PROJET ANNÉE 3

Big Data / IA / Web

Partie Intelligence Artificielle

ANALYSE ET MODÉLISATION DES COMPORTEMENTS DE NAVIGATION DES NAVIRES À PARTIR DES DONNÉES AIS

Trinôme : 5

Membres du groupe :

- JAUFFRIT Nolan
- NEDELEC Nolan
- BOSSER Célian

Formation : Cycle ingénieur année 3

Établissement : ISEN Brest

Année académique : 2024-2025

Date de remise : 13 juin 2025

Enseignant référent : SEDGH GOOYA Eshan

Sommaire

Introduction

L'exploitation des données AIS (Automatic Identification System) représente un enjeu stratégique pour l'industrie maritime moderne. Ces informations en temps réel transmises par les navires constituent une ressource précieuse pour optimiser la navigation et améliorer la sécurité maritime.

Ce projet d'intelligence artificielle répond à trois besoins clients complémentaires : développer une segmentation automatique des trajectoires pour identifier les schémas de navigation, créer un modèle de classification pour prédire le type des navires, et anticiper leurs positions futures.

Pour relever ces défis, nous mobilisons l'ensemble des techniques d'apprentissage automatique : clustering non supervisé, classification supervisée et régression temporelle. Cette approche méthodologique permet d'explorer les différentes facettes de l'analyse prédictive maritime.

Besoin Client 1 :

1. Contexte et objectif

L’objectif de la première phase, appelée "Besoin Client 1", était de réaliser une analyse exploratoire et une segmentation automatique de trajectoires de navires en mer, afin d’identifier des groupes de comportements de navigation similaires. Cela permet au client de mieux comprendre la typologie des trajectoires, de détecter des anomalies, ou encore d’optimiser certaines routes maritimes.

2. Préparation des données

Le fichier de travail principal, df_clean.csv, contient une base de données nettoyée comprenant des informations de positionnement, de mouvement, et de type pour plusieurs milliers de navires.

Nous avons sélectionné les colonnes suivantes pour notre analyse :

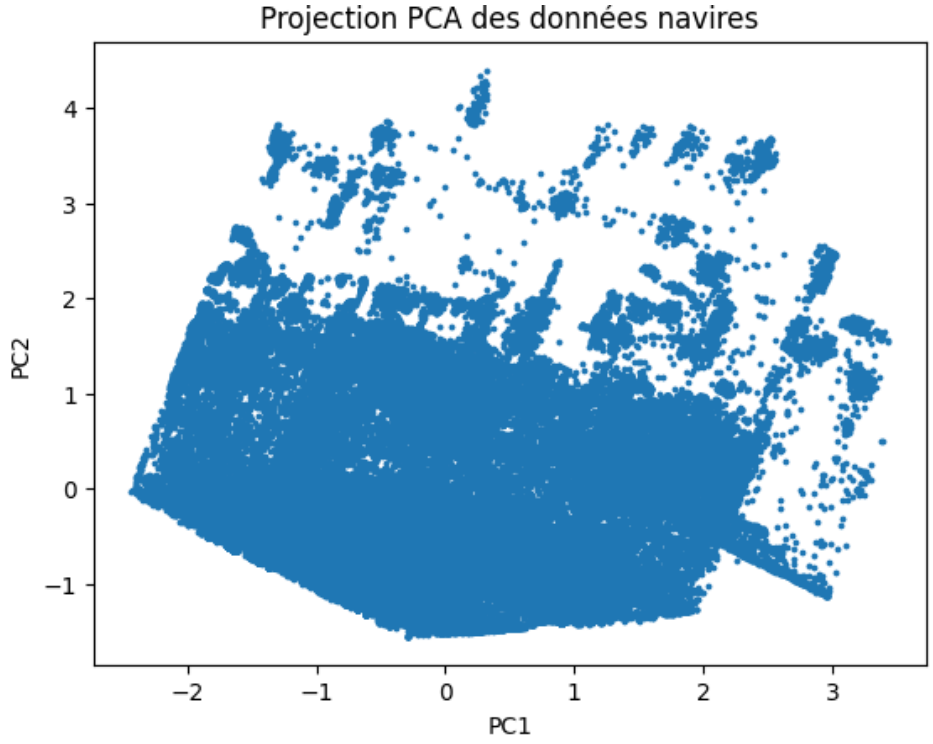
- LAT et LON : latitude et longitude,
- SOG : vitesse sur le fond (Speed Over Ground),
- COG : cap sur le fond (Course Over Ground),
- Heading : direction réelle du navire,
- VesselType : type du navire.

Ces données représentent à la fois la géolocalisation, le comportement dynamique et la typologie des navires. Un filtrage a été appliqué pour ne conserver que les colonnes pertinentes. Ensuite, nous avons procédé à une phase de prétraitement des données. Les variables numériques (SOG, COG, Heading) ont été standardisées à l'aide du StandardScaler, tandis que la variable catégorielle VesselType a été encodée via la méthode One-Hot Encoding. Ce traitement homogénéise les données pour qu'elles soient toutes exprimées dans un espace comparable, indispensable pour les algorithmes de machine learning.

3. Réduction de dimension (PCA)

Afin de visualiser les données de manière plus lisible, nous avons utilisé une Analyse en Composantes Principales (PCA) pour réduire la dimensionnalité des données à deux composantes principales. Cela permet de projeter les données dans un plan 2D tout en conservant un maximum d'informations.

Un nuage de points a été généré à partir de cette projection. Chaque point représente un navire. Nous avons ainsi pu observer visuellement la présence d'amas de navires ayant des comportements similaires, suggérant qu'un regroupement automatique (clustering) était pertinent.

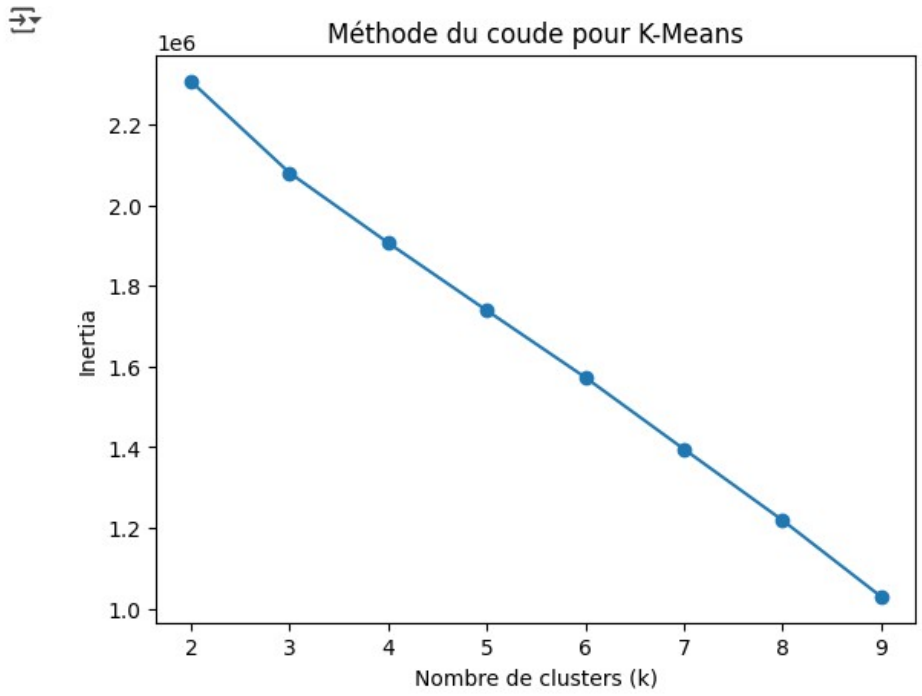


4. Détermination du nombre optimal de clusters

Pour effectuer un clustering non supervisé, nous avons utilisé l'algorithme K-Means. Cet algorithme regroupe les données en k clusters, en minimisant la distance intra-cluster.

Pour déterminer le nombre optimal de clusters, nous avons appliqué la méthode du coude (elbow method). Cette technique consiste à faire varier le nombre de clusters de 2 à 9 et à mesurer l'inertie (somme des distances intra-cluster).

Le graphique obtenu montre une nette cassure (ou "coude") vers k = 5, ce qui suggère que cinq clusters est un bon compromis entre précision et simplicité.



5. Clustering final et évaluation

Nous avons donc fixé k = 5 pour effectuer le clustering final. Chaque navire s'est vu attribuer un label de cluster (de 0 à 4), enregistré dans une nouvelle colonne du DataFrame.

Pour évaluer la qualité de ce regroupement, nous avons utilisé trois métriques standards :

- Le Silhouette Score : il mesure la cohérence interne des clusters. Plus ce score est proche de 1, mieux c'est.
- L'indice de Calinski-Harabasz : il évalue le rapport entre la dispersion entre les clusters et à l'intérieur des clusters.
- L'indice de Davies-Bouldin : plus il est faible, meilleure est la séparation entre les groupes.

Ces indicateurs ont confirmé que le découpage en cinq groupes était cohérent et pertinent.

```
Prédiction de cluster pour un navire

LAT (latitude) : 29.07019
LON (longitude) : -89.29958
SOG (Speed Over Ground) : 13.4
COG (Course Over Ground) : 227.6
Heading : 227
VesselType (valeurs valides : [np.int64(60), np.int64(61), np.int64(70), np.int64(71), np.int64(74), np.int64(79), np.int64(80), np.int64(82), np.int64(84), np.int64(89)]) : 80

✅ Ce navire appartient au cluster : 7
```

6. Visualisation cartographique des clusters

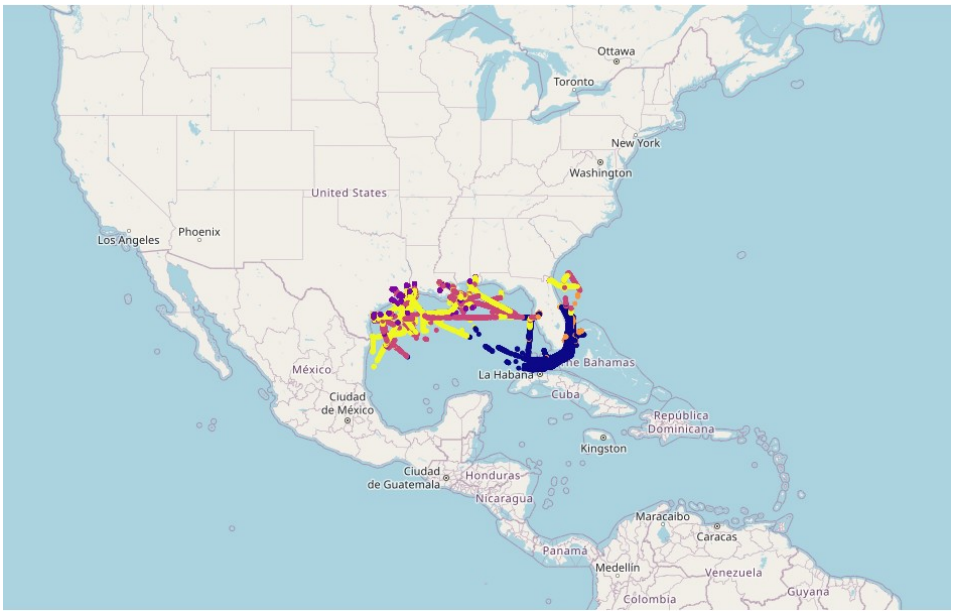
L'un des livrables majeurs de cette partie a été la visualisation interactive des résultats sur une carte géographique, grâce à la bibliothèque Plotly.

Chaque point sur la carte représente un navire, avec :

- Sa position géographique (LAT, LON),
- Sa couleur correspondant à son cluster,
- Et des infos supplémentaires accessibles au survol (vitesse, cap, type).

Cette carte permet de visualiser les zones d'activité propres à chaque groupe de navires, révélant des routes communes, des zones portuaires, ou des regroupements de types similaires de navires.

Carte interactive avec les clusters des navires :



7. Interface de prédiction interactive

Nous avons également mis en place une fonction interactive nommée predict_cluster_interactif(). Elle permet à l'utilisateur de saisir manuellement les caractéristiques d'un navire (latitude, longitude, vitesse, cap), et de recevoir en retour le numéro du cluster auquel il appartient, en se basant sur le modèle préalablement entraîné et sauvegardé (model.pkl).

Cette fonction offre une première ébauche de système de prédiction en temps réel, applicable à des cas concrets de surveillance ou d'aide à la navigation.

Conclusion

Cette première partie du projet a permis d'atteindre les objectifs suivants :

- Nettoyer et prétraiter un jeu de données complexe,
- Mettre en œuvre une analyse de clustering non supervisée avec justification du nombre de groupes,
- Évaluer la qualité des regroupements obtenus,
- Et surtout, représenter visuellement les résultats de manière compréhensible pour un utilisateur non technique.

La segmentation des navires selon leurs caractéristiques de navigation constitue une base solide pour aller plus loin : classification supervisée, détection d'anomalies ou encore modélisation de trajectoires.

Besoin Client 2 : Modèle de prédiction du type de navire

Objectif :

Développer un modèle d'intelligence artificielle capable de prédire le type de navire (VesselTypeGroup) à partir de ses caractéristiques (statut AIS, longueur, largeur, tirant d'eau, cargo).

1. Préparation des données

Variable	Type	Justification
StatusLabel	Catégorielle	Représente le statut AIS du navire (en route, mouillage, etc.) — comportement essentiel.
Length	Numérique	Influence directe sur la classification
Width	Numérique	Corrélée au type de navire (ex : pétroliers plus larges).
Draft	Numérique	Indique la profondeur — souvent plus élevée pour les pétroliers.
Cargo	Numérique	Détermine fortement le type d'usage du navire.

- Standardisation des variables numériques avec StandardScaler.
- Encodage OneHot sur StatusLabel pour gérer les 5 catégories sans ordonnancement.

- Séparation des données en *X_train / X_test* avec train_test_split (stratifié, test_size=20%).

Sauvegardes :

- Le pipeline de préparation est sauvegardé avec joblib pour être réutilisé dans les scripts. Elle se nomme « preprocessor.joblib »

2. Choix du modèle de classification

Modèles testés :

Les modèles ont été comparés grâce aux matrices de confusion et aux métriques liés à celles-ci. Les valeurs de la base de données n'étant pas particulièrement homogènes, ici la métrique à regarder est la « Macro Average » qui, comme expliqué en Annexe 1, calcule indépendamment de la classe pour faire une moyenne pondérée. L'objectif étant d'avoir une valeur forte (>90% idéalement) mais pas parfaite (100%)

- RandomForestClassifier : a donné une précision parfaite (100 %), mais ce résultat est suspect (probablement dû à un data leakage). Écarté pour éviter toute surinterprétation.
- XGBoost et MLPClassifier et k-NN également.
- SVM : mieux mais beaucoup plus lent, sensible à la normalisation et moins lisible que la régression. Après optimisation des hyperparamètres qui sont des réglages externes au modèle influençant la façon dont il apprend, on obtient des résultats trop parfaits également.

```
[[ 2444     0     0]
 [    0 14896     0]
 [    0     0 16574]]
              precision    recall  f1-score   support

          60       1.00      1.00      1.00      2444
          70       1.00      1.00      1.00     14896
          80       1.00      1.00      1.00     16574

    accuracy                           1.00     33914
   macro avg       1.00      1.00      1.00     33914
weighted avg       1.00      1.00      1.00     33914
```

Figure 1 : matrice de confusion et métriques de celle-ci après test randomforestclassifier (voir Annexe 1 pour savoir à quoi correspondent chacune de ces métriques)

```
[[ 2444     0     0]
 [    0 13454  1442]
 [    0     0 16574]]
              precision    recall  f1-score   support

          60       1.00      1.00      1.00      2444
          70       1.00      0.90      0.95     14896
          80       0.92      1.00      0.96     16574

    accuracy                           0.96     33914
   macro avg       0.97      0.97      0.97     33914
weighted avg       0.96      0.96      0.96     33914
```

Figure 2 : matrice de confusion et métriques de celle-ci après test de régression logistique multinomiale (voir Annexe 1 pour savoir à quoi correspondent chacune de ces métriques)

Conclusion : la régression logistique multinomiale a été retenue pour son excellent équilibre performance/simplicité/interprétabilité, avec une accuracy de 96 % sans surapprentissage et également un macro avg moyen de 97%.

Modèle utilisé :

- Régression Logistique Multinomiale (LogisticRegression) avec « solver lbfgs ».
- Approprié pour les problèmes de classification multiclasses, rapide à entraîner et interprétable.

Optimisation :

- Recherche de meilleurs hyperparamètres via GridSearchCV (24 combinaisons testées).
- Hyperparamètres : C, solver, max_iter.

3. Script Python de prédiction

Un script utilisable en ligne de commande a été développé (script_final_client_2.py).

Il :

- Charge le modèle et le préprocesseur sauvegardés.
- Demande et prend les caractéristiques d'un navire entrés par un utilisateur via le terminal.
- Affiche le type de navire prédit à l'utilisateur (ex : "Navire passager", "Cargo", "Pétrolier").

Ceci respecte les exigences du client : pas de réapprentissage à chaque exécution, usage simple.

Conclusion

Le modèle de régression logistique multinomiale mis en place permet une prédiction fiable du type de navire avec une accuracy de 96 % et surtout une macro avg de 97%.

La démarche suit le cahier des charges :

- Prétraitement rigoureux,
- Apprentissage supervisé avec validation croisée,
- Évaluation complète,
- Script opérationnel.

Ce besoin client est donc réalisé de manière robuste et exploitable.

Client 3 – Prédiction de la trajectoire des navires

Objectif :

Le troisième besoin exprimé par notre client consiste à prédire la position future d'un navire à partir de ses caractéristiques présentes et passées. Concrètement, il s'agit d'anticiper la latitude et la longitude à un horizon de 5, 10 ou 15 minutes. Cette prédiction permettrait notamment d'optimiser les dispositifs de prévention de collision ou de faciliter la navigation en milieu encombré.

1. Prétraitement des données

Nous avons tout d'abord nettoyé les données AIS (Automatic Identification System) :

- Suppression des valeurs aberrantes (par exemple, les angles de cap à 511, considérés comme invalides).
- Ajout de variables de lag (ex. : LAT_lag1, SOG_lag2), qui représentent les valeurs précédentes de certains paramètres pour chaque navire. Ces variables permettent au modèle d'intégrer une notion d'historique et d'inertie du déplacement.
- Génération des cibles à prédire : LAT_target et LON_target, qui correspondent à la position réelle du navire à l'horizon choisi (5, 10 ou 15 minutes), en tenant compte d'une tolérance temporelle raisonnable (entre 60 et 120 secondes selon l'horizon).

2. Modélisation

Pour prédire les futures positions géographiques des navires, nous avons choisi un modèle de type RandomForestRegressor, encapsulé dans un MultiOutputRegressor. Ce dernier permet de gérer plusieurs sorties en même temps, ici : LAT_target et LON_target.

Ce choix s'explique par la capacité des forêts aléatoires à capturer des relations non linéaires complexes, tout en étant robustes au bruit et peu sensibles aux valeurs extrêmes. Afin de limiter le risque de surapprentissage, nous avons fixé plusieurs hyperparamètres :

- max_depth : Contrôle la profondeur maximale des arbres de décision pour prévenir le surapprentissage en limitant la complexité du modèle.
- min_samples_split : Définit le seuil minimal d'échantillons nécessaires pour effectuer une division dans un nœud, réduisant ainsi les divisions excessives.
- min_samples_leaf : Impose un nombre minimal d'échantillons par feuille terminale, garantissant des prédictions statistiquement robustes et évitant la création de règles trop spécifiques.

Ces réglages assurent un bon compromis entre complexité et généralisation, ce qui est crucial dans un contexte où les trajectoires peuvent varier selon de nombreux facteurs.

3. Phase d'entraînement

Respect de la temporalité

Dans ce projet, le respect de l'ordre chronologique est essentiel. Les données AIS sont naturellement temporelles, et il aurait été incohérent de mélanger aléatoirement passé et futur.

Nous avons donc conservé les observations dans l'ordre et découpé les jeux d'entraînement et de test en blocs continus, sans shuffle.

Déroulement

Après le prétraitement des données (nettoyage, création des variables de lag, génération des cibles), nous avons entraîné le modèle sur la partie historique (X_train, y_train) en utilisant la méthode fit().

Les prédictions ont ensuite été générées avec predict() sur le jeu de test (X_test), constitué uniquement d'observations futures.

Enfin, pour permettre la réutilisation du modèle sans réentraînement, nous avons utilisé la librairie joblib pour sauvegarder le modèle, ainsi que les jeux de test, avec la fonction joblib.dump().

4. Résultats et interprétation

Pour évaluer les performances de notre modèle, nous avons utilisé deux types de métriques : le RMSE (Root Mean Squared Error), qui mesure l'erreur moyenne en degrés sur la latitude et la longitude, et l'erreur géographique réelle, exprimée en mètres, calculée grâce à la distance géodésique entre la position prédite et la position réelle.

=== Entraînement pour horizon 5 minutes avec hyperparamètres===
RMSE Train ➤ LAT: 0.005269, LON: 0.006440
RMSE Test ➤ LAT: 0.008432, LON: 0.009213
Erreur géographique moyenne : 926.14 mètres
Erreur géographique médiane : 723.77 mètres

=== Entraînement pour horizon 5 minutes sans hyperparamètres===
RMSE Train ➤ LAT: 0.000831, LON: 0.000922
RMSE Test ➤ LAT: 0.005576, LON: 0.007928
Erreur géographique moyenne : 741.94 mètres
Erreur géographique médiane : 562.65 mètres

=== Entraînement pour horizon 15 minutes avec hyperparamètres ===
RMSE Test ➤ LAT: 0.018614, LON: 0.019527
Erreur géographique moyenne : 1989.94 m
Erreur géographique médiane : 1331.33 m

Sans hyperparamètres, le modèle obtient une RMSE très faible en entraînement (0.0008) mais plus élevée en test (jusqu'à 0.0079), ce qui montre un surapprentissage. Avec les hyperparamètres, la RMSE train augmente (0.0052) mais devient plus proche de la RMSE test (0.0084), signe d'un modèle plus généralisable. Ces réglages permettent donc d'éviter que le modèle s'adapte trop aux données d'apprentissage au détriment de sa performance réelle.

Les résultats sont globalement satisfaisants, notamment pour les prévisions à court terme. À 5 minutes, l'erreur moyenne est d'environ 926 mètres, ce qui reste acceptable pour des navires évoluant sur de grandes distances. À 15 minutes, l'erreur atteint 1,9 km en moyenne, ce qui peut encore convenir dans des zones maritimes dégagées où la précision absolue n'est pas critique.

Comme prévu, l'erreur augmente avec l'horizon temporel. Cela s'explique par des facteurs difficiles à anticiper, comme les changements de cap, les variations de vitesse ou des événements externes comme le trafic maritime ou les conditions météo.

Ce type de prédiction peut être utile pour la prévention des collisions, l'aide à la navigation dans des zones encombrées, ou encore la surveillance maritime en détectant des trajectoires anormales par rapport à celles attendues. Le modèle offre donc une capacité d'anticipation fiable, surtout dans des contextes où le comportement des navires reste relativement prévisible.

5. Visualisation des trajectoires

Pour faciliter l'analyse et la présentation des résultats, nous avons mis en place une visualisation cartographique des trajectoires à l'aide de Plotly (scatter_mapbox) et de Folium.

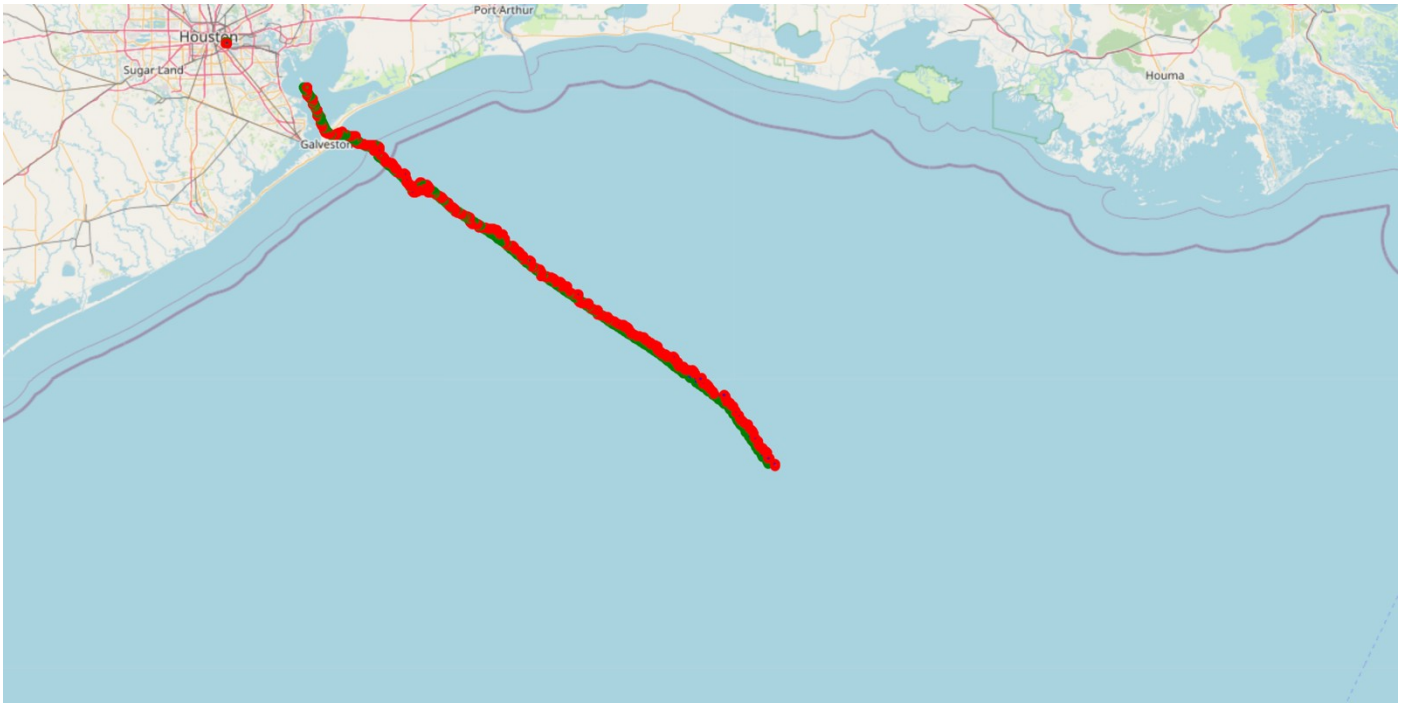


Figure 1: Carte permettant de visualiser la performance du modèle - Trajectoire prédite (rouge) vs réelle (verte)

Pour chaque horizon temporel (5, 10, 15 minutes), nous avons généré un graphique par navire comme ci-dessus, affichant :

- Sa position réelle (en vert),
- Sa position prédite (en rouge),
- Et l’écart entre les deux (modélisée par un trait bleu)

Chaque type de point est représenté avec une couleur ou forme spécifique, et affiché sur un fond de carte interactif.
Cela permet d’évaluer visuellement si la trajectoire prédite suit correctement le mouvement réel du navire, ou s’il y a un écart significatif.

Ces cartes sont particulièrement utiles pour repérer les cas où le modèle fonctionne bien, mais aussi pour identifier les limites en conditions complexes (changements brusques, zones portuaires, etc.).

Conclusion

Ce projet a démontré l'efficacité des techniques d'apprentissage automatique appliquées aux données maritimes en répondant avec succès aux trois besoins clients.

Résultats obtenus

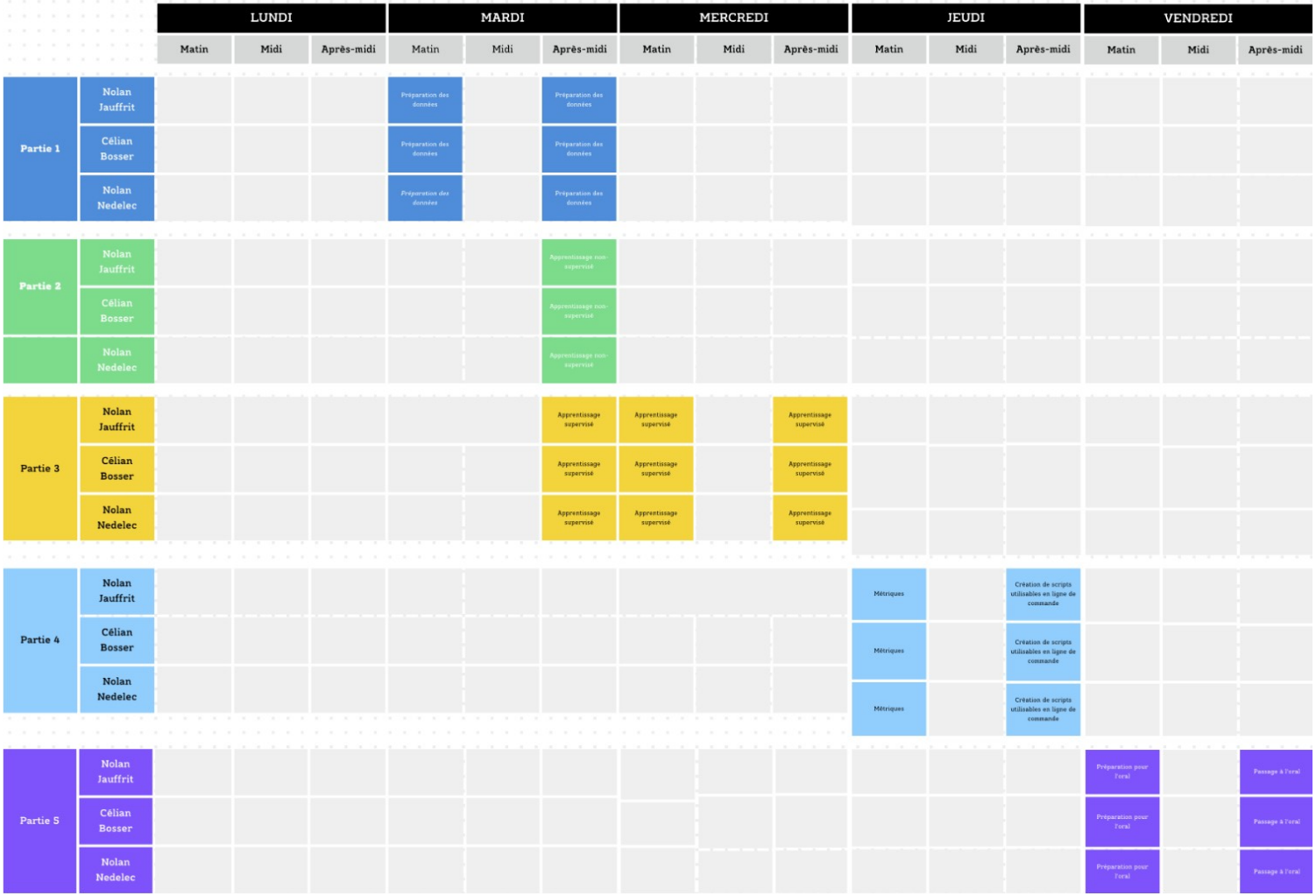
La segmentation K-Means a révélé cinq groupes distincts de comportements de navigation, validés par des métriques de clustering pertinentes. Le modèle de classification a atteint 96% d'accuracy pour la prédiction du type de navire grâce à la régression logistique multinomiale. La prédiction de trajectoires par Random Forest offre une précision de 926 mètres à 5 minutes, suffisante pour des applications de prévention de collision.

Impact et perspectives

Ces trois modèles constituent un écosystème complet d'aide à la décision maritime. Les perspectives d'amélioration incluent l'intégration de données météorologiques et l'exploration de modèles de deep learning pour capturer des dépendances temporelles plus complexes.

Ce projet démontre que l'intelligence artificielle apporte une valeur ajoutée significative au secteur maritime, contribuant à améliorer la sécurité, l'efficacité et la surveillance des activités navales.

Diagramme de Gantt Projet IA



ANNEXES

Annexe 1:

Accuracy (Exactitude globale) :

- C’est la proportion de bonnes prédictions.
- Utile quand les classes sont équilibrées.
- Ici, même si les classes ont des tailles différentes (ex : beaucoup plus de "70" ou "80"), cela donne une vue d'ensemble.

Precision (Précision) :

- Parmi les navires prédits comme étant d’un type donné, combien le sont vraiment ?
- Important si les faux positifs sont coûteux (par exemple, mal identifier un tanker comme un cargo pour des raisons de sécurité maritime).

Recall (Rappel) :

- Parmi les vrais navires d’un type donné, combien ont été bien identifiés ?
- Important si manquer un certain type de navire est risqué (ex : navire de passagers pour sécurité).

F1-score :

- Moyenne harmonique entre précision et rappel.
- Bon compromis si l’on veut équilibrer les deux.

Matrice de confusion :

- Permet de visualiser les types d’erreurs :
    - Quelle classe est confondue avec quelle autre ?
    - Par exemple, certains cargos peuvent être pris pour des tankers.

Macro avg (Macro Average)

- C’est la moyenne arithmétique des scores (précision, rappel, F1) calculée indépendamment pour chaque classe, sans tenir compte du nombre d'exemples.
- Chaque classe a le même poids, peu importe sa fréquence.

Weighted avg (Weighted Average)

- C’est la moyenne pondérée des scores (précision, rappel, F1), où chaque classe est pondérée par son nombre d'exemples (*support*).
- Les classes fréquentes ont plus d’influence.