Documentation

DATA SCIENCE

Table des matières

[Présentation du cas d’usage 3](#_Toc164075756)

[Objectif de l’étude 3](#_Toc164075757)

[Enjeux 3](#_Toc164075758)

[Environnement 3](#_Toc164075759)

[Outils 3](#_Toc164075760)

[Données 3](#_Toc164075761)

[Données disponibles 3](#_Toc164075762)

[Données utilisées pour l’analyse 3](#_Toc164075763)

[Variables explicatives 4](#_Toc164075764)

[Cycle de vie d’un projet Data Science 4](#_Toc164075765)

[Compréhension du problème 4](#_Toc164075766)

[Compréhension des données 5](#_Toc164075767)

[Préparation des données 5](#_Toc164075768)

[Modélisation 5](#_Toc164075769)

[Ajouts de nouveaux champs \ Transformation de données 5](#_Toc164075770)

[Séparation des données transformées en deux échantillons (apprentissage et test) 5](#_Toc164075771)

[Entrainement d’un modèle 6](#_Toc164075772)

[Prédiction sur l’échantillon de test 6](#_Toc164075773)

[Evaluation des performances 6](#_Toc164075774)

[Recherche d’amélioration du modèle 6](#_Toc164075775)

[Évaluation et déploiement 7](#_Toc164075776)

[Evolution des résultats du modèle 7](#_Toc164075777)

[Segmentation des données du modèle 7](#_Toc164075778)

[Résultat par segments 8](#_Toc164075779)

[Segment (présence client pas obligatoire / compteur est inaccessible) 8](#_Toc164075780)

[Matrice de confusion 8](#_Toc164075781)

[Résultats de classification 8](#_Toc164075782)

[Modèle 8](#_Toc164075783)

[Modèles de base faibles 9](#_Toc164075784)

[Optimisation par Gradient Boosting 9](#_Toc164075785)

[Régularisation 9](#_Toc164075786)

[Parallélisation et Optimisation des Performances 9](#_Toc164075787)

[Gestion des valeurs manquantes 10](#_Toc164075788)

[Technique 10](#_Toc164075789)

[Préparation de la donnée 10](#_Toc164075790)

[Création et Entrainement du modèle 11](#_Toc164075791)

[Séparation du modèle 11](#_Toc164075792)

[Initialisation du modèle XGBClassifier 11](#_Toc164075793)

[Entrainement du modèle 11](#_Toc164075794)

[Test et Vérification 13](#_Toc164075795)

[Matrice de confusion 13](#_Toc164075796)

[Courbe ROC 13](#_Toc164075797)

[Précision moyenne et Courbe de Rappel-Précision 14](#_Toc164075798)

[Importance des caractéristiques. 14](#_Toc164075799)

[Problème rencontrés et solutions 15](#_Toc164075800)

[Autres cas d’usage 15](#_Toc164075801)

[Prochaines étapes 15](#_Toc164075802)

# Présentation du cas d’usage

Nous cherchons à prédire les interventions vaines dans le futur. Une intervention vaine est une intervention ou e technicien n’est pas intervenu à cause de problème extérieure à ENEDIS.

# Objectif de l’étude

* Diminuer le nombre d’interventions vaines
* Anticiper et/ou éviter les reprogrammations
* Réduction du nombre de km parcourus
* Amélioration du contexte de travail des techniciens en intervention et des programmateurs en CPA

# Enjeux

* Construire un modèle ML prédictif pour savoir si une intervention risque d’être vaine
* Présenter les principales variables ayant un impact sur l’état de réalisation
* Mise à disposition des résultats avec des scores dans un rapport PowerBI
* Enrichir le rapport « réalisation » : mieux comprendre statistiquement les facteurs de risque d’une intervention

# Environnement

Le travail s’effectue dans l’environnement de **Pre Production**. Les données ont un décalage d’un mois seulement, et l’environnement est sécurisé & cloisonné

# Outils

Modèle réalisé en langage **Python sous Anaconda :**

* Permet d’avoir un environnement fonctionnel de base permettant l’utilisation des outils Jupyter pour programmer
* L’environnement est enrichi via l’installation de librairies Python recueillies à partir d’un artifactory.

# Données

## Données disponibles

Historique des interventions depuis 2018, ce qui représente environ 7 Millions d’interventions (85% Réalisées et 15% Vaines)

## Données utilisées pour l’analyse

Historique des interventions d’Octobre 2019 à Janvier 2023, ce qui représente environ 700 milles interventions (75% Réalisées et 25% Vaines)

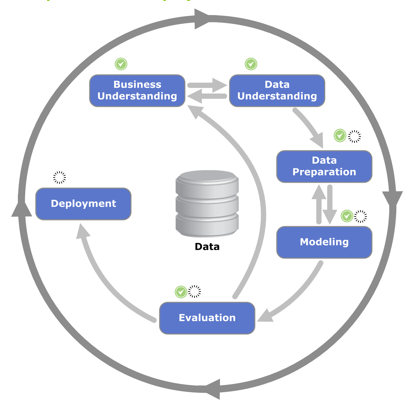
Les interventions récupérer sont des interventions Ginko (interventions clientèle) sur les DR Alpes, Aquitaine et Sillon Rhodanien et les agences de type AI (Agence d’intervention).

## Variables explicatives

Nous récupérons une trentaine de variables explicatives, dont voici un aperçu :

* Interventions précédentes
* Périmètre géographique
* Créneau
* Risque et contrainte
* Type/Domaine d’intervention
* Ouvrage
* Temps de gamme
* Moyens/Ressources théoriques
* Informations générales
* Origine

# Cycle de vie d’un projet Data Science



Les projets en data science suivent généralement un processus structuré pour transformer les données brutes en insights précieux. Ce processus peut varier légèrement selon les méthodologies, mais il inclut souvent les étapes suivantes :

* Compréhension du problème
* Compréhension des données
* Préparation des données
* Modélisation
* Évaluation et déploiement

## Compréhension du problème

Cette première étape est cruciale. Elle consiste à comprendre le problème que l'entreprise cherche à résoudre ou l'opportunité qu'elle souhaite saisir. Cela inclut la définition des objectifs du projet, la formulation de questions spécifiques que la data science peut aider à répondre, et l'établissement de critères de succès. Une bonne compréhension des besoins d'affaires oriente efficacement tout le projet.

## Compréhension des données

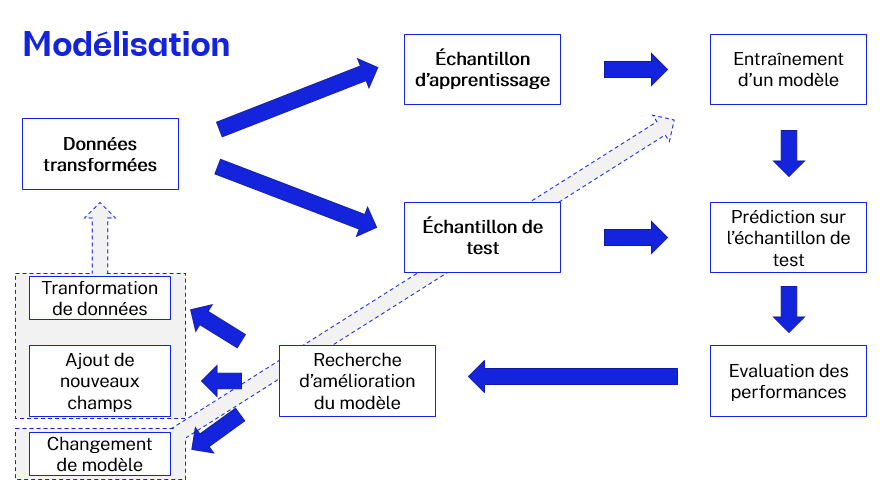
Après avoir défini clairement le problème, l'étape suivante consiste à collecter et à explorer les données disponibles pour avoir une bonne compréhension de celles-ci. Cela inclut l'identification des sources de données pertinentes, la collecte de ces données, et leur exploration initiale par des statistiques descriptives et la visualisation pour détecter des tendances, des anomalies ou des schémas préliminaires.

## Préparation des données

La préparation des données est souvent l'étape la plus longue et la plus laborieuse. Elle implique de nettoyer les données (traiter les valeurs manquantes, corriger les erreurs, etc.), de les transformer en un format approprié pour la modélisation (normalisation, encodage des variables catégorielles, etc.), et de créer de nouvelles variables (feature engineering) qui pourraient être utiles pour la modélisation.

## Modélisation

À cette étape, divers algorithmes et techniques de modélisation sont appliqués pour construire des modèles qui répondent aux objectifs définis lors de la première étape. Cela peut impliquer la sélection de plusieurs modèles, le réglage de leurs hyperparamètres, et l'utilisation de techniques pour évaluer leurs performances de manière robuste.



Nous allons brièvement détailler les différentes étapes dans le graphe ci-dessus :

### Ajouts de nouveaux champs \ Transformation de données

Concerne-la data préparation aborder précédemment.

### Séparation des données transformées en deux échantillons (apprentissage et test)

Après la préparation, les données sont souvent divisées en deux ensembles :

* **L'ensemble d'apprentissage** : utilisé pour entraîner le modèle. Il représente la majorité des données.
* **L'ensemble de test** : utilisé pour tester la performance du modèle. Il permet d'évaluer comment le modèle performe sur des données qu'il n'a jamais vues.

Cette séparation est indispensable pour évaluer la capacité du modèle à généraliser à de nouvelles données, plutôt que de simplement mémoriser les données d'entraînement.

### Entrainement d’un modèle

L'entraînement d'un modèle consiste à utiliser l'ensemble d'apprentissage pour ajuster les paramètres du modèle de manière à minimiser une fonction de perte ou à maximiser une mesure de performance. Cette étape utilise divers algorithmes et techniques selon le type de problème (régression, classification, etc.). Dans notre cas, nous utilisons des algorithmes de classifications.

### Prédiction sur l’échantillon de test

Une fois le modèle entraîné, il est utilisé pour faire des prédictions sur l'ensemble de test. Cette étape teste la capacité du modèle à appliquer ce qu'il a appris à des données non vues auparavant, fournissant une estimation de sa performance en conditions réelles.

### Evaluation des performances

L'évaluation des performances se fait en comparant les prédictions du modèle aux valeurs réelles de l'ensemble de test en utilisant des métriques appropriées (accuracy

Nb , rappel, précision, score F1, etc.). Cette évaluation permet de juger de l'efficacité du modèle à répondre au problème posé.

#### Métrique utilisé pour les résultats

##### Exactitude (Accuracy)

L'exactitude est la mesure la plus intuitive pour évaluer la performance d'un modèle. Elle est définie comme le nombre de prédictions correctes faites par le modèle divisé par le nombre total de prédictions. En formule, cela se traduit par:

##### Rappel (Recall)

Le rappel, quant à lui, est une mesure qui indique la capacité du modèle à identifier correctement tous les cas positifs réels. Autrement dit, parmi tous les cas réellement positifs, combien le modèle a-t-il réussi à identifier ? Le rappel est particulièrement important dans les situations où il est crucial de détecter tous les cas positifs, même au risque de faire quelques erreurs positives (faux positifs). La formule pour le calculer est la suivante:

### Recherche d’amélioration du modèle

Cette dernière étape consiste à chercher des moyens d'améliorer la performance du modèle. Cela peut inclure :

* **Le changement de modèle** : essayer différents algorithmes qui pourraient être mieux adaptés au problème. Application des algorithmes à des données dont on connaît les résultats afin de découvrir des modèles entre leurs caractéristiques et la variable cible.
* **La transformation de nouveau des données** : cela peut impliquer l'ajout de nouvelles variables, une transformation différente des variables existantes, ou la sélection de caractéristiques pour éliminer le bruit ou les informations redondantes.
* **L'ajustement des hyperparamètres** : optimiser les paramètres de l'algorithme pour améliorer les performances.

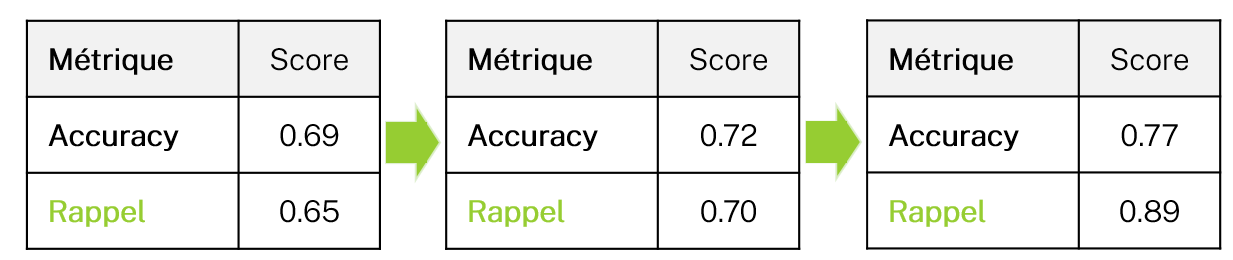
Cette recherche d'amélioration est souvent un processus itératif, nécessitant plusieurs cycles à travers ces étapes pour affiner le modèle et les données jusqu'à atteindre une performance satisfaisante.

## Évaluation et déploiement

Le modèle renvoie des résultats sous la forme de scores de probabilité qui vont permettre de l’évaluer, le comparer et le valider.

# Evolution des résultats du modèle

Après différents essais, nous avons réduit le nombre de variables, passant de 50 à 30.



* **Accuracy :** ratio de prédiction exacte
* **Rappel :** Probabilité de prédire une vaine comme vaine

Nos premiers modèles se sont révélés infructueux, ne répondant pas aux attentes de précision et de performance souhaitées. Cependant, ces tentatives ont joué un rôle crucial en nous permettant de cibler des critères explicatifs pertinents pour la construction d'un modèle prédictif plus efficace. En se concentrant sur un champ d'intervention plus restreint, nous avons pu affiner notre approche et identifier les variables les plus influentes

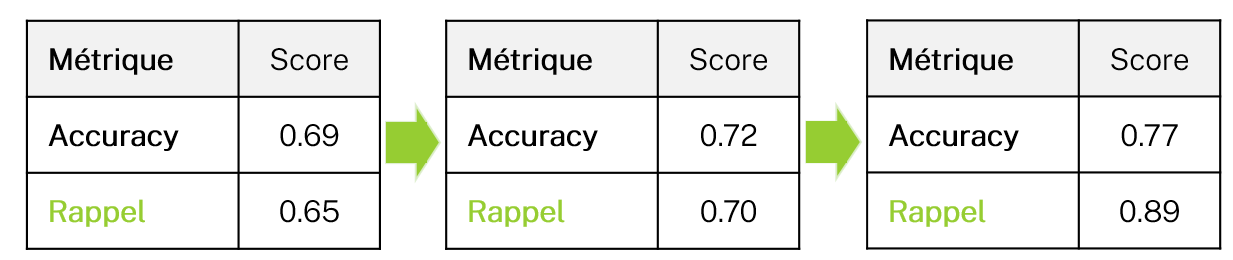
# Segmentation des données du modèle

Dans le contexte d'un modèle de prédiction, un **Segment** peut être compris comme un groupe spécifique au sein d'un ensemble de données, regroupant des éléments qui partagent certaines caractéristiques ou comportements similaires.

L'idée est de diviser l'ensemble de données en sous-ensembles plus petits mais homogènes pour mieux comprendre et prédire le comportement ou les caractéristiques des éléments dans chaque segment. Cela aide à créer des modèles de prédiction plus précis, car les modèles peuvent être ajustés pour refléter les spécificités de chaque segment, plutôt que d'essayer d'appliquer une règle générale à un ensemble très varié.

Par exemple, dans notre cas, suite à de nombreux essais infructueux, nous avons décidé de segmenter les données pour avoir de meilleurs résultats de prédiction. Nous avons donc identifié un segment où nous regroupons les interventions où la présence du client n’est pas obligatoire et où le compteur est inaccessible. Ce segment représente environ 20-25% des interventions totales.

Grâce à cela, nous avons significativement amélioré les résultats du modèle.



* **Accuracy :** ratio de prédiction exacte
* **Rappel :** Probabilité de prédire une vaine comme vaine

# Résultat par segments

## Segment (présence client pas obligatoire / compteur est inaccessible)

### Matrice de confusion

|  |  |  |
| --- | --- | --- |
| Etat de réalisation | Prédits Vains | Prédits Réalisés |
| Interventions Réalisées | 40631 | 99810 |
| Interventions Vaines | 61714 | 7373 |

### Résultats de classification

|  |  |  |  |
| --- | --- | --- | --- |
| Classe | Accuracy | Rappel | Précision |
| Vaines | 0,77 | 0,89 | 0,60 |
| Réalisées | 0,71 | 0,93 |

Pour la classe Vaines :

* **Rappel :** Probabilité de prédire une vaine comme vaine.
* **Précision :** Probabilité de ne pas prédire une réalisée comme une vaines.

Pour la classe Réalisées :

* **Rappel :** Probabilité de prédire une réalisée comme réalisée.
* **Précision :** Probabilité de ne pas prédire une vaine comme une réalisée.

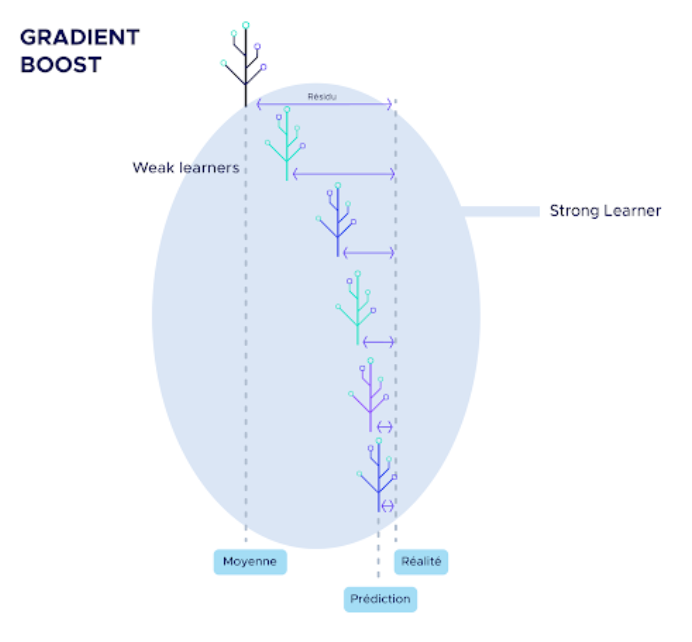
Ces résultats montrent globalement que sur ce segment notre modèle est bon pour prédire les interventions vaines mais pas les interventions réalisées. Cela s’explique par le fait que le modèle ratisse large pour bien récupérer les interventions vaines au détriment des interventions réalisées

# Modèle

**XGBoost** signifie **eXtreme Gradient Boosting**, c’est un algorithme de **Gradient Boosting**.

Le **Gradient Boosting** est un algorithme particulier de **Boosting**.

Le Boosting consiste à assembler plusieurs **Weak Leaners** pour en faire un **Strong Leaner,** c’est-à-dire assembler plusieurs algorithmes ayant une performance peu élevée pour en créer un beaucoup plus efficace et satisfaisant. L’assemblage de Weak Leaners en Strong Leaners se fait par l’appel successif de ceux-ci pour estimer une variable d’intérêt. Cela nous aide à avoir un résultat beaucoup plus pertinent.



### Modèles de base faibles

XGBoost commence par initialiser le modèle avec une prédiction constante, qui peut être la moyenne des étiquettes de formation pour les tâches de régression ou le logarithme des chances pour les tâches de classification. Ensuite, il ajoute itérativement des arbres de décision, chacun étant conçu pour corriger les erreurs laissées par l'ensemble des arbres déjà construits.

### Optimisation par Gradient Boosting

La méthode de "boosting" améliore la prédiction en ajoutant séquentiellement des arbres, où chaque nouvel arbre corrige les erreurs du modèle combiné précédemment formé. Dans le cas du gradient boosting, cette correction est faite en ajustant les nouveaux arbres aux gradients négatifs de la fonction de perte utilisée (d'où le nom). Cela signifie que chaque nouvel arbre est construit pour prédire le gradient de la perte par rapport aux prédictions actuelles, agissant comme une descente de gradient dans l'espace de la fonction de perte.

### Régularisation

XGBoost introduit une régularisation dans le processus de boosting de gradient, ce qui aide à prévenir le surajustement. Cela est réalisé en ajoutant un terme de pénalité sur la complexité des arbres (par exemple, le nombre total de feuilles ou la somme des poids des nœuds) dans la fonction de coût. Cette régularisation est l'une des clés de la performance supérieure de XGBoost par rapport à d'autres algorithmes de boosting de gradient.

### Parallélisation et Optimisation des Performances

XGBoost utilise des techniques avancées pour accélérer l'entraînement et améliorer l'efficacité, y compris la parallélisation de la construction des arbres, l'optimisation du stockage et du calcul des gradients, et une stratégie de partitionnement des données efficace appelée "Quantile Sketch" pour gérer de grands volumes de données.

### Gestion des valeurs manquantes

XGBoost peut automatiquement gérer les valeurs manquantes. Lors de la construction des arbres, il trouve la direction la plus bénéfique pour les données manquantes, soit en les dirigeant vers le branchement gauche, soit vers le droit, en fonction de ce qui réduit le plus la fonction de perte lors de la validation croisée.

# Technique

Cette partie vise à présenter le projet Data Science avec un approche plus technique en abordant le code.

## Préparation de la donnée

Voici une liste de toutes les étapes de la préparation de la donnée :

**1. Récupération des données** : Les données sont extraites de la source ZEP32RP3 en utilisant quatre requêtes distinctes :

* La première requête cible la DataWareHouse (DWH) pour collecter la majorité des informations nécessaires au modèle.
* Les trois requêtes suivantes visent l'Operational Data Store (ODS) afin de récupérer des informations complémentaires non disponibles dans le DWH.
* Ces quatre requêtes sont ensuite converties en DataFrames.
* Un processus de fusion (merge) est réalisé pour assembler toutes les informations collectées en un unique Data Frame.

**2. Filtrage des données** : Le jeu de données est restreint aux informations des deux premières années pour cibler l'analyse.

**3. Intégration des données INSEE** : Les données provenant d'un fichier Excel de l'INSEE sont ajoutées au jeu de données existant par un merge, enrichissant ainsi le DataSet initial.

**4. Normalisation des données** : Certaines valeurs du jeu de données sont remplacées en utilisant un dictionnaire spécifique pour améliorer la cohérence des données.

**5. Incorporation des données techniques** : Des informations techniques stockées dans un fichier pickle sont intégrées au DataSet principal grâce à un autre processus de merge.

**6. Typage des données** : Les types de données sont ajustés pour s'assurer qu'ils correspondent aux besoins du modèle.

**7. Traitement des valeurs manquantes** : Les valeurs indiquées comme « à définir » sont remplacées par -1 pour maintenir l'intégrité numérique des données. Etant donnée le nombre très faible de valeurs manquante, nous supprimons les interventions concernées.

**8. Réduction des modalités** :

* Les sous-activités moins représentées sont regroupées sous la catégorie « Autres ».
* De même, les catégories comptant moins de 1000 occurrences sont également regroupées sous « Autres ».
* Pour les interventions à plusieurs techniciens, seul le NNI du technicien le plus “compétent” a été retenu
* Pour les interventions ayant des doublons à cause de la variable “equ\_type”, les interventions n’ayant pas d’infos ont été supprimées

**9. Filtrage selon des critères spécifiques** : Le DataSet est filtré pour exclure les cas où la présence client n'est pas obligatoire et le compteur n'est pas accessible, en utilisant le critère suivant :

df1=df1.loc[(df1['int\_est\_presence\_client\_obligatoire']==False)&(df1['igk\_est\_compteur\_accessible'] == False)]

**10. Ajout de nouvelles caractéristiques** : Une nouvelle colonne, `nb\_vain\_prev`, est ajoutée pour indiquer le nombre d'interventions infructueuses précédentes, par référence d'affaire et code\_uo.

**11. Organisation des données** : Les données sont triées par date pour faciliter les analyses temporelles.

**12. Épuration du dataset** : Certaines colonnes jugées inutiles sont supprimées pour alléger le jeu de données.

**13. Binarisation** : Les modalités sont remplacées par des valeurs numériques 0 ou 1, ce qui est souvent préféré par les modèles de machine learning pour faciliter leur interprétation des catégories.

**14. Ciblage des interventions** : Seules les interventions qualifiées de vaines ou réalisées sont conservées dans l'analyse, et sont transformées en valeurs numériques 0 ou 1 pour simplifier l'analyse.

## Création et Entrainement du modèle

### Séparation du modèle

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=34)

* **X\_train** et **y\_train** sont les sous-ensembles de vos caractéristiques (**X**) et de votre cible (**y**) utilisés pour l'entraînement du modèle.
* **X\_test** et **y\_test** sont les sous-ensembles de vos caractéristiques (X) et de votre cible (y) qui seront utilisés pour tester la performance du modèle.
* **train\_test\_split(X, y, test\_size=0.2, random\_state=34)** divise l'ensemble de données en ensembles d'entraînement et de test. X est votre tableau de caractéristiques, et y est votre variable cible.
  + **test\_size=0.2** indique que 20% des données seront mises de côté pour l'ensemble de test, et les 80% restants seront utilisés pour l'entraînement.
  + **random\_state=34** est utilisé pour initialiser le générateur de nombres aléatoires interne, qui décidera de la division des données en indices d'entraînement et de test. Fournir un `random\_state` spécifique assure que les divisions que vous générez sont reproductibles.

### Initialisation du modèle XGBClassifier

monit\_model = XGBClassifier()

### Entrainement du modèle

monit\_model.fit(X\_train, y\_train)

XGBoost (eXtreme Gradient Boosting) est une bibliothèque populaire et efficace pour le machine learning qui implémente l'algorithme de boosting de gradient. Le boosting de gradient est une technique d'ensemble qui construit des modèles de manière séquentielle, où chaque nouveau modèle corrige les erreurs faites par les modèles précédents. XGBoost est particulièrement réputé pour sa vitesse et sa performance. Voici une explication technique et mathématique de son fonctionnement :

#### Objectif

XGBoost vise à résoudre le problème d'apprentissage supervisé en minimisant une fonction Objectif qui est la somme d'une fonction de perte ***L*** et d'une fonction de régularisation ***Ω***, donnée par :

où est la valeur réelle, est la prédiction du modèle, est le ***k***-ième arbre de décision, et ***K*** est le nombre total d'arbres.

#### Fonction de perte

La fonction de perte mesure à quel point les prédictions du modèle diffèrent des valeurs réelles. XGBoost est flexible et permet d'utiliser différentes fonctions de perte adaptées à divers problèmes de régression, de classification et d'autres tâches.

#### Fonction de régularisation

La fonction de régularisation est utilisée pour contrôler la complexité du modèle, ce qui aide à prévenir le surapprentissage. Pour un arbre , elle est définie comme :

où T est le nombre de feuilles de l'arbre, est le score de sortie de la j-ième feuille, est le paramètre qui contrôle le coût de l'ajout de nouvelles feuilles, et est le paramètre de régularisation pour les scores de sortie.

#### Optimisation

XGBoost utilise un algorithme d'optimisation basé sur le gradient pour minimiser la fonction Objectif. À chaque étape, il construit un arbre qui améliore le modèle en réduisant la fonction Objectif. Pour un arbre donné, la prédiction pour l'instance ***i*** est mise à jour comme suit :

où est la prédiction de l'étape précédente, est la prédiction de l'arbre actuel pour l'instance , et est le taux d'apprentissage.

#### Approche pour le calcul de gain

Lors de la construction de chaque arbre, XGBoost utilise une approche de gain basée sur le gradient pour décider où faire des divisions. Le gain d'une division potentielle est calculé comme suit :

Où et sont les ensembles d'indices des instances de données dans les fils gauche et droit après la division, respectivement, est le gradient de la fonction de perte par rapport à la prédiction pour l'instance , est le hessien de la fonction de perte, et est l'ensemble d'indices de toutes les instances de données dans le nœud avant la division.

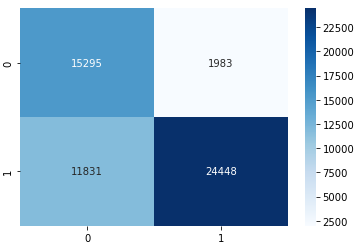
## Test et Vérification

### Matrice de confusion

cf\_matrix = confusion\_matrix(y\_test, y\_pred)

sns.heatmap(cf\_matrix, annot=True, fmt='g', cmap='Blues')

* La matrice de confusion, ***confusion\_matrix*** compare les valeurs prédites ***y\_pred*** aux valeurs réelles ***y\_test*** pour évaluer la précision des prédictions du modèle.
* La visualisation avec ***sns.heatmap*** permet de facilement identifier les vrais positifs, vrais négatifs, faux positifs, et faux négatifs grâce à une représentation en gradation de couleurs, ici en bleu, rendant l'interprétation plus intuitive.

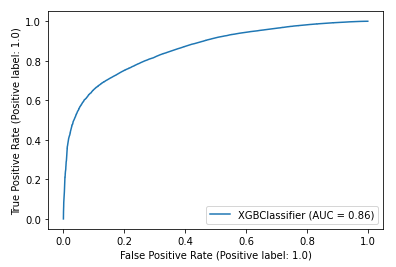


### Courbe ROC

plot\_roc\_curve(monit\_model, X\_test, y\_test)

plt.show()

* La fonction ***plot\_roc\_curve*** trace la courbe caractéristique de fonctionnement du récepteur (ROC) pour le modèle, utilisant les données de test ***X\_test*** et ***y\_test***. Cette courbe évalue la capacité du modèle à distinguer entre les classes.



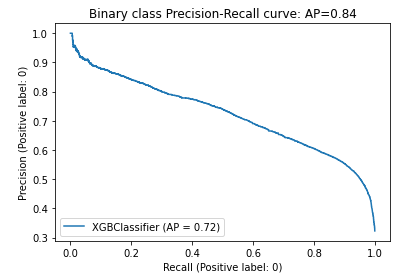
### Précision moyenne et Courbe de Rappel-Précision

average\_precision = average\_precision\_score(y\_test, y\_pred)

disp = plot\_precision\_recall\_curve(monit\_model, X\_test, y\_test, pos\_label=0)

disp.ax\_.set\_title('Binary class Precision-Recall curve: AP={0:0.2f}'.format(average\_precision))

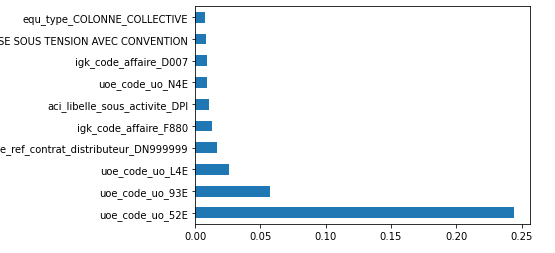
* ***average\_precision\_score*** calcule la précision moyenne à partir des prédictions
* ***plot\_precision\_recall\_curve*** trace la courbe de précision-rappel pour le modèle, une fonction utile pour comprendre la balance entre la précision (la proportion de vrais positifs parmi les prédictions positives) et le rappel (la capacité du modèle à trouver tous les cas positifs réels).



### Importance des caractéristiques.

pd.Series(monit\_model.feature\_importances\_, index=X\_train.columns).nlargest(10).plot(kind='barh')

* Cette ligne de code utilise ***feature\_importances*** pour identifier et visualiser les dix caractéristiques les plus importantes du modèle. Cela permet de comprendre quelles variables contribuent le plus à la prédiction du modèle.



# Problème rencontrés et solutions

Le déséquilibre des classes Vaines et Réalisées dans le jeu de données a pu influencer les performances du modèle. Pour pallier ce problème, nous avons appliqué une technique de sous-échantillonnage afin d'équilibrer les classes.

La collecte manuelle de l'état de réalisation par les techniciens introduit une incertitude sur la valeur de cette variable, difficilement mesurable avec précision. Selon les estimations du métier, cette erreur pourrait affecter environ 10 % des interventions, engendrant un biais significatif. Actuellement, il n'existe pas de solution pour éliminer complètement ce problème. Toutefois, une approche envisageable consiste à examiner les données du compteur Linky après chaque intervention pour vérifier toute modification effectuée. Cela permettrait de confirmer l'intervention du technicien et de s'assurer que l'intervention n’ai pas été vaines.

# Autres cas d’usage

* **Amélioration de la planification des chantiers (Système de recommandation)**
* **Prédiction du nombre de chantiers et d’interventions dans le temps**
  + Prédire le nombre de chantiers/interventions pour anticiper les demandes d’interventions à venir et avoir une estimation de la charge de travail dans le temps
* **Estimation de la charge de préparation d’un chantier**
  + Prédire et estimer de la charge de travail d’un chantier
  + Segmentation des chantiers pour estimer le temps et les ressources en fonction des autres chantiers
* **Prédiction des chantiers par zones géographiques**
* **Analyser le taux de réalisation par groupe de technicien**
  + Etudier et analyser les différents profils de techniciens => Clustering
  + Comprendre le taux de réalisation d’un groupe
* **Prédiction de la disponibilité ou saturation des ressources**
* **Détection des anomalies dans le futur** 
  + Interventions non réalisées à répétition (tentative)
* **Détection des anomalies dans les data**
  + Incohérence des données

## Prochaines étapes

* Amélioration continue du rapport
* Identification d’un nouveau segment
* Implémentation de nouvelles données dans le rapport