Implémentez un modèle de scoring

👥 Note d'accompagnement

L'intention de ce projet est de réaliser un projet de bout en bout jusqu'à sa mise en production.

Le focus sera mis sur :

- La mise en œuvre d'un **logiciel de gestion de version de code** et le déploiement du modèle via une API
- La **conception du modèle,** son évaluation, et son interprétation compréhensible pour les métiers
- La **visualisation des données** et des résultats, à destination des chargés de clientèle

La partie features engineering est assez complexe, car elle nécessite de s'approprier la connaissance métier Crédit, ainsi que le contenu de tous les fichiers à disposition.

L'idée générale est de **systématiser** la création de features, via les fonctions min, max, mean, ou via la combinaison de features (rapport de 2 features, notamment montants, ...).

Etant donné qu'il ne s'agit pas du cœur du projet, vous inciterez l'étudiant à rechercher un **Kernel sur Kaggle** qui propose le codage de la préparation et la transformation des données.

Assurez-vous que l'étudiant s'approprie ce code, voire l'adapte en cas de besoin. La démarche est très formatrice au niveau technique Python de par l'utilisation par exemple de jointures, groupby, labelEncoder, oneHotEncoder, ...

En option, dans le cadre de l'optimisation du modèle, vous pourrez inciter l'étudiant à utiliser Hyperopt (optimisation des hyperparamètres).

L'étudiant devra mettre en œuvre **une fonction coût** adaptée au contexte Crédit, afin de proposer une optimisation orientée métier et non pas technique.

Par exemple le coût d'un faux positif (bon crédit considéré comme mauvais = constitue un manque à gagner modéré pour la banque, une perte de marge) est différent d'un faux négatif (mauvais crédit considéré comme bon = constitue une perte importante pour la banque, un défaut de paiement et/ou une perte de capital non remboursé).

Idéalement, il montrera que l'optimum « métier » est différent de l'optimum du score ou d'autres mesures purement « techniques ».

L'étudiant devra prendre grand soin à élaborer un **dashboard interactif,** en s'appuyant par exemple sur Dash ou Streamlit. Les ressources proposées dans l'énoncé permettent de répondre parfaitement au besoin. Ce sera l'occasion de prendre conscience de la difficulté à définir les bons graphiques et à concevoir un bon design, adaptés aux métiers, domaines de compétence des Data Analysts et Web Designers.

Le modèle final sera déployé sur le cloud à travers une API. Pour créer cette API vous pouvez proposer à l'étudiant d'utiliser la librairie Flask ou FastAPI. Le dashboard interactif devra donc communiquer avec l'API afin de récupérer la prédiction associée à un client. À noter que l'API et le dashboard seront déployés en ligne, sur l'application Heroku par exemple.

Concernant la mise en production de l'API et de l'application, plusieurs solutions s'offrent à l'étudiant, en particulier Heroku, Azure webapp et AWS.

Dans le cadre de l'utilisation de Heroku, devenu payant à partir de fin novembre 2022, vous avez la possibilité d'obtenir un crédit de $13 par mois pendant 12 mois si vous avez un compte "student" sur GitHub. Ce crédit couvre aisément les coûts liés à votre projet, estimés à moins de 10 euros, et qui inclut le temps pour la phase de test de mise en production.

Si GitHub lui demande un certificat de scolarité lorsqu'il passe son compte à un compte "student", il peut obtenir un certificat de scolarité OpenClassrooms en suivant cette procédure.

Quelque soit la solution Cloud choisie, l'étudiant et l'évaluateur veilleront à enregistrer pendant la soutenance la démo de l'application en production, ce qui permettra au jury de visionner cette démo, sans que l'étudiant n'ait à maintenir son application sur le Cloud. Maintenir l'application dans le Cloud pourrait en effet engendrer des coûts.

Milestones

Voici un aperçu des étapes suggérées (mais NB que les étudiants ne sont pas tenus de réaliser les étapes du projet dans cet ordre). Cette section met également en évidence les difficultés potentielles. Veuillez noter :

- La vitesse d'avancement du projet pour chaque étape est donnée à titre indicatif mais varie en fonction de chaque élève.
- Le livrable fait référence aux résultats attendus de l'étudiant. Les étudiants ne sont pas non plus tenus de remplir ces "sous-livrables" suggérés pour valider le projet ; ils ne seront évalués que sur les livrables finaux.

Milestone 1 : Choix d'un Kernel

- **Livrable** :
  Notebook de préparation du jeu de données et de feature engineering
- **Niveau d'avancement** : 15 %
- **Problèmes et erreurs courants** :

    - La problématique est de trouver un kernel suffisamment intéressant d'un point de vue technique et de performance
    - Attention aux jeux de données mis à disposition sur Kaggle : seul le fichier application_train.csv contient les classes, il servira de base aux traitements à réaliser. Le fichier application_test.csv ne contient pas les classes car il est utilisé pour réaliser une prédiction à l'aveugle à soumettre à Kaggle, il ne sera pas exploitable pour entraîner un modèle
- **Recommandations** :
    - Le kernel LightGBM with Simple Features ou ses dérivés est un bon exemple car il a permis à son auteur une performance parmi les meilleures, et il met en œuvre des techniques d'agrégation très intéressantes qui permettent un code très concis
    - L'étudiant ne gardera que le feature engineering de ce kernel, il devra lui-même réaliser la partie entraînement des différents modèles et leur optimisation
    - Attention en sortie du feature engineering certains champs sont à Nan ou Infini, qu'il faudra retraiter car pas compatibles avec certains modèles
    - L'étudiant veillera à adapter le kernel utilisé afin de rendre son projet plus authentique. Cela passe par exemple par l'ajout de commentaires sur les étapes du projet.
    - L'étudiant pourra exploiter d'autres kernels intégrant une analyse exploratoire
- **Ressources :**

    Kernel pour le feature engineering : LightGBM with Simple Features

Milestone 2 : Développement et simulation de modèles

- **Livrable** :

    Notebook de simulation et comparaison des modèles
- **Niveau d'avancement** : 50 %
- **Problèmes et erreurs courants** :
    - Erreur de mesurer l'accuracy qui donne un score 0.92 pour le DummyClassifier avec comme stratégie le choix de la classe la plus représentée …
    - Erreur de comparer les modèles sur les scores du fichier test
    - Erreur de data leakage, par exemple utiliser Smote sur le fichier train avant cross validation

- **Recommandations** :
    - Comparés à un algorithme de base de type DummyClassifier
    - Via un GridsearchCV pour la Cross-Validation et l'optimisation des hyperparamètres
    - Avec différentes mesures adaptées : le temps de traitement (fit et predict) et dans le cas d'une classification avec des classes non équilibrées il est suggéré d'utiliser l'AUC
    - Par oversampling, via par exemple la librairie Smote, pour générer de nouveaux enregistrements de classe 1. Pour éviter le data leakage lors de la cross validation, l'oversampling doit se faire pour chaque jeu de données entraîné, donc uniquement sur les plis utilisés pour le fit et non sur le pli utilisé pour le predict
    - Ou par ajustement des poids des classes par paramétrage des modèles de type « class_weight »
    - Il est attendu de réaliser, comme pour le projet 4, une simulation complète de différents algorithmes :
    - L'AUC implique d'utiliser la fonction « predict_proba » (renvoie une probabilité pour chaque classe permettant d'élaborer une courbe ROC) ou s'il elle n'existe pas, son équivalent « decision_function » en veillant à la transformer pour obtenir des valeurs entre 0 et 1
    - Il est très fortement conseillé pour chaque simulation d'algorithme de ne pas se limiter aux valeurs par défaut des hyperparamètres, par exemple en testant plusieurs valeurs des 2 ou 3 principaux hyperparamètres afin d'obtenir des modèles mieux adaptés au contexte du projet. En effet, les hyperparamètres par défaut n'ont pas de sens particulier, pour comparer les modèles vous devez inciter l'étudiant à rechercher les hyperparamètres afin de les comparer équitablement. Ces valeurs doivent rester dans un spectre large afin de « dégrossir » l'optimisation des hyperparamètres
    - Pour le modèle choisi, une optimisation plus fine (« fine tuning ») permettra pour chaque hyperparamètre optimisé de tester des valeurs autour de la valeur déterminée lors de la première étape
    - Le choix des valeurs optimales des hyperparamètres est réalisée automatiquement dans le cadre du GridSearchCV, donc via la mesure de l'AUC du fichier train "cross_validé" (moyenne des scores des N splits de { (N-1) plis pour le fit et 1 pli pour le predict sur lequel est calculé le score AUC) }.
    - Le choix du meilleur modèle se fera donc avec la même mesure (AUC du fichier train "cross-validé", et non pas AUC du fichier test)
    - En complément du score AUC du train "cross-validé", la mesure du score AUC de chaque split de la cross validation du fichier train permet de vérifier la stabilité de ce score et donc la généralisation du modèle

    - En complément du score AUC du train “cross-validé”, il est fortement conseillé d’afficher en parallèle l’AUC du fichier test en tant que contrôle de cohérence, afin de mettre en évidence un éventuel problème de stratification dans le train_test_split ou bien une éventuelle dérive de type data leakage au niveau du fichier train “cross-validé” (par exemple Smote sur l’ensemble du fichier train avant cross validation)
    - L’étudiant devra prendre en compte le fait que le jeu de données est déséquilibré (92% de bon clients) en appliquant un traitement de rééquilibrage par exemple (non exhaustif) :
- **Ressources :**
    - 10 Techniques to deal with Imbalanced Classes : https://www.analyticsvidhya.com/blog/2020/07/10-techniques-to-deal-with-class-imbalance-in-machine-learning/
    - How to Effectively Predict Imbalanced Classes : https://towardsdatascience.com/how-to-effectively-predict-imbalanced-classes-in-python-e8cd3b5720c4
    - Librairie Imbalanced learn (Smote et autres modules) : https://github.com/scikit-learn-contrib/imbalanced-learn
    - SMOTE tutorial : https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/
    - SMOTE et GridsearchCV : https://stackoverflow.com/questions/50245684/using-smote-with-gridsearchcv-in-scikit-learn
    - Handle Class Imbalance using Class Weight : https://vitalflux.com/class-imbalance-class-weight-python-sklearn/
    - Vidéo tuto « Class Weights for Handling Imbalanced Datasets » : https://www.youtube.com/watch?v=Kp31wfHpG2c

Milestone 3 : Fonction de coût métier – optimisation du modèle d’un point de vue métier

- **Livrable** :
  Notebook partie « Finalisation de l’outil de scoring - optimisation métier »
- **Niveau d’avancement** : 55%
- **Problèmes et erreurs courants** :
  Erreur de comparer les modèles sur les scores du fichier test
- **Recommandations** :
    - Le modèle final doit pouvoir déterminer une classe 0 ou 1, ce qui implique de déterminer le seuil à partir duquel la proba calculée se transforme en classe 1 (pour un predict le seuil par défaut est 0.5). Une possibilité serait d’optimiser ce seuil en se basant sur la mesure « technique » du f1 score, mais ne prend pas en compte le contexte « métier »

- La problématique « métier » est de prendre en compte qu'un faux positifs (bon client considéré comme mauvais = crédit non accordé à tort, donc manque à gagner de la marge pour la banque) n'a pas le même coût qu'un faux négatif (mauvais client à qui on accorde un prêt, donc perte sur le capital non remboursé). Un faux négatif est environ 10 fois plus coûteux qu'un faux positif. Les mesures techniques tels que le f1 score ne le prennent pas en compte
- Il est attendu de mettre en œuvre une approche simple qui consiste, une fois le modèle choisi et les hyperparamètres optimisés finement d'un point de vue technique via l'AUC (CF étape précédente), de calculer une fonction de coût métier de type 10*FN + FP (où FN = nombre de FN dans la matrice de confusion pour un seuil donné, FP = nombre de FP) et de trouver son minimum pour un seuil donné
- Une possibilité de fonction métier adaptée au projet est la fonction fbeta_score de sklearn, qui permet d'attribuer plus de poids à la minimisation des FN à travers la pondération du paramètre beta
- Il est recommandé d'effectuer une nouvelle recherche des hyper-paramètres se basant sur la fonction métier proposée, de cette façon, ils seront choisis de sorte à minimiser la perte pour l'entreprise
- D'autres approches plus complexes seraient de remonter cette mesure de calcul de coût métier (avec optimisation du seuil) dans la simulation fine du modèle final choisi, voire pour tous les modèles (cette mesure remplacerait l'AUC)

Milestone 4 : Analyse de la « feature importance » globale et locale

- **Livrable** :
  Notebook partie « Feature importance »
- **Niveau d'avancement** : 65%
- **Recommandations** :
  - L'objectif métier est de donner des informations pertinentes d'analyse au chargé d'étude, afin qu'il comprenne pourquoi un client donné est considéré comme bon ou mauvais prospect et quelles sont ses données / features qui expliquent son évaluation. Il est donc nécessaire à la fois, de connaître d'une manière générale les principales features qui contribuent à l'élaboration du modèle, et de manière spécifique pour le client qu'elle est l'influence de chaque feature dans le calcul de son propre score (feature importance locale)
  - De nombreux algorithmes proposent des fonctionnalités de feature importance globale, mais spécifiques à l'algorithme par exemple « .coeff_ » pour la Régression Logistique ou « .feature_importances_ » pour le RandomForest

    - Il est attendu d'utiliser des librairies spécialisées de type Lime ou Shap (à privilégier), afin de fournir un calcul de feature importance globale indépendante de l'algorithme, et un calcul de feature importance locale, indispensable pour remplir les objectifs du projet en termes de restitution au chargé d'étude
- **Ressources :**
    - Interpreting machine learning models : https://towardsdatascience.com/interpretability-in-machine-learning-70c30694a05f
    - Exemples de traitement de feature importance : https://machinelearningmastery.com/calculate-feature-importance-with-python/
    - Lime tutorial : https://coderzcolumn.com/tutorials/machine-learning/how-to-use-lime-to-understand-sklearn-models-predictions
    - Shap tutorial : https://coderzcolumn.com/tutorials/machine-learning/shap-explain-machine-learning-model-predictions-using-game-theoretic-approach
    - Lime vs Shap - Article : https://blog.octo.com/introduction-a-linterpretation-de-modeles-de-machine-learning/
    - Lime vs Shap - Exemple de code : https://colab.research.google.com/drive/1pjPzsw_uZew-Zcz646JTkRDhF2GkPk0N

Milestone 5 : Elaboration du dashboard

- **Livrable** :
  Application Dashboard pour le chargé d'étude
- **Niveau d'avancement** : 100%

- **Recommandations** :
    - Le tableau de bord intégrera au minimum les informations client suivantes :
        - N° client, crédit accepté ou non, score détaillé sous forme de jauge colorée selon qu'il est en dessous ou au-dessus du seuil : permet de juger s'il est loin du seuil ou non.
        - Sa feature importance locale sous forme de graphique, qui permet au chargé d'étude de comprendre quelles sont les données du client qui ont le plus influencé le calcul de son score
    - Le tableau de bord présentera également d'autres graphiques sur les autres clients :
        - Deux graphiques de features sélectionnées dans une liste déroulante, présentant la distribution de la feature selon les classes, ainsi que le positionnement de la valeur du client
        - Un graphique d'analyse bi-variée entre les deux features sélectionnées, avec un dégradé de couleur selon le score des clients, et le positionnement du client
        - La feature importance globale
        - D'autres graphiques complémentaires, par exemple :
    - Il est attendu de réaliser 2 applications :
        - Une API qui au minimum renvoie le score d'un client (moteur d'inférence qui réalise le predict à partir du modèle final sauvegardé)
        - Une application Dashboard qui visualise les informations du client et les graphiques associés, et qui appelle l'API via une url pour récupérer le score du client
        - Ces applications seront gérées avec un logiciel de gestion de version comme Git (faire des commit), et l'ensemble du code déposé sur Github via push à partir de Git
    - L'API peut être réalisée par exemple en Flask (conseillé) ou Django (plus complexe)
    - La réalisation technique du Dashboard peut s'appuyer sur les librairies Dash ou Bokeh, ainsi qu'avec Streamlit qui met également à disposition un serveur cloud pour exécuter l'application
    - Conception fonctionnelle du Dashboard :
    - Architecture du Dashboard :
    - Les applications (Dashboard et l'API) seront hébergées sur un serveur cloud, idéalement sur un serveur gratuit PythonAnywhere, Heroku, Streamlit, qui doit rester accessible jusqu'au jury de fin de parcours
- **Ressources :**
    - Git & GitHub : Le Cours Pour Les Débutants (bash) : https://www.youtube.com/watch?v=4o9qzbssfII
    - TUTO GIT GUI: le guide complet : https://www.youtube.com/watch?v=cRounphvOC8
    - Graphics Principles Cheat Sheet : https://www.psiweb.org/docs/default-source/2018-psi-conference-posters/48-julie-jones.pdf
    - Dash documentation : https://dash.plotly.com/installation

- Bokeh documentation : https://docs.bokeh.org/en/latest/
- Créez une API avec Flask : https://openclassrooms.com/fr/courses/4525361-realisez-un-dashboard-avec-tableau/5774811-creez-une-api-avec-flask
- API Flask : http://web.univ-ubs.fr/lmba/lardjane/python/c4.pdf
- Streamlit : https://streamlit.io/
- Pythonanywhere : https://www.pythonanywhere.com/
- Heroku : https://www.heroku.com/

🕵️ Évaluation des compétences

🎯Déployer un modèle via une API dans le Web

❒ Un fichier (par exemple format pickle) contenant un modèle de machine learning sérialisé a été créé et le modèle chargé depuis le fichier fonctionne

❒ Le modèle de machine learning est déployé sous forme d'API (via Flask par exemple) et cette API renvoie bien la prédiction correspondante à un client en réponse à un identifiant client

❒ L'API est déployée sur le web via un outil gratuit et disponible plusieurs mois (en vue du jury)

❒ L'API est appelée par l'application Dashboard via une requête. Ces deux applications sont indépendantes.

🎯Réaliser un dashboard pour présenter son travail de modélisation

❒ Au moins un parcours utilisateur simple permettant de répondre aux besoins des utilisateurs (les différentes actions/clics sur les différents graphiques permettant de répondre à une question que se pose l'utilisateur) a été décrit et conçu

❒ Au moins deux graphiques interactifs permettant aux utilisateurs d'explorer les données clients ont été développés (l'objectif est de répondre à des questions comme "quel est le client avec le plus de transactions ?")

❒ Les graphiques réalisés respectent les règles de lisibilités et de clarté (cf. ressource pédagogique dans le projet)

❒ Les graphiques réalisés sont pertinents (ils permettent de répondre à la problématique métier)

❒ Le dashboard est accessible pour d'autres utilisateurs sur leurs postes de travail (déploiement dans le web)

🎯Présenter son travail de modélisation à l'oral

❒ la méthode d'évaluation de la performance du modèle de machine learning,

❒ la façon d'interpréter les résultats du modèle,

❒ la façon d'interpréter l'importance des variables du modèle,

❒... sont expliqués de manière simple et compréhensible par un public non technique

❒ L'étudiant fournit des explications claires, avec assurance

❒ L'étudiant a su répondre de manière simple (compréhensible par un public non technique) à au moins une question portant sur sa démarche de modélisation

❒ La démarche de modélisation présente une évaluation complète des modèles, en particulier la comparaison de plusieurs modèles via une Cross-Validation et une optimisation des hyperparamètres (via GridsearchCV, RandomizedSearchCV ou équivalent)

❒ La démarche de modélisation prend en compte le déséquilibre des classes (via un oversampling de type SMOTE ou via l'utilisation de "class_weight" par exemple)

🎯Rédiger une note méthodologique afin de communiquer sa démarche de modélisation

❒ La démarche de modélisation est présentée de manière synthétique dans la note (2 pages max)

❒ La fonction coût métier, l'algorithme d'optimisation et la métrique d'évaluation sont explicités (1 page max)

❒ L'interprétabilité globale et locale du modèle est explicitée (façon d'interpréter l'importance des variables n'est pas la même pour une régression logistique que pour un random forest) et les limites éventuelles sont précisées (1 page max)

❒ Les limites et les améliorations envisageables pour gagner en performance et en interprétabilité de l'approche de modélisation sont décrites (1 page max)

❒ Le rapport respecte la structure des document de ce type (cf. ressource pédagogique dans le projet)

❒ Les règles d'orthographe et de syntaxe élémentaires ont été respectées

🎯Utiliser un logiciel de version de code pour assurer l'intégration du modèle

❒ Un dossier contenant tous les scripts du projet a été créé dans un logiciel de version de code (ex : Git) et partagé (ex : Github)

❒ L'historique des modifications du projet affiche au moins trois versions distinctes et l'on peut accéder à ces anciennes versions

❒ La liste des packages utilisés ainsi que leur numéro de version est disponible et tenu à jour

❒ Le travail est réutilisable par d'autres personnes et la collaboration est facilitée :

- un fichier introductif permet de comprendre l'objectif du projet et le découpage des dossiers
- les scripts et les fonctions sont commentés

🎯Respect des consignes

❒ Les livrables sont complets

❒ Les livrables ont été déposés 48h à l'avance

❒ Le temps de présentation est bien géré par l'étudiant

Une soutenance mérite un refus dans les cas suivants :

- Critères d’évaluation non-validés pour une ou plusieurs compétences.
- Plagiat (veillez à poser des questions méthodologiques ou sur la raisonnement de la solution pour s’assurer que le travail a bien été réalisé par l’étudiant)
- Une présentation en dessous de 10 min ou au-dessus de 20 minutes. Néanmoins, une tolérance de +/- 20% est permise, selon les cas.