Apply some concepts & tools seen in the 3 sessions of the course

M2 Big Data and Machine Learning en Apprentissage, EFREI

Année universitaire 2022/2023

Rapport préparé sous la direction de : HADJEM Medina

Élèves : ANGELOT Quentin / ALLIE Arthur / FARAMOND Emile

Introduction

Dans le cadre de notre M2 réalisé à l'EFREI Paris, nous avons réalisé un projet en Application de Big Data.

Nous verrons dans ce rapport comment répondre aux questions suivantes :

●Partie 1 : Construire des projets ML classiques en respectant les meilleures pratiques de codage ML de base.

●Partie 2 : Intégrer MLFlow à votre projet.

● Partie 3 : Intégrer l'interprétabilité ML à votre projet.

Le code source du projet est disponible au lien suivant : https://github.com/ArthurA14/mlflow_project

Remarque : Le code a été pensé pour être industrialisé (commentaire de fonction et fichiers séparés).

Table des matières

1. Présentation des données

Pour ce projet nous avons utilisé le dataset Kaggle disponible au lien suivant : https://www.kaggle.com/c/home-credit-default-risk/data.

Différents dataset

- **application_{train|test}.csv**
  - Il s'agit du tableau principal, divisé en deux fichiers pour Train (avec TARGET) et Test (sans TARGET).
  - Données statiques pour toutes les applications. Une ligne représente un prêt dans notre échantillon de données.

- **bureau.csv**
  - Tous les crédits précédents du client fournis par d'autres institutions financières qui ont été rapportés au Credit Bureau (pour les clients qui ont un prêt dans notre échantillon).
  - Pour chaque prêt de notre échantillon, il y a autant de lignes que le nombre de crédits que le client avait dans le Crédit Bureau avant la date de la demande.

- **bureau_balance.csv**
  - Soldes mensuels des crédits précédents dans Credit Bureau.
  - Ce tableau a une ligne pour chaque mois d'historique de chaque crédit précédent rapporté au Credit Bureau

- **POS_CASH_balance.csv**
  - Instantanés des soldes mensuels des prêts POS (point of sales) et cash précédents que le demandeur a eu avec Home Credit.
  - Cette table a une ligne pour chaque mois de l'historique de chaque crédit précédent dans Home Credit (crédit à la consommation et prêts en espèces).

- **credit_card_balance.csv**
  - Instantanés des soldes mensuels des cartes de crédit précédentes que le demandeur possède auprès de Home Credit.
  - Cette table a une ligne pour chaque mois de l'historique de chaque crédit précédent dans Home Credit (crédit à la consommation et prêts de trésorerie) lié aux prêts dans notre échantillon - c'est-à-dire que la table a (#prêts dans l'échantillon * # de cartes de crédit précédentes relatives * # de mois où nous avons un historique observable pour la carte de crédit précédente) lignes.

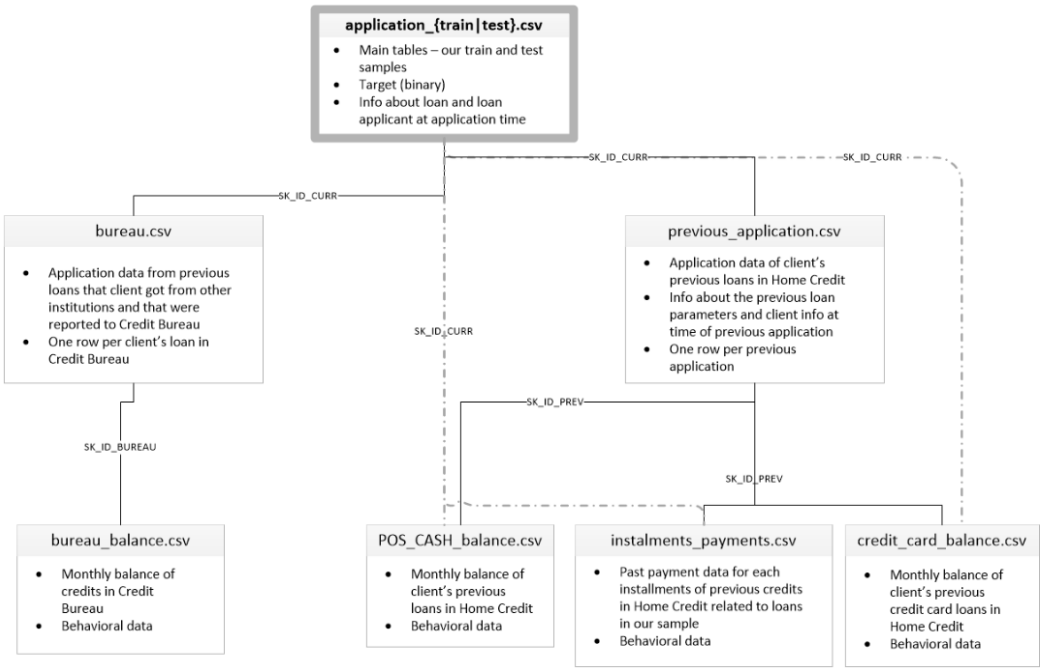
- **previous_application.csv**

- Toutes les demandes précédentes de crédit immobilier des clients qui ont des prêts dans notre échantillon.
- Il y a une ligne pour chaque demande précédente liée aux prêts dans notre échantillon de données.

- **installments_payments.csv**
  - Historique des remboursements pour les crédits déboursés précédemment dans Home Credit liés aux prêts de notre échantillon.
  - Il y a une ligne pour chaque paiement qui a été effectué plus b) une ligne pour chaque paiement manqué.
  - Une ligne est équivalente à un paiement d'un versement OU à un versement correspondant à un paiement d'un crédit Home Credit précédent lié aux prêts de notre échantillon.

- **HomeCredit_columns_description.csv**
  - Ce fichier contient les descriptions des colonnes dans les différents fichiers de données.

Schéma :

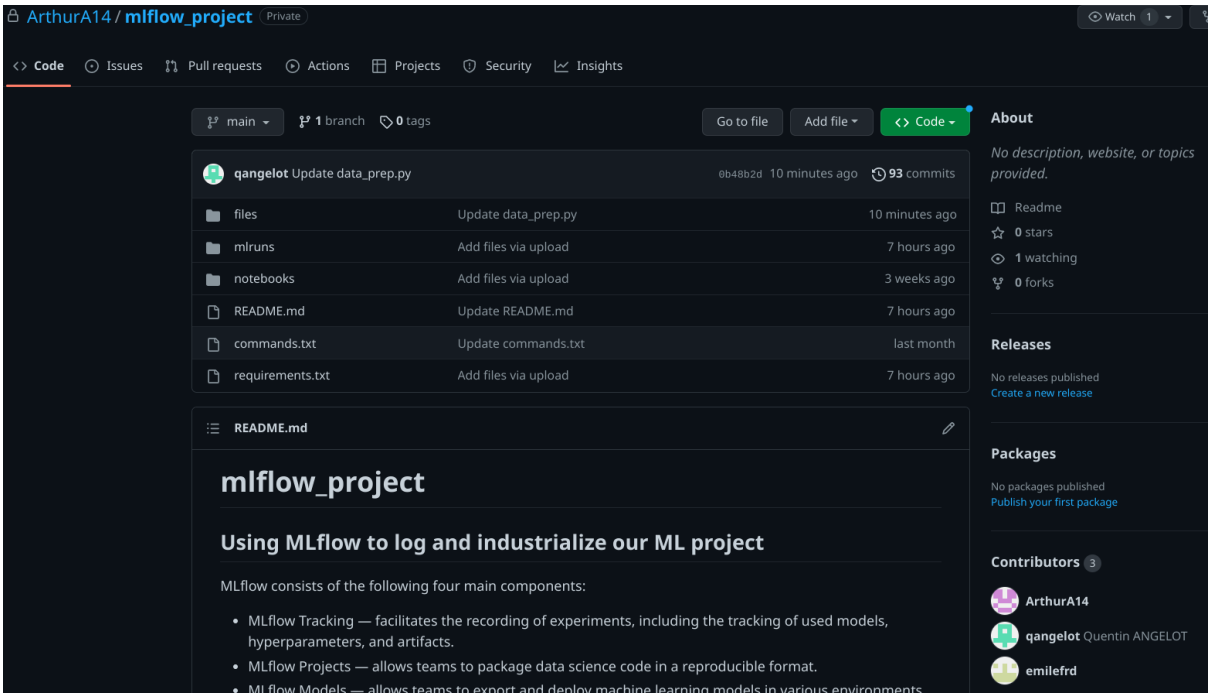


2. Notre projet de Machine Learning avec les bonnes pratiques

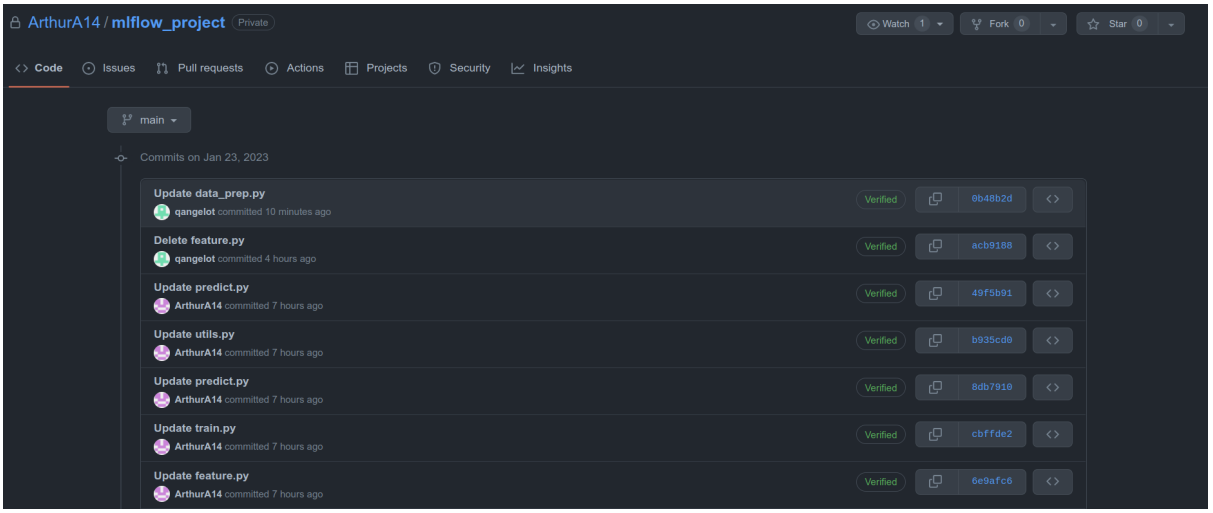
a. Utilisez GIT pour la collaboration entre les équipes, le code et les versions du modèle

Importance : L'utilisation de GIT dans un projet de Machine Learning est cruciale pour la collaboration entre les équipes. Cela permet de faciliter la gestion des différentes versions du code et des modèles, ainsi que de faciliter la collaboration entre les membres de l'équipe. En utilisant GIT, il est donc possible de suivre les modifications apportées au code et de gérer les conflits de manière efficace.

Cas d'usage :



Nous avons hébergé notre projet sur Github. Il y a 3 contributeurs. Cet outil nous permet de gérer l'hébergement du code et de gérer les conflits de manière efficace.

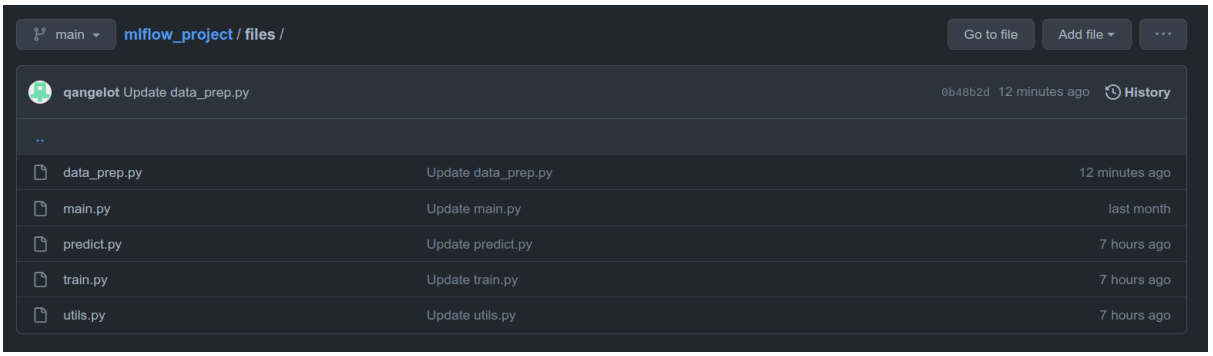


Comme on peut le voir, chaque collaborateur du projet à apporter son lot de commits.

b. Séparez le flux de travail de votre projet ML en différents scripts

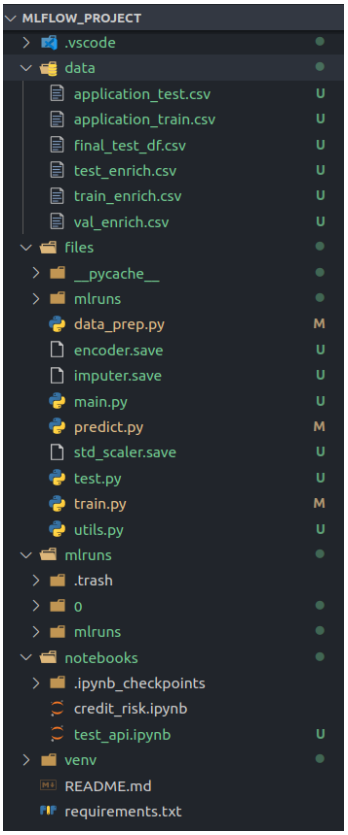
Importance : Séparer le flux de travail de notre projet ML en différents scripts permet de structurer le projet de manière claire et de faciliter l'évolution de celui-ci. Cela permet également de faciliter la maintenance et la compréhension du projet pour les différents membres de l'équipe.

Cas d'usage :



Nous avons créé 5 scripts python, ils sont tous bien différenciés et ont chacun une tâche qui leur est propre.

c. Utilisez un modèle cookie cutter ou adaptez/définissez le vôtre



Utiliser un modèle cookie cutter ou adapter/définir le vôtre est important pour obtenir des résultats de qualité dans un projet de Machine Learning. En utilisant un modèle pré-entraîné, il est possible de gagner du temps et de bénéficier des avancées récentes dans le domaine. Cependant, il peut être nécessaire d'adapter ou de définir un nouveau modèle pour s'adapter aux spécificités de votre projet.

Ici, nous avons un découpage précis entre les fichiers sources contenant la logique applicative, le dossier data comportant les différents datasets bruts et transformés, le dossier mlruns contenant les checkpoints MLflow et le dossier notebook qui contient les différents notebooks de recherches et de tests. Le dossier venv contient les différentes librairies que nous avons installées dans notre environnement virtuel avec le fichier requirements.txt.

d. Utilisez un environnement Conda pour toutes vos bibliothèques

Importance : Utiliser un environnement Conda pour toutes vos bibliothèques permet de gérer efficacement les dépendances de votre projet. Cela évite les problèmes de compatibilité entre les différentes bibliothèques et permet de faciliter la reproduction du projet sur d'autres machines. Cela permet également de partager facilement les dépendances nécessaires à l'exécution du projet avec les autres membres de l'équipe.

Cas d'usage :

```
Conda activate new environment :
https://uoa-eresearch.github.io/eresearch-cookbook/recipe/2014/11/20/conda/

> conda create -n myvenv python=3.9 anaconda
> conda activate myvenv
```

```
-> Puis utiliser Anaconda Prompt :

> conda install numpy   | conda install -n myvenv numpy
> conda install pandas
> conda install sklearn
> conda install matplotlib
> conda install seaborn
> conda install mlflow
```

3. Intégration de ML flow

a. Intégrer MLFlow à votre projet

Importance : Intégrer MLFlow dans un projet de Machine Learning est important pour une meilleure gestion et une meilleure traçabilité des différents modèles testés. Cela permet de suivre l'évolution des performances des modèles au fil du temps et de faciliter la collaboration au sein de l'équipe.

Cas d'usage :

```python
# Create experiment
EXPERIMENT_NAME = "mlflow-demo"
try:
    EXPERIMENT_ID = mlflow.create_experiment(EXPERIMENT_NAME)
except Exception as e:
    logger.exception("Experiment ID already set. Error: %s", e)
```

Ici, nous créons une expérience MLflow afin, entre autres, de logger les performances des modèles.

b. Installez MLFlow dans votre environnement python

Importance : L'installation de MLFlow dans l'environnement Python permet de bénéficier des fonctionnalités de suivi des paramètres et des métriques de votre modèle, ainsi que de l'affichage des résultats dans une interface dédiée.

Cas d'usage :

```
> conda install mlflow
```

c. Suivez les paramètres et les métriques de votre modèle et affichez les résultats dans votre interface mlflow locale (exécutions multiples)

Importance : Suivre les paramètres et les métriques de votre modèle est important pour comprendre comment celui-ci fonctionne et pour identifier les points à améliorer. Cela permet également de faciliter la comparaison entre différents modèles et d'optimiser les performances. Afficher les résultats dans une interface dédiée (comme celle fournie par MLFlow) permet de visualiser facilement ces informations et de les partager avec les autres membres de l'équipe.

Cas d'usage :

```
# Logging the current value of C
mlflow.log_param(key="C", value=param_value)

y_pred = model.predict(X_val_enrich)

# get metrics
c, f1score, accuracy = param_value, f1_score(y_val_enrich, y_pred), accuracy_score(y_val_enrich, y_pred)
print('c: ', param_value, ': ', 'f1_score: ', f1_score(y_val_enrich, y_pred), 'accuracy_score: ', accura
mlflow.log_metric("f1_score", f1score)
mlflow.log_metric("accuracy", accuracy)
```

Ici, nous explicitons les paramètres et les métriques que nous désirons logger.

Start Time	Duration	Run Name	User	Source	Version	Models	accuracy	f1_score	C	learning_rate	n_estimators	subsample
8 hours ago	2.4s	-	qan	predict.py	759848	-	-	-	-	-	-	-
8 hours ago	2.5s	-	qan	predict.py	759848	-	-	-	-	-	-	-
8 hours ago	1.0min	PARENT_RUN	qan	train.py	759848	sklearn	-	-	-	-	-	-
8 hours ago	1.4s	CHILD_RUN	qan	train.py	759848	-	0.688	0.259	0.0001	-	-	-
8 hours ago	2.5s	CHILD_RUN	qan	train.py	759848	-	0.69	0.261	0.001	-	-	-
8 hours ago	5.7s	CHILD_RUN	qan	train.py	759848	-	0.69	0.261	0.01	-	-	-
8 hours ago	9.8s	CHILD_RUN	qan	train.py	759848	-	0.69	0.261	0.1	-	-	-
8 hours ago	11.4s	CHILD_RUN	qan	train.py	759848	-	0.691	0.262	1	-	-	-
8 hours ago	19.0s	CHILD_RUN	qan	train.py	759848	-	0.691	0.261	10	-	-	-
8 hours ago	27.8s	PARENT_RUN	qan	train.py	759848	sklearn	0.73	0.284	-	0.05	10000	0.8

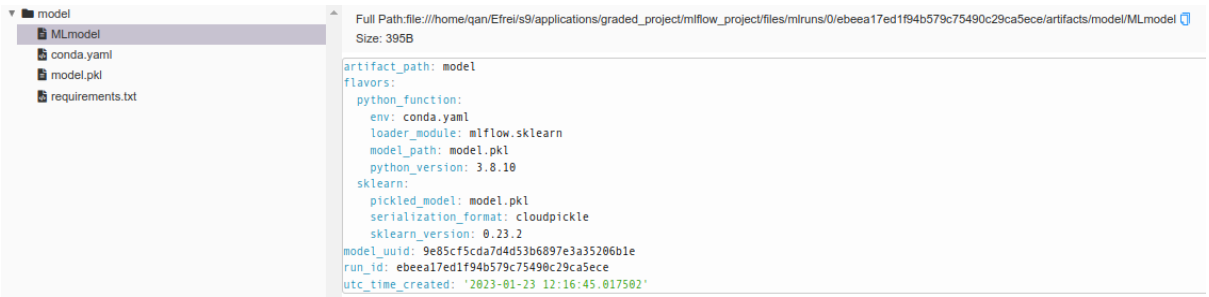
MLflow UI permet de suivre dans le détail les différents runs et de récupérer les meilleurs modèles et paramètres. On peut également récupérer les modèles pour la phase de prédictions.

d. Emballez votre code dans un format de modèle réutilisable et reproductible avec les projets ML Flow

Importance : Empaquetage du code dans un format de modèle réutilisable et reproductible avec MLFlow permet de faciliter la maintenance et la compréhension du projet pour les différents membres de l'équipe. Cela permet également de faciliter la reproduction du projet sur d'autres machines. C'est également utile pour une utilisation ultérieure, pour tester les différents modèles sur des données différentes.

Cas d'usage :

```
# HERE: put the path of the best model (found on MLflow ui)
logged_model = 'runs:/64d99d476e13491299604398404adae1/model'

# Load model as a PyFuncModel.
loaded_model = mlflow.pyfunc.load_model(logged_model)
```



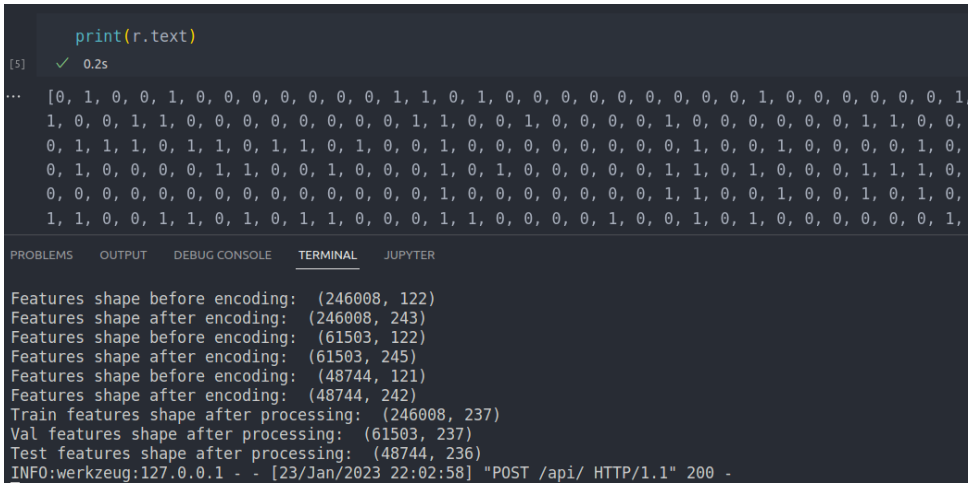
Les différents modèles sont enregistrés dans le dossier mlruns au fil des expériences. Ainsi, par l'intermédiaire de l'UI, nous pouvons récupérer l'emplacement exact du meilleur modèle et l'utiliser dans notre fonction predict afin de réaliser des prédictions sur les nouvelles données.

e. Déployer notre modèle dans un serveur REST local qui vous permettra d'évaluer les prédictions (facultatif) : nous construisons une API REST à cette fin

Importance : Déployer un modèle dans un serveur REST local permet d'évaluer les prédictions et d'utiliser le modèle en production. Cela peut être fait en utilisant une API REST pour exposer les fonctionnalités de prédiction du modèle. Cela permet également de faciliter l'intégration du modèle dans d'autres applications et de permettre aux utilisateurs finaux d'accéder aux prédictions de manière simple et efficace.

Cas d'usage :

```
@app.route('/api/', methods=['POST'])
def get_preds():

    f = request.files['file']
    if f and f.filename.rsplit('.', 1)[1].lower() == 'csv':

        stream = io.StringIO(f.stream.read().decode("UTF8"), newline=None)
        stream.seek(0)
        # result = stream.read()
        # io.StringIO(result)
        df = pd.read_csv(stream)


    # preprocessing pipeline
    _, _, df = data_prep(train, df)


    # get preds
    preds = predict(df)
    preds = preds.tolist()
    json_str = json.dumps(preds)

    return json_str
```



Ici, nous présentons le code de l'API et un aperçu des prédictions retourner par notre endpoint.

4. Interprétabilité

a. Intégrer ML Interpretability à notre projet

Importance : Intégrer ML Interpretability à un projet de Machine Learning est important pour comprendre comment les modèles prennent leurs décisions et pour identifier les points à améliorer. Cela permet également de rendre les modèles plus transparents pour les utilisateurs finaux.

b. Installer SHAP dans notre environnement python (sans oublier de l'ajouter à notre liste de librairies)

Importance : L'installation de SHAP dans l'environnement Python permet d'utiliser cette bibliothèque pour expliquer les prédictions de votre modèle.

Cas d'usage :

```
mlflow_project-main$ pip install shap
```

c. Utilisation de SHAP pour expliquer les prédictions de notre modèle

Importance : Utiliser SHAP pour expliquer les prédictions de votre modèle permet de comprendre les principaux facteurs qui ont conduit à ces prédictions. Cela permet également de rendre les modèles plus transparents pour les utilisateurs finaux.

Cas d'usage :

Les valeurs SHAP (acronyme de SHapley Additive exPlanations) décomposent une prédiction pour montrer l'impact de chaque variable. Où pourriez-vous utiliser ces valeurs ?

- Si un modèle indique qu'une banque ne devrait pas prêter de l'argent à quelqu'un, la banque est légalement tenue d'expliquer le fondement de chaque refus de prêt.
- Un prestataire de soins de santé veut identifier les facteurs qui déterminent le risque de maladie de chaque patient afin de pouvoir s'attaquer directement à ces facteurs de risque par des interventions sanitaires ciblées.

Les valeurs SHAP interprètent l'impact d'une certaine valeur pour une caractéristique donnée par rapport à la prédiction que nous ferions si cette caractéristique prenait une certaine valeur de base.

d. Construire un TreeExplainer et calculer les valeurs Shaplay

Importance : Construire un TreeExplainer et calculer les valeurs SHAP permet d'obtenir des explications détaillées des prédictions de votre modèle pour chaque point de votre ensemble de données.

Cas d'usage :

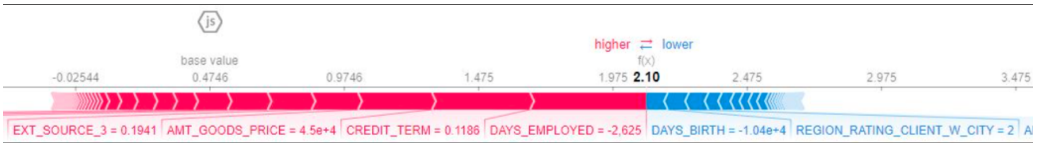
```
import shap  # package used to calculate Shap values

# Create object that can calculate shap values
explainer = shap.TreeExplainer(model)

# calculate shap values. This is what we will plot.
# Calculate shap_values for all of val_X rather than a single row, to have more data for plot.
shap_values = explainer.shap_values(X_test)
```

On applique le TreeExplainer sur le model et ensuite on l'utilise sur notre jeu de données de test.

e. Visualiser les explications pour un point spécifique de notre dataset

Importance : La visualisation des explications pour un point spécifique de votre ensemble de données permet de comprendre les principaux facteurs qui ont conduit à cette prédiction et de faciliter la compréhension.

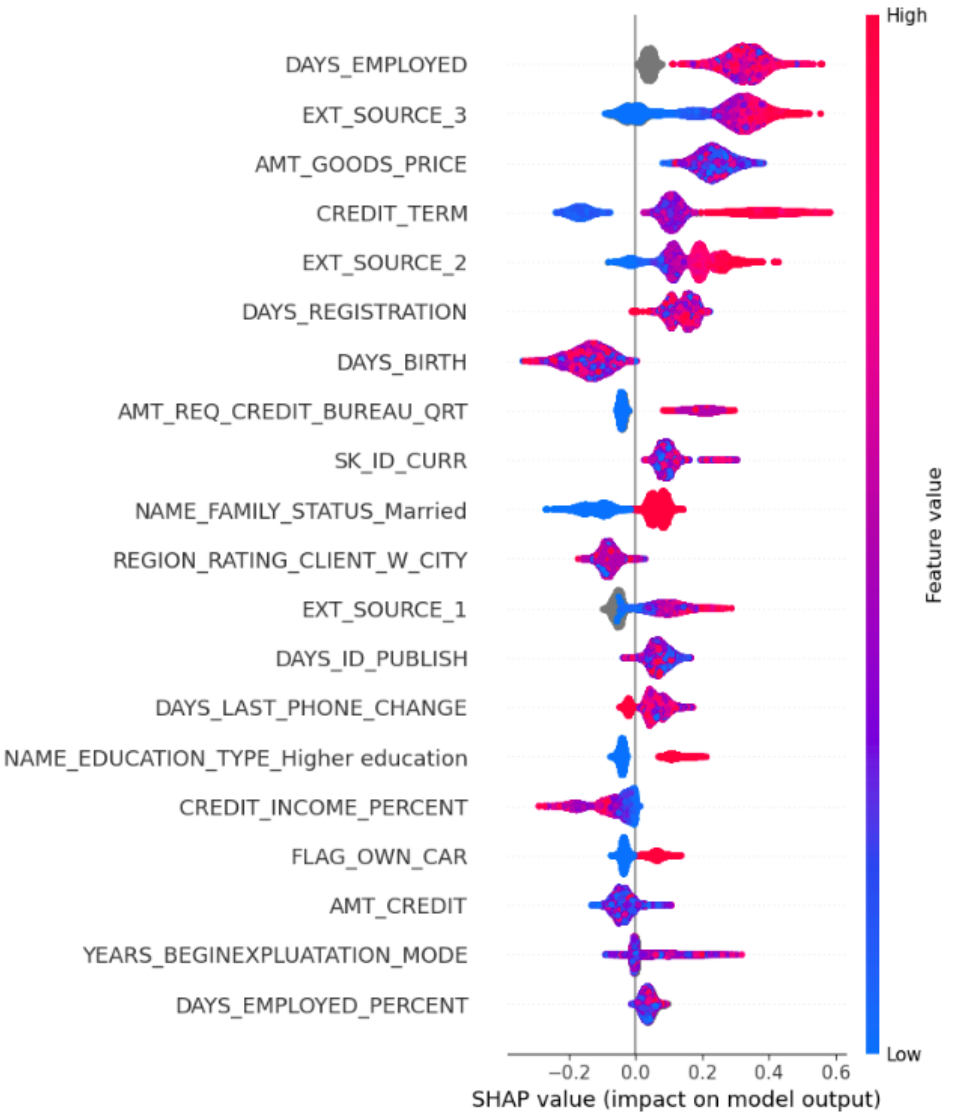
Cas d'usage :



f. Visualiser les explications pour tous les points du dataset

Importance : La visualisation des explications pour tous les points de notre ensemble de données à la fois permet de comprendre les tendances générales de votre modèle et de faciliter l'analyse globale des prédictions.

Cas d'usage :



Ce graphique est constitué de tous les points des données. Il montre les informations suivantes :

- Les variables sont classées par ordre décroissant.

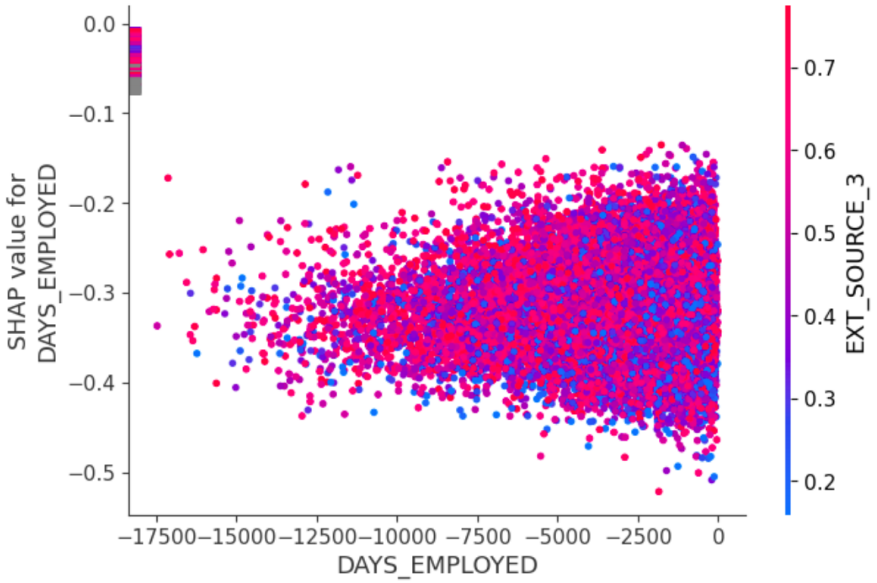
- Impact : L'emplacement horizontal montre si l'effet de cette valeur est associé à une prédiction plus élevée ou plus faible.
- Valeur originale : La couleur indique si cette variable est élevée (en rouge) ou faible (en bleu) pour cette observation.
- Corrélation : Un niveau élevé de la valeur de CREDIT_TERM a un impact élevé et positif sur le défaut de crédit. Le terme "élevé" vient de la couleur rouge et l'impact positif est indiqué sur l'axe des abscisses. On peut répéter l'analyse pour toutes les autres variables du jeu de données, par exemple, FAMILY_STATUS_Married nous montre que le fait d'être marié impact favorablement le bon remboursement d'un crédit.

Attention, les valeurs SHAP n'identifient pas la causalité, qui est mieux identifiée par la conception expérimentale ou des approches similaires.

g. Visualiser un graphique récapitulatif pour chaque classe du dataset

Importance : La visualisation d'un graphique récapitulatif pour chaque classe sur l'ensemble des données permet de comprendre les tendances générales de notre modèle et de faciliter l'analyse globale des prédictions.

Cas d'usage :



Conclusion

En conclusion, dans le cadre de notre projet de M2 réalisé à l'EFREI Paris, nous avons pu construire des projets de Machine Learning classiques en respectant les meilleures pratiques de codage ML de base.

Nous avons mis en place des méthodes pour la gestion des données, la sélection des modèles et la validation de leur performance.

En outre, nous avons intégré MLFlow à notre projet, permettant une meilleure gestion et une meilleure traçabilité des différents modèles testés. Cette intégration a permis de suivre l'évolution des performances des modèles au fil du temps et de faciliter la collaboration au sein de notre équipe.

Enfin, nous avons également intégré l'interprétabilité ML à notre projet, rendant nos modèles plus compréhensibles pour les utilisateurs finaux. Cette intégration a permis de rendre explicites les décisions prises par les modèles et de les rendre plus transparents pour les utilisateurs finaux.

Nous sommes convaincus que les compétences et les connaissances acquises au cours de ce projet nous seront très utiles dans notre futur parcours professionnel dans le domaine de l'IA.