**Décembre 2023 – Juillet 2024**

**Accidents routiers en France**

(Partie Modélisation)

**Projet mené par :**

**Matthieu Claudel**

**Vanessa Ibert**

**Camille Pelat**

**Nadège Reboul**

**Sommaire**

[INTRODUCTION 6](#_Toc168571362)

[I PRÉDICTION DU NOMBRE D’INDEMNES, DE BLESSÉS LÉGERS, DE BLESSÉS HOSPITALISÉS ET DE TUÉS PAR JOUR 7](#_Toc168571363)

[I.1 Séries temporelles 7](#_Toc168571364)

[I.1.1 Création des jeux de données 7](#_Toc168571365)

[I.1.2 Etude de la saisonnalité 8](#_Toc168571366)

[I.1.3 Création d’une baseline 10](#_Toc168571367)

[I.1.4 Prédictions 15](#_Toc168571368)

[I.1.5 Conclusion 27](#_Toc168571369)

[II PRÉDICTION DE LA GRAVITÉ DE L’ACCIDENT 32](#_Toc168571370)

[II.1 Régression logistique 32](#_Toc168571371)

[II.1.1 Modèle de référence 32](#_Toc168571372)

[II.1.2 Optimisation du modèle 33](#_Toc168571373)

[II.1.3 Interprétabilité des résultats 34](#_Toc168571374)

[II.2 Support Vector Machine 37](#_Toc168571375)

[II.2.1 Modèle de référence 37](#_Toc168571376)

[II.2.2 Optimisation du modèle - Ajustement des hyperparamètres penalty, loss, C et multi\_class avec GridSearchCV 37](#_Toc168571377)

[II.3 Decision Tree Classifier - Random Forest Classifier - Balanced Random Forest 39](#_Toc168571378)

[II.3.1 Modèle de référence 39](#_Toc168571379)

[II.3.2 Optimisation du modèle 41](#_Toc168571380)

[II.3.3 Résultats 42](#_Toc168571381)

[II.3.4 Interprétabilité des résultats 44](#_Toc168571382)

[II.4 CatBoost Classifier 49](#_Toc168571383)

[II.4.1 Modèle de référence 49](#_Toc168571384)

[II.4.2 Optimisation du modèle 49](#_Toc168571385)

[II.5 XGBoost Classifier 51](#_Toc168571386)

[II.5.1 Modèle de référence 51](#_Toc168571387)

[II.5.2 Optimisation de la méthode 2 55](#_Toc168571388)

[II.5.3 Interprétabilité des résultats avec SHAP 56](#_Toc168571389)

[II.6 K-nearest neighbors (KNN) 59](#_Toc168571390)

[II.6.1 Modèle de référence 59](#_Toc168571391)

[II.6.2 Recherche de paramètres optimaux avec GridSearchCV 60](#_Toc168571392)

[II.6.3 Sélection des features 62](#_Toc168571393)

[II.7 Modélisation par Deep Learning 63](#_Toc168571394)

[II.7.1 Preprocessing 63](#_Toc168571395)

[II.7.2 Modèle de référence 63](#_Toc168571396)

[II.7.3 Rééquilibrage du jeu de données 64](#_Toc168571397)

[II.7.4 Comparaison avec le meilleur modèle de Machine Learning 70](#_Toc168571398)

[III ESSAIS D’OPTIMISATION DE LA PRÉDICTION DE LA GRAVITÉ DE L’ACCIDENT 71](#_Toc168571399)

[III.1 Classification binaire 71](#_Toc168571400)

[III.1.1 Modélisation 71](#_Toc168571401)

[III.1.2 Comparaison des résultats avec la classification multiclasses 72](#_Toc168571402)

[III.1.3 Interprétabilité des résultats 73](#_Toc168571403)

[III.2 Etude des variables ‘catv’, ‘obs’ et ‘obsm’ 79](#_Toc168571404)

[III.2.1 Modifications des variables 79](#_Toc168571405)

[III.2.2 Modélisation 79](#_Toc168571406)

[III.2.3 Résultats et comparaison avec le jeu de données initial 80](#_Toc168571407)

[III.3 Ajout de la variable ‘nb\_usagers\_gr’ 81](#_Toc168571408)

[III.3.1 Création de la variable ‘nb\_usagers\_gr’ 81](#_Toc168571409)

[III.3.2 Modélisation 81](#_Toc168571410)

[III.3.3 Résultats et comparaison avec le jeu de données initial 82](#_Toc168571411)

[IV CONCLUSION 83](#_Toc168571412)

**Table des Figures**

[Figure 1 : Évolutions du nombre de cas par jour entre 2019 et 2022 7](#_Toc168571290)

[Figure 2 : Lissage de la courbe sur une fenêtre glissante de 365 jours. 8](#_Toc168571291)

[Figure 3 : Moyennes annuelles du nombre de cas par jours pour les années 2021 et 2022 9](#_Toc168571292)

[Figure 4 : Décompositions des séries temporelles 9](#_Toc168571293)

[Figure 5: Baselines avec shift de 1 jour 10](#_Toc168571294)

[Figure 6 : Baselines moyennées sur 7 jours + shift de 1 jour 11](#_Toc168571295)

[Figure 7 : Résultats de SARIMAX avec en variable exogène une série de Fourier 13](#_Toc168571296)

[Figure 8 : Évaluations de SARIMAX avec en variable exogène une série de Fourier sur train et test 14](#_Toc168571297)

[Figure 9 : Évaluation de MSTL avec différents trend\_forecaster sur train et test 16](#_Toc168571298)

[Figure 10 : Évaluation de PROPHET avec différentes saisonnalités sur train et test 21](#_Toc168571299)

[Figure 11 : Évaluations de LSTM sur train et test 26](#_Toc168571300)

[Figure 12 : Comparaison des modèles et prédictions à 1 et 6 mois pour les indemnes 28](#_Toc168571301)

[Figure 13 : Comparaison des modèles et prédictions à 1 et 6 mois pour les blessés légers 29](#_Toc168571302)

[Figure 14 : Comparaison des modèles et prédictions à 1 et 6 mois pour les blessés hospitalisés 30](#_Toc168571303)

[Figure 15 : Comparaison des modèles et prédictions à 1 et 6 mois pour les tués 31](#_Toc168571304)

[Figure 16 : Métriques (a) et matrice de confusion (b) du modèle de régression logistique de référence 32](#_Toc168571305)

[Figure 17 : Résultats du GridSearchCV sur penalty et C, hyperparamètres de la régression logistique 33](#_Toc168571306)

[Figure 18 : Métriques (a) et matrice de confusion (b) du modèle de régression logistique optimisé 34](#_Toc168571307)

[Figure 19 : Coefficients du modèle de régression logistique optimisé lors de la validation croisée 35](#_Toc168571308)

[Figure 20 : Variance expliquée, en cumul (axe de gauche) ou unitairement (axe de droite), en fonction du nombre de composantes retenues 36](#_Toc168571309)

[Figure 21 : Les 20 variables contribuant le plus aux deux premières composantes de l’analyse factorielle 36](#_Toc168571310)

[Figure 22 : Métriques (a) et matrice de confusion (b) du modèle LinearSVC précédé d’une approximation de Nyström 38](#_Toc168571311)

[Figure 23 : Variables sélectionnées selon le modèle avec leur importance 40](#_Toc168571312)

[Figure 24 : Métriques et matrice de confusion selon le modèle 41](#_Toc168571313)

[Figure 25 : Métriques et matrice de confusion avec les paramètres optimisés selon le modèle 43](#_Toc168571314)

[Figure 26 : Métriques et matrice de confusion pour le meilleur modèle 44](#_Toc168571315)

[Figure 27 : Importance des variables pour le meilleur modèle 44](#_Toc168571316)

[Figure 28  : Graphiques d’importance des variables selon SHAP 46](#_Toc168571317)

[Figure 29 : Graphiques de densité des valeurs de SHAP 47](#_Toc168571318)

[Figure 30 : Graphique à coordonnées parallèles présentant le lien entre les hyperparamètres de Catboost et la performance du modèle. 50](#_Toc168571319)

[Figure 31 : Métriques (a) et matrice de confusion (b) du modèle CatBoost 50](#_Toc168571320)

[Figure 32 : Courbes de mlogloss et merror suivant le nombre d’arbres 52](#_Toc168571321)

[Figure 33 : Matrices de confusion pour chaque méthode 52](#_Toc168571322)

[Figure 34 : Rapports de classification selon la méthode 53](#_Toc168571323)

[Figure 35 : Les 20 premières features importance selon weight, gain, cover et selon la méthode 54](#_Toc168571324)

[Figure 36 : Courbes de mlogloss et merror pour le modèle optimisé 55](#_Toc168571325)

[Figure 37 : Matrices de confusion du modèle de référence et du modèle optimisé 55](#_Toc168571326)

[Figure 38 : Rapports de classification pour le modèle de référence et le modèle optimisé 56](#_Toc168571327)

[Figure 39 : Graphiques d’importance des variables selon SHAP 57](#_Toc168571328)

[Figure 40 : Graphiques de densité des valeurs de SHAP 58](#_Toc168571329)

[Figure 41 : Rapport de classification (a) – Matrice de confusion (b) – Courbes d’évolution de la loss et de l’accuracy par epoch 64](#_Toc168571330)

[Figure 42 : Rapports de classification (a) – Matrices de confusion (b) 65](#_Toc168571331)

[Figure 43 : Courbes d’évolution la loss et de l’accuracy par epoch 66](#_Toc168571332)

[Figure 44 : Rapports de classification (a) – Matrices de confusion (b) 68](#_Toc168571333)

[Figure 45 : Courbes d’évolution la loss et de l’accuracy par epoch 69](#_Toc168571334)

[Figure 46 : Métriques et matrice de confusion pour les meilleurs modèles de Machine Learning et de Deep Learning 70](#_Toc168571335)

[Figure 47 : Métriques et matrice de confusion selon le jeu de données binaire utilisé 72](#_Toc168571336)

[Figure 48 : Réarrangement de la matrice de confusion pour de la classification multiclasses et comparaison avec la classification binaire 75](#_Toc168571337)

[Figure 49 : Importance des variables selon le jeu de données binaires 76](#_Toc168571338)

[Figure 50 : Graphiques d’importance des variables selon SHAP selon le jeu de données binaires 77](#_Toc168571339)

[Figure 51  : Graphique de densité des valeurs de SHAP selon le jeu de données binaires 78](#_Toc168571340)

[Figure 52 : Comparaison des métriques et matrice de confusion pour le jeu de données initial et le jeu de données avec modification de variables 80](#_Toc168571341)

[Figure 53 : Histogramme du nombre d’accidents en fonction du nombre de personnes accidentées (a) – Courbe cumulé du pourcentage du nombre d’accidents en fonction du nombre de personnes accidentées 81](#_Toc168571342)

[Figure 54 : Comparaison des métriques et matrices de confusion du jeu de données initial et du jeu de donnée avec ajout de ‘nb\_usagers\_gr’ 82](#_Toc168571343)

[Figure 55 : Diagrammes radars de comparaison des modèles en termnes de Precision, Recall, Specificity, F1-Score et Index Balanced Accuracy 83](#_Toc168571344)

[Figure 56 : Métriques et matrice de confusion pour le meilleur modèle 84](#_Toc168571345)

[Figure 57. Risque relatif (échelle log) d'être hospitalisé ou tué pour une personne de la base de données 85](#_Toc168571346)

**Table des Tableaux**

[Tableau 1 : Paramètres du modèle de régression logistique pris comme référence 32](#_Toc168571347)

[Tableau 2 : Méthodes d’ajustement des paramètres 33](#_Toc168571348)

[Tableau 3 : Paramètres du modèle de régression logistique après optimisation 33](#_Toc168571349)

[Tableau 4 : Paramètres de l’approximation Nyström et du modèle de LinearSVC pris comme référence 37](#_Toc168571350)

[Tableau 5  : Choix d’optimisation des paramètres selon le modèle 42](#_Toc168571351)

[Tableau 6 : Paramètres par défaut de XGBoost 51](#_Toc168571352)

[Tableau 7 : Métriques principales pour chaque méthode 53](#_Toc168571353)

[Tableau 8 : Paramètres optimisés pour XGBoost 55](#_Toc168571354)

[Tableau 9 : Rapport de classification 59](#_Toc168571355)

[Tableau 10 : Matrice de confusion de l’algorithme KNN (paramètres par défaut) 60](#_Toc168571356)

[Tableau 11 : Matrice de confusion normalisée par ligne 60](#_Toc168571357)

[Tableau 12 : Rapport de classification pour le modèle optimisé 61](#_Toc168571358)

[Tableau 13 : Matrice de confusion normalisée par ligne pour le modèle optimisé 61](#_Toc168571359)

[Tableau 14 : Rapport de classification avec sélection de variables 62](#_Toc168571360)

[Tableau 15 : Tableau des paramètres testés pour le Deep Learning 67](#_Toc168571361)

# INTRODUCTION

Pour la prédiction de la gravité des accidents routiers en France, nous avons choisi de traiter le problème sous deux angles :

* prédire le nombre journalier de chacune des modalités de notre cible (‘grav’) par le biais de séries temporelles,
* prédire l’état de gravité de la personne accidentée en fonction des caractéristiques de l’accident par une classification multi-classes.

Pour chacune de ces approches, les prévisions sont réalisées par un apprentissage supervisé.

**Séries temporelles**

Dans le cas des séries temporelles, nous souhaitons prédire le nombre d’indemnes, de blessés légers, de blessés hospitalisés et de tués par jour pour les prochains mois. Pour comparer les différents modèles utilisés, nous employons 3 métriques :

* l’erreur absolue moyenne (MAE : mean absolute error),
* le carré moyen des erreurs (MSE : mean squared error),
* l’erreur quadratique moyenne (RMSE : root mean squared error).

La MAE quantifie les erreurs réalisées par le modèle en pénalisant autant les grandes que les petites erreurs. Étant dans la même unité que la variable à prédire, elle permet d’interpréter facilement l’erreur du modèle.

La MSE quantifie les erreurs réalisées par le modèle, mais contrairement à la MAE, elle pénalise plus fortement les grandes erreurs que les petites erreurs. En revanche, elle ne permet pas d’interpréter facilement le modèle car elle n’est pas dans la même unité que le modèle.

La RMSE est la racine carrée de la MSE. Contrairement à la MSE, elle s’exprime dans la même unité que la variable à prédire permettant donc une interprétation plus facile du modèle.

**Classification multi-classes**

Dans le cas de la classification multi-classes, nous utilisons aussi 3 métriques :

* l’accuracy,
* le f1-score,
* la matrice de confusion.

L’accuracy (nombre de prédictions correctes / nombre total de prédictions) permet de connaître si le modèle est performant ou pas et s’il y a du sur- ou du sous-apprentissage.

Le choix du f1-score est dû au fait que nous avons un jeu de données très déséquilibré. En effet, le f1-score étant la moyenne harmonique de la précision et du rappel, cela permet d’optimiser la précision et le rappel pour obtenir le maximum de vrais positifs.

Enfin, la matrice de confusion permet de comparer dans le détail les modèles ayant des f1-scores semblables.

# PRÉDICTION DU NOMBRE D’INDEMNES, DE BLESSÉS LÉGERS, DE BLESSÉS HOSPITALISÉS ET DE TUÉS PAR JOUR

## Séries temporelles

### Création des jeux de données

Pour les séries temporelles, nous allons créer 4 jeux de données en fonction des 4 modalités de la variable cible ‘grav’ à partir de notre jeu de données initial, soit :

* 1 jeu de données pour la modalité ‘indemnes’,
* 1 jeu de données pour la modalité ‘blessés légers’,
* 1 jeu de données pour la modalité ‘blessés hospitalisés’,
* 1 jeu de données pour la modalité ‘tués’.

Pour réaliser cela, nous procédons en plusieurs étapes :

* conservation uniquement des variables ‘jour’, ‘mois’, ‘an’, ‘grav’,
* séparation en 4 jeux de données, en récupérant les lignes correspondant à la modalité choisie,
* création d’une variable date au format datetime à partir des variables ‘jour’, mois’, ‘an’ et suppression de ces dernières,
* création d’une variable ‘Nbre\_Acc’ qui correspond au nombre d’accidents par jour,
* mise en index de la date.

Cependant, le jeu de données initial que nous avons choisi pour traiter le problème de prédiction de la gravité des accidents routiers en France regroupe les données recueillies pour les années 2019 à 2022. Or la pandémie de Covid-19 et les confinements associés, en mars 2020 et, de moindre mesure, en octobre 2020, ont biaisé les données au niveau temporel. En effet, lors des confinements peu de personnes étaient autorisées à sortir, donc le nombre d’accidents a fortement chuté, induisant une diminution du nombre de cas pour chacune des modalités de notre variable cible ‘grav’ (Figure 1).

|  |  |
| --- | --- |
| **Pour les indemnes** | **Pour les blessés légers** |
|  |  |
|  |  |
| **Pour les blessés hospitalisés** | **Pour les tués** |
|  |  |

Figure 1 : Évolutions du nombre de cas par jour entre 2019 et 2022

Pour ne pas avoir de données biaisées pour prédire le nombre de cas de chacune des modalités de la variable cible, nous avons décidé de ne prendre que les années 2021 et 2022.

### Etude de la saisonnalité

L’étude de la saisonnalité a été effectuée en s’inspirant de la publication suivante : <https://blog.statoscop.fr/timeseries-4.html>.

#### Tendance

Pour visualiser la tendance, on effectue un lissage de la courbe à l’aide des moyennes mobiles sur une fenêtre glissante d’observations. En choisissant différentes tailles de fenêtres glissantes (correspondant à 1, 2, 3 et 4 semaines et à 6 mois et 1 an), on peut clairement observer une tendance linéaire et stable pour une fenêtre de 365 jours quelle que soit la modalité (Figure 2).

|  |  |
| --- | --- |
| **Tendance à 365 jours pour les indemnes** | **Tendance à 365 jours pour les blessés légers** |
|  |  |
|  |  |
| **Tendance à 365 jours pour les blessés hospitalisés** | **Tendance à 365 jours pour les tués** |
|  |  |

Figure 2 : Lissage de la courbe sur une fenêtre glissante de 365 jours.

#### Saisonnalité

Une saisonnalité annuelle est visible dans la Figure 2, avec des pics vers juin et octobre et des « creux » hivernaux dans les quatre séries (moins visible dans la série des tués, où le nombre de cas est plus faible et le bruit plus important). Par ailleurs, le nombre d’accidentés de chaque classe est aussi dépendant du jour de la semaine, cette saisonnalité hebdomadaire étant bien visible dans la Figure 3.

|  |  |
| --- | --- |
| **Pour les indemnes** | **Pour les blessés légers** |
| Une image contenant texte, ligne, capture d’écran, diagramme  Description générée automatiquement | Une image contenant texte, capture d’écran, ligne, Tracé  Description générée automatiquement |
|  |  |
| **Pour les blessés hospitalisés** | **Pour les tués** |
| Une image contenant ligne, Tracé, texte, capture d’écran  Description générée automatiquement | Une image contenant ligne, diagramme, Tracé, texte  Description générée automatiquement |

Figure 3 : Moyennes annuelles du nombre de cas par jours pour les années 2021 et 2022

#### Bruit

Comme le modèle est additif, le bruit s’obtient en soustrayant à la série originale la tendance et la saisonnalité (Figure 4).

|  |  |
| --- | --- |
| **Pour les indemnes** | **Pour les blessés légers** |
| **Une image contenant texte, capture d’écran, ligne, bleu vert  Description générée automatiquement** | **Une image contenant capture d’écran, ligne, texte, bleu vert  Description générée automatiquement** |
|  |  |
| **Pour les blessés hospitalisés** | **Pour les tués** |
| **Une image contenant capture d’écran, ligne, texte, Tracé  Description générée automatiquement** | **Une image contenant texte, capture d’écran, ligne, Police  Description générée automatiquement** |

Figure 4 : Décompositions des séries temporelles

### Création d’une baseline

#### Méthode naïve avec un shift de 1

On crée une nouvelle colonne correspondant au décalage d’une journée des valeurs de la série. On fait de même pour tous les jeux de données et on évalue les résultats avec les métriques préalablement choisies (MAE, MSE et RMSE) pour obtenir les courbes suivantes (Figure 5) :

|  |  |
| --- | --- |
| **Pour les indemnes** | **Pour les blessés légers** |
| TRAIN ⇒ MAE 24.65 - MSE 959.63 - RMSE 30.98 | TRAIN ⇒ MAE 21.80 - MSE 739.50 - RMSE 27.19 |
|  |  |
| **Pour les blessés hospitalisés** | **Pour les tuées** |
| TRAIN ⇒ MAE 10.67 - MSE 181.71 - RMSE 13.48 | TRAIN ⇒ MAE : 3.50 - MSE : 20.94 -RMSE : 4.58 |

Figure 5: Baselines avec shift de 1 jour

#### Méthode naïve avec une moyenne sur 7 jours et un shift de 1

Comme les jeux de données présentent une saisonnalité hebdomadaire, nous décidons de créer une autre baseline moyennée sur 7 jours. La création de cette nouvelle colonne se fait en prenant les valeurs de la série sur 7 jours que l’on moyenne et que l’on décale d’une journée. En faisant de même avec tous les jeux de données et en calculant les métriques, on obtient les courbes de la Figure 6.

On observe que la moyenne sur 7 jours permet d’obtenir de meilleurs résultats ce qui tend à confirmer notre saisonnalité hebdomadaire.

|  |  |
| --- | --- |
| **Pour les indemnes** | **Pour les blessés légers** |
| TRAIN ⇒ MAE 20.23 - MSE 658.16 - RMSE 25.65 | TRAIN ⇒ MAE 17.57 - MSE 494.98 - RMSE 22.25 |
|  |  |
| **Pour les blessés hospitalisés** | **Pour les tués** |
| TRAIN ⇒ MAE 9.18 - MSE 134.24 - RMSE 11.59 | TRAIN ⇒ MAE 2.76 - MSE 12.52 - RMSE 3.54 |

Figure 6 : Baselines moyennées sur 7 jours + shift de 1 jour

#### SARIMAX + Exog(Série de Fourier)

Si la baseline moyennée à 7 jours permet de prendre en compte la saisonnalité hebdomadaire, elle ne prend pas en compte la saisonnalité annuelle. Pour cela, nous décidons d’effectuer une modélisation avec SARIMAX en prenant en variable exogène une série de Fourier. L’implémentation du modèle a été effectuée en se basant sur la méthodologie issue de l’analyse de la qualité de l’air expliquée dans <https://www.kaggle.com/code/saisatishmasina/sarima-with-fourier-terms>.

Avant de modéliser l'algorithme, on effectue :

* une vérification de la stationnarité des jeux de données avec un test ADF (Augmented Dickey-Fuller). La stationnarité est vérifiée pour les tués. Par contre, il faut faire une différenciation d’ordre 1 pour les autres jeux de données pour qu’ils deviennent stationnaires.
* une vérification de la saisonnalité annuelle en traçant le périodogramme et en vérifiant la densité spectrale liée à chaque période. La saisonnalité sur 365 jours est vérifiée pour tous les jeux de données.
* une séparation des jeux de données en train (90%) et test (10%) de manière chronologique.
* une extrapolation de la série de Fourier sur l’ensemble des données qui est ensuite intégrée dans train et test via une nouvelle variable exogène.
* une approximation des paramètres via le traçage des courbes d’autocorrélation et d’autocorrélation partielle. Dans tous les cas, afin d’obtenir un bruit blanc faible, nous rajoutons une différenciation en plus de celle déjà effectuée pour la stationnarisation : ce qui donne une différenciation d’ordre 1 pour les tués et une différenciation d’ordre 2 pour les autres modalités. Les courbes permettent aussi de vérifier la saisonnalité hebdomadaire.

Pour la modélisation avec SARIMAX, nous recherchons les paramètres p, q, P, Q qui minimisent l’AIC (Akaike Information Criterion). Pour cela, nous définissons les paramètres ainsi :

* l’ordre d de la partie non saisonnière est fixé à 1 pour les tués et à 2 pour les autres (comme nous l’avons défini lors du tracé des courbes d’autocorrélation et d’autocorrélation partielle),
* l’ordre D de la partie saisonnière est fixé à 1 pour prendre en compte la saisonnalité sur 7 jours,
* les ordres p, q, P et Q sont fixés entre 0 et 2 afin de trouver les meilleures valeurs.

De plus, on ajoute la variable exogène de train afin de prendre en compte la saisonnalité sur 365 jours.

Si les résultats obtenus ont des P>|z| supérieurs à 0.5, on modifie alors les meilleurs paramètres afin d’obtenir tous les P|z| inférieurs à 0.5. Les meilleurs modèles pour chaque modalité sont présentés sur la Figure 7.

Dans tous les cas, nous obtenons des modèles avec un bruit blanc et une hétéroscédasticité. Seuls les modèles pour les indemnes et les tués ne présentent pas de distribution normale (la distribution est très légèrement décalée à gauche).

Enfin, nous évaluons le modèle à la fois sur la partie train et sur la partie test (Figure 8).

obtiennent des moins bons résultats que la baseline moyennée sur 7 jours avec un shift de 1. De plus ces modèles nécessitent de connaître les valeurs pour faire une extrapolation de la série de Fourier et ne permettent donc pas de prédire les valeurs futures.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Pour les indemnes** | |  | **Pour les blessés hospitalisés** | |
| Une image contenant texte, capture d’écran, nombre, menu  Description générée automatiquement | Une image contenant Tracé, capture d’écran, ligne, diagramme  Description générée automatiquement  Prob(Q) > 5% : Bruit blanc  Prob (H) > 5% : Hétéroscédasticité  Prob (JB) < 5% : Pas de distribution normale |  | Une image contenant texte, capture d’écran, nombre, Police  Description générée automatiquement | Une image contenant Tracé, diagramme, ligne, capture d’écran  Description générée automatiquement  Prob(Q) > 5% : Bruit blanc  Prob (H) > 5% : Hétéroscédasticité  Prob (JB) > 5% : Distribution normale |
|  |  |  |  |  |
| **Pour les blessés légers** | |  | **Pour les tués** | |
| Une image contenant texte, capture d’écran, nombre, menu  Description générée automatiquement | Une image contenant Tracé, capture d’écran, diagramme, ligne  Description générée automatiquement  Prob(Q) > 5% : Bruit blanc  Prob (H) > 5% : Hétéroscédasticité  Prob (JB) > 5% : Distribution normale |  | Une image contenant texte, capture d’écran, nombre  Description générée automatiquement | Une image contenant Tracé, diagramme, ligne, capture d’écran  Description générée automatiquement  Prob(Q) > 5% : Bruit blanc  Prob (H) > 5% : Hétéroscédasticité  Prob (JB) < 5% : Pas de distribution normale |

Figure 7 : Résultats de SARIMAX avec en variable exogène une série de Fourier

|  |  |  |  |
| --- | --- | --- | --- |
| **Pour les indemnes** | | **Pour les blessés hospitalisés** | |
| Une image contenant capture d’écran, Tracé  Description générée automatiquement  TRAIN ⇒ MAE 33.77 - MSE 1715.31 - RMSE 41.42  TEST ⇒ MAE 21.87 - MSE 747.28 - RMSE 27.34 | Agrandissement de la partie droite de la courbe  Une image contenant Tracé, capture d’écran, ligne, texte  Description générée automatiquement | Une image contenant Tracé, texte, capture d’écran  Description générée automatiquement  TRAIN ⇒ MAE 14.75 - MSE 317.85 - RMSE 17.83  TEST ⇒ MAE 10.86 - MSE 168.03 - RMSE 12.96 | Agrandissement de la partie droite de la courbe  Une image contenant texte, Tracé, ligne, diagramme  Description générée automatiquement |
|  | |  |  |
| **Pour les blessés légers** | | **Pour les tués** | |
| Une image contenant capture d’écran, Tracé, ligne  Description générée automatiquement  TRAIN ⇒ MAE 33.12 - MSE 1588.26 - RMSE 39.85  TEST ⇒ MAE 21.35 - MSE 688.50 - RMSE 26.24 | Agrandissement de la partie droite de la courbe  Une image contenant texte, capture d’écran, Tracé, ligne  Description générée automatiquement | Une image contenant capture d’écran, Tracé, ligne  Description générée automatiquement  TRAIN ⇒ MAE 2.69 - MSE 11.31 - RMSE 3.36  TEST ⇒ MAE 2.90 - MSE 12.10 - RMSE 3.48 | Agrandissement de la partie droite de la courbe  Une image contenant texte, Tracé, capture d’écran, ligne  Description générée automatiquement |

Figure 8 : Évaluations de SARIMAX avec en variable exogène une série de Fourier sur train et test

### Prédictions

Pour prédire le nombre d’indemnes, blessés légers, blessés hospitalisés et tués des 6 prochains mois (les 6 premiers mois de 2023), nous utilisons 3 algorithmes :

* 2 algorithmes de machine learning : MSTL (Multiple Seasonal-Trend decomposition using LOESS) et PROPHET,
* 1 algorithme de deep learning : LSTM (Long Short Term Memory).

#### Préparation des jeux de données pour MSTL et PROPHET

Pour implémenter ces algorithmes, il est nécessaire que les jeux de données soient d’une forme particulière. Pour cela, nous transformons nos jeux de données (exemple des premières lignes pour les indemnes):

* création de la colonne ‘ds’ à partir de l’index
* création de la colonne ‘y’ par renommage de la colonne ’Nbre\_Acc’
* création de la colonne ‘unique\_id’ en lui donnant la valeur 1 pour toutes les lignes

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Nbre\_Acc |  | |  | ds | y | unique\_id |
| 2021-01-01 | 71 | **⇒** | 0 | | 2021-01-01 | 71 | 1 |
| 2021-01-02 | 68 | 1 | | 2021-01-02 | 68 | 1 |

Puis on sépare le jeu de données en train (90%) et test (10%) de manière chronologique.

#### MSTL

Les modèles MSTL ont été implémentés à l’aide de la documentation de NIXTLA : <https://nixtlaverse.nixtla.io/statsforecast/docs/models/multipleseasonaltrend.html/>.

Dans les modèles MSTL, nous définissons 2 paramètres :

* season\_length qui est défini à [7, 7 \* 52] pour la saisonnalité hebdomadaire et la saisonnalité annuelle,
* trend\_forcaster où nous essayons successivement 3 types de tendance (AutoARIMA, AutoTheta et AutoCES).

De plus, nous mettons une fréquence ‘D’ pour l’implémentation du StatsForecast pour avoir une fréquence des jours calendaires.

Exemple d’implémentation avec un trend\_forcaster basé sur AutoARIMA :





Les résultats obtenus pour chacun de ces modèles sont les suivants (Figure 9). Dans tous les cas, un surapprentissage sur la partie train tend à apparaître

|  |  |  |
| --- | --- | --- |
| **Pour les indemnes** | | |
| **MSTL + Trend (AutoARIMA)** |  | TRAIN ⇒ MAE 9.29 - MSE 155.71 - RMSE 12.48  TEST ⇒ MAE 22.17 - MSE 888.29 - RMSE 29.80 |
| **MSTL + Trend (AutoTheta)** |  | TRAIN ⇒ MAE 9.37 - MSE 165.83 - RMSE 12.88  TEST ⇒ MAE 23.70 - MSE 1008.28 - RMSE 31.75 |
| **MSTL + Trend (AutoCES)** |  | TRAIN ⇒ MAE 9.40 - MSE 165.67 - RMSE 12.87  TEST ⇒ MAE 26.39 - MSE 1194.27 - RMSE 34.56 |

Figure 9 : Évaluation de MSTL avec différents trend\_forecaster sur train et test

|  |  |  |
| --- | --- | --- |
| **Pour les blessés légers** | | |
| **MSTL + Trend (AutoARIMA)** |  | TRAIN ⇒ MAE 8.70 - MSE 135.01 - RMSE 11.62  TEST ⇒ MAE 19.50 - MSE 630.39 - RMSE 25.11 |
| **MSTL + Trend (AutoTheta)** |  | TRAIN ⇒ MAE 8.89 - MSE 142.09 - RMSE 11.92  TEST ⇒ MAE 20.32 - MSE 692.53 - RMSE 26.32 |
| **MSTL + Trend (AutoCES)** |  | TRAIN ⇒ MAE 8.90 - MSE 141.97 - RMSE 11.92  TEST ⇒ MAE 21.52 - MSE 760.91 - RMSE 27.58 |

Figure 9 : Évaluation de MSTL avec différents trend\_forecaster sur train et test

|  |  |  |
| --- | --- | --- |
| **Pour les blessés hospitalisés** | | |
| **MSTL + Trend (AutoARIMA)** |  | TRAIN ⇒ MAE 4.26 - MSE 31.27 - RMSE 5.60  TEST ⇒ MAE 10.81 - MSE 197.56 - RMSE 14.06 |
| **MSTL + Trend (AutoTheta)** |  | TRAIN ⇒ MAE 4.38 - MSE 33.32 - RMSE 5.77  TEST ⇒ MAE 11.02 - MSE 203.41 - RMSE 14.26 |
| **MSTL + Trend (AutoCES)** |  | TRAIN ⇒ MAE 4.36 - MSE 33.02 - RMSE 5.75  TEST ⇒ MAE 11.23 - MSE 209.58 - RMSE 14.48 |

Figure 9 : Évaluation de MSTL avec différents trend\_forecaster sur train et test

|  |  |  |
| --- | --- | --- |
| **Pour les tués** | | |
| **MSTL + Trend (AutoARIMA)** |  | TRAIN ⇒ MAE 1.44 - MSE 3.89 - RMSE 1.97  TEST ⇒ MAE 3.27 - MSE 18.69 - RMSE 4.32 |
| **MSTL + Trend (AutoTheta)** |  | TRAIN ⇒ MAE 1.46 - MSE 3.99 - RMSE 2.00  TEST ⇒ MAE 3.26 - MSE 18.80 - RMSE 4.34 |
| **MSTL + Trend (AutoCES)** |  | TRAIN ⇒ MAE 1.42 - MSE 3.87 - RMSE 1.97  TEST ⇒ MAE 3.26 - MSE 18.93 - RMSE 4.35 |

Figure 9 : Évaluation de MSTL avec différents trend\_forecaster sur train et test

#### PROPHET

L’implémentation de cet algorithme a été réalisée à l’aide de la documentation de PROPHET : <https://facebook.github.io/prophet/>.

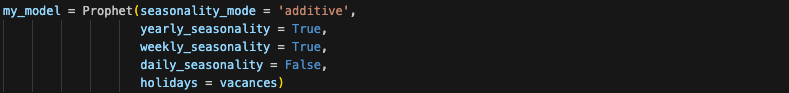
Pour cet algorithme, nous avons fait le choix de l’utiliser de 3 manières :

* avec les paramètres yearly\_seasonality et weekly\_seasonality sur True pour indiquer la saisonnalité annuelle et hebdomadaire,
* avec ajout des vacances scolaires communes à la France métropolitaine et aux DOM-TOM,
* avec ajout des jours fériés communs à la France métropolitaine et aux DOM-TOM.

Pour ajouter les vacances scolaires ou les jours fériés, il est nécessaire de créer un nouveau jeu de données de la forme suivante (exemple des premières lignes pour les vacances scolaires ) :

|  |  |  |
| --- | --- | --- |
|  | holiday | ds |
| 0 | vacances | 2021-01-01 |
| 1 | vacances | 2021-01-02 |

Exemple d’implémentation avec les vacances scolaires :



Les résultats obtenus pour chacun de ces modèles sont présentés à la Figure 10.

|  |  |  |
| --- | --- | --- |
| **Pour les indemnes** | | |
| **PROPHET**  **+**  **saisonnalité annuelle et hebdomadaire** |  | TRAIN ⇒ MAE 16.13- MSE 428.17 - RMSE 20.69  TEST ⇒ MAE 22.76 - MSE 832.50 - RMSE 28.85 |
| **PROPHET**  **+**  **saisonnalité annuelle et hebdomadaire**  **+**  **vacances scolaires** |  | TRAIN ⇒ MAE 16.03 - MSE 416.33 - RMSE 20.40  TEST ⇒ MAE 23.95 - MSE 886.87 - RMSE 29.78 |
| **PROPHET**  **+**  **saisonnalité annuelle et hebdomadaire**  **+**  **jours fériés** |  | TRAIN ⇒ MAE 15.60 - MSE 401.42 - RMSE 20.04  TEST ⇒ MAE 22.12 - MSE 792.38 - RMSE 28.15 |

Figure 10 : Évaluation de PROPHET avec différentes saisonnalités sur train et test

|  |  |  |
| --- | --- | --- |
| **Pour les blessés légers** | | |
| **PROPHET**  **+**  **saisonnalité annuelle et hebdomadaire** |  | TRAIN ⇒ MAE 14.92 - MSE 344.16 - RMSE 18.55  TEST ⇒ MAE 26.98 - MSE 1015.31 - RMSE 31.86 |
| **PROPHET**  **+**  **saisonnalité annuelle et hebdomadaire**  **+**  **vacances scolaires** |  | TRAIN ⇒ MAE 14.69 - MSE 333.32 - RMSE 18.26  TEST ⇒ MAE 27.98 - MSE 1036.04 - RMSE 32.19 |
| **PROPHET**  **+**  **saisonnalité annuelle et hebdomadaire**  **+**  **jours fériés** |  | TRAIN ⇒ MAE 14.66 - MSE 330.53 - RMSE 18.18  TEST ⇒ MAE 25.92- MSE 963.25 - RMSE 31.04 |

Figure 10 : Évaluation de PROPHET avec différentes saisonnalités sur train et test

|  |  |  |
| --- | --- | --- |
| **Pour les blessés hospitalisés** | | |
| **PROPHET**  **+**  **saisonnalité annuelle et hebdomadaire** |  | TRAIN ⇒ MAE 7.39 - MSE 86.42 - RMSE 9.30  TEST ⇒ MAE 9.42 - MSE 127.08 - RMSE 11.27 |
| **PROPHET**  **+**  **saisonnalité annuelle et hebdomadaire**  **+**  **vacances scolaires** |  | TRAIN ⇒ MAE 7.35 - MSE 85.91 - RMSE 9.27  TEST ⇒ MAE 9.41 - MSE 126.29 - RMSE 11.24 |
| **PROPHET**  **+**  **saisonnalité annuelle et hebdomadaire**  **+**  **jours fériés** |  | TRAIN ⇒ MAE 7.38 - MSE 85.33 - RMSE 9.29  TEST ⇒ MAE 9.39 - MSE 126.84 - RMSE 11.26 |

Figure 10 : Évaluation de PROPHET avec différentes saisonnalités sur train et test

|  |  |  |
| --- | --- | --- |
| **Pour les tués** | | |
| **PROPHET**  **+**  **saisonnalité annuelle et hebdomadaire** |  | TRAIN ⇒ MAE 2.36 - MSE 9.13 - RMSE 3.02  TEST ⇒ MAE 2.78 - MSE 11.20 - RMSE 3.35 |
| **PROPHET**  **+**  **saisonnalité annuelle et hebdomadaire**  **+**  **vacances scolaires** |  | TRAIN ⇒ MAE 2.36 - MSE 9.13 - RMSE 3.02  TEST ⇒ MAE 2.79 - MSE 11.23 - RMSE 3.35 |
| **PROPHET**  **+**  **saisonnalité annuelle et hebdomadaire**  **+**  **jours fériés** |  | TRAIN ⇒ MAE 2.35 - MSE 9.11 - RMSE 3.02  TEST ⇒ MAE 2.75 - MSE 11.04 - RMSE 3.32 |

Figure 10 : Évaluation de PROPHET avec différentes saisonnalités sur train et test

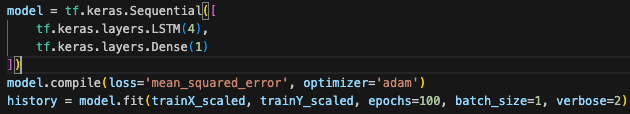
#### LSTM

L’implémentation du modèle a été réalisée à l’aide de la publication : <https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/>.

Pour le deep learning, il est nécessaire de modifier notre jeu de données de la manière suivante

* création d’un jeu de données composé d’un array numpy contenant les valeurs de la colonne ‘Nbre\_Acc’,
* normalisation des données avec MinMaxScaler,
* séparation en train\_scaled et test\_scaled,
* conversion de train\_scaled et de test\_scaled en une matrice de séquences de données (avec le look\_back qui correspond au nombre de pas de temps précédents à utiliser comme variables d'entrée pour prédire la période suivante),
* remodelage de train\_scaled et de test\_scaled pour qu’il ait un format [samples, time steps, features]

Nous choisissons d’implémenter un Vanilla MSTL ne comportant qu’une couche LSTM comportant 4 neurones et une couche Dense. Le modèle est compilé avec comme paramètres : loss = ‘mean\_squared\_error’ et optimizer = ‘adam’. Le modèle est ensuite entraîné sur 100 epochs sur chacun des jeux de données .



Le modèle est ensuite évalué sur les parties train et test. Les résultats obtenus pour un look\_back = 31 (soit 31 jours) sont présentés sur la Figure 11.

En augmentant le look\_back à 2 mois, les résultats des métriques sont moins bons. Le modèle LSTM est donc un bon modèle pour des prédictions à courte échéance pour nos jeux de données.

|  |  |
| --- | --- |
| **Pour les indemnes** | **Pour les blessés légers** |
| Une image contenant capture d’écran, texte, Tracé, ligne  Description générée automatiquement  TRAIN ⇒ MAE 16.48 - MSE 444.77 - RMSE 21.29  TEST ⇒ MAE 19.29 - MSE 599.27 - RMSE 24.48 | Une image contenant texte, Tracé, ligne, diagramme  Description générée automatiquement  TRAIN ⇒ MAE 14.97 - MSE 361.01 - RMSE 19.00  TEST ⇒ MAE 17.13 - MSE 425.66 - RMSE 20.63 |
|  |  |
| **Pour les blessés hospitalisés** | **Pour les tués** |
| Une image contenant capture d’écran, texte, Tracé, ligne  Description générée automatiquement  TRAIN ⇒ MAE 7.07 - MSE 81.45 - RMSE 9.02  TEST ⇒ MAE 7.86 - MSE 81.86 - RMSE 9.05 | Une image contenant capture d’écran, texte, Tracé  Description générée automatiquement  TRAIN ⇒ MAE 2.25 - MSE 8.02 - RMSE 2.83  TEST ⇒ MAE 2.92 - MSE 13.40 - RMSE 3.66 |

Figure 11 : Évaluations de LSTM sur train et test

#### Choix du meilleur modèle et prévision pour les futures dates

En comparant les résultats obtenus pour les différents algorithmes, nous sélectionnons celui qui minimise la MAE et la RMSE.

Le modèle LSTM ne permet de prédire qu’à courte échéance. En effet, les prévisions se font de manière itérative : prédire une nouvelle valeur, l’ajouter aux valeurs précédentes puis prédire la valeur suivante, et ainsi de suite. Donc plus on s’éloigne des valeurs initiales, plus les résultats risquent d’être biaisés. Nous avons choisi de prédire les valeurs pour le prochain mois avec MSTL afin de visualiser la tendance sur les courbes, mais il ne faudra certainement utiliser que les premières valeurs.

Nous avons retenu les modèles suivants selon la modalité de ‘grav’ (Figure 12) :

* Pour les **indemnes**, les meilleurs résultats sont obtenus pour **LSTM (avec look\_back de 31 jours)**. Or celui-ci ne permet de prédire correctement que des données pour les 31 jours suivants. Donc pour une prédiction à plus long terme, il faudrait utiliser **PROPHET avec ajout des jours fériés**.
* Pour les **blessés légers**, les meilleurs résultats sont obtenus pour **LSTM (avec look\_back de 31 jours)**. Or celui-ci ne permet de prédire correctement que des données pour les 31 jours suivants. Donc pour une prédiction à plus long terme, il faudrait utiliser **MSTL avec comme trend\_forcaster AutoARIMA.**
* Pour les **blessés hospitalisés**, les meilleurs résultats sont obtenus pour **LSTM (avec look\_back de 31 jours)**. Or celui-ci ne permet de prédire correctement que des données pour les 31 jours suivants. Donc pour une prédiction à plus long terme, il faudrait utiliser **PROPHET avec ajout des jours fériés**.
* Pour les **tués**, les meilleurs résultats sont obtenus pour **PROPHET avec ajout des jours fériés.**

### Conclusion

Pour nos jeux de données (en dehors des tués), les prédictions sont nettement améliorées avec un modèle de deep learning simple. Il pourrait encore être optimisé avec des modèles plus complexes comme Stacked LSTM, Bidirectional LSTM, CNN LSTM, ConvLSTM.

Cependant, même si les résultats sont meilleurs, cela ne permet de prédire qu’à court, voire très court terme. Il est donc nécessaire d’avoir d’autres modèles de machine learning pour faire des prédictions à plus longue échéance.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Pour les indemnes** | | | | | | | |
|  | **MAE** | | **MSE** | | | **RMSE** | |
| **Baseline (shift de 1)** | 24.65 | | 959.63 | | | 30.98 | |
| **Baseline (moyenne sur 7 jours + shift de 1)** | **20.23** | | **658.16** | | | **25.65** | |
|  | | | | | | | |
|  | **Train** | | | | **Test** | | |
|  | **MAE** | **MSE** | **RMSE** | | **MAE** | **MSE** | **RMSE** |
| **SARIMAX + exog(Fourier)** | 33.77 | 1715.31 | 41.42 | | 21.87 | 747.28 | 27.34 |
| **MSTL + AutiARIMA** | 9.29 | 155.71 | 12.48 | | 22.17 | 888.29 | 29.80 |
| **MSTL + AutoTheta** | 9.37 | 165.83 | 12.88 | | 23.70 | 1008.28 | 31.75 |
| **MSTL + AutoCES** | 9.40 | 165.67 | 12.87 | | 26.39 | 1194.27 | 34.56 |
| **PROPHET** | 16.13 | 428.17 | 20.69 | | 22.76 | 832.50 | 28.85 |
| **PROPHET + vacances scolaires** | 16.03 | 416.33 | 20.40 | | 23.95 | 886.87 | 29.78 |
| **PROPHET + jours fériés** | **15.60** | **401.42** | **20.04** | | **22.12** | **792.38** | **28.15** |
| **LSTM (look\_back de 31 jours)** | **16.47** | **444.77** | **21.09** | | **19.29** | **599.27** | **24.48** |
|  | | | | | | | |
| **Prévisions à 1 mois avec LSTM (look\_back de 31 jours)**  Une image contenant Tracé, texte, ligne, Police  Description générée automatiquement | | | | **Prévisions à 6 mois avec PROPHET + jours fériés**  **Une image contenant texte, Tracé, ligne, Police  Description générée automatiquement** | | | |

Figure 12 : Comparaison des modèles et prédictions à 1 et 6 mois pour les indemnes

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Pour les blessés légers** | | | | | | | |
|  | **MAE** | | **MSE** | | | **RMSE** | |
| **Baseline (shift de 1)** | 21.80 | | 739.50 | | | 27.19 | |
| **Baseline (moyenne sur 7 jours + shift de 1)** | **17.57** | | **494.98** | | | **22.25** | |
|  | | | | | | | |
|  | **Train** | | | | **Test** | | |
|  | **MAE** | **MSE** | **RMSE** | | **MAE** | **MSE** | **RMSE** |
| **SARIMAX + exog(Fourier)** | 33.12 | 317.85 | 17.83 | | 21.35 | 688.50 | 26.24 |
| **MSTL + AutiARIMA** | **8.70** | **135.01** | **11.62** | | **19.50** | **630.39** | **25.11** |
| **MSTL + AutoTheta** | 8.89 | 142.09 | 11.92 | | 20.32 | 692.53 | 26.32 |
| **MSTL + AutoCES** | 8.90 | 141.97 | 11.92 | | 21.52 | 760.91 | 27.58 |
| **PROPHET** | 14.92 | 344.16 | 18.55 | | 26.98 | 1015.31 | 31.86 |
| **PROPHET + vacances scolaires** | 14.69 | 333.32 | 18.26 | | 27.98 | 1036.04 | 32.19 |
| **PROPHET + jours fériés** | 14.66 | 330.53 | 18.18 | | 25.92 | 963.25 | 31.04 |
| **LSTM (look\_back de 31 jours)** | **14.97** | **361.01** | **19.00** | | **17.13** | **425.66** | **20.63** |
|  | | | | | | | |
| **Prévisions à 1 mois avec LSTM (look\_back de 31 jours)**  **Une image contenant Tracé, ligne, Police, texte  Description générée automatiquement** | | | | **Prévisions à 6 mois avec MSTL + AutiARIMA**  **Une image contenant Tracé, texte, ligne, Police  Description générée automatiquement** | | | |

Figure 13 : Comparaison des modèles et prédictions à 1 et 6 mois pour les blessés légers

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Pour les blessés hospitalisés** | | | | | | | |
|  | **MAE** | | **MSE** | | | **RMSE** | |
| **Baseline (shift de 1)** | 10.67 | | 181.71 | | | 13.48 | |
| **Baseline (moyenne sur 7 jours + shift de 1)** | **9.18** | | **134.24** | | | **11.59** | |
|  | | | | | | | |
|  | **Train** | | | | **Test** | | |
|  | **MAE** | **MSE** | **RMSE** | | **MAE** | **MSE** | **RMSE** |
| **SARIMAX + exog(Fourier)** | 14.75 | 317.85 | 17.83 | | 10.86 | 168.03 | 12.96 |
| **MSTL + AutiARIMA** | 4.26 | 31.27 | 5.60 | | 10.81 | 197.56 | 14.06 |
| **MSTL + AutoTheta** | 4.38 | 33.32 | 5.77 | | 11.02 | 203.41 | 14.26 |
| **MSTL + AutoCES** | 4.36 | 33.02 | 5.75 | | 11.23 | 209.58 | 14.48 |
| **PROPHET** | 7.39 | 86.42 | 9.30 | | 9.42 | 127.08 | 11.27 |
| **PROPHET + vacances scolaires** | 7.35 | 85.91 | 9.27 | | 9.41 | 126.29 | 11.24 |
| **PROPHET + jours fériés** | **7.38** | **85.33** | **9.29** | | **9.39** | **126.84** | **11.26** |
| **LSTM (look\_back de 31 jours)** | **7.07** | **81.45** | **9.02** | | **7.86** | **81.86** | **9.05** |
|  | | | | | | | |
| **Prévisions à 1 mois avec LSTM (look\_back de 31 jours)**  **Une image contenant Tracé, ligne, texte, Police  Description générée automatiquement** | | | | **Prévisions à 6 mois avec PROPHET + jours fériés**  **Une image contenant Tracé, texte, ligne, Police  Description générée automatiquement** | | | |

Figure 14 : Comparaison des modèles et prédictions à 1 et 6 mois pour les blessés hospitalisés

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Pour les tués** | | | | | | |
|  | **MAE** | | **MSE** | | **RMSE** | |
| **Baseline (shift de 1)** | 3.50 | | 20.94 | | 4.58 | |
| **Baseline (moyenne sur 7 jours + shift de 1)** | **2.76** | | **12.52** | | **3.54** | |
|  | | | | | | |
|  | **Train** | | | **Test** | | |
|  | **MAE** | **MSE** | **RMSE** | **MAE** | **MSE** | **RMSE** |
| **SARIMAX + exog(Fourier)** | 2.69 | 11.31 | 3.36 | 2.90 | 12.10 | 3.48 |
| **MSTL + AutiARIMA** | 1.44 | 3.89 | 1.97 | 3.27 | 18.69 | 4.32 |
| **MSTL + AutoTheta** | 1.46 | 3.99 | 2.00 | 3.26 | 18.80 | 4.34 |
| **MSTL + AutoCES** | 1.42 | 3.87 | 1.97 | 3.26 | 18.93 | 4.35 |
| **PROPHET** | 2.36 | 9.13 | 3.02 | 2.78 | 11.20 | 3.35 |
| **PROPHET + vacances scolaires** | 2.36 | 9.13 | 3.02 | 2.79 | 11.23 | 3.35 |
| **PROPHET + jours fériés** | **2.35** | **9.11** | **3.02** | **2.75** | **11.04** | **3.32** |
| **LSTM (look\_back de 31 jours)** | 2.25 | 8.02 | 2.83 | 2.93 | 13.40 | 3.66 |
|  | | | | | | |
| **Prévisions à 6 mois avec PROPHET + jours fériés**  Une image contenant capture d’écran, Tracé, texte, ligne  Description générée automatiquement | | | | | | |

Figure 15 : Comparaison des modèles et prédictions à 1 et 6 mois pour les tués

# PRÉDICTION DE LA GRAVITÉ DE L’ACCIDENT

## Régression logistique

### Modèle de référence

#### Description du modèle

Le modèle est construit sous forme de **pipeline** en 2 étapes :

* Etape 1 : preprocessor de type **ColumnTransformer** où sont appliquées les étapes de preprocessing décidées à l’issue du rapport 1 :
  + Transformation des heures et des mois
  + Transformation des latitudes et longitudes (RobustScaler)
  + Transformation de l’âge (StandardScaler)
  + Encodage des variables catégorielles (one\_hot\_encoder)
  + Passage des autres variables de la base de données (remainder)
* Etape 2 : modèle de régression logistique

#### Paramètres du modèle de référence

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| penalty | ‘l2’ | intercept\_scaling | 1 | multi\_class | ‘ovr’ |
| dual | False | class\_weight | ‘balanced’ | verbose | 0 |
| tol | 0.0001 | random\_state | None | warm\_start | False |
| C | 1.0 | solver | ‘lbfgs’ | n\_jobs | None |
| fit\_intercept | True | max\_iter | 5000 | l1\_ratio | None |

Tableau 1 : Paramètres du modèle de régression logistique pris comme référence

Une validation croisée sur 3 échantillons conduit à un score de :

* 0.6008 ± 0.0007 sur l’échantillon d’apprentissage,
* et de 0.5990 ± 0.0047 sur l’échantillon de test.

Le modèle est donc capable de généraliser sans problèmes de sur- ou de sous-apprentissage notables.

La Figure 16 présente les performances obtenues pour chacune des classes, et met en lumière la difficulté du modèle à gérer la classe 2 (des personnes tuées lors de l’accident), qui est largement minoritaire dans le jeu de données. La précision de cette classe n’est pas bonne (0.14) puisque le modèle prédit comme tués de nombreux accidentés des classes 1,3 et 4. Le rappel de la classe 2 est également relativement moyen (0.46), le modèle ayant tendance à positionner les personnes réellement tuées (classe 2) dans la classe des blessés hospitalisés (classe 3). La distinction entre ces 2 classes peut effectivement être relativement difficile et potentiellement relever de caractéristiques qui ne figurent pas dans la base de données mise à notre disposition (antécédents médicaux des accidentés par exemple).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (a) | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  | Classes prédites | | | | |  |  | **1** | **2** | **3** | **4** | | **Classes réelles** | **1** | 37212 | 1973 | 2079 | 4873 | | **2** | 255 | 1412 | 1003 | 380 | | **3** | 1932 | 3575 | 7097 | 4896 | | **4** | 12986 | 3423 | 7194 | 21494 |   (b) |

Figure 16 : Métriques (a) et matrice de confusion (b) du modèle de régression logistique de référence

### Optimisation du modèle

#### Ajustement de paramètres

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Méthode de validation | Référence | Nouveau modèle | Décision |
| Ajout d’un undersampler | Validation croisée (cv=3) | 0.5990 ± 0.0047 | 0.5672 ± 0.0051 | Pas d’undersampler |
| Paramètre multi\_class en ‘auto’ | Validation croisée (cv=3) | 0.5706 ± 0.0063 | Conservation du paramètre  multi\_class = ‘ovr’ |
| Solver ‘newton-cg’ | Validation croisée (cv=3) |  | newton-cg est plus rapide |

Tableau 2 : Méthodes d’ajustement des paramètres

#### Ajustement des hyperparamètres penalty et C avec GridSearchCV

A été également étudiée l’influence de :

* la pénalité (norme L1 ou L2) : le solveur par défaut ne pouvant pas gérer la norme L1, le solveur ‘liblinear’ a été utilisé avec la pénalité ‘L1’, et ‘newton-cg’ avec ‘L2’,
* du paramètre de régularisation C : valeurs étudiées : [0.01, 0.1, 1, 10].

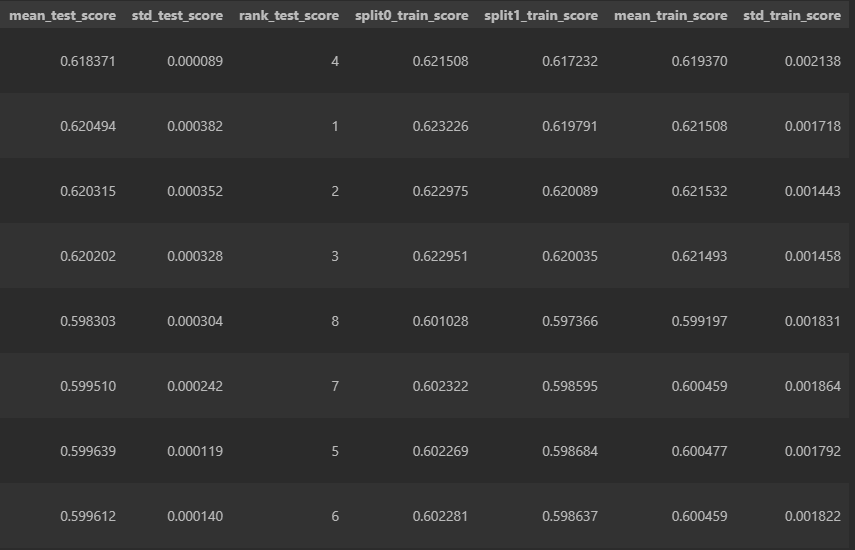
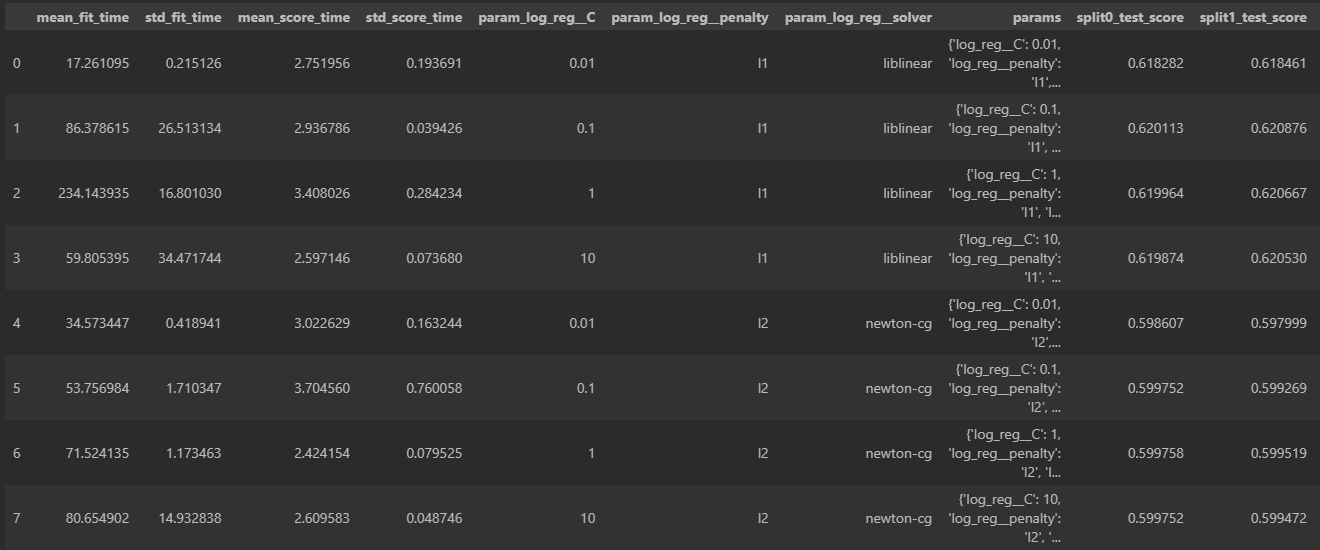


Figure 17 : Résultats du GridSearchCV sur penalty et C, hyperparamètres de la régression logistique

La Figure 17 souligne que, de façon générale, la pénalité L1 (avec le solveur liblinear) conduit à de meilleurs résultats que la pénalité L2. Parmi les coefficients de régularisation analysés, C=0.1 est celui ayant conduit aux meilleurs résultats.

*Figure SEQ Figure \\* ARABIC 2 : Résultats du GridSearchCV sur penalty et C, hyperparamètres de la régression logistique*

Les paramètres du modèle optimisé sont donc récapitulés dans le Tableau 3 et les résultats de ce modèle sur l’échantillon de test sont donnés dans la Figure 18, en termes de métriques et de matrice de confusion.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| penalty | ‘l1’ | intercept\_scaling | 1 | multi\_class | ‘ovr’ |
| dual | False | class\_weight | ‘balanced’ | verbose | 0 |
| tol | 0.0001 | random\_state | None | warm\_start | False |
| C | 0.1 | solver | ‘liblinear’ | n\_jobs | None |
| fit\_intercept | True | max\_iter | 5000 | l1\_ratio | None |

Tableau 3 : Paramètres du modèle de régression logistique après optimisation

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (a) | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  | Classes prédites | | | | |  |  | **1** | **2** | **3** | **4** | | **Classes réelles** | **1** | 37877 | 1183 | 1690 | 5387 | | **2** | 356 | 1143 | 1017 | 534 | | **3** | 2293 | 2589 | 6283 | 6335 | | **4** | 13439 | 2254 | 5214 | 24190 |   (b) |

Figure 18 : Métriques (a) et matrice de confusion (b) du modèle de régression logistique optimisé

L’amélioration globale n’est pas flagrante. On note une amélioration des précisions des classes 2 et 3 (aux dépens des rappels de ces classes), et un f1-score légèrement amélioré pour la classe 2.

### Interprétabilité des résultats

#### Analyse des coefficients de la régression logistique

La Figure 19 présente les boîtes à moustaches des coefficients de la régression logistique (modèle optimisé) obtenus lors d’une validation croisée à 3 échantillons. Il ressort de cette figure la forte influence de :

* Place\_rec\_4.0 : le fait d’être piéton
* Obsm\_1.0 : obstacle mobile heurté de type piéton
* Les catégories de véhicules, dans l’ordre, vélos, motos, poids lourds, autres, transports en commun et voitures,
* Situ\_5.0 : la présence sur une piste cyclable

On remarque que les valeurs absolues des coefficients associés aux variables continues restent relativement faibles. Il est envisageable que cela soit un inconvénient de la régression logistique, qui, cherchant des relations linéaires entre variables, ne permet pas de faire ressortir une influence potentiellement non linéaire des variables continues.

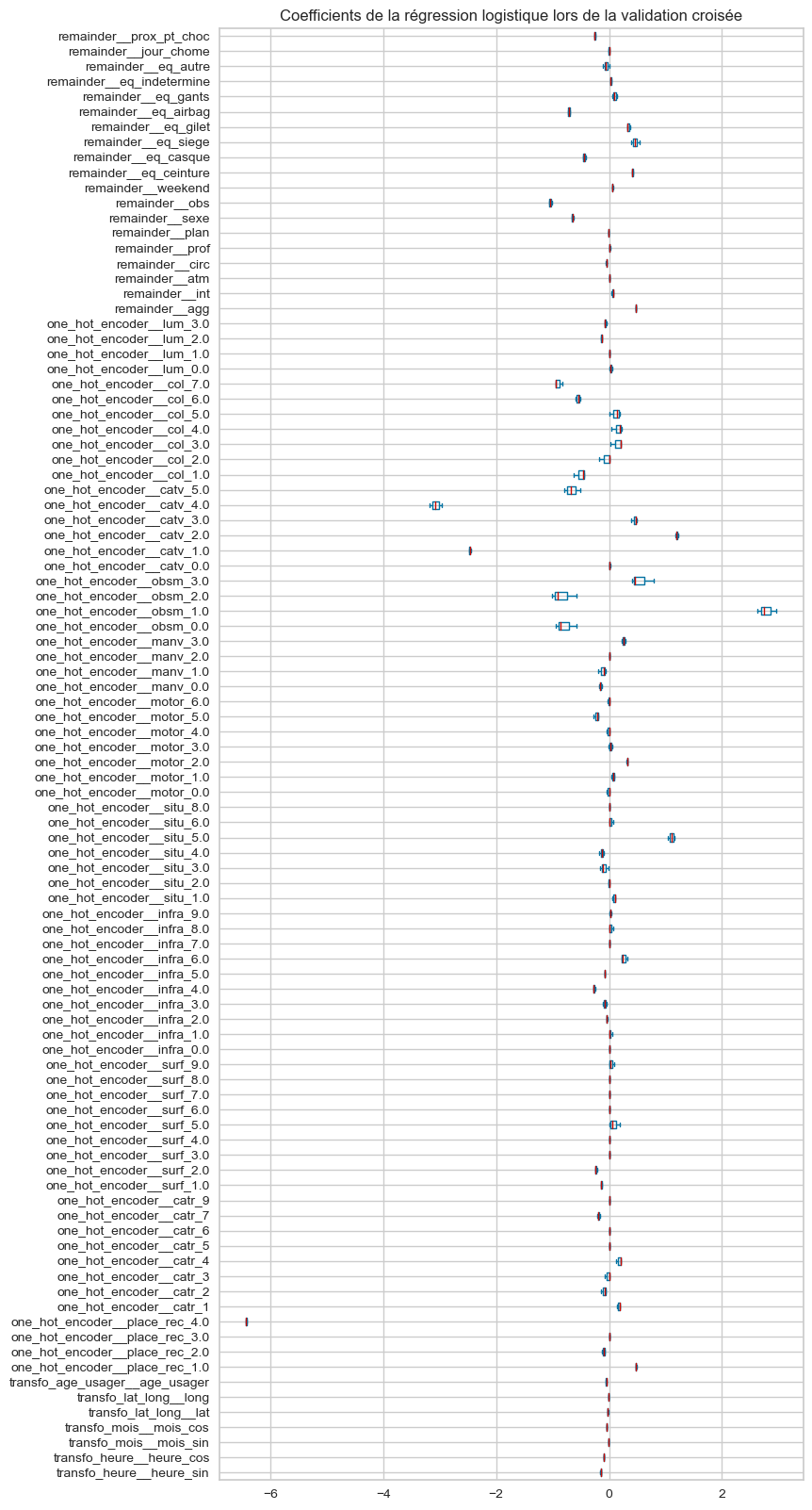


Figure 19 : Coefficients du modèle de régression logistique optimisé lors de la validation croisée

#### Analyse factorielle de données mixtes

Une analyse factorielle de données mixtes a été menée à l’aide de la bibliothèque prince.

La Figure 20 présente l’évolution du pourcentage de variance expliquée en fonction du nombre de composantes. La courbe associée à l’axe des ordonnées de gauche donne le cumul de variance expliquée en fonction du nombre de composantes retenues, tandis que l’histogramme, associé à l’axe des ordonnées de droite, renseigne sur la contribution de chaque composante.

**Cette analyse confirme la complexité de notre jeu de données puisqu’il faut une trentaine de composantes pour atteindre 80% de la variance expliquée, et laisse supposer l’absence dans la base de variables explicatives fortement informatives sur la variance de notre jeu de données.**

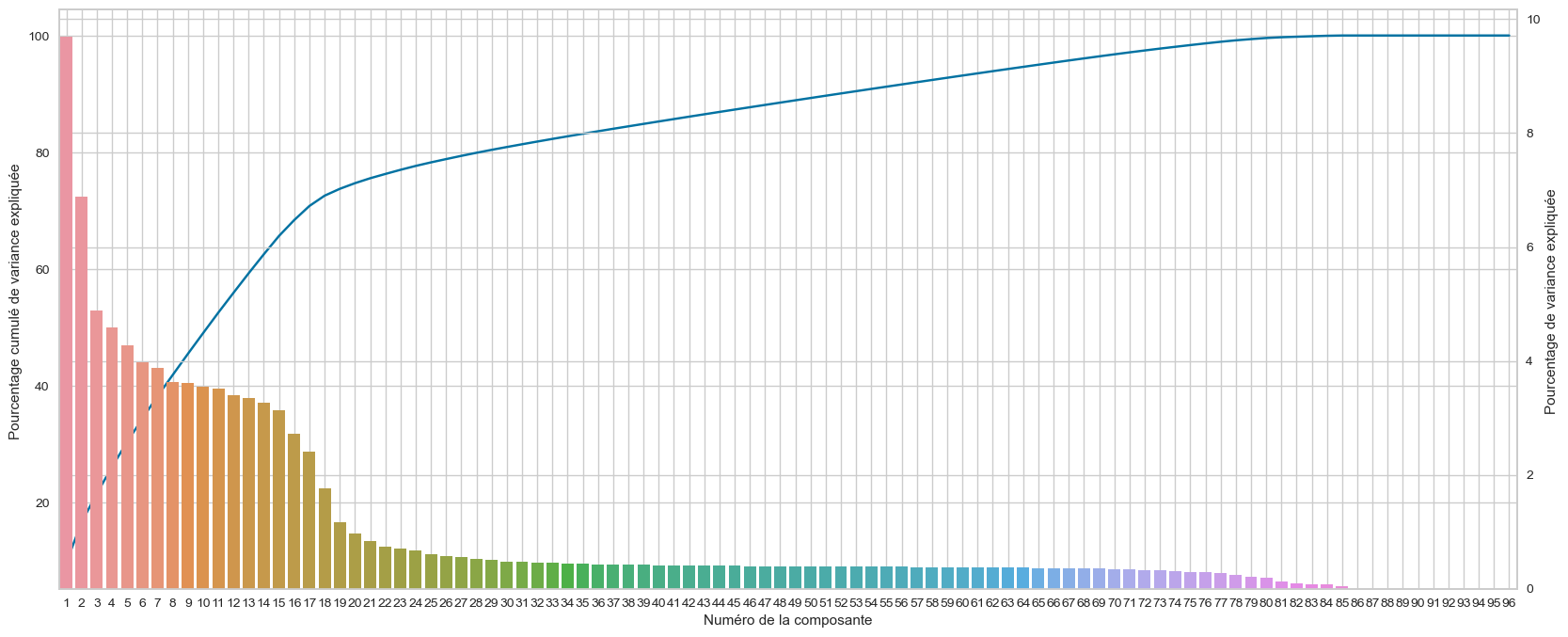


Figure 20 : Variance expliquée, en cumul (axe de gauche) ou unitairement (axe de droite), en fonction du nombre de composantes retenues

A titre d’informations, la Figure 21 présente les variables qui contribuent le plus fortement aux deux premières composantes de l’analyse factorielle (qui ne permettent cependant d’expliquer que 15% de la variance de notre jeu de données)

|  |  |
| --- | --- |
| 1. Première composante | 1. Deuxième composante |

Figure 21 : Les 20 variables contribuant le plus aux deux premières composantes de l’analyse factorielle

## Support Vector Machine

### Modèle de référence

#### Description du modèle

La complexité des SVM est comprise entre

et , ce qui est difficilement envisageable pour notre problème présentant une centaine de variables et près de 500 000 observations. Il est alors plutôt recommandé de considérer un modèle de type LinearSVC ou SGDClassifier, après éventuellement une approximation de type Nystroëm pour introduire des non-linéarités dans le modèle.

Le modèle est construit sous forme de pipeline en 3 étapes :

- Etape 1 : preprocessor de type ColumnTransformer où sont appliquées les étapes de preprocessing décidées à l’issue du rapport 1 :

o Transformation des heures et des mois

o Transformation des latitudes et longitudes (RobustScaler)

o Transformation de l’âge (StandardScaler)

o Encodage des variables catégorielles (one\_hot\_encoder)

o Passage des autres variables de la base de données (remainder)

- Etape 2 : approximation de Nyström

- Etape 3 : modèle de type Linear SVC

#### Paramètres

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nystroëm | kernel | ‘rbf’ | gamma | None | coef0 | None |
| degree | 2 | kernel\_params | None | n\_components | 300 |
| random\_state | None | n\_jobs | None |  |  |
| LinearSVC | penalty | ‘l2’ | C | 1.0 | multi\_class | ‘ovr’ |
| loss | ‘squared\_hinge’ | fit\_intercept | True | verbose | 0 |
| dual | ‘warn’ | class\_weight | ‘balanced’ | intercept\_scaling | 1 |
| tol | 0.0001 | random\_state | 42 | max\_iter | 1000 |

Tableau 4 : Paramètres de l’approximation Nyström et du modèle de LinearSVC pris comme référence

Une validation croisée sur 3 échantillons conduit à un score de :

- 0.6021 ± 0.0005 sur l’échantillon d’apprentissage,

- et de 0.5988 ± 0.0027 sur l’échantillon de test.

Le modèle est donc capable de généraliser sans problèmes de sur- ou de sous-apprentissage notables.

### Optimisation du modèle - Ajustement des hyperparamètres penalty, loss, C et multi\_class avec GridSearchCV

Les influences suivantes ont été regardées, à savoir celles de :

- la pénalité (norme L1 ou L2),

- de la fonction de perte (‘hinge’ ou ‘squared\_hinge’),

- du paramètre de régularisation C : valeurs étudiées : [0.01, 1, 10],

- du paramètre multi\_class (‘ovr’ ou ‘crammer\_singer’).

A l’issue de l’optimisation, les meilleurs paramètres parmi ceux testés sont les suivants :

|  |  |  |  |
| --- | --- | --- | --- |
| penalty | loss | C | Multi\_class |
| L2 | squared\_hinge | 1 | ovr |

Les résultats de ce modèle sur l’échantillon de test sont donnés dans la Figure 22, en termes de métriques et de matrice de confusion. Ils sont relativement comparables à ceux obtenus par la régression logistique. Le recours à l’approximation Nyström, telle qu’implémentée ici, n’a pas permis d’améliorer les performances du modèle de classification.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (a) | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  | **Classes prédites** | | | | |  |  | 1 | 2 | 3 | 4 | | **Classes réelles** | 1 | 37054 | 1638 | 1715 | 5730 | | 2 | 325 | 1198 | 1026 | 501 | | 3 | 2176 | 2744 | 6759 | 5821 | | 4 | 12971 | 2405 | 5491 | 24230 |   (b) |

Figure 22 : Métriques (a) et matrice de confusion (b) du modèle LinearSVC précédé d’une approximation de Nyström

## Decision Tree Classifier - Random Forest Classifier - Balanced Random Forest

Nous choisissons de présenter ces 3 modèles ensemble car ils ont été traités de la même manière.

### Modèle de référence

#### Description du modèle

Pour modéliser, nous reprenons le jeu de données sans dummies que nous avons créé lors du nettoyage des données. Nous supprimons les variables ‘an’, ‘jour’, ‘grav-rec’, ‘date’ et ‘dep’ que nous avions conservées pour les visualisations et les séries temporelles.

Les arbres de décision et les random forest ne nécessitent pas de traitement particulier :

* les heures, mois, l’âge, les latitudes et longitudes sont conservées sans normalisation,
* les variables catégorielles sont conservées sans faire de dummies.

#### Paramètres

Pour chacun des algorithmes, nous réalisons une première modélisation avec les paramètres natifs de chaque algorithme afin de calculer l’importance de chacune des 35 variables. Puis, nous entraînons à nouveau le modèle natif en ajoutant une par une les variables (dans l’ordre d’importance décroissant) dans le but de ne sélectionner que les variables qui maximisent l’accuracy. Ceci nous permet d’éliminer les variables suivantes (Figure 23) selon l'algorithme :

|  |  |  |
| --- | --- | --- |
| **Decision Tree** | **Random Forest** | **Balanced Random Forest** |
| Conservation de 31 variables | Conservation de 34 variables | Conservation de 29 variables |
| Une image contenant texte, menu, capture d’écran, Police  Description générée automatiquement | Une image contenant texte, menu, capture d’écran, conception  Description générée automatiquement | Une image contenant texte, menu, capture d’écran, Police  Description générée automatiquement |

Figure 23 : Variables sélectionnées selon le modèle avec leur importance

En utilisant ce nombre de variables réduit, nous obtenons les rapports de classification et les matrices de confusions suivant (Figure 24) :

|  |  |
| --- | --- |
| **Decision Tree**  **(Train accuracy = 99.9% - Test accuracy = 57.02%)** | |
| Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement | Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement |
|  | |
| **Random Forest**  **(Train accuracy = 99.9% - Test accuracy = 66.96%)** | |
| Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement |  |
|  | |
| **Balanced Random Forest**  **(Train accuracy = 66.69% - Test accuracy = 59.04%)** | |
| Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement |  |

Figure 24 : Métriques et matrice de confusion selon le modèle

Dans tous les cas, on remarque la difficulté des modèles à gérer la classe 2 (celles des personnes tuées) :

* Decision Tree a une précision et un recall équilibré mais très faible,
* Random Forest a un f1-score plus faible (précision beaucoup plus élevée mais au détriment du recall),
* Balanced Random Forest a un f1-score augmenté (recall augmenté au détriment de la précision).

De plus, Decision Tree et Random Forest présentent un fort sur apprentissage.

### Optimisation du modèle

Dans un premier temps, pour chacun des algorithmes, nous allons rechercher avec un grid search les paramètres qui maximisent l’accuracy. Pour cela chaque modèle est défini de la manière suivante (Tableau 5) :

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Decision Tree** | |  | **Random Forest** | |  | **Balanced random Forest** | |
| criterion | [’gini’, ‘entropy’] | criterion | [’gini’, ‘entropy’] | criterion | [’gini’, ‘entropy’] |
| max\_depth | [‘10, 20, None] | max\_depth | [‘10, 20, None] | max\_depth | [‘10, 20, None] |
| max\_features | [‘sqrt’] | min\_samples\_leaf | [1, 10, 20] | min\_samples\_leaf | [1, 10, 20] |
| min\_samples\_leaf | [1, 10, 20] | min\_samples\_split | [2, 10, 20] | min\_samples\_split | [2, 10, 20] |
| min\_samples\_split | [2, 10, 20] | random\_state | [42] | random\_state | [42] |
| random\_state | [42] | n\_estimators | [100] | n\_estimators | [100] |
| splitter | [‘best’, ‘random’] | bootstrap | [False, True] | bootstrap | [True] |
|  | | n\_jobs | [-1] | n\_jobs | [-1] |
|  | | sampling\_strategy | [‘all’] |
| remplacement | [True] |

Tableau 5  : Choix d’optimisation des paramètres selon le modèle

Puis nous avons entraîné chacun de ces modèles avec :

* soit ajout d’un paramètre supplémentaire : weight\_class défini à None, ‘balanced’ ou un dictionnaire de poids que nous avons optimisé,
* soit sur un jeu de données où l’on fait de l’oversampling avec SMOTENC ou RandomOverSampler (non réalisé pour Balanced Random Forest),
* soit sur un jeu de données où l’on fait de l’undersampling avec RandomUnderSampler ou ClusterCentroids (non réalisé pour Balanced Random Forest).

Enfin, dans chaque cas, nous recherchons le meilleur max\_depth en réalisant un grid search avec un scoring basé sur le f1-score-macro.

### Résultats

Pour chacun de ces algorithmes, nous choisissons le meilleur résultat en nous basant sur les modèles qui permettent d’optimiser le macro avg. Voici les rapports de classification et les matrices de confusions (Figure 25) :

|  |  |
| --- | --- |
| **Decision Tree**  **(Train accuracy = 65.54% - Test accuracy = 62.78%)** | |
| Paramètres :  {'class\_weight': {1: 1, 2: 4, 3: 1, 4: 1}, 'criterion': 'gini', 'max\_depth': 12, 'max\_features': None, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2, 'random\_state': 42, 'splitter': 'best'} | |
| Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement | Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement |
|  | |
| **Random Forest**  **(Train accuracy = 81.09% - Test accuracy = 65.22%)** | |
| Paramètres :  {'bootstrap': True, 'class\_weight': 'balanced', 'criterion': 'entropy', 'max\_depth': 18, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2, 'n\_estimators': 100, 'n\_jobs': -1, 'random\_state': 42} | |
| Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement | Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement |
|  | |
| **Balanced Random Forest**  **(Train accuracy = 68.38% - Test accuracy = 60.71%)** | |
| Paramètres :  {'bootstrap': True, 'class\_weight': None, 'criterion': 'entropy', 'max\_depth': 30, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2, 'n\_estimators': 100, 'n\_jobs': -1, 'random\_state': 42, 'replacement': True, 'sampling\_strategy': 'all'} | |
| Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement | Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement |

Figure 25 : Métriques et matrice de confusion avec les paramètres optimisés selon le modèle

Finalement, nous choisissons l’algorithme qui donne les meilleurs résultats parmi ces 3 modèles : le modèle utilisant l'algorithme Random Forest (Figure 26). Cependant, il y a un fort sur-apprentissage. Nous décidons donc de diminuer le max\_depth ce qui nous permet d’obtenir le résultat suivant :

|  |  |
| --- | --- |
| **Random Forest**  **(Train accuracy = 65.52% - Test accuracy = 61.22%)** | |
| Paramètres :  {'bootstrap': True, 'class\_weight': 'balanced', 'criterion': 'entropy', 'max\_depth': 13, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2, 'n\_estimators': 100, 'n\_jobs': -1, 'random\_state': 42} | |
|  |  |

Figure 26 : Métriques et matrice de confusion pour le meilleur modèle

En diminuant le sur apprentissage, on peut observer que la prédiction pour les indemnes et les tués (classe 1 et 2) est améliorée au détriment des blessés hospitalisés et légers (classes 3 et 4).

### Interprétabilité des résultats

#### Analyse de l’importance donnée par le meilleur modèle

Le meilleur modèle utilisant l’algorithme Random forest est entraîné avec 34 variables dont voici le graphique selon l’importance donnée par le modèle (Figure 27) :

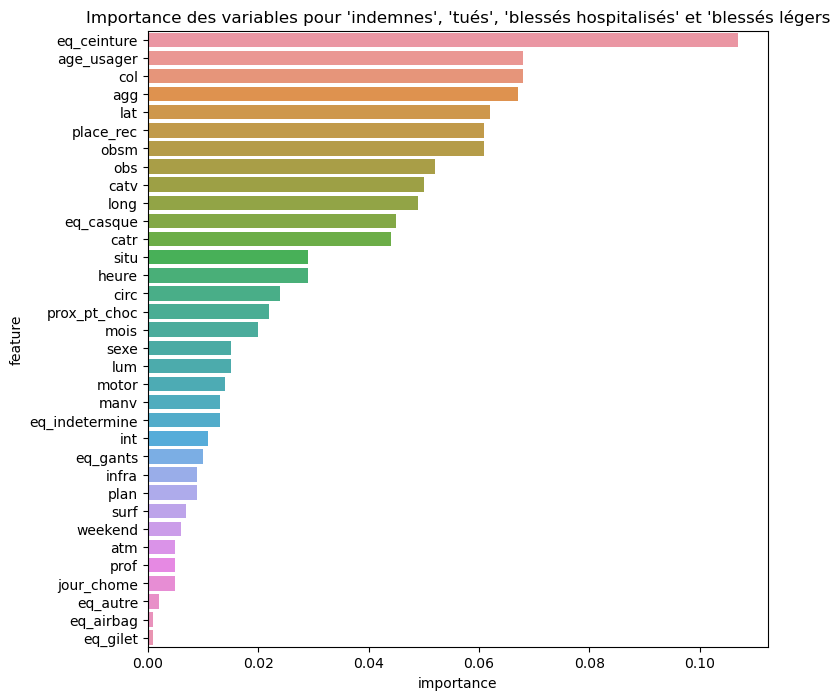


Figure 27 : Importance des variables pour le meilleur modèle

Le modèle accorde une grande importance aux variables suivantes :

* très forte importance de l’utilisation ou non de la ceinture de sécurité,
* forte importance de l’âge de l’usager, du type de collision et du fait de rouler en agglomération ou non,
* importance légèrement moindre pour la latitude, la place dans le véhicule ou piéton et l’obstacle mobile heurté.

#### Analyse de l’interprétation du meilleur modèle avec SHAP

L’analyse de l’interprétation avec SHAP (Shapley additive explanations) permet d’attribuer à chaque variable une valeur d’importance pour une prédiction particulière et donc d’affiner la compréhension de la part de chaque variable dans la prédiction. Ceci nous permet d’obtenir les graphiques d’importance des variables (Figure 28), mais également les graphiques de densité des valeurs SHAP pour chaque variable (Figure 29) en fonction de la modalité de la variable cible ‘grav’.

|  |  |  |  |
| --- | --- | --- | --- |
| **Indemnes (classe 1)** | **Tués (classe 2)** | **Blessés hospitalisés (classe 3)** | **Blessés légers (classe 4)** |
|  |  |  |  |

Figure 28  : Graphiques d’importance des variables selon SHAP

|  |  |  |  |
| --- | --- | --- | --- |
| **Indemnes (classe 1)** | **Tués (classe 2)** | **Blessés hospitalisés (classe 3)** | **Blessés légers (classe 4)** |
|  |  |  |  |

Figure 29 : Graphiques de densité des valeurs de SHAP

L’analyse de ces graphiques, nous permet de prévoir les variables qui influent sur le fait d’être classés indemne, tué, blessé hospitalisé ou blessé léger. Ainsi, en analysant les 5 premières variables les plus importantes selon SHAP, nous pouvons dire que les paramètres qui influent positivement sur la classification sont :

* **pour les indemnes** :
  + l’utilisation de la ceinture de sécurité,
  + rouler en agglomération,
  + être à l’avant du véhicule (les valeurs les plus faibles de la variable catégorielle ‘place\_rec’ sont soit conducteur et/ou passager avant),
  + ne pas porter de casque (certainement pour le fait de ne pas être en vélo/trottinette ou moto),
  + ne pas heurter d’obstacle fixe.
* **pour les tués** :
  + rouler hors d’agglomération,
  + ne pas utiliser de ceinture de sécurité,
  + heurter un obstacle fixe.

Pour les variables ‘catr’ (variable catégorielle regroupant 8 modalités) et ‘col’ (variable catégorielle regroupant 7 modalités) la distinction entre les valeurs sur le graphique de densité des valeurs est plus floue et ne permet pas d’établir formellement quelles modalités influent positivement sur la classification.

* **pour les blessés hospitalisés** :
  + ne pas utiliser de ceinture de sécurité,
  + porter un casque (certainement pour le fait d’être en vélo/trottinette ou moto),
  + rouler hors agglomération.

Pour les variables ‘catv’ (variable catégorielle regroupant 6 modalités) et ‘lat’’ (variable numérique) la distinction entre les valeurs sur le graphique de densité des valeurs est plus floue et ne permet pas d’établir formellement quelles modalités ou valeurs influent positivement sur la classification.

* **pour les blessés légers** :
  + rouler en agglomération,
  + être à l’arrière du véhicule ou piéton (les valeurs les plus élevées de la variable catégorielle ‘place\_rec’ sont soit passager arrière, soit piéton),
  + rouler sur une route unidirectionnelle,
  + être une femme.

Pour la variable ‘lat’’ (variable numérique) la distinction entre les valeurs sur le graphique de densité des valeurs est plus floue et ne permet pas d’établir formellement quelles valeurs influent positivement sur la classification.

## CatBoost Classifier

### Modèle de référence

#### Description du modèle

CatBoost est l’une des librairies permettant d’entraîner des arbres de décision pour la classification. Il prend en charge des données catégorielles (Cat) et utilise le Gradient Boosting (Boost).

#### Paramètres

Ce modèle s’appuie sur une centaine de paramètres pouvant être regroupés en différentes familles (le lecteur trouvera la liste complète sur la documentation à l’adresse suivante :<https://catboost.ai/en/docs/concepts/python-reference_catboostclassifier>) :

- Ceux classiques pour un problème de machine learning : le taux d’apprentissage (learning\_rate), la fonction de perte (loss\_function), le nombre d’itérations (iterations)…

- Ceux contrôlant l’échantillonnage des données pour chaque arbre : choix aléatoire (bagging temperature)

- Ceux contrôlant la structure de l’arbre : sa symétrie (grow\_policy), sa profondeur (depth) …

- Ceux contrôlant la sélection des variables pour la construction des arbres (colsample\_bylevel)

- Ceux régissant la pénalisation du modèle : (l2\_leaf\_reg)

- Ceux permettant de détecter le surapprentissage.

### Optimisation du modèle

L’optimisation des hyperparamètres a été effectuée avec la librairie Optuna. L’espace d’hyperparamètres à tester a été le suivant :

- un nombre d’itérations compris entre 250 et 300,

- une fonction de perte parmi : « MultiClass » et « MultiClassOneVsAll »,

- un taux d’apprentissage compris entre 0.01 et 0.5,

- une profondeur de l’arbre comprise entre 4 et 10,

- un paramètre de pénalisation entre 5 et 10,

- une structure de l’arbre choisie parmi : «SymmetricTree » et « Lossguide »,

- une fraction de variables à choisir dans le processus de division comprise entre 0.05 et 1.

La métrique de performance choisie est le F1-score ‘MICRO’, avec pour ambition sa maximisation. L’optimisation bayésienne a été retenue pour la sélection des hyperparamètres dans l’espace.

La Figure 30 présente l’influence des hyperparamètres sur la performance du modèle pour une sélection de 20 jeux de données différents. Ce graphique met en lumière qu’il est plutôt bénéfique de choisir **des profondeurs importantes** et des **facteurs de pénalisation supérieurs à 7**. De façon générale, pour les méthodes impliquant des arbres, il apparaît que certes, l’augmentation des profondeurs améliore les performances sur l’échantillon test, mais cela accentue également le phénomène de sur-apprentissage.

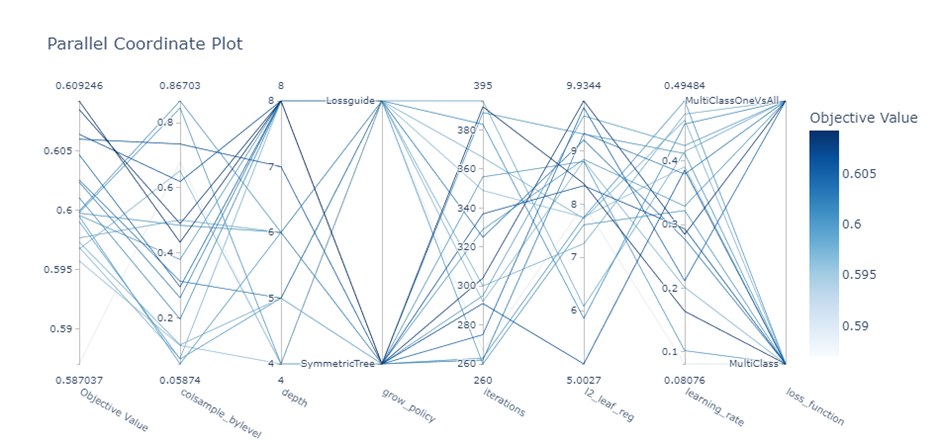


Figure 30 : Graphique à coordonnées parallèles présentant le lien entre les hyperparamètres de Catboost et la performance du modèle.

Les performances du meilleur modèle parmi ceux testés dans le processus d’optimisation sont données à la Figure 31. Encore une fois**, le potentiel gain de performance par optimisation du modèle reste limité**. Comparativement au meilleur modèle retenu jusqu’à présent, à savoir le Random Forest dont les résultats sont présentés à la Figure 26, le modèle Catboost permet une **amélioration des rappels des classes 2, celle des tués et 4, celle des blessés légers, mais au détriment de la précision sur ces classes et in fine, des f1-scores.** En conséquence, si l’on se fixe comme objectif de vouloir maximiser l’identification des tués parmi les accidentés de la route, alors le modèle CatBoost est un bon candidat. Néanmoins, à l’échelle de l’ensemble des classes, le modèle Random Forest apporte le meilleur compromis en termes de performance.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (a) | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  | **Classes prédites** | | | | |  |  | **1** | **2** | **3** | **4** | | **Classes réelles** | **1** | 36223 | 1581 | 2576 | 5757 | | **2** | 163 | 1743 | 914 | 230 | | **3** | 1448 | 4463 | 8518 | 3071 | | **4** | 10738 | 3202 | 8895 | 22262 |   (b) |

Figure 31 : Métriques (a) et matrice de confusion (b) du modèle CatBoost

## XGBoost Classifier

### Modèle de référence

#### Description du modèle

Pour cette partie de modélisation de référence qui permettra d’avoir d’une part une vision de la performance du modèle XGBoost sans hyper-paramètrage (du moins ceux par défaut) et d’autre part une base de comparaison avec les différentes modélisations optimisées, nous avons travaillé avec le jeu de données “data\_cleaned\_final\_sans\_dummies.csv” créé lors du preprocessing.

Nous avons supprimé les variables ‘jour’, ‘grav-rec’, ‘date’ et ‘prox\_pt\_choc’, qui représentent des variables sans signification ou redondantes.

Même si le modèle XGBoost ne souffre pas de l’obligation d’avoir des données normalisées, par cohérence nous avons appliqué une normalisation sur les variables ‘age-usager’, ‘latitude’, ‘longitude’ et ‘heure’.

Le dataset présentant un fort déséquilibre des classes pour la variable cible ‘grav’, nous avons décidé de tester un modèle **xgb\_clf** selon 3 méthodes pour déterminer une base de référence :

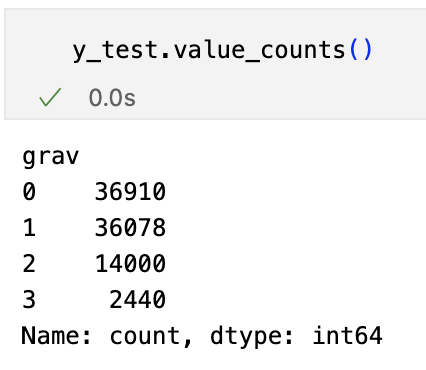
1. Méthode 1 : Entraînement sur le dataset,
2. Méthode 2 : Application d’une *class\_weight* lors du .fit() sur le dataset,
3. Méthode 3 : Application d’un rééchantillonnage sur le dataset.

#### Paramètres par défaut du modèle

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| n\_estimator | **100** | subsample | **1** | reg\_lambda | **1** |
| learning\_rate | **0.3** | sampling\_method | uniform | reg\_alpha | **0** |
| gamma | **0** | colsample\_bytree | 1 | tree\_method | auto |
| max\_depth | **6** | colsample\_bylevel | 1 | scale\_pos\_weight | 1 |
| min\_child\_weight | **1** | colsample\_bynode | 1 | refresh\_leaf | 1 |
| process\_type | default | max\_leaves | 0 | num\_parallel\_tree | 1 |
| grow\_policy | depthwise | max\_bins | 256 | multi\_strategy | one\_output\_per\_tree |
| objective | softmax |  |  |  |  |

Tableau 6 : Paramètres par défaut de XGBoost

Quelques informations sur la répartition des valeurs de y\_test :



#### Courbes d’apprentissage

Application d’un early\_stopping\_rounds = 10 et eval\_metric = [‘merror’, ‘mlogloss’] et d’un seed = 42 avec évaluation eval\_set=[(X\_train, y\_train), (X\_test, y\_test)].

Visualisation des courbes mlogloss et merror suivant le nombre d’arbres (n\_estimators) (Figure 32) :

|  |  |
| --- | --- |
| Méthode 1 | Apparition d’un overfitting quasiment dès que n\_estimators > 10 |
| Méthode 2  application  class\_weight | Décroissance moins rapide de la mlogloss vs méthode 1 et inversement de la merror mais valeurs restent supérieures. |
| Méthode 3  application random over sampling sur le dataset | Idem, courbes similaires à la méthode 2 |

Figure 32 : Courbes de mlogloss et merror suivant le nombre d’arbres

#### Evaluations

* Matrice de confusion

|  |  |  |
| --- | --- | --- |
| Méthode 1 | Méthode 2 | Méthode 3 |
|  |  |  |
| Classe 3 très peu détectée et confusion avec les autres classes (Faux Négatifs). | Beaucoup de Faux Positifs mais plutôt fiable dans la prédiction de la classe 3. | Résultat quasi-similaire à la méthode 2. |

Figure 33 : Matrices de confusion pour chaque méthode

* Key Metrics

|  |  |  |
| --- | --- | --- |
| Méthode 1 | Méthode 2 | Méthode 3 |
|  |  |  |

Tableau 7 : Métriques principales pour chaque méthode

La méthode 2 permet d’avoir la **meilleure “balanced accuracy”** et aussi le **meilleur Macro Recall**, c’est à dire la meilleure proportion de cas correctement détectés mais ceci au détriment d’une précision plus faible et donc d’un taux de FAUX POSITIF plus élevé.

Dans notre cas, il conviendra de **définir le coût entre les FAUX NÉGATIFS et les FAUX POSITIFS**. C’est à dire faut-il mieux prédire la gravité quitte à accentuer cette dernière ou limiter les prédictions graves quitte à en oublier ?

* Rapport de classification

|  |  |
| --- | --- |
| Méthode 1 | Le f1\_score de la classe 3 (Tués) est médiocre, impact du recall très faible |
| Méthode 2 | Légère amélioration du f1\_score de la classe 3 mais au détriment des autres classes. |
| Méthode 3 | Les résultats sont identiques à la méthode 2. |

Figure 34 : Rapports de classification selon la méthode

Malgré une accuracy de 67% sur l’ensemble de test, la méthode 1 semble souffrir du fort déséquilibre de la variable cible du dataset. Les méthodes 2 et 3 avec une accuracy de 61% gèrent mieux ce déséquilibre.

Pour rappel : classe 0 = 41.3%, classe 1 = 40,3%, classe 2 = 15,7%, classe 3 = 2,7%.

* Tracés des features importance (20 premières) selon weight, gain et cover

|  |  |  |
| --- | --- | --- |
| Méthode 1 | Méthode 2 | Méthode 3 |
|  |  |  |
|  |  |  |
|  |  |  |

Figure 35 : Les 20 premières features importance selon weight, gain, cover et selon la méthode

Selon le *weight*, nous avons pour les 3 modèles les 6 features identiques dans le même ordre qui apparaissent dans la construction des arbres. Concernant le *gain* (réduction de la fonction de perte lors de l’utilisation d’une feature pour séparer une branche), on remarque l’importance de la feature ‘eq\_ceinture’ qui est significativement plus élevée sur les 3 méthodes.

Pour terminer concernant le *cover* (utilisation de la feature pour la séparation des données), on remarque une rupture entre les 3 modèles.

En conclusion de cette première partie concernant l’évaluation du modèle de référence, nous retenons pour la suite, c’est-à-dire l’optimisation des hyperparamètres pour améliorer la performance, **la méthode 2 (ajout d’une class\_weight)** qui semble être la plus pertinente par rapport à l’optimisation des métriques recherchées (precision et recall).

### Optimisation de la méthode 2

Le modèle XGBoost étant un modèle comprenant beaucoup d’hyperparamètres, nous avons déployé une méthode agile utilisée par la communauté de Data Scientist pour économiser les temps de calcul, à savoir :

1. Fixer le learning\_rate à 1,
2. Fixer le n\_estimators puis optimiser les autres hyperparamètres,
3. Ajuster le learning\_rate.

#### Paramètres optimisés

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| n\_estimator | **200** | subsample | 1 | reg\_lambda | 1 |
| learning\_rate | **0.05** | sampling\_method | uniform | reg\_alpha | 0 |
| gamma | 0 | colsample\_bytree | 1 | tree\_method | auto |
| max\_depth | **3** | colsample\_bylevel | 1 | scale\_pos\_weight | 1 |
| min\_child\_weight | 1 | colsample\_bynode | 1 | refresh\_leaf | 1 |
| process\_type | default | max\_leaves | 0 | num\_parallel\_tree | 1 |
| grow\_policy | depthwise | max\_bins | 256 | multi\_strategy | one\_output\_per\_tree |
| objective | softmax |  |  |  |  |

Tableau 8 : Paramètres optimisés pour XGBoost

#### Courbe d’apprentissage

|  |  |
| --- | --- |
|  |  |

Figure 36 : Courbes de mlogloss et merror pour le modèle optimisé

On peut constater la convergence entre les courbes de train et test sur les 2 métriques mlogloss et merror.

#### Evaluation et comparaison

* Matrice de confusion

|  |  |
| --- | --- |
| référence | model2f (modèle optimisé) |
|  | On constate que la **classe 3 est bien plus détectée** au détriment d’un taux de faux positif plus élevé |

Figure 37 : Matrices de confusion du modèle de référence et du modèle optimisé

* Rapport de classification

|  |  |
| --- | --- |
| référence |  |
| model2f | L’optimisation du recall de la classe Tué impacte fortement le recall des classes Blessé Léger et Blessé Grave et une perte de la précision sur les 4 classes donc un taux de FAUX POSITIFS important. |

Figure 38 : Rapports de classification pour le modèle de référence et le modèle optimisé

### Interprétabilité des résultats avec SHAP

Les résultats de l’optimisation n’étant pas concluants dans le sens où l’optimisation n’a pas permis de faire un gain significatif, nous avons mené une étude d’interprétabilité pour comprendre comment le modèle prédit ses choix, quelles sont les features déterminantes. L’utilisation de SHAP nous permet de comprendre le fonctionnement du modèle en étudiant les graphiques d’importance des variables (Figure 39) et les graphiques de densité des valeurs SHAP pour chaque variable (Figure 40) en fonction de la variable cible.

La feature *eq\_ceinture* est prépondérante sur les 4 classes de gravité notamment pour les indemnes donc l’utilisation de la ceinture joue un rôle important.

On remarque aussi que pour les classes Blessé Grave et Tué, les 3 features les plus importantes sont les mêmes, ce qui signifie une difficulté à séparer ces 2 classes avec certitude.

La feature *agg* (rouler en agglomération ou pas), induisant de façon sous-jacente une notion de vitesse, influe nettement sur la gravité.

|  |  |
| --- | --- |
| Indemne (classe 0) | Blessé Léger (classe 1) |
| Blessé Grave (classe 2) | Tué (classe 3) |

Figure 39 : Graphiques d’importance des variables selon SHAP

### 

|  |  |  |  |
| --- | --- | --- | --- |
| Indemne (classe 0) | Blessé Léger (classe 1) | Blessé Grave (classe 2) | Tué (classe 3) |

Figure 40 : Graphiques de densité des valeurs de SHAP

## K-nearest neighbors (KNN)

### Modèle de référence

#### Description du modèle

L'algorithme des K plus proches voisins (K-nearest neighbors, KNN) est un algorithme d'apprentissage **supervisé non paramétrique**, qui utilise la proximité des observations entre elles pour effectuer des classifications ou des prédictions. L’idée sous-jacente est que des points similaires peuvent être trouvés les uns à côté des autres : “*Dis-moi qui sont tes voisins, je te dirai qui tu es*”.

Pour les problèmes de classification, un vote aura lieu : le libellé de classe le plus fréquemment représenté parmi les K plus proches voisins du point à classifier lui sera attribué. Les K plus proches voisins peuvent avoir un poids égal dans le vote, ou un poids inversement proportionnel à leur distance du point à prédire.

A noter :

1. Les effectifs des classes étant déséquilibrés, nous avons travaillé sur un **sous-échantillon équilibré** de notre jeu de données, obtenu avec la fonction RandomUnderSampler(). L’échantillon ainsi obtenu contenait **9761 observations pour chaque classe de gravité** (indemnes, tués, blessés, blessés hospitalisés).
2. **Le calcul de la distance dans le modèle KNN nécessite que toutes les variables soient sur la même échelle de valeurs**, pour ne pas donner plus de poids a priori à certaines d’entre elles du simple fait de leur plage de valeur élevée. C’est pourquoi nous avons normalisé nos variables explicatives comme décrit plus haut dans la section sur la régression logistique.

#### Paramètres par défaut du modèle

Nous avons créé une première instance du modèle KNN avec les paramètres par défaut, c'est-à-dire notamment 5 voisins et la mesure de distance de Minkowski de puissance p=1 (équivalent à la distance de Manhattan).

Le tableau des performances de ce modèle obtenu avec classification\_report\_imbalanced est présenté ci-dessous :

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Classe** | **pre** | **rec** | **spe** | **f1** | **geo** | **iba** | **sup** |
| indemnes | 0.69 | 0.73 | 0.77 | 0.71 | 0.75 | 0.56 | 36910 |
| blessés | 0.61 | 0.43 | 0.81 | 0.51 | 0.59 | 0.34 | 36078 |
| blessés hospitalisés | 0.34 | 0.37 | 0.87 | 0.35 | 0.56 | 0.30 | 14000 |
| tués | 0.12 | 0.46 | 0.91 | 0.19 | 0.64 | 0.39 | 2440 |
| avg / total | 0.59 | 0.55 | 0.80 | 0.56 | 0.65 | 0.43 | 89428 |

Tableau 9 : Rapport de classification

On retient qu’avec cette paramétrisation “par défaut”, le f1-score est de 0.56, et que la prédiction de la classe « tués » est celle qui a la plus mauvaise précision (prec=0.12).

L'affichage de la matrice de confusion, ci-dessous, montre en effet que la majorité des personnes classées comme tuées appartiennent en réalité aux autres classes, tout simplement du fait de la prédominance de ces autres classes dans l’échantillon de test. Ainsi, un taux de mauvaise classification même faible sur ces autres classes va ajouter des faux positifs dans la catégorie « tués », en nombre largement plus important que le nombre de vrais positifs de cette classe.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Classe réelle / prédite** | **indemnes** | **blessés légers** | **blessés hospitalisés** | **tués** |
| **indemnes** | **26983** | 5905 | 2318 | 1704 |
| **blessés légers** | 10361 | **15677** | 7031 | 3009 |
| **blessés hospitalisés** | 1626 | 3781 | **5131** | 3462 |
| **tués** | 209 | 379 | 740 | **1112** |

Tableau 10 : Matrice de confusion de l’algorithme KNN (paramètres par défaut)

En normalisant la matrice de confusion par ligne, on voit mieux que les difficultés de classification ont lieu pour les classes du milieu : blessés légers et blessés hospitalisés (rappel < 0.5), que l'algorithme a du mal à discriminer des classes de gravité adjacentes. Ainsi, il classe plus souvent les blessés légers en indemnes ou en blessés hospitalisés que dans la bonne catégorie ([0.29 +0.19] > 0.43) ; et les blessés hospitalisés en blessés légers ou tués ([0.27 + 0.25] > 0.37). Par contre, l’algorithme KNN classe peu les blessés légers en tués (8%) ou les blessés hospitalisés en indemnes (12%). Sa difficulté est donc vraiment de **bien prédire la finesse du continuum de gravité**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Classe réelle / prédite** | **indemnes** | **blessés légers** | **blessés hospitalisés** | **tués** |
| **indemnes** | **0.73** | 0.16 | 0.06 | 0.05 |
| **blessés légers** | 0.29 | **0.43** | 0.19 | 0.08 |
| **blessés hospitalisés** | 0.12 | 0.27 | **0.37** | 0.25 |
| **tués** | 0.09 | 0.16 | 0.30 | **0.46** |

Tableau 11 : Matrice de confusion normalisée par ligne

### Recherche de paramètres optimaux avec GridSearchCV

Nous avons ensuite cherché à optimiser notre algorithme de KNN en explorant avec GridSearchCV la grille construite sur le croisement suivant de paramètres :

* le paramètre de poids “weights”, a été fixé à “uniform” ou “distance”. Cette dernière valeur permet de pondérer la contribution des K plus proches voisins dans le vote en fonction de leur proximité au point que l’on cherche à classifier,
* le nombre de voisins a été étudié sur l’intervalle 5-30, par pas de 5,
* la puissance de la mesure de Minkowski a été fixée à 1 ou 2.

params = {'n\_neighbors':[i for i in range(5,30,5)],

'weights' : ['uniform','distance'],

"p" : [i for i in range(1, 2)]}

Nous avons effectué une validation croisée à 5 volets stratifiés.

Le meilleur modèle était obtenu avec les paramètres suivants : {'n\_neighbors': 25, 'p': 1, 'weights': 'distance'}. Voici ses performances :

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Classe** | **pre** | **rec** | **spe** | **f1** | **geo** | **iba** | **sup** |
| indemnes | 0.70 | 0.78 | 0.77 | 0.74 | 0.77 | 0.60 | 36910 |
| blessés | 0.68 | 0.42 | 0.86 | 0.52 | 0.60 | 0.35 | 36078 |
| blessés hospitalisés | 0.36 | 0.36 | 0.88 | 0.36 | 0.57 | 0.30 | 14000 |
| tués | 0.12 | 0.58 | 0.88 | 0.20 | 0.72 | 0.50 | 2440 |
| avg / total | 0.62 | 0.57 | 0.83 | **0.58** | 0.67 | 0.45 | 89428 |

Tableau 12 : Rapport de classification pour le modèle optimisé

On note une amélioration de 0.02 points du f1-score (0.58 vs 0.56), avec une augmentation de 0.01 point (resp. 0.03 points) et du taux de rappel de la classe des tués : 0.58 vs 0.46, par rapport au modèle avec le paramétrage par défaut.

Malgré cette légère amélioration des performances, le problème reste toujours de distinguer les blessés légers et blessés hospitalisés des classes adjacentes, avec des rappels quasi-inchangés pour ces deux classes et une majorité des mauvais classements dans les classes de gravité adjacentes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Classe réelle / prédite** | **indemnes** | **blessés** | **blessés hospitalisés** | **tués** |
| indemnes | 0.78 | 0.11 | 0.05 | 0.06 |
| blessés | 0.29 | 0.42 | 0.18 | 0.10 |
| blessés hospitalisés | 0.10 | 0.22 | 0.36 | 0.32 |
| tués | 0.07 | 0.09 | 0.25 | 0.58 |

Tableau 13 : Matrice de confusion normalisée par ligne pour le modèle optimisé

### Sélection des features

L’algorithme du KNN “n’apprend pas” au fil des itérations à ignorer une feature qui n’apporte pas d’information par rapport à la variable cible, contrairement aux méthodes paramétriques comme les réseaux de neurones et la régression logistique. Dans le cas extrême, la présence de variables n’apportant que du bruit pourrait dégrader les performances de l’algorithme.

Nous avons décidé de ne conserver que les 20 variables les plus importantes (sur 34) telles que données par le meilleur modèle de Random Forest (voir plus haut) : eq\_ceinture, age\_usager, col, agg, lat, place\_rec, obsm, obs, catv, long, eq\_casque, catr, situ, heure, circ, prox\_pt\_choc, mois, sexe, lum, motor.

Le modèle KNN est toujours le même : 25 voisins et distance de Minkowski de puissance 1.

Les performances sont les suivantes, c’est-à-dire quasiment les mêmes que lorsqu’on utilise les 34 variables :

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Classe** | **pre** | **rec** | **spe** | **f1** | **geo** | **iba** | **sup** |
| indemnes | 0.71 | 0.76 | 0.78 | 0.74 | 0.77 | 0.60 | 36910 |
| blessés | 0.67 | 0.44 | 0.85 | 0.53 | 0.61 | 0.36 | 36078 |
| blessés hospitalisés | 0.36 | 0.36 | 0.88 | 0.36 | 0.56 | 0.30 | 14000 |
| tués | 0.12 | 0.59 | 0.88 | 0.20 | 0.72 | 0.50 | 2440 |
| avg / total | 0.62 | 0.56 | 0.83 | 0.58 | 0.67 | 0.45 | 89428 |

Tableau 14 : Rapport de classification avec sélection de variables

Ce modèle mettant 32.2 secondes à tourner au lieu de 53.7 secondes pour le modèle précédent, on a tout intérêt à supprimer les variables peu informatives avant de lancer le KNN.

Certains chercheurs (notamment [Bhardwaj et al](https://arxiv.org/pdf/1811.05062)) proposent d’aller plus loin en appliquant en préalable du KNN une procédure de scaling qui tient compte de l’importance des variables telle que déterminée dans une random forest. Ainsi, la distance entre deux points sera “tirée” par les variables les plus liées à la réponse, et l’on pourrait ainsi améliorer les performances de classification du KNN, peut-être notamment sur sa capacité à discriminer les blessés légers des indemnes et des blessés hospitalisés, et les blessés hospitalisés des blessés légers et des tués. Faute de temps, cette option n’a pas pu être explorée.

## Modélisation par Deep Learning

### Preprocessing

Nous appliquons les étapes de preprocessing décidées à l’issue du rapport 1 :

o Les classes de la variable cible ‘grav’ sont renumérotés de 0 à 3 (0 - indemnes, 1 - Tués, 2 - Blessés hospitalisés, 3 - Blessés légers)

o Transformation des heures et des mois

o Transformation des latitudes et longitudes (RobustScaler)

o Transformation de l’âge (StandardScaler)

### Modèle de référence

Le modèle choisi est le suivant :

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

Description générée automatiquement

Chaque couche Dense est activée par la fonction d’activation ReLu sauf la dernière couche qui est activée par la fonction Softmax. Les couches de Dropout ont un rate de 0,2 afin de diminuer le surapprentissage.

Le modèle est compilé avec :

* loss : sparse\_categorical\_cross\_entropy
* optimizer: adam
* metrics : sparse\_categorical\_accuracy

Enfin le modèle est entraîné avec :

* epochs = 100
* batch\_size = 512 (nous choisissons un nombre élevé afin que le modèle ait plus de chance de rencontrer la modalité des tués)
* validation\_split = 0,1
* callbacks = [reduce\_learning\_rate]

Le reduce\_learning\_rate a préalablement été défini de la manière suivante :

reduce\_learning\_rate = ReduceLROnPlateau(monitor = 'val\_loss',

min\_delta = 0.01,

patience = 5,

factor = 0.5,

cooldown = 2,

verbose = 1)

Ce modèle permet d’obtenir les résultats suivants (Figure 41):

|  |  |
| --- | --- |
| Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement  (a) | Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement  (b) |
| Une image contenant ligne, Tracé, diagramme, capture d’écran  Description générée automatiquement  (c) | |

Figure 41 : Rapport de classification (a) – Matrice de confusion (b) – Courbes d’évolution de la loss et de l’accuracy par epoch

On remarque que le déséquilibre du jeu de données influe sur les résultats. En effet, les classes minoritaires sont aussi les moins bien prédites. Nous décidons donc d’équilibrer les classes afin d’améliorer les prédictions.

### Rééquilibrage du jeu de données

Le jeu de donnée étant fortement déséquilibré, nous l’avons rééquilibré de 3 manières différentes :

* en ajoutant un class\_weight dans l’entraînement du modèle,
* en diminuant le jeu de données des classes ayant le plus de lignes avec un RandomUnderSampling,
* en augmentant le jeu de données des classes ayant le moins de lignes avec un RandomOverSampling.

En réalisant la même compilation et entraînement que précédemment, nous obtenons les résultats présentés sur la Figure 42 et la Figure 43.

Le rééquilibrage du jeu de données permet d’augmenter le f1-score de la modalité tués dans tous les cas au détriment de l’accuracy. Cependant cette amélioration diminue aussi plus ou moins le f1-score des blessés hospitalisés et blessés légers selon le modèle.

Nous décidons donc d’optimiser les hyperparamètres afin d’améliorer encore les prédictions.

|  |  |  |
| --- | --- | --- |
| **class\_weight** | **undersampling** | **oversampling** |
| Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement  (a) | Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement  (a) | Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement  (a) |
| Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement  (b) | Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement  (b) | Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement  (b) |

Figure 42 : Rapports de classification (a) – Matrices de confusion (b)

|  |  |  |
| --- | --- | --- |
| **class\_weight** | **undersampling** | **oversampling** |
| Une image contenant texte, capture d’écran, diagramme, Tracé  Description générée automatiquement | Une image contenant texte, capture d’écran, Tracé, ligne  Description générée automatiquement | Une image contenant texte, capture d’écran, Tracé, ligne  Description générée automatiquement |
| Une image contenant texte, capture d’écran, Tracé, diagramme  Description générée automatiquement | Une image contenant texte, ligne, capture d’écran, Tracé  Description générée automatiquement | Une image contenant texte, capture d’écran, ligne, diagramme  Description générée automatiquement |

Figure 43 : Courbes d’évolution la loss et de l’accuracy par epoch

*Hyperparamétrage du modèle*

Pour essayer d’optimiser le modèle, nous décidons de rechercher les meilleurs paramètres pour :

* reduction de loss dans la compilation du modèle avec :

loss = tf.keras.losses.SparseCategoricalCrossentropy(

from\_logits=False,

ignore\_class=None,

reduction = reduction,

name="sparse\_categorical\_crossentropy",

)

* activation dans les couches Denses (sauf pour la dernière couche qui est une activation Softmax)
* kernel\_initializer ajouté ou non dans les couches Dense (sauf pour la dernière couche)
* batch\_size dans l'entraînement du modèle
* epochs dans l’entraînement du modèle

Voici le Tableau 15 regroupant les différentes valeurs testées pour chacun de ces paramètres :

|  |  |
| --- | --- |
| reduction de loss | sum, sum\_over\_batch\_size, None |
| activation | relu, sigmoid, tanh, leaky\_relu, swish, elu, selu, gelu |
| kernel\_initializer | RandomNormal, RandomUniform, TruncatedNormal, GlorotNormal, GlorotUniform, HeNormal, HeUniform, Orthogonal, VarianceScaling\_in, VarianceScaling\_out, LecunNormal, LecunUniform |
| batch\_size | 32, 64, 128, 256, 512, 1024, 2048 |
| epochs | 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 |

Tableau 15 : Tableau des paramètres testés pour le Deep Learning

En réalisant ces hyperparamètrages nous obtenons les résultats présentés dans la Figure 44.

Les hyperparamétrages permettent d’augmenter l’accuracy de tous les modèles de 1% à 3%. Pour le modèle utilisant class\_weight, ces modifications influencent peu les résultats. En revanche, cela permet d’augmenter notablement le f1-score des blessés légers dans le cas de l’undersampling et celui des blessés hospitalisés dans le cas de l’oversampling.

En comparant les modèles entre eux, on voit que l’undersampling obtient les moins bons résultats, alors que l’undersampling permet d’avoir la meilleure accuracy et les meilleurs f1-score. Cependant, les courbes de la loss par epoch et d’accuracy par epoch (Figure 45) montrent un surapprentissage dans le cas de l’oversampling et de l’undersampling. On décide donc de conserver le modèle utilisant class\_weight comme meilleur modèle car il n’y a pas de surapprentissage visible sur les courbes.

|  |  |  |
| --- | --- | --- |
| **class\_weight** | **undersampling** | **oversampling** |
| reduction de loss = sum\_over\_batch\_size  activation = gelu,  kernel\_initializer = GlorotNormal  batch\_size = 128  epochs = 70 | reduction de loss = sum  activation = swish  kernel\_initializer = VarianceScaling\_in  batch\_size = 64  epochs = 40 | reduction de loss = sum\_over\_batch\_size  activation = swish,  kernel\_initializer = LecunNormal  batch\_size = 32  epochs = 70 |
| Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement  (a) | Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement  (a) | Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement  (a) |
| Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement  (b) | Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement  (b) | Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement  (b) |

Figure 44 : Rapports de classification (a) – Matrices de confusion (b)

|  |  |  |
| --- | --- | --- |
| **class\_weight** | **undersampling** | **oversampling** |
| Une image contenant texte, capture d’écran, Tracé, diagramme  Description générée automatiquement | Une image contenant texte, Tracé, ligne, diagramme  Description générée automatiquement | Une image contenant texte, Tracé, capture d’écran, ligne  Description générée automatiquement |
| Une image contenant texte, capture d’écran, Tracé, ligne  Description générée automatiquement | Une image contenant texte, Tracé, ligne, diagramme  Description générée automatiquement | Une image contenant texte, capture d’écran, Tracé, ligne  Description générée automatiquement |

Figure 45 : Courbes d’évolution la loss et de l’accuracy par epoch

### Comparaison avec le meilleur modèle de Machine Learning

Le meilleur modèle est donc celui utilisant le class\_weight. En revanche, si on le compare avec le meilleur modèle de Machine Learning (Random Forest Classifier) que nous avons défini précédemment, le modèle de Deep Learning obtient de moins bons résultats (Figure 46). Nous décidons donc de ne pas pousser plus loin l’hyperparamétrage.

|  |  |
| --- | --- |
| **Meilleur modèle de Machine Learning**  **(Random Forest Classifier)** | **Meilleur modèle de Deep Learning**  **(class\_weight)** |
| Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement | Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement |
| Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement | Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement |

Figure 46 : Métriques et matrice de confusion pour les meilleurs modèles de Machine Learning et de Deep Learning

# ESSAIS D’OPTIMISATION DE LA PRÉDICTION DE LA GRAVITÉ DE L’ACCIDENT

## Classification binaire

En réalisant la classification multi-classes, nous nous sommes confrontés au problème de prédiction des personnes tuées (classe 2) et des blessés hospitalisés (classe 3) qui sont respectivement des classes très fortement et fortement minoritaires. Nous avons donc décidé de tester une classification binaire en créant 4 jeux de données à partir de notre jeu de données initial : ‘indemnes’ vs ‘autres’, ‘tués’ vs ‘autres’, ‘blessés hospitalisés’ vs ‘autres’, ‘blessés légers’ vs ‘autres’.

### Modélisation

#### Paramétrage du modèle

Les meilleurs résultats ayant été obtenus avec un algorithme Random Forest, nous décidons d’appliquer le même algorithme en recherchant, de la même manière que décrite précédemment, les paramètres optimaux.

Puis, nous appliquons un nouveau paramètre (possible uniquement sur la classification binaire) : monotonic\_cts. Pour chaque variable, nous testons les 3 valeurs de monotonic\_cts possibles (-1, 0 et 1) et nous enregistrons les résultats dans un tableau que l’on trie par f1-score-macro décroissant. Si le f1-score-macro est amélioré alors on garde cette valeur de monotonic\_cts pour cette variable et on relance la recherche pour savoir s’il faut aussi utiliser une autre variable dans monotonic\_cts. En revanche, si le f1 macro est moins bon, on ne garde pas cette valeur.

#### Résultats

|  |  |
| --- | --- |
| **Pour les indemnes**  **(Train accuracy = 85.42% - Test accuracy = 80.03%)** | |
| Paramètres :  {'bootstrap': True, 'class\_weight': 'balanced’, 'criterion': 'entropy’, 'max\_depth’: 20, 'min\_samples\_leaf': 1, 'min\_samples\_split': 10, 'n\_estimators': 100, 'n\_jobs': -1, 'random\_state': 42}   1. ‘monotonic\_cts’ à -1 pour ‘eq\_gilet’ et à 0 pour toutes les autres variables | |
| Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement | Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement |
|  | |
| **Pour les tués**  **(Train accuracy = 95.58% - Test accuracy = 94.06%)** | |
| Paramètres :  {'bootstrap': True, 'class\_weight': {0: 1, 1: 18}, 'criterion': 'entropy’, 'max\_depth': None, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2, 'n\_estimators': 100, 'n\_jobs': -1, 'random\_state': 42 }   1. ‘monotonic\_cts’ à -1 pour ‘circ’ et à 0 pour toutes les autres variables | |
| Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement | Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement |
|  | |
| **Pour les blessés hospitalisés**  **(Train accuracy = 85.6% - Test accuracy = 82.96%)** | |
| Paramètres :  {'bootstrap': True, 'class\_weight': {0: 1, 1: 3}, 'criterion': 'entropy’, 'max\_depth': 14, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2, 'n\_estimators': 100, 'n\_jobs': -1, 'random\_state': 42}   1. ‘monotonic\_cts’ à 1 pour ‘int’, à -1 pour ‘jour\_chome’ et à 0 pour toutes les autres variables | |
| Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement | Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement |
|  | |
| **Pour les blessés légers**  **(Train accuracy = 80.54% - Test accuracy = 72.78%)** | |
| Paramètres :  {'bootstrap': False, 'class\_weight': {0: 3, 1: 4}, 'criterion': 'gini’, 'max\_depth’: 15, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2, 'n\_estimators': 100, 'n\_jobs': -1, 'random\_state': 42}   1. ‘monotonic\_cts’ à -1 pour ‘motor’ et à 0 pour toutes les autres variables | |
| Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement | Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement |

Figure 47 : Métriques et matrice de confusion selon le jeu de données binaire utilisé

La classification binaire connaît les mêmes difficultés que la classification multiclasses pour prédire les classes minoritaires (les tués et les blessés hospitalisés). Cependant, l’accuracy est très nettement améliorée.

### Comparaison des résultats avec la classification multiclasses

Pour permettre la comparaison, nous allons diviser la matrice de confusion obtenue lors de la classification multi-classes de sorte à obtenir des résultats sous forme de classification binaire (Figure 48) :

La classification binaire permet d’améliorer de presque 2%, dans tous les cas, le nombre de vrais positifs :

* ‘indemnes’ vs ‘autres’ : diminution de 232 ‘indemnes’ vrais positifs, mais augmentation de 2108 ‘autres’ vrais positifs,
* ‘tués’ vs ‘autres’ : diminution de 287 ‘tués’ vrais positifs, mais augmentation de 2323 ‘autres’ vrais positifs
* ‘blessés hospitalisés’ vs ‘autres’ : augmentation de 1626 ‘blessés hospitalisés’ vrais positifs et de 602 ‘autres’ vrais positifs,
* ‘blessés légers’ vs ‘autres’ ; augmentation de 9565 ‘blessés légers’ vrais positifs, mais diminution de 7354 ‘autres’ vrais positifs.

La classification binaire est donc meilleure que la classification multi-classes.

### Interprétabilité des résultats

#### Analyse de l’importance donnée par le meilleur modèle

Nous nous sommes intéressés à l’importance donnée à chaque variable par l’algorithme pour chaque jeu de données binaire (Figure 49).

Nous pouvons constater que, dans les 3 premières variables, on retrouve systématiquement la variable correspondant à l’utilisation ou non de la ceinture de sécurité qui était d’ailleurs la variable ayant le plus d’importance lors de la classification multi-classes.

De plus, les variables correspondant au type de collision, à la latitude, à la longitude, à l’obstacle mobile heurté sont présentes dans 3 jeux de données sur 4.

#### Analyse de l’interprétation du meilleur modèle avec SHAP

En étudiant les graphiques d’importance des variables (Figure 50), mais également les graphiques de densité des valeurs SHAP pour chaque variable (Figure 51) en fonction de la modalité de la variable cible ‘grav’, nous pouvons dire que les 5 paramètres qui influent positivement sur la classification sont :

* **pour les indemnes** :
  + l’utilisation de la ceinture de sécurité,
  + être à l’avant du véhicule (les valeurs les plus faibles de la variable catégorielle ‘place\_rec’ sont soit conducteur et/ou passager avant),
  + ne pas porter de casque (certainement pour le fait de ne pas être en vélo/trottinette ou moto),
  + ne pas heurter d’obstacle fixe.

Pour la variable ‘catv’’ (variable catégorielle regroupant 6 modalités), la distinction entre les valeurs sur le graphique de densité des valeurs est plus floue et ne permet pas d’établir formellement quelles modalités influent positivement sur la classification.

* **pour les tués** :
  + ne pas utiliser de ceinture de sécurité,
  + rouler hors d’agglomération,
  + un âge plus avancé.

Pour les variables ‘catr’ (variable catégorielle regroupant 8 modalités) et ‘col’ (variable catégorielle regroupant 7 modalités) la distinction entre les valeurs sur le graphique de densité des valeurs est plus floue et ne permet pas d’établir formellement quelles modalités influent positivement sur la classification.

* **pour les blessés hospitalisés** :
  + ne pas utiliser de ceinture de sécurité,
  + rouler hors d’agglomération,
  + rouler sur une route bidirectionnelle,

Pour les variables ‘lat’ (variable numérique) et ‘col’ (variable catégorielle regroupant 7 modalités), la distinction entre les valeurs sur le graphique de densité des valeurs est plus floue et ne permet pas d’établir formellement quelles modalités ou valeurs influent positivement sur la classification.

* **pour les blessés légers** :
  + être à l’arrière du véhicule ou piéton(les valeurs les plus élevées de la variable catégorielle ‘place\_rec’ sont soit passager arrière, soit piéton),
  + être une femme,
  + ne pas utiliser de ceinture de sécurité,
  + porter un casque (certainement pour le fait d’être en vélo/trottinette ou moto),

Pour la variable ‘catv’ (variable catégorielle regroupant 6 modalités), la distinction entre les valeurs sur le graphique de densité des valeurs est plus floue et ne permet pas d’établir formellement quelles modalités influent positivement sur la classification.

On peut noter une légère différence de l’influence des variables par rapport à la classification multi-classes.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Classification multiclasses** | | | | | |
|  | | | | | |
|  |  | |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | **Indemnes** | **Tués** | **Blessés hospitalisés** | **Blessés légers** | | **Indemnes** | 39232 | 1262 | 2339 | 3304 | | **Tués** | 288 | 1488 | 1016 | 258 | | **Blessés hospitalisés** | 2225 | 3578 | 8632 | 3065 | | **Blessés légers** | 14677 | 2275 | 9058 | 19087 | | |  |  |
|  | | | | | |
| **Classification multiclasses réarrangée pour obtenir des résultats binaires** | | | | | |
|  | | | | | |
| |  |  |  | | --- | --- | --- | |  | **Autres** | **Indemnes** | | **Autres** | 48457 | 17190 | | **Indemnes** | 6905 | 39232 | | (78.45% de vrais positifs) | | | | |  |  |  | | --- | --- | --- | |  | **Autres** | **Tués** | | **Autres** | 101619 | 7115 | | **Tués** | 1562 | 1488 | | (92.24% de vrais positifs) | | | | | |  |  |  | | --- | --- | --- | |  | **Autres** | **Blessés hospitalisés** | | **Autres** | 81871 | 12413 | | **Blessés hospitalisés** | 8868 | 8632 | | (80.96% de vrais positifs) | | | | | |  |  |  | | --- | --- | --- | |  | **Autres** | **Blessés légers** | | **Autres** | 60060 | 6627 | | **Blessés légers** | 26010 | 19087 | | (70.80% de vrais positifs) | | | |
|  | | | | | |
| **Classification binaire** | | | | | |
|  | | | | | |
| |  |  |  | | --- | --- | --- | |  | **Autres** | **Indemnes** | | **Autres** | 50565 | 15082 | | **Indemnes** | 7137 | 39000 | | (80.12% de vrais positifs) | | | | |  |  |  | | --- | --- | --- | |  | **Autres** | **Tués** | | **Autres** | 103942 | 4792 | | **Tués** | 1849 | 1201 | | (94.06% de vrais positifs) | | | | | |  |  |  | | --- | --- | --- | |  | **Autres** | **Blessés hospitalisés** | | **Autres** | 82473 | 11811 | | **Blessés hospitalisés** | 7242 | 10258 | | (82.96% de vrais positifs) | | | | | |  |  |  | | --- | --- | --- | |  | **Autres** | **Blessés légers** | | **Autres** | 52706 | 13981 | | **Blessés légers** | 16445 | 28652 | | (72.78% de vrais positifs) | | | |

Figure 48 : Réarrangement de la matrice de confusion pour de la classification multiclasses et comparaison avec la classification binaire

|  |  |  |  |
| --- | --- | --- | --- |
| **‘indemnes’ vs ‘autres’**  **(34 variables)** | **‘tués’ vs ‘autres’**  **(34 variables)** | **‘blessés hospitalisés’ vs ‘autres’**  **(33 variables)** | **‘blessés légers’ vs ‘autres’**  **(29 variables)** |
|  |  |  |  |

Figure 49 : Importance des variables selon le jeu de données binaires

|  |  |  |  |
| --- | --- | --- | --- |
| **‘indemnes’ vs ‘autres’** | **‘tués’ vs ‘autres’** | **‘blessés hospitalisés’ vs ‘autres’** | **‘blessés légers’ vs ‘autres’** |
|  |  |  |  |

Figure 50 : Graphiques d’importance des variables selon SHAP selon le jeu de données binaires

|  |  |  |  |
| --- | --- | --- | --- |
| **‘indemnes’ vs ‘autres’** | **‘tués’ vs ‘autres’** | **‘blessés hospitalisés’ vs ‘autres’** | **‘blessés légers’ vs ‘autres’** |
|  |  |  |  |

Figure 51  : Graphique de densité des valeurs de SHAP selon le jeu de données binaires

## Etude des variables ‘catv’, ‘obs’ et ‘obsm’

Nous nous sommes aussi intéressés à l’amélioration des résultats par modification du jeu de données initial. Pour cela, nous avons regardé plus précisément l’influence des variables ‘catv’, ‘obs’ et ‘obsm’.

### Modifications des variables

#### Modification de la variable ‘catv’ et création de ‘catv\_percute’

La variable ‘catv’, qui prend comme modalité les différents types de véhicule, comporte 2 informations :

* le type de véhicule dans lequel la personne accidentée se situe quand elle est conducteur ou passager,
* le type de véhicule qui a percuté la personne accidentée quand elle est piétonne.

Pour rendre compte de cela, nous décidons de créer la variable ‘catv\_percute’ qui indique le véhicule qui a percuté la personne accidentée. Les modalités de ‘catv\_percute’ pour les piétons (ayant la ‘place\_rec’ = 4) sont les mêmes que celles de ‘catv’. Par contre pour les autres modalités de ‘place\_rec’ correspondant au conducteurs et passagers, on attribue une nouvelle modalité ‘6’ car on ne connaît pas le véhicule qui a percuté :

0 - Voiture

1 - Moto

2 - Poid lourd

3 - Transport en commun

4 - Vélo/Trottinette

5 - Autre véhicule

6 - Véhicule inconnu

Un nouvelle modalité ‘6’ est créée dans la variable ‘catv’ et est attribuée à tous les piétons :

0 - Voiture

1 - Moto

2 - Poid lourd

3 - Transport en commun

4 - Vélo/Trottinette

5 - Autre véhicule

6 - Piéton

#### Modification des variables ‘obs’ et ‘obsm’

Les variables ‘obs’ et ‘obsm’ correspondent respectivement à l’obstacle fixe ou mobile heurté par le véhicule. Or un piéton, par définition, n’est pas un véhicule et ne va donc pas heurter d’objet. Cependant, lorsque l’on regarde les valeurs de ‘obs’ et ‘obsm’ pour les piétons (‘place\_rec’ = 4), on se rend compte que certaines valeurs sont différentes de 0. Étant donné que, pour les piétons, la case ‘catv’ a été remplie par rapport au véhicule qui a percuté le piéton, il se peut que ces cases l’aient été de la même manière. Nous décidons donc de mettre les valeurs de ‘obs’ et ‘obsm’ à 0 lorsqu’il s’agit d’un piéton.

### Modélisation

Nous reprenons le meilleur modèle que nous avons obtenu pour la classification multiclasses : un algorithme de Random forest.

### Résultats et comparaison avec le jeu de données initial

La comparaison avec le jeu de données initiale (Figure 52) nous montre que ces modifications permettent d’obtenir un meilleur classement des personnes accidentées :

|  |  |
| --- | --- |
| **Jeu de données initial** | **Jeu de données avec modification de ‘catv’, ‘obs’, obsm’ + création de ‘catv\_percute’** |
| **Random Forest**  **(Train accuracy = 65.52%)**  **(Test accuracy = 61.22%)** | **Random Forest**  **(Train accuracy = 65.83%)**  **( Test accuracy = 61.36%)** |
| paramètres :  {'bootstrap': True, 'class\_weight': 'balanced', 'criterion': 'entropy', 'max\_depth': 13, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2, 'n\_estimators': 100, 'n\_jobs': -1, 'random\_state': 42} | paramètres :  {'bootstrap': True, 'class\_weight': 'balanced', 'criterion': 'entropy', 'max\_depth': 13, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2, 'n\_estimators': 100, 'n\_jobs': -1, 'random\_state': 42} |
|  |  |
|  |  |
| 61.22% de vraies positifs | 61.36% de vraies positifs |

Figure 52 : Comparaison des métriques et matrice de confusion pour le jeu de données initial et le jeu de données avec modification de variables

En comparant les 2 modèles, on peut voir que le retraitement des variables ‘catv’, ‘obs’ et ‘obsm’ permet de légèrement améliorer l’accuracy. On peut noter que les valeurs de précision et de recall sont soit identiques, soit légèrement améliorées (amélioration de la précision pour les tués et amélioration du recall pour les blessés hospitalisés et les blessés légers).

L’ajout de la variable ‘catv\_percute’ pour les piétons (qui représente seulement 7.62% du jeu de données) a permis d’améliorer les résultats de 0.14%. Nous ne disposons pas de cette information lorsque la personne accidentée est conducteur ou passager. Or le type de véhicule qui percute peut forcément impacter la gravité de l’accident. Il serait donc intéressant d’ajouter cette variable lors de la collecte des informations à la suite d’un accident afin d’avoir la variable ‘catv\_percute’ remplie pour tous les accidentés et non pas seulement pour les piétons.

## Ajout de la variable ‘nb\_usagers\_gr’

Nous avons déjà fait des essais d’optimisation des prédictions en réalisant des modélisations avec classifications binaires ou par ajout de la variable ‘catv\_percute’.

### Création de la variable ‘nb\_usagers\_gr’

Nous décidons d’essayer d’ajouter une variable nombre d’usagers (‘nb\_usagers’) en calculant, avant même de supprimer des lignes de notre jeu de données, le nombre de personnes accidentées lors d’un accident. Ceci peut facilement être connu grâce à la variable ‘Num\_Acc’ (numéro de l’accident) est qui est reprise pour chaque personne impliquée dans l’accident. La visualisation de cette variable permet de voir que la majorité des accidents impliquent 2 personnes (Figure 53a) et que l’on obtient 90% des personnes accidentées en prenant les accidents impliquant de 1 à 5 personnes (Figure 53b).

|  |  |
| --- | --- |
| Une image contenant texte, capture d’écran, ligne, Tracé  Description générée automatiquement  (a) | Une image contenant texte, ligne, diagramme, Tracé  Description générée automatiquement  (b) |

Figure 53 : Histogramme du nombre d’accidents en fonction du nombre de personnes accidentées (a) – Courbe cumulé du pourcentage du nombre d’accidents en fonction du nombre de personnes accidentées

De ce fait, nous décidons de créer une nouvelle variable ‘nb\_usagers\_gr’ qui conserve les modalités suivantes :

* 1 ⇒ 1 usager
* 2 ⇒ 2 usagers
* 3 ⇒ 3 usagers
* 4 ⇒ 4 usagers
* 5 ⇒ 5 ou + usagers

### Modélisation

Nous reprenons le meilleur modèle que nous avons obtenu pour la classification multiclasses : un algorithme de Random forest.

### Résultats et comparaison avec le jeu de données initial

La comparaison avec le jeu de données initiale (Figure 54) nous montre que ces modifications permettent d’obtenir :

|  |  |
| --- | --- |
| **Jeu de données initial** | **Jeu de données avec ajout de ‘nb\_usagers\_gr’** |
| **Random Forest**  **(Train accuracy = 65.52%)**  **(Test accuracy = 61.22%)** | **Random Forest**  **(Train accuracy = 65,99%)**  **( Test accuracy = 61,67%)** |
| paramètres :  {'bootstrap': True, 'class\_weight': 'balanced', 'criterion': 'entropy', 'max\_depth': 13, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2, 'n\_estimators': 100, 'n\_jobs': -1, 'random\_state': 42} | paramètres :  {'bootstrap': True, 'class\_weight': 'balanced', 'criterion': 'entropy', 'max\_depth': 13, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2, 'n\_estimators': 100, 'n\_jobs': -1, 'random\_state': 42} |
| Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement | Une image contenant texte, capture d’écran, Police  Description générée automatiquement |
| Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement | Une image contenant texte, capture d’écran, Police, nombre  Description générée automatiquement |
| 61.22% de vraies positifs | 61,67% de vraies positifs |

Figure 54 : Comparaison des métriques et matrices de confusion du jeu de données initial et du jeu de donnée avec ajout de ‘nb\_usagers\_gr’

En ajoutant cette variable, on note une augmentation de l’accuracy de 0,45% qui se traduit par une légère augmentation du f1-score de la modalité tués. Ainsi il peut donc être intéressant d’ajouter la variable ‘nb\_usagers\_gr’ en plus des autres optimisation déjà évoquée (ajout de la variable ‘catv\_percute’ et classification binaire).

# CONCLUSION

Pour répondre à la problématique de mieux comprendre la gravité des blessures causées par les accidents de la route en France, nous avons eu recours à des algorithmes de classification issus de différentes familles, à savoir :

* des algorithmes linéaires : régression logistique,
* des algorithmes de regroupement : machines à vecteurs de support (SVM), K plus proches voisins (KNN)
* des algorithmes à arbres de décision : arbre de décision, forêt aléatoire,
* des apprentissages d’ensemble : CatBoost, XGBoost,
* du Deep Learning.

Pour optimiser les hyperparamètres de ces modèles, différentes méthodes d’exploration ont été mises en oeuvre, telles que la validation croisée (cross-validation), la recherche par grille (GridSearchCV), la recherche aléatoire (RandomSearch) ou l’optimisation bayésienne (Optuna).

De façon générale, les gains obtenus par ces procédures d’optimisation restent relativement modérés (Figure 55), ce qui peut être lié à l’absence dans la base de données de variables fortement influentes pour prédire la gravité des accidents routiers. Rappelons que des informations telles que les vitesses lors des accidents, l’état de santé des usagers, … sont absentes (non communiquées) de la base de données mise à disposition par le gouvernement.

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

Description générée automatiquement

Figure 55 : Diagrammes radars de comparaison des modèles en termnes de Precision, Recall, Specificity, F1-Score et Index Balanced Accuracy

Finalement, notre meilleur modèle repose sur les forêts aléatoires et atteint les performances listées à la Figure 56.

|  |  |
| --- | --- |
| **Random Forest**  **(Train accuracy = 65.52% - Test accuracy = 61.22%)** | |
| Paramètres :  {'bootstrap': True, 'class\_weight': 'balanced', 'criterion': 'entropy', 'max\_depth': 13, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2, 'n\_estimators': 100, 'n\_jobs': -1, 'random\_state': 42} | |
|  |  |

Figure 56 : Métriques et matrice de confusion pour le meilleur modèle

Tous ces algorithmes se heurtent à la difficulté de gestion d’un jeu de données déséquilibré. La création d’un modèle pénalisé par l’ajout de poids différents aux différentes classes améliore les résultats (plus que les techniques de rééquilibrage du jeu de données), mais les résultats continuent de montrer une faiblesse dans la prédiction des classes minoritaires.

Les techniques d’interprétabilité des résultats des modèles de machine learning ont été appliquées pour mieux comprendre les facteurs contribuant aux accidents de la route. Des plus simples (ordres de grandeur des coefficients de régression logistique, permutation feature importance) aux plus complexes (celles basées sur la valeur de Shapley), toutes mettent en évidence l’importance des équipements de sécurité (notamment l’utilisation ou non de la ceinture de sécurité et le port ou non du casque), mais aussi le fait de rouler en agglomération ou non. On note aussi que la meilleure méthode, une forêt aléatoire, permet, comme les autres méthodes basées sur les arbres de classification, de prendre en compte l’effet *non-linéaire* des variables continues ainsi que les *interactions* entre variables. C’est probablement grâce à cette flexibilité qu’elle a pu surpasser la régression logistique : il apparaît en effet lors de l’interprétabilité du modèle que les variables continues « âge usager » et « longitude, latitude » ont une forte importance dans la forêt aléatoire, mais une importance faible dans la régression logistique, dans laquelle elles ont été introduites linéairement.

Nous pouvons effectivement vérifier dans la Figure 57que la position géographique influe sur la probabilité qu’un accidenté soit hospitalisé ou tué, ceci de façon non-linéaire. Cette carte a été réalisée dans R avec un [modèle bayésien a effet spatial aléatoire](https://www.tandfonline.com/doi/abs/10.1080/1361593022000029467), et indique les zones où le risque d’accident grave excède le risque moyen national (rouge), est équivalent (jaune) ou est en deçà (vert). La plus faible gravité des accidents en agglomération est visible à l’œil nu, sur Paris notamment, alors que le modèle de lissage ne contient aucune information autre que la position géographique. Une piste d’amélioration du modèle pourrait donc être de modéliser plus précisément le processus spatial, soit avec une approche géostatistique pure, comme publié par [Barmoudeh et al](https://www.sciencedirect.com/science/article/pii/S0001457522000069)  pour classifier les accidents en Iran en 3 classes de gravité, soit en intégrant des composantes géospatiales à des modèles de machine learning de type « arbres de classification », comme proposé par [Effati et al](https://www.researchgate.net/publication/277724282_A_semantic-based_classification_and_regression_tree_approach_for_modelling_complex_spatial_rules_in_motor_vehicle_crashes_domain_A_semantic-based_classification_and_regression_tree_approach), en Iran aussi.

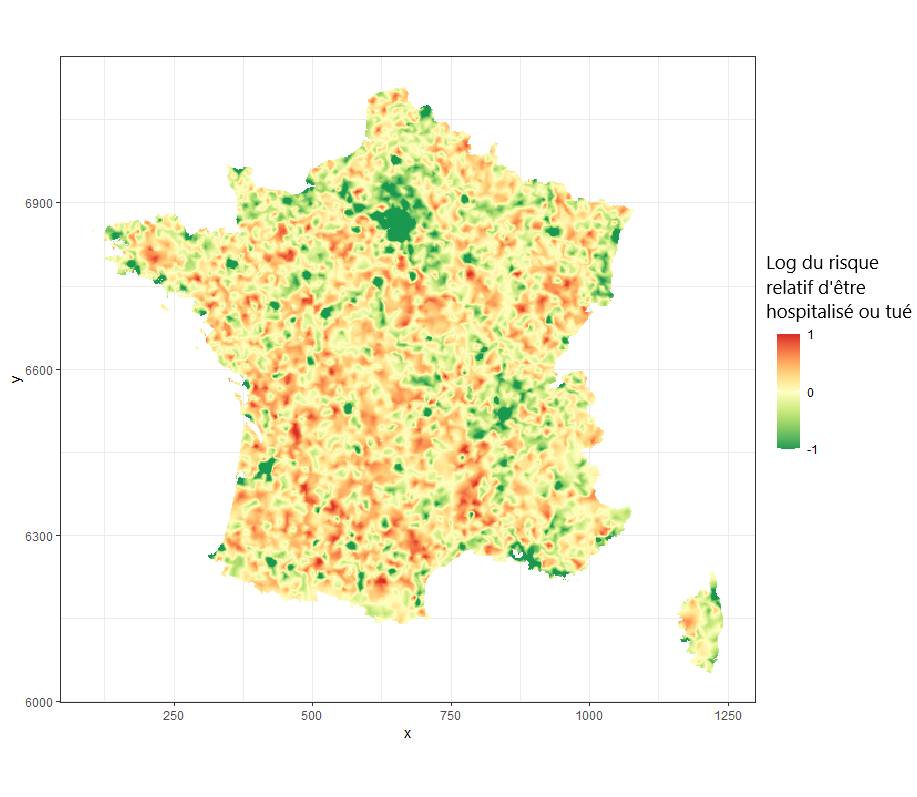


Figure 57. Risque relatif (échelle log) d'être hospitalisé ou tué pour une personne de la base de données

Finalement, nous avons pu observer une légère amélioration des résultats :

* augmentation en moyenne de 2% des vrais positifs pour chaque jeu de donnée binaire lorsque l’on sépare le jeu de données composé de 4 classes en 4 jeux de données binaires,
* augmentation de 0.14% des vrais positifs en modifiant les variables ‘catv’, ‘obs’, obsm’ et en créant la variable ‘catv\_percute’,
* augmentation de 0.45% des vrais positifs en ajoutant la variable ‘nb\_usagers\_gr’.

Nous pouvons espérer qu’en combinant les 3 approches cela permettra encore une amélioration des résultats. En revanche, la seconde approche reste incomplète car nous ne disposons des types de véhicules qui ont percuté la personne accidentée que pour les piétons. C’est pourquoi nos résultats laissent penser que notre approche de modélisation pourrait être optimisés en incluant cette variable pour tous les accidentés. De plus, l’inclusion de variables telles que la vitesse au moment de l’accident, l’état de santé de la personne ou encore le fait d’être sous emprise de l’alcool ou de stupéfiants permettrait certainement d’affiner les prédictions.