**LES ACCIDENTS ROUTIERS EN FRANCE**

Projet Fil Rouge

DataAnalyst – Mai 2024

Une image contenant plein air, Véhicule terrestre, arbre, Panneau de signalisation

Description générée automatiquement

**Rapport Final**

Table des matières

[I. Introduction du projet 3](#_Toc183933018)

[I.1. Contexte 3](#_Toc183933019)

[I.2. Objectifs 3](#_Toc183933020)

[II. Compréhension et manipulation des données 4](#_Toc183933021)

[II.1. Cadre et pertinence de la donnée 4](#_Toc183933022)

[II.1.1 Data Gouv 4](#_Toc183933023)

[II.1.2 Jeux de données issus de Kaggle 5](#_Toc183933024)

[II.1.3 Choix de la source de données et de la variable cible 5](#_Toc183933025)

[II.1.4 Analyse du jeu de données 5](#_Toc183933026)

[II.1.5 Concaténation des fichiers par types et fusion d’un DF 6](#_Toc183933027)

[II.2. Visualisations et statistiques 7](#_Toc183933028)

[II.3. Pre-processing et feature engineering 11](#_Toc183933029)

[II.3.1 Recherche de corrélation 11](#_Toc183933030)

[III. Modélisation 14](#_Toc183933031)

[III.1. Première modélisation 14](#_Toc183933032)

[III.2. Classification binaire 17](#_Toc183933033)

[III.2.1 Choix du modèle pour une variable cible binaire 17](#_Toc183933034)

[III.2.2 Interprétabilité du modèle 20](#_Toc183933035)

[III.3. Classification multiclasse 27](#_Toc183933036)

[III.3.1 Optimisation des hyperparamètres 27](#_Toc183933037)

[III.3.2 Resampling des données 28](#_Toc183933038)

[III.3.3 Interprétation multiclasse 29](#_Toc183933039)

[IV. Conclusions 32](#_Toc183933040)

[IV.1. Difficultés rencontrées lors du projet 32](#_Toc183933041)

[IV.2. Bilan 32](#_Toc183933042)

[IV.3. Suite du projet 32](#_Toc183933043)

[IV.4. Bibliographie 33](#_Toc183933044)

[IV.5. Annexes 33](#_Toc183933045)

# Introduction du projet

## Contexte

*Contexte d’insertion du projet dans votre métier ou intérêts professionnels.*

*Du point de vue technique.*

*Du point de vue économique.*

Chaque année depuis 2005, l'Observatoire National Interministériel de la Sécurité Routière (ONISR) met à disposition (via le site data.gouv) des bases de données relatives aux accidents corporels de la circulation routière.

Pour chaque accident corporel (accident survenu sur une voie ouverte à la circulation publique, impliquant au moins un véhicule et causant au moins une victime), des informations sont recueillies par les forces de l'ordre présentes sur les lieux. Ces informations sont consignées dans un document appelé bulletin d’analyse des accidents corporels. L'ensemble de ces bulletins constitue le fichier national des accidents corporels de la circulation, communément appelé « Fichier BAAC », géré par l'ONISR.

Les bases de données issues de ce fichier recensent tous les accidents corporels survenus au cours d'une année donnée en France métropolitaine, dans les départements et territoires d'Outre-mer avec une description simplifiée. Elles contiennent des informations sur la localisation de l'accident, les caractéristiques de l'accident et du lieu, ainsi que sur les véhicules impliqués et leurs victimes, qui sont rangées dans quatre bases de données : usagers, véhicules, lieux et caractéristiques.

Dans le cadre de notre projet « fil rouge » de notre formation de Data Analyst, nous sommes amenés à manipuler ces bases de données.

## Objectifs

*Quels sont les principaux objectifs à atteindre ? Décrivez en quelques lignes.*

*Pour chacun des membres du groupe, préciser le niveau d’expertise autour de la problématique adressée ?*

*Êtes-vous entré en contact avec des experts métiers pour affiner la problématique et les modèles sous-jacents ? Si oui, détaillez l’apport de ces intéractions.*

*(Avez-vous connaissance d’un projet similaire au sein de votre entreprise, ou bien dans votre entourage ? Quel est son état d’avancement ? En quoi vous a-t-il aidé dans la réalisation de votre projet ? En quoi votre projet contribue-t-il à l’améliorer ?)*

L’objectif de ce projet est d’essayer de prédire la gravité des accidents routiers en France. Les prédictions seront basées sur les données de l’ONISR.

# Compréhension et manipulation des données

## Cadre et pertinence de la donnée

Dans le cadre de notre projet sur les accidents routiers en France et leur gravité, nous avions accès à deux sources de données principales :

- <https://www.data.gouv.fr>

- <https://www.kaggle.com>

### **Data Gouv**

Sur la plateforme Data.gouv.fr, nous disposions des données couvrant la période de 2005 à 2022, réparties en plusieurs fichiers :

- Usager\_annee.csv

- Véhicules\_annee.csv

- Lieux\_annee.csv

- Caractéristiques\_annee.csv

- Véhicule\_immatricules\_baac\_annee.csv

- Des fichiers descriptifs en format PDF, variables selon les années

- Des bases de données déjà agrégées

#### Récupération des fichiers

Pour extraire ces fichiers, nous avons utilisé l'outil « Octoparse » afin d'obtenir un fichier CSV contenant l'ensemble des liens vers les différents fichiers de données.

Ensuite, à l'aide de Python, nous avons développé une fonction spécifique permettant de télécharger et d'extraire automatiquement toutes les données disponibles sur la plateforme (voir Annexe 1 pour le code).

#### Analyse des données

Nous avons utilisé le modèle de template pour examiner en détail les colonnes de chaque fichier, en fournissant des informations telles que :

- Le nom de la colonne

- La description de la colonne

- Le type de données

- Le taux de valeurs manquantes (NaN)

- La distribution des valeurs

- Des observations et remarques pertinentes

Cette analyse a révélé que certaines colonnes ont évolué au fil du temps, n'apparaissant qu'à partir de certaines années, ou cessant d'être collectées à partir d'une certaine date.

Fichier Caractéristiques: Ce fichier décrit les circonstances générales de chaque accident. Il contient 17 colonnes, dont deux sont des doublons (\*Accident\_ID\* et \*Num\_ACC\*). Nous avons constaté un grand nombre de valeurs manquantes dans les données géographiques (coordonnées GPS, adresses).

Fichier Lieux : Ce fichier décrit le lieu principal de l'accident, y compris lorsque l'accident se produit à une intersection. Il contient 19 colonnes. De nombreuses valeurs sont manquantes, notamment pour des variables liées à la route (ex. indices de route comme 2bis, 3ter), ou des détails comme la distance par rapport à une borne, la largeur du terre-plein central, et la vitesse maximale autorisée.

Fichier Véhicules : Ce fichier décrit les véhicules impliqués dans l'accident. Il contient 11 colonnes. De nombreuses valeurs sont manquantes, en particulier pour les variables liées au nombre d'occupants, à l'identifiant du véhicule et au type de moteur.

Fichier Usager : Ce fichier décrit les personnes impliquées dans l'accident. Il contient 17 colonnes. Les valeurs manquantes concernent principalement les variables relatives à la sécurité (celles-ci ayant évolué dans le temps), l'identifiant du véhicule et celui de l'usager.

Nous avons décidé de ne pas inclure, pour le moment, le fichier Véhicule\_immatriculé\_baac dans notre analyse.

### Jeux de données issus de Kaggle

Le jeu de données fourni par Kaggle est déjà agrégé et contient 55 colonnes. Toutefois, il ne couvre que la période jusqu'à 2016.

Détail fichier de Margot

### Choix de la source de données et de la variable cible

Après cette première phase d'analyse, nous avons déterminé que la gravité de l'accident serait notre variable cible pour l'étude.

Après une discussion collective, nous avons fait le choix de conserver les données issues de Data.gouv.fr pour deux raisons principales :

Maîtrise des données : En sélectionnant cette source, nous avons un contrôle total sur la construction du jeu de données, incluant le choix des années à analyser et des jointures à réaliser.

Actualité des données : Les données sur Data.gouv.fr sont plus récentes. Nous estimons que l'évolution des infrastructures routières, les progrès en matière de sécurité des véhicules, ainsi que les mesures mises en place par l'État en faveur de la sécurité routière, sont des éléments importants à prendre en compte dans l'analyse de la gravité des accidents.

### **Analyse du jeu de données**

Une fois la source choisie, nous avons analyser avec Python plus en détails les composantes que nous avions à notre disposition en vue de créer un DF pour la data visualisation.

L'objectif de l'analyse a été de traiter et d'examiner des fichiers CSV provenant de différentes années et de divers types.

#### Contexte et Objectif

L'analyse portait sur plusieurs fichiers CSV situés dans un dossier spécifique, couvrant les années de 2005 à 2022. Les fichiers appartenaient à différents types, notamment des caractéristiques, des lieux, des usagers, des véhicules, et des immatriculations de véhicules.

#### Méthodologie

Les fichiers ont été traités en essayant plusieurs séparateurs (virgule, point-virgule, tabulation) pour assurer une lecture correcte des données. Pour chaque fichier, un DataFrame a été créé et les statistiques descriptives ont été calculées pour chaque colonne. Ces statistiques incluent le type de données, le taux de valeurs manquantes, le mode, les valeurs minimales et maximales, la moyenne, la médiane, et l'écart-type.

#### Résultats

Résultats d'Analyse : Un résumé détaillé a été produit pour chaque fichier, incluant le nombre de colonnes et de lignes, les noms des colonnes, et les informations sur le succès ou l'échec de la lecture du fichier.

Statistiques des Colonnes: Des statistiques détaillées ont été extraites pour chaque colonne des fichiers analysés, fournissant des informations clés sur la distribution des données.

#### Exportation des Résultats

Les résultats d'analyse et les statistiques des colonnes ont été préparés pour une éventuelle exportation en fichier Excel, facilitant la présentation et l'interprétation des données.

En conclusion, cette analyse a permis de compiler des informations précieuses sur la structure et la qualité des données contenues dans les fichiers CSV, offrant une base solide pour des analyses ultérieures.

### Concaténation des fichiers par types et fusion d’un DF

Une fois, le choix de la source, et une fois l’analyse faite, nous avons fait le choix de ne travailler que sur un seul dataframe.

Tout d’abord nous avons réalisé plusieurs DF afin de concaténer par année et par type de fichier.

Nous avons remarqué lors de l’analyse que dans le fichier des caractéristiques, une variable était un doublon depuis une certaines années, à savoir Num\_Acc et Accident\_id, nous avons donc remplacer les valeurs manquantes et supprimer la colonne ‘Accident\_Id’.

Notre variable cible étant dans le fichier des usagers, nous sommes partis de ce fichier pour appliquer des merges sur les différentes clés présentes dans les deux fichiers afin de doubler les lignes inutilement.

Nous avons utilisé un merge sur ‘Num\_Acc’, ‘id\_vehicule’, et ‘num\_veh’ entre usagers et véhicules. Puis ‘Num\_acc’, pour les lieux et caractéristiques.

Nous avions donc notre ‘merge\_df’ pour tous les fichiers et toutes les années.

Notre objectif étant de prédire la gravité d’un accident, nous avons choisi de limiter notre jeu de données à trois années, soit de 2020 à 2022.

Ce choix est motivé par le souhait d’optimiser nos performances matérielles, car nous travaillons déjà sur un volume de données conséquent, comptant 361 205 lignes.

Par ailleurs, nous avons estimé que la gravité d’un accident ne pouvait pas être comparée de manière uniforme entre aujourd’hui et 2005. Cela s’explique notamment par les évolutions dans les travaux de voirie et les progrès technologiques des véhicules.

Pour ces raisons nous avons donc filtré notre ‘merge\_df’ de 2020 à 2022 pour se nommer df\_3, que nous avons sauvegardé en csv.

## Visualisations et statistiques

*Avez-vous identifié des relations entre différentes variables ? Entre variables explicatives ? et entre vos variables explicatives et la/les cible(s) ?*

*Décrivez la distribution de ces données, répartition, outliers.. (pré/post traitement éventuellement)*

*Présentez les analyses statistiques utilisées permettant de confirmer les informations présentes sur les graphiques.*

*Tirer des conclusions des éléments relevés ci-dessus permettant de se projeter dans la partie de modélisation*

Après avoir fusionné nos différents fichiers, nous avons procédé à la visualisation des données afin d’identifier des relations entre les différentes variables, observer la distribution des données, leur composition.

Afin de pouvoir réaliser la data visualisation des données de manière efficace, nous avons réalisé un premier pre-processing dont les étapes sont résumées ci-dessous :

* Remplacement des valeurs « -1 » (signifiant que la donnée n’a pas été renseignée par les forces de l’ordre) par un NaN ;
* Suppression des variables présentant un taux de valeurs manquantes trop important (gestion des NaN peu réalisable) ;
* Suppression des variables concernant les piétons. La plupart de ces variables sont inutilisables du fait du taux de NaN trop important ;
* Pour la localisation des accidents, du fait d'avoir les latitudes et longitudes, nous pouvons supprimer les autres variables. Nous gardons tout de même la variable dep (département) et com (commune).

Une image contenant texte, capture d’écran, ligne, Tracé

Description générée automatiquement

Figure 1 : Proportion de valeurs manquantes par variables

Nous représentons ci-dessous le nombre d’accidents par jour durant la période d’étude considérée (2020 à 2022). De manière générale, le nombre d’accidents est stable sur ces années et oscille autour de 300 accidents par jour. A noter toutefois une forte diminution sur les mois de mars et avril 2020, correspondant à la période de confinement liée au Covid-19.

Une image contenant capture d’écran, texte, Tracé

Description générée automatiquement

Figure 2 : Répartition du nombre d'accidents par jour de 2020 à 2022

Le graphique suivant confirme également l’impact du confinement et des restrictions de circulation sur l’année 2020. Concernant les années 2021 et 2022, nous pouvons observer une répartition du nombre d’accidents par mois assez homogène.

Une image contenant texte, capture d’écran, ligne, nombre

Description générée automatiquement

Figure 3 : Répartition du nombre d'accidents par mois entre 2020 et 2022

Si nous nous intéressons à la variable gravité, nous pouvons remarquer que une répartition homogène entre les années. La variable gravité est décomposée en 4 classes : l’usager a été après l’accident soit indemne, soit tué, soit blessé hospitalisé ou soit blessé léger. Nous remarquons une hétérogénéité entre les classes avec seulement 2.5% des usagers tués lors d’un accident contre 44% indemne ou 38% étant blessé léger.

Une image contenant texte, capture d’écran, diagramme, Tracé

Description générée automatiquement

Figure 4 : Répartition de la gravité des accidents entre 2020 et 2022

Il est également possible de représenter la distribution de l’âge des usagers accidentés suivant la gravité de leur accident.

Une image contenant texte, diagramme, Tracé, ligne

Description générée automatiquement

Figure 5 : Distribution de l'âge des usagers lors de l'accident

On remarque un pic d'accidents chez les personnes nées aux alentours des 23 ans. Comme les données que nous utilisons proviennent des années 2020, 2021 et 2022, et que les jeunes conducteurs comptent parmi les catégories les plus à risque, ce résultat semble cohérent.

A noter toutefois d’une distribution clairement asymétrique et non normale. Un traitement sera à réaliser en vue du Machine Learning (la plupart des algorithmes de Machine Learning se basent sur l’hypothèse que les variables utilisées suivent une loi normale). Un exemple de traitement possible sera de discrétiser cette variable en créant des groupes.

## Pre-processing et feature engineering

*Avez-vous eu à nettoyer et à traiter les données ? Si oui, décrivez votre processus de traitement.*

*Avez-vous dû procéder à des transformations de vos données de type normalisation/standardisation ? Si oui, pourquoi ?*

*Envisagez-vous des techniques de réduction de dimension dans la partie de modélisation ? Si oui, pourquoi ?*

La première étape du projet nous a permis de comprendre le jeu de données, de visualiser certaines tendances ainsi que d’avoir déterminé notre variable cible : la gravité des accidents.

Afin de pouvoir utiliser le jeu de données et ainsi travailler avec les différents algorithmes de Machine Learning, nous avons dû transformer nos jeux de données. Les raisons principales sont les suivantes :

* Les algorithmes de Machine Learning nécessitent d’avoir en entrée des formats spécifiques, comme par exemple une classification binaire des variables, des formats numériques.
* Discrétiser certaines variables en sous-ensembles afin d’améliorer les résultats des algorithmes.
* Sélectionner les variables réellement utiles pour les modèles.

Parmi les différentes transformations réalisées, nous avons pu :

* Discrétiser les éléments de sécurité (variable ‘secu1’, ‘secu2’ et ‘secu3’). Ces variables contenaient une liste d’équipements de sécurité qui étaient présents lors de l’accident. Cela pouvait correspondre au port du casque, de la ceinture, l’utilisation du airbag. Les forces de l’ordre ont donc la possibilité de faire remonter la présence de jusqu’à 3 équipements. Afin de corriger une part de valeurs manquantes et pour avoir des variables utilisables, nous avons créé une variable pour chaque équipement, indiquant si celui-ci était présent lors de l’accident.
* Regrouper des modalités ensemble de certaines variables afin de recréer des sous-ensembles plus simple et ainsi améliorer les performances des modèles. Par exemple, la variable ‘place’ indique la position de l’usager dans le véhicule lors de l’accident. Les modalités possibles vont de 1 à 9, avec pour chaque numéro une position dans une voiture. Ces différentes modalités ont été regroupées en 3 sous-ensemble ;chaque sous-ensemble étant une nouvelle variable créée : ‘conducteur’, ‘passager\_avant’, ‘passager-arrière’.

### Recherche de corrélation

Afin d’avoir un premier aperçu sur les liens entre les variables et avec la variable cible, nous avons réalisé une matrice de corrélation afin de faire apparaitre l’importance de certaines variables vis-à-vis de la gravité de l’accident. Cette matrice de corrélation est présentée ci-dessous via une heatmap.

Une image contenant texte, capture d’écran, ligne, diagramme

Description générée automatiquement

Figure 6 : Matrice de corrélation suivant les variables du jeu de données

Concernant la variable ‘gravité’, nous pouvons voir une corrélation plus marquée avec les variables suivantes :

* Port de la ceinture ;
* Si le véhicule impliqué est une moto ;
* Le type de motorisation ;
* L’obstacle heurté ;
* L’année de naissance de l’usager ;
* …

Il est également possible de voir que certaines variables (autre que la variable cible) sont corrélées entre elles. C’est le cas notamment du port du casque et si l’usager était à moto. Cette représentation permet de commencer à détecter des problème de multicolinéarité. Cela peut créer de mauvais comportement au modèle ou rendre son interprétation plus difficile dans le cas ou deux features sont fortement corrélées entre elles et avec la variable cible. Alors il sera plus difficile de savoir quelle variable influence vraiment la variable cible.

Nous avons également étudié la présence d’un lien statistique entre les variables explicatives et la variable cible. Nos variables sont des variables catégorielles. Nous avons donc utilisé le test du khi 2. Les hypothèses sont décrites ci-dessous :

* H0 : La variable est indépendante de la variable cible (gravité)
* H1 : La variable n'est pas indépendante de la variable cible (gravité)

Nous fixons la p-value à 0.05 : si la valeur de p est inférieure à 0.05, cela signifie que nous pouvons rejeter l’hypothèse H0 et affirmer que le résultat soutient l’hypothèse H1.

Après réalisation du test sur les variables, il en ressort que les p-values sont toutes nulles. Il existe donc bien une relation statistique entre les variables qualitatives et la variables cible. Cela signifie qu’il est donc possible et pertinent d’utiliser ces différentes variables pour la modélisation. En revanche, cette méthode ne peut être utilisée simplement ainsi pour sélectionner les variables pertinentes pour les modèles. Il sera donc nécessaire de regarder quelles sont les features les plus utiles à chaque modèle pour permettre de répondre à cette question.

# Modélisation

L’objectif de cette deuxième 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 suivante : « Pourquoi le modèle a-t-il prédit cette valeur ? ». L’objectif d’un Data Analyst est de permettre de comprendre les résultats obtenus et de pouvoir le retranscrire au client.

## Première modélisation

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

La variable cible choisie pour la modélisation est la variable « gravité ». Cette dernière est présente quatre classes : indemnes, blessé léger, blessé grave et tué. Nous sommes donc en présence d’un problème de classification multiclasse.

L’une des premières caractéristiques concernant la variable cible est la proportion hétérogène entre ses classes. Ainsi, nous pouvons constater que les usagers tués lors d’un accident routier représente 2.5% de notre dataset.

Une image contenant texte, capture d’écran, diagramme, Police

Description générée automatiquement

A la fin du premier pre-processing, nous avons lancé la modélisation afin d’avoir des premiers résultats et donc un premier retour sur notre travail de préparation des données. L’objectif n’est pas d’avoir un modèle parfait mais des pistes d’amélioration.

Nous avons décomposé notre jeu de données en 2 sous-ensembles : 80% pour entraîner nos modèles de prédiction et 20% pour les tester. L’utilisation de la fonction « train\_test\_split » de la librairie sklearn permet de diviser 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.

La modélisation s’est faite en utilisant dans un premier temps deux modèles vus en cours afin d’appliquer les notions apprises. Le RandomForestClassifier et le GradientBoostingClassifier ont été utilisés avec des hyperparamètres renseignés de manière arbitraire (quelques tests ont été faits pour voir les impacts sur les résultats).

Ci-dessous la matrice de confusion et le rapport de classification obtenus pour le RandomForest :

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

En première approche, nous pouvons voir un score important pour la précision, le recall et le F1-score sur les classes 1 (indemne) et 4 (hospitalisé léger). Cela est cohérent car il s’agit des classes majoritaires.

Ci-dessous la matrice de confusion et le rapport de classification obtenus pour le GradientBoosting. Cet algorithme a été choisi suite à des recherches sur les différents modèles de Machine Learning. Il apparait que ce modèle présente depuis ces derniers temps des meilleures performances. A noter cependant que cet algorithme est très sensibles aux profondeurs des arbres.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

En première approche, nous pouvons voir que cette modélisation présente des résultats légèrement meilleurs que le RandomForest.

Afin d’avoir une vision plus complète, nous avons réalisé la modélisation en utilisant différents modèles de manière automatique via une pipeline déjà programmée via la libraire « lazypredict ». Nous reviendrons dessus plus en détail dans la prochaine partie.

Ci-dessous les performances obtenues sur l’ensemble d’entrainement :

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Ci-dessous les performances obtenues sur l’ensemble de test :

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Pour résumer cette première étape de modélisation, il a été constaté que :

* Les tests de corrélation ont mis en évidence des variables déterminantes sur lesquelles il a été possible de construire une modélisation ;
* Les scores plafonnant des modèles testés. Cela peut signifier que les variables pertinentes sont absentes du jeu de données ou un besoin d’un meilleur traitement des données ou d’une recherche de meilleurs hyperparamètres.

Une première étape d’optimisation consistera de voir l’impact d’un meilleur rééquilibrage des données.

## Classification binaire

Comme vu précédemment, l’une des premières caractéristiques concernant la variable cible est la proportion hétérogène entre ses classes. Ainsi, nous pouvons constater que les usagers tués lors d’un accident routier représente 2.5% de notre dataset.

Une image contenant texte, capture d’écran, diagramme, Police

Description générée automatiquement

Les classes « Tué » et « Hospitalisé grave » sont sous-représentées dans le dataset. Afin de ne pas considérer dans un premier temps les effets d’un déséquilibre entre classes, nous redécomposons la variable en deux classes : « indemne » ou « blessé » avec des proportions similaires. Ainsi, il sera possible d’utiliser les modèles de classification binaire sur des données équilibrées.

Une image contenant texte, capture d’écran, diagramme, Police

Description générée automatiquement Une image contenant texte, capture d’écran, Police, diagramme

Description générée automatiquement

### Choix du modèle pour une variable cible binaire

#### Test de différents modèles

Pour identifier les modèles les plus appropriés, nous avons testé plusieurs algorithmes de base, notamment les classificateurs suivants :

* DummyClassifier,
* LogisticRegression,
* RandomForestClassifier,
* BaggingClassifier,
* AdaBoostClassifier,
* DecisionTreeClassifier,
* ExtraTreesClassifier,
* KNeighborsClassifier,
* XGBClassifier,
* LGBMClassifier,
* GradientBoostingClassifier.

Cette étape s’est faite de manière automatisée via une pipeline de la librairie « lazypredict ». A noter également que nous avons lancé la modélisation d’un RandomForest et GradientBoosting manuellement afin d’appliquer ce qui a été vu lors de notre formation.

Les performances de chaque modèle ont été évaluées à l'aide de métriques de précision telles que l'exactitude (accuracy), le score F1 (F1-score) et l'aire sous la courbe ROC (ROC-AUC), calculées sur les ensembles d'entraînement et de test.

* L'exactitude a permis d'estimer la précision globale de chaque modèle ;
* Le score F1 a été utilisé pour équilibrer les notions de précision et de rappel ;
* L'aire sous la courbe ROC a mesuré la capacité du modèle à différencier les classes.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Modèle** | **Train Accuracy** | **Test Accuracy** | **Train F1\_score** | **Test F1\_score** | **Train ROC\_AUC** | **Test ROC\_AUC** | **Train time** | **Test time** |
| XGBClassifier | 0.75 | 0.74 | 0.75 | 0.74 | 0.75 | 0.74 | 3.7 | 2.6 |
| LGBMClassifier | 0.75 | 0.74 | 0.75 | 0.74 | 0.74 | 0.74 | 3.1 | 2.0 |
| GradientBoostingClassifier | 0.74 | 0.74 | 0.74 | 0.74 | 0.74 | 0.73 | 28.8 | 27.2 |
| LogisticRegression | 0.73 | 0.73 | 0.73 | 0.73 | 0.73 | 0.73 | 1.3 | 0.9 |
| AdaBoostClassifier | 0.73 | 0.73 | 0.73 | 0.73 | 0.73 | 0.73 | 11.6 | 9.6 |
| RandomForestClassifier | 0.78 | 0.73 | 0.78 | 0.73 | 0.78 | 0.73 | 34.2 | 29.9 |
| ExtraTreesClassifier | 0.78 | 0.73 | 0.79 | 0.73 | 0.79 | 0.73 | 45.6 | 39.4 |
| BaggingClassifier | 0.78 | 0.73 | 0.78 | 0.73 | 0.78 | 0.72 | 11.0 | 9.8 |
| DecisionTreeClassifier | 0.78 | 0.72 | 0.79 | 0.72 | 0.79 | 0.72 | 2.2 | 1.8 |
| KNeighborsClassifier | 0.73 | 0.71 | 0.73 | 0.71 | 0.73 | 0.70 | 83.1 | 20.7 |
| DummyClassifier | 0.56 | 0.56 | 0.40 | 0.40 | 0.50 | 0.50 | 0.8 | 0.5 |

Le tableau des résultats ci-avant présente les performances de chaque modèle sur les ensembles d'entraînement et de test.

Il apparaît que certains modèles, comme le RandomForestClassifier, l'ExtraTreesClassifier et le BaggingClassifier présentent une tendance au surapprentissage (overfitting), se traduisant par des performances légèrement inférieures sur l'ensemble de test par rapport à l'ensemble d'entraînement. Nous pouvons également voir que le DummyClassifier a présenté un temps d’exécution plus long que les autres modèles.

En revanche, des modèles tels que le XGBClassifier, LGBMClassifier et le GradientBoostingClassifier ont montré des performances à la fois stables et élevées sur les ensembles d'entraînement et de test.

Concernant la notion de surapprentissage ou overfitting ou surajustement, nous rappelons ci-après sa définition. 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.

#### Rappel sur les métriques

Afin de bien comprendre les performances des modèles et du tableau présenté, nous rappelons ci-dessous la définition des métriques utilisées.

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.

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.

#### Caractéristiques des modèles utilisés

Afin de bien comprendre le résultat des modèles, nous avons également réalisé une recherche bibliographique sur les caractéristiques des modèles utilisés, avec leurs avantages et 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.

GradientBoostingClassifier : Ce modèle utilise un algorithme de gradient boosting pour améliorer la précision des prédictions. Il est également efficace pour les problèmes de classification binaire et multi-classes. Cependant, il peut être sensible au bruit et au sur-ajustement des données.

### Interprétabilité du modèle

#### Définition de l’interprétabilité

Le rôle d’un Data Analyst est de pouvoir, entre autre, expliquer la donnée à ses clients. Pour cela, il convient de comprendre les modèles de Machine Learning et les processus qui les ont conduits à faire les prédictions. La notion d’interprétabilité des modèles est donc importante.

L’interprétabilité correspond à la mesure dans laquelle un être humain peut prédire de manière cohérente le résultat d’un modèle. Plus l'interprétabilité d'un modèle de Machine Learning est élevée, plus il est facile pour un individu de comprendre le raisonnement derrière certaines décisions ou prédictions. Un modèle est plus facilement interprétable qu'un autre si ses décisions sont plus faciles à comprendre pour un humain.

Les mesures de performances sont importantes mais elles n'expliquent pas les décisions prédictives d'un modèle. Au fil du temps, les performances peuvent diminuer car le modèle n'est plus optimal par rapport à divers changements de distribution des variables explicatives. Il est donc important de comprendre ce qui pousse un modèle à prendre certaines décisions afin d'identifier les features auxquels un modèle serait très sensible et dont il faut surveiller la distribution au fil du temps.

Parfois un modèle peut réaliser de bonnes performances, mais après déploiement, on peut se rendre compte qu'il ne correspond pas à ce qu'on attendait. Il peut avoir une différence d'interprétation entre ce qu'on demande au modèle et sa compréhension de notre formulation.

Les trois aspects les plus importants de l'interprétation des modèles sont expliqués par les questions suivantes :

Qu'est-ce qui motive les prédictions du modèle ? Interroger notre modèle et découvrir les interactions cachées entre les différentes features pour avoir une idée de celles qui pourraient être les plus importantes dans les prises de décisions du modèle. Cela permet de garantir l'équité du modèle.

Pourquoi le modèle a-t-il pris une certaine décision ? Valider et justifier pourquoi certaines caractéristiques clés ont été à l'origine de certaines décisions prises par un modèle lors des prédictions. Cela permet de garantir la responsabilité et la fiabilité du modèle.

Comment pouvons-nous faire confiance aux prédictions du modèle ? Evaluer et valider la façon dont un modèle prend des décisions pour tout point de données. Ainsi, on peut démontrer et expliquer facilement que le modèle fonctionne comme prévu. Cela permet de garantir la transparence du modèle.

Nous retrouvons deux types d'interprétabilité : l'interprétabilité intrinsèque et l'interprétabilité post hoc.

Interprétabilité intrinsèque : Propre aux modèles eux-mêmes et découle de leur structure simple et transparente. Des exemples incluent les régressions linéaires (où les coefficients sont directement interprétables), les arbres de décision (où l'on peut suivre les branches pour comprendre les décisions), et les modèles basés sur des règles. Ces modèles sont conçus pour être compréhensibles par les humains sans nécessiter de méthodes d'analyse supplémentaires.

Interprétabilité post hoc : Pour les modèles complexes, qui ne sont pas naturellement transparents et nécessite des techniques supplémentaires pour expliquer leurs prédictions après coup. Les méthodes post hoc incluent LIME (qui génère des explications locales en ajustant des modèles simples autour d'une instance spécifique) et SHAP (qui utilise des concepts de théorie des jeux pour attribuer à chaque feature une valeur d'importance). Ces techniques peuvent être appliquées à n'importe quel modèle de machine learning sans se baser sur la structure interne du modèle.

#### Interprétation de modèle avec SHAP

C’est dans cet optique que nous avons décidé d’utiliser SHAP pour essayer de comprendre nos modèles.

SHAP pour *SHapley Additive exPlanations* est une librairie Python proposant une approche unifiée pour expliquer le résultat de tout modèle 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. SHAP relie donc la théorie des jeux aux explications locales, en se basant sur les valeurs de Shapley.

Les prédictions des modèles peuvent être expliquées en supposant que chaque variable explicative est un "joueur" dans un jeu où la prédiction est le paiement. La valeur de Shapley, une méthode issue de la théorie des jeux coopératifs, nous indique comment répartir équitablement le gain entre les variables. La valeur de Shapley est une méthode qui permet d'attribuer les paiements aux joueurs en fonction de leur contribution au paiement total. Les joueurs coopèrent au sein d'une coalition et obtiennent un certain gain de cette coopération. Ainsi :

* Le jeu est la tâche de prédiction pour une seule instance du jeu de données ;
* Le gain est la prédiction réelle pour cette instance moins la prédiction moyenne de toutes les instances ;
* Les joueurs sont les valeurs pour chaque variable de l'instance qui collaborent pour recevoir le gain (= prédire une certaine valeur).

En d’autres termes, 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. C'est la contribution de chaque feature à la prédiction finale, en supposant que toutes les autres features restent constantes.

SHAP 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. L'objectif est d'expliquer la différence entre la prévision réelle et la prévision moyenne.

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.

#### Interprétation globale du modèle GradientBoostingClassifier binaire

Afin de mettre en pratique l’interprétabilité avec SHAP, nous avons choisi arbitrairement un modèle parmi les trois plus performants : le GradientBoostingClassifier.

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. Le diagramme d’importance permet de répondre à la question : Quelles sont les variables globalement les plus importantes pour comprendre la prédiction ?

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Nous pouvons voir que les 10 premières features ont un impact significatif sur la prédiction avec la moyenne des valeurs absolues des valeurs de Shap plus importante que les features qui suivent. Après la 10e feature, on observe une cassure dans les valeurs.

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 violet 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 ?

Une image contenant texte, capture d’écran

Description générée automatiquement

Ce graphique est souvent préféré car il apporte plus d’informations et de manière claire. Il donne pour chaque feature un ensemble des valeurs de Shap (représenté par un nuage de point) qui représentent chacune une instance donnée.

Sur ce graphique, la couleur représente la valeur de la feature pour l’instance donnée. Dans notre cas, les variables sont catégoriques avec des valeurs de 1 ou 0. Cela explique donc pourquoi nous n’avons que deux couleurs possibles pour chaque point. La valeur de Shap pour chaque point est ensuite soit positive (et impacte les prédictions de manière positive), soit négative (et impacte les prédictions de manière négative).

On peut donc constater que la caractéristique « catv\_moto » est significative et que lorsque le véhicule utilisé lors d’un accident est une moto alors cette variable a une influence positive sur le modèle (valeur de SHAP positive et élevée). Cela signifie que lorsque cette caractéristique est présente dans une prédiction, il y a de fortes chances que le modèle ne commette pas d’erreur.

A l’inverse, dans le cas où la variable « obsm\_1 » prend la valeur 1 (c’est-à-dire que l’obstacle mobile heurté est un piéton), cette variable a une influence négative sur le modèle. Cela signifie que lorsque cette caractéristique est présente dans une prédiction, il y a de fortes chances que le modèle commette une erreur.

Il faut également considérer que dans certains cas, la relation entre la variable cible et la caractéristique dépend de la valeur d'une autre caractéristique. Il y a donc une interaction entre caractéristique. Grâce à un graphique d'interaction, il est possible de constater cela.

Une image contenant texte, capture d’écran, Tracé, diagramme

Description générée automatiquement

Sur ce graphique d’interaction, nous pouvons constater que si la catégorie du véhicule impliqué dans l'accident est une moto (catv\_moto = 1), alors la valeur Shap augmente et par conséquent la probabilité que l'accident soit corporel augmente aussi. Grâce au graphique d'interaction, on constate aussi que si la personne accidentée est le conducteur du véhicule (conducteur = 1), on peut dire que lorsque son véhicule est une moto (valeur = 1), la valeur Shap augmente et par conséquent la probabilité que l'accident soit corporel augmente aussi.

De manière similaire, prenons en exemple le graphique d’interaction pour la deuxième variable la plus impactante pour le modèle, obsm\_1. Cette variable prend la valeur 1 quand l’obstacle mobile heurté est un piéton.

Une image contenant texte, capture d’écran, diagramme, Tracé

Description générée automatiquement

Nous pouvons constater que lors d’un accident avec un obstacle mobile heurté étant un piéton, la valeur Shap diminue fortement et par conséquent la probabilité que la gravité de l’accident soit bien prédit aussi. Grâce au graphique d'interaction, on constate aussi que si un piéton a été heurté via une collision qui n’implique pas plusieurs véhicules, alors cette combinaison de caractéristiques pousse le modèle à faire de mauvaise prédiction. Cela peut s’expliquer par le fait que la combinaison de ces deux variables est peu présente dans le dataset. En effet, 81% du temps ces deux variables ont la valeur 0 toutes les deux (c’est-à-dire que les piétons sont peu heurtés par un véhicule). Le modèle commet plus d’erreurs car il a peu d’exemples sur lesquels se baser. Dans le cadre d’une amélioration du modèle, un travail de resampling de ces variables permettrait d’obtenir de meilleures performances.

#### Interprétation locale du modèle GradientBoostingClassifier binaire

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

Dans notre cas, une instance correspond à un usager de la route impliqué dans un accident. Pour une instance, ces trois représentations sont redondantes car elles représentent l’information de manière similaire. Cependant, ces représentations peuvent être complémentaires et en les associant, il est possible de comprendre d’une manière plus intuitive le résultat.

Le Force plot est utile pour voir où se place le “output value” par rapport à la “base value” (nous reviendrons sur ces termes via un exemple ci-après). 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.

Prenons l’exemple pour un usager accidenté. Il s’agit de l’usager n°228673. Ci-dessous son Force plot.

Une image contenant texte, Police, ligne, capture d’écran

Description générée automatiquement

En rouge sont regroupées les variables qui ont un impact positif (c’est-à-dire qui contribuent à ce que la prédiction soit plus élevée que la valeur de base (« base value »), encadrée en violet) et en bleu, celles ayant un impact négatif (c’est-à-dire qui contribuent à ce que la prédiction soit plus basse que la valeur de base).

La valeur de base du dataset est à 0.392 (moyenne des valeurs entre 0 (indemne) et 1 (blessé)). La prédiction de la valeur médiane de l'accident corporel par usagers pour l'usager dans l’exemple est 1.84 (encadré en vert, c’est la « output value »). Nous étudions donc un exemple d'usager qui a une prédiction de valeur plus élevée que la moyenne et nous voyons les raisons du modèle pour lesquelles les valeurs sont plus élevées (contribution positive de obs\_8, obsm\_1, ...). Cela signifie aussi que la prédiction pour cet usager est plus fiable.

Regardons maintenant le Waterfall plot pour cet usager :

Une image contenant texte, capture d’écran, diagramme, nombre

Description générée automatiquement

Ce graphique 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.

Ci-dessous le Decision plot pour le même usager :

Une image contenant texte, nombre, capture d’écran, Police

Description générée automatiquement

## Classification multiclasse

### Optimisation des hyperparamètres

Afin d’optimiser et d’améliorer les prédictions, nous nous sommes intéressés aux hyperparamètres. Nous avons dans un premier temps sélectionné 3 modèles déjà testé et qui avaient donné des résultats satisfaisant :

* RandomForestClassifier ;
* LGBMClassifier ;
* GradientBoostingClassifier.

Sur ces trois modèles, une recherche des meilleurs hyperparamètres a été réalisée. Cela s’est fait via les outils « RandomSearch » et « GridSearch » de la librairie sklearn. La méthodologie a été la suivante :

* Utilisation du RandomSearch afin d’obtenir un ordre de grandeur sur les différents hyperparamètres ;
* Utilisation du GridSearch pour affiner les hyperparamètres autour des valeurs trouvées précédemment.

Nous résumons dans le tableau ci-dessous les résultats obtenus.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Modèle** | **Train Accuracy** | **Test Accuracy** | **Train Recall** | **Test Recall** | **Train**  **F1-score** | **Test**  **F1-score** |
| GradientBoostingClassifier | 0.685 | 0.661 | 0.502 | 0.464 | 0.526 | 0.473 |
| RandomForestClassifier | 0.773 | 0.662 | 0.568 | 0.452 | 0.595 | 0.456 |
| LGBMClassifier | 0.681 | 0.660 | 0.490 | 0.461 | 0.507 | 0.468 |

Nous faisons un rappel sur les métriques utilisées. Pour cela nous allons utiliser un exemple avec une modélisation binaire pour commencer. En face d’un tunnel, nous devons prédire quel type de véhicule sortira : une voiture (résultat positif) ou une moto (résultat négatif).

* Le Recall correspond au nombre de voiture que notre modèle a prédit et qui se sont avérés être effectivement des voitures, divisé par l’ensemble des voitures qui ont traversé le tunnel ;
* La Précision correspond au nombre de voiture que notre modèle a prédit et qui se sont avérés être effectivement des voitures, divisé par le nombre total de voiture que notre modèle a prédit et qui se sont avérés être vrai (voiture) ou fausse (moto) ;

Via le Recall, on regarde le nombre de positif que le modèle a bien prédit sur l’ensemble des positifs. Si le modèle prédit uniquement « positif », le Recall sera élevé.

Via la Précision, on regarde le nombre de positif que le modèle a bien prédit sur l’ensemble des positifs prédit. Si le modèle ne prédit jamais « positif », la précision sera élevée.

Ainsi, plus le Recall est élevé et plus le modèle repère de positif. Plus la précision est élevée et moins le modèle ne se trompe sur les positifs.

De manière synthétique, nous pouvons constater que :

* Les scores sont assez similaires entre les modèles ;
* Le GradientBoosting présente des performances légèrement meilleures ;
* La précision est plus élevée que le recall. Cela signifie que les modèles prédisent plus facilement des vrais positifs ;
* Un léger over-fitting est présent, principalement pour le RandomForest.

Avec une modélisation multiclasse, les métriques sont moyennées suivant les classes. Avec l’exemple du GradientBoosting, nous observons les scores suivants :

Précision (66%) : La précision pondérée est la moyenne des précisions pour chaque classe, pondérée par le nombre d'observations de chaque classe. Une précision de 66% signifie que lorsque le modèle prédit une classe, il est correct environ 66% du temps, ce qui est supérieur à une classification aléatoire.

Recall (46%) : Le rappel pondéré est la moyenne des rappels pour chaque classe, pondérée par le nombre d'observations de chaque classe. Un Recall de 46% signifie que le modèle identifie correctement environ 46% des observations de chaque classe.

F1-score (47%) : Le F1-score pondéré est la moyenne harmonique des précisions et rappels pondérés pour chaque classe. Un F1-score pondéré de 47% indique que le modèle a un équilibre convenable entre la précision et le rappel pour chaque classe, compte tenu du déséquilibre des classes.

### Resampling des données

Comme nous l’avons vu précédemment, notre problème de Machine Learning présente une forte disparité entre ses classes. Le test réalisé en transformant la classification en binaire a permis d’apporter de meilleurs résultats. Ainsi, le rééquilibre des classes permet au modèle de mieux prédire. Nous allons donc effectuer un resampling des données.

Le resampling se fait uniquement sur le jeu de donnée d’entrainement, cela afin de ne pas modifier la distribution du jeu de données test (X\_test et y\_test doivent rester inchangés car ils doivent représenter la distribution réelle des données). Il convient également de ne pas sur-équilibrer le jeu de donnée pour éviter de créer du bruit dans le dataset.

Plusieurs librairies sont disponibles comme SMOTETomek ou SMOTEENN. Nous préférons toutefois utiliser deux fonctions de la librairie imblearn : RandomOverSampler et RandomUnderSampler.

Nous rappelons la répartition avant resampling :

* Classe 1 (indemne) : 116 472 usagers (44.4%) ;
* Classe 4 (blessé léger) : 101 313 usagers (38.6%) ;
* Casse 3 (blessé grave) : 38 043 usagers (14.5%) ;
* Classe 2 (tué) : 6 469 usagers (2.5%).

Nous allons voir l’impact du resampling en modifiant toutes les classes pour obtenir 25% de répartition chacune. Ainsi, le RandomOverSampler va augmenter le nombre d’échantillons des classes minoritaires (classes 2 et 3). Le RandomUnderSampling va réduire le nombre d’échantillons des classes majoritaires (classes 1 et 4).

Nous utilisons le nouveau jeu d’entrainement avec le modèle GradientBoosting. Ci-dessous les résultats obtenus :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Modèle** | **Train Accuracy** | **Test Accuracy** | **Train Recall** | **Test Recall** | **Train**  **F1-score** | **Test**  **F1-score** |
| GradientBoostingClassifier  **Avec resampling** | **0.648** | **0.596** | **0.648** | **0.554** | **0.642** | **0.480** |
| GradientBoostingClassifier | 0.685 | 0.661 | 0.502 | 0.464 | 0.526 | 0.473 |
| RandomForestClassifier | 0.773 | 0.662 | 0.568 | 0.452 | 0.595 | 0.456 |
| LGBMClassifier | 0.681 | 0.660 | 0.490 | 0.461 | 0.507 | 0.468 |

Nous pouvons constater les points suivants :

* Légère baisse de la précision mais augmentation du recall. Cela signifie que le modèle permet de mieux identifier chaque classe mais avec des erreurs plus souvent (baisse de la précision) ;
* Le F1-score est également légèrement augmenté ;
* L’overfitting est plus marqué.

### Interprétation multiclasse

Après avoir suivie quelques pistes d’amélioration des modèles, nous allons essayer de comprendre comment le modèle a prédit ses classes, de la même manière que lors de la modélisation binaire. Nous utilisons le modèle GradientBoosting optimisé.

#### Interprétation globale

Le graphique ci-dessous 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 sommes avec un problème de classification multiclasses, les variables sont classées en fonction de leur contribution globale à toutes les classes et décomposée en code en couleur suivant l'ampleur de chaque classe.

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

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, cela signifie que la caractéristique correspondante a un impact positif sur la classification de cet usager.

Chaque point sur les graphiques est un usager, dont la couleur correspond à la valeur d'origine de l'entité et positionnée en fonction de sa distribution.

Une image contenant texte, capture d’écran, menu

Description générée automatiquement

Ci-dessous l’équivalence des classes :

* Classe 0 : Indemne ;
* Classe 1 : Tué ;
* Classe 2 : Blessé grave ;
* Classe 3 : Blessé léger.

#### Interprétation locale

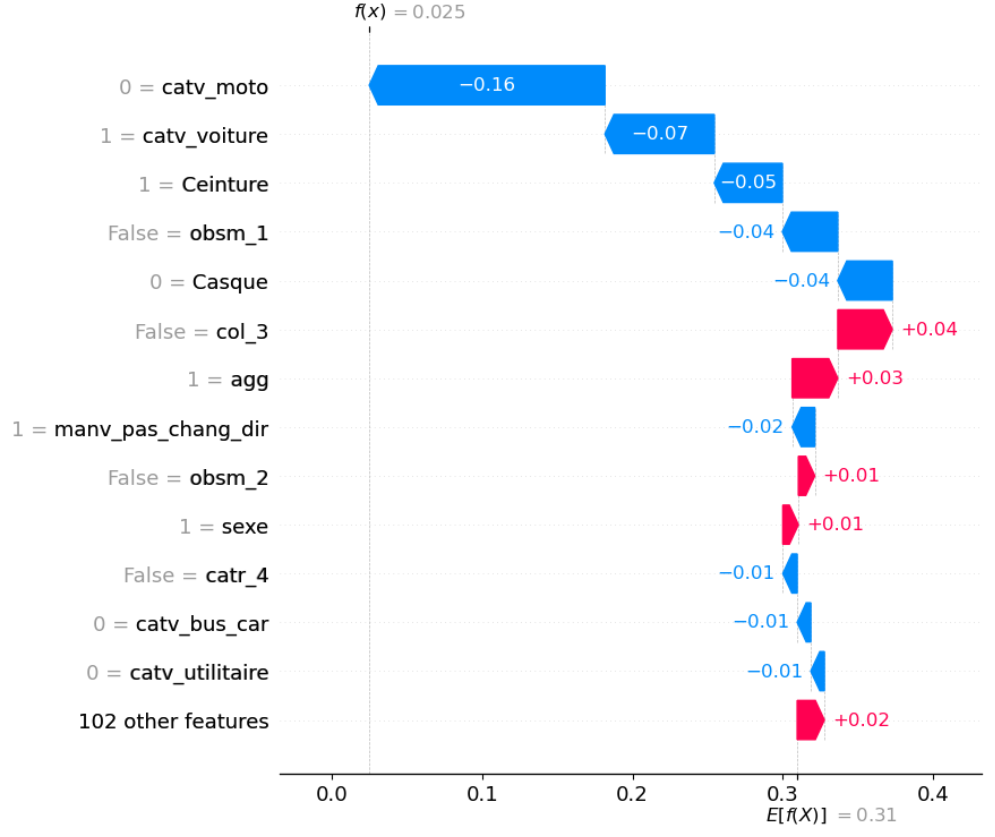
Lorsque nous avons un modèle de classification avec plusieurs classes et que nous créons un graphique waterfall plot pour une observation donnée, chaque classe a son propre graphique waterfall plot avec sa propre contribution marginale de chaque fonctionnalité à la prédiction de cette classe.

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 multiclasses 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 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.

Nous montrons ci-dessous un waterfall pour un usager et une classe (indemne) :



# Conclusions

## Difficultés rencontrées lors du projet

Quel a été le principal verrou scientifique rencontré lors de ce projet ?

Pour chacun des points suivants, si vous avez rencontré des difficultés, détaillez en quoi elles vous ont ralenti dans la mise en place de votre projet.

Prévisionnel : tâches qui ont pris plus de temps que prévu, etc.

Jeux de données : acquisition, volumétrie, traitement, agrégation, etc.

Compétences techniques / théoriques : timing d’acquisition des compétences, compétence non proposée en formation, etc..

Pertinence : de l’approche, du modèle, des données, etc..

IT : puissance de stockage, puissance computationnelle, etc.

Autres

La recherche de l’importance des features est une étape importante dans l’optimisation du modèle car elle permet de comprendre le raisonnement du modèle dans ses prédictions mais permet également de faire un état des lieux sur ce qui a été utilisé. Nous pouvons ainsi voir les variables qui apportent à la bonne prédiction et celles qui peuvent biaiser les résultats du modèle. Ainsi, en refaisant une étape de pre-processing, il est possible de corriger ces biais et obtenir de meilleures performances.

## Bilan

Détaillez quelle a été votre contribution principale dans l’atteinte des objectifs du projet.

Avez-vous modifié le modèle depuis la dernière itération ? Si oui, détaillez.

Présentez les résultats obtenus et comparez-les au benchmark

Pour chacun des objectifs du projet, détaillez en quoi ils ont été atteints ou non.

S’ils ont été atteints, dans quel(s) process(es) métier(s) votre modèle peut-il s’inscrire ? Détaillez.

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 66%.

## Suite du projet

Quelles pistes d’amélioration suggérez-vous pour augmenter les performances de votre modèle ?

En quoi votre projet a-t-il contribué à un accroissement de la connaissance scientifique ?

*développer un scoring des zones à risque en fonction des informations météorologiques, l’emplacement géographique (coordonnées GPS, images satellite, …)*

* Faire attention au biais dans la donnée. Le dataset est issu des formulaires remplis par les forces de l’ordre lors d’un accident. Il peut y avoir un facteur humain introduisant un biais dans la donnée.
* Faire encodage cyclique sur les variables temps (créneau horaire par exemple).
* Mise en lien entre proportion des accidents par rapport au pourcentage de la population par région 🡪 dans la data visualisation pour faire apparaitre d’autres corrélations ou nuancer certains résultats.
* Utilisation des variables non catégorielles (département, âge, vitesse, …).
* Utilisation de l’ACP pour réduire le nombre de variables (attention car perte des noms des axes).

A noter que le dataset ne contient que des informations sur les accidents. Il serait intéressant de voir la proportion de gens véhiculés par exemple, les taux d’alcoolémie, …

## Bibliographie

Sur quels éléments bibliographiques (articles de recherches, blog, livres, etc) vous êtes vous appuyé pour réaliser votre projet ?

[Bases de données annuelles des accidents corporels de la circulation routière - Années de 2005 à 2021 - data.gouv.fr](https://www.data.gouv.fr/en/datasets/bases-de-donnees-annuelles-des-accidents-corporels-de-la-circulation-routiere-annees-de-2005-a-2019/#community-discussions)

[https://www.kaggle.com/ahmedlahlou/accidents-in-france-from-2005-to-2016](https://www.kaggle.com/ahmedlahlou/accidents-in-france-from-2005-to-2016%20)

[Bases de données annuelles des accidents corporels de la circulation routière - Années de 2005 à 2021 - data.gouv.fr](https://www.data.gouv.fr/en/datasets/bases-de-donnees-annuelles-des-accidents-corporels-de-la-circulation-routiere-annees-de-2005-a-2019/#community-discussions)

## Annexes

Description des fichiers de code.

Fichier gouv pour voir l’équivalence des noms des variables