

Rapport - Machine learning interprétable

Morgane Besnier - Marjolaine Claret - Yovana Dentika - Uthamy Thanabalasingam

Tuteurs : Giovanni Chierchia - Thibaud Vienne



Introduction

Dans un contexte de massification des données (90% des données collectées ont moins de deux ans) que l'on qualifie d'ère du Big Data, les entreprises illustrent la capacité atteinte d'accumulation de plus de données et l'envie de tirer profit au maximum des informations extraites. Des algorithmes comme les réseaux de neurones ou les forêts aléatoires sont largement déployés à cet effet dans un nombre croissants de secteurs. Ces modèles permettent généralement un gain non négligeable en terme d'efficacité et de performance. Mais ce gain se fait au détriment de la transparence et de l'interprétabilité de ces algorithmes qui sont qualifiés de "boîtes noires". Et pourtant, la lisibilité et la compréhension des modèles deviennent des nécessités dans de nombreuses situations. Une première raison est réglementaire, en effet de nombreuses institutions - telles que le Règlement Général sur la Protection des Données (RGPD) ou l'Autorité de Contrôle Prudentiel et de Résolution (ACPR)- exigent la justification des décisions prises et luttent contre une responsabilité reportée sur l'algorithme. L'explicabilité est aussi essentielle pour pouvoir expliquer les résultats fournis aux utilisateurs et aux intervenants et pour garantir une confiance dans les outils déployés. L'aspect éthique est une autre préoccupation incontournable et qui prend tout son sens dans un contexte de collecte massive de données et d'appel à plus de transparence. La confluence de ses raisons modifie la dynamique en place : les modèles ne sont plus seulement jugés sur leur performance mais aussi sur leur lisibilité et leur facilité de compréhension par un humain. Cette considération nouvelle a conduit au développement d'un secteur de recherche porté sur le machine learning interprétable (IML). L'interprétabilité est donc devenue un critère prépondérant pour les scientifiques, les utilisateurs et les décisionnaires car elle permet d'assurer la conformité avec les stratégies, les normes mais aussi les réglementations gouvernementales.

Rapport - Machine learning interprétable	1
Introduction	2
Notes sur le rendu	4
Tenants et aboutissants de l'interprétabilité	5
Définitions	5
Contexte	6
Enjeux actuels	7
Méthodes d'évaluation de l'interprétabilité	9
Etat de l'art des techniques existantes	12
Partial Dependence Plots	12
Individual Conditional Expectation	13
ALE: Accumulated local effect [19]	15
Counterfactual	17
LIME [7]	19
Scoped rules: Anchor [20]	21
SHAP [10]	23
Permutation Features Importance [12]	24
Classification d'image : races de chiens	27
Modèles	35
Boîte à outils avec Tkinter	36
Application de TOOL BOX à un problème de machine learning supervisé : prédiction de la qualité d'un vin par un algorithme Random forest classifier	38
Améliorations envisagées	41
Sources	42
Annexes	44

Notes sur le rendu

toolbox.py : le script générant la boîte à outils. Nécessite de placer les images *BG.png*, *document.png*, *logo.png* et *toolbox.png* dans le répertoire courant.

techniques_by_function.py : les fonctions par technique d'interprétabilité et l'enregistrement des graphiques générés en .png

Notebooks développés sur plusieurs modèles pour mettre en place les fonctions sur un jeu de données tabulaire :

Neural_network.ipynb

Random_forest_class.ipynb

Random_forest_reg.ipynb

XGBoost.ipynb

Linear_regression.ipynb

Notebook développés pour la classification d'image :

Xception_dog_breed_CNN.ipynb

Données mises à disposition pour tester les scripts :

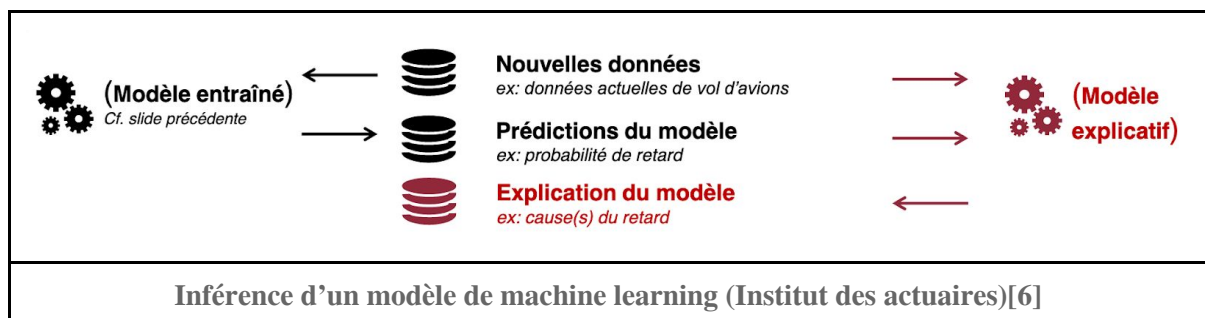
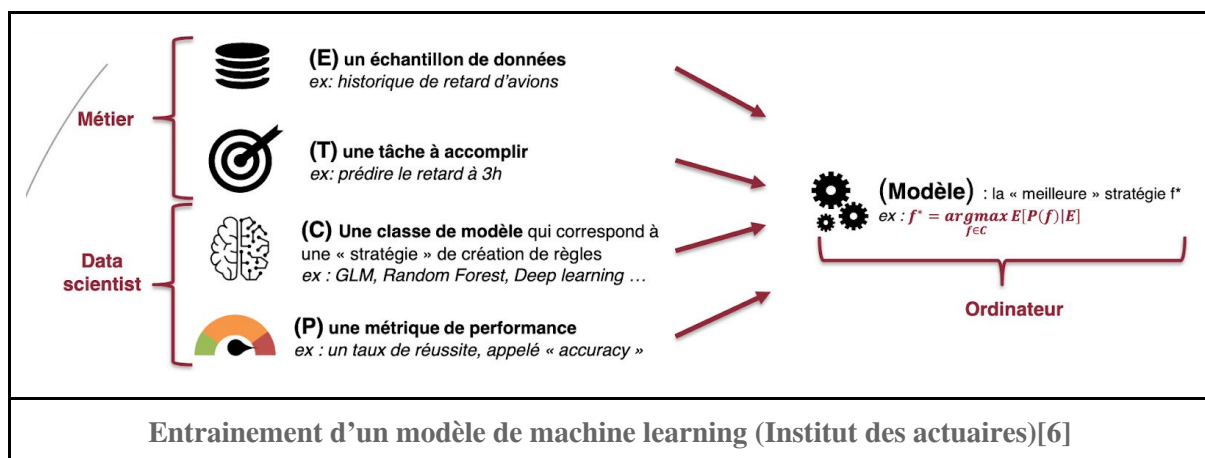
winequality-red.csv : dataset de données physico chimiques sur un vin portugais

Random_forest_model.pkl : modèle de type Random Forest

X_train.npy, *X_test.npy*, *Y_train.npy*, *Y_test.npy* : données d'entraînement et de test

Tenants et aboutissants de l'interprétabilité

Définitions



Les très bons résultats des modèles de machine learning ne les excluent pas de commettre des erreurs et notamment en présence d'un jeu de données biaisé le modèle reproduit et propage le biais. Et ce biais est difficile à détecter car les sources peuvent être nombreuses et le modèle fonctionne selon des règles plus ou moins complexes et non triviales pour un humain. Aussi, pour s'assurer que le modèle ne présente pas de biais il faut le comprendre.

L'interprétabilité explicite la façon dont la décision a été prise par l'algorithme (réponse au "comment ?" et réponse mathématique) et l'explicabilité satisfait la recherche des causes (réponse au "pourquoi ?").

La nuance est une question d'échelle : l'interprétation évalue globalement un processus de décision et l'explication offre des informations sur les variables déterminantes dans la décision. [5] L'explication doit en particulier permettre:

- à un développeur, de comprendre le fonctionnement de l'application
- à un utilisateur, de comprendre le périmètre d'utilisation et les hypothèses sous-jacentes
- à un expert, de statuer sur un audit

Contexte

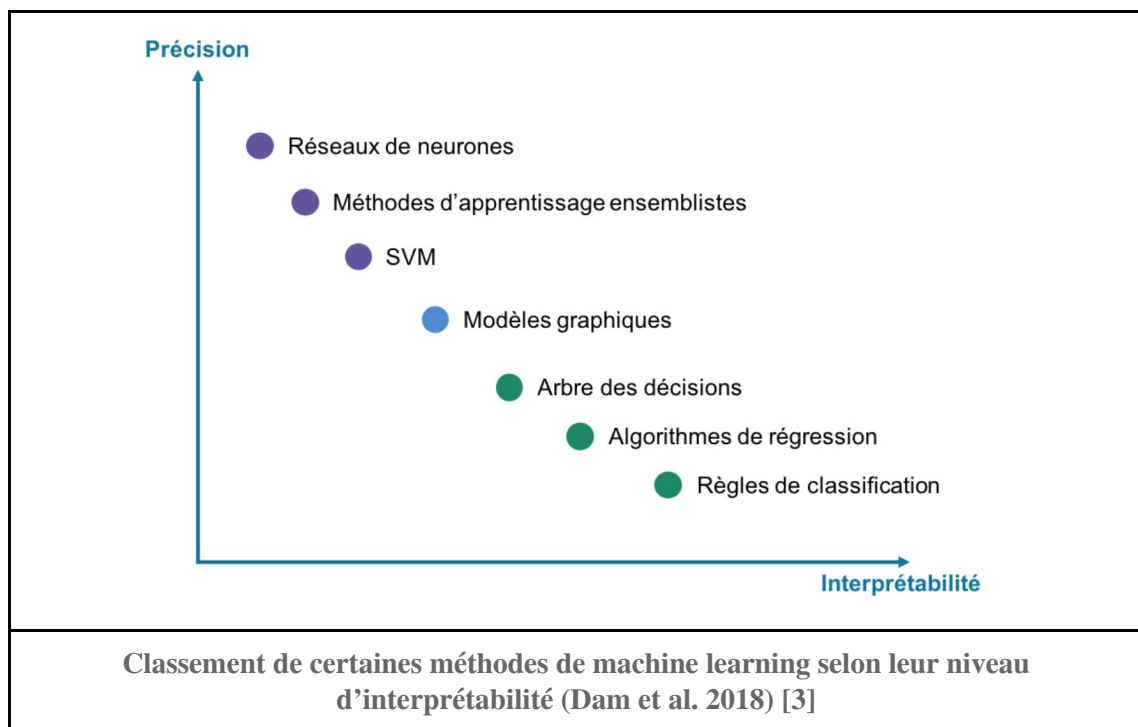
L'intelligibilité des algorithmes et en particulier de ceux de l'intelligence artificielle (IA) est devenue un critère prépondérant suite au rapport Villani de 2018 en France - feuille de route sur l'IA (recensement de problématiques et recommandations), commandé en 2017 par le gouvernement français - et à la mise en place du règlement général sur la protection des données (RGPD) en 2018 Europe. « L'IA ne peut pas être une nouvelle machine à exclure », résumé M. Villani. [1], et les analystes ajoutent que face au RGPD l'IA présente des risques systémique, d'opacité et d'atteinte à la liberté et à la sécurité. [2]

Les algorithmes de l'IA sont souvent qualifiés de "boîte noire", notamment par les professionnels dont des grands noms comme Stéphane Mallat, du fait de leur complexité, de leur rapide développement et de leurs performances presque inégalées. Le fonctionnement d'un algorithme d'IA peut être résumé en l'optimisation d'une certaine fonction de coût ce qui implique que ses performances puissent être mesurées assez trivialement par de correctes métriques, et ainsi une telle base de mesure a permis la construction d'algorithmes toujours plus sophistiqués au détriment de l'explicabilité. Et dans de nombreux cas d'algorithmes, la performance est corrélée à l'augmentation de la complexité ce qui a encouragé la tendance. La seconde observation qu'il est possible de tirer et que l'augmentation de la performance est corrélée à la baisse de l'interprétabilité.

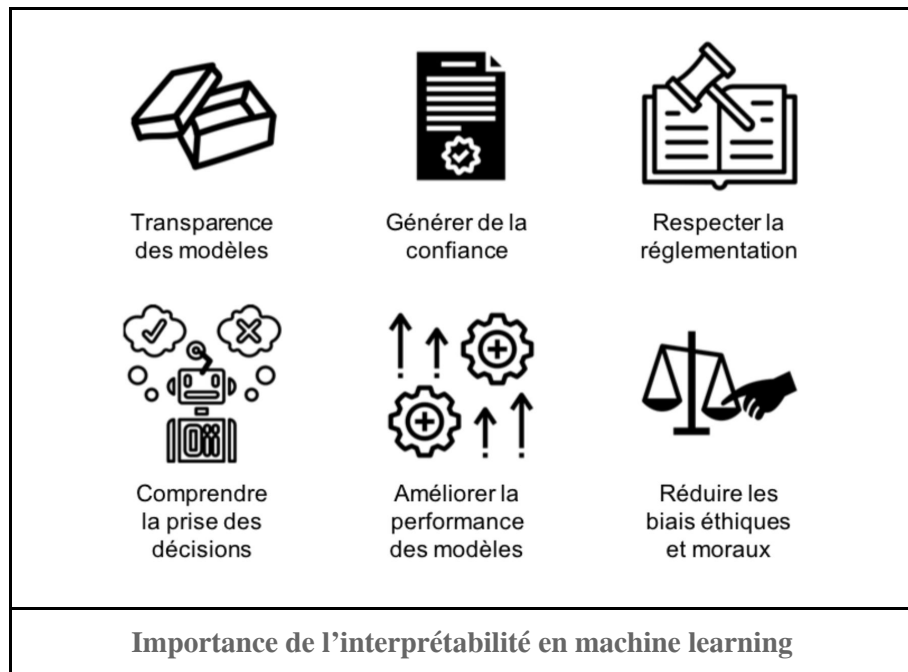
Le développement et l'engouement rapides autour de l'IA ont conduit à prioriser la performance des algorithmes, alors qu'à présent les questionnements et la prise de conscience font émerger une nouvelle dynamique dans laquelle l'explicabilité pourrait devenir le nouveau critère d'évaluation des modèles.

L'interprétabilité d'un algorithme de machine learning peut être déclinée en trois niveaux : [3]

- Faible interprétabilité : au mieux les algorithmes fournissent des informations sur l'importance des variables.
- Interprétabilité moyenne : inclut des algorithmes plus avancés, ils sont contraints à l'approximation de fonctions monotones non linéaires.
- Haute interprétabilité : ce niveau inclut les algorithmes de régression, les arbres de décision et les règles de classification traditionnelle. Ils se rapprochent des fonctions monotones linéaires.



Enjeux actuels



A l'heure actuelle trouver un compromis entre précision et explicabilité devient de plus en plus nécessaire. En effet, la compréhension du modèle se fait vitale dans de nombreux cas.

Tout d'abord, la prise en compte d'un apprentissage automatique centré vers l'utilisateur grandit et pourrait également accélérer le changement et l'appropriation de cette priorité. Un modèle transparent et explicable pourrait en effet rendre plus évidente son acceptation et son adaptation pour effectuer les tâches professionnels ou quotidiennes de façon plus pérenne et réaliste.

Un second aspect est la nécessité de la complaisance à des normes et des réglementations. Car tous les secteurs ne sont pas égaux face à l'urgence et au besoin d'explicabilité : la banque, la justice, la santé sont les plus demandeurs car pour ces domaines en particulier il s'applique des exigences légales et éthiques très strictes qui limitent de ce fait l'utilisation de "boîtes noires". L'article 22-1 du RGPD l'explique : « La personne concernée a le droit de ne pas faire l'objet d'une décision fondée exclusivement sur un traitement automatisé, y compris le profilage, produisant des effets juridiques la concernant ou l'affectant de manière significative de façon similaire. ». Dans ces cas, l'Homme est acteur du processus de décision et doit de ce fait interagir avec les résultats des algorithmes de l'IA, ce qui implique la nécessité effective d'expliquer et de comprendre les raisons de la décision retournée.

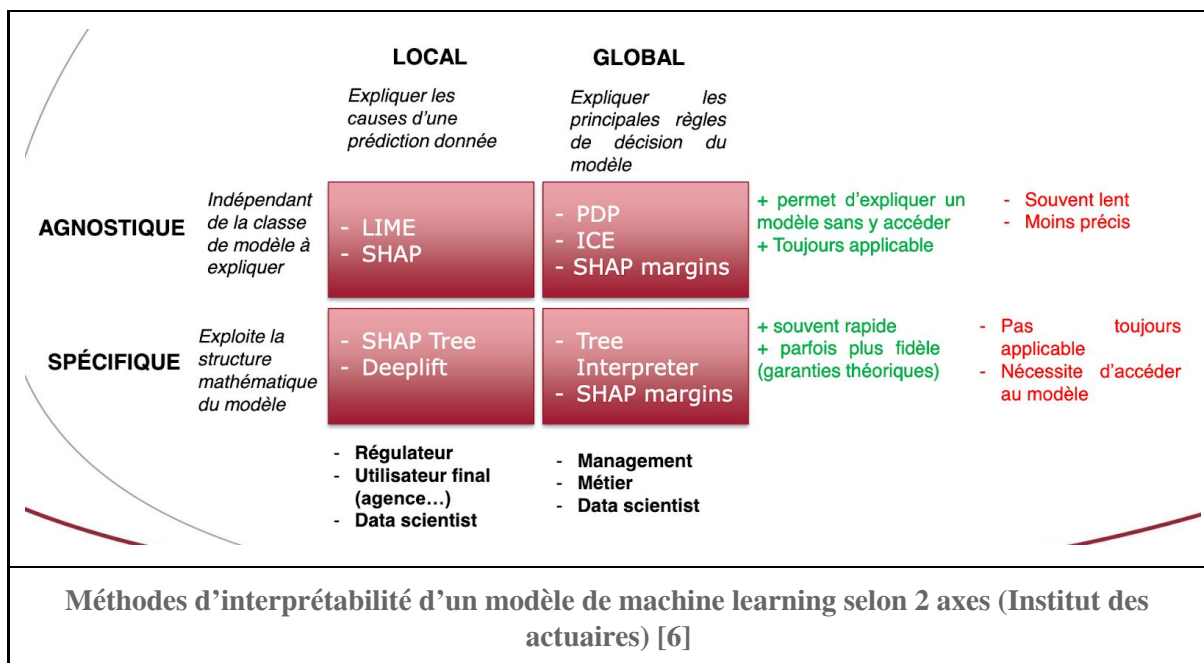
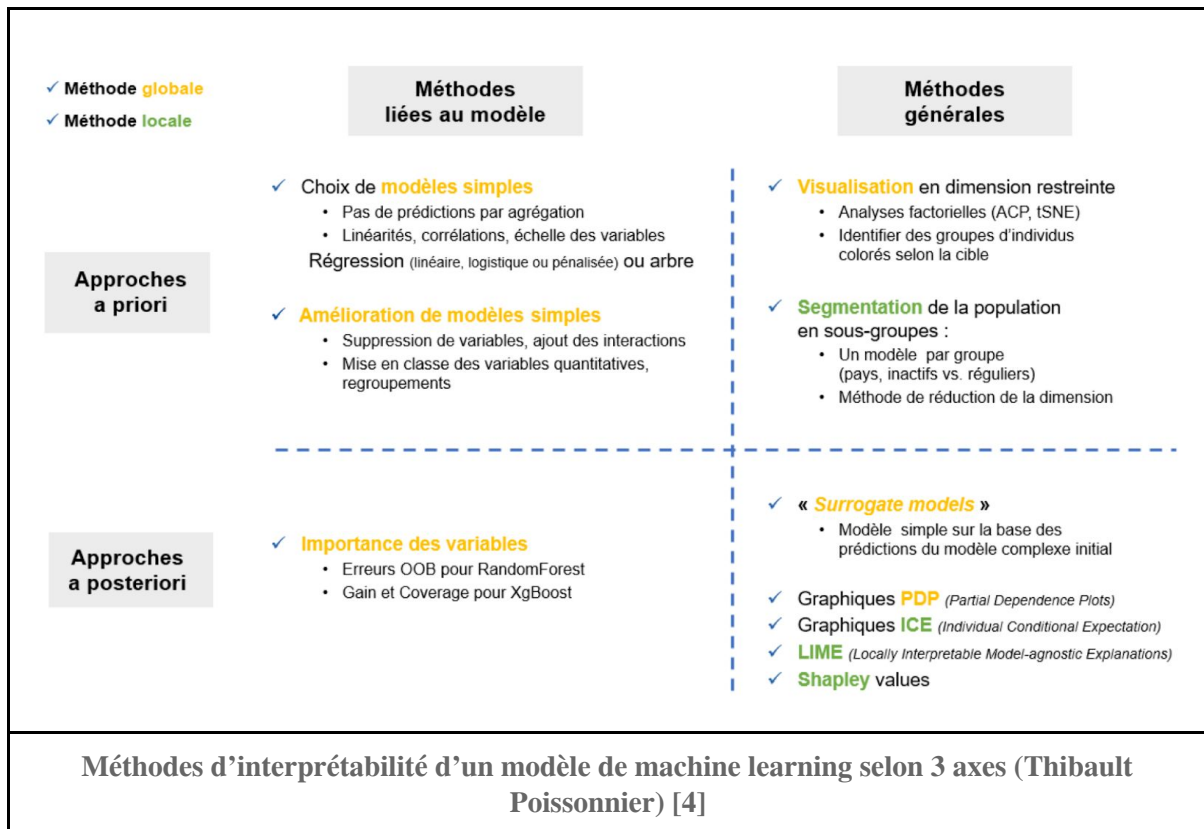
Un dernier aspect à ne pas négliger est la réticence actuelle et persistantes de nombreux acteurs ou secteurs à déployer des outils basés ou utilisant le machine learning, qui porte la recherche sur le

développement de l'explicabilité tout en maîtrisant la performance - et qui vise à générer une confiance dans les outils.

Méthodes d'évaluation de l'interprétabilité

Différentes catégories de méthodes existent pour déterminer le niveau d'interprétabilité : [3]

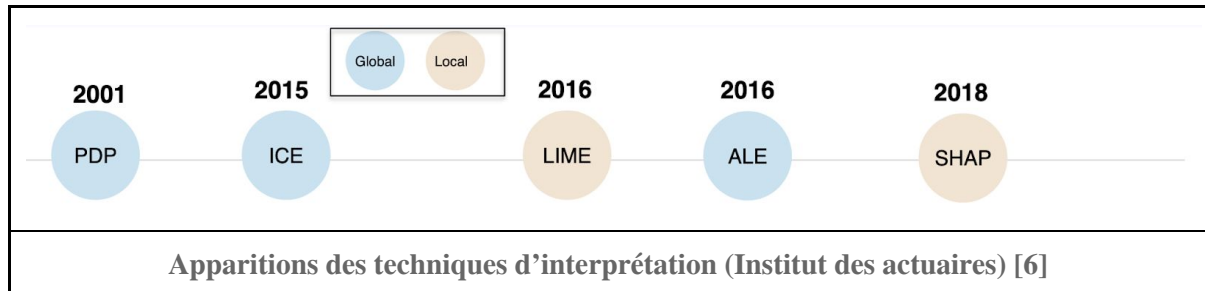
- *Intrinsèque* ou *Post-hoc* : l'interprétabilité intrinsèque implique que le modèle d'apprentissage automatique choisi est un modèle facile à expliquer tel qu'un arbre de décision. Tandis que l'interprétabilité post-hoc implique l'utilisation d'un modèle dit boîte noire pour l'entraînement et l'application des méthodes d'interprétabilité par la suite pour expliquer les résultats tels que la méthode « feature importance » pour Random Forest.
- *Model-Specific* (ou lié au modèle) ou *Model-agnostic* (ou général) : l'approche model-agnostic regroupe les méthodes qui ne dépendent pas du type du modèle, alors que l'approche model-specific est dépendante.
- *Local* ou *Global* : l'approche globale donne une interprétation pour tous les individus (l'ensemble des données) alors que l'approche locale fournit une interprétation pour quelques individus (voisinage restreint de données). Ces deux critères définissent donc le périmètre de l'interprétabilité du modèle. Le choix du périmètre d'interprétabilité d'un modèle peut influencer la décision sur le choix des algorithmes qui peuvent être implémentés. Par exemple en choisissant une explicabilité globale, un modèle simple et facile à expliquer peut être amélioré en le complexifiant petit à petit et ce qui permettra d'obtenir une explicabilité globale. Tandis que si on choisit une explicabilité locale, alors avec un modèle complexe il est possible de le rendre plus interprétable en réduisant l'analyse à un voisinage.
- *A priori* ou *A posteriori* : l'approche à priori est employée en amont de la création du modèle alors que l'approche a posteriori est employée après.



Il n'existe pas de consensus concernant la méthode d'évaluation de l'interprétabilité, néanmoins dans la littérature trois niveaux sont suggérés : [3]

- *Évaluation au niveau application* (tâche réelle) : L'évaluation est faite par les utilisateurs finaux ou les experts métiers. Cela nécessite la mise en place d'un processus objectif qui permet de mesurer la qualité de l'évaluation. Un bon point de départ serait de comparer l'interprétation du résultat d'un modèle avec l'interprétation faite par un expert.
- *Évaluation au niveau humain* (tâche simple) : L'évaluation est faite par des personnes non-expertes dans le domaine. Par exemple, on peut montrer plusieurs explications pour une prédiction, et les personnes peuvent choisir la meilleure. Une agrégation des évaluations sera nécessaire pour évaluer le résultat du modèle.
- *Évaluation au niveau fonctionnelle* (tâche proxy) : L'évaluation est faite par des machines. Ce type d'évaluation est possible si le fonctionnement des modèles utilisés est facile à comprendre ou à expliquer tels que les modèles de régression ou les arbres de décision.

Etat de l'art des techniques existantes



Les graphiques de dépendance partielle (DP) et d'attentes conditionnelles individuelles (ICE) illustrent les relations entre une ou plusieurs variables d'entrée et les prédictions d'un modèle de boîte noire.

Partial Dependence Plots

Le Partial Dependence Plots (PDP) montre l'effet marginal qu'une ou deux features ont sur le résultat prévu d'un modèle d'apprentissage machine. Un diagramme de dépendance partielle peut montrer si la relation entre la cible et une caractéristique est linéaire, monotone ou plus complexe. Le PDP permet donc d'identifier le sens (positif ou négatif) et la forme (linéaire, non linéaire, par paliers) de la relation.

Mathématiquement le PDP se base sur :

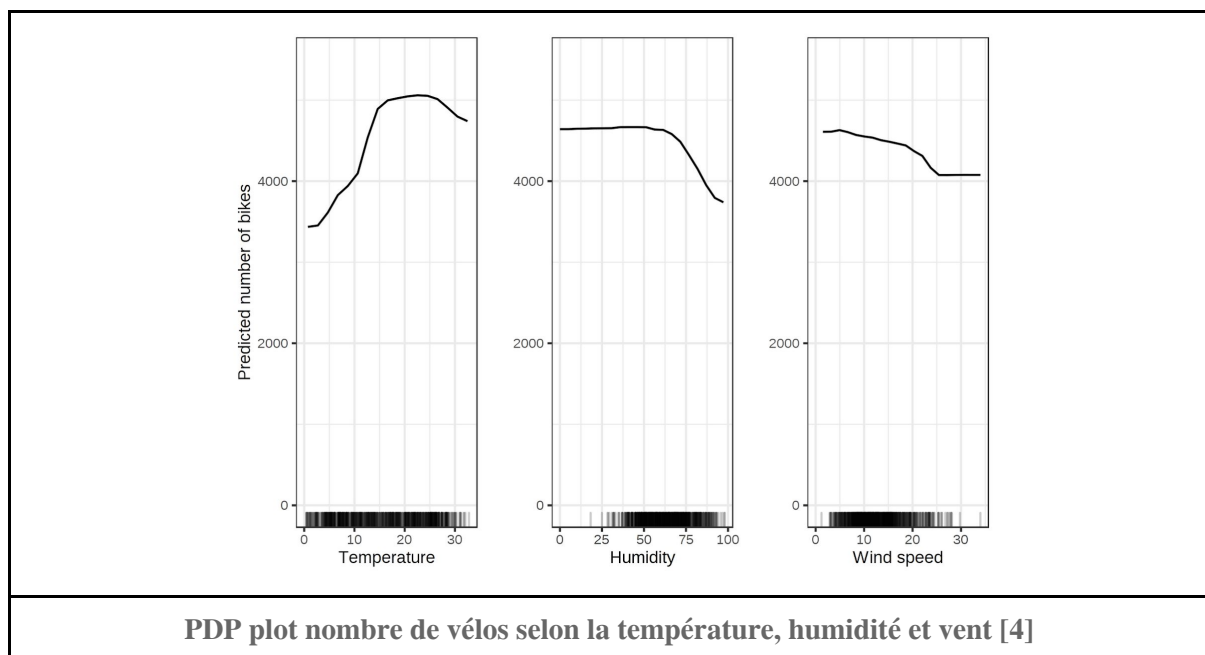
- Une variable S fixée (en pratique, l'ensemble des caractéristiques S ne contient généralement qu'une seule caractéristique et deux au maximum)
- L'ensemble des autres variables C

⇒ calcul du prédicteur (fonction partielle) à S fixé en faisant varier C .

La fonction partielle indique, pour une ou plusieurs valeurs données des caractéristiques S , quel est l'effet marginal moyen sur la prédiction.

En général le PDP est utilisé pour réaliser de la classification (production de probabilités), le diagramme de dépendance partielle affiche ainsi la probabilité pour une classe donnée en fonction de différentes valeurs pour les caractéristiques dans S. Graphiquement une classe est représentée par une ligne ou un diagramme. Et un diagramme de dépendance partielle (DP) illustre la relation fonctionnelle entre un petit nombre de variables d'entrée et les prédictions.[4]

Si le PDP présente les avantages d'être intuitif et facile à comprendre, il requiert la satisfaction d'hypothèses pour être exploitable : le PDP suppose que les caractéristiques ne sont pas corrélées - alors il représente parfaitement la façon dont la caractéristique influence la prédiction en moyenne [6]. Un second inconvénient est que les effets croisés ne sont pas visibles, la relation est en fait causale pour le modèle, car le résultat est explicitement modélisé en fonction des caractéristiques, mais pas nécessairement pour le monde réel. [4]



Individual Conditional Expectation

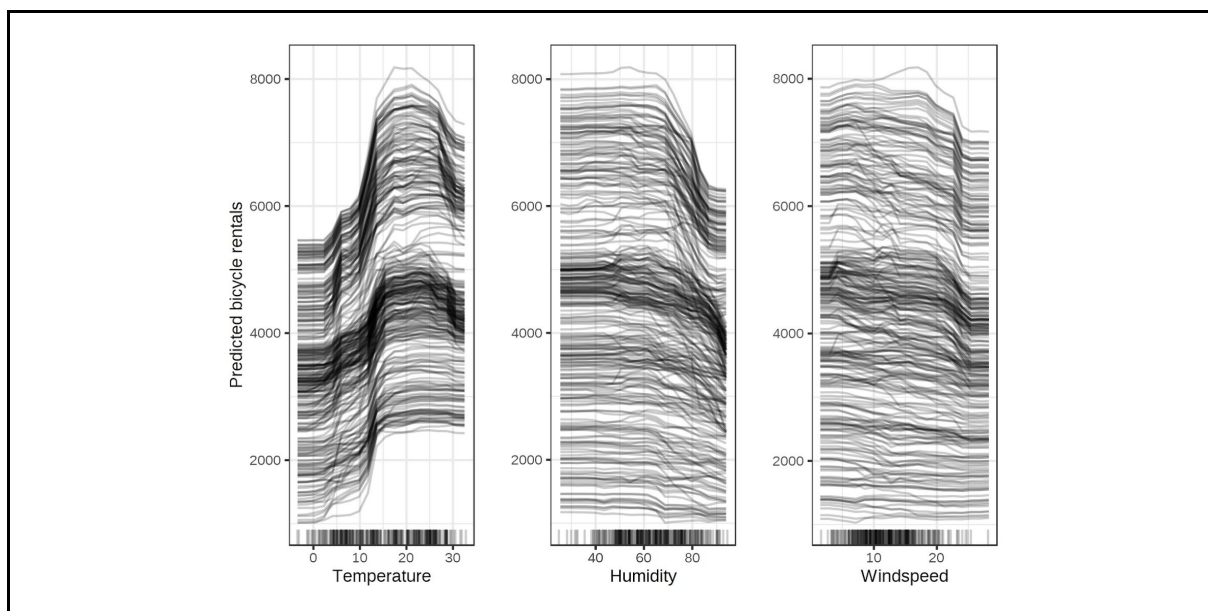
Le diagramme de dépendance partielle pour l'effet moyen d'une caractéristique est une méthode globale car il ne se concentre pas sur des cas spécifiques, mais sur une moyenne globale. L'équivalent d'un PDP pour des cas de données individuels est appelé ICE. Les tracés ICE peuvent être déployés

pour explorer les différences individuelles et identifier les sous-groupes et les interactions entre les entrées du modèle.

Comme décrit dans les inconvénient qui incombent au PDP, les tracés de DP peuvent masquer une relation hétérogène créée par les interactions (effet croisé) et sont pertinents dans le cas de faibles interactions entre les caractéristiques intervenantes. Dans le cas d'interactions, le tracé de l'ICE est plus pertinent car le résultat obtenu est plus précis. [4] Un tracé ICE visualise la dépendance de la prédiction à une caractéristique pour chaque instance séparément, ce qui donne graphiquement une ligne par instance et alors un DP est la moyenne des lignes d'un tracé ICE.

Les courbes ICE présentent l'avantage d'être encore plus intuitives à comprendre que les tracés DP. Une ligne représente les prédictions pour un cas donné si nous faisons varier la caractéristique qui nous intéresse. Et contrairement aux diagrammes de dépendance partielle, les courbes ICE peuvent mettre en évidence des relations hétérogènes.

Néanmoins les courbes ICE ne peuvent afficher qu'une seule caractéristique de manière significative, car deux caractéristiques nécessitent le dessin de plusieurs surfaces superposées ce qui serait illisible. Les courbes ICE souffrent également du même problème que les PDP : si l'élément d'intérêt est corrélé avec les autres éléments, certains points des lignes peuvent être des points de données non valides selon la distribution commune des éléments. Et pour finir, sur les courbes ICE, il n'est pas toujours facile de voir la moyenne. [3]



ICE plot du nombre de vélos selon les conditions de température, humidité et vent [4]

ALE: Accumulated local effect [19]

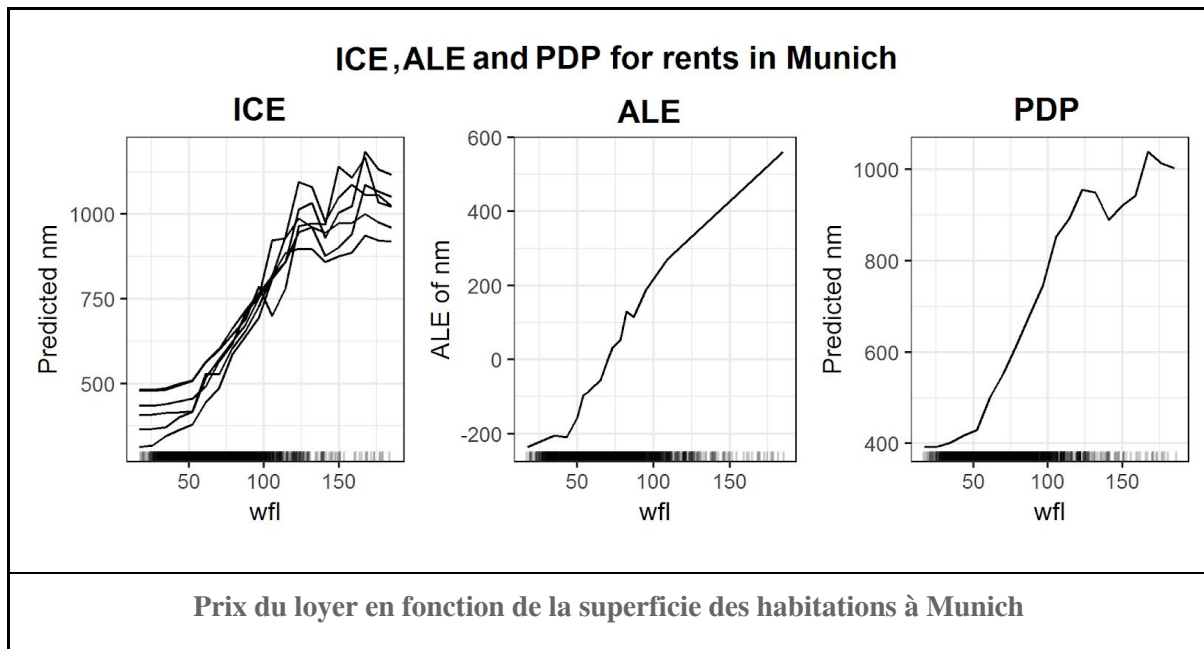
L'ALE est une technique applicable sur des régressions linéaire et des données de classification. Elle va déterminer l'effet que chaque paramètres de manière individuelle à sur la prédiction, sans l'influence des autres.

Par exemple, si nous analysons le nombre de joggeur qui traversent un sentier par jour. Nous incluons des facteurs tel que la température, la région rural ou urbaine, la présence de matériel sportif, les précipitations, la période de l'année... Dans ce cadre, l'ALE, nous montrerait alors comment un seul paramètre comme la période de l'année influe sur la prédiction du nombre de joggeur qui est probable de passer.

Elle va d'abord définir des "zones localisées" ou fenêtre dans lesquelles le paramètre se situe, et pour chaque zone dites local elle va sélectionner un ensemble de petits échantillons et analyser l'évolution de ce paramètre dans chacunes d'elles. Puis, finira par comparer en faisant une moyenne des variations entre le début et la fin de zone, pour avoir une explication sur l'ensemble des données. Une "accumulation" (comme l'indique son nom) des moyennes de variation de chaque zone locale indiquera l'effet complet du paramètre sur la prédiction.

Le fait d'utiliser des fenêtres, permet de donner des résultats plus précis et d'éviter des conditions contradictoire. C'est-à-dire si l'on reprend notre exemple de joggeurs, on ne prendra pas en compte une situation où la période de l'année est l'hiver et que la température soit de 35°.

L'ALE est souvent comparable au PDP(partial dependence plot) or que l'ALE est plus rapide et moins biaisés. Pour un même tracé, alors que ICE représenter l'effet individuel de chaque entrées et PDP l'effet le plus représentatif de cet ensemble, l'ALE va donner un graphe plus complexe avec des variations précises.



Cet exemple nous montre qu'en général plus la superficie augmente et plus le loyer est cher. Dans la zone où les données sont nombreuses i.e de 0 à 100 PDP est plus lisse que l'ALE, et dans la zone avec peu de données c'est l'inverse. Du coup PDP illustre de manière irréaliste cette portion tandis que l'ALE est plus droite. Le comportement le plus attendu pour la prédiction des loyers est plus probable au vu de l'ALE. On peut penser que l'ALE surpasse le PDP du fait que ce dernier combine des situations improbable d'où l'étrange baisse.

- Les avantages[22]:

- Les résultats sont non biaisés et fonctionnent malgré des données corrélées
- Les tracés sont rapides dû à une analyse par fenêtres
- L'interprétation est claire et agréable, avec des graphes centrés en zéro

- Les inconvénients[22]:

- Les tracés peuvent vite devenir instables i.e avec beaucoup de haut et de bas
- Les tracés sont complexes et moins intuitifs dans le cas où il y a beaucoup de données et plus encore s'ils sont corrélés
- Les estimations du second ordre sont plus complexes à analyser

Counterfactual

Une explication contrefactuelle décrit une situation causale de la forme : "Si X ne s'était pas produit alors Y ne se serait pas produit". Bien que dans la réalité, la relation entre les entrées et la prédiction n'est pas forcément causale, ces inputs peuvent tout de même être vus comme étant la cause de la prédiction. Le terme "contrefactuel" résume que l'on imagine en fait une situation hypothétique qui contredit les faits observés. Il s'agit d'une méthode model-agnostic car elle n'exploite que les input et la sortie.

En pratique les valeurs des features d'une instance sont modifiées avant de réaliser les prédictions et on regarde comme celles-ci en sont impactées. Un changement net dans la prédiction est un scénario intéressant (changement de catégorie, atteinte d'un seuil, etc.). Ainsi une explication contrefactuelle décrit tout changement des valeurs des caractéristiques qui modifie la prédiction en sortie.

Une bonne illustration d'explication contrefactuelle peut être la suivante : Peter Parker demande un prêt et est rejeté par le logiciel bancaire (alimenté par l'apprentissage automatique). Il se demande pourquoi sa demande a été rejetée et comment il pourrait améliorer ses chances d'obtenir un prêt. La question du "pourquoi" peut être formulée comme un contrefactuel : quelle est la plus petite modification des caractéristiques (revenu, nombre de cartes de crédit, âge, ...) qui ferait passer la prédiction de rejet à approbation ? Une réponse possible pourrait être la suivante : Si Peter gagnait 10 000 euros de plus par an, il obtiendrait le prêt, malheureusement être un super héros ne paye pas toujours très bien...

Pour générer des contrefactuels, on peut par exemple procéder par essai en changeant les valeurs des features de façon aléatoire jusqu'à obtenir la prédiction attendue. Une meilleure alternative se base sur l'optimisation d'une loss function qui prend comme entrée l'instance d'intérêt, le contrefactuel et la sortie attendue.

Parmi les critères permettant de qualifier la qualité de l'explication contrefactuelle :

- L'utilisateur définit un changement pertinent dans la prédiction, aussi un premier critère est que l'instance produise la prédiction attendue. On peut alors regarder les changements minimums à apporter aux features pour que la probabilité attendue augmente.
- Un second critère est qu'un contrefactuel soit aussi proche que possible de l'instance concernant les valeurs de features : le contrefactuel doit être proche de l'instance originale et modifier le moins de caractéristiques possibles.
- Un dernier critère est qu'une instance contrefactuelle doit avoir des valeurs de caractéristiques probables. Dans la pratique on favorise des explications contrefactuelles qui sont probables en

croisant toutes les données disponibles - par exemple on rejettera un contrefactuel pour un appartement de 10 pièces et de 20 m².

L'explication contrefactuelle présente l'avantage d'être interprétable : si les valeurs des features d'une instance sont modifiées selon l'explication contrefactuelle, la prédiction change par rapport à la prédiction attendue. Il n'y a pas d'hypothèses supplémentaires et aucun fonctionnement "mystérieux". Également, et comme présenté la méthode contrefactuelle ne nécessite pas l'accès aux données ou au modèle. Les explications contrefactuelles offrent donc un compromis entre l'explication des prédictions du modèle et la protection des intérêts du propriétaire du modèle.

Malheureusement les contrefactuels souffrent de "l'effet Rashomon"- film japonais dans lequel le meurtre d'un samouraï est raconté par différentes personnes -. Chaque contrefactuel explique différemment la façon dont un résultat a été atteint et les contrefactuels peuvent se contredire entre eux. Cette problématique des "vérités multiples" peut être traitée en rapportant toutes les explications contrefactuelles obtenues ou en définissant un critère d'évaluation pour sélectionner le meilleur.[4]

Score	GPA	LSAT	Race	GPA x'	LSAT x'	Race x'
0.17	3.1	39.0	0	3.1	34.0	0
0.54	3.7	48.0	0	3.7	32.4	0
-0.77	3.3	28.0	1	3.3	33.5	0
-0.83	2.4	28.5	1	2.4	35.8	0
-0.57	2.7	18.3	0	2.7	34.9	0

Exemple de contrefactuels [4]						
<i>La première colonne correspond aux scores attendus (prédictions attendues). Les 3 suivantes aux valeurs originales des features et les 3 dernières les explications contrefactuelles conduisant à un score proche de 0. Les contrefactuels pour les deux premières lignes décrivent comment les caractéristiques de l'étudiant devraient changer pour diminuer le score prévu et pour les trois autres lignes, comment elles devraient changer pour augmenter le score jusqu'à la moyenne.</i>						

LIME [7]

LIME va regarder l'incidence des variations des features dans le modèle d'apprentissage automatique sur les prédictions. Il va ensuite générer un nouvel ensemble de données composé d'échantillons permutés. Sur ce nouvel ensemble de données, il va former un modèle interprétable qui peut prendre plusieurs forme.

La méthode pour former des modèles de substitution locaux est la suivante:

- Sélectionner la feature d'intérêt pour laquelle vous souhaitez avoir une explication de sa prédiction de boîte noire.
- Perturber le jeu de données et obtenez les prédictions de la boîte noire pour ces nouveaux points.
- Pondérer les nouveaux échantillons en fonction de leur proximité avec la feature d'intérêt.
- Former un modèle pondéré et interprétable sur l'ensemble de données avec les variations.
- Expliquer la prédiction en interprétant le modèle local.

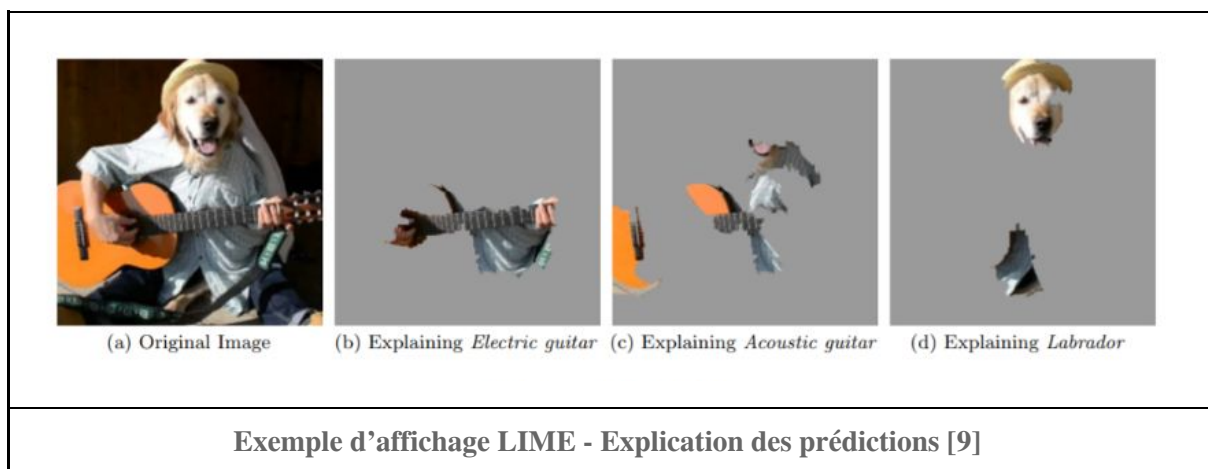
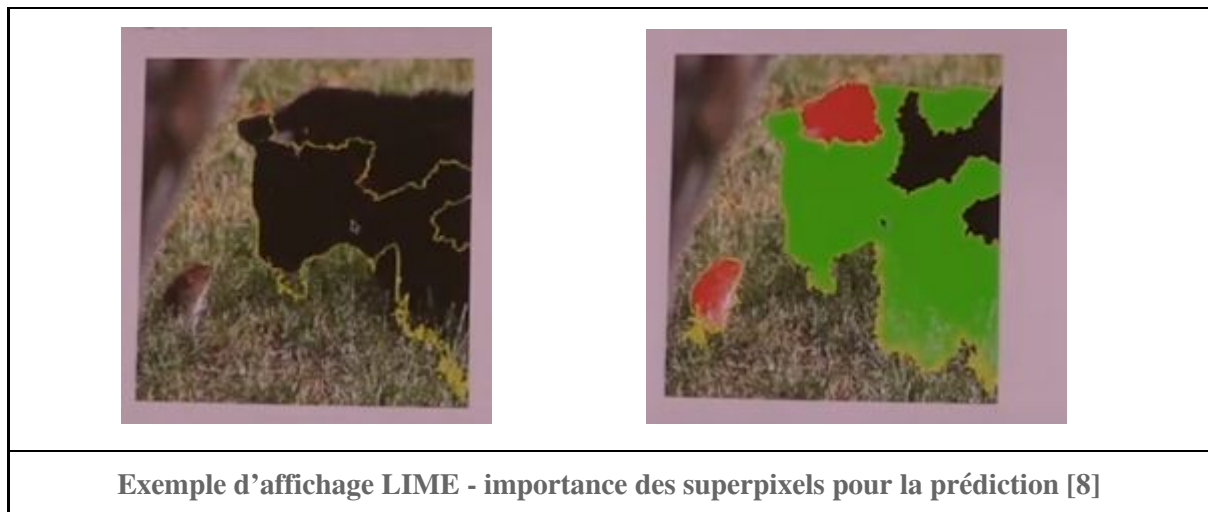
❖ Lime pour les données tabulaire

Pour l'utilisation de LIME sur les données tabulaires, il va commencer par définir des "quartiers" autour d'un point. Il va ensuite utiliser un noyau de lissage, c'est à dire une fonction qui va prendre deux "instances de données" afin de renvoyer des mesures de proximité de celles-ci. Plus les instances sont proches, plus la largeur du noyau est élevée. La largeur du noyau peut influencer le modèle. Un mauvais choix d'instances pourraient donc être un problème tant pour le modèle utilisé que pour l'interprétation des prédictions de celui-ci.

❖ Lime pour les images

Il va d'abord créer des super pixel (zone d'image). On peut ensuite voir les zones de l'images, donc quels superpixels, lui on permet de déterminer ce que cela représente. Il est également possible de faire l'inverse: c'est à dire, de voir les superpixels qui n'ont pas contribué à la prédiction.

Voici quelques exemples d'applications :



Avantages :

LIME fonctionne pour les données tabulaires, le texte et les images.

Il permet de donner une idée de la fiabilité du modèle interprétable.

Implanté en Python et R ce qui le rend simple d'utilisation.

Inconvénients:

Échantillonnages parfois approximatif qui mériterait d'être amélioré.

Problème de définition du quartier dans le cas de données tabulaires.

La complexité du modèle d'explication doit être définie à l'avance. (Cependant l'utilisateur doit toujours définir le compromis entre fidélité et rareté.)

Scoped rules: Anchor [20]

L'anchor ou anchor box est une technique qui donne des explications sur les prévisions individuelles des modèles de classification en trouvant une "règle de décision" autrement dit, en explicitant le principe de sa prédiction. Sa stratégie est basée sur des perturbations afin de générer des explications locales, du coup elle explore l'ensemble du problème grâce aux perturbations qu'elle même crée et évalué seule, elle ne tient pas compte des paramètres internes et agit indépendamment du modèle.

L'explication est exprimée sous forme de règles "IF-THEN" très faciles à comprendre et réutilisable, car elle utilise une couverture: À savoir, elle indique à quel autre paramètre même masqué elle peut s'appliquer en plus de celui sélectionné. Ainsi, l'anchor construit une couverture adapté à un modèle et explique ces limites en indiquant clairement qu'elles cas sont valides.

Elle corrige alors un défaut majeur des méthodes d'explication locales comme le LIME, car elle à la capacité de conserver une prédiction. C'est-à-dire qu'elle ne subit pas de modifications lorsque d'autres caractéristiques sont modifiées: la prédiction est immuable.

Feature	Value
Age	20
Sex	female
Class	first
TicketPrice	300\$
More attributes	...
Survived	true

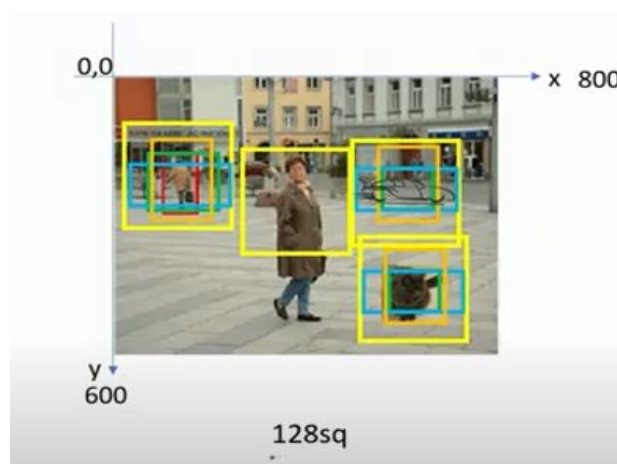
Dans cet exemple, on suppose que l'on a un modèle qui prédit si un passager ou non a survécu au naufrage du Titanic en fonction du sex de ce dernier.

```
IF SEX = female
AND class = first
THEN PREDICT Survived = true
WITH PRECISION 97%
AND COVERAGE 15%
```

Résultats: les femmes en première classe ont survécu avec une précision de 97%.

L'anchor box peut utiliser des données textes, tabulaires ou des images. Pour les images, elle les segmente en superpixels tout en conservant la structure d'image locale puis la représentation interprétable est identifiée par l'absence ou la présence de ces superpixels. Pour se faire, elle utilise des techniques de segmentations d'image pour diviser l'image en superpixel comme le slic ou le quickshift.

Dans certains algorithmes de détection d'éléments de vision par ordinateur elle aide à identifier des objets de différentes formes. Elle représente des boîtes qui recueillent différentes données de rapports hauteur / largeur afin de correspondre aux rapports relatifs des classes d'objets détectés. Elles enregistrent la taille et le rapport d'aspect des classes d'objets spécifiques qu'on souhaite détecter qui sont généralement choisies en fonction de la taille des objets dans le jeu de données d'apprentissage.

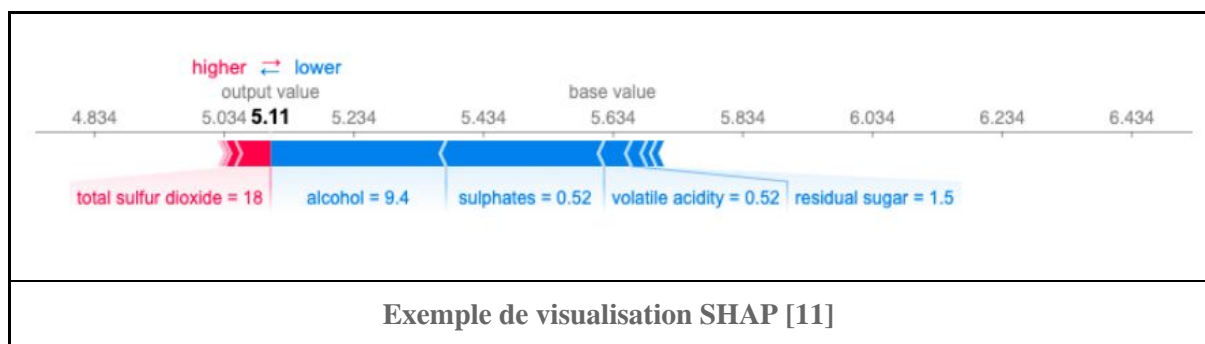


Exemple d'identification des objets sur une image [21]

SHAP [10]

SHAP permet l'interprétation d'un modèle peu importe celui-ci. Pour une utilisation optimale, SHAP utilise plusieurs ancienne méthode comme LIME et Shapley values. De ce fait SHAP est plus complexe que shap values, il en récupère la formule pour transformer les "joueurs" en variable du modèle et leur "récompense" en sortie correspond à leur importance. C'est une méthode qui peut être longue d'un point de vue calculatoire mais qui a été optimisé en utilisant la formule de LIME.

SHAP propose différents moyens de visualisation. Ceux-ci montrent l'évolution de la prédiction à chaque ajout de variable. La valeur initiale correspondant à la moyenne des prédictions sur les données d'entrainements du modèle.



Avantages :

Base théorique solide en théorie des jeux.

Prédiction équitablement répartie entre les valeurs des entités

Meilleure compréhension de LIME et Shapley. Contribue à unifier le domaine de l'apprentissage automatique interprétable.

Possède une implémentation rapide pour les modèles arborescents. Calcul rapide qui permet de calculer les nombreuses valeurs de Shapley nécessaires à l'interprétation du modèle global.

Inconvénients :

KernelSHAP est lent ce qui le rend peu pratique à utiliser lors des calculs des valeurs de Shapley. Il ignore également la dépendance des fonctionnalités.

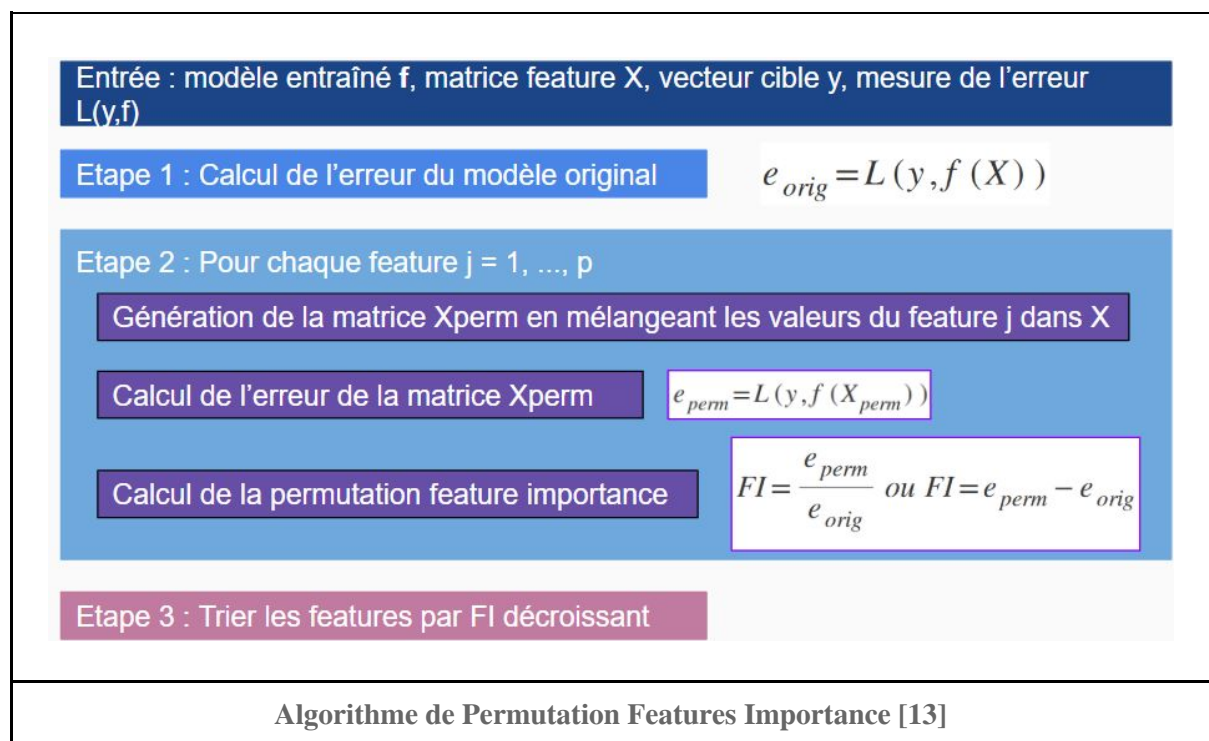
TreeSHAP peut produire des attributions de fonctionnalités non intuitives.

Les inconvénients des valeurs de Shapley ainsi que LIME s'appliquent également à SHAP.

Permutation Features Importance [12]

La “Permutation Features Importance” est une technique permettant de mesurer l’importance des différents “features” d’un jeu de données tabulaire. Le principe est simple, on mélange aléatoirement les différentes valeurs d’un “feature” et on regarde comment cela impacte les prédictions. On fait de même pour le reste des “features”. Cette technique permet de “casser” le lien entre les features et la vecteur cible de notre jeu de données ce qui en conséquence entraîne une baisse dans la prédiction et une hausse des erreurs si la “feature” mélangée est importante. A l’inverse, si la “feature” est peu importante, la précision et l’erreur seront peu impactées.

Le principe de la technique est décrit avec l’algorithme ci-dessous :



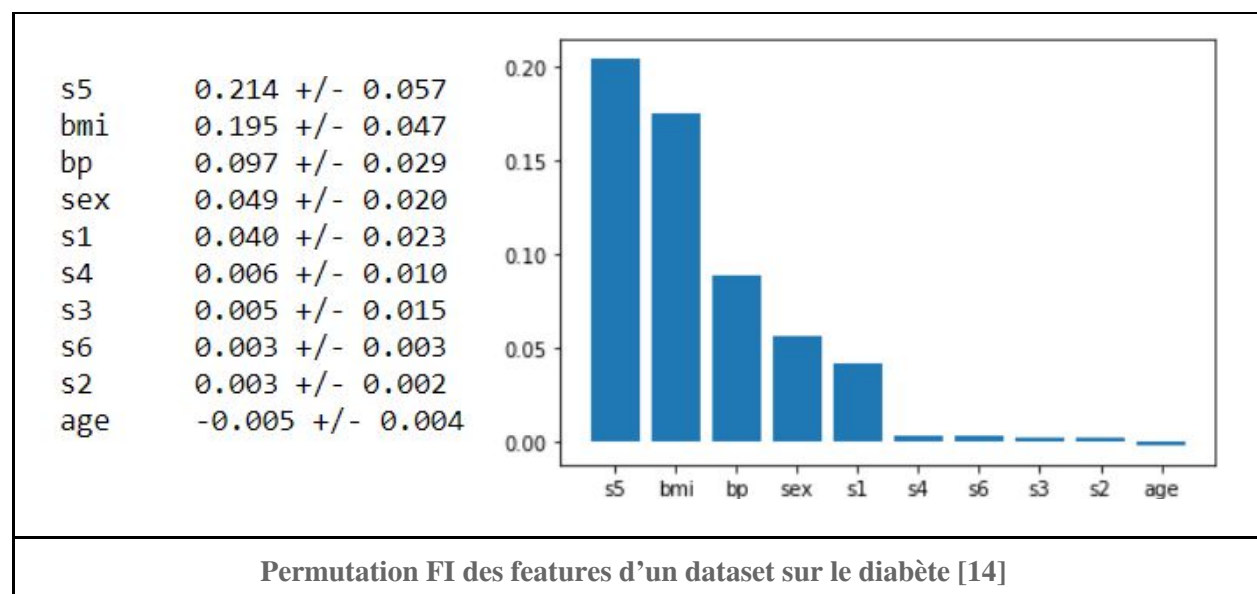
La bibliothèque Sklearn propose une fonction python qui permet de calculer les “Permutation Features Importances” d’un jeu de données. La fonction *permutation_importance* de Sklearn prend en entrée plusieurs paramètres :

- le modèle entraîné
- la matrice de test X avec les différents features
- le vecteur cible de test y
- la fonction d’erreur à utiliser

- le nombre de répétition (nombre de fois que l'on mélange un feature, la FI finale est égale à la moyenne des FI des répétitions)

(FI = score de feature importance)

La fonction renvoie un score pour chaque feature du dataset, le score d'un feature est composé d'un premier nombre caractérisant la diminution de la performance du modèle suivi par un second nombre qui montre la variation du score pour toutes les répétitions. Plus un score est haut plus le feature est important, c'est pour cela que généralement les scores de Permutation FI sont présentés dans un tableau décroissant. La fonction peut parfois renvoyer des scores négatifs pour certains features dû à des prédictions plus précises du dataset permuté. Ces précisions sont causés par le hasard, elles ne sont pas significatifs. Les features avec des scores négatifs ont en réalité des scores très proche de 0. Prenons l'exemple d'un jeu de donnée sur le diabète, nous allons chercher à trouver les features les plus importants pour savoir si une personne est atteinte de diabète. Pour cela, on commence par entraîné notre jeu de données avec la fonction *train_test_split*, cette fonction va renvoyer les matrices d'entraînements *X_train* et *y_train* et les matrices de tests *X_test* et *y_test*. Les matrices d'entraînements vont permettre de construire le modèle. On peut alors calculer les FI de nos features avec le modèle et les matrices de tests. Les features les plus importantes sont donc le s5 (lamotrigine), l'imc, la pression artérielle moyenne et le sexe.



La technique de "Permutation features importance" est très facile d'utilisation et elle a l'avantage de pouvoir être utilisée pour tout type de modèle mais elle a l'inconvénient d'introduire des fausses

estimations des features lorsque celles-ci sont corrélées. Il est donc important d'étudier également la corrélation entre les features pour avoir la meilleure analyse du dataset.

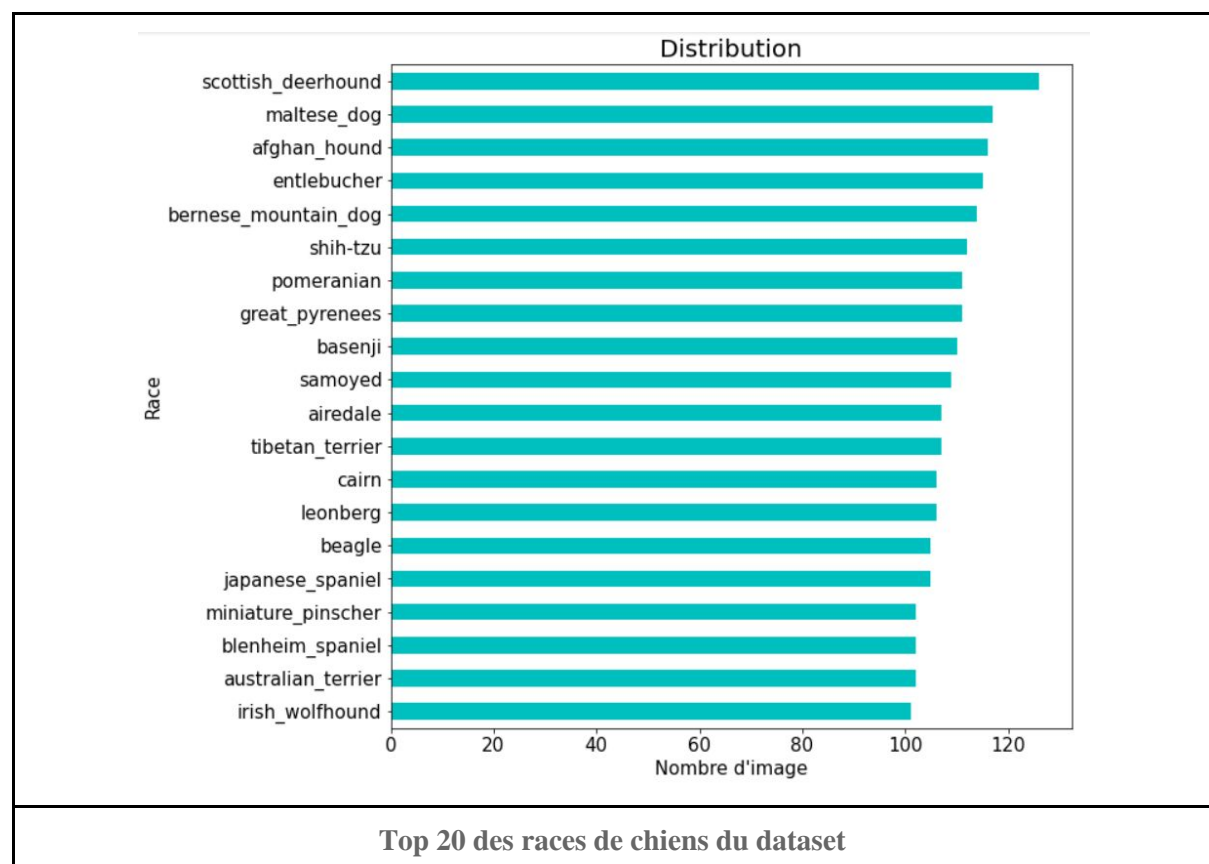
Classification d'image : races de chiens

En ce qui concerne la classification d'image, nous avons décidé de travailler sur un dataset sur les races de chiens récupéré sur le site Kaggle. Ce dataset contient plus de 120 races de chiens avec plus de 20000 photos de chiens séparée en deux pour l'entraînement et le test du modèle de prédiction.

Il n'existe pour l'instant que très peu de technique d'interprétabilité pour des données de classification d'image, nous n'avons donc pu que utiliser la technique LIME pour interpréter notre dataset. Pour réaliser l'étude avec Lime, nous avons repris des travaux publiés par Nexmo [15].

Tout le travail sur ce dataset a été réalisé sur [Google Colab](#). Nous avons passé beaucoup de temps sur l'importation des photos sur le colab. Nous avons initialement essayé d'importer le dataset en le téléchargeant de l'ordinateur vers colab mais cela prenait beaucoup de temps. Nous avons donc cherché une manière plus rapide pour réaliser l'importation. En faisant l'importation avec Kaggle API, l'importation était beaucoup plus efficace mais cela n'a pas été concluant puisque l'on a eu des erreurs avec le traitement des images ensuite. Nous avons donc conclu que la manière de faire l'importation était de charger les images sur le drive pour avoir un accès direct aux images.

A cause des limitations des systèmes de nos ordinateurs, il a été impossible de travailler avec tous les données de notre dataset. Nous avons donc réalisé notre étude avec les 20 races de chiens les plus fréquentes dans le dataset.



Le premier modèle que l'on a étudié avec ce dataset est un réseau de neurone convolutif. L'entraînement d'un modèle comme celui-ci prend beaucoup de temps et de ressources afin de contourner cela on utilise la méthode de Transfert d'apprentissage.

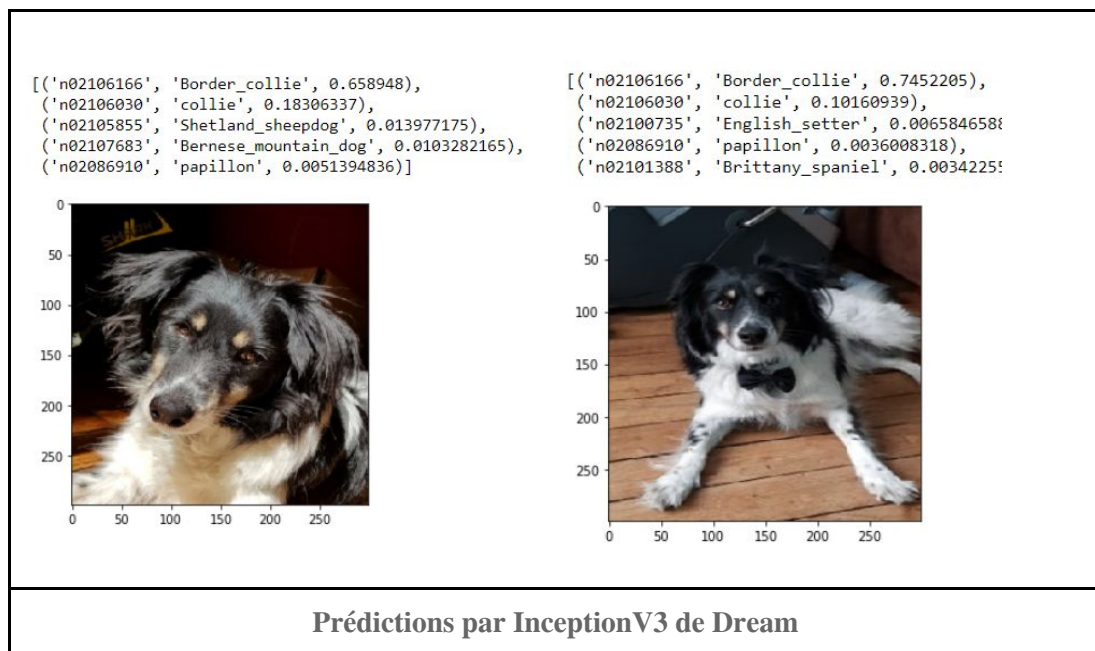
La technique consiste à créer un modèle d'apprentissage en prenant pour base un modèle déjà entraîné et performant. Dans le cas d'un réseau de neurones où chaque couche apprend de la précédente, les premières couches du réseau sont spécialisés dans la reconnaissance de formes simples, les couches qui suivent sur les reconnaissances de formes de plus en plus complexes. Les dernière couche s'occupe à l'objet de l'apprentissage. Il suffit donc de remplacer la ou les dernières couches du réseaux par une couche dédiée à notre problème (ici l'identification de race de chien) et d'entraîner le réseau sur le dataset [16].

Keras propose aujourd'hui de nombreux modèle pré-entraîné pour du transfert d'apprentissage. Ces modèles peuvent reconnaître plus de 1000 catégories d'objets. Parmi ces modèles, il en existe 5 très performant sur la classification d'image : VGG16, VGG19, ResNet50, InceptionV3 et Xception [17].

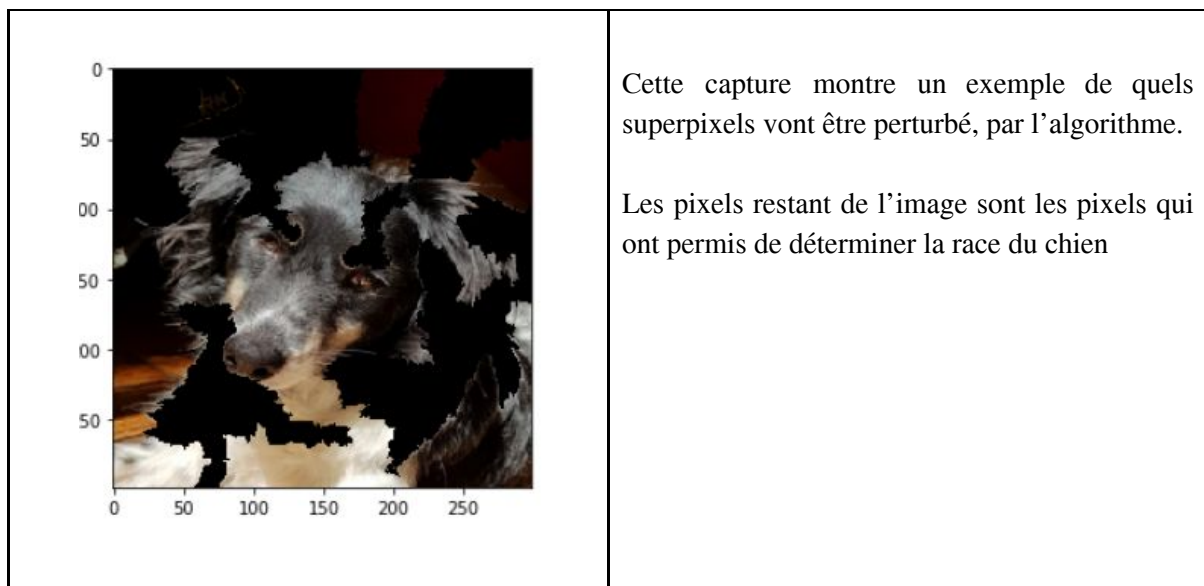
Test sur LIME avec InceptionV3 [18]

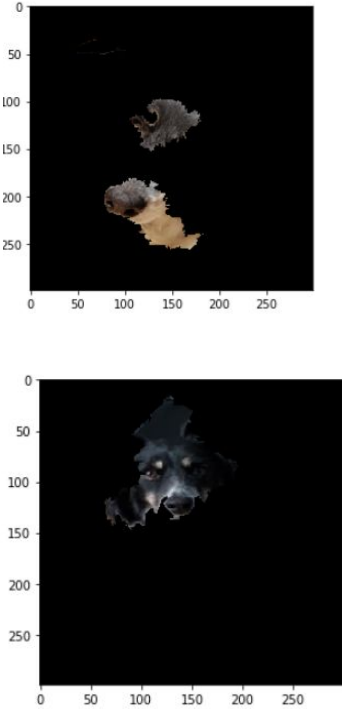
Nous avons précédemment traité le modèle InceptionV3 pour l'étude de la technique LIME.

Morgane avait pris des photo de son chien Dream, un chien abandonné donc les origines étaient peu connu. Elle avait fait des recherches pour déterminer les origines de Dream. Avec InceptionV3, elle a retrouvé les résultats de ses recherches :



L'algorithme renvoie les prédictions de races pour le chien en fonction des photos fournis. On remarque qu'en fonction de la photo les prédictions peuvent varier mais que les deux premier choix reste prédominant.




	<p>Ces images nous montre les features qui ont eu le plus de poids pour la prise de décision.</p> <p>Pour cette étape Morgane a fait une estimation du modèle linéaire à l'aide de <code>sklearn.linear_model</code>.</p> <p>Nous voyons ici les 4 super pixels les plus important.</p>
---	---

Préapprentissage de notre dataset avec Xception

En utilisant le modèle créé en prenant en modèle de base Xception qui a de très bonnes performances, nous commençons par tester le modèle avec une image d'un spaniel japonais.

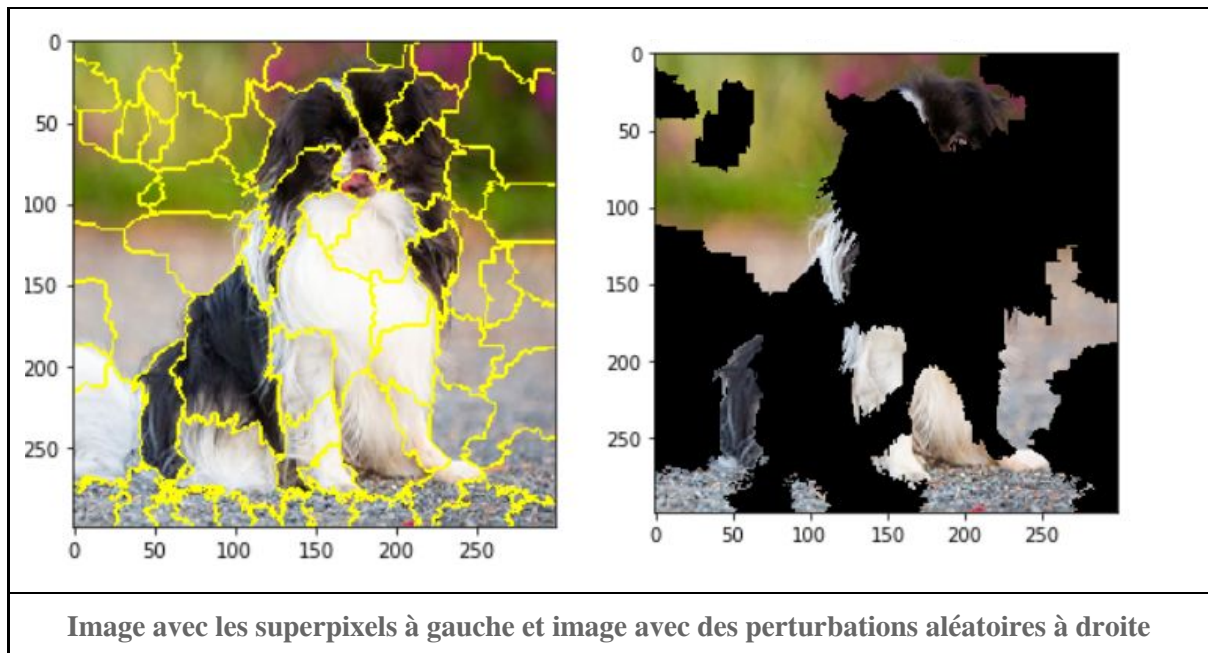
Le modèle renvoie la bonne prédiction, il a bien déterminé la race du chien :

 <table data-bbox="408 1756 772 1872"> <tr> <td>japanese_spaniel</td> <td>0.9999999</td> </tr> <tr> <td>shih-tzu</td> <td>5.209092e-08</td> </tr> <tr> <td>tibetan_terrier</td> <td>1.2274754e-08</td> </tr> <tr> <td>pomeranian</td> <td>1.0954396e-08</td> </tr> <tr> <td>blenheim spaniel</td> <td>9.2852845e-09</td> </tr> </table>	japanese_spaniel	0.9999999	shih-tzu	5.209092e-08	tibetan_terrier	1.2274754e-08	pomeranian	1.0954396e-08	blenheim spaniel	9.2852845e-09	<table data-bbox="804 1451 1182 1800"> <tr> <td>scottish_deerhound</td> <td>2.5037283e-09</td> </tr> <tr> <td>miniature_pinscher</td> <td>1.0149122e-09</td> </tr> <tr> <td>bernese_mountain_dog</td> <td>9.52235e-10</td> </tr> <tr> <td>entlebucher</td> <td>4.023719e-10</td> </tr> <tr> <td>afghan_hound</td> <td>2.641428e-10</td> </tr> <tr> <td>samoyed</td> <td>2.3459049e-10</td> </tr> <tr> <td>maltese_dog</td> <td>1.566362e-10</td> </tr> <tr> <td>irish_wolfhound</td> <td>1.176409e-10</td> </tr> <tr> <td>great_pyrenees</td> <td>1.12679754e-10</td> </tr> <tr> <td>leonberg</td> <td>6.0490314e-11</td> </tr> <tr> <td>airedale</td> <td>3.3977002e-11</td> </tr> <tr> <td>australian_terrier</td> <td>2.0878953e-11</td> </tr> <tr> <td>basenji</td> <td>6.941756e-12</td> </tr> <tr> <td>cairn</td> <td>6.0446453e-12</td> </tr> <tr> <td>beagle</td> <td>2.7521436e-12</td> </tr> </table> <p>Prediction : <u>japanese_spaniel</u></p>	scottish_deerhound	2.5037283e-09	miniature_pinscher	1.0149122e-09	bernese_mountain_dog	9.52235e-10	entlebucher	4.023719e-10	afghan_hound	2.641428e-10	samoyed	2.3459049e-10	maltese_dog	1.566362e-10	irish_wolfhound	1.176409e-10	great_pyrenees	1.12679754e-10	leonberg	6.0490314e-11	airedale	3.3977002e-11	australian_terrier	2.0878953e-11	basenji	6.941756e-12	cairn	6.0446453e-12	beagle	2.7521436e-12
japanese_spaniel	0.9999999																																								
shih-tzu	5.209092e-08																																								
tibetan_terrier	1.2274754e-08																																								
pomeranian	1.0954396e-08																																								
blenheim spaniel	9.2852845e-09																																								
scottish_deerhound	2.5037283e-09																																								
miniature_pinscher	1.0149122e-09																																								
bernese_mountain_dog	9.52235e-10																																								
entlebucher	4.023719e-10																																								
afghan_hound	2.641428e-10																																								
samoyed	2.3459049e-10																																								
maltese_dog	1.566362e-10																																								
irish_wolfhound	1.176409e-10																																								
great_pyrenees	1.12679754e-10																																								
leonberg	6.0490314e-11																																								
airedale	3.3977002e-11																																								
australian_terrier	2.0878953e-11																																								
basenji	6.941756e-12																																								
cairn	6.0446453e-12																																								
beagle	2.7521436e-12																																								
<p align="center">Prédiction d'un chien Spaniel Japonais avec le modèle CNN</p>																																									

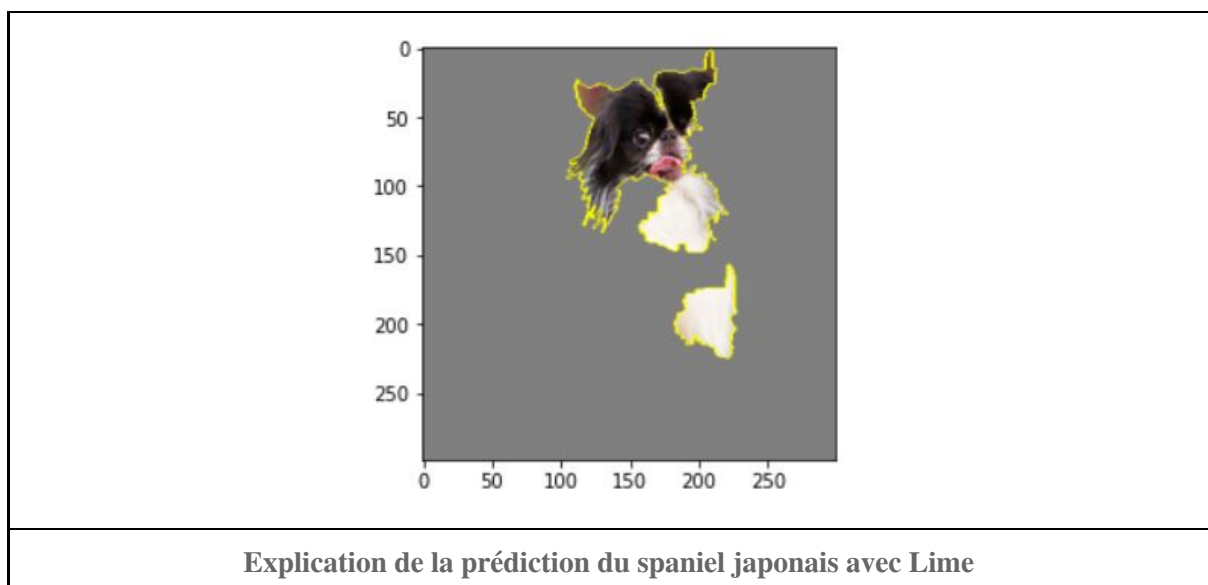
Le modèle calcule des scores pour chaque races du modèle, on a un score très haut pour la race Spaniel Japonais et les scores des autres races sont très faibles.

En faisant l'étude avec Lime, on détermine les pixels qui ont permis la prédiction et les calculs des scores.

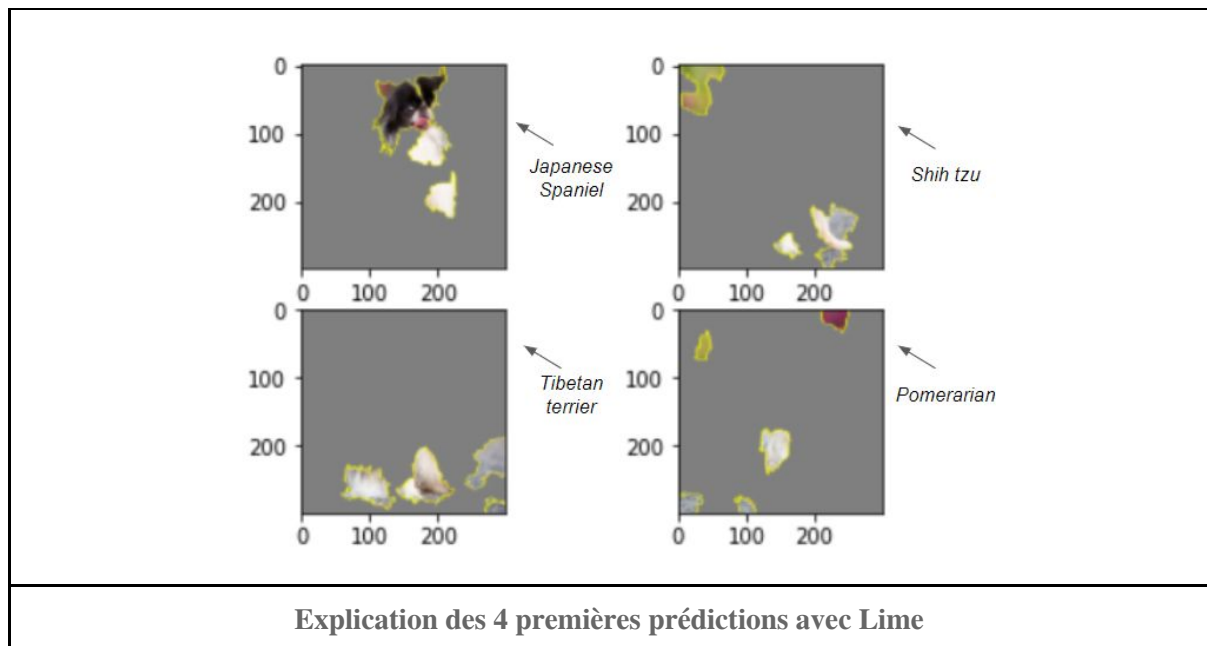
L'application de la technique commence pas les déterminations des superpixels et l'affichage de perturbations aléatoires.



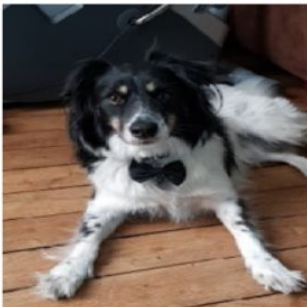
Les pixels qui ont permis de déterminer la race sont mis en avant grâce à Lime :



Les scores des autres races du modèles sont causés par l'environnement comme on peut le voir avec Lime :



Nous avons ensuite testé la prédiction du modèle avec la photo de Dream. Ce chien, contrairement aux autres chiens du dataset, n'est pas un chien de race pure. Ce qui sa prédiction plus intéressante. Le modèle retourne comme race le spaniel japonais, cette race ne correspond pas au races que l'on avait déterminé :

 <pre> japanese_spaniel 0.99991906 bernese_mountain_dog 1.6473736e-05 tibetan_terrier 1.4942895e-05 great_pyrenees 1.312797e-05 shih-tzu 1.09209195e-05 scottish_deerhound 1.06063e-05 blenheim_spaniel 5.7596535e-06 irish_wolfhound 3.050399e-06 samoyed 2.8224065e-06 entlebucher 1.1645515e-06 maltese_dog 7.947923e-07 pomeranian 4.9610725e-07 beagle 2.1091556e-07 miniature_pinscher 1.4814833e-07 leonberg 9.03924e-08 basenji 5.795351e-08 airedale 4.832749e-08 australian_terrier 3.8832894e-08 cairn 3.356146e-08 afghan_hound 1.4511872e-08 </pre>	<p>Résultats des prédiction de races du modèle CNN.</p> <p>On peut voir les différents scores attribués pour les différentes races que l'on avait sélectionnés au début du traitement du dataset.</p> <p>La race "Japanese Spaniel" a le score le plus important avec 0.99.</p>
--	---

Nous avons donc cherché les similitudes entre Dream et un spaniel japonais.




Comparaison entre Dream et un spaniel japonais

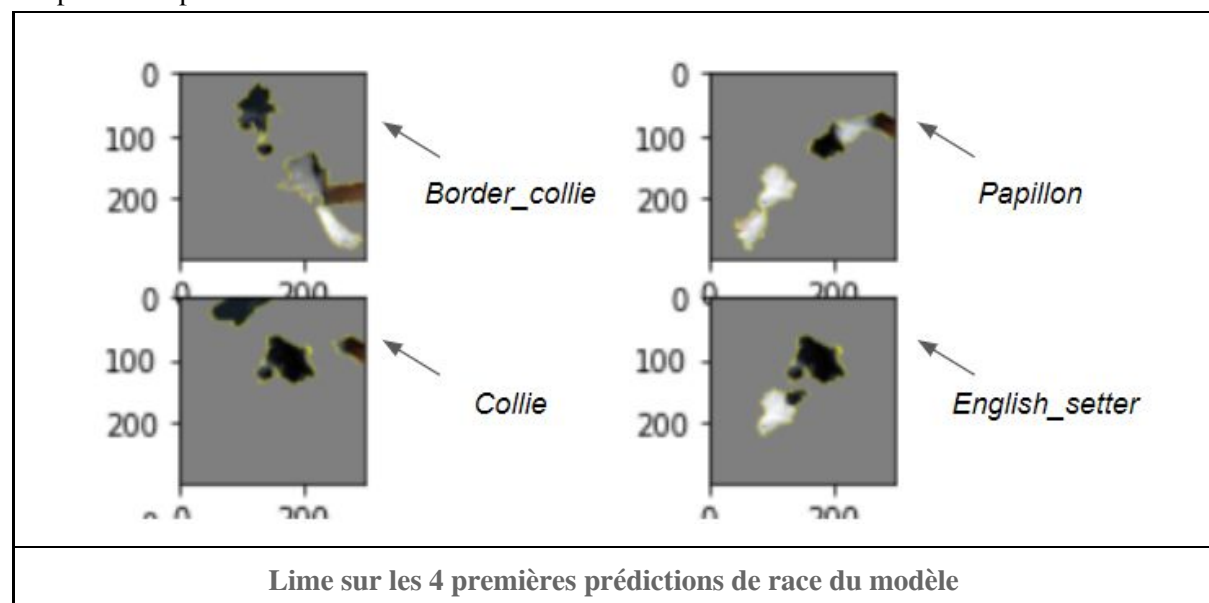
Nous constatons que les deux chiens ont plusieurs points en commun : la texture de leur poil, la couleur des poils et la répartition des couleurs sur leur fourrure.

Les races que l'on avait trouvées avec InceptionV3 ne sont pas présentes dans le modèle car nous avons gardé seulement les 20 premières races du dataset. Le modèle a donc associé Dream avec la race de chien la plus ressemblante.

Nous avons donc refait l'apprentissage du modèle en sélectionnant les races pertinentes. Avec ce nouveau modèle, nous avons réussi à retrouver les prédictions que l'on attendait.

 <pre> border_collie 0.9698392 papillon 0.023476725 collie 0.006406041 english_setter 0.00016752523 brittany_spaniel 4.797258e-05 samoyed 2.797359e-05 tibetan_terrier 1.9683961e-05 great_pyrenees 5.601904e-06 bernese_mountain_dog 2.3009645e-06 pomeranian 2.1805215e-06 maltese_dog 2.0071818e-06 shih-tzu 1.0851442e-06 scottish_deerhound 3.7671632e-07 entlebucher 3.457029e-07 basenji 2.621255e-07 cairn 2.1121512e-07 beagle 1.4221621e-07 airedale 1.2804885e-07 leonberg 5.165822e-08 afghan_hound 1.0849361e-08 Prediction : border collie </pre>	<p>Résultat des prédictions du modèle en intégrant préalablement le bonnes races.</p> <p>On retrouve ici les races que l'on avait déterminées avec InceptionV3.</p>
--	---

Nous avons donc pu poursuivre en faisant l'analyse avec Lime. Lime nous permet de déterminer quels pixel ont permis au modèle de faire ses précisions. On peut voir ci-dessous les zones sur l'image qui ont permis de prédire les différentes races.



Nous pourrions améliorer notre travail en ajoutant de nouveau modèle comme une régression logistique et un random forest.

Modèles

Pour la suite nous avons choisi de créer une fonction par technique d'interprétabilité afin de les intégrer dans la TOOL BOX dans l'objectif de la rendre la plus adaptable possible.

De ce fait, afin de réaliser notre TOOL BOX il nous a fallu développer des fonctions applicables à n'importe quel modèle de machine learning. Pour cela nous avons travaillé sur Google colab sur 5 notebooks - un notebook par modèle testé : Linear regression, Random forest classifieur, Random forest regressor, XGBoost et Neural Network - de telle sorte à trouver les architectures de fonction applicables à l'identique à tous les modèles. Nous avons cherché à tester ça sur un nombre maximal de modèles pour nous en assurer.

Le format des données doit être géré pour pouvoir appliquer les différentes techniques (applicables sur des dataframe par choix), aussi si les données sont au format np.array elles sont converties en dataframe par la suite.

En ce qui concerne SHAP, différents explainers existent selon le type de modèle, à savoir :

- `shap.TreeExplainer(model)` pour les arbres
- `shap.DeepExplainer(model)` pour les réseaux de neurones
- `shap.KernelExplainer(model)` pour tous les modèles, mais il est plus lent et renvoie des approximations des SHAP values

Nous avons décidé d'utiliser le dernier explainer pour la raison exposée précédemment : avoir une fonction présentant le moins de sous cas possible et applicable telle quelle à l'ensemble des modèles.

Au final, nous avons généré des graphiques pour chaque technique et pour tous les features du jeu de données, pour que l'utilisateur ait la possibilité de tous les passer en revue.

Boîte à outils avec Tkinter

Pour la réalisation de la boîte à outils nous avons créé une interface graphique en Python avec Tkinter, notre produit se nomme TOOL BOX. TOOL BOX permet la génération d'un report complet des résultats des techniques d'interprétabilité suivantes : PDP, ICE, LIME, SHAP et Permutation importance dont les graphiques générés sont enregistrés sous forme de .png dans le répertoire courant de l'utilisateur.

La boîte à outils prend en entrée le modèle entraîné, le dataset - données tabulaires - et les données (d'entraînement et de test). L'utilisateur choisit les techniques d'interprétabilité voulues.

Un dataframe peut être envisagé comme possédant les extensions suivantes d'après le [site Towards datascience](#) :

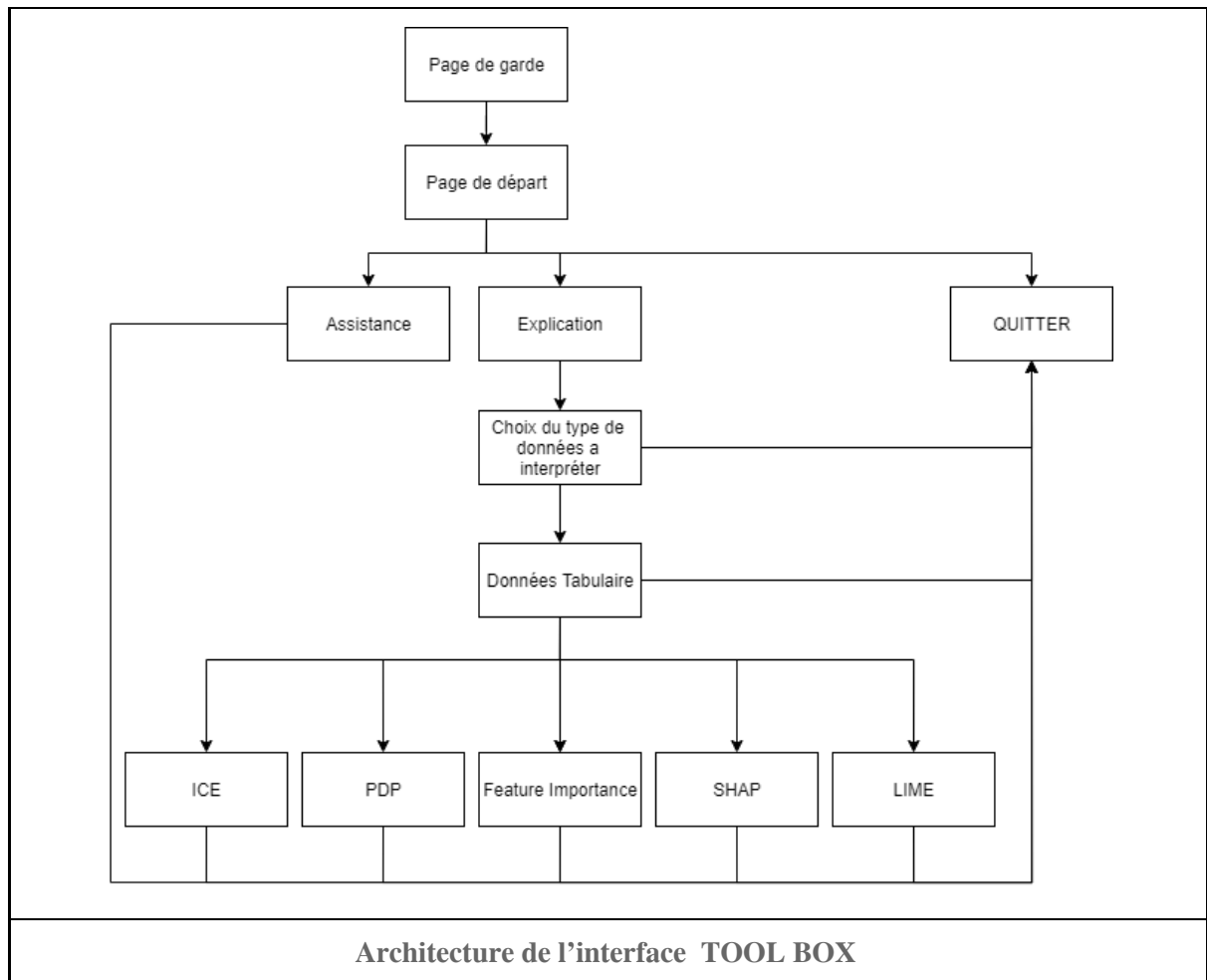
- npy
- csv
- pickle
- hdf5

De ce fait le dataset et les données (d'entraînement et de test) seront considérés comme susceptibles de présenter l'une de ces extensions.

Nous considérerons que le modèle est sous l'extension pickle ou .sav.

Une fois les fichiers chargés dans la boîte à outils l'utilisateur peut appuyer sur le bouton "Générer" afin d'obtenir le résultat de la technique d'interprétabilité choisie et un .png correspondant dans son répertoire courant.

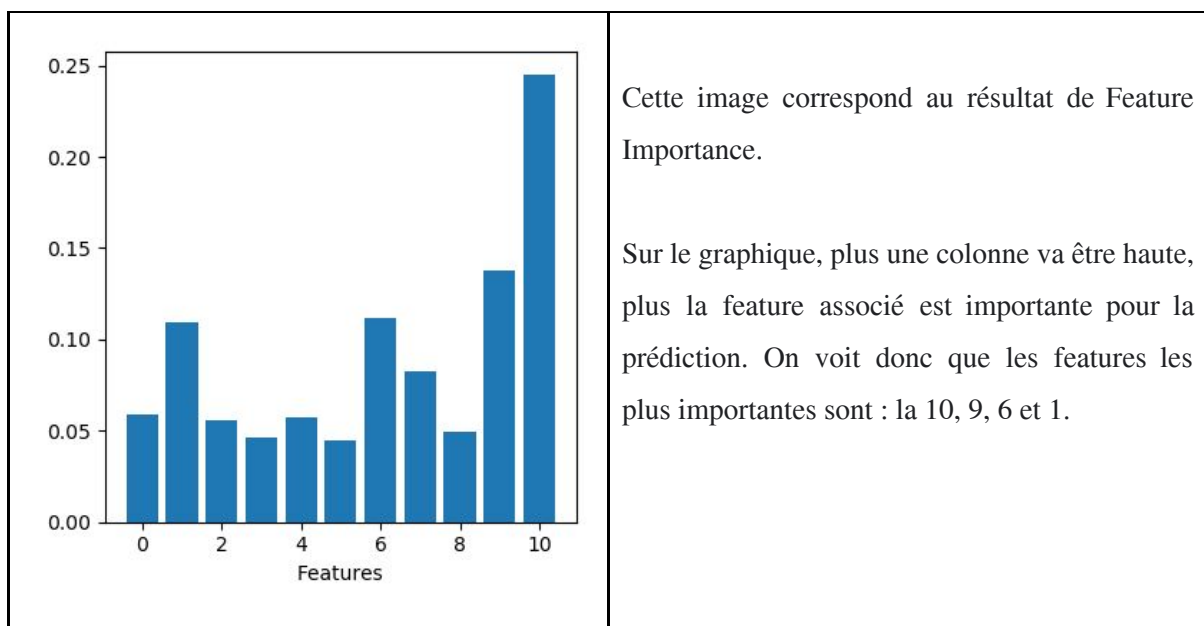
La boîte à outils possède une page de présentation - qui expose le principe d'utilisation et les bibliothèques nécessaires - d'assistance - qui présente le sujet, la TOOL BOX et ses entrées - et une page permettant de quitter.

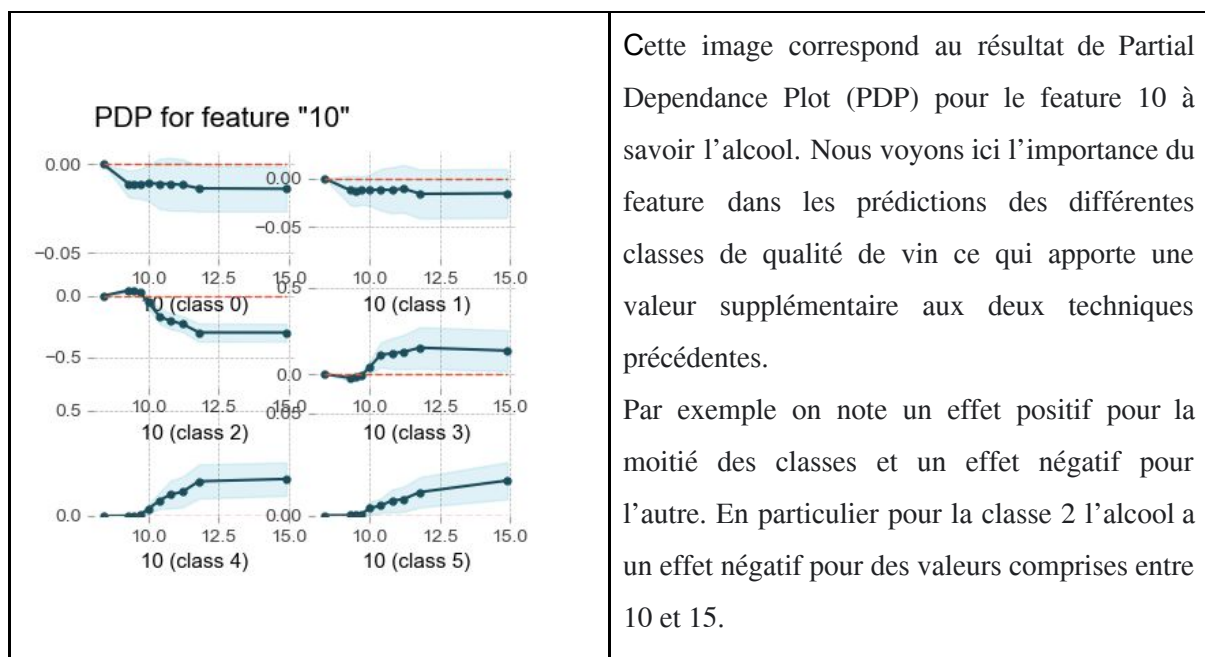
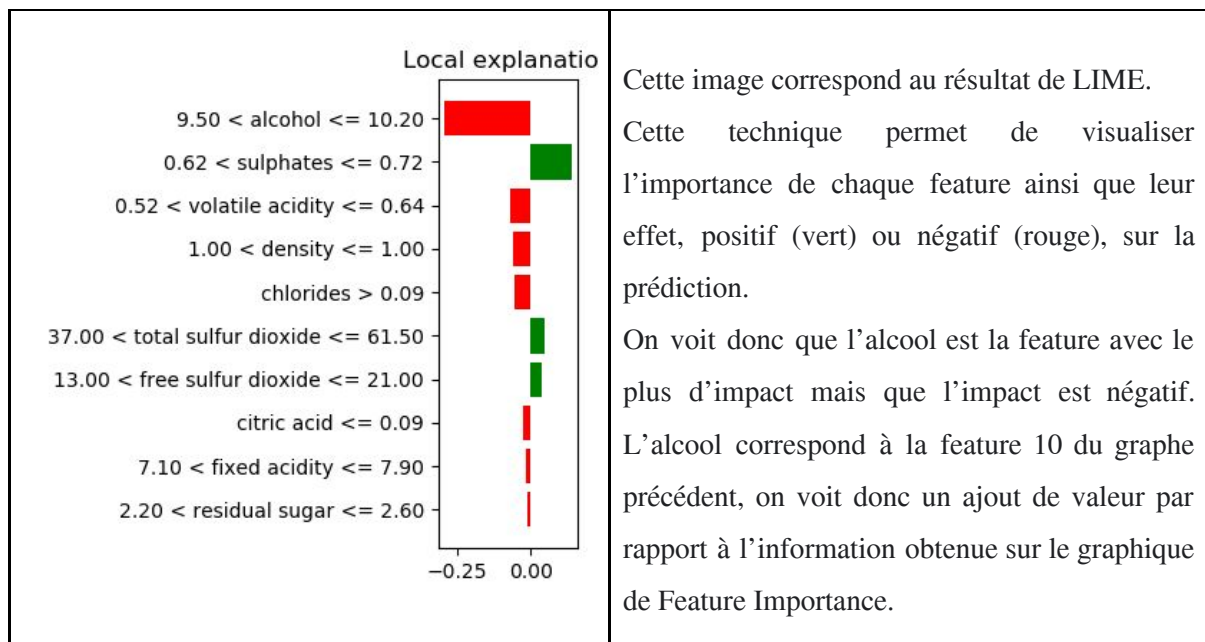


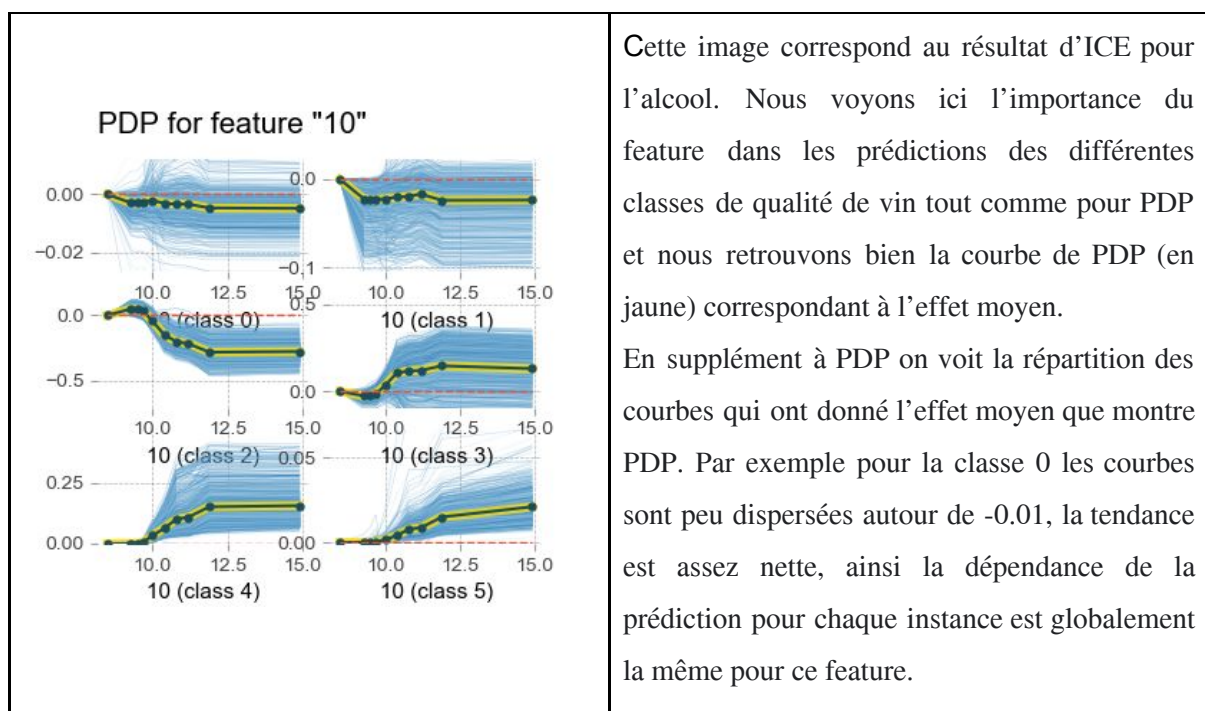
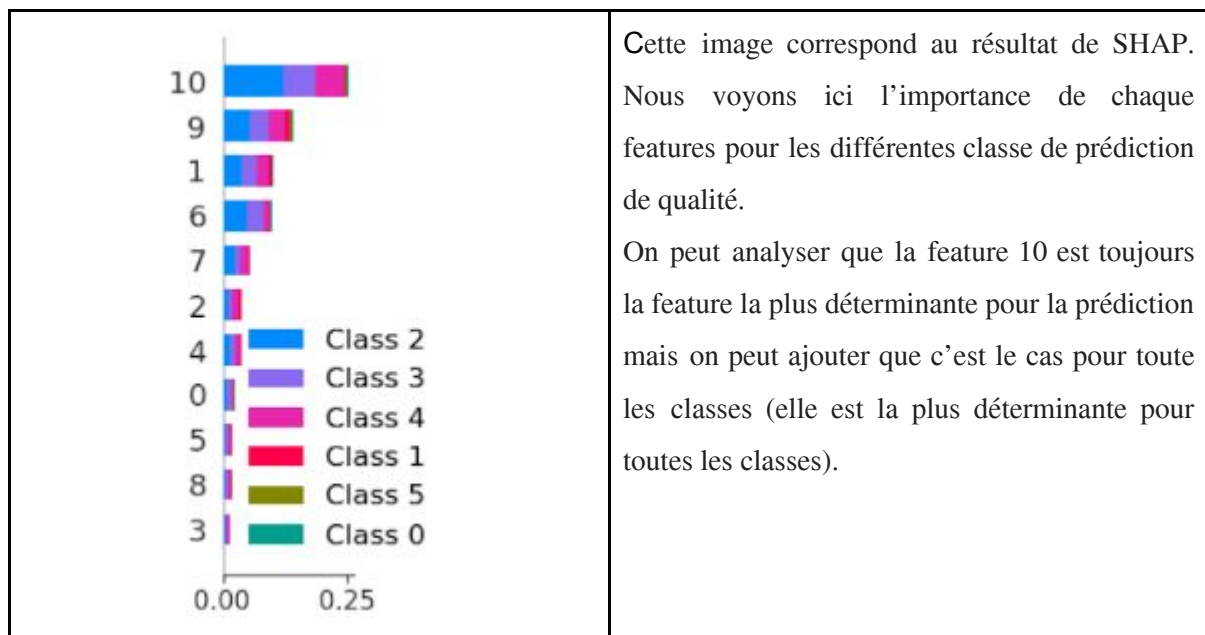
Application de TOOL BOX à un problème de machine learning supervisé : prédiction de la qualité d'un vin par un algorithme Random forest classifieur

Afin de tester nos différents code d'interprétation, nous les avons testé sur un jeu de données tabulaire Kaggle à savoir un fichier .csv regroupant des mesures physico chimiques sur un vin portugais. Nous avons ensuite entraîné un modèle de type Random Forest classifieur pour prédire la variable "quality". Le dataset contient les variables suivantes: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol.

Voici une sélection des résultats obtenus avec notre toolbox sur ce jeu de données et mis en parallèle pour l'analyse :







Améliorations envisagées

Afin de proposer un maximum de techniques pertinentes, pour générer le report le plus complet possible, nous voulions ajouter Permutation importance aux techniques déployées. La mise en place de la fonction sur Python et son exécution ne posait aucun problème mais lorsque intégrée dans l'interface graphique une erreur est générée. L'erreur en question semble être causée par l'implémentation avec Tkinter et nous n'avons pas eu le temps de trouver une solution. Une amélioration logique aurait donc été de pouvoir intégrer cette technique dans le panel de méthodes proposées.

Concernant PDP, ICE et SHAP les fonctions génèrent plusieurs graphiques et nous avons commencé à travailler sur la mise en place de boutons permettant de passer d'une image à l'autre - type Previous/Next. Afin de perfectionner et de finaliser notre TOOL BOX il nous aurait fallu pouvoir les intégrer.

La suite logique aurait été de proposer les mêmes fonctionnalités sur des données de type image ce que nous n'avons pas pu réaliser par manque de temps. De ce fait il aurait fallu développer des fonctions - pour les techniques adaptées aux images - identiques peu importe le modèle pris en entrée.

Nous aurions souhaité ajouter la possibilité d'afficher des statistiques sur le jeu de données entré (nombre de features, dimension du jeu, boîtes à moustache, distribution des variables, etc.) et également un affichage du modèle - nous avons pensé à l'affichage d'une branche de la forêt pour un Random forest, aux coefficients pour la régression et XGBoost, aux courbes d'accuracy et de lost pour un réseau de neurones. Pour l'intégrer dans la boîte à outils, nous aurions ajouté un bouton sur la page de choix du type de données à interpréter (tabulaire/image) "Statistiques et modèle".

Un dernier point concerne l'amélioration de la boîte à outils en terme de performances : il serait intéressant de rendre l'affichage des graphiques plus rapide, et de créer automatiquement un dossier et des sous-dossiers dans le répertoire courant où seraient enregistrées les images générées pour que l'utilisateur récupère un report structuré directement.

Sources

[1] :

https://www.lemonde.fr/pixels/article/2018/03/28/intelligence-artificielle-ce-qu-il-faut-retenir-du-rapport-de-cedric-villani_5277697_4408996.html

[2] :

<https://www.usine-digitale.fr/article/intelligence-artificielle-et-rgpd-peuvent-ils-cohabiter.N924164>

[3] :

<https://www.actuia.com/contribution/jean-cupe/linterpretabilite-de-lia-le-nouveau-defi-des-data-scientists/>

[4] :

<https://christophm.github.io/interpretable-ml-book/ice.html>

[5] :

https://www.wavestone.com/app/uploads/2019/09/Wavestone_Interpretabilite_Machine_learning.pdf

[6] :

https://www.institutdesactuaire.com/global/gene/link.php?doc_id=15779&fg=1

[7]:

<https://christophm.github.io/interpretable-ml-book/lime.html>

[8]:

<https://www.youtube.com/watch?v=PHMR6j8DvKk>

[9]:

<https://homes.cs.washington.edu/~marcotcr/blog/lime/>

[10]:

<https://christophm.github.io/interpretable-ml-book/shap.html>

[11]:

<https://towardsdatascience.com/explain-any-models-with-the-shap-values-use-the-kernelexplainer-79de9464897a>

[12]:

<https://www.kaggle.com/dansbecker/permutation-importance>

[13]:

<https://christophm.github.io/interpretable-ml-book/feature-importance.html>

[14]:

https://scikit-learn.org/stable/modules/permutation_importance.html

[15]

<https://www.nexmo.com/blog/2018/12/04/dog-breed-detector-using-machine-learning-dr>

[16]

<https://intelligence-artificielle.agency/transfer-learning-lapprentissage-par-transfert/>

[17]

<https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/>

[18]:

<https://towardsdatascience.com/interpretable-machine-learning-for-image-classification-with-lime-ea947e82ca13>

[19]:

https://compstat-lmu.github.io/iml_methods_limitations/ale-pdp.html

[20]:

<https://christophm.github.io/interpretable-ml-book/anchors.html>

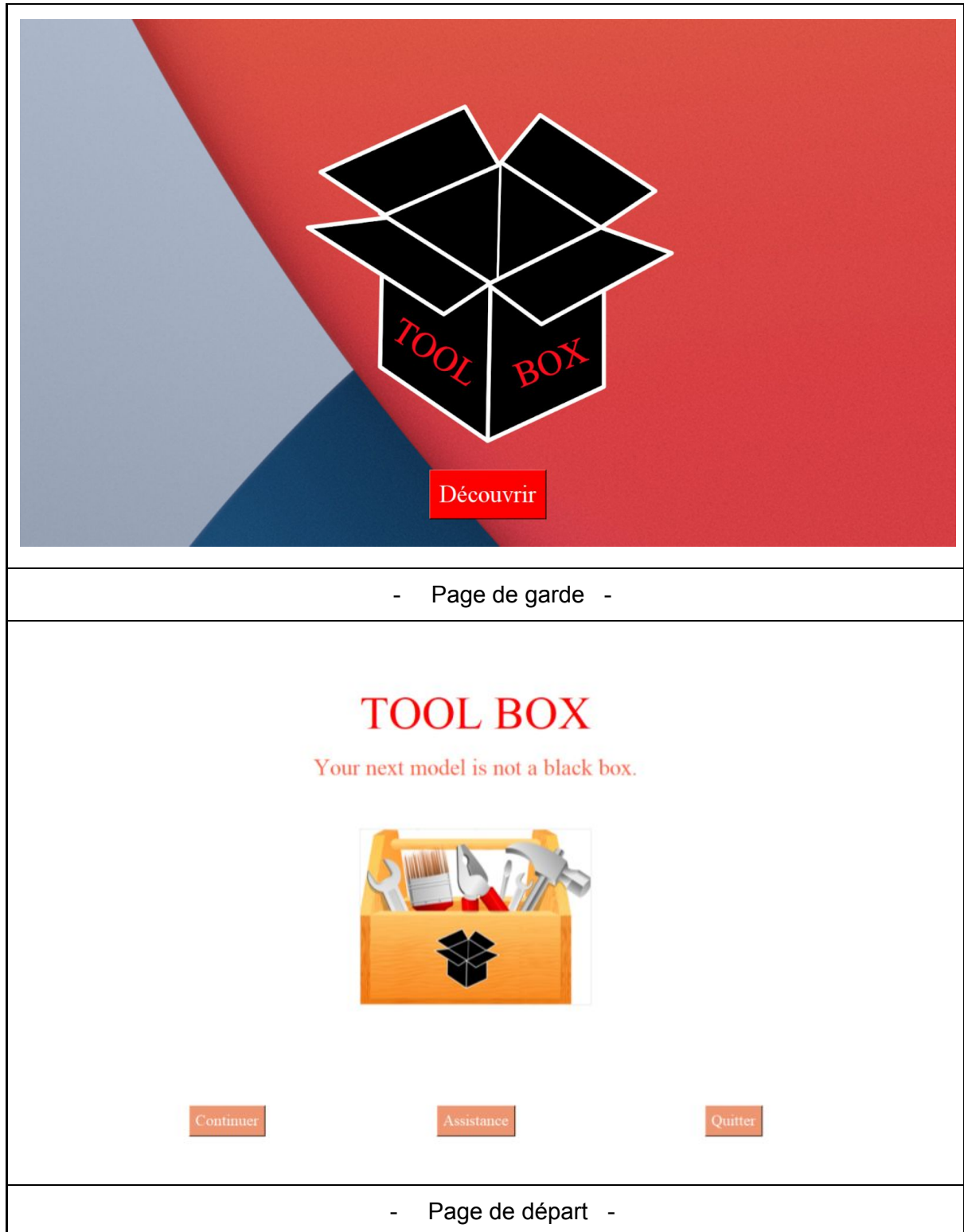
[21]:

<https://www.youtube.com/watch?v=5sYK0z--lqU>

[22]:

<https://christophm.github.io/interpretable-ml-book/ale.html>

Annexes



Découvrez la TOOL BOX

En savoir plus sur le machine learning interprétable.

Le développement et l'engouement rapides autour de l'IA ont conduit à prioriser la performance des algorithmes, alors qu'à présent la confluence de considérations réglementaires, éthiques, économiques et fonctionnelles font émerger une nouvelle dynamique dans laquelle l'interprétabilité pourrait devenir le nouveau critère d'évaluation des modèles. Cette considération nouvelle a conduit au développement d'un secteur de recherche porté sur le machine learning interprétable. L'interprétabilité explicite donc la façon dont la décision a été prise par l'algorithme (réponse au 'comment' ?).

Choisissez la bonne technique d'interprétabilité.

PDP montre l'effet marginal qu'une ou deux features ont sur le résultat prévu d'un modèle d'apprentissage machine.

ICE visualise la dépendance de la prédiction à une feature pour chaque instance séparément.

LIME va regarder l'incidence des variations des features dans le modèle d'apprentissage automatique sur les prédictions.

SHAP montre l'évolution de la prédiction à chaque ajout de variable.

Permutation feature permutation permet de mesurer l'impact de chaque Feature sur les prédictions.

Offres de la TOOL BOX

Afficher et sauvegarder les résultats d'une technique d'interprétabilité choisie sur votre modèle entraîné, votre dataset et vos données (entraînement et test) aux formats demandés. Les résultats sont sauvegardés en .png dans votre répertoire courant.

Extensions acceptées pour vos fichiers

Modèles .sav .pkl

Dataset et données .csv .pkl .hdf5 .h5 .npy

Retour

Quitter

- Page Assistance -

Interprétez et comprenez avec TOOL BOX

TOOL BOX permet de générer des explications de techniques de machine learning.

Destiné à recevoir des données tabulaires ou des images et des modèles parmi Random Forest, Neural Network, Regression linéaire et XGBoost, TOOL BOX vous propose une interprétabilité de vos résultats.

Nécessite l'installation de tkinter, pdpbox, eli5, shap, lime.


Continuer

Retour

Quitter

- Page Explication -

Données à interpréter



Tabulaire

Retour

- Choix du type de données à interpréter -

Choisir la technique d'interprétabilité

Sélectionnez une technique parmi les options suivantes.

ICE

Features importance

LIME

SHAP

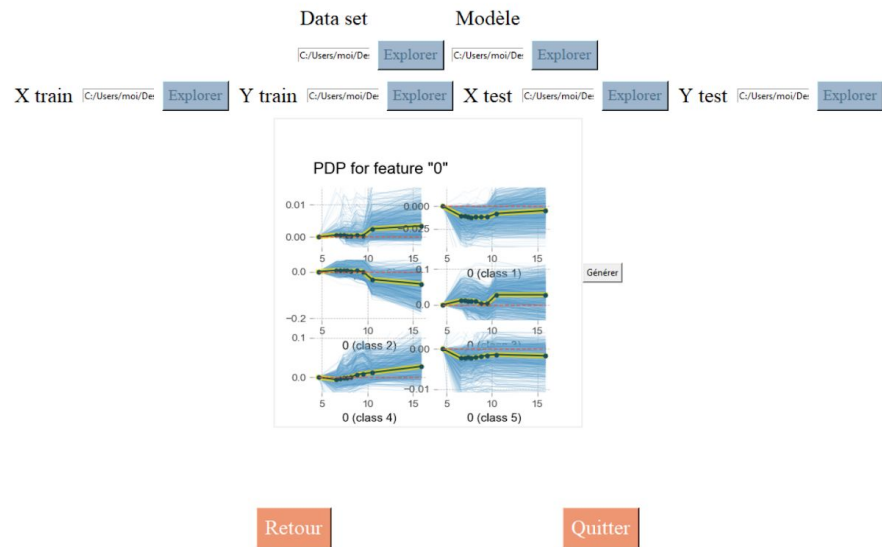
PDP

Retour

Quitter

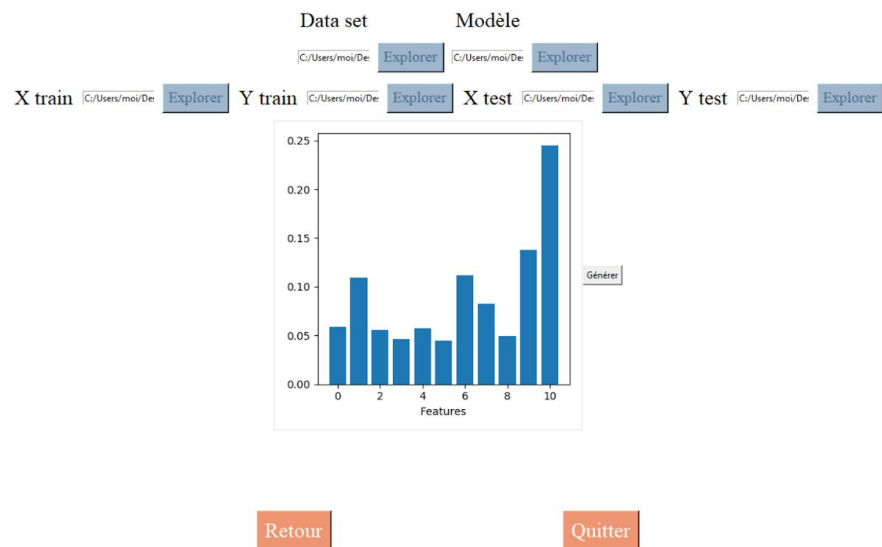
- Page données tabulaire -

ICE



Page ICE

Feature Importance



Page Feature Importance

LIME

Data set

X train C:/Users/moi/De [Explorer](#) Y train C:/Users/moi/De [Explorer](#) X test C:/Users/moi/De [Explorer](#) Y test C:/Users/moi/De [Explorer](#)

Modèle

C:/Users/moi/De [Explorer](#) C:/Users/moi/De [Explorer](#)

Local explanation

9.50 < alcohol <= 10.20 : ■

0.62 < sulphates <= 0.72 : ■

0.52 < volatile acidity <= 0.64 : ■

1.00 < density <= 1.00 : ■

chlorides > 0.09 : ■

37.00 < total sulfur dioxide <= 61.50 : ■

13.00 < free sulfur dioxide <= 21.00 : ■

citric acid <= 0.09 : ■

7.10 < fixed acidity <= 7.90 : ■

2.20 < residual sugar <= 2.60 : ■

[Générer](#)

[Retour](#)
[Quitter](#)

Page LIME

SHAP

Data set

X train C:/Users/moi/De [Explorer](#) Y train C:/Users/moi/De [Explorer](#) X test C:/Users/moi/De [Explorer](#) Y test C:/Users/moi/De [Explorer](#)

Modèle

C:/Users/moi/De [Explorer](#) C:/Users/moi/De [Explorer](#)

[Retour](#)
[Quitter](#)

Page SHAP

