

**PROJET IA** : **Analyse et Modélisation des Comportements de Navigation des Navires à partir des Données AIS**

**Groupe 1 : Antoine RIOM, Luca VANJEK, Alban LE MAIGAT**

**Nombre de mots : 2706**

**Date rendu : 20/06/25**

# DEROULE DU PROJET IA

1. Préparation des données

2. Apprentissage non-supervisé

3. Apprentissage supervisé

4. Métriques

5. Création de scripts utilisables en ligne de commande

# PARTIE 1 : Alban Le Maigat ,Client 1

**Préparation des données**

Dans cette première phase du projet, l’objectif était de préparer les données de manière à faciliter leur exploitation par les modèles d’intelligence artificielle. La base de données initiale comportait environ 390 000 lignes, représentant les positions enregistrées à différents instants pour un grand nombre de navires. Ce volume important a rapidement posé des problèmes de temps de calcul pour les algorithmes de traitement. Afin de contourner cette contrainte, nous avons choisi de réduire l’échelle du jeu de données en nous concentrant sur 150 navires distincts. Pour chacun d’eux, nous avons conservé uniquement les statistiques associées, à savoir la moyenne et l’écart-type des coordonnées et des paramètres de navigation (latitude, longitude, vitesse et cap). Cette approche permet d’obtenir un jeu de données condensé de 150 lignes, représentatif des comportements de navigation, tout en réduisant fortement les temps de traitement et en préservant le caractère représentatif de la base de données d’origine.

**Apprentissage non-supervisé**

Dans la suite du projet, nous avons cherché à regrouper les navires en fonction de leurs caractéristiques de navigation (cap, vitesse, position, etc.) en appliquant des techniques de clustering non supervisé. Parmi les différentes méthodes possibles, nous avons choisi de nous concentrer sur l’algorithme KMeans, qui repose sur le principe de la partition des données en K groupes distincts, de manière à minimiser la distance intra-cluster (entre chaque point et le centre de son groupe).

Une fois la méthode définie, il est essentiel de choisir le nombre optimal de clusters (K) afin d’obtenir un regroupement pertinent et représentatif. Pour cela, nous avons utilisé la méthode du coude (Elbow method). Cette approche consiste à entraîner le modèle avec différents nombres de clusters croissants et à observer l’évolution de l’inertie, c’est-à-dire la somme des distances entre les points et leur centre de cluster. Le "coude" de la courbe indique le point à partir duquel augmenter K n’apporte plus d’amélioration significative, ce qui permet d’estimer visuellement la valeur optimale de K.

On obtient donc ce graphique :

Une image contenant texte, ligne, Tracé, diagramme

Le contenu généré par l’IA peut être incorrect.

Figure 1 : Diagramme "coude" pour la recherche du nombre optimal de clusters

D’après la méthode du coude, le nombre optimal de clusters correspond au point à partir duquel la diminution de l’inertie devient moins significative. Dans notre cas, on observe que la courbe commence à s’infléchir autour de k = 5, ce qui en fait un bon candidat pour le nombre de clusters à utiliser avec l’algorithme KMeans.

**Métriques pour l’apprentissage non-supervisé**

Pour évaluer l’efficacité de notre méthode de clustering, nous avons utilisé plusieurs métriques d’évaluation internes, qui permettent de juger la qualité du regroupement sans disposer de labels préexistants.

* Le Silhouette Score mesure la cohésion d’un point avec son propre cluster comparée à sa séparation vis-à-vis des autres clusters. Il varie entre -1 et 1 : plus il est proche de 1, plus le cluster est bien défini. Un score élevé traduit donc une bonne densité.
* Le Calinski-Harabasz Index (aussi appelé "Variance Ratio Criterion") quantifie le rapport entre la dispersion inter-clusters et la dispersion intra-cluster. Plus cette valeur est élevée, plus les clusters sont denses et bien séparés.
* Enfin, le Davies-Bouldin Score mesure la similarité entre les clusters : une valeur faible indique que les clusters sont bien séparés les uns des autres. Contrairement aux deux scores précédents, ici plus la valeur est basse, meilleure est la qualité du clustering.

L’utilisation conjointe de ces trois métriques nous permet d’avoir une évaluation plus robuste et fiable du comportement de l’algorithme KMeans pour différents nombres de clusters.

Pour des apprentissages KMeans entre 2 et 11 clusters on obtient les scores suivant :

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

Le contenu généré par l’IA peut être incorrect.

Figure 2 : Scores des méthodes KMeans pour différents nombres de clusters ( k )

Cela confirme nos résultats de la méthode du coude, 5 clusters semble être le paramètre idéal car il possède les meilleurs scores d’évaluation du clustering. On remarque également que les scores évoluent de manière similaires que sur la méthode du coude.

**Visualisation sur une carte**

Une fois le clustering effectué avec les bons hyperparamètres, il est nécessaire d’en faire l’affichage pour mieux se rendre compte des regroupements effectués. Pour ce faire nous avons affiché les bateaux sur une carte du Golfe du Mexique.

**Une image contenant carte, texte, atlas

Le contenu généré par l’IA peut être incorrect.**

Figure 3 : Visualisation des clusters regroupant les bateaux aux données de naviguation similaires

Le clustering des trajectoires à l’aide de l’algorithme KMeans (k = 5) a permis de regrouper les navires selon leurs comportements de navigation. La carte révèle des zones géographiques bien distinctes, notamment dans le golfe du Mexique et autour de la Floride, correspondant à des schémas de déplacement. Certains clusters représentent des trajectoires longues et rectilignes, cohérent avec le commerce maritime, tandis que d’autres regroupent des mouvements plus concentrés, liés à des activités locales comme la pêche ou le cabotage. Bien qu’il y ai des chevauchements entre certaines zones, le modèle offre une segmentation globalement cohérente. Ce résultat valide l’intérêt du clustering non supervisé pour identifier des profils de navigation à partir de données.

# PARTIE 2 : Luca Vanjek, Client 2

**Préparation des données**

Avant d'entraîner le modèle de classification, un important travail de préparation des données a été réalisé. Le jeu de données initial contenait de nombreuses informations sur les navires, dont certaines inutilisables ou manquantes. Nous avons sélectionné uniquement les variables pertinentes pour la prédiction du type de navire : la vitesse sur le sol (sog), le cap (cog), l’orientation (heading), ainsi que les dimensions du navire (length, width, draft).

Les lignes contenant des valeurs manquantes ont été supprimées afin de garantir un apprentissage propre du modèle. La variable cible vessel\_type, qui représente le type de navire à prédire, a été encodée numériquement pour être compatible avec les algorithmes de classification.

Pour éviter tout biais et assurer une évaluation fiable du modèle, le jeu de données a été divisé en deux ensembles distincts : un ensemble d'entraînement (80 %) et un ensemble de test (20 %), en veillant à ce que la distribution des types de navires soit respectée dans chaque groupe.

Enfin en amont de cette division les données ont été triées par mmsi afin de ne pas retrouver des données similaires dans la base d’entrainement et celle de test.

**Apprentissage supervisé**

L'apprentissage supervisé repose sur l'utilisation de données étiquetées pour entraîner un modèle à prédire des résultats à partir de caractéristiques. Dans ce projet, plusieurs approches ont été appliquées, adaptées aux spécificités des données et aux objectifs d'analyse.

Régression Logistique :  
La régression logistique est un modèle linéaire utilisé pour la classification. Elle tente de séparer les classes en traçant une frontière linéaire dans l'espace des caractéristiques. Ce modèle est simple et interprétable, mais ses performances peuvent être limitées si les relations entre les variables ne sont pas linéaires.

Résultats : Les métriques montrent que la régression logistique offre une précision globale modeste (66%), avec des performances variables entre les classes. Cela indique qu'il s'agit d'un modèle basique, performant pour certaines classes mais peu adapté à des distributions plus complexes.

Random Forest (Approche Complète) :

Random Forest est un ensemble d'arbres de décision, chaque arbre étant entraîné sur un sous-ensemble des données. Il est robuste, flexible, et capable de capturer des relations complexes entre les caractéristiques. Dans cette approche, toutes les observations (multiples lignes par mmsi) ont été utilisées.

Résultats : Le modèle affiche une performance parfaite (100% sur toutes les métriques), indiquant un potentiel surapprentissage des données. Cela peut suggérer que le modèle s'est trop ajusté aux données d'entraînement, rendant sa généralisation douteuse.

Random Forest (Une Ligne par MMSI) :

Pour éviter les biais introduits par des répétitions de données par navire, une variante a été entraînée sur un ensemble où chaque navire (mmsi) est représenté une seule fois. Cette approche simplifie la structure des données et réduit le risque de surapprentissage.

Résultats : Le modèle atteint une précision globale de 59%. Bien que les performances soient moins extrêmes que dans l’approche complète, elles offrent une représentation plus réaliste des capacités du modèle.

**Métriques**

Les performances des modèles sont évaluées sur plusieurs classes (vessel\_type) à l’aide des métriques suivantes : précision, rappel, F1-score, et accuracy. Voici une description des résultats obtenus pour chaque méthode.

Régression Logistique

La régression logistique donne des résultats globaux modestes, avec une accuracy globale de 66%. Voici les points principaux concernant ses performances par classe :

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

Le contenu généré par l’IA peut être incorrect.

La classe 60 atteint une précision de 0.72, un rappel de 0.92, et un F1-score de 0.81, ce qui reflète de bonnes performances, tandis que la classe 70 affiche des performances modérées, avec une précision de 0.58, un rappel de 0.50, et un F1-score de 0.54.

Globalement, le macro average est correct (environ 0.7) ce qui montre une performance égale entre les classes. Le weighted average (pondéré par le support des classes) quant à lui est le même à chaque fois (0.66).

Random Forest (Ensemble Complet)

Ce modèle affiche des performances parfaites (100%) sur toutes les métriques, ce qui est inhabituel et soulève des questions sur un possible surapprentissage dû à l’utilisation de l’ensemble complet. Tous les scores (précision, rappel, F1-score) sont à 1.00, et l'accuracy globale est également 100%.

Bien que ces résultats semblent idéaux, ils indiquent que le modèle s'est probablement ajusté à la structure des données d’entraînement (incluant les doublons liés au mmsi), rendant sa généralisation douteuse pour de nouvelles données.

Random Forest (Une Ligne par MMSI)

Lorsque chaque navire (mmsi) est représenté une seule fois, les performances sont plus réalistes, avec une accuracy globale de 59%. Les performances varient selon les classes :

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

Le contenu généré par l’IA peut être incorrect.

La classe 60 atteint des bons scores (précision : 1.00, rappel : 0.67, F1-score : 0.80), mais cela peut être lié à son faible support (3 exemples dans l'ensemble de test).

La classe 70 obtient une précision de 0.67, un rappel de 0.22, et un F1-score de 0.33, ce qui reflète des performances peu solides.

La classe 80 est globalement bien prédite (précision : 0.53, rappel : 0.90, F1-score : 0.67).

Globalement, le macro average est modéré (précision : 0.73, rappel : 0.60, F1-score : 0.60), ce qui montre une performance inégale entre les classes. Le weighted average est équivalent (précision : 0.65, rappel : 0.59, F1-score : 0.55), grâce aux classes majoritaires bien prédites.

| Méthode | Précision Globale | Macro Avg (F1-score) | Weighted Avg (F1-score) |
| --- | --- | --- | --- |
| Régression Logistique | 66% | 0.70 | 0.66 |
| Random Forest (Complet) | 100% | 1.00 | 1.00 |
| Random Forest (Par MMSI) | 59% | 0.60 | 0.55 |

Le Random Forest par mmsi est la méthode la plus faible, performante uniquement pour certaines classes.

Le Random Forest complet affiche des résultats parfaits mais est sujet à un fort surapprentissage.

Le Régression Logistique équilibre mieux les performances et offre une meilleure généralisation, malgré des lacunes sur les classes peu représentées.

**Prépartion d’un script python**

Le script Python est conçu pour charger un modèle préalablement entraîné (fichier .pkl) et effectuer des prédictions sur le type de navire à partir de nouvelles données fournies directement dans le terminal. Voici les étapes principales du script :

Chargement du Modèle :  
Le fichier contenant le modèle sauvegardé (model\_vessel\_type.pkl) est chargé à l'aide de la bibliothèque joblib. Ce modèle inclut toutes les étapes nécessaires, comme le prétraitement des données et la prédiction.

Entrée des Données :  
L'utilisateur peut saisir les caractéristiques d'un navire (comme sog, cog, heading, length, width, et draft) en ligne de commande. Ces données représentent les mesures observées pour le navire.

Prédiction :  
Les données saisies sont automatiquement normalisées et transformées selon le prétraitement intégré dans le modèle. Le script utilise ensuite le modèle pour prédire le type de navire (vessel\_type) correspondant.

Affichage du Résultat :  
Une fois la prédiction effectuée, le type de navire prédit est affiché directement dans le terminal, permettant une utilisation rapide et pratique.



Ce script est une solution simple et efficace pour exploiter le modèle de classification dans un environnement réel sans nécessiter de connaissances avancées en programmation.

# PARTIE 3 : Client 3

Avant d’entamer l’analyse et l’entraînement, nous avons d’abord structuré et nettoyé le jeu de données à l’aide d’un script R afin d’offrir au modèle un socle fiable et cohérent.

Un obstacle est vite apparu : les positions des navires étaient relevées à des intervalles irréguliers, rendant impossible un entraînement basé sur un pas de temps constant. Nous avons donc évalué plusieurs stratégies :

1. Filtrer par intervalles fixes (5 min, 10 min, 15 min)

En n’utilisant que ces écarts, nous aurions dû entraîner trois modèles distincts, mais la perte de données était trop importante (176 points à 5 min, 60 à 10 min et 209 à 15 min).

1. Interpoler un point toutes les 5 minutes

Si cette méthode homogénéisait la série, sa mise en œuvre se révélait excessivement lourde et complexe.

1. Intégrer le “delta t” comme variable explicative

Solution finalement retenue : nous ajoutons, pour chaque observation, le délai séparant la position courante de la suivante. Le modèle apprend ainsi à prédire n’importe quelle position en fonction du delta t fourni. La précision diminue toutefois lorsque cet intervalle s’allonge.

Cette dernière approche concilie flexibilité et volume de données, tout en restant techniquement maîtrisable pour la suite du projet.

Ainsi, le modèle reçoit en entrée :

* lat, lon : latitude et longitude actuelles
* sog : vitesse surface (Speed Over Ground)
* cog : cap vrai (Course Over Ground)
* heading : direction du navire
* length : longueur du navire
* draft : tirant d’eau
* delta\_seconds : intervalle de temps jusqu’à la prochaine observation

Il produit en sortie :

* lat\_next, lon\_next : latitude et longitude prévues pour l’instant suivant.

**Apprentissage supervisé :**

Pour prédire au mieux les positions futures des navires, nous avons privilégié l’apprentissage supervisé : les coordonnées à estimer sont des valeurs continues, la régression s’impose donc naturellement face à la classification. Parmi les nombreuses techniques possibles, deux approches se distinguent et seront mises en concurrence :

Régression linéaire : solution de référence, rapide à entraîner, complètement explicable grâce à ses coefficients. Elle fixe notre seuil de performance minimal, mais sa linéarité limite la prise en compte d’interactions complexes.

Random Forest Regressor : ensemble d’arbres de décision entraînés en bagging, capable de modéliser des relations non linéaires et de résister au sur-apprentissage. Il exige cependant davantage de calcul et son interprétation globale requiert des outils spécifiques.

**Quelle régression choisir ?**

Pour éclairer notre décision, nous avons d’abord examiné les indicateurs globaux :

Régression linéaire

* RMSE : 0 .0669
* MAE : 0 .0031
* R² : 0 .9992

Random Forest Regressor

* RMSE : 0 .1685
* MAE : 0 .0489
* R² : 0 .9931

Nous avons ensuite détaillé les performances selon la longueur de l’intervalle de temps:

Régression linéaire

* Petit (< 5 min) : RMSE = 0 .0019
* Moyen (5–20 min) : RMSE = 0 .0106
* Grand (> 20 min) : RMSE = 0 .9056

Random Forest Regressor

* Petit (< 5 min) : RMSE = 0 .0694
* Moyen (5–20 min) : RMSE = 0 .1019
* Grand (> 20 min) : RMSE = 0 .5330

On en déduit que :

Pour des horizons courts à moyens (jusqu’à 20 minutes), la régression linéaire domine nettement : ses erreurs sont anecdotiques et son R² frôle la perfection. Elle reste donc la solution la plus précise, la plus légère et la plus interprétable pour un déploiement temps réel. Au-delà de 20 minutes, la Random Forest se montre moins sensible à la dérive : son RMSE chute d’environ 40 % par rapport à la régression linéaire, signe qu’elle capture mieux les non-linéarités à long terme.

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

Le contenu généré par l’IA peut être incorrect.Nous avons aussi placé les positions prédites dans un graphique avec la longitude ou latitude prédite en fonction de la latitude ou de la longitude réel.

Une image contenant ligne, Tracé, diagramme, pente

Le contenu généré par l’IA peut être incorrect.

Le second graphique (régression linéaire) suit presque parfaitement la droite rouge ; les points restent serrés, ce qui traduit des résidus faibles et homogènes sur l’ensemble du jeu de données. Les deux nuages « vérité vs prédiction » confirment les métriques chiffrées : le premier graphique (Random Forest) montre une dispersion plus large autour de la diagonale, surtout pour certaines zones de latitudes et longitudes, signe d’un biais et d’une variance plus élevés

**Représentation sur des cartes**

Au cours de l’analyse, nous avons constaté que les relevés de position présentaient des intervalles très irréguliers, parfois minimes, parfois très étendus. Pour améliorer la continuité des trajectoires, nous avons décidé de combler ces « trous » à l’aide de nos modèles de régression.

Dans un premier temps, nous avons tenté de générer des observations toutes les cinq minutes dès qu’un écart dépassait dix minutes. Cette approche s’est révélée peu concluante : qu’il s’agisse de la régression linéaire ou du Random Forest Regressor, les prédictions dérivaient fortement dès que l’intervalle à reconstituer devenait trop long. Le problème s’accentuait sur les lacunes supérieures à vingt-quatre heures, domaine pour lequel le modèle manquait clairement de données d’entraînement.

Pour limiter ces dérives, nous avons donc fixé une borne supérieure : nous ne comblons désormais que les intervalles compris entre dix minutes et une heure. Cette contrainte réduit la difficulté de la tâche pour le modèle, garantit une cohérence spatio-temporelle satisfaisante et évite les extrapolations hasardeuses sur de très longues

Nous avons fait cette reconstitution sur les deux modèles :

Une image contenant texte, carte

Le contenu généré par l’IA peut être incorrect.

Figure 4 reconstitution avec Random Forest Regressor

Une image contenant texte, carte

Le contenu généré par l’IA peut être incorrect.

Figure 5 Reconstitution avec Linear regretion

Sur ces cartes, chaque point bleu correspond à une position réellement mesurée, tandis que chaque point rouge représente une position reconstruite par le modèle.

Sur la première carte, celle obtenue avec le Random Forest Regressor, on distingue deux comportements. D’un côté, certains points rouges collent parfaitement au couloir formé par les points bleus : le modèle sait extrapoler quand les relevés sont rapprochés. De l’autre, dès que l’on s’éloigne d’une mesure de référence, les prédictions se dispersent. Au centre du golfe, on observe même des paquets rouges totalement hors trajectoire ; sur la partie droite de la carte, la trace devient franchement brouillonne.

La deuxième carte illustre la reconstruction effectuée par la régression linéaire. Ici, les points rouges restent quasiment confondus avec la route bleue, sans dérive latérale visible. Les rares écarts sont de faible amplitude et se résorbent rapidement. Visuellement, la cohérence spatiale confirme les métriques : lorsque l’intervalle à combler reste inférieur à une heure, la régression linéaire maintient une précision très élevée.

En somme, si le Random Forest peut parfois bien prédire, il introduit aussi des sauts et des zigzags indésirables dès que le delta t s’allonge. La régression linéaire, à l’inverse, fournit des interpolations beaucoup plus régulières et fiables sur la plage 10 minutes – 1 heure. Le modèle de régression linéaire est donc le plus performant.

# Organisation

Une image contenant capture d’écran, ligne, texte, Caractère coloré

Le contenu généré par l’IA peut être incorrect.

Figure 6 : Diagramme de Gantt

# Conclusion

Nous avons conçu un pipeline complet qui transforme des relevés AIS bruts en informations directement exploitables : regroupement de navires par comportement, identification automatique de leur type et projection fiable de leur trajectoire. Un modèle simple et interprétable s’est avéré suffisant pour les prévisions à court terme, tandis que des algorithmes plus puissants prennent le relais lorsque les intervalles s’allongent. Les scripts livrés montrent que la solution est déjà prête pour un usage opérationnel, tout en restant évolutive vers des méthodes plus sophistiquées à mesure que les besoins grandiront.