Script Oral PFE

# Chapitre 1. Introduction et présentation du cadre

Bonjour, je m’appelle Lila Roig et je vais vous présenter mon stage qui s’est déroulé sous la forme d’un contrat professionnalisation à la Caisse d’Epargne Midi-Pyrénées (CEMP).

slide

J’ai effectué mon stage dans l’équipe Data & Décisionnel qui a pour rôle de gérer les données et de les analyser pour les différents services.

L’équipe est constituée de 14 personnes dont le manageur Jérôme Martin ici présent, Bertrand Michelot qui est chef de projet Data et Mathieu Le Pajolec qui était prestataire Data Scientist.

Le but de mon stage est de développer des projets de Data Science en lien avec la stratégie de la CE.

Pour cela, j’avais à ma disposition 2 outils : le langage Python et l’ETL Alteryx, qui est un logiciel de no-code.

Je n’avais pas de GPU et je devais mener mes projets en autonomie, avec l’aide de Bertrand, Jérôme et Mathieu.

slide

Je vais donc décrire mes deux principaux projets en présentant à chaque fois les méthodes utilisées et les résultats obtenus. Je terminerai cette présentation avec une conclusion.

# Chapitre 2. Projet Back-Office crédit

slide

Je vais commencer par présenter le premier projet réalisé.

## Introduction

L’objectif de ce projet est de prédire le nombre de crédits immobiliers qui arrivent au back-office par semaine. 🡺 Back-office : organisme qui saisit sur le système informatique les informations du dossier de crédit.

* Ceci permettra de prévoir les variations de personnel pour diminuer le temps de traitement des dossiers.

Ce projet fait suite aux travaux de l’ancien alternant, qui avait déjà testé plusieurs méthodes sans succès, affichées ici

Comme les données crédits ont une dépendance temporelle, on reprend le projet avec une nouvelle approche Python qui utilise des méthodes spécifiques aux séries temporelles.

On rappelle qu’une série temporelle est une suite de points représentant l’évolution d’une variable au cours du temps.

slide

## Présentation et traitement des données

### Description des données

Les données crédits disponibles s’étendent de 2018 à fin 2022 et comportent 36 variables telles que l’identifiant ou les caractéristiques du dossier.

La variable réponse à prédire est NB\_DOSS\_DAY qui est le nombre de dossiers qui arrivent au back-office par jour.

### Pré-traitement

On se lance d’abord dans le pré-traitement des données :

* On sélectionne les variables explicatives pertinentes,
* On impute les données manquantes par 0 ou par divers calculs

### Analyse exploratoire

On effectue à présent l’analyse exploratoire des données. On calcule les matrices de corrélation et les boxplots pour analyser les dépendances entre les variables.

On observe que :

* Les variables correspondant à une **date** sont toutes fortement corrélées entre elles. On ne conserve donc qu’1 seule variable pour la date.

slide

* Les variables explicatives quantitatives et qualitatives sont très peu corrélées avec la variable réponse.

On peut donc émettre l’hypothèse qu’une approche multivariée ne va pas très bien fonctionner. Mais on va tout de même tester pour s’en assurer.

Et on met en forme le jeu de données.

slide

### Mise en forme des jeux de données

Pour la mise en forme, nous allons utiliser 2 approches :

* La première approche la plus simple est l’approche **univariée** où on prédit uniquement l’évolution du nombre de dossiers (NB\_DOSS\_DAY) en fonction de ses valeurs passées.

On construit alors la série temporelle TS\_day qui est le nombre de crédits par jour et TS\_week qui est le nombre de crédits par semaine.

* La deuxième approche plus complexe est l’approche **multivariée** où on prédit le nombre de dossiers (NB\_DOSS\_DAY) en fonction des valeurs passés d’autres variables explicatives.

On construit ainsi la série temporelle multivariée credits\_MTS, qui représente le nb de crédits par jour.

Le pas de temps n’est pas constant pour credits\_MTS car il y a des jours ou des semaines où il n’y a pas de dossiers. Et donc on ne peut pas « inventer » la valeur que prendraient les variables explicatives ces jours-là. Pour TS\_week et TS\_day, on peut rajouter des 0 où jours il n’y a pas de dossiers donc le pas de temps est constant.

Avec les séries temporelles, le jeu d’entrainement est constitué de 80% des premières valeurs et le jeu de test de 20% des dernières valeurs.

Nous pouvons maintenant passer aux méthodes.

slide

## Méthodes utilisées

### Modélisation par SARIMA sur la série univariée TS\_week

La première méthode utilisée pour la prédiction est une méthode statistique simple avec le modèle SARIMA. SARIMA ne fonctionne que pour les séries univariées. Comme le besoin métier est à la semaine et que TS\_day semble très bruitée, on travaille sur TS\_week.

SARIMA veut dire Seasonal ARIMA.

ARIMA est un modèle non-saisonnier à 3 paramètres pour la prédiction de ST, le paramètre Integrated, le paramètre auto-regressive et le paramètres moving average. Et SARIMA est une extension de ARIMA rajoute à ce modèle 4 paramètres prenant en compte la saisonnalité de la ST.

slide

#### Débruitage du signal

Avant d’appliquer SARIMA, on effectue d’abord un débruitage du signal : ~~avec la décomposition en ondelettes.~~

* Pour ce faire, on estime l’écart-type du bruit sigma à partir des coefficients d’ondelettes du signal.
* On effectue un débruitage par seuillage doux en ondelettes et sigma intervient dans le calcul du seuil.

#### Transformation Box-Cox & décomposition STL du signal

Par la suite, une série temporelle peut se décomposer en 3 parties :

* La tendance mt,
* La saisonnalité st
* Et le résidu ou erreur rt

On veut retirer la saisonnalité st de la ST pour améliorer les prédictions avec SARIMA.

La décomposition d’une ST peut être additive ou multiplicative. Dans notre cas, comme la saisonnalité varie avec le niveau de la série, la décomposition multiplicative est appropriée.

On décompose donc la série temporelle avec la méthode STL. C’est une méthode robuste de décomposition mais elle ne permet pas d’effectuer des décompositions multiplicatives.

Pour avoir l’équivalent d’une décomposition multiplicative, on transforme le signal avec la transformation Box-Cox et on effectue une décomposition additive sur le signal transformé. La transformation Box-Cox permet également d’avoir un modèle plus stable.

slide

#### Correction de saisonnalité

Une fois la décomposition effectuée, on peut retirer enfin la saisonnalité.

#### Application du modèle SARIMA

Et on applique maintenant le modèle SARIMA pour lequel il faut estimer 7 paramètres.

Pour déterminer ces paramètres manuellement, on trace les graphes ACF (auto-corrélation) et PACF (partial ACF) sur les données d’entrainement de wchapeau.

Et on décompose le signal dans le domaine de fourier pour trouver m.

Nous entraînons alors le modèle.

slide

### Modélisation par le réseau de neurones MiniRocket

Comme on le verra plus tard, la méthode SARIMA ne donne pas des résultats satisfaisants. On utilise donc le réseau de neurones MiniRocket basé sur des noyaux convolutifs aléatoires. Il permet de faire des prédictions sur des ST univariées et multivariées. Il obtient généralement de bonnes performances sans nécessiter de puissance de calcul importante.

#### Approche multivariée sur credits\_MTS

MiniRocket traite le cas multivarié donc

Pour l’approche multivariée, on travaille sur credits\_MTS.

On effectue d’abord le one-hot encoding des variables qualitatives.

On transforme ensuite la série multivariée pour qu’elle ait le bon format pour MiniRocket, en appliquant des fenêtres glissantes sur la série.

On entraîne ensuite le modèle.

Pour l’approche univariée, on travaille sur TS\_day et TS\_week de la même façon.

## Résultats

On a testé plusieurs configurations :

* SARIMA sur TS\_week
* MiniRocket sur TS\_day, TS\_week et credits\_MTS

Le meilleur résultat est obtenu avec MiniRocket sur les données TS\_week. On a un RMSE de 40.43 et un MAE de 30.37 soit environ une erreur moyenne de 30 dossiers par jour.

Cependant, l’erreur est trop importante pour le besoin métier et on ne va pas poursuivre le projet.

# Chapitre 3. Projet détection de signature

## Introduction

L’objectif de ce projet est de détecter la présence de signature sur plusieurs types de documents PDF.

* Ceci permettra d’automatiser le contrôle des documents qui était jusqu’alors manuel et non exhaustif.

## Présentation des données

On souhaite vérifier la signature sur 4 types de PDF : le LEA, le EAI, le BS et le QCF-QR. La méthode doit pouvoir se généraliser à d’autres types de documents et ne doit pas requérir de puissance de calcul car nous n’avons pas de GPUs.

On doit considérer un maximum de cas :

* Il existe plusieurs versions possibles d’un même document : le EAI en possède 2 et le BS plus de 10.
* Un document peut ne pas correspondre au type indiqué, il peut y avoir une erreur
* Le nombre de pages est variable et la signature ne se trouve pas toujours sur la même page pour tous les documents.
* Certains documents sont mal scannés (rotation, rognés)

## Méthodes utilisées

Etant donné que nous n’avons trouvé de méthode déjà implémentée qui a fonctionné, on implémente notre propre méthode Python reposant sur l’OCR Tesseract, qui est un réseau de neurones permettant de reconnaitre du texte dans des images.

slide

#### Pré-traitement de l’image

Tout d’abord, pour optimiser la lecture des images par l’OCR, on effectue un pré-traitement des images avec la librairie Python OpenCV :

- On rotationne les images pour qu’elles aient toutes un angle de 0°.

- On applique un débruitage et un seuillage sur les images pour obtenir une image la plus nette possible et en noir et blanc.

#### Création du modèle

On va ensuite définir un modèle par type de document, par exemple un modèle pour le document LEA. Ce modèle n’est à définir qu’une seule fois et servira à analyser tous les documents LEA.

Pour ce modèle, on va sélectionner un document bien scanné et bien lisible.

Sur la page contenant la signature, on va dessiner le ou les rectangles supposés contenir la signature. Une fenêtre implémentée avec OpenCV apparaît à l’écran et permet de dessiner ces rectangles.

Par exemple, on dessine 2 rectangles sur le LEA car la signature peut se trouver à 2 endroits.

On saisit ensuite une liste de mots clés contenus sur la page

et on calcule leur position avec Tesseract. Par exemple ici MotsCles = « Signature du titulaire », « Représentant », « Banque ».

slide

Enfin, on calcule la position des rectangles par rapport à la position des mots clés. On conserve cette donnée pour la suite.

#### Détection de la signature

Une fois que le modèle est sélectionné, on va pouvoir analyser l’ensemble des documents du même type que le modèle.

Si le document est numérique, on le repère grâce à une librairie Python et il est considéré signé par défaut.

Sinon,

on identifie tout d’abord la page contenant la signature. Pour cela, on va lire toutes les pages et s’arrêter lorsque le texte de la page en question est similaire au texte de la page signée du PDF modèle. On calcule la similarité entre les 2 textes avec l’indice de Jaccard.

slide

Une fois en présence de la bonne page, on redimensionne la page aux dimensions du modèle puis on calcule la position du premier élément de la liste MotsCles que l’on trouve sur la page. Par exemple, « Représentant ».

On a la position relative de tous les MotsCles du PDF modèle par rapport aux rectangles. On peut donc positionner sur le document à analyser, les nouveaux rectangles par rapport au mot clé « Représentant ».

Les rectangles sont donc placés au bon endroit, même si le PDF est décalé par rapport au PDF modèle.

Enfin, on regarde à l’intérieur des rectangles et on calcule le pourcentage de pixels continus. Si ce pourcentage est suffisant, le document est signé. Sinon, il n’est pas signé.

## Résultats

On réimplémente une méthode similaire sous Alteryx. La principale différence est que les rectangles contenant la signature sont fixes avec Alteryx.

slide

On obtient les résultats plutôt satisfaisants avec les 2 méthodes. Sur des jeux de tests constitués de 100 documents de chaque type, Python est 2 à 4 fois plus rapide que Alteryx.

Alteryx est meilleur que Python pour les documents LEA et EAI et Python obtient de meilleures performances pour les documents BS et QCF-QR.

Alteryx ne fonctionne pas pour le BS car celui-ci possède plus d’une dizaine de versions différentes et Alteryx ne peut pas adapter les rectangles à chaque version.

Ces résultats sont cependant à nuancer car nous avons constitué les jeux de tests au hasard et ils n’ont pas les mêmes proportions de documents signés, non signés, mal scannés…

Limites :

* Cette solution requiert autant de modèles que de versions différentes du document. Par exemple, le EAI possède 2 versions et on doit donc utiliser un premier modèle sur les documents puis récupérer les documents qui n’ont pas fonctionné et les soumettre à un 2e modèle. S’il y a trop de versions très différentes entre elles, cela ne fonctionnera pas.
* Avec le BS, il y a + de 10 versions mais un seul modèle suffit en Python si on choisit bien les MotsClés proches de la zone signée (les modèles différaient mais pas autour de la zone signée)

15min

# Chapitre 4. Projet mot interdit

## Introduction

Je vais maintenant parler rapidement du 3ème projet où

l’objectif est de détecter les mots interdits par le RGPD dans du texte.

## Présentation des données

On recherche la présence de mots interdits dans une base de données Osirisk contenant

50 000 lignes. Pour cela, on dispose d’une liste de mots interdits.

## Méthodes utilisées

Pour la première méthode utilisée,

- nous avons tout d'abord effectué un pré-traitement des mots interdits et du texte de la table Osirisk

- Puis nous avons recherché les mots interdits présents dans la table Osirisk.

- Cependant, certains noms tels que 'Nègre Véronique' sont considérés à tort comme des mots interdits. Ainsi, lorsque nous recherchons les mots interdits, nous considérons qu'un mot n'est pas un mot interdit si celui-ci est précédé ou suivi d'un prénom.

- La deuxième méthode utilisée s'apparente à la première méthode mais inclut de plus une étape supplémentaire qui consiste à lemmatiser (prendre la forme neutre canonique du mot) les mots interdits et la table Osirisk.

Ceci permet de capturer davantage de mots interdits.

- Puis on réaliser les mêmes étapes que précédemment

Avec la première méthode, nous avons trouvé 409 mots interdits et 455 avec la deuxième méthode.

Les 2 première méthodes ne permettent pas de traiter les cas nécessitant une compréhension du texte, comme pour cette phrase : … où blaireau est considéré à tort comme un mot interdit

On pourrait implémenter une 3ème méthode en utilisant des embeddings et/ou des réseaux de neurones ayant une meilleure compréhension du texte.

# Chapitre 5. Conclusion & perspectives

Pour conclure, je suis très satisfaite de mon alternance à la CE :

* Le fait de travailler sur des projets très variés m’a permis d’appliquer ce que j’avais appris à l’insa mais aussi d’apprendre énormément
* J’ai travaillé en autonomie ce qui m’a permis de prendre confiance en ma capacité à savoir rechercher et implémenter des solutions qui fonctionnent. J’aurai également apprécié de travailler en équipe
* J’ai vu le fonctionnement d’une grande entreprise tout en étant dans un petit service.
* Je me suis rendu compte que j’aimerais travailler dans une entreprise avec davantage de moyens en data Science, en RetD ou en recherche.

J’ai travaillé sur un dernier projet que je ne présenterai pas ici.