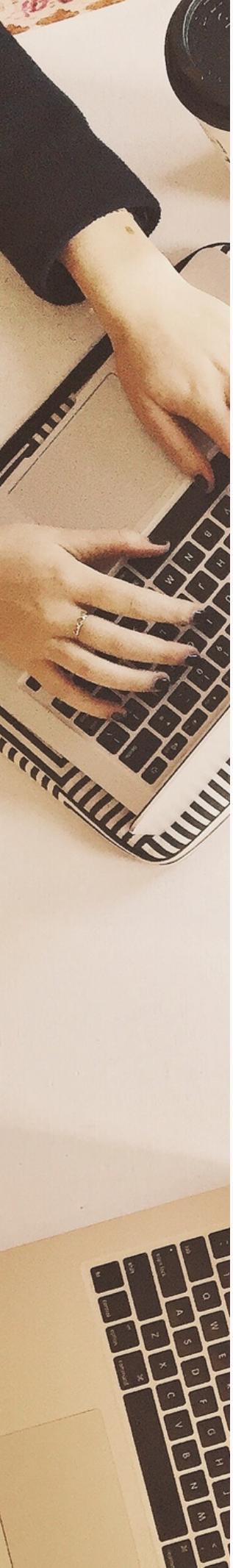




Les accidents de la Route en France

Projet Fil Rouge

DataScientest - Janvier 2023



SOMMAIRE

01

Contexte et Objectif

02

Partie I :
Exploration, Visualisation et
Pré-Processing des données

03

Partie II :
Modélisation et
Interprétation des prédictions

04

Conclusion et Perspectives



Contexte et Objectifs

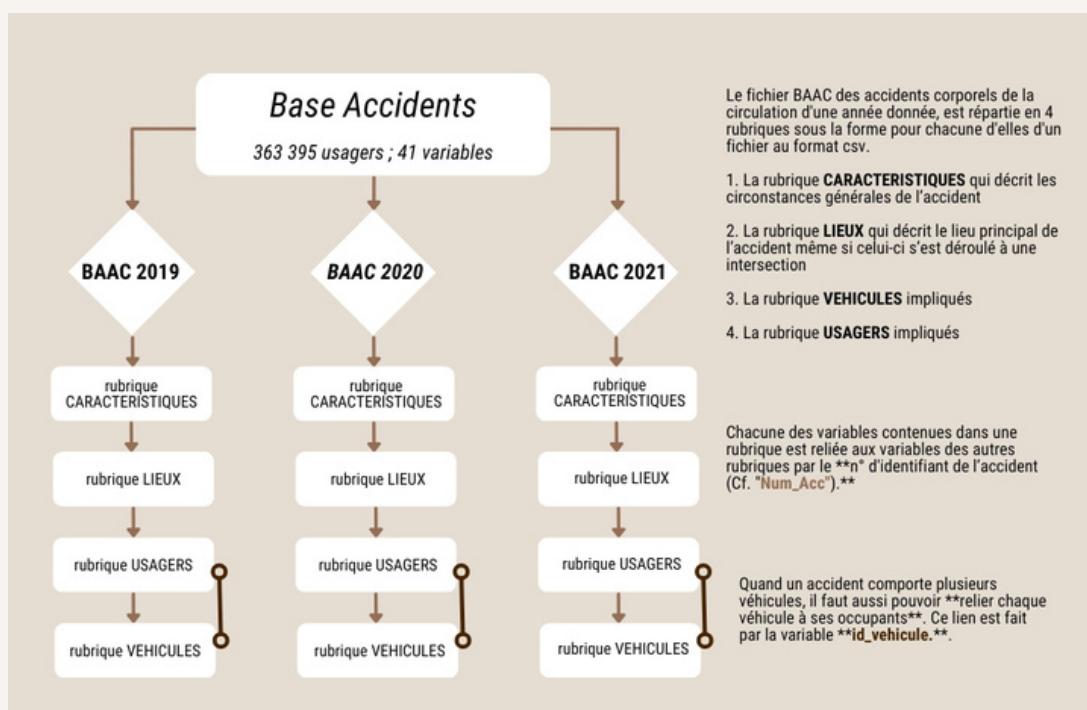
L' Observatoire national interministériel de la sécurité routière met à disposition chaque année depuis 2005, des bases de données des accidents corporels de la circulation routière.

Pour chaque accident corporel (soit un accident survenu sur une voie ouverte à la circulation publique, impliquant au moins un véhicule et ayant fait au moins une victime ayant nécessité des soins), des saisies d'information décrivant l'accident sont effectuées par l'unité des forces de l'ordre (police, gendarmerie, etc.) qui est intervenue sur le lieu de l'accident.

Ces saisies sont rassemblées dans une fiche intitulée bulletin d'analyse des accidents corporels. L'ensemble de ces fiches constitue le fichier national des accidents corporels de la circulation dit « Fichier BAAC » administré par l'Observatoire national interministériel de la sécurité routière "ONISR".

Les bases de données, extraites du fichier BAAC, répertorient l'**integralité des accidents corporels de la circulation, intervenus durant une année précise en France métropolitaine, dans les départements d'Outre-mer (Guadeloupe, Guyane, Martinique, La Réunion et Mayotte depuis 2012) et dans les autres territoires d'outre-mer** (Saint-Pierre-et-Miquelon, Saint-Barthélemy, Saint-Martin, Wallis-et-Futuna, Polynésie française et Nouvelle-Calédonie; disponible qu'à partir de 2019 dans l'open data) avec une description simplifiée.

Cela comprend des informations de localisation de l'accident, telles que renseignées ainsi que des informations concernant les caractéristiques de l'accident et son lieu, les véhicules impliqués et leurs victimes.



Objectif du Projet

L' objectif de ce projet est de prédire la gravité de l'ensemble des accidents routiers en France intervenus entre 2019 et 2021.



Les auteurs du rapport

- Anicet Arthur KOUASSI
- Isabelle CONTANT
- Omar DIANKHA
- Idelphonse GBOHOUNME

Notre repository Github : [cliquez ici](#)



Les accidents de la Route en France

Projet Fil Rouge - Partie I

**Exploration, Data Visualisation et
Pré-Processing des données**



Sommaire Partie I

01

Exploration des données

02

La variable cible : la gravité des accident

03

Pré-Processing des données

04

Conclusion Partie I

Objectif Partie I

L'objectif de cette première partie du rapport est d'effectuer l'exploration, la visualisation et la pré-traitement de ces bases de données.



01. Exploration des données

Bien que l'on puisse penser qu'il suffit d'un grand nombre de données pour avoir un algorithme performant, les données dont nous disposons sont souvent non adaptées. Il faut donc les comprendre et les traiter préalablement pour pouvoir ensuite les utiliser : c'est l'étape d'exploration et de visualisation des données.

01. Exploration des données

En effet, des erreurs d'acquisition liées à des fautes humaines ou techniques peuvent corrompre notre dataset et biaiser l'entraînement.

Parmi ces erreurs, nous pouvons citer des informations incomplètes, des valeurs manquantes ou erronées ou encore des bruits parasites liés à l'acquisition de la donnée.

Il est donc souvent indispensable d'établir une stratégie de pré-traitement des données à partir de nos données brutes pour arriver à des données exploitables qui nous donneront un modèle plus performant.

La particularité de notre jeu de données est que nous avons 4 fichiers par année avec les mêmes variables.

Afin d'effectuer les mêmes traitements pour les 3 années, nous avons décidé d'analyser minutieusement chaque variable de chaque rubrique de l'année 2021, puis d'appliquer exactement les mêmes transformations pour les rubriques des années 2019 et 2020.

En utilisant cette méthodologie de travail, nous nous sommes assurés d'obtenir un jeu de données par année avec les mêmes variables et les mêmes transformations.

Traitement effectué sur chacune des 4 rubriques de l'année 2021

Traitement des Observations :

- Suppression des observations "vides"
- Suppression des observations dupliquées

Traitement des Variables :

- Suppression des variables avec un taux élevé de valeurs manquantes
- Suppression des variables non pertinentes

Traitement des variables qualitatives :

- Regroupement des modalités à faible effectif
- Suppression des variables non discriminantes

Traitement des Variables numériques :

- Analyse de la distribution et test de Normalité
- Transformation en variable de classe

Fusion des 4 fichiers par leur identifiant commun

=> 3 bases complètes annuelles

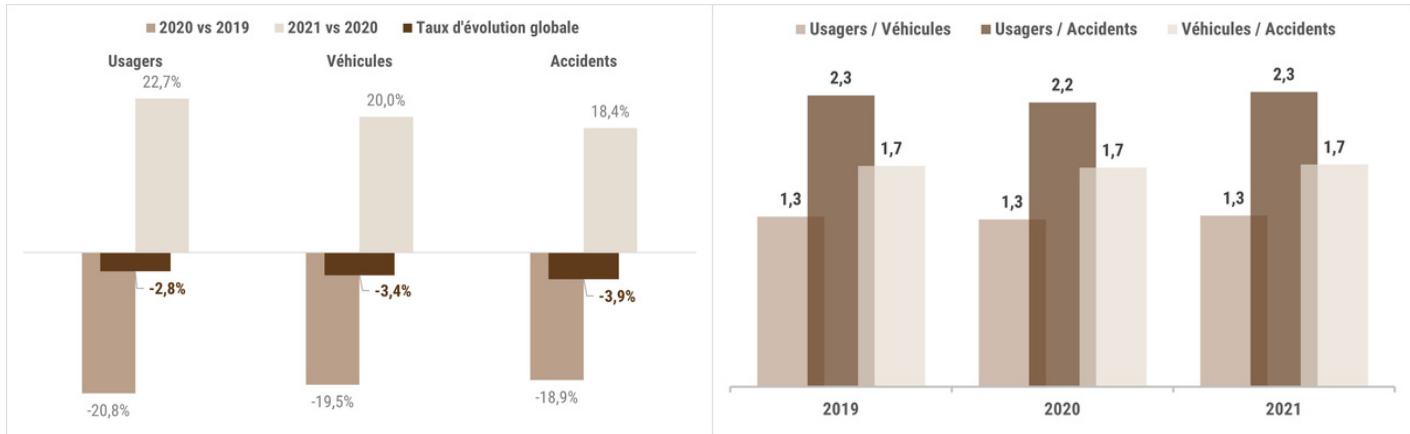
Concaténation de ces 3 bases : le fichier d'analyse

Les grandes tendances :

En 2019, il y a eu 132 876 usagers et 100 710 véhicules impliqués dans 58 840 accidents.

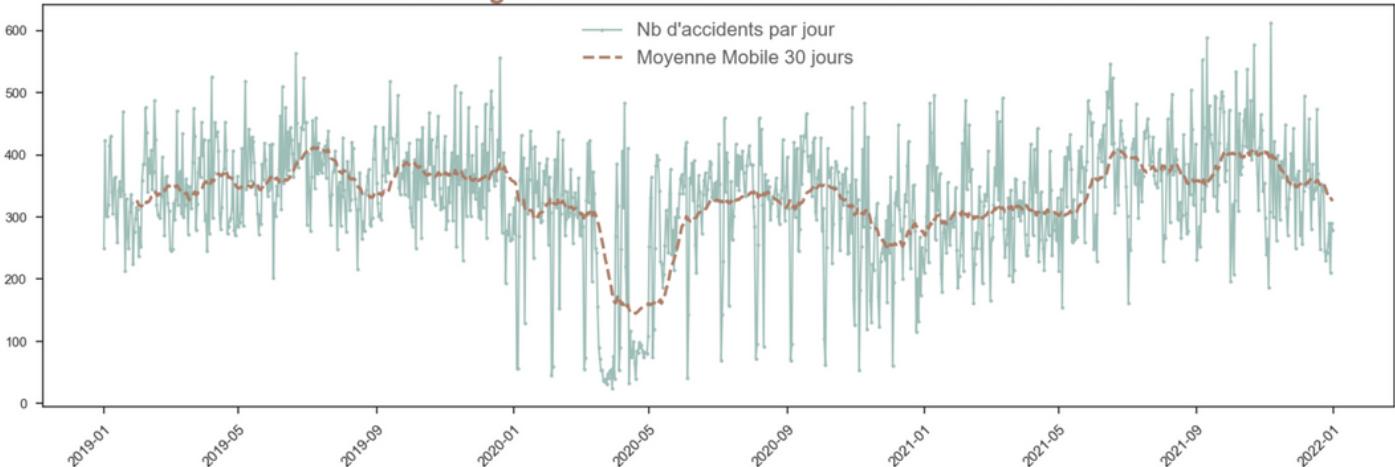
En 2020, il y a eu 105 232 usagers et 81 066 véhicules impliqués dans 47 744 accidents.

En 2021, il y a eu 129 153 usagers et 97 315 véhicules impliqués dans 56 518 accidents.



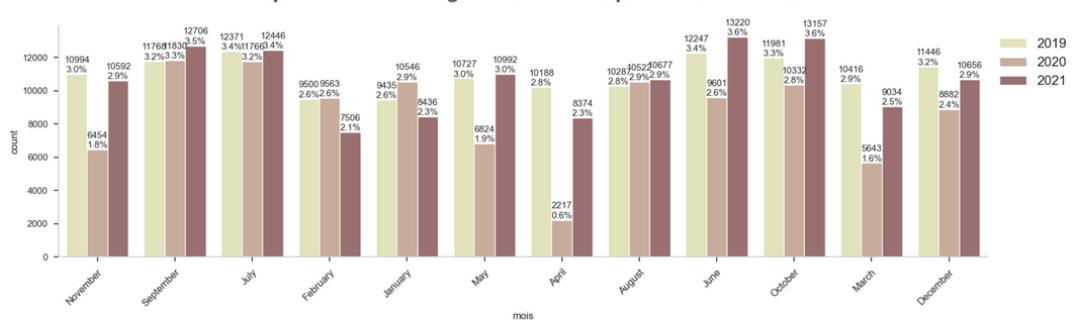
Même si les évolutions annuelles sont très erratiques, on constate une légère diminution du taux de croissance global de 4% tandis que le nombre moyen d'usagers par accident et le nombre moyen de véhicules par accident, restent stables.

Nombre d'usagers accidentés du 01 Jan 2019 au 31 Dec 2021

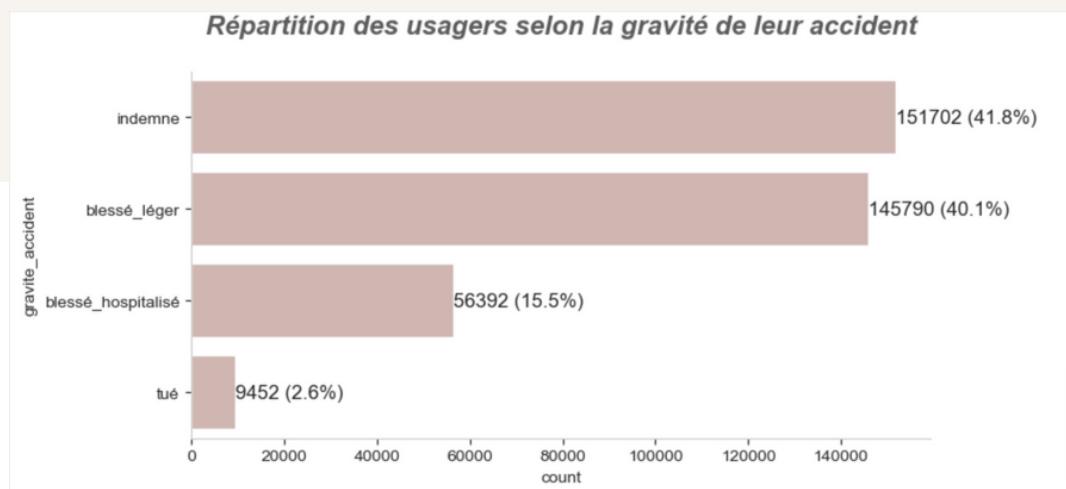


L'année 2020 est en effet très différente des 2 autres années. Alors qu'en 2019 et 2021, le nombre d'accidents est réparti de manière homogène sur les 12 mois de l'année, on constate que ce n'est pas le cas en 2020 ; il y a eu nettement moins d'accidents en avril, mai et novembre, en raison des confinements liés au Covid-19.

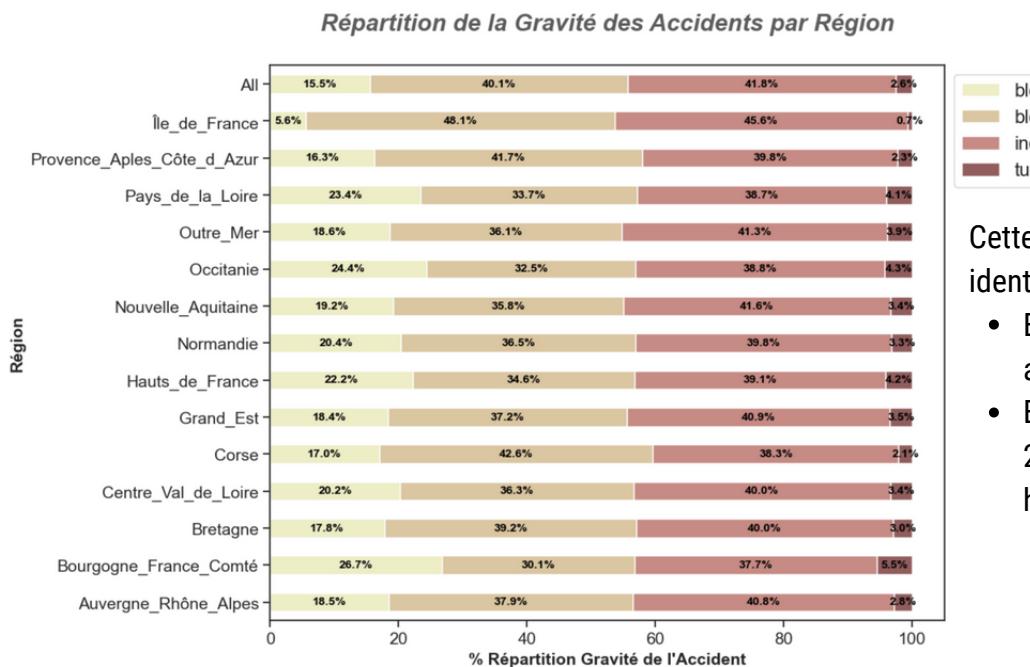
Répartition des usagers accidentés par mois et année



02. La variable cible : la gravité des accident

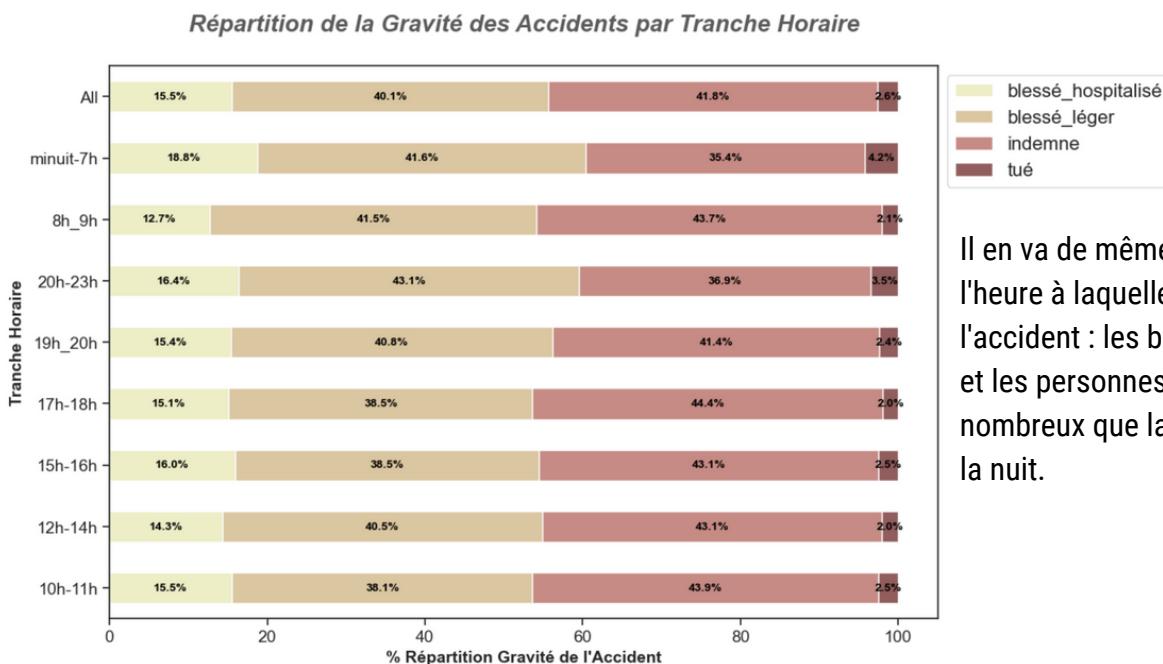


Fort heureusement, Près de 82% des usagers impliqués dans un accident routier sont indemnes ou ont de légères blessures. En revanche, près de 3% d'entre eux sont tués (environ 3 150 personnes par an)!



Cette répartition nationale n'est pas identique selon les régions :

- En Ile de France 94% des accidents sont sans gravité
- En Bourgogne-Franche-Comté, 27% des accidentés doivent être hospitalisés et 5,5% sont tués.



Il en va de même au regard de l'heure à laquelle s'est passé l'accident : les blessés hospitalisés et les personnes tuées sont plus nombreux que la moyenne nationale la nuit.

03. Pre-Processing

Après avoir nettoyé et décrit nos données brutes, puis déterminé notre variable cible, nous obtenons un dataset de 363 336 usagers impliqués dans un accident routier caractérisés par 40 variables qualitatives.

Nous devons maintenant les transformer en données utilisables par les différents algorithmes de Machine Learning pour les 3 raisons principales suivantes :

- La majorité des algorithmes ont des contraintes sur le format des données en entrée. Cela peut concerter leur type, par exemple uniquement des variables numériques, ou des contraintes sur leur format, comme des réels entre 0 et 1.
- Préparer les données permet de grandement améliorer les résultats des algorithmes, en extrayant ou en créant des colonnes plus adaptées au problème
- Ce n'est pas parce que la fonctionnalité existe dans notre jeu de données qu'elle est pertinente pour notre modèle et que nous devons l'utiliser. En effet, "**Les bons sous-ensembles de caractéristiques contiennent des caractéristiques hautement corrélées avec la cible, mais non corrélées les unes aux autres**".

04.1. Quelles sont les caractéristiques corrélées à la cible ?

L'objectif ici est de sélectionner les variables qui ont un lien statistique avec la variable cible.

Pour cela, nous avons utilisé le **Test du χ^2 d'indépendance**.

Le test d'indépendance du χ^2 est utilisé pour **déterminer s'il existe une relation significative entre deux variables catégorielles (nominales)**.

Cela signifie que le test d'indépendance du χ^2 est un test d'hypothèse avec 2 hypothèses présentes ; l'hypothèse nulle et l'hypothèse alternative.

- **Hypothèse nulle (H_0) : Il n'y a pas de relation entre les variables**
- **Hypothèse alternative (H_1) : Il existe une relation entre les variables**

Comme tout test statistique, nous le testons par rapport à notre valeur p choisie (souvent elle est de 0,05).

Si la valeur de p est significative ($p \leq 0,05$), nous pouvons rejeter l'hypothèse nulle et affirmer que les résultats soutiennent l'hypothèse alternative.

Or, toutes nos p-values sont nulles : Il existe une relation significative entre la variable cible et les 40 variables qualitatives explicatives dont nous disposons.

Cependant, comme nous avons plusieurs modalités dans chacune de nos variables catégorielles, nous ne sommes pas en mesure de dire facilement quelle est celle qui est responsable de la relation (si la table χ^2 est supérieure à 2×2).

Pour identifier quelle(s) modalité(s) est(sont) responsable(s), nous avons besoin d'un test post hoc. Mais, il faut d'abord quantifier nos variables catégorielles.

03. Pre-Processing

04.2. Transformation des caractéristiques

Cette étape de prétraitement regroupe les changements effectués sur la structure même de la donnée. Ces transformations sont liées aux définitions mathématiques des algorithmes et à la manière dont ceux-ci traitent les données, de manière à optimiser les performances.

Nous avons essentiellement utilisé :

- La **discrétisation de variables continues** (à l'aide du découpage en intervalles) qui permet d'abaisser le nombre de modalités d'une variable et d'en supprimer les éventuelles valeurs aberrantes.
- La "**dummification**" des variables catégorielles nominales ou la variable initiale est remplacée par un ensemble de nouvelles variables dites fictives (ou indicatrices) prenant les valeurs 0 ou 1 pour indiquer l'absence ou la présence de la modalité.

Base de données après Transformation

Nombre d'usagers ayant eu un accident de la

route entre 2019 et 2021 : 363 336

Nombre de variables explicatives : **126**

04.3. Réduction du nombre de caractéristiques

Lors de la construction d'un modèle prédictif, nous avons souvent de nombreuses fonctionnalités ou variables dans notre ensemble de données qui peuvent être utilisées pour former notre modèle.

Cependant, ce n'est pas parce que la fonctionnalité existe dans notre jeu de données qu'elle est pertinente pour notre modèle et que nous devons l'utiliser. En effet, "**Les bons sous-ensembles de caractéristiques contiennent des caractéristiques hautement corrélées avec la cible, mais non corrélées les unes aux autres**".

Alors, comment savons nous quelles fonctionnalités utiliser pour notre modèle ?

C'est là qu'intervient la **sélection des caractéristiques**.

La sélection des caractéristiques est simplement un processus qui réduit le nombre de variables d'entrée, afin de ne conserver que les plus importantes.

Il existe **trois catégories de méthodes de sélection de fonctionnalités**, en fonction de la manière dont elles interagissent avec la variable à prédire, à savoir les **méthodes de filtrage, d'encapsulation et intégrées**. Nous les avons utilisées toutes les 3.

04.3.1. Test du χ^2 d'indépendance avec les valeurs p ajustées de Bonferroni

La correction de Bonferroni consiste à diviser le niveau alpha souhaité par le nombre de comparaisons pour utiliser le nombre ainsi calculé comme valeur p afin de déterminer la signification. Ainsi, par exemple, avec alpha fixé à 0,05 et trois comparaisons, la valeur p-ajusté requise pour la signification serait de $0,05/3 = 0,0167$.

03. Pre-Processing

En utilisant la valeur P ajustée, nous pouvons alors tester tous les résultats précédemment significatifs pour voir **quelle modalité est responsable de la création d'une relation significative entre la variable cible et nos 126 variables indicatrices.**

Or, toutes nos p-values sont nulles : Il existe une relation significative entre la variable cible et les 126 modalités des 40 variables qualitatives explicatives dont nous disposons.

04.3.2. Analyse des corrélations

L'analyse des corrélations évalue des sous-ensembles de caractéristiques sur la base de l'hypothèse suivante : "**Les bons sous-ensembles de caractéristiques contiennent des caractéristiques hautement corrélées avec la cible, mais non corrélées les unes aux autres**".

Nous avons déjà constaté que toutes nos variables indicatrices sont hautement corrélées avec notre variable cible grâce au Test du χ^2 d'indépendance et les valeurs p ajustées de Bonferroni.

Nous allons maintenant effectuer des **tests d'indépendance entre les variables explicatives par paire afin d'exclure celles potentiellement non porteuses d'information.**

Pour cela nous allons analyser les **coefficients de corrélations de Spearman** pour ne pas évaluer que les relations linéaires.

La corrélation de Spearman entre deux variables est égale à la corrélation de Pearson entre les valeurs de rang de ces deux variables ; alors que la corrélation de Pearson évalue les relations linéaires, la corrélation de Spearman évalue les relations monotones (qu'elles soient linéaires ou non). Le coefficient de corrélation sur les rangs (Rho de Spearman) s'interprète de la même manière qu'un coefficient de corrélation de Pearson : une valeur positive (maximum = +1) indique une variation simultanée dans le même sens, une valeur négative (minimum = -1) une variation simultanée en sens inverse.

Mais nous obtenons un tableau de coefficients de Spearman de 101 x 101 (puisque nous avons pu éliminer 25 caractéristiques quasi-constantes qui montrent la même valeur pour la grande majorité des observations de l'ensemble de données).

Après avoir supprimer les p_value des variables avec elles-mêmes et les pairs identiques, il reste 4207 pairs de variables non indépendantes ($p\text{-value} < 0.05$) et 842 pairs de variables indépendantes !

On ne peut donc pas utiliser cette méthode pour sélectionner les variables pertinentes pour la réalisation de nos objectifs, mais elle nous montre le potentiel de réduction de nos variables explicatives puisqu'il n'y a que 842 pairs de variables indépendantes !

03. Pre-Processing

04.3.3. Les méthodes Embedded

Les méthodes Embedded intègrent la sélection de fonctionnalités à la construction de l'algorithme d'apprentissage automatique.

Les méthodes intégrées **effectuent la sélection des fonctionnalités lors de l'apprentissage du classifieur ou des régresseurs**. Ces méthodes sont donc embarquées dans l'algorithme.

Les méthodes embarquées ont les avantages à la fois des Méthodes Wrapper (que nous n'utiliserons pas ici car elles demandent beaucoup de temps et de ressources) et Filter. Elles incluent l'interaction des caractéristiques avec le modèle de classifieur ou le régresseur modèle tout comme les méthodes Wrapper, et comme les méthodes de filtrage, elles sont beaucoup moins gourmandes en calcul. De plus, les méthodes embarquées sont capables de détecter l'interaction entre les variables car elles évaluent l'ensemble des données en même temps et trouvent le sous-ensemble de fonctionnalités qui convient à l'algorithme en cours de formation.

Cela signifie que les méthodes intégrées sont généralement la méthode de choix au moment de la sélection des fonctionnalités.

Les méthodes embarquées font partie de la formation de l'algorithme d'apprentissage automatique. Ainsi, les étapes typiques impliquent la formation d'un algorithme d'apprentissage automatique en utilisant toutes les fonctionnalités, puis en dérivant l'importance de ces caractéristiques selon l'algorithme utilisé, et enfin en supprimant les fonctionnalités non importantes en suivant certains critères qui dépendront de l'algorithme lui-même.

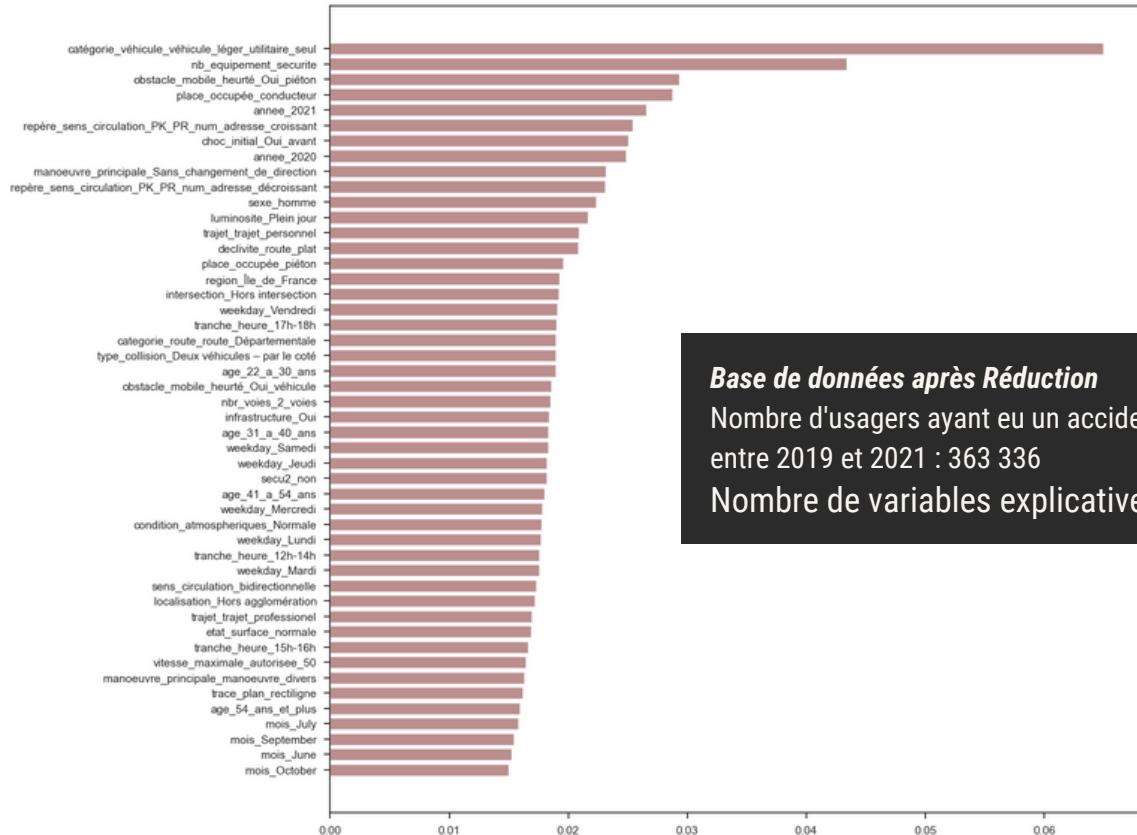
Un exemple de méthodes embarquées est la régularisation LASSO (pour la régression). Et dans une certaine mesure, nous pourrions également utiliser les coefficients de régression des modèles linéaires pour évaluer l'importance des différentes caractéristiques de l'ensemble de données et décider lesquelles conserver et lesquelles exclure.

Les arbres et les forêts aléatoires sont également très utiles pour sélectionner des fonctionnalités.

En résumé, les méthodes embarquées donnent une **meilleure précision prédictive** que les méthodes de filtrage car elles sont rapides et moins coûteuses en calcul que les méthodes wrapper. Elles ont également **tendance à trouver un très bon sous-ensemble de fonctionnalités** pour l'algorithme donné. En revanche, nous devons garder à l'esprit les caractéristiques des différents algorithmes pour comprendre l'importance dérivée de ce que nous pouvons faire avec eux.

Le processus d'ingénierie des caractéristiques consiste à **sélectionner les caractéristiques minimales requises pour produire un modèle valide, car plus un modèle contient de caractéristiques, plus il est complexe** (et plus les données sont rares), **donc plus le modèle est sensible aux erreurs dues à la variance**.

Une approche courante pour éliminer les caractéristiques consiste à décrire leur importance relative pour un modèle, puis à éliminer les caractéristiques faibles ou les combinaisons de caractéristiques et à réévaluer pour voir si le modèle se comporte mieux lors de la validation croisée, c'est la **méthode RFE pour élimination récursive des fonctionnalités**.



Base de données après Réduction

Nombre d'usagers ayant eu un accident de la route entre 2019 et 2021 : 363 336

Nombre de variables explicatives : 48

04. Conclusion Partie I

Notre jeu de données est maintenant prêt pour la phase suivante de modélisation.

Nous l'avons décomposé en 2 sous-ensembles : 70% pour entraîner nos modèles de prédiction et 30% pour les tester.

Nous n'avons pas réalisé de standardisation des données puisque nous ne disposons que de variables explicatives dichotomiques.

Nous avons choisi, dans un premier temps, d'entraîner des modèles de Classification binaire en transformant la variable cible :

- "blessés légers", "blessés hospitalisés", "tués"
- Indemnités

Dans une seconde étape, nous utiliserons des modèles de Classification multi-classes avec la variable cible en 4 classes.



Les accidents de la Route en France

Projet Fil Rouge - Partie II

Modélisation et
Interprétation des prédictions



Sommaire Partie II

01

Classification Binaire

02

**Classification Multi-
Classes**

03

Modélisation CatBoost

04

Conclusion Partie II

Objectif Partie II

L'objectif de cette seconde partie est d'effectuer la modélisation de la gravité des accidents de la route en France et d'interpréter les prédictions obtenues afin de répondre à la question :

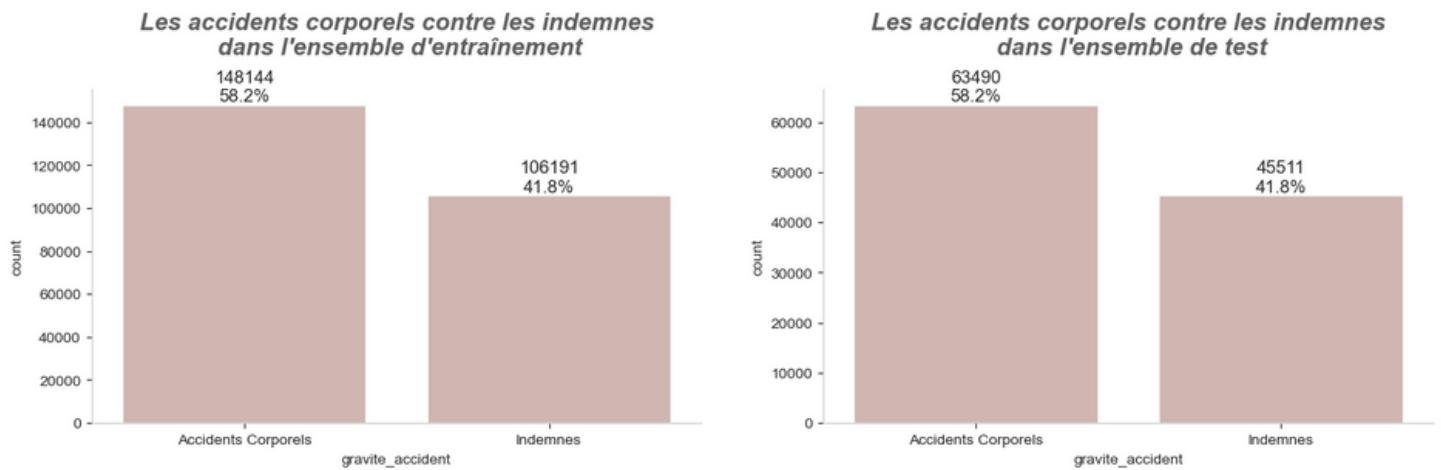
"Pourquoi le modèle a-t-il prévu cette valeur ?".



01. Classification Binaire

Notre étude a pour objectif de modéliser la gravité des accidents de la route en France en se concentrant sur quatre classes d'usagers : indemnes, blessés légers, hospitalisés et tués.

Étant donné que la proportion d'usagers tués est faible (2,6%), nous avons choisi de commencer par une classification binaire qui consiste à **modéliser les usagers indemnes (41,8%) contre les trois autres classes (classe 1 - 58,2%)** afin d'utiliser les modèles de classification binaire sur des données équilibrées.



01.1. Choix du modèle le mieux adapté à la prédiction de la variable cible binaire

Pour sélectionner les modèles les plus adaptés, nous avons utilisé les modèles de base tels que le DummyClassifier, la Régression Logistique, le Random Forest Classifier, le Bagging Classifier, l'AdaBoost Classifier, le Decision Tree Classifier, l'Extra Trees Classifier, le K-Neighbors Classifier, le XGBoost Classifier et le LightGBM Classifier.

Nous avons évalué les performances de chaque modèle en calculant les métriques de précision telles que l'exactitude (Accuracy), le score F1 (F1-score) et l'aire sous la courbe ROC (Roc-Auc) sur l'ensemble d'entraînement et l'ensemble de test.

L'exactitude a été utilisée pour mesurer la précision globale du modèle, le score F1 pour prendre en compte à la fois la précision et le rappel, tandis que l'aire sous la courbe ROC a été utilisée pour évaluer la capacité du modèle à discriminer les classes.

Nous avons également pris en compte le temps d'exécution de chaque modèle pour tenir compte des contraintes de temps réel.

01. Classification Binaire (suite)

Model	Train Accuracy	Test Accuracy	Train F1_Score	Test F1_Score	Train ROC_AUC	Test ROC_AUC	Train Time	Test Time
DecisionTreeClassifier	1.00	0.69	1.00	0.70	1.00	0.69	2.79	2.79
ExtraTreesClassifier	1.00	0.77	1.00	0.77	1.00	0.76	55.99	52.07
RandomForestClassifier	1.00	0.77	1.00	0.77	1.00	0.77	40.69	37.25
BaggingClassifier	0.98	0.75	0.98	0.75	0.98	0.75	17.01	16.14
KNeighborsClassifier	0.81	0.71	0.81	0.71	0.80	0.70	968.70	417.82
XGBClassifier	0.79	0.78	0.79	0.78	0.78	0.77	10.60	9.35
LGBMClassifier	0.78	0.78	0.78	0.78	0.77	0.77	2.62	2.07
LogisticRegression	0.75	0.75	0.75	0.75	0.74	0.74	1.43	1.07
AdaBoostClassifier	0.75	0.75	0.75	0.75	0.74	0.74	13.84	12.70
DummyClassifier	0.58	0.58	0.43	0.43	0.50	0.50	0.68	0.51

Le tableau des résultats montre les performances de chacun des modèles sur l'ensemble d'entraînement et l'ensemble de test.

Nous avons pu observer que les modèles tels que Random Forest, Extra Trees et Bagging Classifiers ont tendance à sur-apprendre sur l'ensemble d'entraînement, ce qui a entraîné une baisse des performances sur l'ensemble de test par rapport à l'ensemble d'entraînement. D'autre part, les modèles tels que K-Neighbors Classifier ont montré des performances très faibles en terme de temps d'exécution.

En revanche, **les modèles tels que LogisticRegression, AdaBoost, XGBoost et LightGBM ont montré des performances stables et élevées sur l'ensemble d'entraînement et l'ensemble de test.**

Encadré n°1 : Qu'est-ce que l'Overfitting :

Un modèle qui correspond bien à l'ensemble d'apprentissage mais qui est mal adapté à l'ensemble de test est dit sur-ajusté à l'ensemble d'apprentissage et un modèle qui correspond mal aux deux ensembles est dit sous-ajusté.

En d'autres termes, le surajustement signifie que le modèle d'apprentissage automatique est capable de trop bien modéliser l'ensemble d'apprentissage. Le surajustement se produit lorsque le modèle s'adapte non seulement au signal qui est utile pour les caractéristiques, mais commence également à mémoriser les fluctuations aléatoires, les anomalies et le bruit qui peuvent exister uniquement dans l'ensemble de données d'apprentissage et non dans la distribution plus large.

Certains types de modèles peuvent être plus sujets au surajustement que d'autres, comme les arbres de décision ou KNN.

Encadré n°2 : Définition des Métriques utilisées pour la sélection des modèles

- **Exactitude (Accuracy)** : Il s'agit de la proportion de prédictions correctes parmi toutes les prédictions effectuées. Cette métrique est facile à interpréter et peut être utile pour des problèmes de classification binaire équilibrés. Cependant, elle peut être trompeuse pour des problèmes déséquilibrés, car elle peut donner des résultats élevés même si le modèle ne prédit correctement qu'une classe. De plus, elle ne prend pas en compte le coût des erreurs de prédiction.
- **Score F1 (F1-score)** : Il s'agit de la moyenne harmonique de la précision et du rappel. Cette métrique est utile pour les problèmes de classification binaire car elle prend en compte à la fois la précision et le rappel pour calculer un score global.
- **Aire sous la courbe ROC (Roc-Auc)** : Il s'agit de la mesure de la capacité d'un modèle à distinguer les classes positives des classes négatives en fonction d'un seuil de classification donné. Cette métrique est utile pour des problèmes de classification binaire et peut être utilisée pour sélectionner un seuil de classification optimal pour un modèle.
- **Temps d'exécution** : Il s'agit du temps nécessaire pour effectuer les prédictions sur les données d'entraînement ou de test. Cette métrique est importante pour prendre en compte les contraintes de temps réel, car elle peut affecter la performance et l'utilité d'un modèle. Cependant, il peut varier en fonction du matériel et de la configuration de l'environnement informatique.

Encadré n°3 : Synthèse des modèles utilisés avec leurs avantages et leurs inconvénients

DummyClassifier : Ce modèle est utilisé comme référence pour évaluer la performance des autres modèles. Il effectue des prédictions aléatoires ou utilise la stratégie la plus fréquente dans les données d'entraînement. Cela peut être utile pour comparer les performances des autres modèles par rapport à une référence aléatoire. Cependant, il ne peut pas être utilisé pour de vraies prédictions.

LogisticRegression : Ce modèle est simple à mettre en œuvre et peut être interprété facilement. Il est également efficace pour les problèmes de classification binaire. Cependant, il peut ne pas être optimal pour des problèmes plus complexes.

RandomForestClassifier : Ce modèle utilise plusieurs arbres de décision pour améliorer la précision des prédictions. Il peut être utilisé pour des problèmes de classification binaire et multi-classes. Cependant, il peut être difficile à interpréter.

BaggingClassifier : Ce modèle utilise plusieurs modèles d'ensemble pour améliorer la précision des prédictions. Il peut être utilisé pour des problèmes de classification binaire et multi-classes. Cependant, il peut également être difficile à interpréter.

AdaBoostClassifier : Ce modèle utilise plusieurs modèles d'ensemble pondérés pour améliorer la précision des prédictions. Il est efficace pour les problèmes de classification binaire et multi-classes. Cependant, il peut être sensible aux valeurs aberrantes.

DecisionTreeClassifier : Ce modèle utilise des arbres de décision pour effectuer des prédictions. Il est facile à interpréter et à visualiser. Cependant, il peut être sensible aux valeurs aberrantes et au sur-ajustement des données.

ExtraTreesClassifier : Ce modèle est similaire à RandomForestClassifier, mais utilise des arbres de décision supplémentaires pour améliorer la précision des prédictions. Il peut être utilisé pour des problèmes de classification binaire et multi-classes. Cependant, il peut être difficile à interpréter.

KNeighborsClassifier : Ce modèle utilise une méthode de classification basée sur la proximité des données. Il peut être utilisé pour des problèmes de classification binaire et multi-classes. Cependant, il peut être sensible aux valeurs aberrantes et nécessite une grande quantité de données pour être efficace.

XGBClassifier : Ce modèle utilise un algorithme de gradient boosting pour améliorer la précision des prédictions. Il est efficace pour les problèmes de classification binaire et multi-classes. Cependant, il peut être lent à entraîner.

LGBMClassifier : Ce modèle utilise un algorithme de gradient boosting plus rapide et plus efficace que XGBClassifier. Il est également efficace pour les problèmes de classification binaire et multi-classes. Cependant, il peut être sensible aux valeurs aberrantes.

01. Classification Binaire (suite)

01.2. Optimisation des 4 "meilleurs" modèles

Nous avons choisi d'optimiser les hyperparamètres des quatre modèles sélectionnés : la Régression Logistique, AdaBoost, XGBoost et LGBMClassifier.

Pour ce faire, nous avons utilisé trois techniques d'optimisation : GridSearchCV pour la Régression Logistique et AdaBoost, Hyperopt pour XGBoost et Optuna pour LGBMClassifier.

GridSearchCV est une méthode exhaustive qui explore tous les hyperparamètres possibles, ce qui peut être coûteux en termes de temps de traitement. Hyperopt et Optuna, en revanche, utilisent une approche basée sur la recherche d'hyperparamètres à l'aide d'algorithmes d'optimisation, ce qui peut être plus rapide et plus efficace pour les problèmes de modélisation complexes.

Après avoir optimisé les hyperparamètres, nous avons évalué les performances de chaque modèle en utilisant les mêmes métriques que précédemment : Accuracy, Score F1, Roc-Auc et Recall.

Les résultats ont montré que la Régression Logistique et AdaBoost n'ont pas bénéficié de l'optimisation des hyperparamètres et ont maintenu des performances similaires à celles obtenues avec les paramètres par défaut.

En revanche, l'optimisation des hyperparamètres pour XGBoost et LGBMClassifier a entraîné une nette amélioration des performances de ces deux modèles. En effet, le score F1 et le Roc-Auc ont été significativement améliorés pour ces deux modèles.

Pour départager ces 2 modèles, nous avons tenu compte du Recall.

Model	Test Accuracy	Test Recall	Test F1_Score	Test ROC_AUC
XGBClassifier	0.78		0.78	0.77
<i>XGBClassifier Optimisé</i>	0.78	0.83	0.82	0.86
LGBMClassifier	0.78		0.78	0.77
<i>LGBMClassifier optimisé</i>	0.78	0.82	0.81	0.86
LogisticRegression	0.75		0.75	0.74
<i>LogisticRegression Optimisé</i>	0.75	0.80	0.75	0.82
AdaBoostClassifier	0.75		0.75	0.74
<i>AdaBoostClassifier Optimisé</i>	0.75	0.79	0.75	0.82

Encadré n°4 : Qu'est-ce que l'Interprétabilité d'un modèle ?

L'interprétation des modèles est importante en Machine Learning, ne serait-ce que parce que dans de nombreux domaines, il faut expliquer – justifier – la prise de décision induite par le modèle prédictif.

La question de l'**identification des variables pertinentes** reste centrale, ne serait-ce que pour comprendre le mécanisme d'affectation sous-jacent au modèle.

L'importance des variables mesure l'impact global de chaque descripteur dans le modèle.

La technique de "permutation feature importance" est destinée à mesurer l'importance des variables. Son énorme avantage est qu'elle s'applique à n'importe quel type de modèle prédictif.

L'idée de la "permutation feature importance" consiste à opposer les performances du modèle en prédiction avec et sans la variable à évaluer. On peut alors se baser sur le taux d'erreur, mais tout autre indicateur peut convenir (Log Loss, etc.).

Pour neutraliser la variable, on mélange aléatoirement les valeurs à l'intérieur du vecteur et, par conséquent, on casse le lien qu'elle peut entretenir avec la classe à prédire (et les autres variables par la même occasion). En réitérant l'opération plusieurs fois, nous disposons de résultats plus stables (en prenant la médiane du ratio mesuré par exemple) avec de surcroit une indication sur la variabilité des résultats.

L'importance des variables peut être estimée en modélisation (sur l'échantillon d'apprentissage) ou en prédiction (sur l'échantillon test).

L'échantillon d'apprentissage permet de mettre en avant le rôle des variables dans la modélisation, alors que l'échantillon test éclaire sur l'influence de la variable sur les performances en généralisation (lorsque le modèle est déployé dans la population).

Dans les deux cas, les principales étapes sont les mêmes :

1. calculer le taux d'erreur de référence,
2. calculer ensuite le même indicateur en neutralisant tour à tour chaque variable prédictive,
3. former le ratio entre les deux valeurs.

En résumé, **les valeurs de Shapley calculent l'importance d'une variable en comparant ce qu'un modèle prédit avec et sans cette variable.**

Cependant, étant donné que l'ordre dans lequel un modèle voit les variables peut affecter ses prédictions, cela se fait dans tous les ordres possibles, afin que les fonctionnalités soient comparées équitablement. Cette approche est inspirée de la théorie des jeux.

SHAP (SHapley Additive exPlanations) est une bibliothèque Python qui permet d'expliquer les prédictions des modèles de machine learning. Elle utilise la théorie des jeux pour attribuer une importance à chaque variable (feature) d'entrée en mesurant la contribution marginale de chaque feature à la prédiction finale.

La contribution marginale d'une feature à la prédiction finale est la différence entre la valeur de la prédiction avec la feature et la valeur de la prédiction sans la feature. En d'autres termes, c'est la contribution de chaque feature à la prédiction finale, en supposant que toutes les autres features restent constantes.

SHAP utilise la théorie des jeux pour calculer la contribution marginale de chaque feature à la prédiction finale. Elle considère toutes les combinaisons possibles de features, calcule la contribution de chaque feature à chaque combinaison et prend la moyenne de toutes les combinaisons pour obtenir la contribution marginale de chaque feature.

Les contributions marginales sont utiles pour comprendre comment chaque feature affecte la prédiction du modèle. Elles peuvent également être utilisées pour identifier les features les plus importantes pour la prédiction. En effet, une feature avec une contribution marginale élevée aura un impact important sur la prédiction, et sera donc considérée comme plus importante qu'une feature avec une contribution marginale faible.

En d'autres termes, SHAP permet de répondre à la question : "**Pourquoi le modèle a-t-il prédit cette valeur ?**". Elle fournit ainsi des explications claires et compréhensibles pour les décideurs et les utilisateurs finaux, ce qui renforce la transparence et la confiance dans les prédictions du modèle.

L'intelligibilité globale cherche à expliquer le modèle dans sa globalité. C'est-à-dire quelles sont les variables les plus importantes en moyenne pour le modèle. A contrario, **l'intelligibilité locale**, consiste à expliquer la prévision $f(x)$ d'un modèle pour un individu x donné.

01. Classification Binaire (suite)

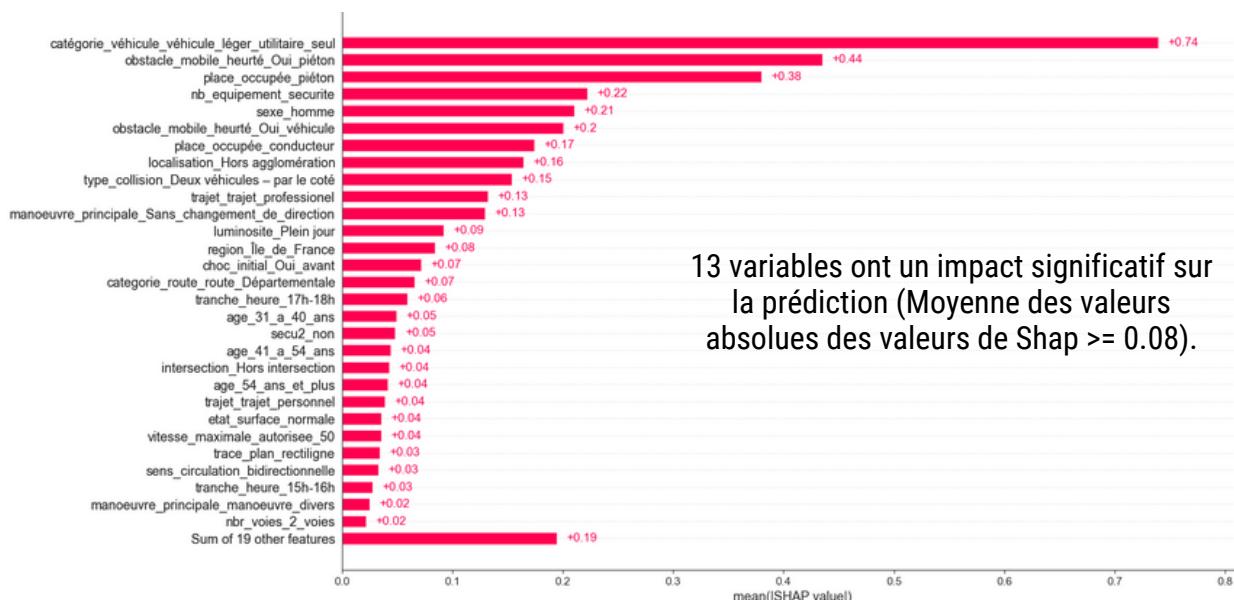
01.3. Interprétation Globale du modèle XGBoost Binaire

L'importance des variables est calculée en moyennant la valeur absolue des valeurs de Shap.

Les caractéristiques sont classées de l'effet le plus élevé au plus faible sur la prédiction.

Le calcul prend en compte la valeur SHAP absolue, donc peu importe si la fonctionnalité affecte la prédiction de manière positive ou négative.

Le diagramme d'importance des variables répertorie les variables les plus significatives par ordre décroissant. Les variables en haut contribuent davantage au modèle que celles en bas et ont donc un pouvoir prédictif élevé. Il répond à la question : **Quelles sont les variables globalement les plus importantes pour comprendre la prédiction ?**

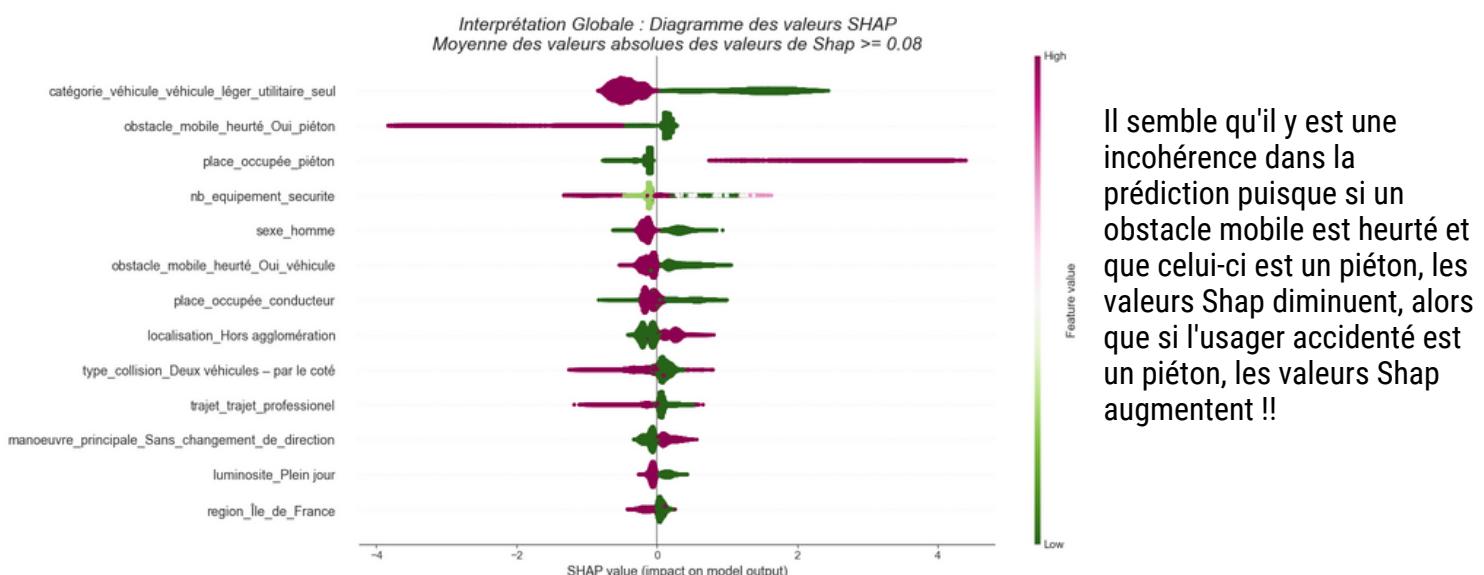


13 variables ont un impact significatif sur la prédiction (Moyenne des valeurs absolues des valeurs de Shap ≥ 0.08).

Le diagramme des valeurs SHAP indique également comment chaque caractéristique impacte la prédiction.

Les valeurs de Shap sont représentées pour chaque variable dans leur ordre d'importance. Chaque point représente une valeur de Shap (c'est-à-dire un usager de la route). Les points fuchsia représentent des valeurs élevées de la variable et les points verts des valeurs basses de la variable.

Ce diagramme répond à la question : **Quel est l'Impact de chaque caractéristique sur la prédiction ?**

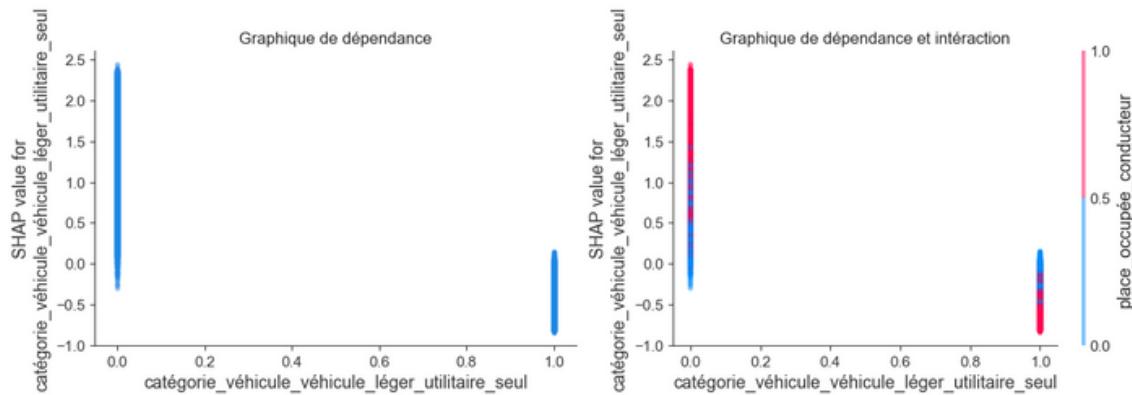


01. Classification Binaire (suite)

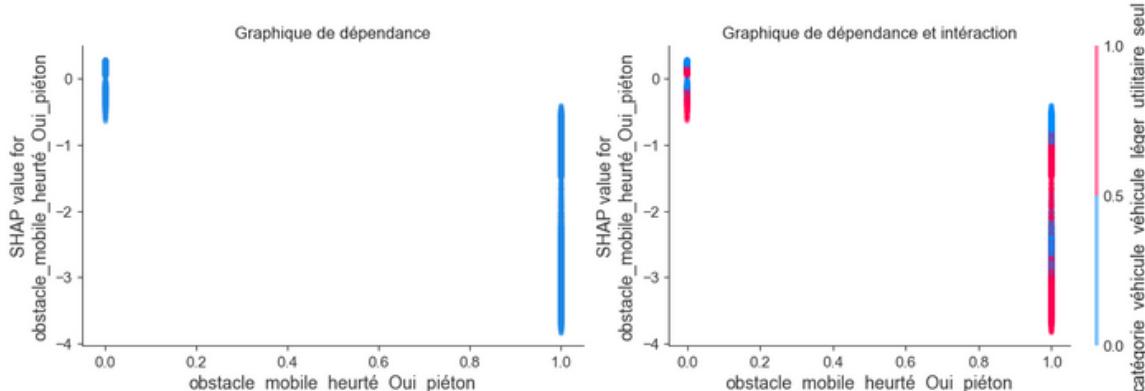
Mais, dans certains cas, la relation entre la **variable cible et la caractéristique dépend de la valeur d'une autre caractéristique**. C'est ce qu'on appelle une **interaction**.

Si la catégorie du véhicule impliqué dans l'accident est un véhicule léger (voiture légère ; valeur = 1), la valeur Shap diminue et par conséquent la probabilité que l'accident soit corporel décroît aussi.

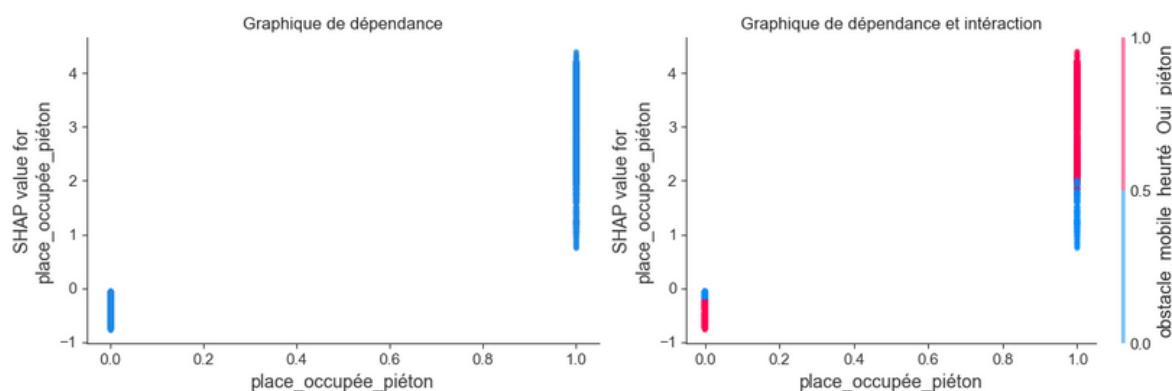
Grâce au graphique d'interaction on constate aussi que si la personne accidentée est le conducteur du véhicule, on peut dire que lorsque son véhicule n'est pas un véhicule léger (valeur = 0), la valeur Shap augmente et par conséquent la probabilité que l'accident soit corporel augmente aussi.



Si un obstacle mobile est heurté dans l'accident et que celui-ci est un piéton (valeur = 1), la valeur Shap diminue et par conséquent la probabilité que l'accident soit corporel décroît aussi. Ce résultat apparaît très curieux !



En revanche, si l'usager accidenté est un piéton (valeur = 1), la valeur Shap augmente et par conséquent la probabilité que l'accident soit corporel augmente aussi. Et c'est d'autant plus vrai si ce dernier a été heurté par le véhicule impliqué dans l'accident.



01. Classification Binaire (suite)

01.4. Interprétation Locale du modèle XGBoost Binaire

Pour comprendre et interpréter visuellement les résultats du modèle localement, c'est-à-dire pour une instance il existe trois alternatives : Force plot, Decision plot et Waterfall plot.

Pour une instance (ici un usager de la route), ces trois représentations sont redondantes, elles représentent l'information de manière très similaire. En même temps, certains éléments de ces graphiques sont complémentaires. En mettant côté à côté les trois, on comprend d'une manière plus intuitive le résultat.

Le Force plot est bon pour voir où se place le "output value" par rapport à la "base value". Nous observons également quelles variables ont un impact positif (rouge) ou négatif (bleu) sur la prédiction et l'amplitude de cet impact.

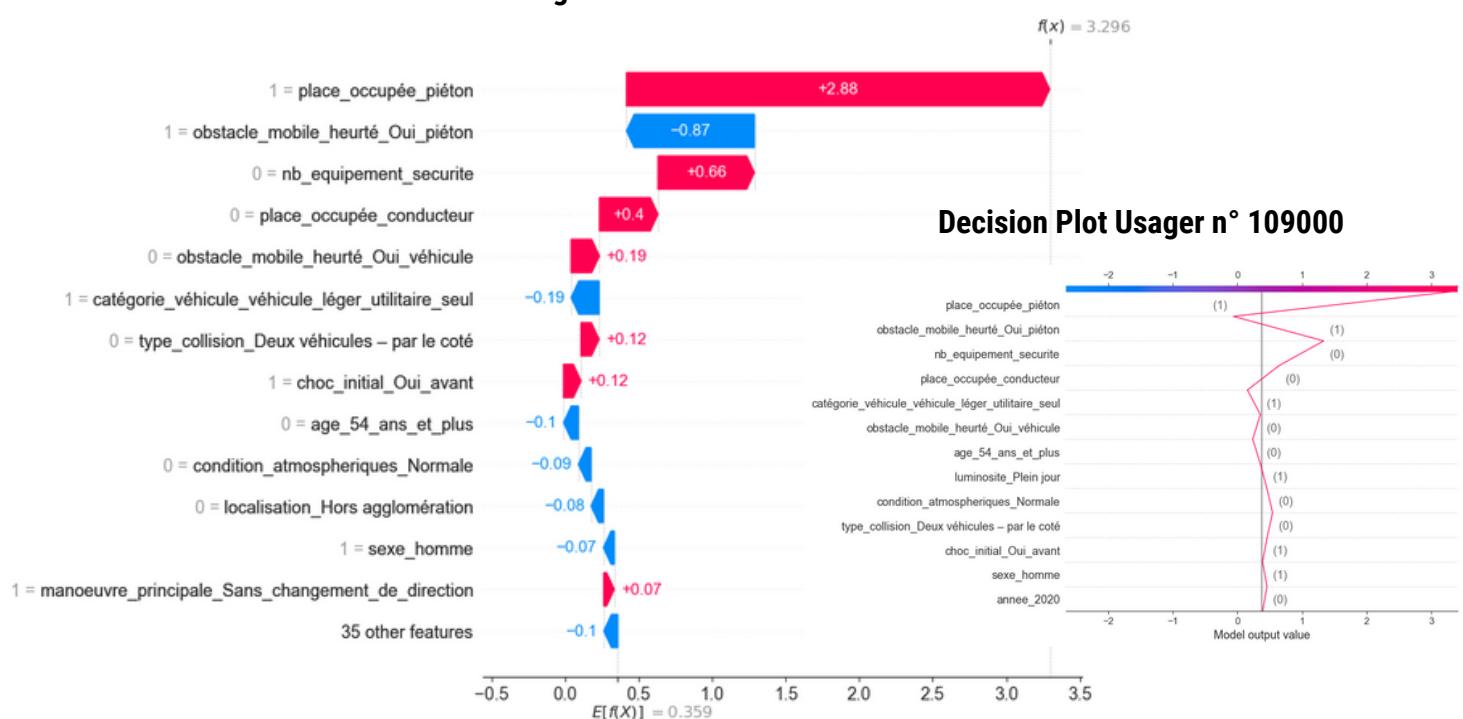
Le Waterfall plot permet aussi de voir l'amplitude et la nature d'impact d'une variable avec sa quantification. Il permet de voir également l'ordre d'importance des variables et les valeurs prises par chacune des variables pour l'instance étudiée. Il affiche les contributions marginales de chaque feature pour l'observation spécifique sélectionnée. Les features qui ont une contribution positive sont affichées en rouge, tandis que les features qui ont une contribution négative sont affichées en bleu. Chaque barre représente la contribution marginale de chaque feature à la prédiction finale du modèle, et la barre totale représente la prédiction moyenne du modèle pour l'ensemble des observations. Le graphique permet de visualiser l'effet cumulatif des features sur la prédiction finale du modèle.

Le Decision plot permet d'observer l'amplitude de chaque changement, "une trajectoire" prise par une instance pour les valeurs des variables affichées.

Force Plot Usager n° 109000

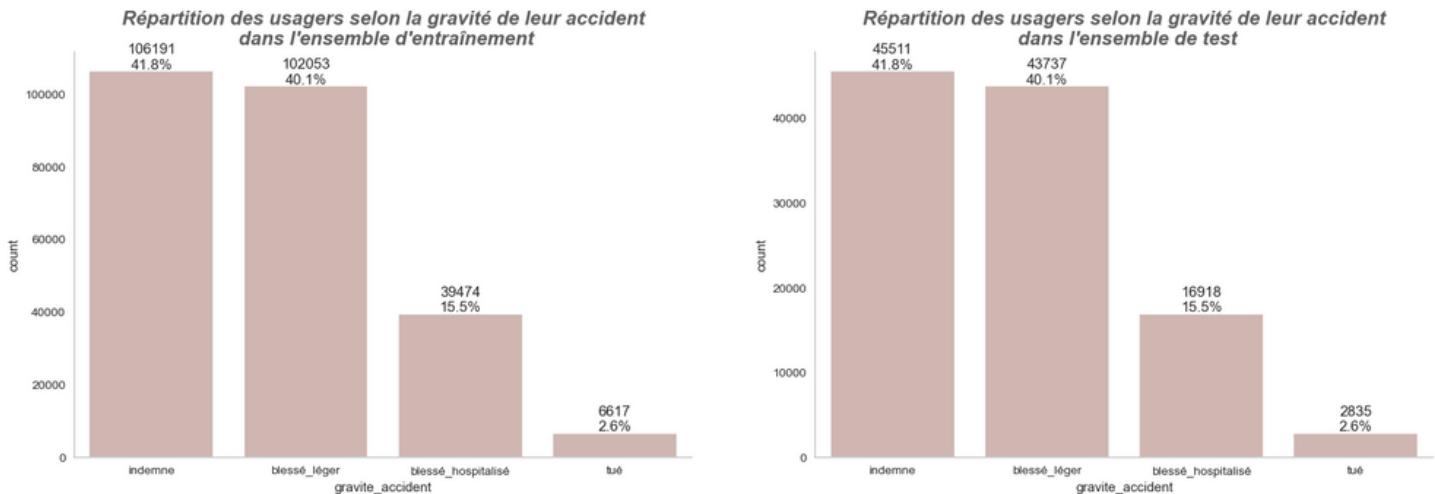


Waterfall Plot Usager n° 109000



02. Classification Multi-Classes

Notre étude a pour objectif de modéliser la gravité des accidents de la route en France en se concentrant sur quatre classes d'usagers : indemnes, blessés légers, hospitalisés et tués.



Nous avons décomposé notre jeu de données en 2 sous-ensembles : 70% pour entraîner nos modèles de prédiction et 30% pour les tester ; en respectant la répartition des 4 classes par "stratification". La "stratification", divise le jeu de données de manière à maintenir la même proportion de chaque classe dans les ensembles d'apprentissage et de test. Cette approche est particulièrement utile lorsque certaines classes sont moins représentées que d'autres, car elle garantit que les sous-ensembles contiennent suffisamment d'exemples de chaque classe pour une évaluation précise du modèle.

02.1. Choix du modèle le mieux adapté à la prédiction de la variable cible multi-classes

Pour sélectionner les modèles les plus adaptés, plusieurs modèles de classification ont été testés afin de déterminer le plus performant.

- Sélection des modèles :** Plusieurs modèles de classification ont été choisis pour être évalués, y compris des modèles de base (DummyClassifier) et des modèles plus avancés (LogisticRegression, RandomForestClassifier, XGBClassifier, LGBMClassifier, et BalancedRandomForestClassifier).
- Préparation des données :** Les données ont été divisées en ensembles d'entraînement et de test. La division des données permet d'entraîner les modèles sur un ensemble de données et de les tester sur un ensemble de données différent pour évaluer leur performance en termes de généralisation.
- Validation croisée :** Pour obtenir une estimation plus fiable de la performance des modèles, une validation croisée à 5 plis a été utilisée. Les données d'entraînement ont été divisées en 5 sous-ensembles, et chaque modèle a été entraîné et testé sur ces sous-ensembles, répétant le processus 5 fois. Cela permet de réduire les effets du surapprentissage et d'améliorer la capacité de généralisation du modèle choisi.
- Calcul des métriques de performance et des temps d'exécution :** Pour chaque modèle, les métriques de performance suivantes ont été calculées pour les ensembles d'entraînement et de test : balanced_accuracy_score, precision_score (weighted), recall_score (weighted) et f1_score (weighted). De plus, les temps d'entraînement et de test ont été mesurés pour chaque modèle.
- Analyse des résultats :** Les résultats ont été rassemblés dans un DataFrame pour faciliter leur analyse. Les métriques de performance et les temps d'exécution ont été comparés pour choisir le modèle le plus adapté au problème en fonction de la performance et de l'efficacité.

02. Classification Multi-Classes

En intégrant la validation croisée et les temps d'entraînement et de test, cette méthodologie permet d'évaluer de manière plus robuste et complète les modèles de classification pour résoudre un problème de classification multi-classe déséquilibrée. L'analyse des résultats permet de prendre en compte non seulement la performance des modèles, mais aussi leur efficacité en termes de temps d'exécution, ce qui peut être important dans certaines situations où les ressources de calcul ou le temps sont limités.

	balanced accuracy train	balanced accuracy test	precision weighted train	precision weighted test	recall weighted train	recall weighted test	f1 weighted train	f1 weighted test	Train time	Test time
dummy	25,0	25,0	36,0	36,0	36,0	36,1	36,0	36,0	0,1	0,1
logreg_multi	40,8	40,8	58,1	58,1	60,9	60,9	58,5	58,5	11,6	0,1
rf_multi	99,2	44,9	99,5	61,9	99,5	63,7	99,5	62,1	42,1	2,8
xgb_multi	49,5	46,2	66,3	63,0	66,9	64,3	65,8	63,1	185,6	0,6
lgbm_multi	46,7	45,8	64,0	63,0	65,0	64,3	63,7	62,9	9,2	1,5
balanced_rf	72,3	53,1	71,0	63,1	64,2	55,4	65,8	57,8	9,9	1,6

Dans le contexte, où nous accordons autant d'importance à la performance qu'au temps d'exécution **le "meilleur" modèle** pour prédire la gravité des accidents de la route avec une distribution de classes déséquilibrée, **est le modèle lgbm_multi (LightGBM)** pour les raisons suivantes :

- Performance** : Le modèle lgbm_multi a des métriques de performance (balanced accuracy, précision pondérée, rappel pondéré et score F1 pondéré) sur l'ensemble de test qui sont proches de celles du modèle xgb_multi (XGBoost), qui est l'un des modèles les plus performants parmi ceux que vous avez testés. Cela indique que LightGBM est capable de prédire la gravité des accidents de la route avec une précision comparable à celle de XGBoost.
- Temps d'exécution** : Le modèle lgbm_multi a des temps d'entraînement et de test nettement plus courts que le modèle xgb_multi. Ceci est important si nous devons entraîner et tester fréquemment le modèle, ou si nous disposons de ressources informatiques limitées.
- Gestion des classes déséquilibrées** : LightGBM est capable de gérer efficacement les problèmes avec des classes déséquilibrées grâce à des méthodes intégrées, telles que la pondération des instances ou l'échantillonnage des instances.

Le modèle RandomForest (rf_multi) présente un surapprentissage (overfitting), car il a une performance très élevée sur l'ensemble d'entraînement (99,5% pour la précision, le rappel et le F1 score), mais une performance beaucoup plus faible sur l'ensemble de test. Cela indique que le modèle est trop complexe et qu'il a appris le bruit présent dans les données d'entraînement, ce qui limite sa capacité à généraliser sur de nouvelles données.

Le modèle Dummy est le moins performant, comme prévu, car il sert de point de référence pour comparer les performances des autres modèles.

02. Classification Multi-Classes

02.2. Optimisation des "meilleurs" modèles

Même si nous avons constaté que le modèle LightGBM était le plus performant si nous prenions en compte le temps d'entraînement des modèles, nous avons choisi d'explorer aussi les modèles XGBoost et Balanced Random Forest Classifier en prenant en compte le déséquilibre des classes et en optimisant les hyperparamètres.

Pour prendre en compte le déséquilibre des classes avec XGBoost, nous avons utilisé trois techniques :

- Cost-Sensitive Learning** : Cette méthode attribue des poids différents aux erreurs de classification des différentes classes. Les classes sous-représentées se voient attribuer un poids plus élevé, ce qui permet au modèle d'accorder plus d'importance à leur classification correcte.
- Sous-échantillonnage des classes majoritaires (SMOTE Tomek)** : Cette technique combine le sur-échantillonnage des classes minoritaires avec le sous-échantillonnage des classes majoritaires en utilisant la méthode SMOTE (Synthetic Minority Over-sampling Technique) pour générer de nouvelles instances des classes minoritaires, et la méthode Tomek Links pour supprimer les instances des classes majoritaires situées à proximité des instances des classes minoritaires.
- Sous-échantillonnage avec ClusterCentroids** : Cette méthode effectue un sous-échantillonnage des classes majoritaires en calculant les centroïdes de clusters pour chaque classe. Les instances les plus proches des centroïdes sont conservées pour représenter les classes majoritaires.

Les résultats obtenus pour les différentes techniques utilisées avec XGBoost sont les suivants :

Performances Globales							
Modèles	Paramètres Optimisés	Balanced Accuracy	Weighted Precision	Weighted Recall	Weighted F1-score	Average AUC	Score Géo Global
XGBoost Cost-Sensitive Learning	Non	55%	64%	58%	60%	82%	69%
XGBoost SMOTETomek	Non	51%	63%	59%	60%	80%	69%
XGBoost ClusterCentroids	Non	28%	55%	6%	5%	57%	16%

Le modèle **XGBoost Cost-Sensitive Learning montre les meilleures performances** parmi les trois techniques testées pour gérer le déséquilibre des classes, principalement en raison de sa capacité à attribuer des poids différents aux erreurs de classification des différentes classes. Voici quelques raisons pour lesquelles cette approche fonctionne bien :

- Pénalisation adaptée des erreurs de classification** : Dans le cas du Cost-Sensitive Learning, les erreurs de classification des classes sous-représentées sont attribuées avec des poids plus élevés. Cela incite le modèle à accorder plus d'importance à la classification correcte de ces classes, ce qui peut améliorer les performances globales du modèle pour les classes déséquilibrées.
- Adaptabilité du modèle XGBoost** : XGBoost est un algorithme d'ensemble qui utilise le boosting pour combiner les résultats de plusieurs classificateurs faibles. Le boosting fonctionne en corrigeant les erreurs des classificateurs précédents, en accordant plus de poids aux observations mal classées. Le Cost-Sensitive Learning s'intègre naturellement dans ce cadre, permettant au modèle d'adapter les poids des erreurs en fonction de la distribution des classes.
- Réduction du biais et de la variance** : L'utilisation de l'approche Cost-Sensitive Learning aide à réduire à la fois le biais et la variance du modèle. En accordant plus d'importance aux classes sous-représentées, le modèle est moins susceptible d'être biaisé en faveur des classes majoritaires. En même temps, le modèle XGBoost utilise le boosting et la régularisation pour contrôler la variance, ce qui contribue à améliorer la précision de la prédiction.

02. Classification Multi-Classes

Après avoir constaté que le modèle XGBoost Cost-Sensitive Learning présentait les meilleures performances, nous avons décidé d'optimiser ses paramètres en utilisant le module 'hgboost' qui utilise l'optimisation bayésienne pour sélectionner les hyperparamètres optimaux.

Les résultats obtenus avec le modèle **XGBoost Cost-Sensitive Learning optimisé avec hgboost** sont les suivants :

1. **Balanced Accuracy (52%)** : La Balanced Accuracy est une mesure de performance qui prend en compte le déséquilibre des classes. Elle est calculée comme la moyenne des taux de rappel pour chaque classe. Une Balanced Accuracy de 52% indique que le modèle est capable de classer correctement les observations pour chaque classe environ 52% du temps, ce qui est supérieur à une classification aléatoire.
2. **Weighted Precision (64%)** : La précision pondérée est la moyenne des précisions pour chaque classe, pondérée par le nombre d'observations de chaque classe. Une Weighted Precision de 64% signifie que, lorsque le modèle prédit une classe, il est correct environ 64% du temps.
3. **Weighted Recall (62%)** : Le rappel pondéré est la moyenne des rappels pour chaque classe, pondérée par le nombre d'observations de chaque classe. Un Weighted Recall de 62% signifie que le modèle identifie correctement environ 62% des observations de chaque classe.
4. **Weighted F1-score (63%)** : Le F1-score pondéré est la moyenne harmonique des précisions et rappels pondérés pour chaque classe. Un F1-score équilibré de 63% indique que le modèle a un bon équilibre entre la précision et le rappel pour chaque classe, compte tenu du déséquilibre des classes.
5. **Average AUC (83%)** : L'aire sous la courbe ROC (AUC) est une mesure de performance qui évalue la capacité d'un modèle à discriminer entre les classes positives et négatives. Un Average AUC de 83% indique que le modèle a une bonne capacité à distinguer entre les différentes classes.
6. **Score Géo Global (71%)** : Le Score Géo Global est la moyenne géométrique (ou G-Mean) des rappels pour chaque classe. Cette mesure de performance tient compte à la fois de la capacité du modèle à classifier correctement les différentes classes et de l'équilibre entre les classes. Un G-Mean élevé indique que le modèle est performant pour toutes les classes, y compris celles qui sont moins représentées. Un G-Mean de 71% montre que le modèle est capable de classer correctement les observations pour chaque classe, y compris les classes minoritaires, environ 71% du temps. Cela indique que le modèle a une performance globale relativement élevée en tenant compte à la fois de la capacité à classifier les différentes classes et de l'équilibre entre les classes.

Encadré n° 5 : le module 'hgboost' et l'optimisation bayésienne

Le module 'hgboost' est une implémentation de l'optimisation bayésienne pour les modèles de boosting basés sur les arbres, tels que XGBoost, LightGBM et CatBoost. Il utilise la bibliothèque 'scikit-optimize' pour effectuer l'optimisation bayésienne et peut être utilisé pour ajuster automatiquement les hyperparamètres de ces modèles en fonction de la performance sur un ensemble de validation.

Voici quelques avantages de l'utilisation de l'optimisation bayésienne avec 'hgboost':

- Exploration et exploitation intelligentes: L'optimisation bayésienne explore et exploite l'espace des hyperparamètres de manière intelligente, en tenant compte des performances passées pour orienter la recherche vers des régions prometteuses de l'espace des hyperparamètres. Cela permet de trouver des solutions plus rapidement que les approches d'optimisation aléatoire ou de recherche exhaustive.
- Incorporation des incertitudes: L'optimisation bayésienne prend en compte les incertitudes dans l'estimation des performances des modèles, ce qui peut aider à éviter les solutions trop optimistes ou pessimistes et à obtenir un modèle plus robuste.
- Optimisation globale: L'optimisation bayésienne cherche à trouver les hyperparamètres optimaux en tenant compte de l'ensemble de l'espace des hyperparamètres, ce qui peut conduire à une meilleure performance globale du modèle.

02. Classification Multi-Classes

Nous avons ensuite optimisé les modèles LightGBM et Balanced Random Forest et obtenu l'ensemble des résultats suivants :

Performances Globales							
Modèles	Paramètres Optimisés	Balanced Accuracy	Weighted Precision	Weighted Recall	Weighted F1-score	Average AUC	Score Géo Global
XGBoost Cost-Sensitive Learning	Non	55%	64%	58%	60%	82%	69%
XGBoost SMOTETomek	Non	51%	63%	59%	60%	80%	69%
XGBoost ClusterCentroids	Non	28%	55%	6%	5%	57%	16%
XGBoost Cost-Sensitive Learning	hgboost	52%	64%	62%	63%	83%	71%
XGBoost Cost-Sensitive Learning	GridSearchCV	51%	63%	61%	62%	81%	70%
BalancedRandomForestClassifier	Non	54%	63%	56%	58%	80%	68%
BalancedRandomForestClassifier	RandomizedSearchCV	54%	64%	56%	59%	82%	68%
LightGBMClassifier	Optuna	54%	64%	60%	61%	82%	70%

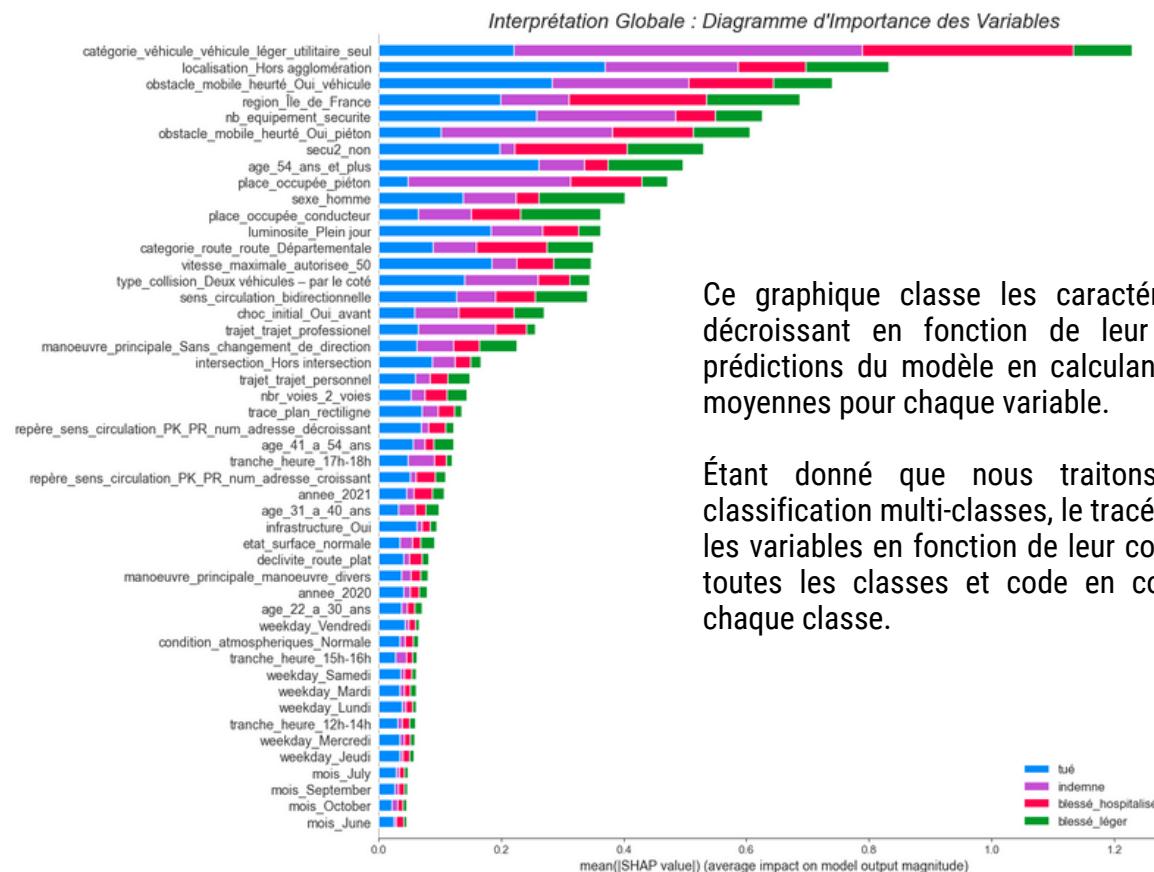
Quel est le modèle qui répond le mieux à notre problématique de prédire la gravité des accidents de la route ?

Sans contexte, même si globalement les performances ne sont pas excellentes, c'est le modèle XGBoost Cost-Sensitive Learning dont les hyperparamètres sont optimisés selon la technique de l'optimisation bayésienne.

Mais comment interpréter les résultats de ce modèle ?

Comme pour la classification binaire, nous avons utilisé les valeurs de Shapley qui calculent l'importance d'une variable en comparant ce qu'un modèle prédit avec et sans cette variable.

02.3. Interprétation Globale du modèle XGBoost Multi-Classes



Ce graphique classe les caractéristiques par ordre décroissant en fonction de leur influence sur les prédictions du modèle en calculant les valeurs SHAP moyennes pour chaque variable.

Étant donné que nous traitons d'une tâche de classification multi-classes, le tracé récapitulatif classe les variables en fonction de leur contribution globale à toutes les classes et code en couleur l'ampleur de chaque classe.

02. Classification Multi-Classes

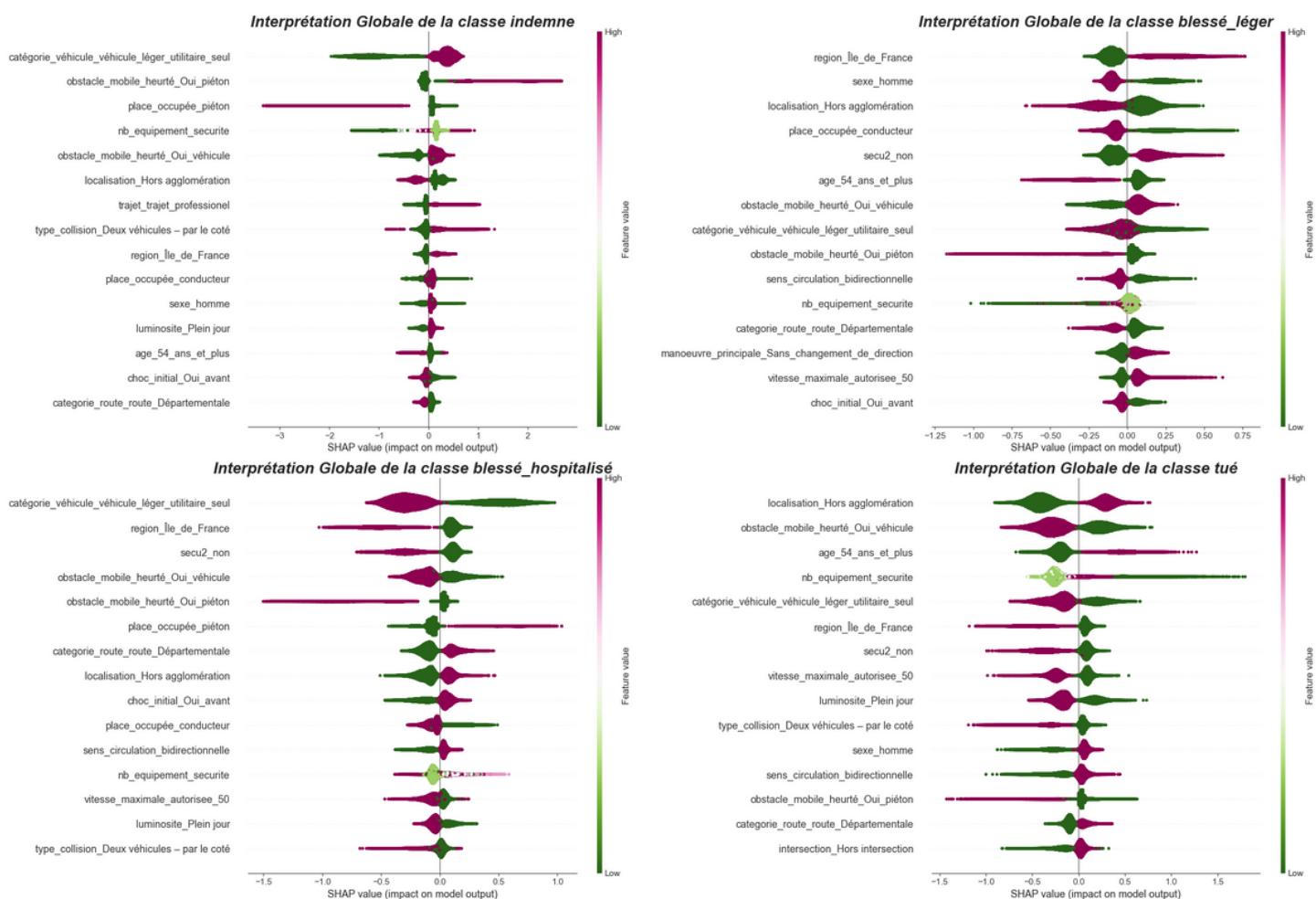
Les graphiques ci-dessous illustrent plus en détail l'impact des caractéristiques sur le processus de classification.

Ils examinent la classification une classe à la fois.

Par conséquent, les caractéristiques importantes sur l'axe Y sont classées par ordre décroissant en fonction de leur impact sur la classe.

L'impact est représenté sur l'axe X par la valeur SHAP calculée pour chaque usager de la route. Si la valeur SHAP d'un usager est positive (négative), cela signifie que la caractéristique correspondante a un impact positif (négatif) sur la classification de cet usager.

Chaque point du tracé récapitulatif est un usager, codé par un dégradé de fuchsia à vert représentant la valeur d'origine de l'entité et positionnée en fonction de sa distribution. La couleur fuchsia représente les valeurs élevées de la caractéristique, tandis que la couleur verte représente les valeurs faibles



Par exemple, le risque que l'usager soit tué lors de l'accident de la route augmente lorsqu'il a lieu hors agglomération et que l'accidenté est âgé de 54 ans et plus, et diminue si l'obstacle mobile heurté est un véhicule.

02. Classification Multi-Classes

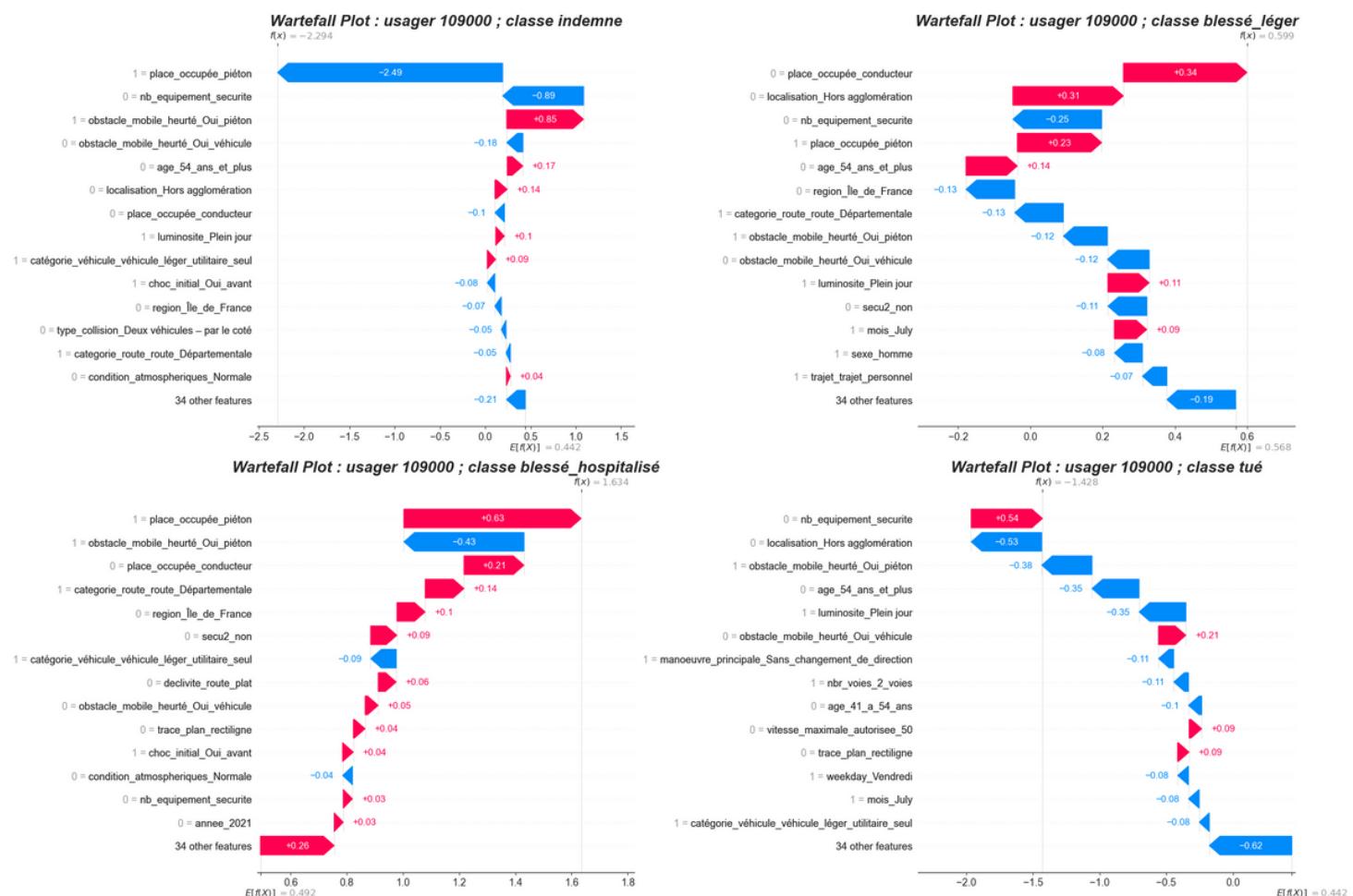
02.4. Interprétation Locale du modèle XGBoost Multi-Classes

Lorsque nous avons un modèle de classification avec plusieurs classes et que nous créons un graphique '**waterfall_plot**' pour une observation donnée, **chaque classe a son propre graphique waterfall_plot avec sa propre contribution marginale de chaque fonctionnalité à la prédiction de cette classe.**

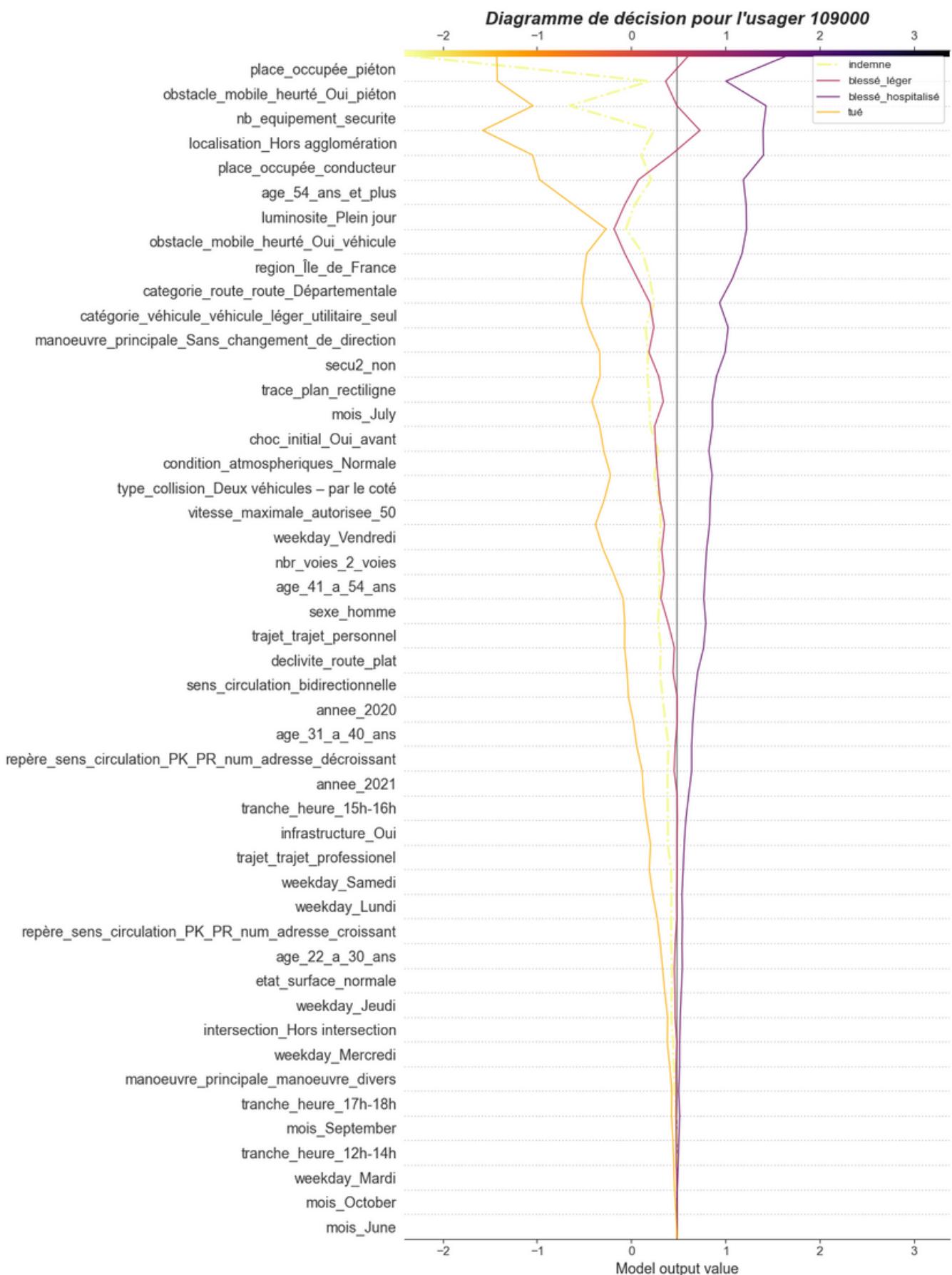
Cela signifie que nous obtenons un graphique **waterfall_plot** pour chaque classe, et nous devons interpréter chaque graphique individuellement.

L'interprétation des graphiques **waterfall_plot** pour les modèles de classification multi-classes peut être un peu plus complexe que pour les modèles binaires, car nous devons considérer les contributions marginales pour chaque classe. Cependant, nous pouvons suivre les étapes générales suivantes pour interpréter les graphiques **waterfall_plot** pour chaque classe :

1. La barre verticale centrale représente la valeur de base pour la prédiction de la classe. La position de la barre verticale est déterminée par la valeur de l'intercepteur ou de la valeur attendue pour la classe.
2. Les barres horizontales représentent la contribution marginale de chaque fonctionnalité à la prédiction de la classe. Les barres sont triées de haut en bas en fonction de leur contribution à la prédiction de la classe. Les barres rouges indiquent une contribution positive, tandis que les barres bleues indiquent une contribution négative.
3. Les barres horizontales sont étiquetées avec le nom de la fonctionnalité et la valeur de cette fonctionnalité pour l'observation donnée.
4. La longueur de chaque barre horizontale représente la contribution marginale de la fonctionnalité à la prédiction de la classe. Les barres plus longues ont une contribution plus importante à la prédiction de la classe.



02. Classification Multi-Classes



03. Modélisation avec CatBoost

CatBoost est un algorithme d'apprentissage automatique à gradient développé par Yandex.

Moins utilisée que XGBoost et LightGBM, il bénéficie pourtant de caractéristiques qui la rendent pertinente dans certaines situations.

Ses concepteurs revendiquent en particulier une simplicité d'utilisation qui ne se fait pas au détriment des performances. Utiliser la librairie telle quelle, sans changer les paramètres par défaut, doit conduire à la génération d'un modèle avec de bonnes performances. Le support natif des features catégorielles n'est pas étranger à cette facilité de mise en œuvre.

En effet l'une des fonctionnalités très intéressantes de CatBoost est qu'il gère les variables catégorielles prêtes à l'emploi (d'où le nom de l'algorithme).

Nous savons que les algorithmes d'apprentissage automatique ne gèrent que les caractéristiques numériques. C'est pourquoi, comme nous l'avons vu précédemment, nous avons remplacé chaque variable catégorielle par des variables factices (une pour chaque catégorie distincte qui indique si un facteur spécifique est présent (valeur 1) ou absent (valeur 0)).

Mais le problème majeur alors est si nous avons beaucoup de catégories différentes, nous souffrirons de problèmes de grande dimensionnalité.

Dans un tel cas, lorsqu'elles seront utilisées conjointement avec un arbre de décision ou un ensemble d'arbres de décision, parce que nos variables factices contiendront principalement des zéros, elles seront très déséquilibrées.

Lors de la sélection d'une caractéristique pour une division, cela augmentera le risque que les variables factices soient ignorées au profit des autres prédicteurs. Cela se produit parce que le gain d'information associé (réduction d'entropie en cas de classification et réduction de la variance cible en cas de régression) sera probablement plus faible lors de la sélection de cette caractéristique pour un fractionnement.

Par conséquent, dans le cas d'un nombre élevé de niveaux, l'utilisation d'un codage à chaud n'est peut-être pas la meilleure transformation à utiliser car elle augmente les chances de diminuer l'importance de la variable catégorielle dans son ensemble dans le modèle résultant.

C'est pourquoi, CatBoost utilise une autre technique (Ordered Target Statistics) pour encoder les variables catégorielles.

Encadré n°6 : Qu'est-ce que la méthode "Ordered Target Statistics" ?

"Ordered Target Statistics" est une technique de CatBoost qui permet l'encodage de caractéristiques catégorielles en utilisant des statistiques sur la variable cible (ou target).

L'encodage de caractéristiques catégorielles consiste à convertir des variables catégorielles en valeurs numériques afin que les algorithmes d'apprentissage automatique puissent les traiter. Dans CatBoost, l'encodage des caractéristiques catégorielles est effectué en utilisant les statistiques de la variable cible, telles que la moyenne, la médiane, la somme, etc.

Dans "Ordered Target Statistics", les catégories sont triées en fonction de leur fréquence de la variable cible, et les statistiques sont calculées en fonction de cet ordre. Cette approche permet de capturer les relations entre les caractéristiques catégorielles et la variable cible de manière plus précise et de mieux modéliser les variations de la variable cible en fonction des caractéristiques catégorielles.

03. Modélisation avec CatBoost

Quelques éléments clés pour comprendre la modélisation CatBoost ?

1. **Boosting** : CatBoost utilise l'approche de boosting, qui consiste à combiner plusieurs modèles faibles (généralement des arbres de décision) pour créer un modèle global plus fort et plus précis. Les modèles sont construits de manière séquentielle, où chaque modèle corrige les erreurs du modèle précédent.
2. **Arbres de décision** : CatBoost utilise des arbres de décision comme modèles de base. Les arbres de décision sont des structures arborescentes où chaque nœud représente une caractéristique, chaque branche représente une règle de décision, et chaque feuille représente un résultat ou une prédition.
3. **Catégorisation des variables** : L'une des caractéristiques clés de CatBoost est son traitement efficace des variables catégorielles. Il utilise des techniques spécifiques, telles que le "target encoding" (encodage de la cible), pour gérer ces variables sans avoir besoin de les convertir manuellement en variables numériques.
4. **Régularisation** : CatBoost implémente la régularisation pour réduire le surapprentissage et améliorer la performance du modèle sur les données de test. La régularisation ajoute une pénalité aux modèles complexes, favorisant des modèles plus simples et évitant ainsi de trop coller aux données d'apprentissage.
5. **Apprentissage par gradient** : CatBoost utilise l'apprentissage par gradient pour optimiser la construction de l'ensemble d'arbres de décision. À chaque étape, il construit un nouvel arbre pour minimiser la perte sur l'ensemble d'apprentissage en utilisant le gradient de la fonction de perte.
6. **Fonction de perte** : La fonction de perte mesure la différence entre les prédictions du modèle et les valeurs réelles. CatBoost prend en charge plusieurs fonctions de perte, telles que la perte logarithmique pour les problèmes de classification et la perte de la somme des erreurs au carré pour les problèmes de régression.
7. **Vitesse et performance** : CatBoost est optimisé pour offrir une performance de calcul élevée et une utilisation efficace de la mémoire. Il utilise des optimisations telles que l'apprentissage sur GPU et l'apprentissage par lots pour accélérer le processus d'entraînement.

En résumé, CatBoost est un algorithme de boosting d'arbres de décision qui se distingue par son traitement efficace des variables catégorielles, sa régularisation pour éviter le surapprentissage et ses optimisations pour une haute performance. Il est utilisé pour résoudre des problèmes de classification et de régression avec des données mixtes (numériques et catégorielles). **CatBoost est particulièrement avantageux lorsqu'il s'agit de traiter des ensembles de données avec un grand nombre de variables catégorielles.**

Et c'est effectivement le cas pour notre problématique puisque nous ne disposons que de variables catégorielles pour prédire la gravité des accidents de la route en France.

Performances Globales							
Modèles	Paramètres Optimisés	Balanced Accuracy	Weighted Precision	Weighted Recall	Weighted F1-score	Average AUC	Score Géo Global
XGBoost Cost-Sensitive Learning	Non	55%	64%	58%	60%	82%	69%
XGBoost SMOTETomek	Non	51%	63%	59%	60%	80%	69%
XGBoost ClusterCentroids	Non	28%	55%	6%	5%	57%	16%
XGBoost Cost-Sensitive Learning	hgboost	52%	64%	62%	63%	83%	71%
XGBoost Cost-Sensitive Learning	GridSearchCV	51%	63%	61%	62%	81%	70%
BalancedRandomForestClassifier	Non	54%	63%	56%	58%	80%	68%
BalancedRandomForestClassifier	RandomizedSearchCV	54%	64%	56%	59%	82%	68%
LightGBMClassifier	Optuna	54%	64%	60%	61%	82%	70%
CatBoost Multi-classes	Non	58%	66%	62%	63%	84%	71%
CatBoost Multi-classes	Optuna	58%	67%	63%	64%	85%	72%

Même sans optimiser les hyperparamètres de la modélisation Catboost, nous obtenons de "meilleurs" résultats que notre précédent modèle XGBoost optimisé.

03. Modélisation avec CatBoost

De plus, le modèle est plus parcimonieux puisque nous avons 33 variables explicatives complètes contre 48 variables fictives sélectionnées parmi 126. Et par conséquent, le modèle devient plus compréhensif.

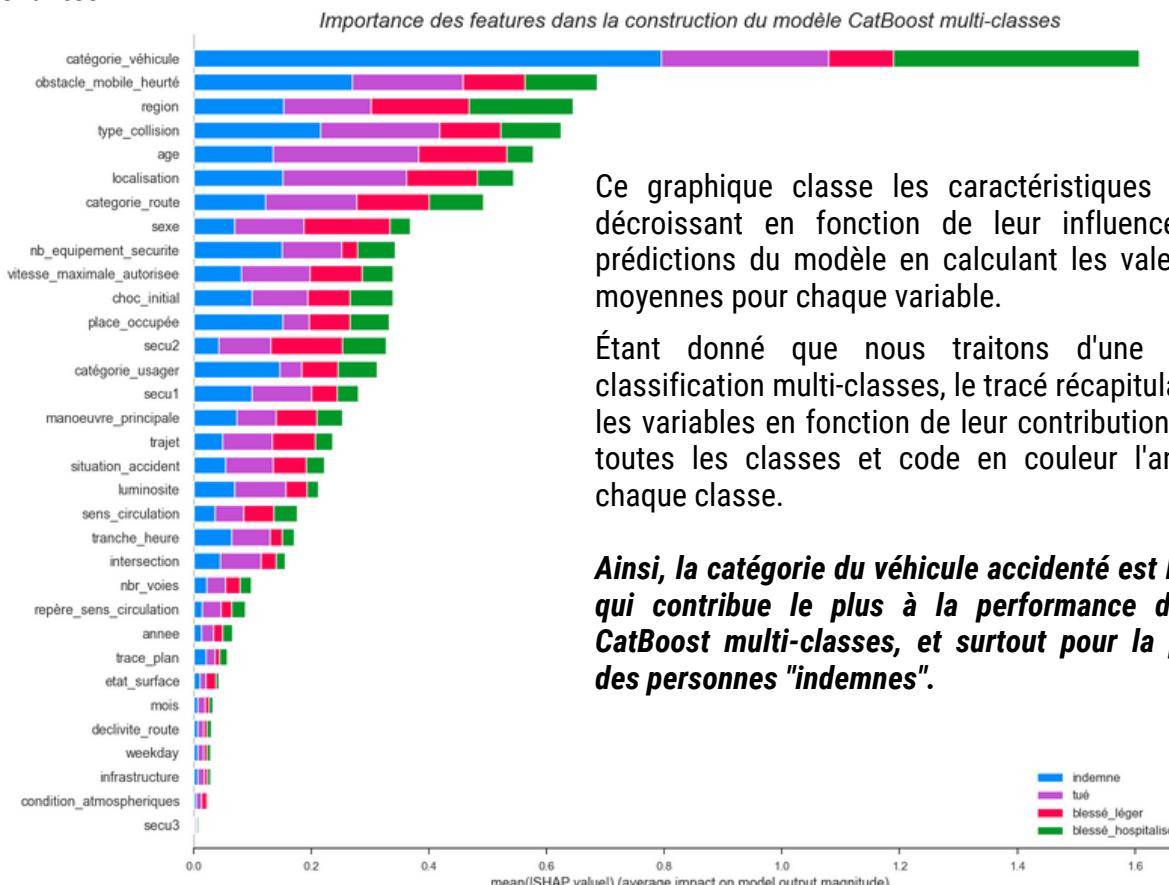
En effet, quand on utilise CatBoost pour traiter les variables catégorielles sans dummmification, l'interprétation des résultats est facilitée par rapport aux modèles où les variables ont été transformées manuellement et sélectionnées pour éviter l'augmentation de la dimensionnalité.

Les explications de SHAP pour les modèles CatBoost sont plus intuitives et significatives, car elles sont basées sur les variables catégorielles d'origine plutôt que sur les variables transformées.

L'utilisation conjointe de CatBoost et de SHAP offre donc les avantages suivants :

- 1. Interprétabilité** : Grâce à SHAP, les résultats des modèles CatBoost sont plus facilement compréhensibles pour les utilisateurs finaux et les parties prenantes. Les valeurs SHAP aident à déterminer l'importance des caractéristiques et leur impact sur les prédictions individuelles.
- 2. Conservation des variables catégorielles d'origine** : CatBoost permet de conserver les variables catégorielles dans leur format d'origine, ce qui facilite l'explication des résultats. Les valeurs SHAP sont directement liées aux variables d'origine, rendant l'interprétation plus intuitive.
- 3. Moins de biais de sélection de variables** : Comme CatBoost gère les variables catégorielles sans nécessiter de sélection manuelle de variables pour éviter l'augmentation de la dimensionnalité, l'interprétation des modèles est moins susceptible d'être biaisée par des choix arbitraires de variables à inclure ou exclure du modèle.
- 4. Efficacité** : L'utilisation de CatBoost et SHAP peut réduire le temps et les efforts consacrés au prétraitement et à la sélection des variables, permettant aux data scientists de se concentrer sur l'interprétation des résultats et l'amélioration de la qualité du modèle.

En résumé, l'utilisation de CatBoost en combinaison avec SHAP permet d'améliorer l'interprétabilité des modèles tout en conservant les avantages de CatBoost pour gérer efficacement les variables catégorielles sans dummmification. **Cela conduit à des explications plus significatives et intuitives pour les utilisateurs finaux et les parties prenantes.**



Ce graphique classe les caractéristiques par ordre décroissant en fonction de leur influence sur les prédictions du modèle en calculant les valeurs SHAP moyennes pour chaque variable.

Étant donné que nous traitons d'une tâche de classification multi-classes, le tracé récapitulatif classe les variables en fonction de leur contribution globale à toutes les classes et code en couleur l'ampleur de chaque classe.

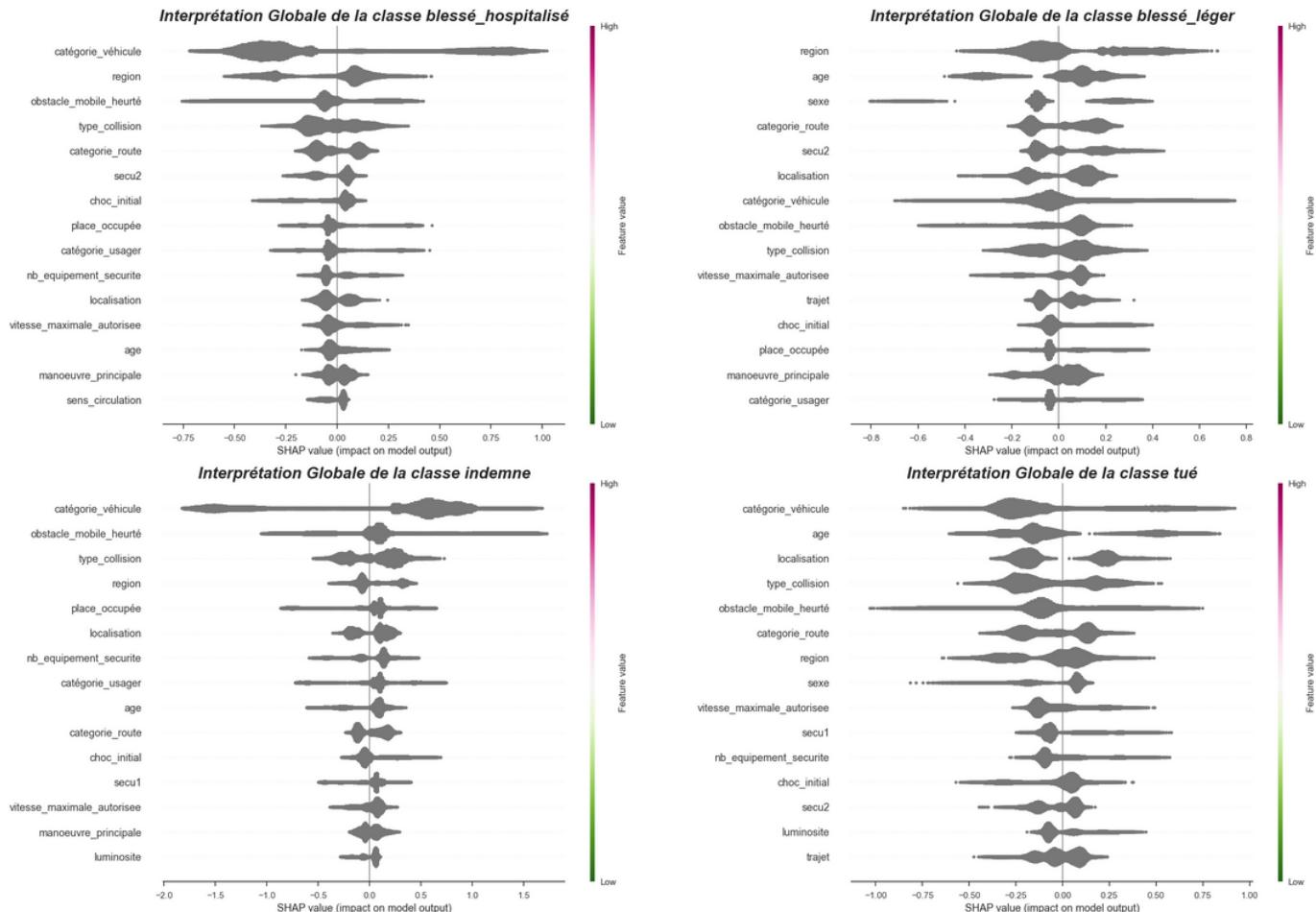
Ainsi, la catégorie du véhicule accidenté est la variable qui contribue le plus à la performance du modèle CatBoost multi-classes, et surtout pour la prédiction des personnes "indemnes".

03. Modélisation avec CatBoost

Quel est l'impact de chaque feature sur la prédiction de la classification multi-classes ?

Le tracé récapitulatif SHAP donne des informations détaillées en combinant l'importance des fonctionnalités avec ses effets.

Que signifie la couleur gris foncé ? CatBoost gère les variables catégorielles en interne pour cette raison, les caractéristiques catégorielles ne sont pas incluses dans le modèle SHAP en tant que versions codées numériques.



On remarque que les variables n'ont pas la même importance dans la prédiction des différentes gravités des accidents de la route.

La catégorie du véhicule et l'âge de l'usager sont les 2 variables plus importantes pour prédire si l'accidenté sera tué.

Pour les blessés hospitalisés nous retrouvons la catégorie du véhicule mais aussi la région qui est aussi la variable principale pour prédire la classe des blessés légers. Et pourtant, les différentes régions n'apparaissaient pas comme déterminantes dans les précédentes modélisations.

Enfin, la catégorie du véhicule et le fait d'avoir heurté un obstacle mobile sont les variables les plus importantes pour prédire si l'usager est indemne.

Voyons maintenant l'importance en détail de chacune de ses variables avec le tracé de dépendance des valeurs Shap.

03. Modélisation avec CatBoost

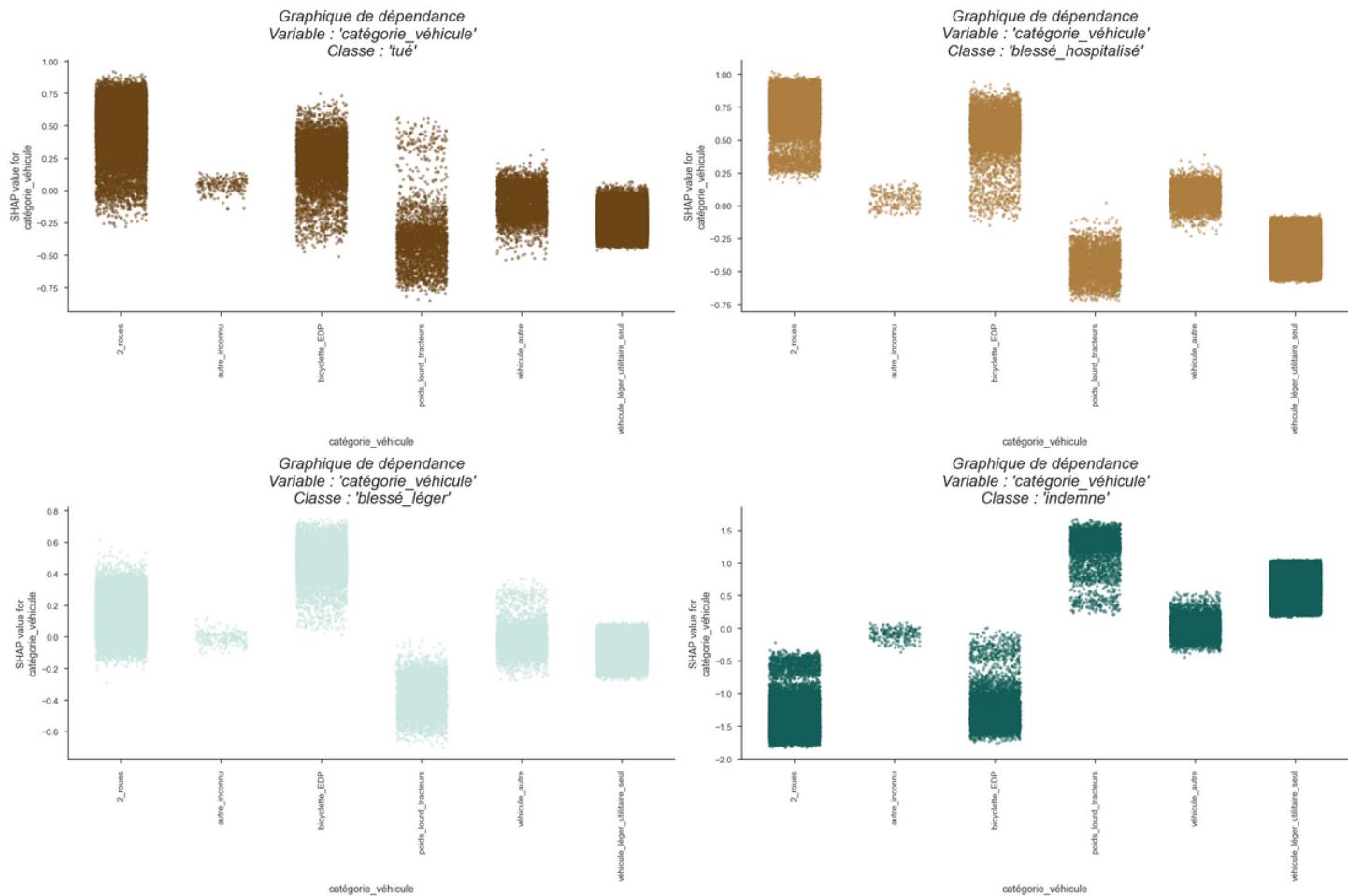
Interprétation des prédictions par classe : le graphique de dépendance

Le graphique de dépendance des valeurs SHAP est un type de graphique de dépendance qui montre comment la valeur de la fonction SHAP pour une variable d'entrée donnée varie en fonction de la valeur de cette variable.

Plus précisément, le graphique de dépendance des valeurs SHAP montre la relation entre la valeur de la fonction SHAP pour une variable d'entrée et la valeur de cette variable.

En utilisant la fonction SHAP pour chaque variable d'entrée, le graphique de dépendance des valeurs SHAP permet de visualiser **comment chaque variable d'entrée contribue à la prédiction du modèle**. Le graphique montre comment la valeur de la fonction SHAP change lorsque la valeur de la variable d'entrée change, **en gardant toutes les autres variables d'entrée constantes**.

Comment la catégorie du véhicule contribue-t-elle à la prédiction du modèle, toutes choses égales par ailleurs ?

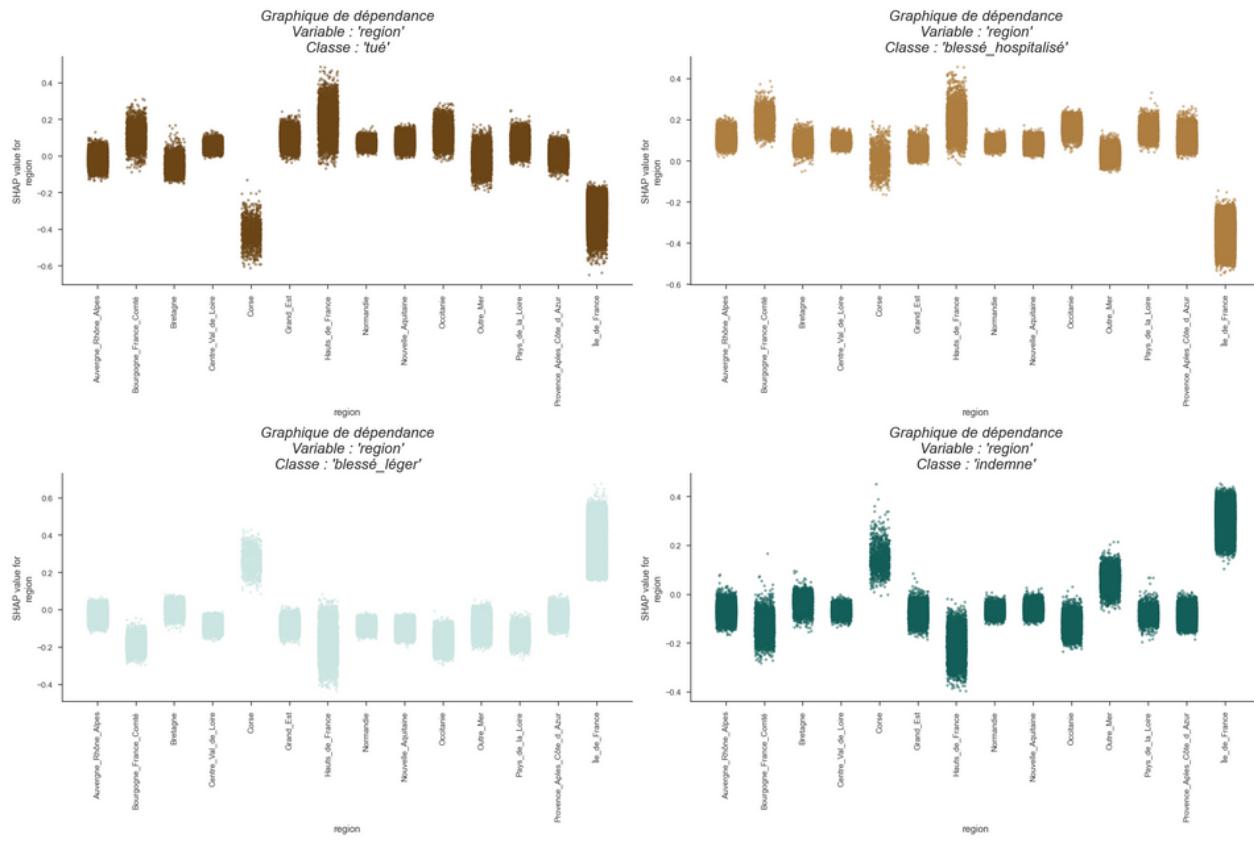


Le risque d'être tué, hospitalisé ou blessé légèrement augmente si vous roulez en 2 roues, en bicyclette et en engin de déplacement personnel (EDP) motorisé, lors de l'accident de la route.

A contrario, ce risque diminue si vous roulez dans un véhicule léger (voiture) et encore plus avec un poids lourd.

03. Modélisation avec CatBoost

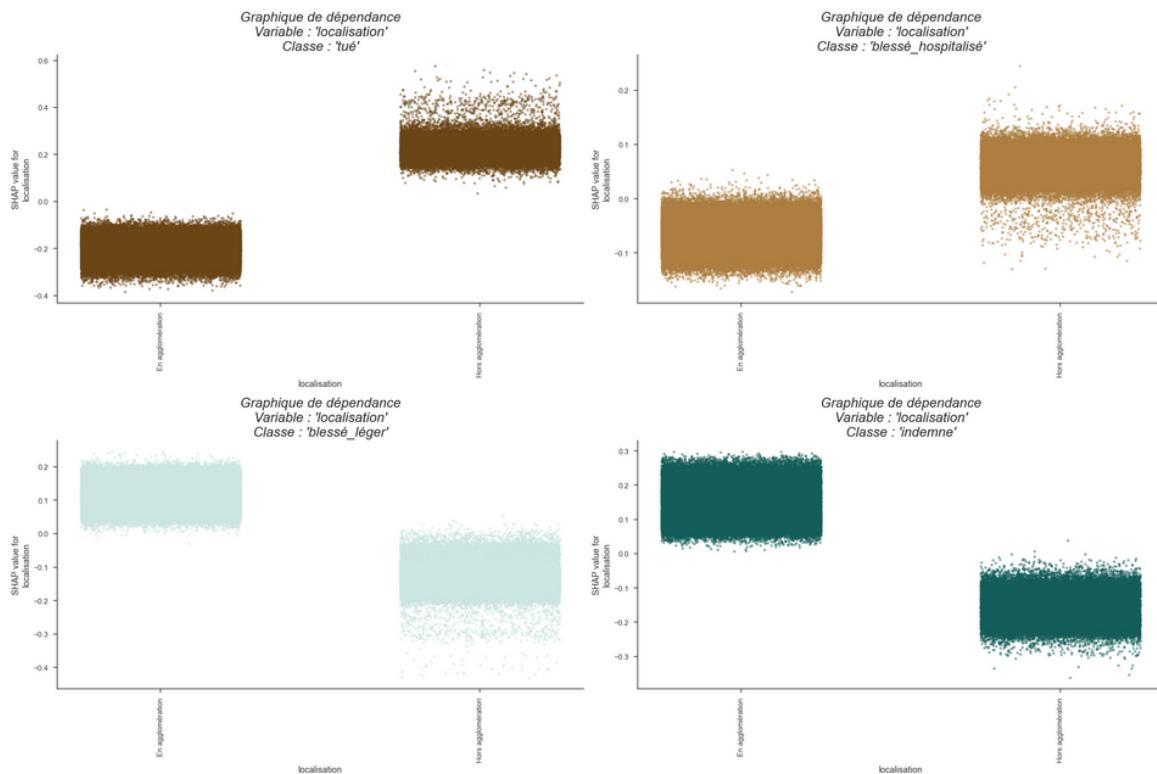
Comment la région contribue-t-elle à la prédiction du modèle, toutes choses égales par ailleurs ?



Le risque d'être tué ou hospitalisé augmente si l'accident a lieu dans la région Hauts-de-France. A contrario, ce risque diminue fortement si l'accident a lieu en Ile-de-France ou en Corse.

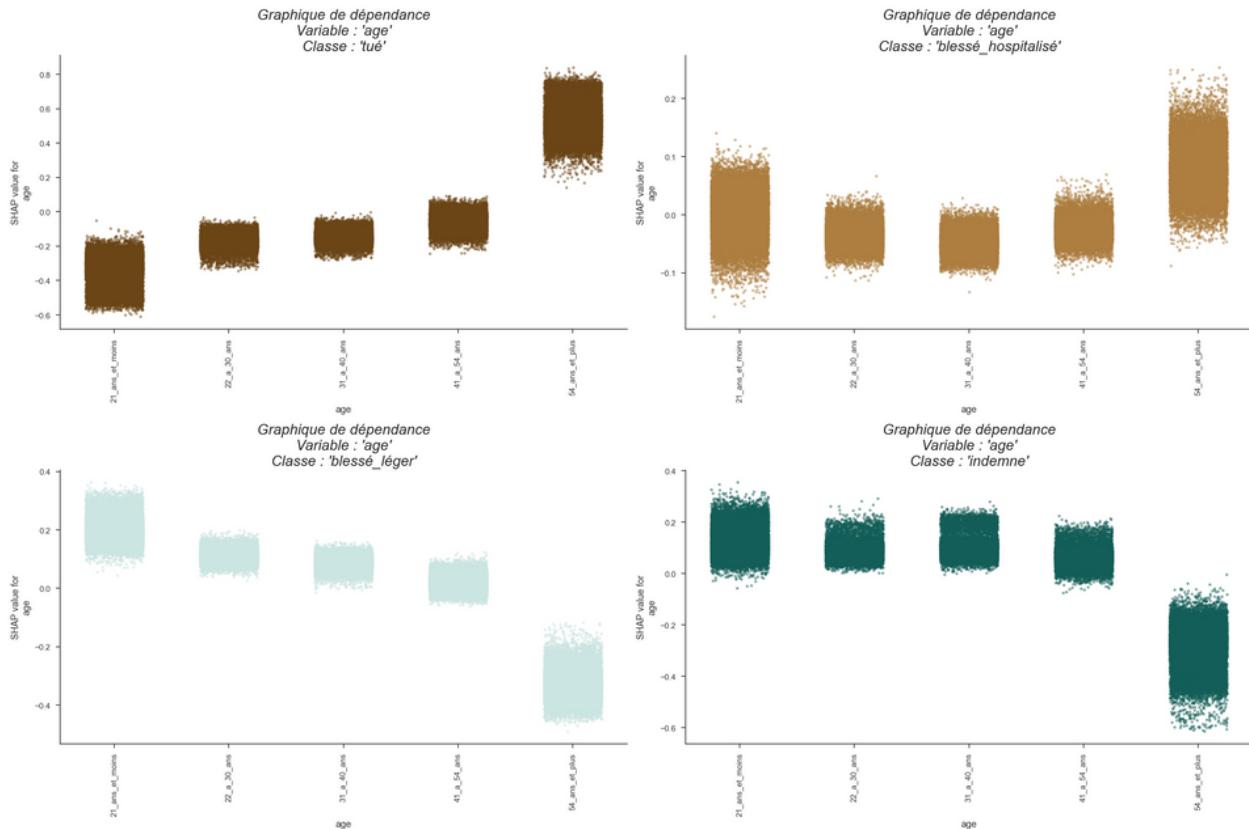
Comment la localisation de l'accident contribue-t-elle à la prédiction du modèle, toutes choses égales par ailleurs ?

Le risque d'être tué ou blessé hospitalisé augmente lorsque l'accident a lieu hors agglomération.



03. Modélisation avec CatBoost

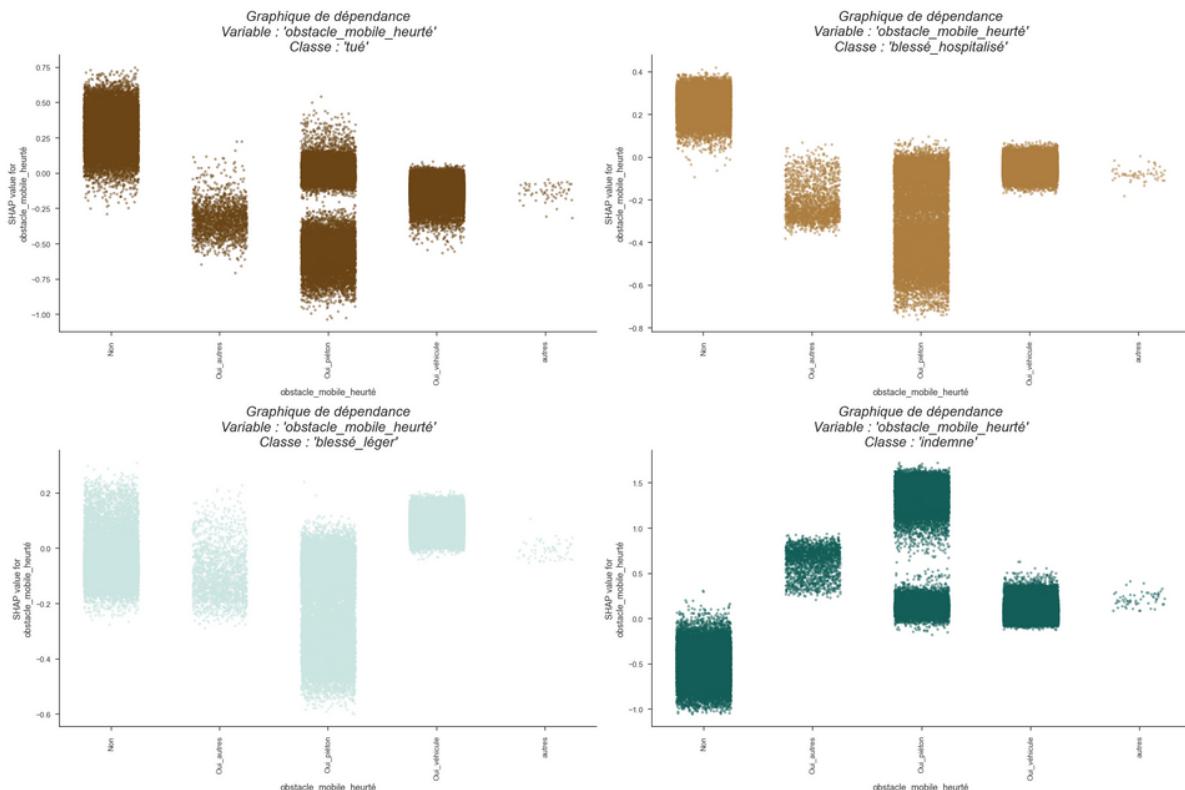
Comment l'âge contribue-t-elle à la prédition du modèle, toutes choses égales par ailleurs ?



Plus l'accidenté est âgé, plus le risque d'être tué est important.

Comment le fait de heurté un obstacle mobile contribue-t-elle à la prédition du modèle, toutes choses égales par ailleurs ?

Le risque d'être tué ou blessé hospitalisé augmente lorsqu'il n'y a pas d'obstacle mobile heurté.



03. Modélisation avec CatBoost

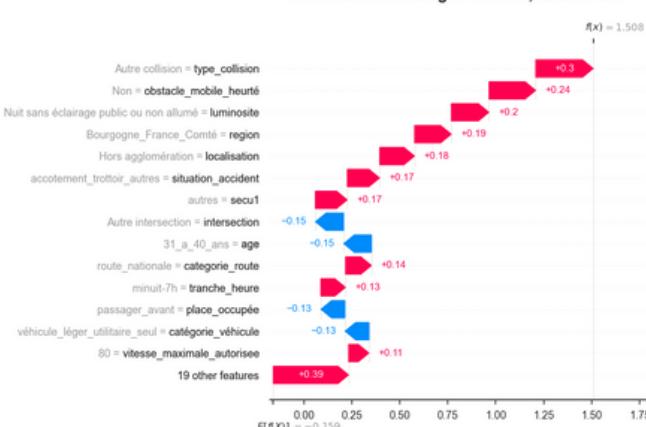
Comme pour les modélisations précédentes, nous pouvons interpréter localement, c'est-à-dire pour chaque observation les résultats du modèle, à l'aide des graphiques Force Plot, Waterfall Plot et Decision Plot.

Usager numero : 169170
 Classe Prédite de l'usager : tué
 Classe Réelle de l'usager : tué
 Il y a 58.5% de chances que l'usager appartienne à la classe 'tué'.

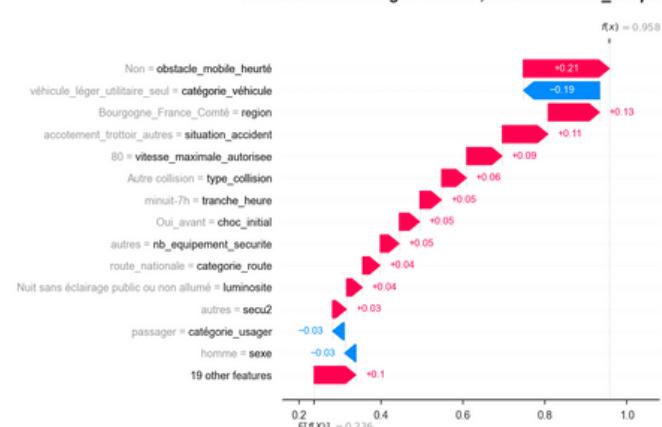


localisation = Hors agglomération | region = Bourgogne_France_Comté | luminosite = Nuit sans éclairage public ou non allumé | obstacle_mobile_heurté = Non | type_collision = Autre collision | intersection = Autre intersection | age = 31_a_40_ans | place_occu

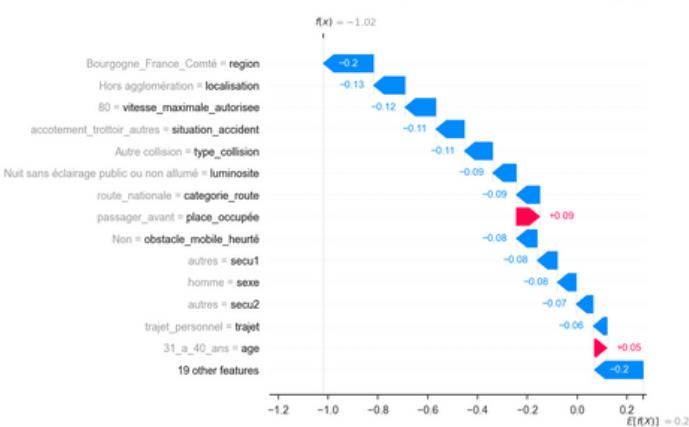
Waterfall Plot : usager 169170 ; classe 'tué'



Waterfall Plot : usager 169170 ; classe 'blessé_hospitalisé'



Waterfall Plot : usager 169170 ; classe 'blessé_léger'



Waterfall Plot : usager 169170 ; classe 'indemne'

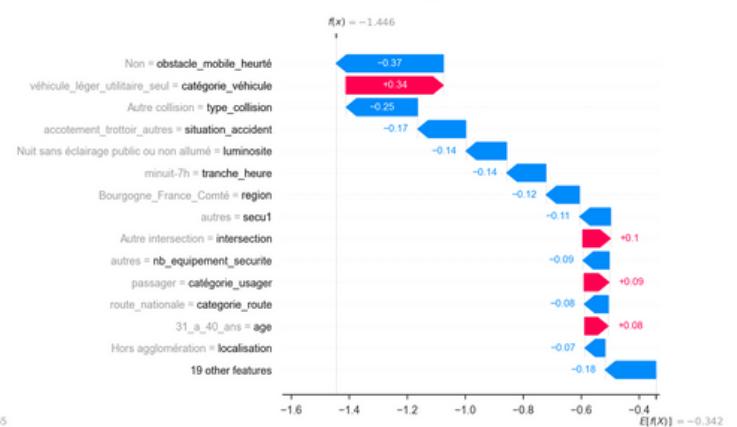
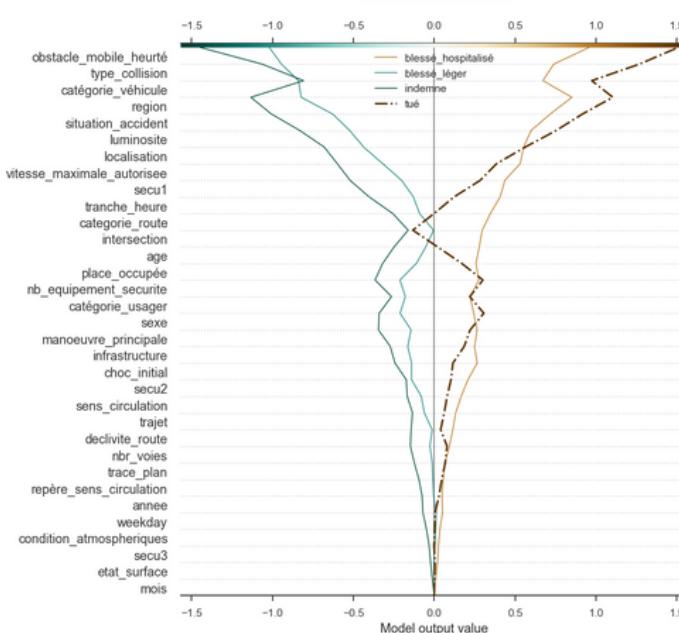


Diagramme de décision pour l'usager 169170
 Classe Réelle : tué
 Classe Prédite : tué



04. Conclusion de la Modélisation

Bien que CatBoost ne soit pas utilisé autant que certains autres algorithmes d'apprentissage automatique, il continue de gagner en popularité en raison de ses performances solides et de son traitement efficace des variables catégorielles.

Alors pourquoi n'est-il pas plus souvent utilisé dans le monde de la data ?

Voici ce que répond ChatGPT

Il existe plusieurs raisons pour lesquelles il n'est pas utilisé autant que d'autres algorithmes dans le monde de la data :

1. **Concurrence** : CatBoost fait face à une concurrence importante d'autres bibliothèques d'apprentissage automatique, telles que XGBoost et LightGBM, qui sont également très performantes et populaires. XGBoost et LightGBM ont été introduits avant CatBoost et ont donc bénéficié d'une adoption plus précoce et d'une plus grande notoriété dans la communauté.
2. **Familiarité** : Les praticiens de la science des données peuvent être plus familiarisés avec d'autres algorithmes et bibliothèques, comme scikit-learn, XGBoost et LightGBM. Ils peuvent donc choisir de rester avec les outils qu'ils connaissent déjà plutôt que d'adopter CatBoost.
3. **Documentation et support** : Bien que la documentation de CatBoost soit en constante amélioration, elle n'est pas aussi complète que celle de certains autres packages d'apprentissage automatique. En outre, la communauté d'utilisateurs de CatBoost est plus petite que celle d'autres bibliothèques, ce qui peut limiter les ressources d'aide et de support disponibles.
4. **Complexité** : CatBoost a des paramètres et des méthodes spécifiques pour traiter les variables catégorielles et pour effectuer la régularisation. Cela peut rendre l'algorithme un peu plus complexe à comprendre et à ajuster pour les praticiens par rapport à d'autres algorithmes plus simples.
5. **Interprétabilité** : Les modèles de boosting basés sur des arbres, tels que CatBoost, peuvent être moins interprétables que d'autres modèles, comme la régression logistique ou les arbres de décision simples. Dans certaines situations, les praticiens peuvent préférer des modèles plus simples et interprétables, même au détriment de la performance.
6. **Deep learning et autres approches** : Dans certaines applications, les réseaux de neurones profonds et d'autres approches d'apprentissage automatique peuvent surpasser les algorithmes basés sur des arbres, ce qui conduit les praticiens à choisir ces méthodes plutôt que CatBoost.

Pourtant il permet de traiter efficacement les données catégorielles sans avoir à les dummmifier. Ce qui a pour conséquence de se passer de toute la partie de pré-processing nécessaire aux autres modèles ?

Voici ce que répond ChatGPT

Vous avez absolument raison. CatBoost est particulièrement efficace pour gérer les variables catégorielles sans avoir besoin de les dummmifier ou de les transformer manuellement en variables numériques. Cette capacité est l'un des avantages clés de CatBoost par rapport à d'autres algorithmes d'apprentissage automatique, et c'est pourquoi il est de plus en plus populaire dans certaines situations. Voici quelques raisons pour lesquelles cette caractéristique n'a pas encore propulsé CatBoost à la première place en termes d'utilisation :

1. **Habitude et inertie** : Les data scientists et les chercheurs ont souvent leurs préférences et leurs habitudes en matière d'outils et de bibliothèques. S'ils ont déjà une méthode de prétraitement pour les variables catégorielles et sont satisfaits des performances de leurs modèles actuels, ils peuvent ne pas ressentir le besoin d'essayer CatBoost.
2. **Connaissance des avantages** : La communauté de la science des données peut ne pas être entièrement consciente des avantages offerts par CatBoost pour traiter les variables catégorielles. Étant donné que CatBoost est une technologie relativement récente par rapport à XGBoost et LightGBM, il peut falloir un certain temps pour que les praticiens découvrent et adoptent ces avantages.
3. **Cas spécifiques** : CatBoost est particulièrement avantageux lorsqu'il s'agit de traiter des ensembles de données avec un grand nombre de variables catégorielles. Cependant, dans les cas où il y a peu ou pas de variables catégorielles, les avantages de CatBoost peuvent ne pas être aussi évidents, et les praticiens peuvent préférer utiliser d'autres algorithmes avec lesquels ils sont plus familiers.
4. **Écosystème et intégration** : Les autres bibliothèques d'apprentissage automatique, comme scikit-learn, XGBoost et LightGBM, ont été intégrées dans un large éventail d'outils et de plateformes de science des données. Cette intégration facilite leur utilisation par les praticiens, tandis que l'adoption de CatBoost peut nécessiter un effort supplémentaire pour s'intégrer dans l'environnement existant.

En dépit de ces obstacles, CatBoost gagne progressivement en popularité et en reconnaissance, en particulier pour les problèmes qui impliquent un grand nombre de variables catégorielles. À mesure que les praticiens deviennent plus conscients des avantages de CatBoost et acquièrent de l'expérience avec la bibliothèque, il est probable que son adoption continuera de croître.



Les accidents de la Route en France

Projet Fil Rouge

Conclusion et Perspectives



Prédire la gravité des accidents de la route ?

La gravité des accidents de la route en France est un sujet de préoccupation pour les autorités, les citoyens et les associations de prévention routière. Les accidents de la route peuvent entraîner des conséquences graves, notamment des décès, des blessures corporelles et des dégâts matériels.

D'après l'Observatoire national interministériel de la sécurité routière, plusieurs facteurs contribuent à la gravité des accidents de la route en France, notamment la vitesse excessive, l'alcool, la drogue, la fatigue, le non-respect des règles de circulation et le manque d'entretien des véhicules. Les comportements irresponsables des usagers de la route augmentent le risque d'accidents graves.

Et c'est précisément pourquoi, puisque nous ne disposons pas de ces facteurs explicatifs dans notre jeu de données, que notre meilleur modèle de prédiction n'a pas, malgré tout, d'excellents résultats puisque l'Accuracy n'est que de 58% !