

Projet 4

Identifiez les causes d'attrition au sein d'une ESN

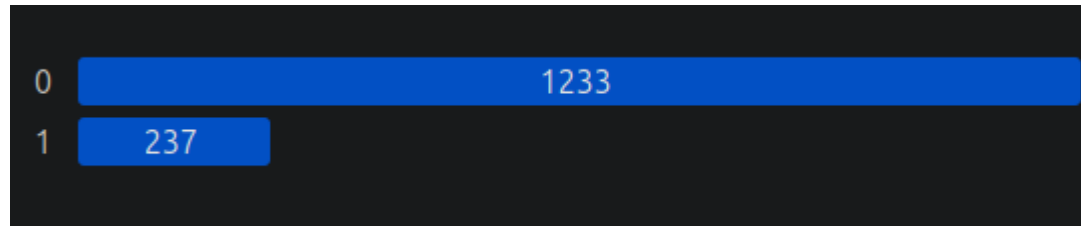
Exploration des données

Exploration des données

- 3 fichiers csv à regrouper selon un id
- Des colonnes inutiles car ayant une seule valeur
- Transformation de certaines valeurs, par exemple 'oui'/'non' => 0/1
- Des chaînes de caractères à transformer en nombre

Exploration des données

- On constate que la target est déséquilibrée, on a bien plus de personnes qui restent que de personnes qui partent
- Cela pose problème car prédire tout le temps qu'une personne reste donnerait un modèle qui se trompe que très rarement et semblerait plutôt bon



Répartition de la target (environ 16 % du dataset)

Stratégie pour lutter contre le déséquilibre

Trois méthodes pour nos modèles :

- **SMOTE (oversampling)** pour créer artificiellement de nouvelles données à partir de celles déjà existantes
- **L'undersampling** où on enlève une partie de la classe majoritaire pour régler le déséquilibre (pas retenue dans notre cas car on a trop peu de données)
- **Donner plus de poids aux gens qui partent** (avec `class_weight = 'balanced'`)

Stratégie pour lutter contre le déséquilibre

- On découpe notre dataset en respectant les proportions de notre target
- On utilise la cross-validation stratifié afin de conserver dans chaque fold la proportion de gens qui restent / gens qui partent

Feature engineering

- **Plusieurs ratios : la fréquence à laquelle une personne change d'entreprise, la fréquence des promotions ou encore de changement de poste**
- **On aide le modèle en créant des features booléennes, est-ce que la personne habite loin ? Est-ce qu'elle a un nouveau responsable ?**

Feature engineering

- On calcule également la compétitivité du salaire selon le poste et le niveau d'étude, on donne la valeur de -1 pour un salarié sous-payé, 0 s'il est dans la norme, 1 sinon
- On fait le calcul après le split train / test pour éviter le data leakage

Entraînement et évaluation des modèles

- Dans un premier temps on compare un dummy classifier, une régression logistique et une forêt aléatoire en utilisant la librairie imblearn et en utilisant notre méthode de cross-validation

Modèle	F1-Score	Recall	Precision
Dummy	0.1644	0.1579	0.1714
Reg. Logistique	0.5047	0.7053	0.3953
Random Forest	0.4058	0.2947	0.7004

Entraînement et évaluation des modèles

- On procède de la même manière mais cette fois avec l'importance des poids prenant en compte le déséquilibre des classes

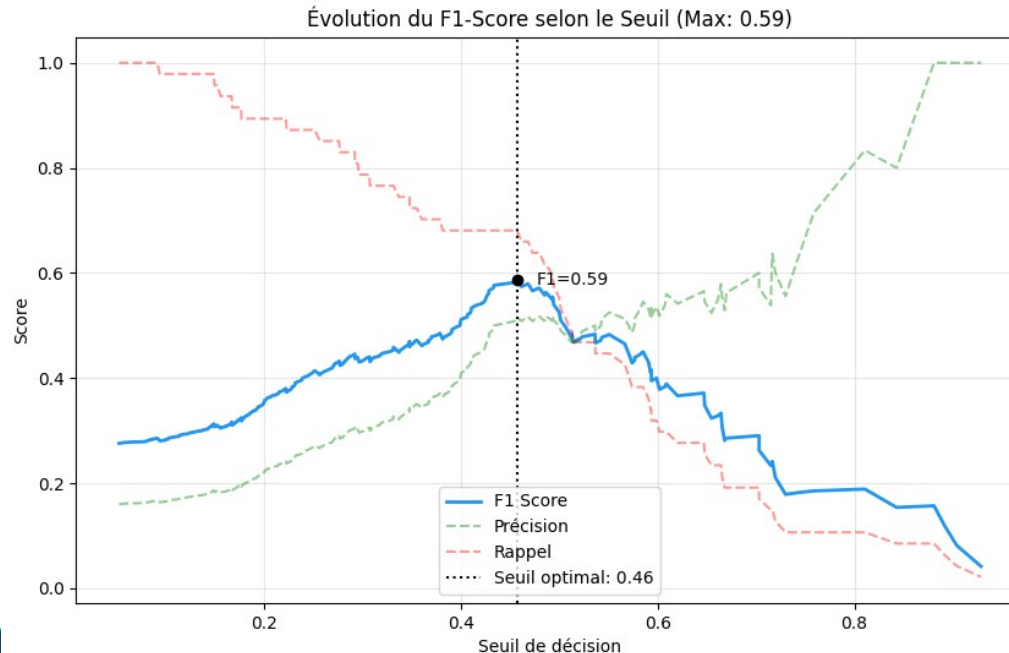
Modèle	F1-Score	Recall	Precision
Dummy	0.1644	0.1579	0.1714
Reg. Logistique	0.5072	0.7474	0.3863
Random Forest	0.5267	0.4947	0.5893

Recherche d'hyperparamètre et de seuil optimal

- On fait une recherche d'hyperparamètres pour les modèles suivants : régression logistique avec SMOTE, forêt aléatoire avec SMOTE et forêt aléatoire sans Smote mais avec `class_weight = 'balanced'` ou `'balanced subsample'`
'balanced_subsample' est spécifique aux forêts aléatoires et calcule des poids pour chaque arbre

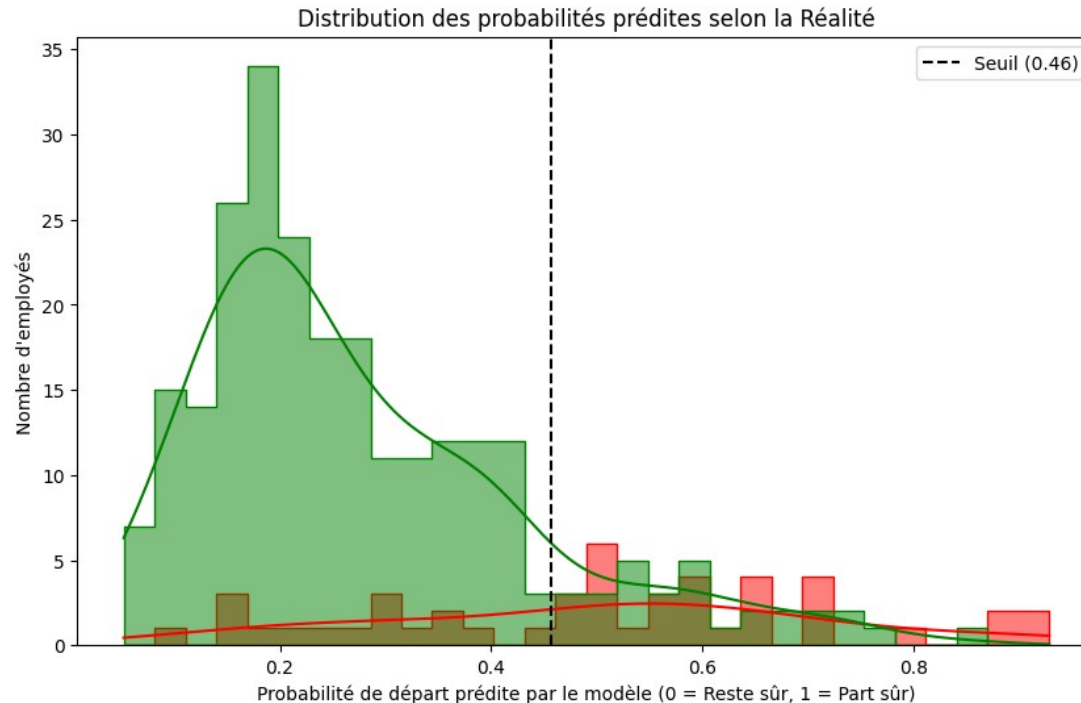
Recherche d'hyperparamètre et de seuil optimal

- Une fois le meilleur modèle récupéré (random forest sans SMOTE) on peut rechercher le seuil maximisant le f1 score



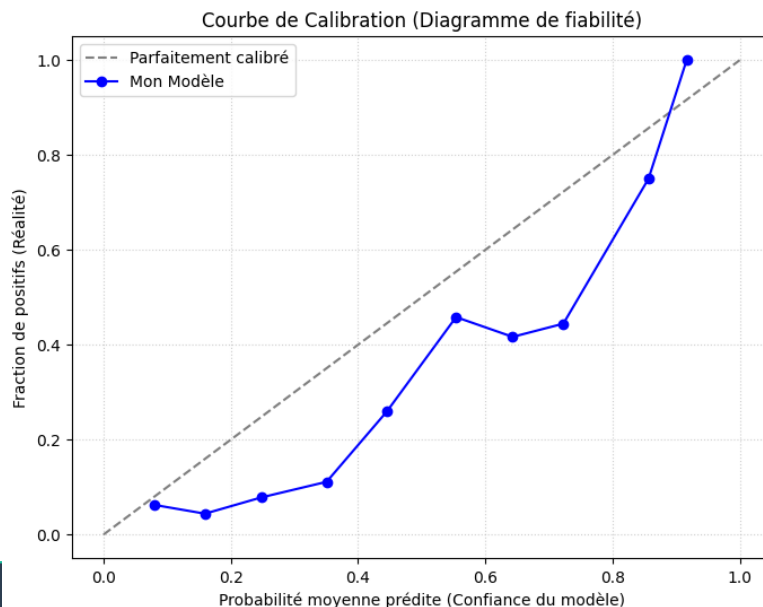
Confiance des prédictions

- On peut regarder de quelle manière le modèle se trompe



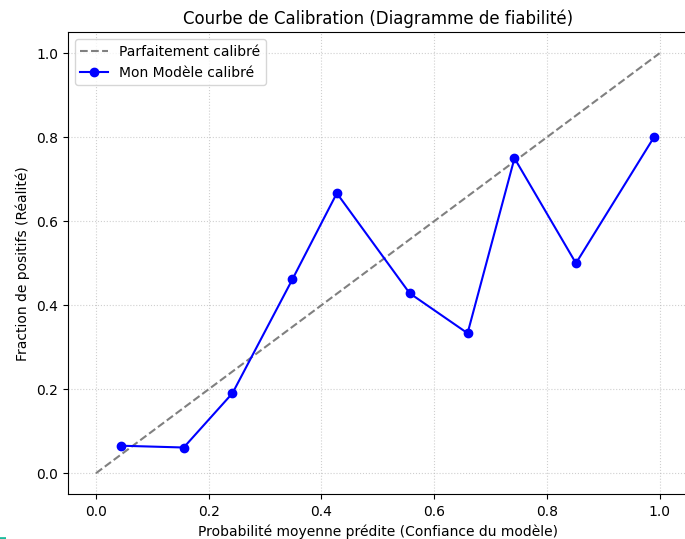
Confiance des prédictions

- La courbe de calibration nous renseigne sur les tendances du modèle, dans notre cas il a tendance à surestimer les chances qu'une personne a de partir



Confiance des prédictions

- On peut essayer de corriger cela avec `CalibratedClassifierCV` qui est une surcouche autour de notre modèle pour corriger l'exactitude des probabilités



Confiance des prédictions

- Le modèle obtenu après calibration a un score f1 un peu plus bas mais est par contre un peu meilleur dans certains cas, par exemple il est plus précis concernant le top 10 % des plus hautes probabilités

```
Précision sur le top 10.0% les plus risqués (Non Calibré) : 51.72%  
Précision sur le top 10.0% les plus risqués (Calibré)      : 58.62%
```

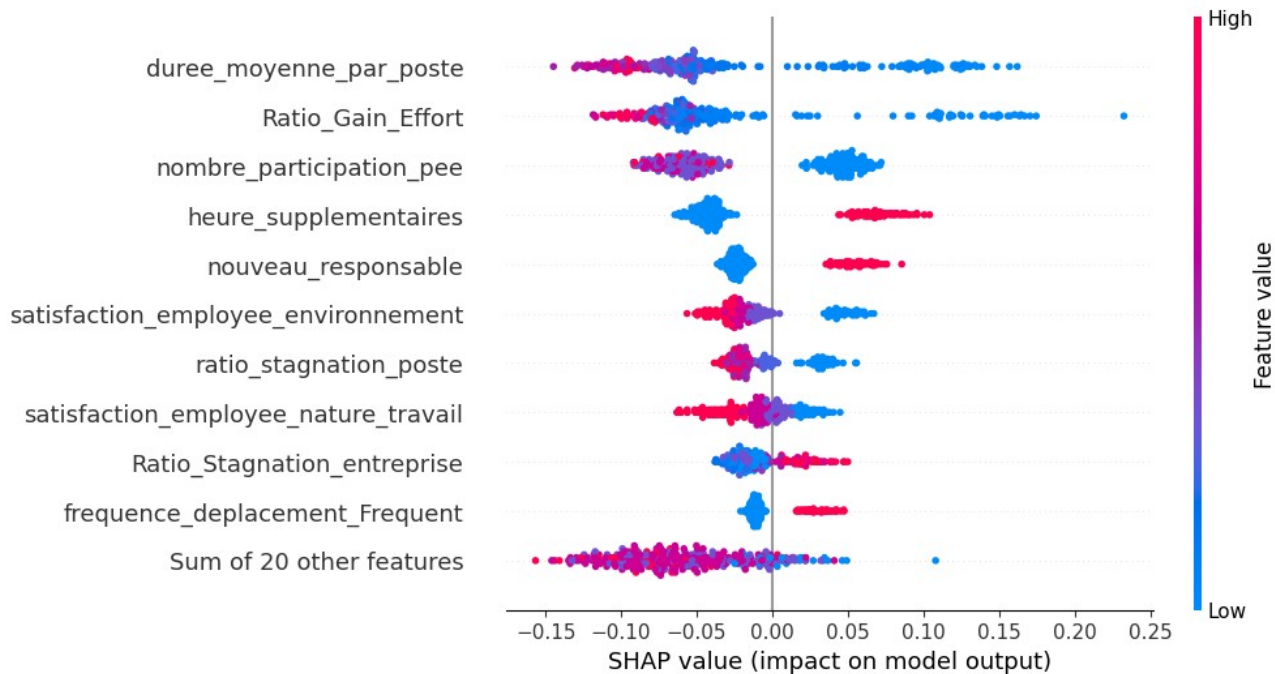
Importance des features

- On utilise les SHAP values ainsi que les permutations values afin de comprendre quelles features le modèle utilise pour faire ses prédictions

Cela nous permet également de juger de l'efficacité de notre feature engineering, pour une nouvelle feature si le score f_1 monte et son importance est haute dans les SHAP values c'est qu'on a trouvé un bon indicateur pour notre modèle

Importance des features

- Les 10 features ayant le plus gros impact sur les prédictions

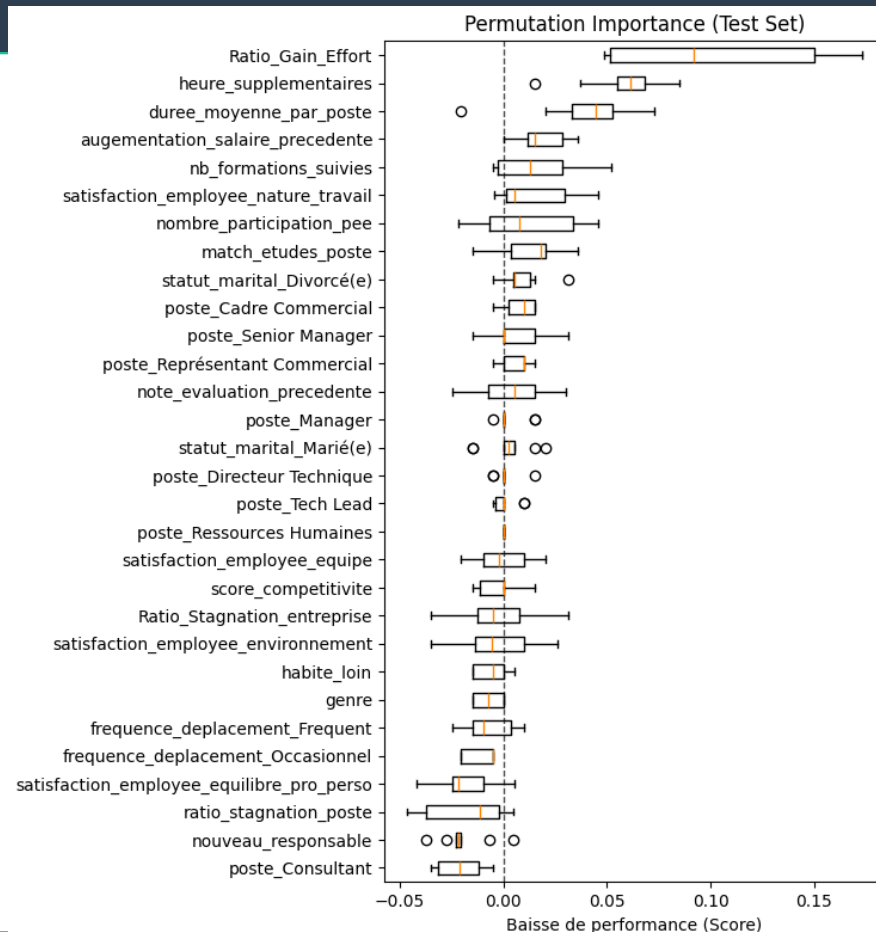


Importance des features

- Pour des variables comme `nombre_participation_pee`, la distinction n'est pas binaire. Le fait que les points rouges et bleus se chevauchent indique que l'impact de cette variable dépend du contexte (d'autres variables). Ce n'est pas un indicateur 'imprécis', mais un indicateur nuancé : participer au PEE n'a pas le même effet sur la rétention selon le profil de l'employé.

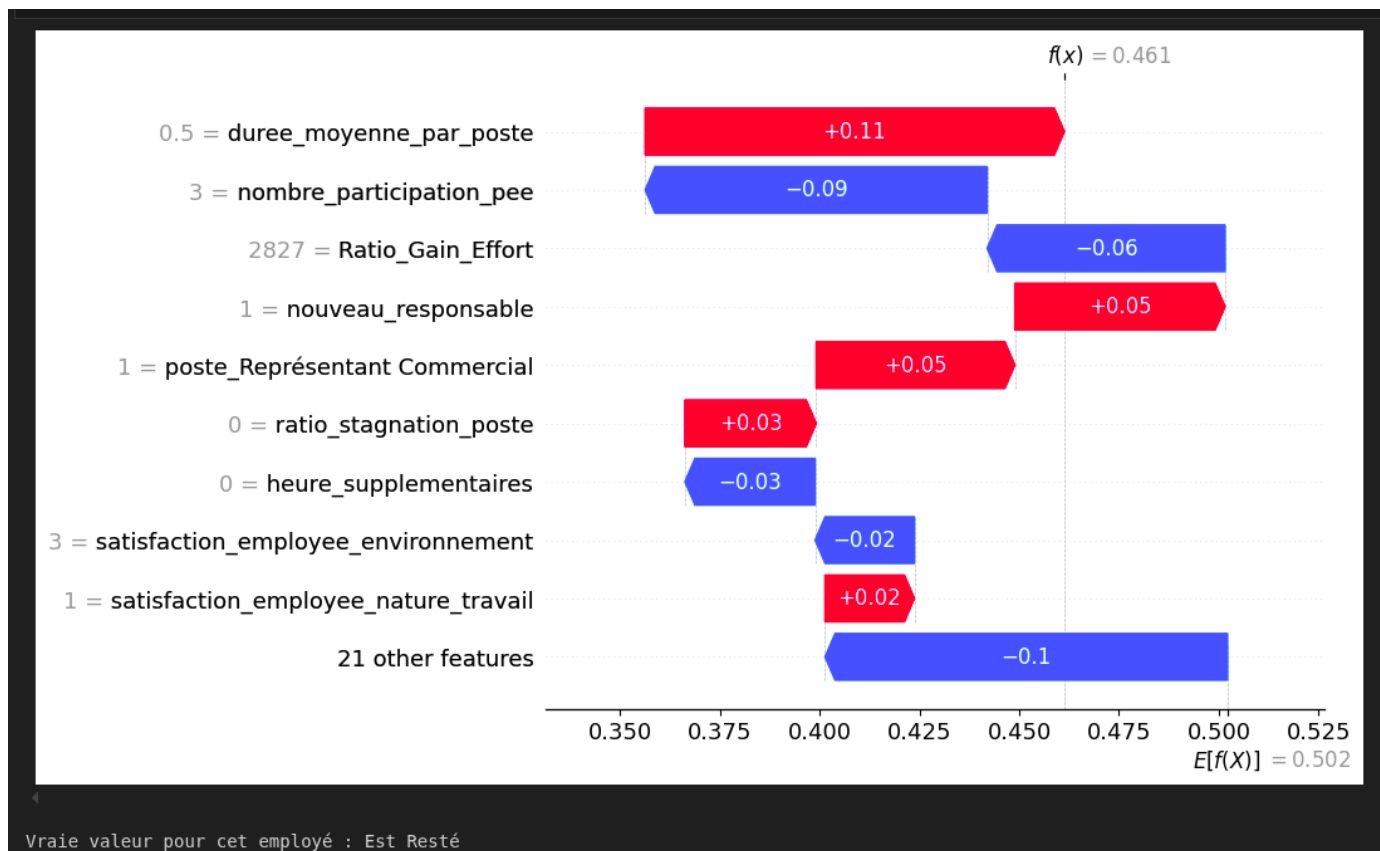
Importance des features

- Les permutations values indiquent que des features comme *nouveau_responsable* ont potentiellement un impact négatif sur le modèle bien que le modèle s'appuie beaucoup dessus (selon les shap values). Cependant enlever cette variable baisse significativement le f1 score .



Importance des features

- Importance locale des features obtenues avec SHAP



Importance des features

- Le modèle ne fait pas la bonne prédiction dans ce cas, on remarque que tout laisse penser qu'effectivement la personne pourrait rester

