Rapport de Projet – IA

Projet A3 : Accidentologie

Développement d'une application d'étude des incidents de la route en 2009

Tristan Saëz – Adrien Le Boucher – Vincent LeBrenn

Année 2022 - 2023

Table des matières

Préambule

Ce rapport est le résultat des travaux effectués lors de notre projet en Intelligence Artificielle effectué du 30 mai 2023 au 2 juin 2023. Il consiste en l'étude des accidents de la route en 2009 et au développement d'un projet d'apprentissage automatique en lien avec ces accidents.

Le projet se divise en plusieurs partie : Préparation et Visualisation des données, Apprentissage non-supervisé (clustering), Apprentissage supervisé, Evaluation et Analyse des résultats et Création de scripts d'automatisation via commandes bash.

Les objectifs de ce projet furent les suivants :

• Comprendre les étapes d'un projet d'apprentissage automatique

• Apprendre à évaluer une méthode d'apprentissage automatique

• Manipuler les différentes techniques d'apprentissage automatique :

• Apprentissage non supervisé : réduction de dimensionnalité, clustering

• Apprentissage supervisé : régression, classification

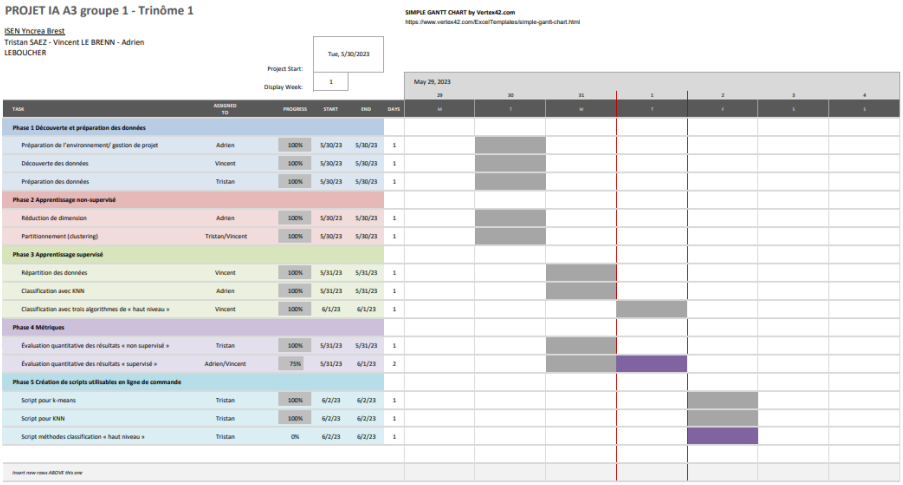
La prochaine partie est dédiée à l'environnement de travail et la gestion de projet

Gestion de projet & Environnement de travail

La première partie fut dédiée à la configuration de notre environnement de travail :

- Un dépôt GIT utilisé pour la programmation collaborative
- Un diagramme de Gantt pour le partage des tâches

Dépôt Git : https:/github.com/TrLSeysh/ProjetCIR3_IA



Découverte & Préparation des données

Comme n'importe quelle base de données, elle ne fut pas exploitable sans la préparer. Plusieurs étapes furent nécessaires lors de la préparation du jeu de données pour le rendre exploitable :

Découverte

-Permet de découvrir le nombre dimensions des données
-Permet de connaître la répartition des données. Ici on a des cas déséquilibrés pour la valeur cibles et les instances

Préparation

- Convertir les valeurs non-numériques
- Mettre les dates et heures au bon format

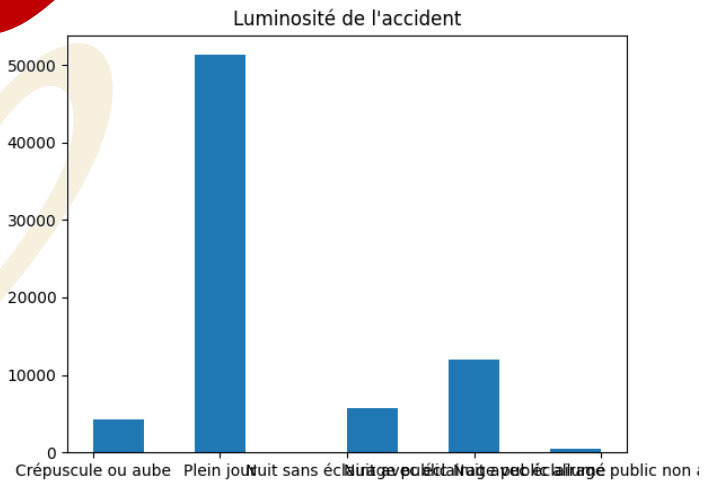
Découverte des données

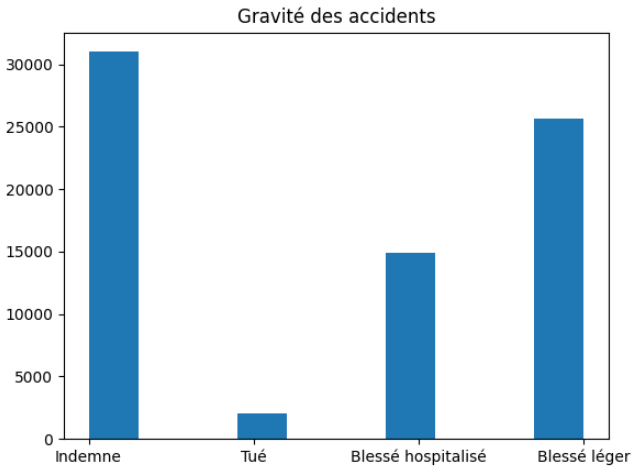
Nous avons tout d'abord étudié l'élément : {descr_grav} qui représente la gravité de l'accident. Nous en avons déduit qu'un regroupement de label pouvait être judicieux pour n'en garder que 3 :

- Indemne
- Blessé léger
- Accident grave (Tué & Blessé Hospitalisé)

Ceci permettra de combler les écarts entre les labels pour avoir une précision accrue.

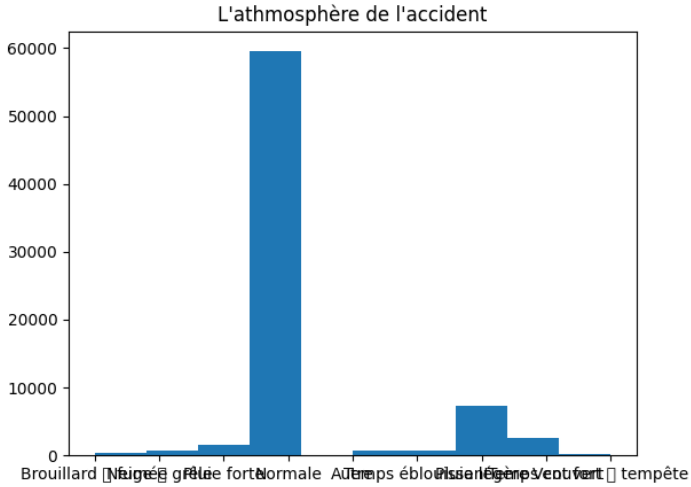
Ensuite, nous avons étudié le paramètre {descr_lum} qui décrit la luminosité lors de l'accident. Bien que ce trait puisse être intéressant à étudier, nous avons décidé de ne pas le modifier pour se concentrer sur l'élément le plus important de la base : la gravité de l'accident.

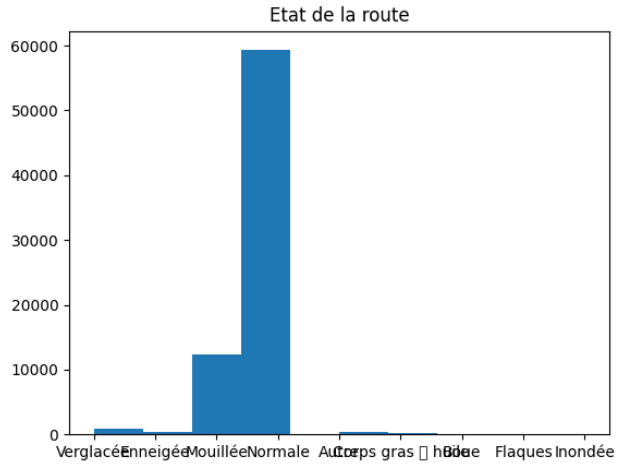




On a 22 features pour 59004 instances par classes ce qui est conséquent, donc nous allons faire appel à la réduction de dimension. Ensuite nous allons équilibré les données afin d'optimiser le modèle afin d'obtenir une meilleur classification.

Notre objectif est avant tout de prédire la gravité de l'accident, de ce fait tous les autres traits, bien que pertinent pour préciser la prédiction des accidents, ils ont été mis de côté. Le constat est donc le même pour les paramètres {descr_athmo} et {descr_etat_surf} qui ont, eux, été retirés de la base d'apprentissage.





Préparation des données

2 modifications principales ont dû être effectuée sur la base de données pour permettre une exploitation idéale de son contenu. La première fut la conversion des valeurs non-numériques en entiers.

Pour se faire, il a d'abord fallu récupérer chaque colonne et vérifier le type de chacune d'elles. Nous avons utilisé la fonction isinstance() de python qui permet de vérifier le type d'une variable. Il était également nécessaire de ne pas modifier certaines colonnes telles que le code_insee et la date qui sont traités différemment.

Ensuite, la fonction factorize() de la library pandas nous a permis de donner à chaque valeur non-numérique un entier.

Enfin, pour garder en mémoire la valeur liée à chaque indice nous les avons intégré à un fichier excel via la fonction to_excel(). Le résultat est le suivant :

	index	value
descr_cat_veh	0	PL seul > 7,5T
descr_cat_veh	1	VU seul 1,5T <= PTAC <= 3,5T avec ou sans remorque
descr_cat_veh	2	VL seul
descr_cat_veh	3	Autocar
descr_cat_veh	4	PL > 3,5T + remorque
descr_cat_veh	5	Cyclomoteur <50cm3
descr_cat_veh	6	Motocyclette > 125 cm3
descr_cat_veh	7	Tracteur routier + semi-remorque
descr_cat_veh	8	Tracteur agricole
descr_cat_veh	9	PL seul 3,5T <PTCA <= 7,5T
descr_cat_veh	10	Autobus
descr_cat_veh	11	Train
descr_cat_veh	12	Scooter > 125 cm3
descr_cat_veh	13	Scooter < 50 cm3
descr_cat_veh	14	Voiturette (Quadricycle à moteur carrossé) (anciennement "voiturette ou tricycle à moteur")
descr_cat_veh	15	Autre véhicule
descr_cat_veh	16	Bicyclette
descr_cat_veh	17	Motocyclette > 50 cm3 et <= 125 cm3
descr_cat_veh	18	Scooter > 50 cm3 et <= 125 cm3
descr_cat_veh	19	Engin spécial
descr_cat_veh	20	Quad lourd > 50 cm3 (Quadricycle à moteur non carrossé)
descr_cat_veh	21	Tramway
descr_cat_veh	22	Tracteur routier seul
descr_cat_veh	23	Quad léger <= 50 cm3 (Quadricycle à moteur non carrossé)
descr_agglo	0	Hors agglomération
descr_agglo	1	En agglomération
descr_athmo	0	Brouillard – fumée
descr_athmo	1	Neige – grêle
descr_athmo	2	Pluie forte
descr_athmo	3	Normale
descr_athmo	4	Autre
descr_athmo	5	Temps éblouissant
descr_athmo	6	Pluie légère
descr_athmo	7	Temps couvert
descr_athmo	8	Vent fort – tempête

Premières lignes du fichier de conversion obtenu

La seconde modification nécessaire fut le changement de format des dates. Ici, nous avons dû utiliser la library datetime et ses fonctions .month et .hour qui permettent de récupérer le mois et l'heure de l'accident (l'année étant toujours 2009, il n'était pas intéressant de la conserver) et de les mettre au bon format datetime.

Nous avons donc ajouté 2 colonnes mois et heure puis supprimé la colonne date.

Apprentissage non-supervisé

La première partie de notre réseau d'apprentissage automatique fut l'apprentissage non-supervisé et plus précisément les méthodes de clustering des données. Mais pour pouvoir faire des clusters cohérents de manière efficace, il y eut la nécessité de réduire le nombre de features et de corréler certains attributs entre eux : réduire les dimensions de la base de données.

Réduction de dimension

Nous avons commencé par calculer le coefficient de corrélation en fonction de la gravité de l'accident (feature maîtresse de l'apprentissage). Voici les résultats :

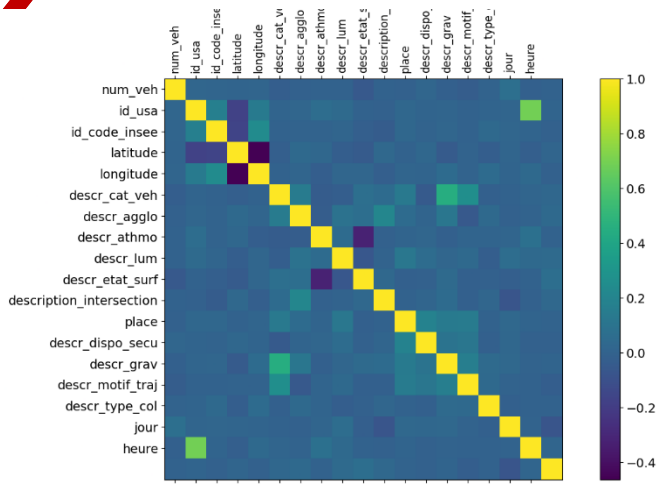
Coefficient de corrélation en fonction de la gravité des accidents		
feature	Coef de corrélation	Supprimée ?
jour	0.01671	✗
id_usa	0.001948	✗
latitude	0.006072	
longitude	0.008392	
descr_athmo	0.012155	✗
id_code_insee	0.012865	✗
descr_etat_surf	0.015064	✗
descr_agglo	0.016364	✗
description_intersection	0.019234	
heure	0.025678	✗
descr_lum	0.030840	
num_veh	0.034247	✗
descr_motif_traj	0.054042	✗
descr_type_col	0.055053	
X (id)	0.061824	✗
place	0.110989	✗
an_nais	0.132073	
descr_dispo_secu	0.222666	✗
descr_cat_veh	0.239771	
descr_grav	1.000000	

Le coefficient de corrélation est contenu entre 1 et -1. Plus la valeur s'approche d'un extrême, plus la corrélation est forte où, proche de 1, elle est très corrélée positivement, et proche de -1, elle est très corrélée négativement. Plus le coefficient se rapproche de 0

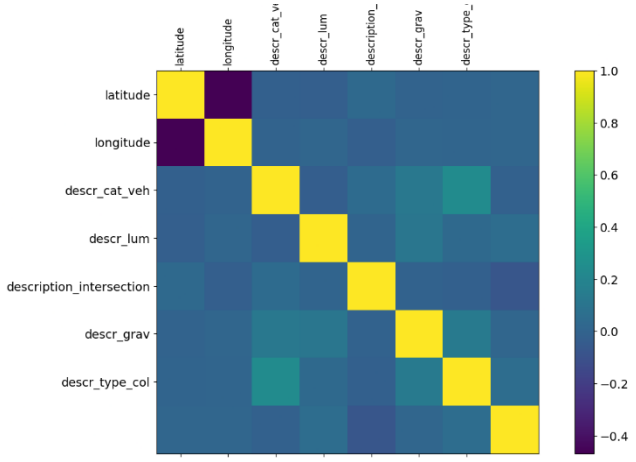
moins la corrélation est forte et moins les variables s'influencent les unes aux autres. Ici on peut voir qu'à partir de X(id) la corrélation est faible. Mais encore plus faible à partir de longitude.

La colonne « Supprimée ? » montre les features qui ont été supprimées par la suite

Pour représenter les différentes corrélations nous avons décidé de les mettre sous forme de matrices, ce qui a donné les résultats suivant en comparant avant et après réduction de dimension :



Matrice de corrélation avant réduction de dimension



Matrice de corrélation après réduction de dimension

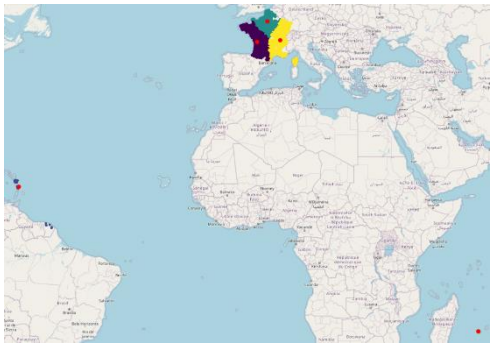
Clustering

Nous avons décomposé le clustering en 2 étapes, une première avec un calcul dit « from scratch » et une seconde en utilisant la bibliothèque scikit-learn pour comparer les résultats obtenus.

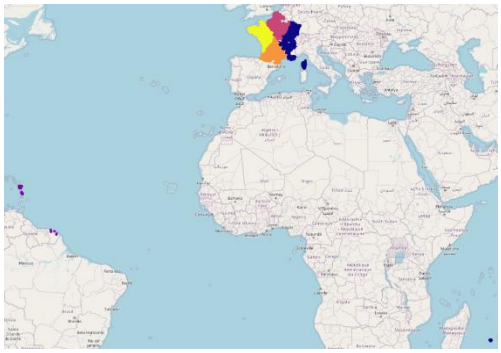
Pour obtenir des résultats pertinents, nous avons fait varier le nombre de cluster : n = [3, 5, 8, 12] et également la méthode de calcul « from scratch » : méthode = [L1, L2, Haversine].

Voici les résultats obtenus lors du clustering avec les différentes méthodes pour 8 clusters :

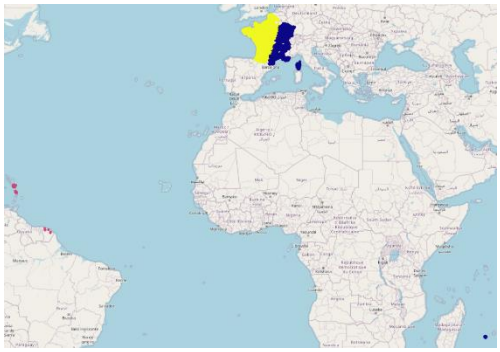
Méthode : Scikit-learn



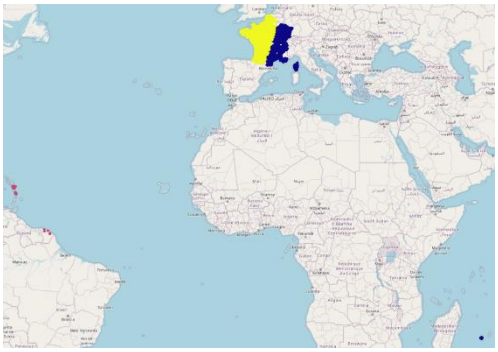
Méthode : Haversine



Méthode : L2



Méthode : L1



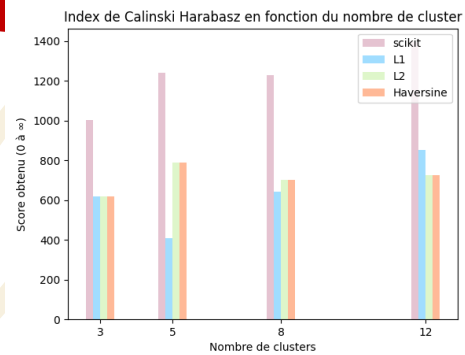
Après avoir obtenu des résultats sur un nombre de clusters variable nous les avons analysé avec des indicateurs de rendement :

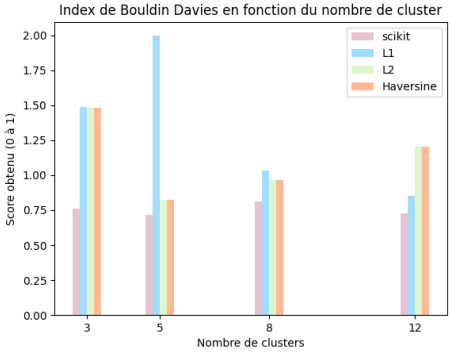
Nombre de cluster				
3				
Méthode	scikit	L1	L2	Haversine
Metric	Score	Score	Score	Score
Coefficient de silhouette	0.4396002394042591	0.3099044630729121	0.3066844697028553	0.3066844697028553
Index de Calinski Harabasz	1002.1034962689257	619.8313597309689	617.8453619262061	617.8453619262061
Index de Bouldin Davies	0.759965228551648	1.4849925933563453	1.4822945609067348	1.4822945609067348

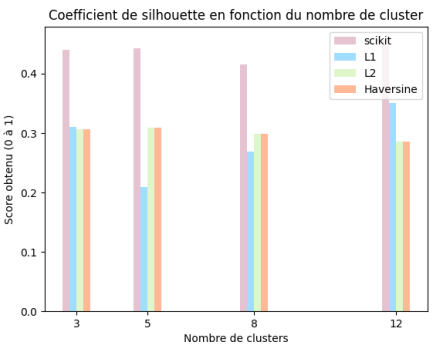
Nombre de cluster				
5				
Méthode	scikit	L1	L2	Haversine
Metric	Score	Score	Score	Score
Coefficient de silhouette	0.44375880274301455	0.20855898682664645	0.3088980353693082	0.3088980353693082
Index de Calinski Harabasz	1242.0986175669645	407.9566647218841	790.2322194733962	790.2322194733962
Index de Bouldin Davies	0.7179728279735788	1.9968265749287402	0.8234573037440249	0.8234573037440249

Nombre de cluster				
8				
Méthode	scikit	L1	L2	Haversine
Metric	Score	Score	Score	Score
Coefficient de silhouette	0.41307080643298366	0.26924551509454425	0.29935651624751863	0.29935651624751863
Index de Calinski Harabasz	1230.1642992694271	642.0362689580388	700.0842140392815	700.0842140392815
Index de Bouldin Davies	0.8160015678900034	1.033938304118349	0.9646450498949747	0.9646450498949747

Nombre de cluster				
12				
Méthode	scikit	L1	L2	Haversine
Metric	Score	Score	Score	Score
Coefficient de silhouette	0.4622257479910289	0.35067055901472927	0.28521006951152567	0.28521006951152567
Index de Calinski Harabasz	1366.3215155872706	851.2028692835758	724.7676272093031	724.7676272093031
Index de Bouldin Davies	0.7528093020232839	0.8516612387507877	1.2022495277774983	1.2022495277774983







Apprentissage supervisé

Répartition des données

Hold out : train_test_split() de 20% test et 80% train

Leave one out : itérations des entrainements et test sur la base de données

Classifications kNN

Nous avons décomposé la classifications kNN en 2 étapes, une première avec un calcul dit « from scratch » et une seconde en utilisant la bibliothèque scikit-learn pour comparer les résultats obtenus.

Dans la partie « from scratch », nous avons utilisé la distance euclidienne pour prédire les classes de gravité d'accidents. Les résultats sont :

- Prédictions : [0 0 0 0 0 0 0]

- Base d'entrainement : [47.1 , -1.81667, 6, 3, 0, 1966, 5]

- Base de test : [46.3167, -0.466667, 5, 2, 0, 1992, 1]

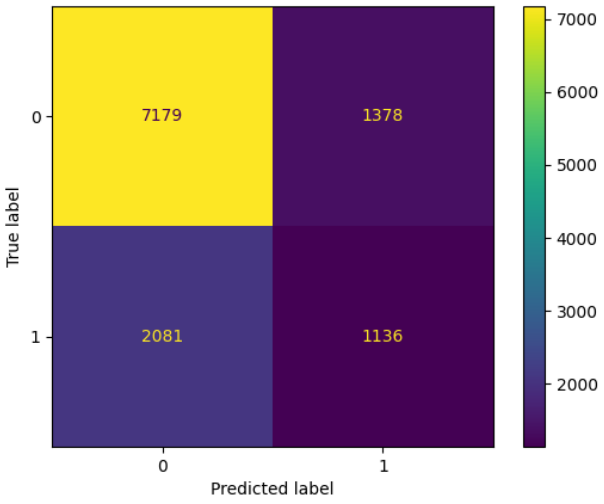
Dans la partie scikit-learn, nous avons utilisé le model KNeighborsRegressor. En premier lieu à 3 voisins, les résultats étaient concluants. Néanmoins grâce à GridSearchCV nous avons pu améliorer sa performance avec 48 voisins. Nous avons donc utilisé plusieurs méthodes d'évaluation quantitative.

La première, le calcul du taux d'apprentissage, de la précision et du rappel :

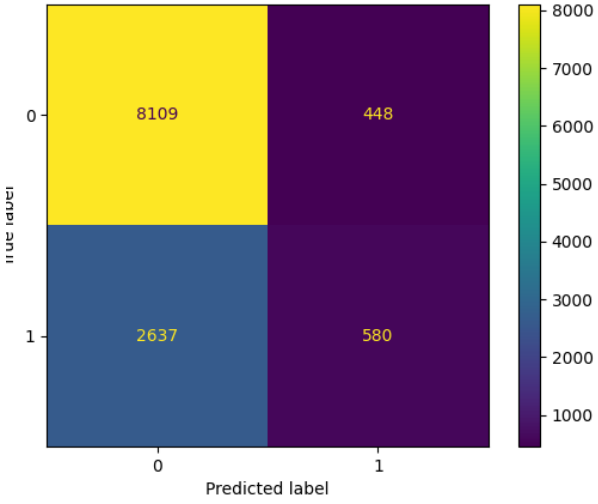
	K_neighbors = 3	K_neighbors = 48
Taux d'apprentissage	0.7062170885000849	0.7379819942245626
Précision	0.6135697545151042	0.6594043498947395
Rappel	0.5960431408620842	0.563968700228848

- Le taux d'apprentissage est une mesure de la capacité du classifieur à apprendre à partir des données d'entraînement. Ici le taux k=3 est inférieur à celui de k=48 donc le model à 48 voisins à été celui qui a le mieux appris.
- La précision indique combien du nombre total de prédictions spécifiées comme positives sont correctement affectées. Une précision élevée indique que le modèle a tendance à faire moins de fausses prédictions positives. Donc le model 48 est le meilleur.
- Le rappel est une mesure de la capacité du classifieur à trouver toutes les instances positives. Un rappel élevé indique que le modèle est capable de détecter la plupart des instances positives réelles. Cependant, ici le model 3 est meilleur.

Ensuite les matrices de confusion qui permettent d'avoir une vision graphique des résultats.



Matrice de confusion K = 3



Matrice de confusion K = 48

- Pour le modèle de classification kNN à k = 3 avec les classes 0(Indemne et blessé léger) et 1(Tué et blessé grave).

Le modèle a correctement prédit que l'accident était de la classe 0 (Indemne et blessé léger), il y a 7179 accidents correctement classifiés comme Indemne et blessé léger.

1378 accidents qui ont été classés à tort comme Indemne et blessé léger, mais ils étaient en réalité Tué et blessé grave.

2081 accidents qui ont été classés à tort comme Tué et blessé grave, mais ils étaient en réalité Indemne et blessé léger.

Et 1136 accidents correctement classifiés comme Tué et blessé grave.

- Pour le modèle de classification kNN à k = 48.

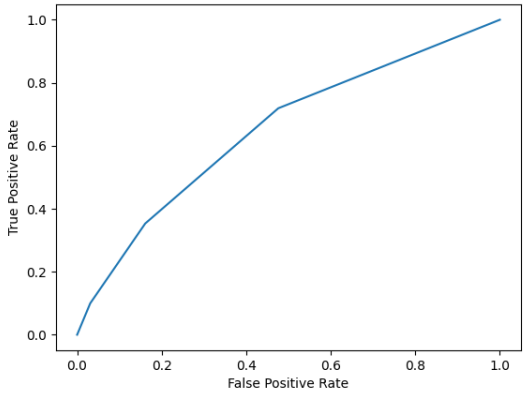
Le modèle a correctement prédit que l'accident était de la classe 0 (Indemne et blessé léger), il y a 8109 accidents correctement classifiés comme Indemne et blessé léger.

448 accidents qui ont été classés à tort comme Indemne et blessé léger, mais ils étaient en réalité Tué et blessé grave.

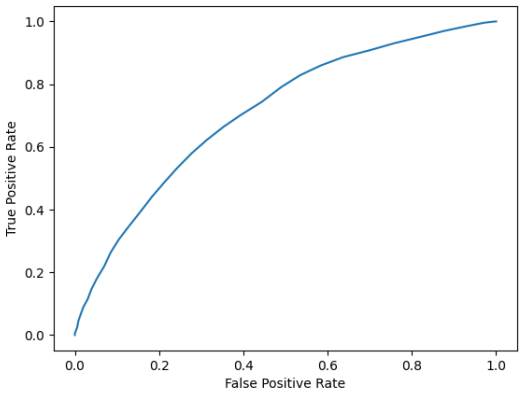
2637 accidents qui ont été classés à tort comme Tué et blessé grave, mais ils étaient en réalité Indemne et blessé léger.

Et 580 accidents correctement classifiés comme Tué et blessé grave.

Puis les courbes ROC (Receiver Operating Characteristic) graphique qui représente la performance d'un modèle de classification binaire.



Courbe ROC – K = 3

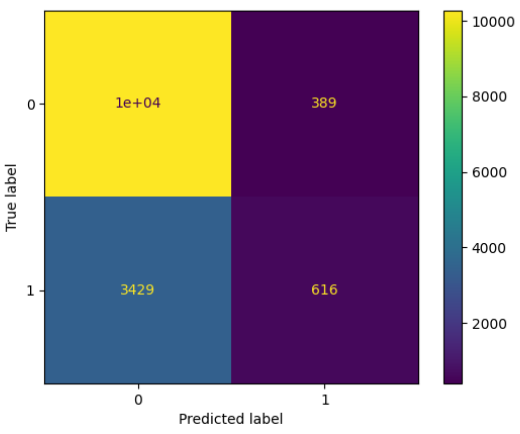


Courbe ROC – K = 48

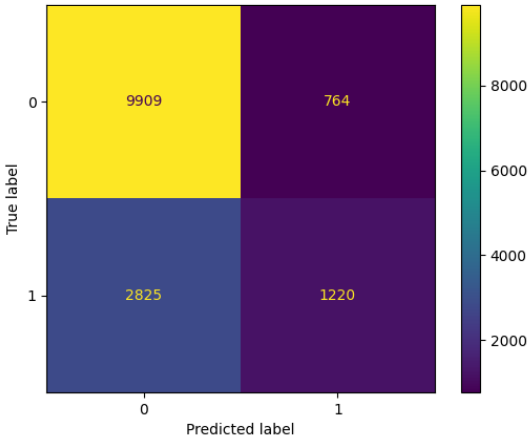
Les deux courbes sont presques similaire, néamoins celle de k=48 est légerement plus concave vers l'axe des True Positive Rate(TPR) ce qui fais que le model k=48 est plus performant.

Classifications « haut niveau »

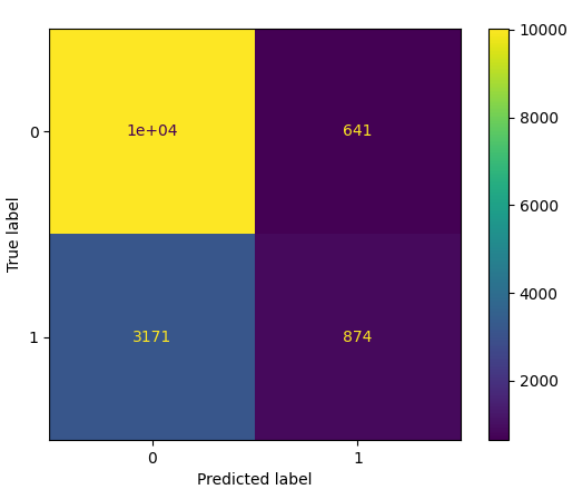
Attendus : résultats de classification, tableaux/graphes d'optimisation des paramètres, comparatif entre les 3 méthodes avec les interprétations



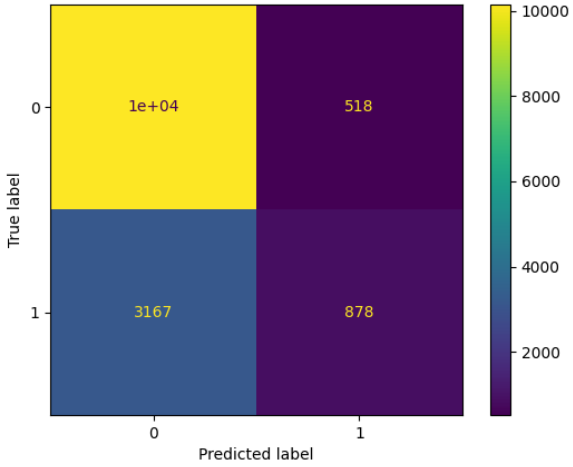
SVM



Random Forest



MLP



Vote par majorité

Utilisation de la méthode Hold out optimisé avec Grid Search :

Dans l'ensemble des données les prédictions en True Positives et True Négatives sont excellentes et cohérentes. En effet il y a un déséquilibre dans la répartition de la valeur cible. Indemne et Blessé léger sont majoritaire comparé à tuer et blessé lourd. Ce déséquilibre peut influencer les performances du modèle, car il est plus facile pour le modèle de prédire les classes majoritaires que les classes minoritaires dû au nombre de cas.

En comparant ces données SVM et MLP sont similaires True négatives élevée et True positives plus faible par rapport au classifier Random Forest qui a une prédiction des True Positives mais plus faible en True négatives.

Chaque modèle a ses forces et ses faiblesses, et il est important de prendre en compte le contexte spécifique de la tâche et les exigences particulières lors du choix d'un modèle.

	Accuracy	Précision	F1-Score
Support Vector Machine	0.7405897540	0.712290368	0.67867909
Random Forest	0.7561489332	0.733229090	0.72520458
Multi layer perceptron	0.7409974181	0.7095521160	0.69579216
Vote majorité	0.7496263079	0.7256285234	0.702498338

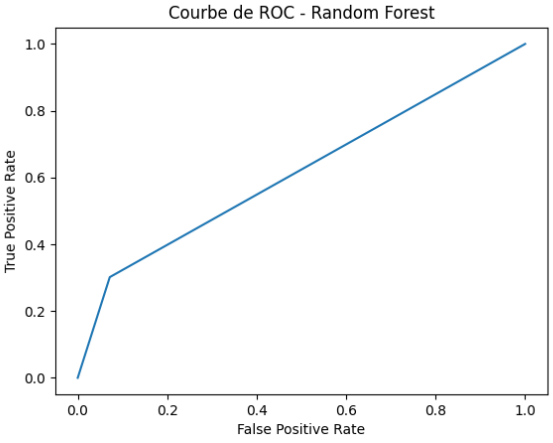
On peut observer que le modèle Random Forest a obtenu les valeurs d'Accuracy, Précision et F1-Score les plus élevées que les autres modèles, avec respectivement 0.756, 0.733 et 0.725. Cependant, le modèle Multi layer perceptron présente une performance similaire avec des valeurs d'Accuracy de 0.741, Précision de 0.710 et F1-Score de 0.696. Le modèle Support Vector Machine et le modèle Vote majorité ont des performances légèrement inférieures avec des valeurs d'Accuracy de 0.741 et 0.750, des valeurs de Précision de 0.712 et 0.726, et des valeurs de F1-Score de 0.679 et 0.702 respectivement.

Si on n'est pas satisfait du modèle, les axes d'améliorations :

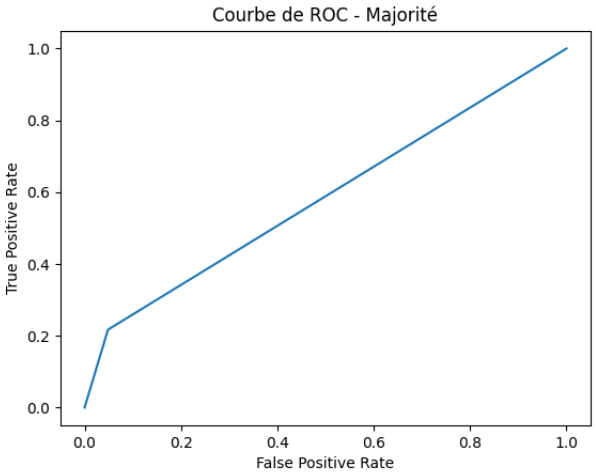
-Réduction de la dimensionnalité (PCA) pour extraire les caractéristiques les plus informatifs, dans ce cas-là on a 7 dimensions et 1 dimension pour la valeur cible.

-Equilibrage de la valeur cible : On a une répartition déséquilibrée entre 0(indemne et blessé léger) et 1 (blessé grave et tué). 0 est largement majoritaire par rapport 1. En équilibrant la répartition des valeurs cibles la performance de prédiction va augmenter.

-Sélection des caractéristiques : On peut identifier les plus informatifs basé sur le modèle ou l'analyse de corrélations.



Random Forest



Vote majorité

On a fait le choix d'afficher seulement deux courbes de ROC car le Vote par majorité est très similaire à SVM et MLP.

Concernent les courbes de roc, elles sont insuffisantes et son légèrement mieux que le cas où x=y. On peut améliorer cette courbe avec la fonction predict_proba qui est plus optimisé le cas de la fonction ROC.

Cas de load out one sans Grid Search

	Accuracy	Precision	Rappel	F1-Score
SVM	0.541	1.0	0.541	0.541
Random Forest	0626	1.0	0.626	0.626
MLP	0.5385	1.0	0.5385	0.5385

On peut observer que le modèle Random Forest a obtenu les valeurs d'Accuracy. Les plus élevées que les autres modèles, avec respectivement 0.626. Quant à SVM et MLP ils ont une performance similaire.

Cependant il est intéressant de se pencher sur la précision de 1 et la redondance de l'Accuracy, le rappel et le F1 score. Cela peut s'expliquer le faible échantillon de données, ici on utilise uniquement les 1000 premiers et il y a un déséquilibre léger pour la valeur cible et déséquilibre très conséquent quant aux valeurs caractéristiques.

Les axes d'améliorations concernant ce modèle :

-Equilibrage des valeurs caractéristiques et sélection des caractéristiques les plus informatifs

- Reduction de la dimensionnalité (PCA) pour extraire les caractéristiques les plus informatifs.

-Augmentation de la taille de l'échantillon, l'intégralité si possible.

-Utilisation GridSearchCV afin comparer les modèles obtenus avec toutes les combinaisons de valeurs pour les hyperparamètres afin d'optimiser et obtenir le meilleur résultat de classification possible.

Scripts

La dernière étape du projet fut la création de scripts bash pour permettre la récupération d'informations provenant de nos différents programmes en .json. Pour cela, nous en avons créé 3 :

- Un premier renvoyant le cluster lié à un point donné

```
.\scripts.sh -m kmean [latitude] [longitude] [centroïdes]
```

- Un deuxième renvoyant la prédiction de la gravité, basé sur l'algorithme KNN

```
.\scripts.sh -m knn [info accident] [CSV]
```

- Un troisième renvoyant la prédiction de la gravité basée sur un algorithme haut-niveau

```
.\scripts.sh -m classification [info accident] [méthode]
```