Paul LE GOFF **A3 - G1**
Gauthier GLOANEC **A3 - G1**
Pol Nerisson **A3 - G1**

Rapport de projet IA

10/06/2025
13/06/2025

ISEN
ALL IS DIGITAL!
BREST

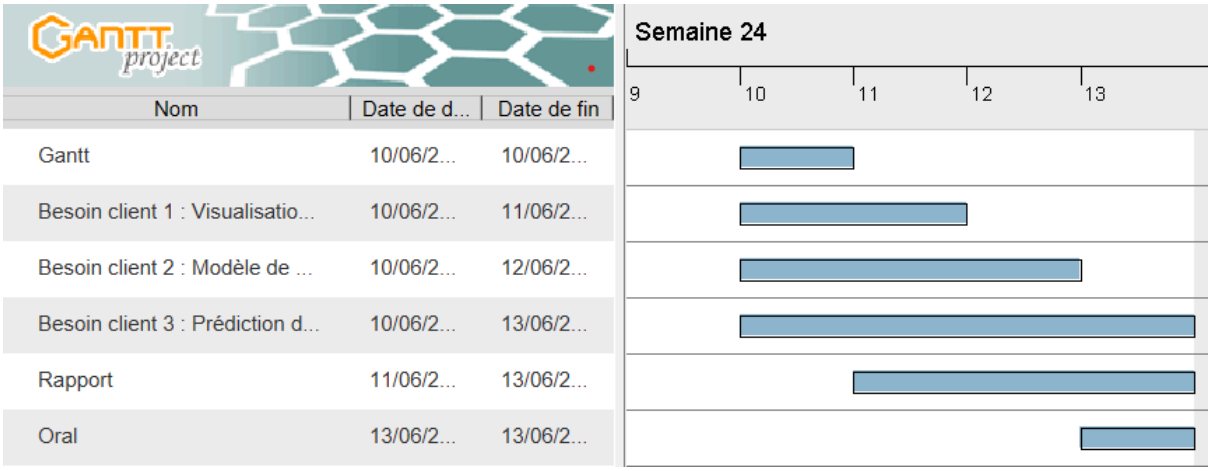
Préambule :

À la suite du projet Big Data consacré à l'analyse des trajectoires maritimes à partir des données AIS, nous avons entamé un second volet centré sur l'intelligence artificielle. Ce nouveau projet a pour objectif de tirer parti des résultats précédemment obtenus afin de développer des modèles prédictifs exploitables dans un cadre opérationnel.

En s'appuyant sur les caractéristiques techniques des navires et leurs trajectoires historiques, nous avons conçu plusieurs outils de prédiction. Ceux-ci permettent notamment d'estimer la trajectoire future d'un navire, de déterminer son appartenance à un cluster comportemental et d'anticiper son type à partir de ses données de navigation.

Sommaire :

Diagramme de gantt



Gantt : Gauthier Gloanec
Besoin 1 : Gauthier Gloanec
Besoin 2 : Pol Nerrison
Besoin 3 : Paul Le Goff
Rapport : Tous
Oral : Tous

1 - Besoin client 1 : Visualisation sur carte

Préparation et traitement des données

Les données utilisées pour cette analyse sont extraites du fichier CSV intitulé `"After_Sort.csv"`. Celui-ci contient des informations relatives à la position et aux caractéristiques techniques de divers navires. Afin de préparer ces données en vue d'une application de techniques de clustering, plusieurs étapes préliminaires ont été nécessaires. Tout d'abord, seules les colonnes pertinentes ont été conservées : latitude (`LAT`), longitude (`LON`), vitesse sur le fond (`SOG`), cap sur le fond (`COG`), longueur (`Length`), largeur (`Width`), tirant d'eau (`Draft`), direction (`Heading`) et type de navire (`VesselType`).

Dans un souci de rigueur et pour garantir la robustesse du traitement, les lignes contenant des valeurs manquantes ont été systématiquement supprimées. La variable `VesselType`, initialement catégorielle, a été encodée sous forme numérique grâce à la méthode `LabelEncoder`, rendant ainsi la variable compatible avec les algorithmes d'apprentissage automatique. Afin d'uniformiser les échelles des variables, et éviter qu'une variable à forte amplitude n'influence de manière disproportionnée la formation des groupes, les données ont été normalisées à l'aide du `StandardScaler`.

Par la suite, une réduction de la dimensionnalité a été réalisée avec l'algorithme `PCA` (Analyse en Composantes Principales), réduisant l'espace des données à cinq composantes principales. Cette transformation permet d'optimiser les performances du clustering tout en conservant l'essentiel de l'information : la variance expliquée par ces cinq composantes dépasse les 90 %, ce qui atteste de leur représentativité.

Clustering des navires

Pour effectuer le regroupement des navires selon leurs comportements et caractéristiques, un algorithme de clustering non supervisé de type `KMeans` a été utilisé. Le nombre de clusters a été fixé à trois, ce choix étant guidé à la fois par l'expérimentation et l'analyse des métriques de qualité. L'entraînement du modèle a été réalisé sur les données préalablement normalisées et réduites. Chaque navire s'est ainsi vu attribuer une étiquette de cluster, stockée dans une nouvelle colonne du jeu de données principal.

Une évaluation rigoureuse des résultats a été menée à l'aide de trois indicateurs standards : le Silhouette Score, le Calinski-Harabasz Index et le Davies-Bouldin Index. Pour ces calculs, un sous-échantillon aléatoire de 10 000 observations a été utilisé afin de maintenir un bon compromis entre précision et performance. Les résultats obtenus ont confirmé la cohérence des clusters formés, avec un bon équilibre entre compacité intra-cluster et séparation inter-cluster.

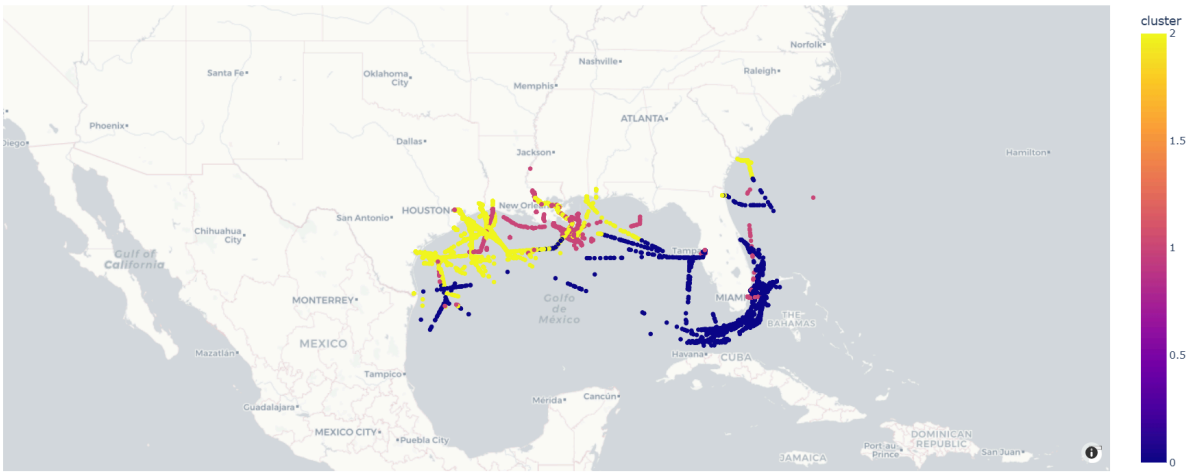
Dans le but d'assurer la réutilisabilité du modèle, l'algorithme `KMeans`, le `StandardScaler`, le `LabelEncoder` et le modèle PCA ont été sauvegardés localement sous forme de fichiers `.joblib`. Ce mécanisme permet une réutilisation directe sans nécessiter de recalcul des clusters à chaque lancement de l'application.

Visualisation cartographique des clusters

La visualisation des résultats du clustering a été réalisée à l'aide de la bibliothèque `Plotly Express`, qui permet une représentation interactive des points géolocalisés. Afin de préserver la fluidité de l'affichage tout en assurant une représentativité suffisante, un échantillon de 10 000 points a été extrait de l'ensemble du jeu de données. Chaque navire est représenté sur la carte selon sa position géographique (latitude et longitude) et coloré en fonction de son appartenance à un cluster.

Des métadonnées complémentaires, telles que la vitesse, le cap, la direction, les dimensions et le type du navire, sont accessibles au survol, permettant ainsi une lecture fine et intuitive des regroupements. L'objectif de cette représentation est de faire apparaître visuellement les patterns géospatiaux propres à chaque groupe, que ce soit des trajectoires similaires, des zones de concentration ou encore des comportements de navigation distincts.

Clustering des navires sur la carte (toute la base)



Prédiction du cluster d'un nouveau navire

Un script de prédiction a été développé pour permettre l'affectation automatique d'un nouveau navire à l'un des clusters existants. Ce script repose sur la chaîne complète de traitement précédemment sauvegardée. Lorsqu'un dictionnaire de caractéristiques est fourni (latitude, longitude, vitesse, cap, dimensions, direction et type de navire), le script charge les objets de traitement (`Scaler`, `PCA`, `KMeans`, `LabelEncoder`), applique les mêmes transformations que lors de l'entraînement, puis prédit l'étiquette de cluster correspondante.

Un contrôle est prévu pour vérifier que la valeur du type de navire est bien connue du modèle. Dans le cas contraire, une erreur informative est retournée à l'utilisateur. Cette fonctionnalité garantit la cohérence de l'usage en phase d'exploitation.

Analyse des clusters

Enfin, une synthèse des caractéristiques moyennes de chaque cluster a été réalisée. Pour cela, les moyennes des variables principales (latitude, longitude, vitesse, cap, dimensions, direction et type de navire) ont été calculées pour chaque groupe. Cette analyse statistique permet d'interpréter les regroupements obtenus en mettant en évidence les spécificités de chaque cluster. Par exemple, un groupe peut correspondre à des navires plus lents et massifs, un autre à des navires plus petits et rapides, ou encore à des zones de navigation spécifiques.

Cette étape d'analyse contribue à donner du sens aux résultats du clustering et à orienter les interprétations futures dans le cadre de décisions logistiques, de gestion de trafic maritime ou d'optimisation de routes.

2 - Besoin Client 2 : Prédiction du type de navire.

```
--- Regression ---

=== Régression VT 60 ===
LinReg | RMSE 28.52 | MAE 16.86 | R²=0.00
RF     | RMSE  4.67 | MAE  3.02 | R²=0.97
GBR    | RMSE  4.67 | MAE  3.03 | R²=0.97

=== Régression VT 61 ===
LinReg | RMSE  8.50 | MAE  8.31 | R²=-0.00
RF     | RMSE  8.49 | MAE  8.31 | R²=-0.00
GBR    | RMSE  8.50 | MAE  8.31 | R²=-0.00
❗ VT 62: jeux insuffisant (0)
❗ VT 63: jeux insuffisant (0)
❗ VT 64: jeux insuffisant (0)
❗ VT 65: jeux insuffisant (0)
❗ VT 66: jeux insuffisant (0)
❗ VT 67: jeux insuffisant (0)
❗ VT 68: jeux insuffisant (0)

=== Régression VT 69 ===
LinReg | RMSE  1.21 | MAE  0.38 | R²=0.64
RF     | RMSE  1.21 | MAE  0.38 | R²=0.64
...
LinReg | RMSE  4.54 | MAE  3.82 | R²=0.15
RF     | RMSE  4.33 | MAE  3.48 | R²=0.23
GBR    | RMSE  4.33 | MAE  3.48 | R²=0.23
```

La régression nous montre que les jeux de donnée est inégale et même si nous avons des données, certaines classes seront négligées surtout les passengers visiblement.

Classification 1,SGD « régression logistique pénalisée »

Le premier modèle que nous avons fait est une régression logistique entraînée en Stochastic Gradient Descent.

Le principe, on lit les observations (ou de petits mini-lots) l'une après l'autre ; à chaque passage, on met à jour les poids du modèle selon le gradient de la fonction-coût log_loss (entropie croisée multiclasse). Cette stratégie « stochastique » limite l'emprunt de mémoire et supporte aisément plusieurs millions de lignes.

Pour éviter le sur-apprentissage, Régularisation Elastic-Net – un mélange de L1 (qui pousse certains poids à 0 et réalise ainsi une sélection de variables) et L2 (qui stabilise la solution).

Le coefficient global est α ; s'il est trop faible, le pas d'apprentissage optimal devient trop grand et l'algorithme diverge, s'il est trop fort le modèle sous-apprend.

early_stopping – arrêt automatique quand le score se stabilise sur un petit jeu de validation interne.

Comme une frontière purement linéaire est trop pauvre pour séparer nos types de navires, nous enrichissons l'espace avec des polynômes de degré 2 (carrés + produits croisés). Une simple hyperplan dans cet espace étendu se traduit par des frontières courbes dans l'espace d'origine. Les hyper-paramètres (α et l1_ratio) sont ajustés par GridSearchCV : pour chaque couple (α, l1_ratio) on effectue une validation croisée GroupKFold (les folds contiennent des MMSI distincts, donc impossible qu'un même navire se retrouve à la fois en apprentissage et en test).

La métrique d'optimisation est le F1-macro, plus exigeant que l'accuracy dès que les classes sont déséquilibrées. Malgré ces précautions, la frontière reste essentiellement linéaire ; le modèle est ultrarapide, mais plafonne autour de 0.55 F1-macro.

SMOTE (Synthetic Minority Over-sampling Technique)

Pour rééquilibrer les classes avant d'entraîner des modèles plus complexes, nous employons SMOTE : Pour chaque point minoritaire A, on choisit aléatoirement l'un de ses k plus proches voisins minoritaires B.

On génère un nouveau point A′ par interpolation sur le segment AB. On répète jusqu'à ce que la classe atteigne le pourcentage souhaité.

SMOTE n'invente pas des valeurs catégorielles et ne copie pas les majoritaires : il densifie uniquement la zone minoritaire dans l'espace numérique.

Attention : la méthode exige que le tableau soit entièrement numérique et sans NaN ; d'où notre passage préalable par une imputation ou un filtrage des lignes incomplètes.

Classification 2 : ExtraTrees + RandomizedSearchCV

Après équilibrage, nous formons une forêt d'arbres extrêmement aléatoires (ExtraTrees). Différences clés avec une Random Forest classique : Au lieu de tester toutes les coupures possibles, chaque nœud reçoit un sous-ensemble totalement aléatoire de variables et de seuils, puis retient le meilleur.

Moins de corrélation entre arbres donne une variance plus faible donc un apprentissage plus rapide.

Pour améliorer la lisibilité des données, nous devons faire plusieurs choses, par exemple, une imputation de la médiane, tout en méttant un marqueur pour ne pas perdre l'information, nous uniformisons pour stabiliser le borderline-SMOTE qui densifie la zone la plus proche de la frontiere de decision. (l'hypersurface qui permet de definir la limite entre les classes que l'on recherche)

ExtraTreesClassifier avec class_weight="balanced". Pour régler les hyper-paramètres (nombre d'arbres, profondeur max, max_features, etc.) nous utilisons RandomizedSearchCV : on échantillonne 25 combinaisons au hasard dans l'espace défini.

Avantages : couvrir un domaine beaucoup plus large qu'un GridSearch exhaustif, en un temps très raisonnable et facilement parallélisable (n_jobs=-1). Validation : GroupKFold 3 folds (séparation stricte par MMSI).

Résultat typique : ~ 0.62 F1-macro, un compromis équilibré entre vitesse et performance – bien supérieur au SGD linéaire, mais sans la lourdeur d'un boosting gradients complet.

Classification 3 : CatBoost Gradien Boosting

Pour cette phase, nous avons opté pour CatBoost, un algorithme de gradient boosting basé sur des arbres de décision. Sa petite astuce, appelée « ordered boosting », consiste à entraîner à nouveau chaque nouvel arbre sur une copie des données où la vraie étiquette de la ligne en cours reste cachée jusqu'au dernier moment. Cela permet de limiter sérieusement le sur-apprentissage, un problème qui survient souvent quand le jeu de données n'est pas gigantesque.

Autre point fort : en réglant le paramètre `auto_class_weights` sur « Balanced », CatBoost corrige de lui-même les déséquilibres entre classes. Même si certaines catégories de navires sont rares, l'algorithme leur donne plus de poids et on n'a donc pas besoin de recourir au sur-échantillonnage (type SMOTE) ni de modifier les poids à la main.

Pour les valeurs manquantes, Cat Boost sélectionne, au moment de chaque découpe, de quel côté envoyer les NaN. Pas besoin d'imputer quoi que ce soit comme aux codes précédents ; on garde les données brutes tout en évitant les problèmes que le manque de données peuvent poser.

L'algorithme sait aussi tracer des frontières non linéaires sans qu'on doive créer des dizaines de variables supplémentaires ou des polynômes artificiels. Dans notre pipeline, on se contente de standardiser les variables numériques puis de lancer 500 itérations d'arbres de profondeur 8, avec un taux d'apprentissage de 0,05. La séparation des données se fait toujours par identifiant MMSI grâce à `GroupShuffleSplit`, histoire de ne pas mélanger les bateaux d'un même navire entre apprentissage et test.

Côté résultats, CatBoost tourne autour de 0,75 – 0,80 en F1-macro, ce qui, d'après notre batterie de tests, représente le meilleur compromis entre biais et variance. Surtout, le temps de prédiction est très court : idéal pour suivre le flux AIS quasi en temps réel, et donc pour classer les navires sans ralentir la chaîne de décision.

En résumé, CatBoost gagne la partie parce qu'il sait apprendre des relations complexes sans exiger des heures de préparation, puisqu'il gère directement les NaN et les classes déséquilibrées, et comme son ordered boosting limite le sur-apprentissage. ExtraTrees est rapide, mais moins précis ; SGD reste un bon choix si l'on manque de ressources. Néanmoins, CatBoost nous offre le trio gagnant : précision, robustesse et vitesse, le tout dans un pipeline très léger.

Voilà pourquoi Catboost est la meilleure option pour répondre au besoin du client 2.

3 - Besoin Client 3 : Prédiction des trajectoires des bateaux

Objectif client 3:

Cette partie du projet a pour but de créer un système capable de prédire la trajectoire des navires grâce à des techniques d'apprentissage automatique. L'objectif est d'estimer la position d'un navire dans 5, 10 et 15 minutes en utilisant sa position actuelle et ses données de navigation. Il s'agit d'un apprentissage supervisé, car le modèle apprend à partir de données d'entrées associées à des données de sorties connues.

Préparation du dataset:

Préparation du dataset: On a 7 nouvelles colonnes :
LAT_5, LONG_5 : Où sera le navire dans 5 minutes
LAT_10, LONG_10 : Où sera le navire dans 10 minutes
LAT_15, LONG_15 : Où sera le navire dans 15 minutes
Delta_temps :Temps écoulé entre le signal précédent et le signal actuel pour un même navire.

Création des données d'entraînement:

Les navires n'envoient pas leurs signaux à des intervalles réguliers. Pour contourner ce problème, tous les signaux reçus entre 3 et 7 minutes après une position de référence sont considérés comme représentant la position du navire à +5 minutes. Il s'agit ici d'une fenêtre de tolérance de +/- 2 minutes.

Division des données:

Division des données en 3 bases :
Base d'apprentissage: 65% - 97 bateaux
Base de test: 20% - 30 bateaux
Base de validation: 15% - 22 bateaux

Aucun navire ne doit apparaître dans plusieurs ensembles de données afin d'éviter que le modèle n'apprenne des trajets spécifiques de chaque bateau, ce qui fausserait le test sur de nouveaux navires.

Les modèles

On utilise 6 modèles différents : 3 pour prédire la longitude (à 5, 10 et 15 min) et 3 pour prédire la latitude (à 5, 10 et 15 min).

Pour l'entrainement du modèle, les variables explicatives (features) sont:
LAT ou LON : Position actuelle
SOG : Vitesse du navire
COG : Direction de déplacement
Heading : Orientation de la coque

Variables dépendantes (à prédire)

LAT ou LON à 5, 10 et 15 min

On mesure la qualité de nos prédictions avec le RMSE (Racine Carrée de l'Erreur Quadratique Moyenne) est une mesure qui indique l'erreur moyenne de prédiction d'un modèle. Plus il est grand, plus l'erreur est grande.

Conclusion

Ce projet nous a permis d'améliorer nos compétences en intelligence artificielle. On a appris à créer des datasets, à entraîner des modèles et faire des prédictions.
La partie 3 (prédiction des trajectoires) était la plus difficile du projet. Il fallait gérer beaucoup de problèmes : les signaux à intervalles de temps irréguliers, la création des bonnes données d'entraînement, et l'optimisation des modèles.