

2. L'acquisition et la description du dataset

Après avoir défini nos objectifs et notre typologie de problème, nous devons faire l'acquisition d'un dataset qui correspond à cette typologie. Les dataset sont établis en général suivant la typologie de problème qu'on a en face de nous. L'acquisition étant faite, il faudra faire une description du dataset pour comprendre les terminologies employées dans celui-ci. La description du dataset est importante car elle permettra plus tard à comprendre le travail effectué.

3. L'analyse exploratoire des données.

L'exploration des données encore appelé data mining, a pour objet l'extraction d'un savoir ou d'une connaissance à partir de grandes quantités de données, par des méthodes automatiques ou semi-automatiques. Dans cette partie indispensable à la réalisation d'un projet de machine Learning, nous présenterons les répartitions et les comportements des données afin de mieux comprendre leur distribution et leur relation avec la variable cible. En sommes dans cette partie, nous allons :

- Faire un inventaire de nos données (types de données): numériques, catégorielles, nombre d'observations(lignes), nombre de features/variables (nombre de colonnes)
- Détecter les outliers
- Détecter les valeurs manquantes
- Visualiser les distributions des features afin de comprendre les données etc...

4. La préparation des données

Le prétraitement des données est une étape intégrante de l'apprentissage automatique, car la qualité des données et les informations utiles qui peuvent en être dérivées affectent directement la capacité d'apprentissage de notre modèle; par conséquent, il est extrêmement important que nous prétraitions nos données avant de les alimenter dans notre modèle. Cette phase se fait suivant plusieurs étapes :

4.1.Feature split

Le but de ceci, est de permettre une meilleure interprétation des données par les modèles qu'on utilisera dans la suite.

4.2.L'imputation des valeurs manquantes

L'imputation des données manquantes revient à "réparer" le jeu de données pour qu'il puisse être utilisable par les algorithmes de Machine Learning. La réparation d'un jeu de données peut prendre plusieurs formes : Comme supprimer les données manquantes ou les remplacer les valeurs manquantes par des valeurs artificielles (on parle d'imputation).

4.3.Le traitement des valeurs aberrantes

En statistique, une donnée aberrante est une valeur ou une observation qui est « distante » des autres observations effectuées sur le même phénomène, c'est-à-dire qu'elle contraste grandement avec les valeurs « normalement » mesurées. Il faudra donc mettre sur pied une stratégie pour traiter ces valeurs

4.4.La mise à l'échelle des données ou features scaling

La mise à l'échelle des fonctionnalités ou feature scaling, est une méthode utilisée pour normaliser la gamme de variables ou de fonctionnalités indépendantes des données. Dans le traitement des données, il est également connu sous le nom de normalisation des données et est généralement effectué pendant l'étape de prétraitement des données. Etant donné que toutes les données dans un dataset ne peuvent pas être à la même échelle, il faut les ramener à la même échelle tout en conservant les écarts entre les variables. De toutes les méthodes disponibles, les plus courantes sont les suivantes :

- **La normalisation** : c'est une technique d'échelle dans laquelle les valeurs sont décalées et recalculées de sorte qu'elles finissent par se situer entre 0 et 1. Il est également connu sous le nom de mise à l'échelle Min-Max,
- **La standardisation** : c'est une autre technique d'échelle où les valeurs sont centrées autour de la moyenne avec un écart type unitaire. Cela signifie que la moyenne de l'attribut devient nulle et que la distribution qui en résulte a un écart type unitaire,
- **Le RobustScaler** : il est similaire à la normalisation mais il utilise plutôt la plage interquartile, de sorte qu'il est robuste pour les valeurs aberrantes.

4.5.L'encodage des données

La présence de ces variables dans les données complique généralement l'apprentissage. En effet, la plupart des algorithmes d'apprentissage automatique prennent des valeurs numériques en entrée. Ainsi, il faut trouver une façon de transformer nos modalités en données numériques. De plus, la façon dont cette transformation est opérée est très importante. En effet, le codage des variables catégoriques nuit généralement à la performance des algorithmes d'apprentissage. Un codage peut s'avérer plus judicieux qu'un autre. Ainsi, nous distinguons de nombreuses techniques d'encodage des variables catégorielles (One hot encoding, label encoding, hand encoding, binary encoding etc...).

4.6.L'équilibrage des données

Pour ne pas avoir des prédictions privilégiant une classe au lieu d'une autre, nous devons équilibrer les données de notre dataset et pour ce faire nous devons d'abord au préalable vérifier s'il y a déséquilibre.

4.7.Le fractionnement de l'ensemble de données

Pour former n'importe quel modèle d'apprentissage automatique quel que soit le type d'ensemble de données utilisé, vous devez diviser l'ensemble de données en données de formation, en données de test et en données de validation. Nous comprenons donc que le split est une méthode de division d'un dataset en plusieurs sous dataset.



Figure 11 Fractionnement des données [20]

Dans un premier temps nous devons effectuer un split train-test et puis un split test-validation. Voyons en détail ce que c'est que ces différents nouveaux dataset.

- **Train set** : Ensemble de données utiliser pour entrainer les modèles
- **Test set** : Ensemble de données pour mesurer la performance de nos modèles sur des données non apprises lors de l'entraînement. La performance d'un modèle sur l'ensemble de test correspond à une mesure de ce modèle sur des données réelles, qui permet d'évaluer la capacité de généralisation du modèle.

- **Validation set :** Pour déterminer les meilleurs hyperparametres de nos modèles. Avec ce jeu de données, nous allons ainsi rechercher le meilleur paramétrage de nos modèles sans pour autant nous servir de notre ensemble de test.

5. La modélisation des données

La modélisation est le processus au cours duquel nous cherchons à représenter les données sous formes mathématique en utilisant les algorithmes de machine Learning. La modélisation des données, est une étape délicate ; elle doit être bien faite pour que les algorithmes utilisés comprennent bien les données réelles et ainsi faire une meilleure prédiction. Cette étape se divise en différentes sous-étapes : le choix de l’algorithme, le réglage des hyperparametres, etc...

6. Le choix du modèle et les benchmarks

A cette étape, nous allons évaluer les différents modèles suivant différentes métriques de machine Learning telles que :

- La matrice de confusion,
- L’accuracy,
- La précision,
- Le rappel,
- Le f1_score. [21]

Notons tout de même que grâce à la matrice de confusion nous pouvons déterminer les autres métriques. Après cela, nous devons choisir les modèles ayant la meilleure performance en fonction des métriques que nous avons retenues.

7. Le déploiement

Il s’agit de la phase finale de la méthodologie d’un problème de machine Learning. A cette étape il est question de mettre le modèle en production soit en construisant une API qui est basée sur toute le pipeline ou en construisant une application.

Ainsi, après avoir présenté notre méthodologie de travail, nous pouvons passer à la phase de conception basée sur cette méthodologie et présenter par la suite les résultats finaux.

CHAPITRE 4 : CONCEPTION ET IMPLEMENTATION

I. L'analyse fonctionnelle du besoin de notre système

L'analyse fonctionnelle est une étape importante pour la conception d'un système d'où l'importance pour nous de la faire.

Le diagramme bête à corne

Un diagramme bête à cornes est un outil pour l'analyse fonctionnelle du besoin. C'est un schéma qui démontre si le produit est utile pour l'utilisateur, s'il répond à ses besoins [9]. Pour se faire, il faut répondre aux questions suivantes : « à qui le produit rend-il service ? », « sur quoi agit-il ? », « dans quel but ? ». Pour notre système, nous avons notre diagramme suivant :

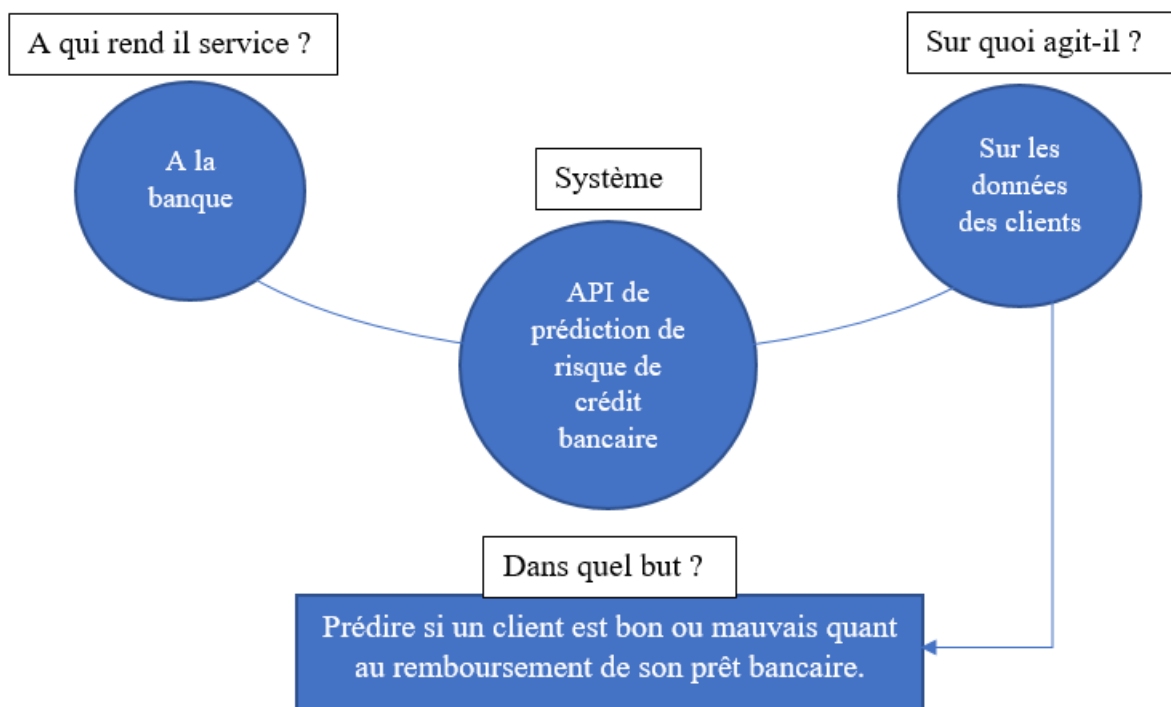


Figure 12 Diagramme bête à corne

II. La conception du système

Pour concevoir notre système nous sommes obligés de suivre la méthodologie présentée dans le chapitre 3, car elle est propre à tout projet de machine Learning.

1. Objectifs et typologie de problème

Dans notre problématique actuelle qui est celle de mettre sur pied un système de prédiction des bon et des mauvais clients bancaires quant au remboursement de leur prêt, il est donc nécessaire de comprendre dans un premier temps le problème posé.

Mettons-nous dans une explication concrète, nous voulons savoir si un client est bon ou mauvais en fonction de ses différentes informations fournies lors de sa demande de crédit bancaire. Pour mettre sur pied un tel système, il faut au préalable avoir une base de connaissances encore appelée base d'exemple. Cette base de connaissance ou dataset est la pierre angulaire de tout projet de data science. Dans notre cas actuel, nous voulons prédire une valeur bien ciblée ce qui nous met dans le cas d'une approche d'apprentissage supervisé. Rappelons que, l'apprentissage supervisé est l'apprentissage dans lequel nous avons une base de connaissance avec une variable à atteindre(cible) communément appelé Target value. On veut prédire Y (la cible) en connaissant les valeurs de X ; dans ce cas d'apprentissage supervisée les données sont labélisées

S'agissant d'une approche supervisée, nous pouvons donc déduire qu'il s'agit d'une classification car la variable cible est classé en catégorie (bon et mauvais clients). En somme, nous avons déjà compris dans quel contexte nous sommes : Apprentissage Supervisé et classification. Connaissant déjà dans quelle approche et quelle typologie de problèmes à résoudre nous sommes, nous pouvons donc nous baser sur ces informations pour nous fixer des objectifs à atteindre dans notre projet.

Comme objectifs nous avons :

- Avoir un accuracy supérieur à 80% et une matrice de confusion équilibrée entre les deux classes (bon et mauvais). Nous allons revenir sur ces notions dans la section métrique d'évaluation plus tard
- Développer notre API suivant les modèles ayant les meilleures performances.

Tableau 2 Typologie de problème et objectifs fixés

Typologie de problème	Objectifs fixés
✓ Apprentissage supervisé	✓ Accuracy supérieur à 80%
✓ Classification	✓ Bonne matrice de confusion
	✓ API avec les meilleurs performances

Maintenant que les objectifs sont définis, nous devons passer à l'une des phases les plus importantes : l'acquisition du dataset

2. Acquisition et description des données

2.1.Acquisition

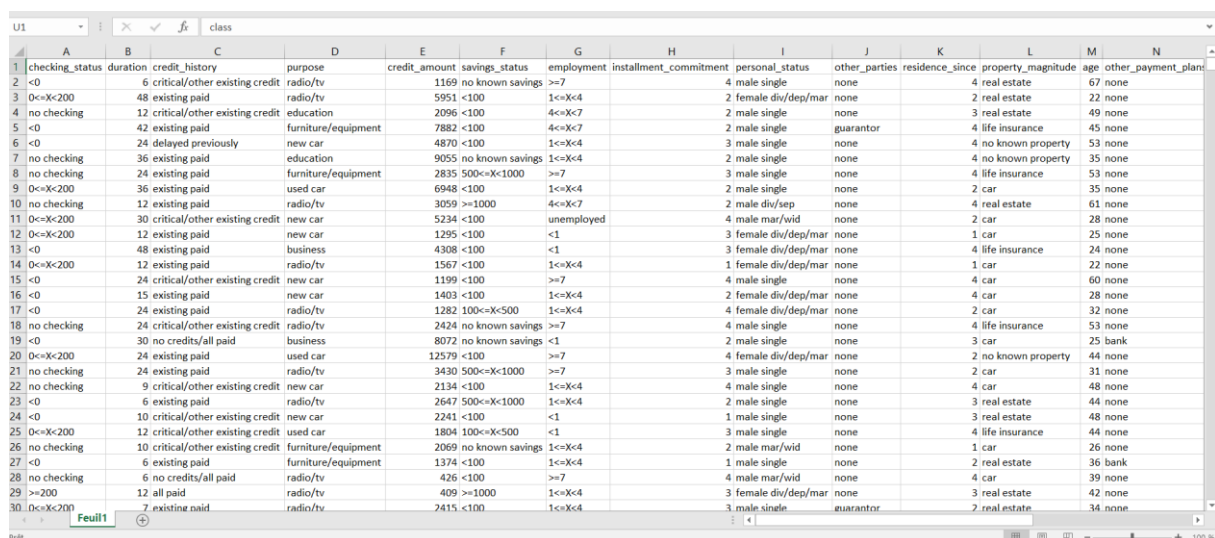
La seconde étape de l'étude consiste à récupérer des données cohérentes afin de former la base de données sur laquelle le modèle de prédiction sera appris. Notons que les données sont le cœur même de notre travail et la sélection de bonnes données est impératifs.

Etant donné que nous sommes dans une Fintech et non dans une banque, nous n'avons pas pu avoir accès aux données d'une banque camerounaise car cela est très confidentiels. Nous avons donc récolté des données bancaires publiques. Ces données ont été récupéré sur le site www.datahub.io. Ce site est une plateforme offrant de nombreuses bases de données ou dataset accessible de façon gratuite et suivant toute catégorie.

Les données qui nous intéressent sont des données des clients bancaires ayant demandé ou effectué un prêt bancaire. Ainsi nous avons choisi le dataset nommé « credit-g.csv » qui est toujours disponible sur ce site web à l'adresse datahub.io/machine-learning/credit-g. ou vous pouvez le télécharger. Le format du dataset est .csv (qui est une extension des tableurs avec virgules)

2.2.Description du dataset

Le dataset « credit-g.csv », contient les données de crédit d'une banque allemande. Il classe les personnes décrites par un ensemble d'attribut comme de bons ou de mauvais risques de crédit. Ce dataset a été publié et conçu par le Dr Hans Hofmann en 1994 (source : **UCI Machine Learning Repository: Citation Policy**).



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	checking_status	duration	credit_history	purpose	credit_amount	savings_status	employment	installment_commitment	personal_status	other_parties	residence_since	property_magnitude	age	other_payment_plan
2	<0	6	critical/other existing credit	radio/tv	1169	no known savings	>=7	4	male single	none	4	real estate	67	none
3	0<=X<200	48	existing paid	radio/tv	5951	<100	1<=X<4	2	female div/dep/mar	none	2	real estate	22	none
4	no checking	12	critical/other existing credit	education	2096	<100	4<=X<7	2	male single	none	3	real estate	49	none
5	<0	42	existing paid	furniture/equipment	7882	<100	4<=X<7	2	male single	guarantor	4	life insurance	45	none
6	<0	24	delayed previously	new car	4870	<100	1<=X<4	3	male single	none	4	no known property	53	none
7	no checking	36	existing paid	education	9055	no known savings	1<=X<4	2	male single	none	4	no known property	35	none
8	no checking	24	existing paid	furniture/equipment	2835	500<=X<1000	>=7	3	male single	none	4	life insurance	53	none
9	0<=X<200	36	existing paid	used car	6948	<100	1<=X<4	2	male single	none	2	car	35	none
10	no checking	12	existing paid	radio/tv	3059	>=1000	4<=X<7	2	male div/sep	none	4	real estate	61	none
11	0<=X<200	30	critical/other existing credit	new car	5234	<100	unemployed	4	male mar/wid	none	2	car	28	none
12	0<=X<200	12	existing paid	new car	1295	<100	<1	3	female div/dep/mar	none	1	car	25	none
13	<0	48	existing paid	business	4308	<100	<1	3	female div/dep/mar	none	4	life insurance	24	none
14	0<=X<200	12	existing paid	radio/tv	1567	<100	1<=X<4	1	female div/dep/mar	none	1	car	22	none
15	<0	24	critical/other existing credit	new car	1199	<100	>=7	4	male single	none	4	car	60	none
16	<0	15	existing paid	new car	1403	<100	1<=X<4	2	female div/dep/mar	none	4	car	28	none
17	<0	24	existing paid	radio/tv	1282	100<=X<500	1<=X<4	4	female div/dep/mar	none	2	car	32	none
18	no checking	24	critical/other existing credit	radio/tv	2424	no known savings	>=7	4	male single	none	4	life insurance	53	none
19	<0	30	no credits/all paid	business	8072	no known savings	>=7	2	male single	none	3	car	25	bank
20	0<=X<200	24	existing paid	used car	12579	<100	>=7	4	female div/dep/mar	none	2	no known property	44	none
21	no checking	24	existing paid	radio/tv	3430	500<=X<1000	>=7	3	male single	none	2	car	31	none
22	no checking	9	critical/other existing credit	new car	2134	<100	1<=X<4	4	male single	none	4	car	48	none
23	<0	6	existing paid	radio/tv	2647	500<=X<1000	1<=X<4	2	male single	none	3	real estate	44	none
24	<0	10	critical/other existing credit	new car	2241	<100	<1	1	male single	none	3	real estate	48	none
25	0<=X<200	12	critical/other existing credit	used car	1804	100<=X<500	<1	3	male single	none	4	life insurance	44	none
26	no checking	10	critical/other existing credit	furniture/equipment	2069	no known savings	1<=X<4	2	male mar/wid	none	1	car	26	none
27	<0	6	existing paid	furniture/equipment	1374	<100	1<=X<4	1	male single	none	2	real estate	36	bank
28	no checking	6	no credits/all paid	radio/tv	426	<100	>=7	4	male mar/wid	none	4	car	39	none
29	>=200	12	all paid	radio/tv	409	>=1000	1<=X<4	3	female div/dep/mar	none	3	real estate	42	none
30	0<=X<200	7	existing paid	radio/tv	2415	<100	1<=X<4	3	male single	guarantor	2	real estate	34	none

Figure 13 Notre dataset

Ce dataset contient les données des clients, classées par catégories en fonction de plusieurs attributs rangés sur les colonnes et les clients rangés sur les lignes. Jetons un coup d'œil à une partie de ce dataset et décrivons ses différents attributs.

Dans ce dataset nous avons au total 1000 lignes qui représentent les clients et nous avons 20 colonnes qui représentent les attributs des clients.

Rappelons que nous sommes en apprentissage supervisé, et nous avons donc une variable cible Y à prédire et des variables X nécessaires à la prédiction.

Avant de spécifier quel est la variable cible, décrivons d'abord tous les attributs (colonnes) de notre dataset. Ce qui suit est classé suivant l'ordre fournit par le dataset d'origine :

- Checking_status : Il s'agit du statut du compte courant et il est exprimé en Point allemand
- Credit_history : Il s'agit de la durée en mois du remboursement du crédit demandé.
- Purpose : Il s'agit des antécédents de crédit (crédit pris, remboursés dûment, retards, comptes critiques)
- Credit_amount : Il s'agit du montant du credit demandé (exprimé en Euros)
- Savings_status : Il s'agit du statut du compte d'épargne/ obligations exprimé en Deutsche mark
- Employment : Il s'agit du nombre d'années passée dans son emploi actuel
- Installment_commitment : Il s'agit du taux d'acompte en pourcentage du revenu disponible
- Personal status : Il s'agit du statut personnel du client (marié, célibataire, ...) et de son sexe (homme, femme)
- Other_parties : Autre garants/ débiteur
- Residence_since : Nombre d'année passée dans sa résidence actuelle
- Property_magnitude : propriétés
- Age : Age du client en années
- Other_payment : Autres plans d'acompte (banques, magasins)
- Housing : Etat de son logement (loyer, propre, ...)
- Existing_credit : nombre de crédit existant que le client a dans cette banque
- Job : Le travail actuel du client
- Num_dependent : le nombre de personnes susceptibles d'assurer l'entretien du compte
- Own_telephone : Existence d'un téléphone (oui, non)

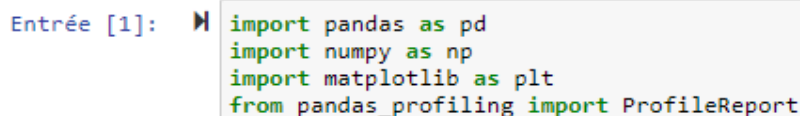
- Foreign_worker : Travailleur étranger (oui, non)
- Class : Classification des clients (bon, mauvais)

Après avoir défini et donner la signification de chacun des attributs, nous pouvons maintenant déterminer qui est la variable cible et dans notre problématique, puisqu'on veut déterminer les bons et les mauvais risques de crédit pour chaque client, il est donc logique que notre variable cible est $Y = \text{class}$ et nos variables X sont le reste des attributs.

Après avoir identifié et présenté notre dataset, nous pouvons maintenant passer à la phase d'analyse et de l'exploration des données contenues dans ce dataset afin de bien comprendre les données qui y sont répertoriées en fonction de notre problématique.

3. Analyse Exploratoire des Données (EDA)

Avant de procéder à ces différentes étapes, nous avons décider garder 10 lignes de notre dataset qui nous permettra de faire les tests plus tard et ainsi nous travaillons avec un dataset de 990 lignes à présent. Ensuite il faut au préalable ouvrir son environnement de programmation et dans notre cas nous avons fait cette partie avec jupyter notebook. Après son ouverture il faut importer toutes les bibliothèques nécessaires pour faire cette analyse. Dans l'image qui suit, nous présentons l'importation des bibliothèques nécessaire à cette partie de notre travail :

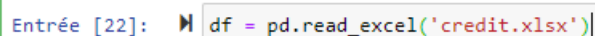


```
Entrée [1]: import pandas as pd
import numpy as np
import matplotlib as plt
from pandas_profiling import ProfileReport
```

Figure 14 Importation de librairie pour l'exploration des données

3.1. Inventaire des données

Pour faire l'inventaire des données, nous devons d'abord importer notre dataset initial. Et pour l'importer on procède comme suit :



```
Entrée [22]: df = pd.read_excel('credit.xlsx')
```

Figure 15 Importation du dataset

Après l'importation du dataset, nous pouvons procéder à l'inventaire de nos données. Pour visualiser le type de nos différentes données, on fait appel à la commande **df.info()** (df est le nouveau nom attribué à notre dataset lors de son importation) et grâce à elle on aura le type de nos différentes features ainsi que le nombre de features et le nombre de lignes.

Ainsi nous avons :

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 990 entries, 0 to 989
Data columns (total 23 columns):
#   Column                Non-Null Count  Dtype
---  -
0   id                    990 non-null   int64
1   checking_status       990 non-null   object
2   duration              990 non-null   int64
3   credit_history        990 non-null   object
4   purpose               990 non-null   object
5   credit_amount         990 non-null   int64
6   savings_status        990 non-null   object
7   employment            990 non-null   object
8   installment_commitment 990 non-null   int64
9   sex                   990 non-null   object
10  status_matrimonial    990 non-null   object
11  other_parties         990 non-null   object
12  residence_since       990 non-null   int64
13  property_magnitude   990 non-null   object
14  age                   990 non-null   int64
15  other_payment_plans   990 non-null   object
16  housing               990 non-null   object
17  existing_credits      990 non-null   int64
18  job                   990 non-null   object
19  num_dependents        990 non-null   int64
20  own_telephone         990 non-null   object
21  foreign_worker        990 non-null   object
22  class                 990 non-null   object
dtypes: int64(8), object(15)
memory usage: 178.0+ KB
```

Figure 16 inventaire des données

Avec cette image, nous pouvons voir que nous avons un total de 990 cas (lignes) et de 21 features en excluant la feature « id ». Nous remarquons aussi le type des différentes features (colonnes), certaines sont numériques(int64), d'autres sont catégorielles(Object). Ainsi, nous pouvons donc répertorier 7 variables numériques et 14 variables catégorielles. Mais nous devons identifier les variables catégorielles ordinales et nominales aussi.

Une variable nominale est une variable qui n'a pas de valeur quantitative et ne suit pas d'ordre spécifique tandis qu'une variable ordinale suit un ordre bien définie. Dans le tableau suivant nous avons répertoriés les différentes variables avec leur catégorie

Tableau 3 Type des données

Variables numériques	Variables catégorielles
Duration, Credit_amount, Installment_commitment Residence_since age Existing_credits Num_dependent	➤ Ordinales: Own_telephone Foreign_worker, class, Other_parties, employment, Savings_status, sex, Checking_status ➤ Nominales: Credit_history, purpose, status_matrimonial, Property_magnitude, other_payment_plans, housing, job

Après avoir déterminé le type des données ainsi que le nombre de ligne et de colonnes, nous pouvons nous attaquer à la détection des valeurs manquantes.

3.2. Détection des valeurs manquantes

Pour la détection des valeurs manquantes dans notre dataset, nous pouvons procéder de plusieurs façons : par visualisation des graphes ou par visualisation du résultat de `df.info()` précédent.

Pour notre travail nous avons utilisé une visualisation par graphique. Gardons à l'esprit que notre dataset comporte 990 lignes ce qui signifie que chaque features devraient avoir aussi 990 lignes. Dans la pratique, les valeurs manquantes sont détectées par des N/A mais il peut arriver qu'on a à faire à des valeurs manquantes sans s'en rendre compte. Comment donc prendre en compte toutes les valeurs manquantes qui existent pour être sûr de bien les détecter ?

Pour cela nous avons défini un dictionnaire de valeurs manquantes à l'intérieur duquel nous avons répertoriés les différentes valeurs manquantes qui existent.

```
missing_values = ["n/a", "na", "--", "NaN", "nan", "N/A"]
```

Figure 17 Dictionnaire des valeurs manquantes

Après avoir créé notre dictionnaire des valeurs manquantes, nous pouvons maintenant visualiser s'il y a des valeurs manquantes grâce à la bibliothèque pandas profiling qu'il faudra d'abord importer.

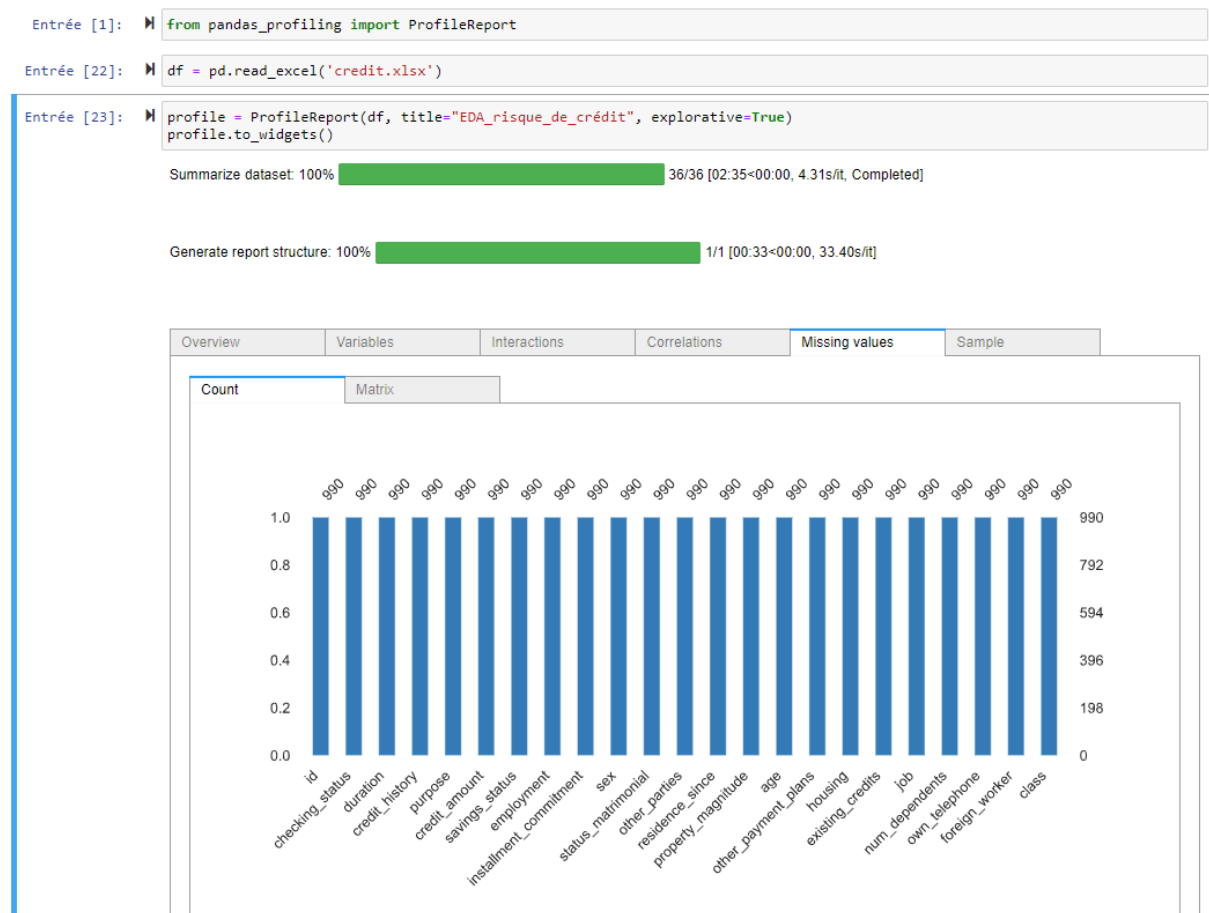


Figure 18 Détection des valeurs manquantes

Après avoir généré ce graphe, nous pouvons voir que, toutes les features ont 990 lignes, ce qui veut dire qu'aucune valeur manquante n'a été détecté. Mais au cas où nous aurions eu à faire à des valeurs manquantes, il est important de mettre sur pied des stratégies de traitement de ces valeurs manquantes, et nous verrons cela dans la partie prétraitement des données

Ainsi, nous pouvons passer à la détection des valeurs aberrantes encore appelées outliers.

3.3. La détection des valeurs aberrantes

Il convient d'examiner les valeurs aberrantes car elles peuvent fournir des informations utiles sur les données ou le procédé. Bien souvent, il est plus facile d'identifier les valeurs aberrantes en représentant les données sur un graphique.

Avant de procéder à cette détection, il est nécessaire de diviser son dataset original en deux dataframe distinct : un dataframe qui va contenir les valeurs numériques et l'autre dataframe qui va contenir les valeurs catégorielles. Et ceci a été fait comme suit :

```
Entrée [17]: # création de deux nouveaux dataframe cat et num
cat = df[[
    'checking_status', 'employment', 'credit_history', 'purpose',
    'savings_status', 'sex', 'status_matrimonial', 'other_parties',
    'property_magnitude', 'other_payment_plans', 'housing', 'job',
    'own_telephone', 'foreign_worker', 'class'
]]
num = df[[
    'duration', 'credit_amount', 'installment_commitment', 'residence',
    'age', 'existing_credits', 'num_dependents'
]]

Entrée [19]: print(cat.shape)
print(num.shape)

(990, 15)
(990, 7)
```

Figure 19 Dataframe Cat et Num

Pour détecter et identifier les valeurs aberrantes de notre dataset, nous avons utilisé les boxplot ou boîtes à moustaches de la bibliothèque Matplotlib de python. Rappelons que cette étape est appliquée uniquement aux variables numériques d'où la nécessité de créer un dataframe de variables numériques avec l'image précédente.

➤ Prenons le cas de la feature « Credit_amount » et visualisons ces valeurs aberrantes :

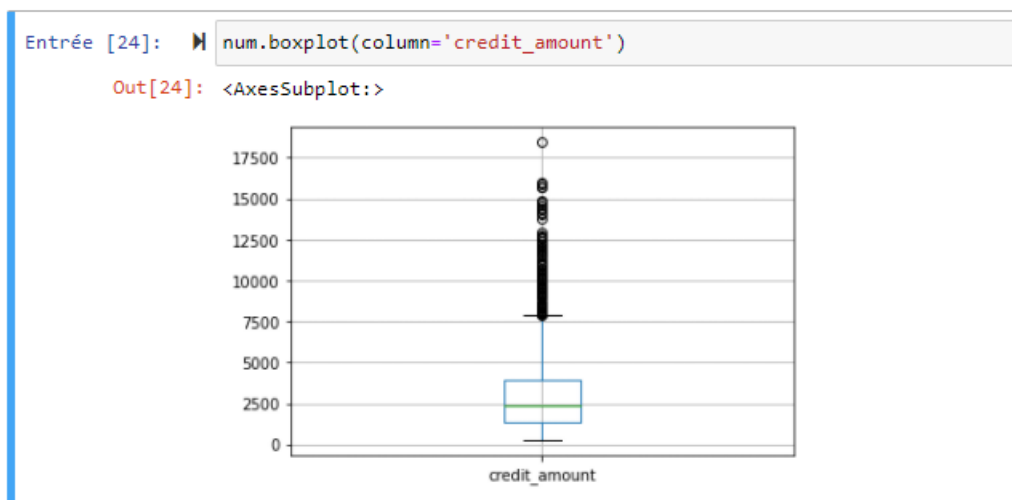


Figure 20 valeurs aberrantes de la variable Credit_amount

Nous pouvons voir sur ce graphe que nous avons de nombreuses informations qui sont loin de la moyenne de crédit et qui est détecté comme valeur aberrante. Elles sont représentées par ces pointillés noirs au-dessus de 7500 suivant l'axe y.

- Prenons ensuite le cas de la feature « duration » :

```
Entrée [36]: num.boxplot(column='duration')
```

```
Out[36]: <AxesSubplot:>
```

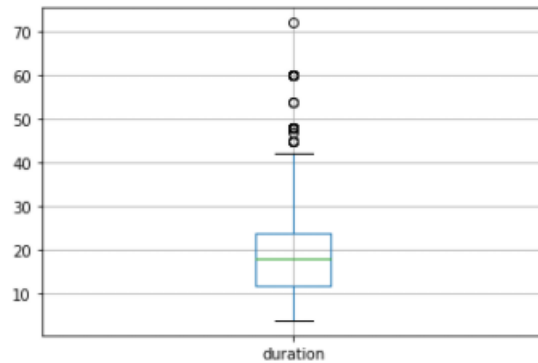


Figure 21 Valeurs aberrantes de la variable duration

Nous pouvons aussi voir que la feature « duration » a des valeurs aberrantes.

- Prenons enfin le cas de la feature « âge »

```
Entrée [27]: num.boxplot(column='age')
```

```
Out[27]: <AxesSubplot:>
```

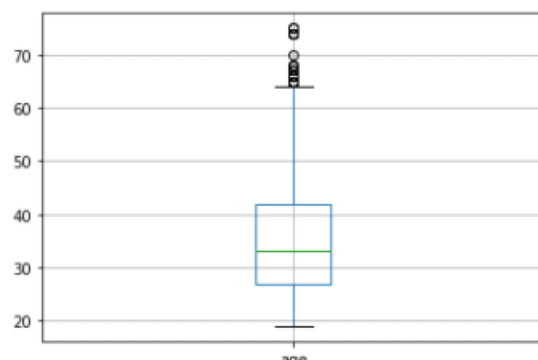


Figure 22 Valeurs aberrantes de la variable âge

Ici, nous avons aussi à faire a des valeurs aberrantes.

Quelles peuvent être les causes de ces valeurs aberrantes ?

Nous pouvons citer entre autre :

- Erreur de saisie des données

- Problème lié au procédé
- Facteur manquant
- Risque aléatoire

Comment faire pour corriger ce problème de valeurs aberrantes ? Nous répondrons à cette question dans la phase de prétraitement des données.

Après avoir visualisé les valeurs manquantes puis aberrantes, il est primordial de visualiser les distributions des différentes features pour mieux interpréter les données de notre dataset.

3.4. La compréhension des données.

Pour mieux comprendre les données, nous avons procédé à la production d'une série de graphiques que nous avons interprétés et commentés. Pour une bonne lisibilité nous ne présenterons pas tous les graphes générés dans notre Jupyter Notebook.

L'outil pandas-profiling nous a simplifié la tâche quant au tracé des différents histogrammes des features.

3.4.1. Les histogrammes

Nous ne présenterons que 4 histogrammes parmi les 21 existant dans notre jupyter notebook

Histogramme de « checking_status »

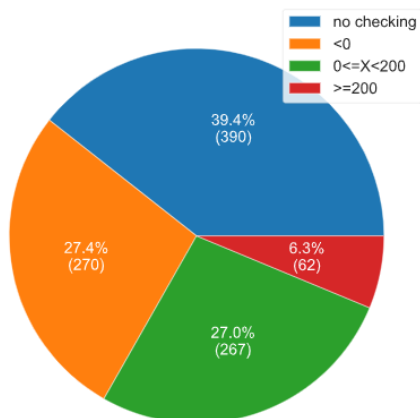


Figure 24 histogramme de la variable checking_status

Histogrammes de « credit_history »

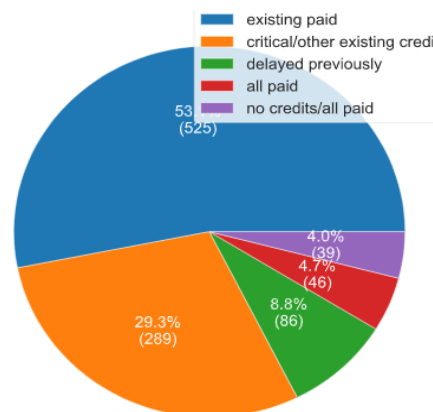


Figure 23 histogramme de la variable credit_history

➤ Histogramme credit_amount

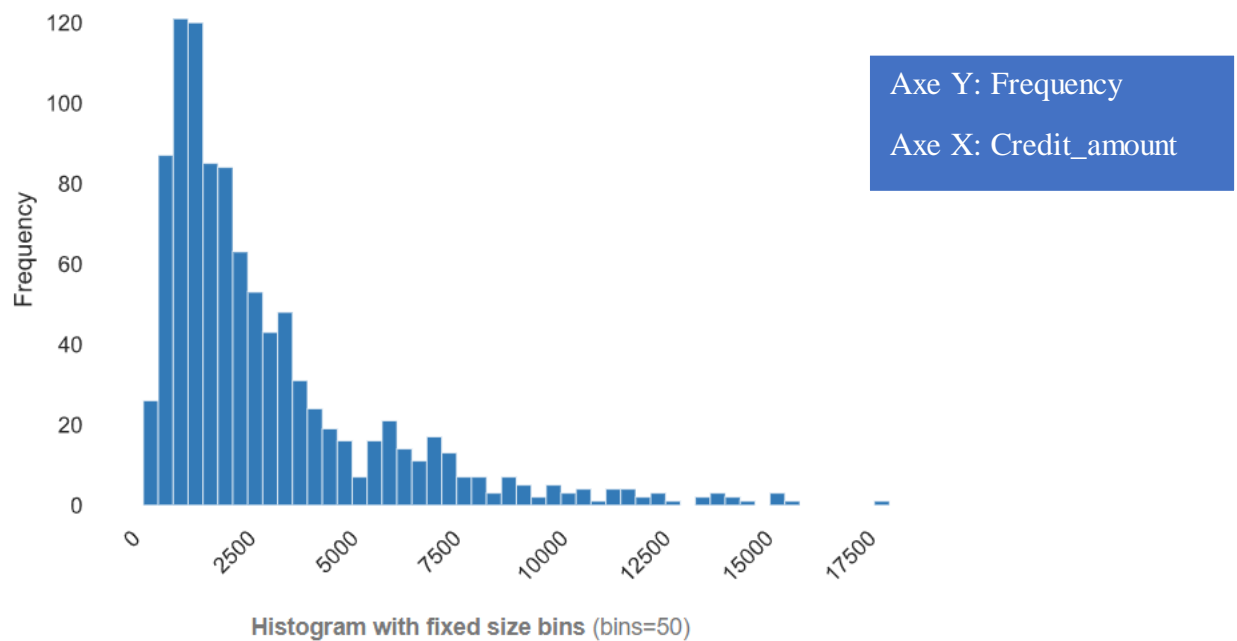


Figure 25 Histogramme de la variable credit_amount

Nous pouvons voir que le montant des crédits les plus demandé oscille entre 0 et 5000 euros.

➤ Histogramme class

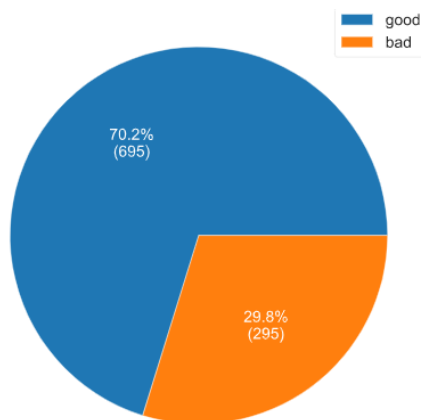


Figure 26 Histogramme de la variable class

La visualisation de l'histogramme de la variable cible est importante, car elle permettra de voir si nos données sont déséquilibrées. Visualisons son histogramme :

En observant ce graphique, nous pouvons voir que les classes good et Bad sont déséquilibrées. En travaillant avec des classes déséquilibrées, nous aurons des performances médiocres mais dans notre cas nous avons corrigés ce déséquilibre et cela sera expliqué dans la section qui lui est dédiée.

Etant donné que nous avons 21 features, nous ne pourrions pas présenter tous ces histogrammes dans ce mémoire. Pour en savoir plus nous mettrons en ligne notre jupyter notebook où la visualisation des histogrammes sera plus complètes.

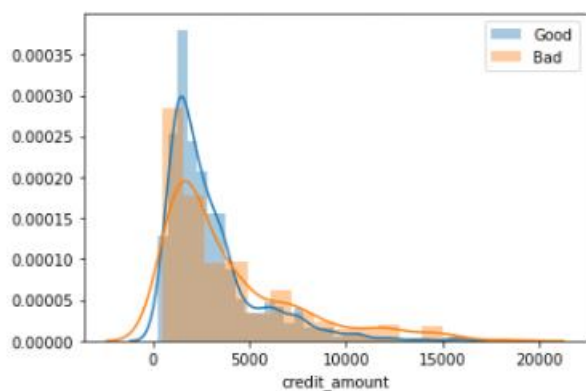
3.4.2. Les distributions

Une distribution est une fonction qui associe une fréquence d'apparition à une classe de valeur. Cette fonction permet de résumer l'information contenue dans un ensemble de données.

Pour notre dataset, nous avons tracés des distributions de certaines variables pour comprendre leur répartition. Il faut toujours garder à l'esprit que, la bibliothèques python que nous avons utilisé pour tous nos graphiques est Matplotlib.

Nous avons tracé, les distributions en fonction de la variable cible.

➤ Distribution de la variable credit_amount



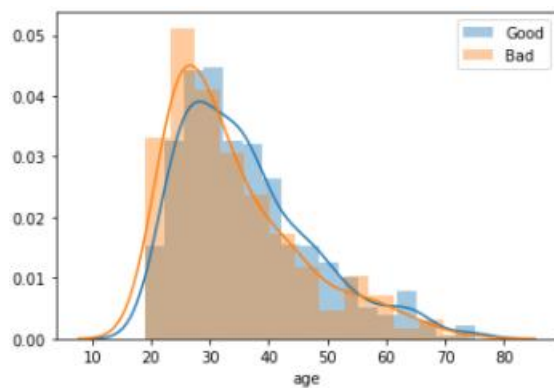
Axe Y: Pourcentage/1000

Axe X: Credit_amount

Figure 27 Distribution de la variable credit_amount en fonction de la variable class

- Le montant de la plupart de bon et de mauvais risque se situe entre 0 et 5000.
- Chevauchement important ce qui signifie que la variable cible dépend énormément du montant crédit demandé,
- Pic des bons risques de crédit est plus élevé que celui des mauvais, il semble donc que le montant moyen des bons risques de crédit soit supérieur à celui des mauvais de risque.

➤ Distribution de la variable âge



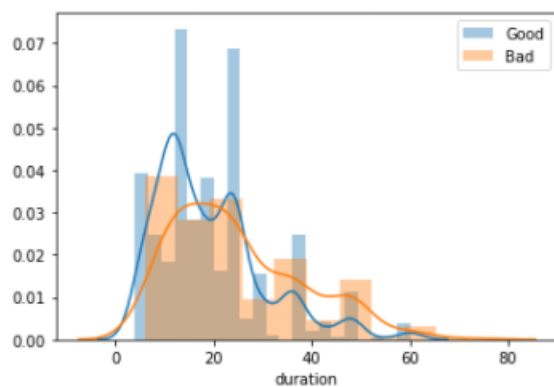
Axe Y: pourcentage/10

Axe X: age

Figure 28 Distribution âge en fonction de class

- L'âge de la plupart des bons et mauvais risques de crédit se situe dans la fourchette de 20 à 60 ans.
- Le chevauchement est important, il semble donc que l'âge moyen des dossiers de bons et de mauvais risques de crédit soit presque égal

➤ Distribution de la variable duration



Axe Y: pourcentage/10

Axe X: duration

Figure 29 Distribution duration en fonction de class

- La durée de la plupart des bons et mauvais risques de crédit se situe dans une fourchette d'environ 9 à 50 mois.
- Le chevauchement est très faible, il semble donc que la durée moyenne de bon et de mauvais risque de crédit diffère significativement

3.4.3. La corrélation

Est une mesure statistique qui exprime la notion de liaison linéaire entre deux variables (ce qui veut dire qu'elles évoluent ensemble à une vitesse constante).

Nous avons de nombreux types de corrélation (Pearson, Spearman, Kendall, Phik, etc...), mais dans notre projet, nous avons mis l'accent sur la corrélation de Pearson.

En statistique, le coefficient de corrélation de Pearson, également appelé r de Pearson, est une mesure de corrélation linéaire entre deux variables. Sa valeur se situe entre -1 et +1, -1 indiquant la corrélation linéaire négative totale, 0 indiquant aucune corrélation linéaire et 1 indiquant la corrélation linéaire positive totale. En outre, r est invariant en raison de changements distincts dans l'emplacement et l'échelle des deux variables, ce qui implique que pour une fonction linéaire l'angle à l'axe x n'affecte pas r .

Pour calculer r pour deux variables X et Y , on divise la covariance de X et Y par le produit de leurs écarts types. Dans notre cas, nous avons utilisé cette corrélation pour évaluer si la valeur d'une variable influence ou est en étroite liaison avec une autre variable.

Ceci est illustré par la figure suivante :

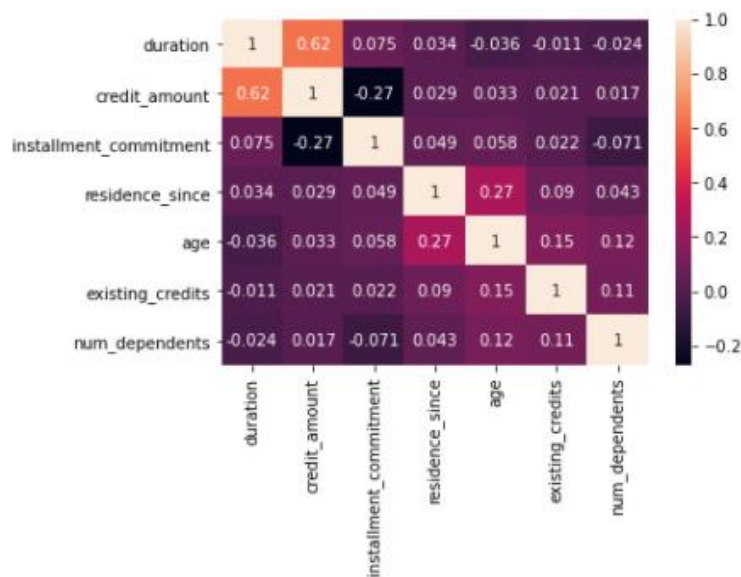


Figure 30 Corrélation entre les variables numériques

En observant cette figure encore appelé heatmap, nous pouvons voir qu'il y a une forte corrélation positive entre duration et credit_amount, une basse corrélation positive entre Residence_since et âge et une basse corrélation négative entre duration et âge. Ceci veut tout simplement dire que, le crédit demandé est fonction de l'âge du demandeur en étant donné que, plus on est âgé, plus on a des besoins.

En conclusion de cette analyse, nous pouvons dire :

- La plupart des Bad class se situe entre 20 et 60 ans

- Le montant de ces risques de mauvais crédit se situe dans la fourchette 0 à 12500 euros
- La plupart des risques de mauvais crédit ont une durée d'au moins 12 mois et peuvent aller jusqu'à 60 mois.
- La plupart de ces mauvais risques de crédit sont âgés de 20 à 60 ans
- Plus la durée de risque de crédit augmente, plus le montant du crédit augmente.

Ainsi, après avoir fini notre analyse, qui nous a permis de comprendre les données de notre dataset, nous pouvons passer à la phase suivante : l'ingénierie des fonctionnalités

4. Le prétraitement des données ou preprocessing

Le prétraitement des données est une étape intégrante de l'apprentissage automatique, car la qualité des données et les informations utiles qui peuvent en être dérivées affectent directement la capacité d'apprentissage de notre modèle; par conséquent, il est extrêmement important que nous prétraitions nos données avant de les alimenter dans notre modèle. Cette phase se fait suivant plusieurs étapes : feature split, l'imputation des valeurs manquantes, le traitement des valeurs aberrantes, la mise à l'échelle des données, l'encodage des données, l'équilibrage des données, le fractionnement de l'ensemble de données.

4.1.Feature split

Nous avons utilisé une séparation de fonctionnalités, pour pouvoir créer deux catégories grâce à l'original.

Au départ, dans notre dataset original, rappelons-nous que nous avons une colonne `Personal_status`, et maintenant le but de la manœuvre est de séparer cette colonne en deux colonnes. Nous avons choisi de séparer cette colonne en `sex` et en `status_matrimonial`. Pour le faire, nous avons procédé comme suit :

```
df[['sex','status_matrimonial']] = df['Personal_status'].str. split (" ", expand = True);
```

n'oublions pas que tout en haut nous avons déclarés `df` comme étant notre dataset.

4.2.Traitement des valeurs manquantes

Dans notre projet nous n'avons pas à faire à des valeurs manquantes, mais cela n'empêche pas de définir une stratégie adéquate pour traiter ces valeurs manquantes. Pour notre cas, nous avons opter pour l'imputation des valeurs manquantes car au moins nous allons garder de l'information que de la supprimer et la perdre.

L'imputation de données manquante réfère au fait qu'on remplace les valeurs manquantes dans le jeu de données par des valeurs artificielles. Ces valeurs artificielles peuvent être entre autre,

la moyenne des valeurs de la colonne où se trouve la valeur manquante, la médiane des valeurs de la colonne, le mode.

Etant donné que nous avons deux dataframe cat (pour les variables catégorielles) et Num (pour les variables numériques) que nous avons créés plus haut, les stratégies d'imputation seront différentes suivant le type de données qu'on a en face de nous.

- Imputation des variables numériques.

Pour traiter les variables manquantes numériques, nous avons utilisé la médiane des variables de la colonne de la variable manquante. Et ceci a été fait grâce à la méthode fillna illustrée comme suit :

```
Entrée [22]: # imputation variables numériques

num['duration'].fillna((num['duration'].median()), inplace=True)
num['credit_amount'].dropna(inplace=True)
num['installment_commitment'].fillna((num['installment_commitment'].mean()),
                                     inplace=True)
num['residence_since'].fillna((num['residence_since'].median()), inplace=True)
num['age'].fillna((num['age'].median()), inplace=True)
num['existing_credits'].fillna((num['existing_credits'].median()),
                              inplace=True)
num['num_dependents'].fillna((num['num_dependents'].median()), inplace=True)
```

Figure 31 Imputation des variables numériques

- Imputation des variables catégorielles

Pour traiter les variables manquantes catégorielles, nous avons utilisé le mode des variables de la colonne de la variable manquante (NB : Le mode est la valeur la plus fréquente dans un échantillon). Alors nous avons procédé comme suit :

```
Entrée [22]: # imputation variables numériques

num['duration'].fillna((num['duration'].median()), inplace=True)
num['credit_amount'].dropna(inplace=True)
num['installment_commitment'].fillna((num['installment_commitment'].mean()),
                                     inplace=True)
num['residence_since'].fillna((num['residence_since'].median()), inplace=True)
num['age'].fillna((num['age'].median()), inplace=True)
num['existing_credits'].fillna((num['existing_credits'].median()),
                              inplace=True)
num['num_dependents'].fillna((num['num_dependents'].median()), inplace=True)
```

Figure 32 Imputation des variables catégorielles

4.3.Traitement des valeurs aberrantes

Dans le cadre d'un projet de Machine Learning on fera souvent le choix de supprimer une valeur aberrante. En effet, pour obtenir une meilleure qualité de prédiction il est nécessaire de traiter ces données car un modèle pourra être très sensible aux données extrêmes ce qui va biaiser les

prédictions. Prenons l'exemple de la feature âge, essayons de voir comment la suppression des valeurs aberrantes est faite grâce à ce bout de code :


```
Entrée [30]:  # On supprime Les valeurs aberrantes pour La colonne âge  
# On calcule Q1  
# q1=num["age"].quantile(q=0.25)  
# On calcule Q3  
# q3=num["age"].quantile(q=0.75)  
# On calcule l'écart interquartile (IQR)  
# IQR=q3-q1  
# On calcule la borne inférieure à l'aide du Q1 et de l'écart interquartile  
#borne_inf = q1-1.5*IQR  
# On calcule la borne supérieure à l'aide du Q3 et de l'écart interquartile  
#borne_sup = q3 +1.5*IQR  
# On garde Les valeurs à l'intérieur de la borne inférieure et supérieure  
#num= num[num["age"]<borne_sup]  
# num=num[num["age"]>borne_inf]
```

Figure 33 Suppression des valeurs aberrantes

En effet, pour supprimer des valeurs aberrantes, un intervalle sera calculé et toute donnée se trouvant hors de cet intervalle se verra supprimé.

Tout de même notons que, l'écart interquartile(IQR) est une mesure de dispersion qui s'obtient en faisant la différence entre le troisième et le premier quartile et est donné par :

$$IQR = Q3 - Q1.$$

L'intervalle interquartile est ce que nous pouvons utiliser pour déterminer si une valeur extrême est effectivement une valeur aberrante. L'intervalle interquartile est basé sur une partie du résumé à cinq chiffres d'un ensemble de données, à savoir le premier quartile et le troisième quartile. Le calcul de l'intervalle interquartile implique une seule opération arithmétique. Tout ce que nous avons à faire pour trouver l'intervalle interquartile est de soustraire le premier quartile du troisième quartile. La différence qui en résulte nous indique à quel point la moitié médiane de nos données est répartie.

Multiplier l'intervalle interquartile (IQR) par 1,5 nous donnera un moyen de déterminer si une certaine valeur est une valeur aberrante. Si nous soustrayons 1,5 x IQR du premier quartile, toutes les valeurs de données inférieures à ce nombre sont considérées comme des valeurs aberrantes. De même, si nous ajoutons 1,5 x IQR au troisième quartile, toutes les valeurs de données supérieures à ce nombre sont considérées comme des valeurs aberrantes.

Notons que, les valeurs aberrantes sont seulement présentes sur les valeurs numériques. Mais dans notre cas, nous n'avons pas supprimé les valeurs aberrantes pour conserver les données, et aussi notons que les valeurs sont dites aberrantes suivant le contexte. Dans le domaine bancaire, nous ne pouvons pas considérer les valeurs renseignées comme aberrantes, car toutes ces données, qu'elles soient extrêmes ou pas, contiennent de l'information.

4.4.L'encodage des variables catégorielles

Pour notre projet nous avons utilisés le hand encoding pour les variables ordinales car nous déterminons nous même le rangement des variables suivant un ordre, et nous avons utilisé le one hot encoding pour les variables nominales.

4.4.1. Hand Encoding

L'encoding à la main, est un procédé grâce auquel nous attribuons à chaque variables une valeur de classification en fonction de son ordre. Aussi, cette approche nous permet d'avoir le contrôle sur nos données. Avec l'image qui suit, nous pouvons voir comment cela a été fait (à gauche) et le résultat qui découle (à droite) :

▼ 4.9.1 Encodage des variables ordinales

```
Entrée [37]: M cat['own_telephone'].replace({'yes': '1', 'none': '0'}, inplace=True)
cat['foreign_worker'].replace({'yes': '1', 'no': '0'}, inplace=True)
cat['class'].replace({'good': '1', 'bad': '0'}, inplace=True)
cat['other_parties'].replace(
    {
        'none': '0',
        'co_applicant': '1',
        'guarantor': '2'
    }, inplace=True)
cat['employment'].replace(
    {
        'unemployed': '0',
        '<1': '1',
        '1<=X<4': '2',
        '4<=X<7': '3',
        '>=7': '4'
    },
    inplace=True)
cat['savings_status'].replace(
    {
        'no known savings': '0',
        '<100': '1',
        '100<=X<500': '2',
        '500<=X<1000': '3',
        '>=1000': '4'
    },
    inplace=True)
cat['sex'].replace({'male': '1', 'female': '0'}, inplace=True)
cat['checking_status'].replace(
    {
        'no checking': '0',
        '<0': '1',
        '0<=X<200': '2',
        '>=200': '3'
    },
    inplace=True)
```

cat		
	checking_status	employment
id		
1	2	2
2	2	1
3	1	1
4	2	2
5	1	2
...
996	1	0
997	0	4
998	0	4
999	1	2
1000	2	0

990 rows × 15 columns

Figure 35 encodage variable ordinales

Figure 34 Résultat encodage à la main

4.4.2. One-hot encoding

Consiste à encoder une variable à n états sur n bits dont un seul prend la valeur 1, le numéro du bit valant 1 étant le numéro de l'état pris par la variable. L'avantage principal de cet encodage est que pour passer d'un état à un autre, seules deux transitions sont nécessaires : un chiffre

passer de 1 à 0, un autre de 0 à 1. Grâce à la bibliothèque scikit-learn, nous avons effectué l'encodage de nos variables nominales comme suit et ainsi afficher le résultat de l'encodage.

4.9.2 Encodage des variables nominales

```
Entrée [39]: cat = pd.get_dummies(cat,
                                columns=[
                                    'credit_history', 'purpose', 'status_matrimonial',
                                    'property_magnitude', 'other_payment_plans',
                                    'housing', 'job'
                                ])

Entrée [40]: cat
```

Out[40]:

	checking_status	employment	savings_status	sex	other_parties	own_telephone	foreign_worker	class	credit_history_all paid	credit_history_critical/oth existing cred
id										
1	2	2	1	0	0	0	1	0	0	
2	2	1	1	0	0	0	1	0	0	
3	1	1	1	0	0	0	1	0	0	
4	2	2	1	0	0	1	1	1	0	
5	1	2	1	0	0	0	1	1	0	
...
996	1	0	1	1	0	1	1	1	0	
997	0	4	0	1	0	1	1	1	0	
998	0	4	1	1	0	0	1	1	0	
999	1	2	1	1	0	1	1	0	0	
1000	2	0	2	1	0	0	1	1	0	

990 rows x 41 columns

Figure 36 One Hot encoding

Nous pouvons voir que de nouvelles variables factices ont été créées et forment une matrice 900*41. Ces nouvelles variables factices sont constituées de 0 pour la non existence et de 1 pour l'existence de la valeur.

Après avoir fini de faire l'encodage des données, nous pouvons passer à l'étape suivante : Le feature scaling

4.5.Feature Scaling

Pour notre projet, nous avons utilisé le RobustScaler fourni par la bibliothèque scikit-learn de python. Et nous l'avons fait comme suit :

```

Entrée [33]: scaler = RobustScaler().fit(num)
num1 = scaler.transform(num)
num1 = pd.DataFrame(num1, index=num.index, columns=num.columns)
num = num1

Entrée [34]: # Données normalisées grâce au RobustScaler
num

```

Out[34]:

	duration	credit_amount	installment_commitment	residence_since	age	existing_credits	num_dependents
id							
1	2.50	1.393015	-0.5	-0.5	-0.733333	0.0	0.0
2	-0.50	-0.393974	0.0	-1.0	-0.533333	0.0	0.0
3	2.50	0.762426	0.0	0.5	-0.600000	0.0	0.0
4	-0.50	-0.289580	-1.0	-1.0	-0.733333	0.0	0.0
5	-0.25	-0.352524	-0.5	0.5	-0.333333	0.0	0.0
...
996	1.50	0.628478	0.5	0.0	-0.200000	0.0	0.0
997	-0.50	0.026291	0.5	0.0	1.133333	0.0	0.0
998	-0.50	-0.582422	0.5	0.5	0.333333	0.0	0.0
999	2.25	-0.182882	0.5	0.5	-0.666667	0.0	0.0
1000	2.25	0.865285	0.0	0.5	-0.400000	0.0	0.0

990 rows × 7 columns

Figure 37 RobustScaler

Après cette étape, nos données sont presque prêtes. Nous devons maintenant associer nos deux dataframe cat et Num en un seul. Grâce à la fonction « merge », cela est possible et nous avons maintenant un nouveau dataset qui est presque prêt. Mais un dernier problème réside, comment équilibrer les données ?

4.6.Équilibrage des données

Après la visualisation de notre rapport de pandas-profiling, nous nous rendons compte que nous avons 695 good et 295 Bad Une approche pour remédier aux ensembles de données déséquilibrés consiste à suréchantillonner la classe minoritaire. L'approche la plus simple consiste à dupliquer des exemples dans la classe minoritaire, bien que ces exemples n'ajoutent aucune nouvelle information au modèle. Au lieu de cela, de nouveaux exemples peuvent être synthétisés à partir des exemples existants. Il s'agit d'un type d'augmentation des données pour la classe minoritaire et est appelé la technique synthétique de suréchantillonnage de minorité, ou SMOTE pour faire court.

Le sous échantillonnage supprimera certaines lignes de la classe majoritaire pour rendre ses valeurs égales à celle de la classe majoritaire, alors que le suréchantillonnage ajoutera des données à la classe minoritaire pour qu'elle soit égale à la classe majoritaire.

Dans notre cas nous avons utilisés un suréchantillonnage pour éviter les pertes de données avec SMOTE. SMOTE fonctionne en sélectionnant des exemples qui sont proches dans l'espace de

fonctionnalité, en dessinant une ligne entre les exemples dans l'espace de fonctionnalité et en dessinant un nouvel échantillon à un point le long de cette ligne.

Plus précisément, un exemple aléatoire de la classe minoritaire est choisi pour la première fois. Puis k des voisins les plus proches pour cet exemple se trouvent (généralement $k = 5$). Un voisin choisi au hasard est choisi et un exemple synthétique est créé à un point choisi au hasard entre les deux exemples dans l'espace de fonctionnalité.

Nous l'avons effectué comme suit :

5.3.1 Suréchantillonnage des dataset grâce à la méthode SMOTE

```
Entrée [47]: # extraction de la variable cible y_s pour le dataset scaler et de du dataframe X_s
y_s = df_scale['class']
del df_scale['class']
X_s = df_scale

# extraction de la variable cible y_no pour le dataset non scaler et de du dataframe X_no
y_no = df_no_scale['class']
del df_no_scale['class']
X_no = df_no_scale

# suréchantillonnage du dataset df_scale afin d'équilibrer les classes grâce à la méthode SMOTE.
sm = SMOTE(sampling_strategy='minority', k_neighbors=2, random_state=42)
Xs, ys = sm.fit_sample(X_s, y_s)

# suréchantillonnage du dataset df_no_scale afin d'équilibrer les classes grâce à la méthode SMOTE.
sm2 = SMOTE(sampling_strategy='minority', k_neighbors=2, random_state=42)
Xno, yno = sm2.fit_sample(X_no, y_no)
```

Figure 38 Suréchantillonnage

Notons que, avant d'effectuer notre suréchantillonnage nous avons spécifié qui est la variable cible Y et qui est la variable X. Et nous avons choisir $k = 2$, pour plus de précision dans la génération des nouvelles.

Ainsi s'achève la partie prétraitement ; mais avant de passer à la partie suivante, nous devons effectuer une étape des plus importante pour la suite de notre travail : le split de nos données.

4.7. Le split des données

En général, la séparation est donnée suivant les proportion : 60% train, 20% test et 20% validation. Dans notre cas, nous avons effectué un premier split de 20% test et 80% de train et ensuite, nous avons pris ces 80% de train que nous avons encore diviser en 80% nouveau train et 20% pour la validation.

Grâce à la fonction `train_test_split` de la bibliothèque `scikit-learn` de python, nous avons procédé comme suit :

```

Entrée [48]: # Split du dataset Scale en train et test
Xs_train, Xs_test, ys_train, ys_test = train_test_split(Xs,
                                                         ys,
                                                         test_size=0.2,
                                                         random_state=42)

# Split du dataset non Scale en train et test
Xno_train, Xno_test, yno_train, yno_test = train_test_split(Xno,
                                                            yno,
                                                            test_size=0.2,
                                                            random_state=42)

Entrée [49]: # Split de train scale en train et validation
Xs_train, Xs_val, ys_train, ys_val = train_test_split(Xs_train,
                                                       ys_train,
                                                       test_size=0.2,
                                                       random_state=42)

# Split de train no scale en train et validation
Xno_train, Xno_val, yno_train, yno_val = train_test_split(Xno_train,
                                                           yno_train,
                                                           test_size=0.2,
                                                           random_state=42)

```

Figure 39 Split des dataset

Cette méthode contient des paramètres tels que : le dataset à utiliser pour faire le split, le pourcentage de l'échantillon désiré, et le paramètre aléatoire.

Après avoir effectué ces split, on se retrouve avec un dataset complet prêt pour la modélisation et sans attendre nous passons à la phase modélisation des données.

5. Modélisation des données

C'est dans cette étape que nous devons entraîner nos modèles avec nos données du train set que l'on a établi plus haut. Le but de l'entraînement est de permettre à la machine de comprendre les données qui lui sont fournies afin que plus tard elle puisse faire des prédictions sur des données nouvelles sur la base de ce qu'elle a appris.

5.1.Le choix

Etant donné que nous sommes dans un cas d'apprentissage supervisée appliquée à une classification, nous devons faire un choix parmi les modèles correspondant à notre problématique. Ces différents modèles ont déjà été présenté dans le chapitre 2.

Pour choisir les meilleurs modèles, nous avons fait des tests sur différents modèles afin de visualiser ceux ayant les meilleures performances. Comme modelé nous avons utilisés : La régression logistique, le random Forest, le KNeighbors, Le SVC, les modèles d'ensembles etc...

Mais avant de passer à la modélisation proprement dite, rappelons-nous que nous avons effectué un split de notre dataset d'origine qui est devenu 3 dataset distincts. Nous allons utiliser le dataset de train pour entrainer nos modèles, ensuite le dataset de validation pour voir les modèles à retenir pour l'amélioration et nous allons enfin tester nos modèles modifier avec le dataset de test.

5.2.L'entrainement des modèles

L'entrainement consiste à passer à la machine les données réelles quelle devra retenir pour effectuer une conjecture sur des données futures quelles n'a pas encore apprises. Dans notre cas nous avons entrainer nos modèles avec le train set (dataset train). Grace à la puissance de python et de ses librairies, pour faire un entrainement il faut utiliser la fonction 'fit' avec en paramètre la matrice X du trainset et sa matrice Y.

Ayant entrainer ces modèles, il faut éventuellement voir s'ils arrivent à bien prédire des données qu'ils n'ont pas apprises lors de l'entrainement. Raison pour laquelle dans un premier temps on entraine les modèles sans paramètres et on les tests pour le test set pour voir s'il généralise bien et ainsi voir leurs performances et déduire s'ils peuvent être amélioré.

Visualisons un exemple d'entrainement grâce au modèle de random Forest et voyons ses performances et vérifions que le modèle généralise bien les données apprises.

Après la visualisation des résultats de notre entrainement, nous avons deux courbes distinctes, une en bleue qui est la courbe d'apprentissage et l'autre en orange qui est la courbe de test. Nous voyons que les deux courbes sont assez éloignées ; ceci veut dire que nous sommes dans un cas d'overfitting.

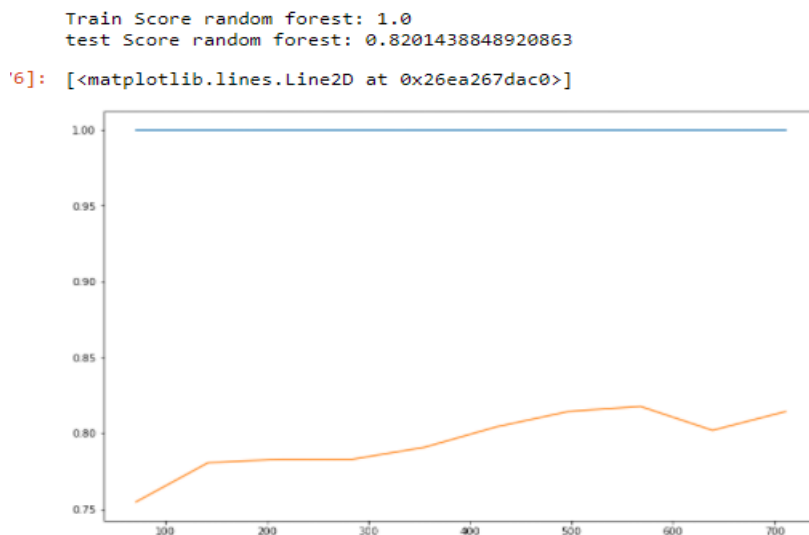


Figure 40 entrainement random Forest

5.2.1. L'overfitting

Ici, le modèle a été trop entraîné sur les détails des données d'entraînement. Ceci se produit généralement lorsqu'il apprend le bruit présent dans les données. Un modèle en overfitting ne fonctionnera pas bien sur de nouvelles données. Il s'agit du problème le plus courant dans l'apprentissage automatique car un modèle qui semble être très précis fonctionnera en réalité mal dans la nature et il faut corriger ce problème. Dans notre cas de random forest, nous voyons que les deux courbes sont assez éloignées et pour savoir que l'entraînement de notre modèle est optimale il faudrait que les deux courbes soient assez proches l'une de l'autre.

5.2.2. L'underfitting

Il peut arriver aussi que l'on rencontre des cas d'underfitting, ceci veut dire que le modèle n'a pas suffisamment appris pour faire une généralisation. Généralement dans une telle situation, la courbe bleue est très basse dès le début ainsi que celle en orange.

Comment donc corriger ce problème d'overfitting ou d'underfitting ? Ceci peut être corrigé par une méthode appelée la validation croisée.

5.2.3. La validation croisée.

Il s'agit d'une méthode d'estimation de fiabilité d'un modèle fondée sur une technique d'échantillonnage. Ce qui va se passer avec la validation croisée est que, le trainset va être divisé en sous-ensemble aléatoire et ces sous-ensembles vont servir à l'entraînement du modèle et ainsi tester à tour de rôle sur le testset. Ainsi en procédant ainsi, le modèle aura appris par plusieurs vagues et pourra ainsi mieux prédire sur de nouvelles données futures.

Dans notre cas nous avons utilisé une validation croisée à k blocs : On divise l'échantillon original en k échantillons, puis on sélectionne un des k échantillons comme ensemble de validation pendant que les $k-1$ autres échantillons constituent l'ensemble d'apprentissage. Après l'apprentissage, on calcule une performance de validation. Puis on répète l'opération en sélectionnant un autre échantillon de validation parmi les blocs prédéfinis. À l'issue de la procédure, nous obtenons ainsi k scores de performances, un par bloc. La moyenne et l'écart type des k scores de performances peuvent être calculés pour estimer le biais et la variance de la performance de validation.

Ainsi après avoir effectué notre validation croisée, nous avons à présent un modèle qui généralise bien nos données et cela se voit par la figure suivante :


```
Out[55]: [<matplotlib.lines.Line2D at 0x23bb54b5e20>]
```

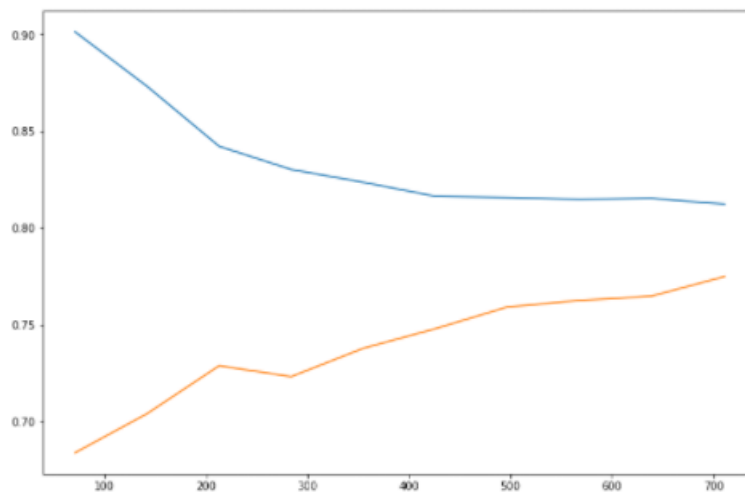


Figure 41 Random Forest bonne généralisation

Nous voyons que l'écart entre les deux courbes (courbes d'apprentissage) s'est réduites significativement. Ainsi cette opération a été effectuée à tous les modèles présentés au chapitre 2.

Après avoir entraîné nos modèles, nous pouvons essayer d'améliorer leur performances en améliorant leurs paramètres et cette méthode est appelée l'optimisation des hyperparamètres.

5.3.L'optimisation des hyperparamètres

En apprentissage automatique, l'optimisation ou le réglage des hyperparamètres est le problème du choix d'un ensemble de paramètres optimaux pour un algorithme d'apprentissage. Un hyperparamètre est un paramètre dont la valeur est utilisée pour contrôler le processus d'apprentissage. En revanche, les valeurs d'autres paramètres (généralement les poids des nœuds) sont apprises.

Le même type de modèle d'apprentissage automatique peut nécessiter différentes contraintes, pondérations ou taux d'apprentissage pour généraliser différents modèles de données. Ces mesures sont appelées hyperparamètres et doivent être ajustées pour que le modèle puisse résoudre de manière optimale le problème. L'optimisation des hyperparamètres trouve un tuple d'hyperparamètres qui produit un modèle optimal qui minimise une fonction de perte prédéfinie sur des données indépendantes. Il existe deux approches d'optimisation des hyperparamètres, une recherche de grille et une recherche aléatoire. Dans notre cas nous avons utilisé une recherche de grille.

La manière traditionnelle d'effectuer l'optimisation des hyperparamètres est une recherche par grille, qui simplement est une recherche exhaustive dans un sous-ensemble spécifié manuellement de l'espace d'hyperparamètres d'un algorithme d'apprentissage automatique.

Entre autre, on passe à la machine un intervalle de valeur d'hyperparamètres quelle devra faire des combinaisons pour retenir au final une combinaison d'hyperparamètres produisant les meilleures performances sur le train set.

Grâce à la bibliothèque, Scikit-learn de python, nous avons effectué une recherche par grille sur tous nos différents modèles présentés dans le chapitre 2. Prenons un exemple pour observer cela de plus près :

```
Entrée [55]: # définition de du modèle à utiliser
log = LogisticRegression()

# entraînement du modèle de regression logique grace à la méthode
log.fit(Xs_train, ys_train)
print('Train Score Regression Logistique:', log.score(Xs_train, ys_train))
print('Test Score Regression Logistique:', log.score(Xs_test, ys_test))

# tracé de la learning curve pour observer l'overfitting ou l'underfitting
N, train_score, val_score = learning_curve(log,
                                           Xs_train,
                                           ys_train,
                                           scoring='accuracy',
                                           cv=5,
                                           train_sizes=np.linspace(0.1, 1.0, 10))

plt.figure(figsize=(12, 8))
plt.plot(N, train_score.mean(axis=1), label='train score')
plt.plot(N, val_score.mean(axis=1), label='validation score')
```

Train Score Regression Logistique: 0.8076490438695163
Test Score Regression Logistique: 0.7805755395683454

Figure 42 régression logistique avant de grid search

```

Entrée [23]: # Création d'une bibliothèque d'hyperparamètres qui sera utiliser pour la recherche par grille
paramRL = {
    'C': [1, 0],
    'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga']
}

# application du GridSearch avec comme paramètres le modèle, la bibliothèque définie précédemment
gridRL = GridSearchCV(LogisticRegression(max_iter=82, penalty='none'),
                      paramRL,
                      cv=5)

# entraînement avec les hyperparamètres optimaux trouvés lors du GridSearch
gridRL.fit(Xs_train, ys_train)

```

```

Out[23]: GridSearchCV(cv=5, estimator=LogisticRegression(),
                    param_grid={'C': [1, 0], 'max_iter': [82], 'penalty': ['none'],
                                'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag',
                                           'saga']})

```

```

Entrée [24]: gridRL.best_score_

```

```

Out[24]: 0.7907446200723671

```

```

Entrée [25]: gridRL.best_params_

```

```

Out[25]: {'C': 1, 'max_iter': 82, 'penalty': 'none', 'solver': 'newton-cg'}

```

```

Entrée [26]: start = timeit.default_timer()

modelRL = gridRL.best_estimator_
modelRL.score(Xs_val, ys_val)

stop = timeit.default_timer()
execution_time = stop - start
print("Le temps d'exécution est de : " + str(execution_time))
print("train:", modelRL.score(Xs_train, ys_train))
print("val", modelRL.score(Xs_val, ys_val))
print("test", modelRL.score(Xs_test, ys_test))

```

```

Le temps d'exécution est de :0.004538999999965654
train: 0.8278965129358831
val 0.7533632286995515
test 0.802158273381295

```

Figure 43 régression logistique après le grid search

Nous pouvons voir que sur la figure 42, la performance sur le test est de 0.78 avant le grid search et de 0.80 sur la figure 43 après le grid search. Nous pouvons ainsi voir une amélioration significative des performances après une recherche par grille. Ainsi cette opération a été répétée sur tous nos différents modèles pour ainsi obtenir des modèles optimisés. Après avoir fait cette recherche par grille, nous avons des modèles optimiser mais il faudrait choisir une métrique pour leur évaluation.

6. Evaluation des modèles et comparaison.

Etant donné que nous avons entraîné nos modèles sur le train set, il faut donc l'évaluer sur les données du test set pour voir ses performances en temps réelle. Et pour ainsi apprécier les performances d'un modèle, plusieurs métrique d'évaluation sont disponibles.

6.1. La précision

C'est le nombre de résultats positifs corrects divisé par le nombre de résultats positifs prédits par le classificateur disponible sur la matrice de confusion. Elle est donnée par la formule :

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

Équation 1 Précision

6.2. Le rappel (recall)

C'est le nombre de résultats positifs corrects, divisé par le nombre de tous les échantillons pertinents (tous les échantillons qui auraient dû être identifiés comme positifs) disponible sur la matrice de confusion.

$$Recall = \frac{True\ Positive}{Predicted\ Results} \quad \text{or} \quad \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Équation 2 Recall

6.3. L'accuracy

L'accuracy est le rapport entre le nombre de prédictions correctes et le nombre total d'échantillons d'entrée.

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

Équation 3 Accuracy

6.4. Le f1-score

Le score F1 est la moyenne harmonique entre la précision et le rappel. La plage pour le score F1 est [0, 1]. Il nous indique à quel point notre classificateur est précis (combien d'instances il classe correctement), ainsi que sa robustesse (il ne manque pas un nombre important d'instances). Une haute précision mais un rappel plus faible, vous donne un résultat extrêmement précis, mais il manque alors un grand nombre d'instances difficiles à classer.

Donné par la formule :

$$F1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

Équation 4 F1_score

6.5. La matrice de confusion

Une matrice de confusion est un résumé des résultats de prédictions sur un problème de classification. Les prédictions correctes et incorrectes sont mises en lumière et réparties par classe. Les résultats sont ainsi comparés avec les valeurs réelles.

Cette matrice permet de comprendre de quelle façon le modèle de classification est confus lorsqu'il effectue des prédictions. Ceci permet non seulement de savoir quelles sont les erreurs commises, mais surtout le type d'erreurs commises. Elle est représentée comme suit :

		Réponse de l'expert	
		p	n
Réponse du classifieur	Y	Vrai Positif	Faux Positif
	N	Faux Négatif	Vrai Négatif

Figure 44 Matrice de confusion

- Les vrais positifs se sont les valeurs que nous avons prédit oui (il est mauvais client), et il est effectivement mauvais client.
- Les vrais négatifs se sont les valeurs que nous avons prédit oui (il est un bon client) et il est effectivement un bon client
- Les faux positifs ce sont les valeurs que nous avons prédit oui (il est mauvais client) et il bon client au final
- Les faux négatifs se sont les valeurs que nous avons prédit oui (il est un bon client) et il mauvais client au final

Pour visualiser tous les résultats du calcul des métriques nous avons le tableau ci-dessous

6.6. Comparaison des résultats obtenus.

Après avoir présenté toutes les différentes métriques d'évaluation d'une classification, nous avons utilisé l'accuracy et la matrice de confusion comme métriques principales. Et grâce au tableau suivant nous pouvons voir les performances de tous nos modèles ainsi que leur

évaluation sur les différentes métriques sans oublier le temps d'exécution des différents modèles.

Modèles	Hyperparamètres	Temps d'exécution(s)	Accuracy	Precision	F1_score	Recall	
Regression Logistique	C=1,max_iter=82, penalty='none', solver='newton-cg'	0.0045	0.8022	0.81(1) 0.70(0)	0.82(1) 0.78(0)	0.84(1) 0.76(0)	<pre>[[100 31] [24 123]]</pre>
KNeighbors	weights='uniform', leaf_size=20, p=2,algorithm='auto', metric='minkowski',n_neighbors=5	0.0245	0.7668	0.82 0.74	0.71 0.81	0.89 0.62	<pre>[[108 14] [38 63]]</pre>
Naive Bayes	alpha=4, binarize=0.92	0.0035	0.7085	0.68 0.73	0.67 0.74	0.66 0.75	<pre>[[91 31] [34 67]]</pre>
SVC	C=2, degree=1, kernel='rbf', gamma='auto', tol=1	0.0202	0.7769	0.81 0.75	0.78 0.77	0.76 0.79	
Random Forest	ccp_alpha=0.0, min_samples_leaf=1,min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=117	0.0214	0.8523	0.83 0.87	0.86 0.83	0.89 0.79	<pre>[[104 27] [16 131]]</pre>
Bagging Classifier	base_estimator=LogisticRegression(C=1, max_iter=82,penalty='none', solver='newton-cg') max_features=7, max_samples=9, n_estimators=22, verbose=2,	0.0101	0.7014	0.77 0.65	0.69 0.71	0.62 0.79	<pre>[[104 27] [56 91]]</pre>
XGClassifier	base_estimator=RandomForestClassifier(), learning_rate=0.25,base_score=0.5, n_estimators=98, max_depth=3,gamma=1	0.0046	0.8489	0.85 0.84	0.86 0.84	0.86 0.83	<pre>[[109 22] [20 127]]</pre>
AdaBoost	algorithm='SAMME.R', n_estimators=73, learning_rate=0.5714285714285714)	0.0229	0.8201	0.77 0.86	0.81 0.82	0.85 0.79	<pre>[[96 26] [15 86]]</pre>

Figure 45 Benchmark et comparaison des modèles

Pour la suite de notre travail, nous optons le choix du random Forest, la régression logistique et l'adaboost car le random Forest a un accuracy de 0.85, la régression logistique et le XGBClassifier ont une bonne matrice de confusion. Ces modèles auront pour hyperparamètres ceux présentés dans le tableau précédent. Pour une meilleure performance, nous utiliserons aussi le modèle d'ensemble 'votingclassifier' qui combinera les 3 modèles pour former un super modèle.

Ainsi parvenu au terme de notre modélisation, nous pouvons donc passer à la phase d'implémentation de nos modèles sous formes d'API.

III. Implémentation sous forme d'API

Dans cette partie, nous allons réaliser notre API qui sera basée sur notre travail effectué à la section précédente.

1. Raison de la mise sur pied d'une API

Nous avons pensé à mettre sur pied une API de prédiction du risque de crédit bancaire, pour permettre qu'elle soit intégrée dans d'autres systèmes pour faciliter la vie à d'autres structures.

L'API est le cœur des applications de nos jours, car c'est en son sein que se passe tous les différents processus utilisés par les différentes applications que nous connaissons.

Prenons un exemple, grâce à notre API, la programmation de l'application pour la consommer sera beaucoup plus simple car tout le travail réside déjà dans l'API en question car elle est le cœur de l'application et nous pouvons aussi l'appeler back end.

2. Les conteneurs

Pour rendre notre API fonctionnelle sur n'importe quel environnement, nous avons utilisé un conteneur docker. Un conteneur est un ensemble léger, autonome et exécutable d'un logiciel qui comprend tout ce qui est nécessaire à son exécution : code, temps d'exécution, outils système, bibliothèques système, paramètres.

Les conteneurs Docker sont basés sur des standards ouverts et fonctionnent sur toutes les distributions Linux majeures, Microsoft Windows, et sur n'importe quelle infrastructure, y compris les VMs et le Cloud.

Les conteneurs docker fonctionnant sur une seule machine partagent le noyau du système d'exploitation de cette machine ; ils démarrent instantanément et utilisent moins de calcul et de RAM. Les images sont construites à partir de couches du système de fichiers et partagent des fichiers communs. Cela minimise l'utilisation du disque et les téléchargements d'images sont beaucoup plus rapides.

Les conteneurs Docker isolent les applications les unes des autres et de l'infrastructure sous-jacente. Docker fournit l'isolation par défaut la plus forte pour limiter les problèmes d'applications à un seul conteneur au lieu de l'ensemble de la machine.

3. Méthode utilisée

Dans un premier temps nous nous sommes servis de notre méthodologie énoncée au chapitre précédent et de ses résultats pour construire notre API. Tout d'abord nous avons défini un squelette de l'API avec les méthodes GET et POST via le protocole HTTP. Le protocole HTTP (HyperText Transfer Protocol) est conçu pour permettre les communications entre les clients et les serveurs. HTTP fonctionne comme un protocole requête-réponse entre un client et un serveur. Exemple : Un client (navigateur) envoie une requête HTTP au serveur ; puis le serveur renvoie une réponse au client. La réponse contient des informations d'état sur la demande et peut également contenir le contenu demandé.

- **La méthode GET** est utilisé pour demander des données à partir d'une ressource spécifiée.
- **La méthode POST** est utilisée pour envoyer des données à un serveur afin de créer/mettre à jour une ressource.

Après avoir établi notre API, nous l'avons mis dans un conteneur pour qu'elle soit déployer sur n'importe quel serveur plus tard

4. Résultats finaux

Nous avons notre API présentée comme suit :

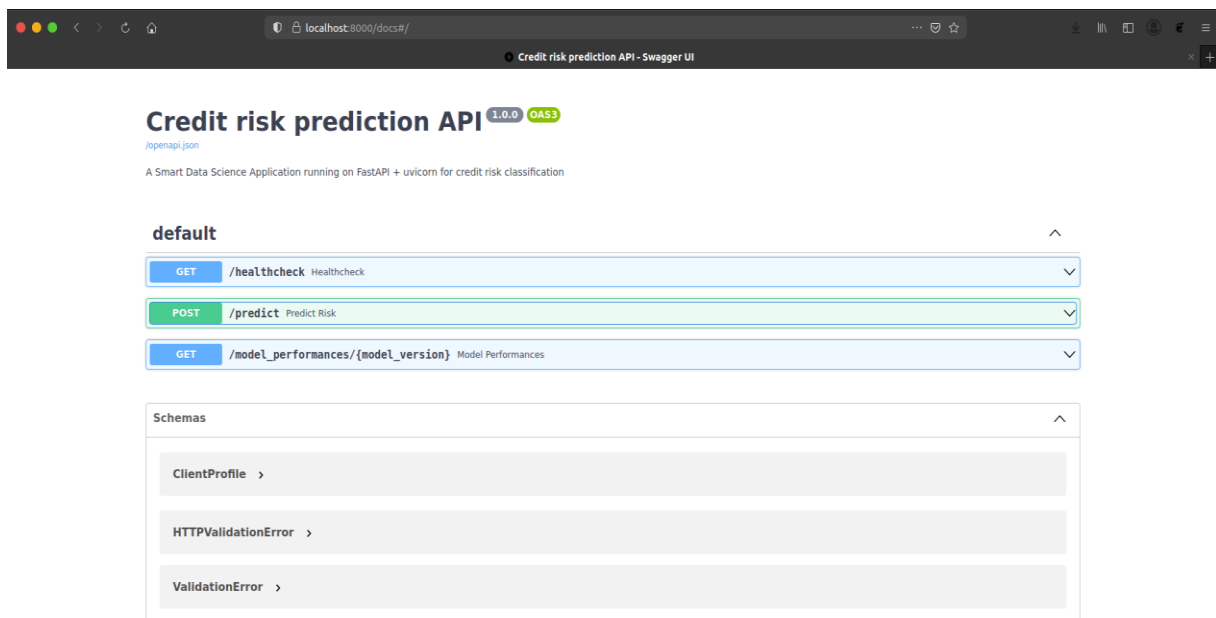


Figure 46 Page d'accueil de l'API

Notre API est présentée plus haut avec les méthodes GET et POST. La première méthode GET nous permet de vérifier si nous n'avons aucun problème lors de l'ouverture de l'API et du chargement des fichiers requis. Si tout est ok, on recevra un message « healthy ». Alors pour le vérifier, il suffit de cliquer sur le premier GET une liste se déroule et il suffira par la suite de cliquer sur try it out pour initier et ensuite sur execute pour lancer la vérification. On aura à la sortie une fenêtre comme celle-ci :

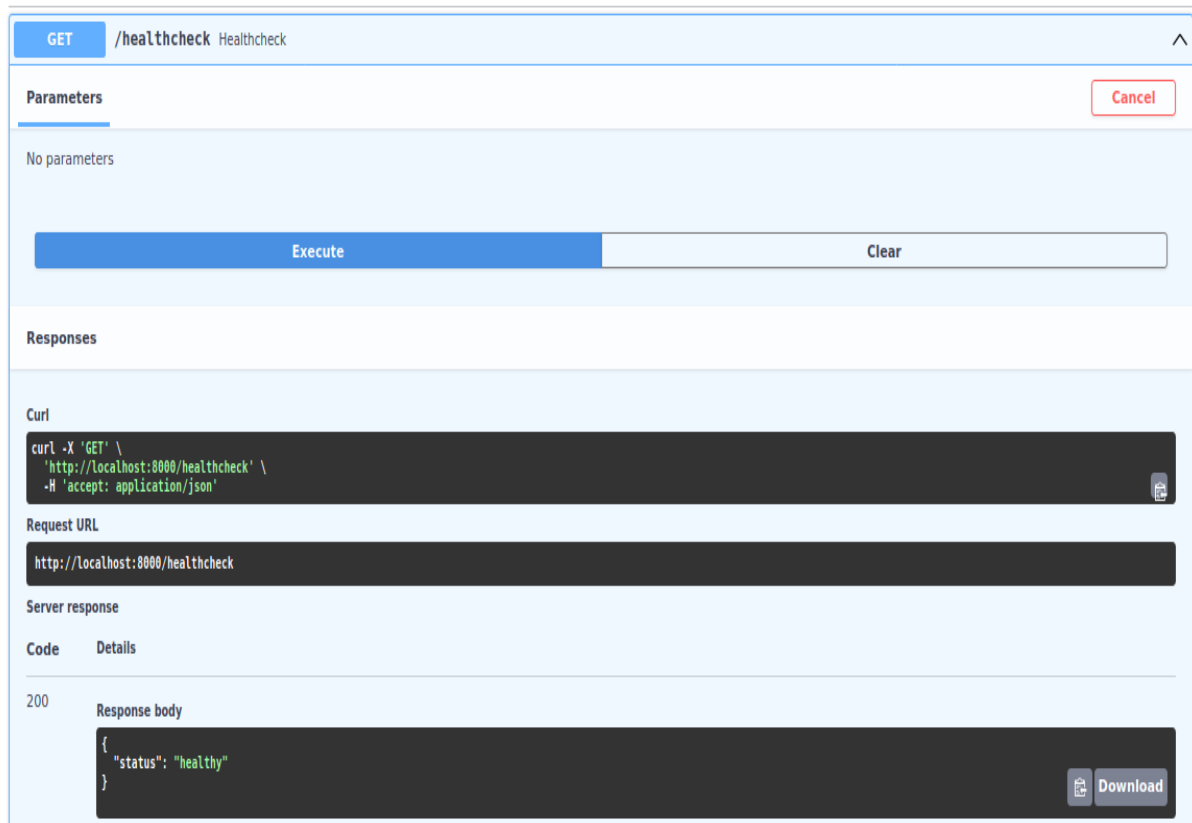


Figure 47 Méthode GET /healthcheck

Après avoir effectué cette vérification, nous pouvons voir les performances des différents modèles qui s'exécutent en arrière-plan grâce à la deuxième méthode GET. Notons que nous avons défini dans notre construction de l'api 4 modèles : le modèle v0 pour la régression logistique, v1 pour le random Forest, v2 pour l'adaboosting et v3 pour le votingclassifier.

Pour vérifier les performances de ses modèles il suffira de cliquer sur la deuxième méthode GET (/model_pereformances), de faire un try it out et de spécifier la version du model à vérifier dans le textbox « model_version » qui se présente à nous et de cliquer sur execute pour ainsi avoir les performances réelles du modèle choisi.

GET /model_performances/{model_version} Model Performances

Parameters Cancel

Name	Description
model_version * required (path)	<input type="text" value="v0"/>

Execute Clear

Responses

Curl

```
curl -X 'GET' \
'http://localhost:8080/model_performances/v0' \
-H 'accept: application/json'
```

Request URL

```
http://localhost:8080/model_performances/v0
```

Server response

Code	Details
200	<p>Response body</p> <pre>{ "model_version": "v0", "model_name": "LogisticRegression", "accuracy": 0.795, "precision": 0.6730769230769231, "recall": 0.5932203389830508 }</pre> Download

Figure 48 Méthode GET/model_performances

Nous pouvons remarquer que le résultat est renvoyé sous forme de JSON, et nous montre les performances du modèle suivant les différentes métriques que nous avons présentés dans le chapitre précédent.

Après avoir effectué toutes ces vérifications, nous avons la méthode POST pour effectuer notre prédiction proprement dite. La boucle d'exécution est pareil c'est-à-dire : cliquer sur le bouton POST (/predict) ensuite Try in out. Après cela, nous avons un formulaire qui s'affiche, et nous pouvons entrer les informations du client à risque tout en respectant la notation définie par la banque puis cliquer sur execute.

Prenons un exemple :

The screenshot shows a REST client interface with the following sections:

- POST /predict** Predict Risk
- Parameters**: No parameters
- Request body** required: application/json
- Request body**:

```
{  "checking_status": "no checking",  "duration": 100,  "credit_history": "critical/other existing credit",  "purpose": "used car",  "credit_amount": 10000,  "savings_status": "100<=X<500",  "employment": "1<=X<4",  "installment_commitment": 3,  "personal_status": "male mar/wid",  "other_parties": "none",  "residence_since": 4,  "property_magnitude": "car",  "age": 35,  "other_payment_plans": "none",  "housing": "own",  "existing_credits": 1,  "job": "unskilled resident",  "num_dependents": 2,  "own_telephone": "yes",  "foreign_worker": "yes",  "model_version": "v0"}
```
- Execute** button
- Responses**:
 - Curl**:

```
curl -X 'POST' \  'http://localhost:8080/predict' \  -H 'accept: application/json' \  -H 'Content-Type: application/json' \  -d '{  "checking_status": "no checking",  "duration": 100,  "credit_history": "critical/other existing credit",  "purpose": "used car",  "credit_amount": 10000,  "savings_status": "100<=X<500",  "employment": "1<=X<4",  "installment_commitment": 3,  "personal_status": "male mar/wid",  "other_parties": "none",  "residence_since": 4,  "property_magnitude": "car",  "age": 35,  "other_payment_plans": "none",  "housing": "own",  "existing_credits": 1,  "job": "unskilled resident",  "num_dependents": 2,  "own_telephone": "yes",  "foreign_worker": "yes",  "model_version": "v0"  }'
```
 - Request URL**: http://localhost:8080/predict
 - Server response**:

Code	Details
200	<pre>{ "prediction": "good", "probability": 0.706256614417217 }</pre>

Figure 49 Méthode POST/predict

Pour ce client, nous pouvons voir qu'il s'agit d'un bon client avec une probabilité de 0.79. Au vu de ceci, le banquier pourra alors prendre la décision finale en fonction de ce que le système lui affiche.

5. Discussion

Après avoir mis sur pied notre API, nous pouvons dire que notre objectif de départ a été atteint car nous avons eu un accuracy global de 0.8 et nos modèles tournent et prédisent bien les bons et les mauvais clients quant au remboursement de leur prêt bancaire.

Pour une meilleure mise en production de notre API, une interface plus attrayante qui utilise notre API, doit être mise sur pied suivant les préférences de la banque ayant posée la demande.

Tout au long de notre réalisation, nous avons rencontré deux difficultés majeures à savoir :

- La grande puissance de calcul nécessaire pour entrainer les modèles
- L'intégration de l'API dans le projet Loan 360, d'IWOMI Technologies

Nous pouvons proposer une architecture du workflow de notre API :

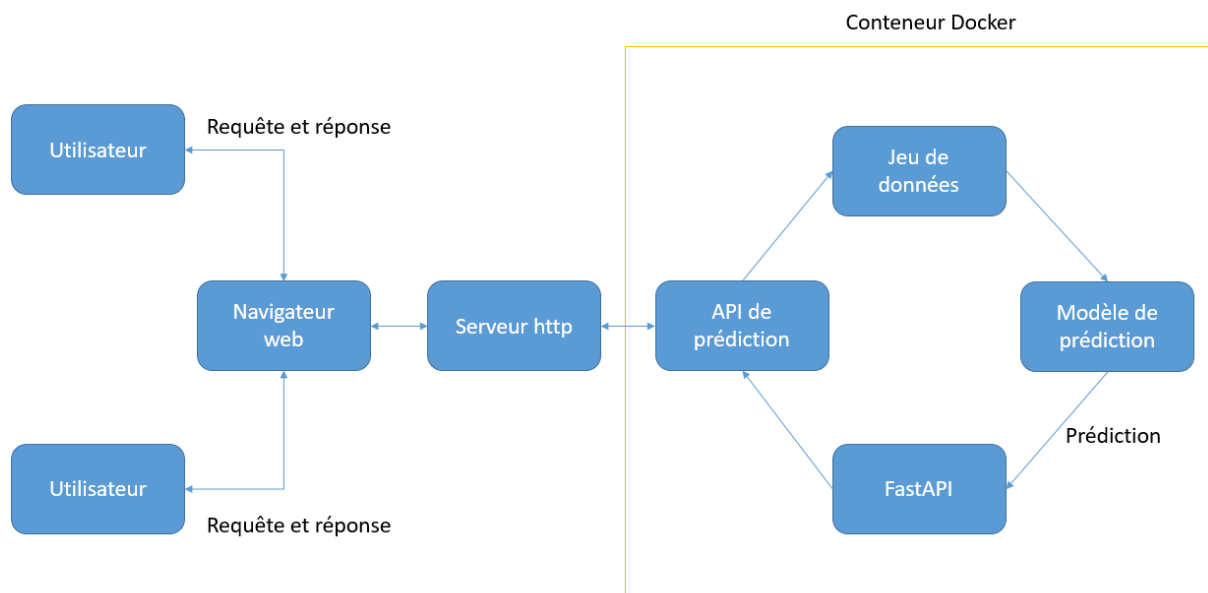


Figure 50 Architecture de communication de l'API

CONCLUSION ET PERSPECTIVES

Dans ce mémoire, il était question pour nous de mettre sur pied une API de prédiction du risque de crédit bancaire. La réalisation de notre travail s'est fait en plusieurs étapes : tout d'abord nous avons présenté l'entreprise dans laquelle nous avons effectué notre stage dans sa généralité, ensuite nous avons présentés la littérature qui gravite autour de notre thème dans le chapitre intitulé revue de la littérature, après nous avons présentés le matériel et les méthodes utilisés pour concevoir notre API, et enfin nous avons fait la conception de notre API et présenté son implémentation sous forme de résultats. D'après les tests effectués sur de nouvelles données, nous avons pu constater que notre API prédit les bons et les mauvais clients avec une bonne accuracy générale et une bonne matrice de confusion. Néanmoins, il faut noter qu'un tel système qui automatise les traitements aura un impact sur la société et changera les comportements des individus. Pour anticiper le changement sur les comportements, nous recommandons aux institutions financières des formations et des campagnes de sensibilisation sur l'usage de cette nouvelle technologie et par la suite une réévaluation (mise à jour) des algorithmes du système tous les 3 mois pour permettre à l'API de se réadapter à de nouvelles données. Les résultats obtenus dans ce travail peuvent être améliorés suivant plusieurs directions. Nous proposons ci-après quelques pistes qui nous ont semblé intéressantes :

- Un projet est destiné à évoluer avec le temps et pour se faire, nous pourrions passer à l'utilisation des réseaux de neurones (deep Learning) compte tenu de l'évolution de la quantité de données pour rendre notre système beaucoup plus précis, plus performant et plus optimal.
- Nous pourrions aussi par la suite nous attaquer au domaine de l'assurance qui est aussi un sous domaine du secteur financier en ajoutant à notre API des fonctionnalités liée aux assurances.

REFERENCES BIBLIOGRAPHIQUES

1. *01/09/1874/m_la-Gestion-du-Risque-de-Credit--un-enjeu-majeur---pour-les-Banques3*. (2021, Avril 16). Récupéré sur [www.memoireonline.com](https://www.memoireonline.com/01/09/1874/m_la-Gestion-du-Risque-de-Credit--un-enjeu-majeur---pour-les-Banques3.html): https://www.memoireonline.com/01/09/1874/m_la-Gestion-du-Risque-de-Credit--un-enjeu-majeur---pour-les-Banques3.html
2. (2021, Mai 02). Récupéré sur fastapi.tiangolo.com: <https://fastapi.tiangolo.com/>
3. *3/tutorial*. (2021, Avril 20). Récupéré sur [docs.python.org](https://docs.python.org/fr/3/tutorial/): <https://docs.python.org/fr/3/tutorial/>
4. *api-definition-utilisation*. (2021, Avril 20). Récupéré sur [www.agencedebord.com](https://www.agencedebord.com/api-definition-utilisation/): <https://www.agencedebord.com/api-definition-utilisation/>
5. *blog/numpy-pandas-introduction*. (2021, Mai 02). Récupéré sur [cloudxlab.com](https://cloudxlab.com/blog/numpy-pandas-introduction/): <https://cloudxlab.com/blog/numpy-pandas-introduction/>
6. *conseil/Machine-Learning-les-9-types-dalgorithmes-les-plus-pertinents-en-entreprise*. (2021, Avril 16). Récupéré sur [www.lemagit.fr](https://www.lemagit.fr/conseil/Machine-Learning-les-9-types-dalgorithmes-les-plus-pertinents-en-entreprise): <https://www.lemagit.fr/conseil/Machine-Learning-les-9-types-dalgorithmes-les-plus-pertinents-en-entreprise>
7. *courses/4011851-initiez-vous-au-machine-learning/4020611-identifiez-les-differents-types-dapprentissage-automatiques*. (2021, Avril 16). Récupéré sur [openclassrooms.com](https://openclassrooms.com/fr/courses/4011851-initiez-vous-au-machine-learning/4020611-identifiez-les-differents-types-dapprentissage-automatiques): <https://openclassrooms.com/fr/courses/4011851-initiez-vous-au-machine-learning/4020611-identifiez-les-differents-types-dapprentissage-automatiques>
8. *directory/20009/financial-risk-management/software*. (2021, Mai 01). Récupéré sur [www.capterra.fr](https://www.capterra.fr/directory/20009/financial-risk-management/software): <https://www.capterra.fr/directory/20009/financial-risk-management/software>
9. *gestion-de-projet/dossiers-methodes/bete-a-cornes*. (2021, Avril 30). Récupéré sur [www.manager-go.com](https://www.manager-go.com/gestion-de-projet/dossiers-methodes/bete-a-cornes): <https://www.manager-go.com/gestion-de-projet/dossiers-methodes/bete-a-cornes>
10. Iwomi_Technologies. (2021). *Organigramme IwomiTechnologies*. Douala.
11. *iwomitechnologies*. (2021, Avril 18). Récupéré sur [iwomitech](https://www.iwomitechnologies.com/fr/): <https://www.iwomitechnologies.com/fr/>
12. *le-nombre-de-creation-de-pme-en-hausse-au-cameroun-en-2019*. (2021, Avril 23). Récupéré sur <http://dias-invest.cm/>: <http://dias-invest.cm/le-nombre-de-creation-de-pme-en-hausse-au-cameroun-en-2019/>
13. *machine-learning/credit-g*. (2021, Mars 10). Récupéré sur [datahub.io](https://datahub.io/machine-learning/credit-g): <https://datahub.io/machine-learning/credit-g>
14. *machine-learning/usages*. (2021, Avril 15). Récupéré sur [ia-data-analytics.fr](https://ia-data-analytics.fr/machine-learning/usages/): <https://ia-data-analytics.fr/machine-learning/usages/>

15. *machine-learning-cas-dusage-dans-la-finance*. (2021, Avril 16). Récupéré sur analyticsinsights.io: <https://analyticsinsights.io/machine-learning-cas-dusage-dans-la-finance/>
16. *machine-learning-tout-savoir*. (2021, Avril 15). Récupéré sur datascientest.com: <https://datascientest.com/machine-learning-tout-savoir>
17. Microsoft. (2021, Mars 18). *articles/intelligence-artificielle/comprendre-utiliser-intelligence-artificielle*. Récupéré sur experiences.microsoft.fr: <https://experiences.microsoft.fr/articles/intelligence-artificielle/comprendre-utiliser-intelligence-artificielle/>
18. Oracle. (2021, Avril 14). *data-science/machine-learning/what-is-machine-learning*. Récupéré sur www.oracle.com: <https://www.oracle.com/ch-fr/data-science/machine-learning/what-is-machine-learning/>
19. *outil-gestion-risques-credit*. (2021, Avril 20). Récupéré sur obertys.com: <http://obertys.com/presse/outil-gestion-risques-credit/>
20. *principales-etapes-dun-projet-de-machine-learning-applique*. (2021, Mai 01). Récupéré sur notivica.com: <https://notivica.com/principales-etapes-dun-projet-de-machine-learning-applique/>
21. *quel-sens-metier-pour-les-metriques-de-classification*. (2021, Mai 02). Récupéré sur blog.octo.com: <https://blog.octo.com/quel-sens-metier-pour-les-metriques-de-classification/>
22. *topics/api/what-is-a-rest-api*. (2021, Avril 22). Récupéré sur www.redhat.com: <https://www.redhat.com/fr/topics/api/what-is-a-rest-api>