Détails sur quelques projets réalisés

Lucas - lprtk

April 20, 2022

Page Github: lprtk

- **Challenge inter-universitaire – NeuroSpin, Machine Learning et neuroimagerie (computer vision)**

  Dans le cadre d'un cours de Machine Learning avancé dispensé par Edouard Duchesnay, chercheur chez NeuroSpin et développeur principal de Scikit-learn, nous avons réalisé un challenge afin de nous introduire la neuroimagerie, et ce domaine de recherche, avec l'université de Paris-Saclay auquel nous sommes arrivés 3ème.

  Quel est le contexte du projet ?
  Le vieillissement, ainsi que les problèmes de santé qui lui sont associés, ont toujours été un défi majeur pour les sciences médicales de manière à accroître l'espérance de vie moyenne d'une société. Récemment de nombreuses études médicales, notamment dans le domaine de la neuro-psychiatrie, se sont concentrées sur l'analyse d'imageries par résonance magnétique (IRMs) afin de décrypter et extraire le volume du tissu cérébral (noyaux et axones des neurones composant le tissu cérébral dans son ensemble) et d'estimer l'âge cérébral d'un patient.

  Pourquoi se lancer dans une telle problématique ?
  Quand on évoque le concept d'« âge cérébral », l'idée est de savoir si l'âge civil correspond à l'âge mental observé lors de l'analyse du volume de tissu cérébral. Les enjeux sont très simples et à la fois colossaux : plus la différence entre l'âge du cerveau d'un patient et son âge réel est élevée, plus son risque de mauvaise santé mentale, physique et même de décès prématuré l'est également.

  Comment y sommes-nous arrivés ?
  Pour relever ce défi, notre tentative combine : 1) des approches computationnelles de la neuroanatomie mesurant la concentration du tissu cérébral : Voxel-Based Morphometry (VBM) et Region Of Interest (ROI) ; 2) le recours à des algorithmes de Machine Learning spécialement calibrés afin d'apporter une forte robustesse et explicabilité des résultats ; ainsi que 3) la relation bien établie entre l'âge et la neuroanatomie tout au long de la vie, de manière à détecter en amont les maladies liées à l'âge dans le but ultime de prévenir ou de retarder leur progression.

  Programmation en Python

- **Réaliser une analyse sémantique à l'aide du Deep Learning (NLP)**

Ce projet a été réalisé dans le cadre d'un cours d'introduction au Deep Learning appliqué au NLP. À partir de phrases, composant un texte, ainsi que les positions grammaticales de chaque mot, nous devions utiliser des modèles de Part-Of-Speech Tagging (POS-Tagging) et de Shallow-Parsing (chuncking) afin d'extraire de l'information « cachée » et les relations existantes entre les mots d'un texte. L'objectif étant d'utiliser le Deep Learning pour quantifier :

- La différence entre un modèle de POS-Tagging et de Shallow-Parsing ;
- L'apport, dans l'architecture, d'une couche d'embedding pré-entraînée et de la back-propagation à la représentation vectorielle de la couche d'embedding ;
- L'impact sur la capacité prédictive des modèles de plus ou moins de contexte autour du mot, dont le tag grammatical doit être prédit, en faisant varier le ngram range ;
- La différence entre une architecture implémentant un modèle par tâche par rapport à une architecture multi-task ;
- La différence entre un modèle multi-task simple par rapport à un modèle multi-task hiérarchique consistant à construire une architecture en cascade où les tâches n'interviennent pas à la même profondeur du réseau de neurones.

Programmation en Python

- **Prévision du volume horaire du trafic routier avec différentes approches de Deep Learning (séries temporelles)**

  Ce projet porte sur la modélisation et la prévision des volumes horaires de trafic urbain entre deux grandes villes américaines de l'État du Minnesota, Minneapolis et St. Paul, et sur l'évaluation des effets de diverses caractéristiques contextuelles telles que les jours de la semaine, les jours fériés ou encore les conditions météorologiques sur les volumes de trafic. L'objectif fut de tester un large spectre d'approches, de la plus simple à la plus complexe, en confrontant l'approche économétrique traditionnelle et les nouvelles techniques de Machine et Deep Learning :

  - Régression linéaire ;
  - LightGBM ;
  - GRU ;
  - LSTM ;
  - Bidirectionnel LSTM.

  Dans un deuxième temps, nous avons proposé une ouverture vers les approches de Deep Hybrid Learning avec un modèle CNN-LSTM. Puis, inspirés par l'ES-RNN, ou le MES-LSTM, observé lors du concours de prévision Makridakis-4 (M4), nous avons voulu essayer d'implémenter un modèle hybride fusionnant une régression linéaire et le LightGBM. Nous avons opéré très simplement, l'objectif est d'attribuer un poids α à la prévision par régression et 1-α à la prévision LightGBM. Ce coefficient α est déterminé en modélisant une courbe exponentielle, avec un certain coefficient, afin de donner plus de force aux prévisions à court terme de la régression et d'inverser la tendance par la suite dans l'optique de donner plus de poids aux

prévisions à long terme du modèle LightGBM. Les performances de ces deux modèles fusionnés ne sont pas phénoménales mais nous avons voulu explorer cette piste pour aller plus loin et envisager des combinaisons plus intéressantes comme les modèles ARDL-LSTM.

Programmation en Python

- **Web scraping et analyse de sentiments (NLP)**

  Afin de renforcer et acquérir de nouvelles compétences quant à l'utilisation de Python, des outils Big Data, avec pySpark, mais aussi et surtout sur le NLP, nous avons réalisé un projet combinant de multiples notions : scraping, analyse de sentiments, topic modeling ou encore Machine Learning. Notre démarche applicative se compose de la sorte :

  1. Web Scraping : construction de notre propre base de données en allant scraper des commentaires de clients sur le site Amazon US, puis les parser de manière à obtenir une base complète et composée de plusieurs variables.
  2. Text Cleaning : dans un projet de NLP, le nettoyage des données textuelles est un enjeu très important pour la capacité prédictive des futurs modèles mis en place. Ici, nous avons recouru à pySpark afin d'appliquer de nombreuses transformations (caractères, espaces, hastags, urls, emojis, balises html, stopwords, lemmatisation, etc.).
  3. Analyse de sentiments : calculer, à l'aide de l'algorithme VADER, les scores sentimentaux de chaque commentaire en combinant des caractéristiques lexicales des sentiments (lexicons), des règles grammaticales ainsi que des règles syntaxiques pour exprimer la polarité (positive / neutre / négative) et l'intensité (puissance) des émotions.
  4. Topic Modeling (apprentissage non-supervisé) : il peut être intéressant de classer chaque avis client dans un topic précis afin, pour un vendeur, d'améliorer le délai des réponses et la satisfaction client. Ainsi pour chaque commentaire, un cluster lui est attribué afin de mieux séparer les différents commentaires recensés.
  5. Prédiction du sentiment (apprentissage supervisé) : à partir du score sentimental estimé par notre modèle VADER, entrainer des algorithmes de Machine Learning qui permettent, sans lire les avis, d'être en mesure de connaître la satisfaction d'un nouveau commentaire publié.

  Programmation en Python

- **Challenge – Valeo, Deep Learning pour la prédiction des pièces défaillantes sur les lignes de production (computer vision)**

  Dans le cadre d'un cours de Deep Learning dispensé par Roman Yurchak, chercheur et développeur principal de Scikit-learn, nous avons réalisé un projet de computer vision dans le domaine de l'industrie. Ce projet s'inscrit dans le cadre d'un challenge Data ouvert en 2020 par Valeo, un équipementier automobile.

  Valeo nous met à disposition un échantillon de photos de pièces automobiles qui sont, ou non, défaillantes. L'objectif de ce défi est de prédire les défauts sur les lignes de production de moteurs de démarrage. Pendant l'assemblage des échantillons de production, différentes valeurs (couples, angles, etc.) sont mesurées sur différents postes de montage. En fin de ligne,

des mesures supplémentaires sont effectuées sur deux bancs d'essai afin d'isoler les défauts. En conséquence, les échantillons sont marqués "OK", "KO". L'objectif est donc de concevoir un modèle qui identifie de tels défauts avant l'étape du banc d'essai.

Programmation en Python

- **Créer une webapp sécurisée, avec un système de login/logout, et y implémenter un moteur permettant de résumer des textes**

  Ce projet est à l'intersection entre la Data Science et le développement web. En tant que Data Scientist, nous développons de nombreux outils mais la finalité reste de rendre ces outils, facilitant les décisions, disponibles à tous (codeurs ou non). C'est tout l'objectif de ce projet : 1) réaliser un outil de NLP et 2) le rendre disponible à travers une webapp sécurisée.

  1. Le moteur à résumé : en considérant nos phrases comme un ensemble de vecteurs, nous utilisons le cosine similarity, une mesure de similarité entre deux vecteurs non nuls, et TextRank, un algorithme de NLP de classement basé sur des graphes, afin résumer un texte avec ses phrases les plus pertinentes.
  2. La webapp sécurisée : créer une webapp, avec un système de login/logout, pour accéder au service de NLP. Pourquoi parle-t-on de webapp « sécurisée » ? Si nous souhaitons développer une webapp publique, elle ne pourrait être utilisée par les professionnels car, en tant que collaborateur, il n'est pas possible de soumettre des documents internes sur des sites externes. D'où la notion de webapp « sécurisée ».

  Comment ça marche en pratique ?

  - Se créer un compte (id + mdp + question de sécurité + réponse à la question) ;
  - Se connecter afin d'avoir accès au service de résumé ;
  - Importer un fichier texte ;
  - Sélectionner le nombre de lignes que doit faire le résumé ;
  - Télécharger une fois le programme exécuté ;
  - Se déconnecter de l'application.

  Le déploiement de l'application s'accompagne d'une table au format base de données (.db), en backend stockée dans une zone sécurisée du réseau, qui stocke, elle-même, toutes les informations de connexion des utilisateurs (id, mot de passe, question de sécurité, etc.). Concrètement, pour avoir accès à ce service de NLP, l'utilisateur doit s'identifier par le biais d'un profil utilisateur (identifiant + mot de passe). Cela permet de réguler et sécuriser le trafic sur l'app. Afin de renforcer la sécurité de nos utilisateurs, nous avons implémenté un algorithme de hashing permettant de chiffrer les informations de connexion stockées dans la base de données et donc de ne pas y avoir accès. Le concept de hashing permet d'introduire une sécurité supplémentaire dans notre application.

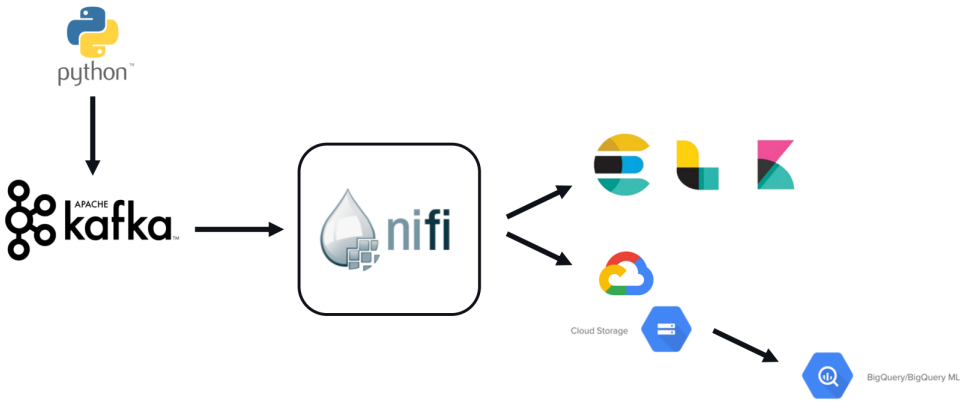
  En complément de la gestion de cette base de données, en backend, plusieurs autres fonctionnalités ont été implémentées :

- Lorsque l'utilisateur a oublié son mot de passe, il est possible de lui en générer un nouveau aléatoire en répondant à la question de sécurité renseignée lorsqu'il a créé son compte ;
- Il est possible de modifier son mot de passe afin de le changer régulièrement ;
- Il est également possible de supprimer, définitivement, son compte utilisateur.

Programmation en Python

- **Utiliser des outils Big Data pour créer sa propre architecture de données cloud**

  Dans le cadre d'un cours portant sur les architectures de données et le cloud, nous avons réalisé un Proof Of Concept (POC) afin concevoir notre propre architecture et flux de données. Ci-dessous, voici un aperçu schématique de la composition de notre flux ainsi qu'une explication détaillée des outils utilisés.



Premièrement, commençons par notre producer : nous avons écrit un script Python permettant de publier des données dans un topic Kafka. Plus précisément, ce script nous permet d'ajouter un nouveau message à la fin de la séquence d'une partition. Par ailleurs, pour s'assurer qu'un type de message donné se retrouve toujours sur la même partition, nous leur ajoutons la même clé.

Le script Python envoie les données vers Kafka, qui est, ici, utilisé comme broker (bus de données). Par ailleurs, afin de sécuriser davantage nos données, nous fixons le facteur de réplication à 3 cela signifie que la partition est présente 3 fois (2 partitions de réplicas plus une partition principale). De plus, nous considérons seulement 1 topic, car le script n'envoie qu'un seul type de message, et 11 partitions (calculé à partir des contraintes techniques).

À présent, nous arrivons à l'outil constituant le cœur de notre architecture Big Data. À l'aide de Nifi, nous récupérons les données depuis Kafka pour ensuite les redispatcher vers les autres outils de notre architecture : ELK et Google Cloud Storage.
Premièrement, nous utilisons la suite ELK car elle nous apporte une polyvalence : 1) Logstash : nous permet de collecter, filtrer, parser et transformer des données pour ensuite les stocker dans 2) ElasticSearch et 3) Kibana nous permet d'analyser en temps réel nos données avec des dashboards.

Pour finir, nous avons créé un Data Lake, stockant nos données chaudes, avec Google Cloud

Storage qui reçoit des données renvoyées depuis Nifi. Ensuite, afin d'utiliser, manipuler et requêter nos données stockées dans notre Data Lake, nous recourons à Google BigQuery ainsi qu'à BigQuery ML pour réaliser des modèles de Machine Learning en temps réel.

Programmation en Python

- **Challenge inter-universitaire – DRiM Game 2021 Deloitte, SAS et RCI Bank & Services (séries temporelles)**

  Le DRiM Game est un challenge inter-universitaire, portant sur la gestion des risques financiers et notamment sur le risque de crédit, qui fait concourir environ 100 équipes provenant des cinq plus grandes formations Économétriques en France, auquel nous sommes arrivés 5ème. Il est organisé, en partenariat, par Deloitte, SAS et RCI Bank & Services.

  Quel est le contexte du projet ?
  Le renforcement des dispositifs prudentiels, en matière bancaire et financière (accords de Bâle, BCBS 239, IFRS9, etc.), pousse les établissements financiers à mieux considérer les risques auxquels ils s'exposent. Ces derniers cherchent à prendre en compte des éléments prospectifs dans l'estimation de leurs provisions. Ce sujet possède de réels enjeux de gouvernance pour les établissements financiers. La projection des paramètres de risque permet de calculer les impacts directs et indirects en termes de solvabilité.

  Quelle est la problématique ?
  À partir d'un portefeuille de crédit, de RCI Bank & Services, sur des tiers particuliers italiens, nous devions déterminer l'évolution des migrations annuelles de notation de crédit en fonction de la conjoncture macroéconomique sur des différents horizons temporels. Autrement dit, cela revient à proposer une approche de projection de matrices de migration annuelles de notation de crédit par l'application de méthodes dites « traditionnelles » (modèles de séries temporelles classiques) et par l'utilisation de méthodes alternatives (Machine et Deep Learning).

  Comment y sommes-nous arrivés ?
  De manière plus détaillée, notre démarche scientifique s'articule comme suit :

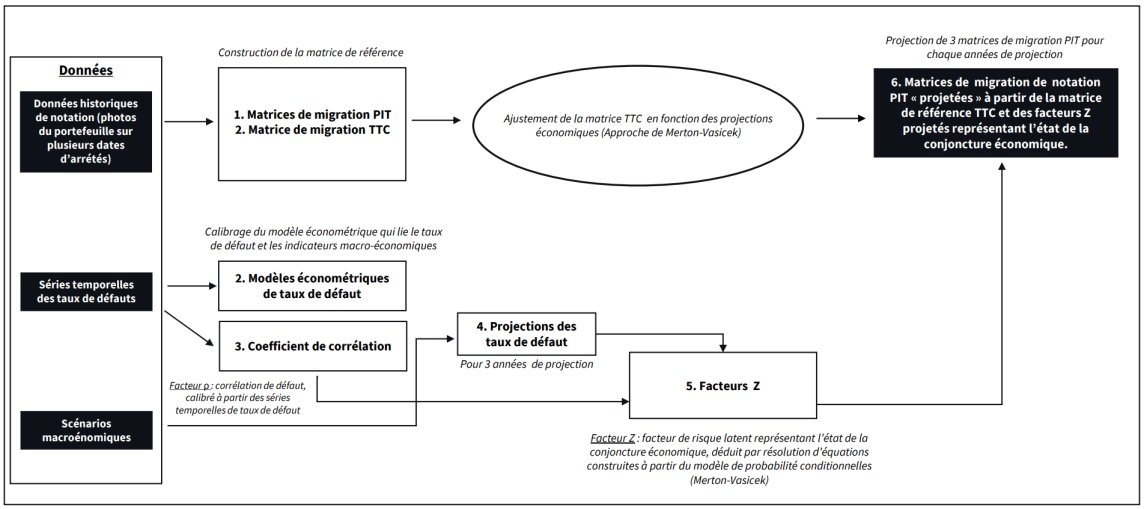
  – Construction d'un ensemble de matrices de migration de rating « Point in Time » (PIT), représentant des probabilités conditionnelles à l'état de la conjoncture économique, afin de former la matrice de migration de référence « Trough the Cycle » (TTC), représentant, elle, des probabilités non-conditionnelles à l'état de la conjoncture, sur notre période d'entrainement du portefeuille de crédit ;

  – S'approprier l'approche traditionnelle afin de réaliser une projection du taux de défaut à l'aide d'un modèle satellite et d'approches alternatives. Puis, calibrer la projection de la matrice de migration TTC, sur la période d'entrainement, à l'aide du modèle Merton-Vasicek ;

  – Enfin, réaliser une nouvelle projection des matrices stressées pour différents horizons temporels (à l'aide de scénarios). Puis, valider les capacités prédictives de la projection des matrices de migration, sur les horizons considérés, par le biais d'un backtesting avec un échantillon de données de test.

  Programmation en Python

De manière schématique, voici, ci-dessous, un résumé de notre démarche lors de cette édition du DRiM Game.



- **Asset's correlation: first eigenvalues an early warning indicator for market crashes? (séries temporelles)**

  Ce mémoire vise à utiliser des méthodes statistiques empruntées à la biologie et la physique, afin de déterminer si les futurs manques de résilience d'une économie sont prévisibles. Le but est d'utiliser des séries temporelles financières, comme les cours boursiers d'une entreprise. Ces séries sont retravaillées afin de les rendre stationnaires et propres (création d'un lag – différence première –, puis normalisation de la série avec un GARCH studentisé : on utilise la variance conditionnelle de chaque série pour normaliser et supprimer les biais d'hétéroscédasticité de type White).
  Nous créons trois portefeuilles de 15 actifs (small – SP600 –, mid – SP400 – et large – SP500 – cap index) afin d'avoir une vision sur chaque secteur de cotation. Sur chaque portefeuille l'objectif est de calculer des matrices de corrélation en overlapping et non overlapping, puis de récupérer les valeurs propres maximales de chaque matrice.

  Deux méthodes sont alors utilisées pour valider notre théorie : premièrement nous réalisons des heatmaps, avec les matrices de corrélation, afin de voir si plus nous nous approchons d'un point de retournement – une crise –, plus la corrélation entre les rendements d'actifs boursiers augmentent. De plus, il s'agit de valider si un secteur de cotation est leader dans la détection précoce des changements de régime. Dans un second temps, il s'agit d'utiliser la première valeur propre d'un vecteur propre associé à une matrice de corrélation, et de voir comment évolue cet indicateur sur la période 2002 – 2020, s'il permet d'être un « Early Warning Indicateur » des futurs changements de régime et surtout comment ce dernier se comporte face à d'autres indicateurs comme le CISS (un indice de stress financier) et le VIX (un indice de volatilité).

  Mots-clés: Tipping point · Complex System Theory · Time series · Predictive Modelization · Correlation matrix · Eigenvalue · Heatmap · Random Matrix Theory · Heteroskedasticity (GARCH) · Critical transition · Market crashes.

  Programmation en Python et SAS