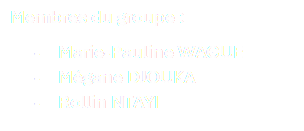


Etude des données AIS

Etude des données AIS

PROJET IA

**SOMMAIRE**

1. **Visualisation des navires**
2. Préparation des données
3. Apprentissage non supervisé
4. Métriques pour l'apprentissage non-supervisé
5. Visualisation sur une carte
6. Préparation d’un script
7. **Modèle de prédiction du type de navire**
8. Préparation des données
9. Apprentissage supervisé
10. Choix du meilleur modèle
11. Réalisation du script
12. **Prédiction de la trajectoire des navires**
13. Préparation des données
14. Apprentissage supervisé

2-1- Entrainement du meilleur modèle

2-2- Vérification de la précision du modèle et enregistrement

1. Utilisation du modèle dans le script

3-1- Objectif du script

3-2- Fonctionnement du script

## **Introduction**

Dans le cadre de ce projet d’intelligence artificielle appliqué à la navigation maritime, nous avons conçu un système capable de **prédire le type de navire** (Passenger, Cargo, Tanker, Other) à partir de **caractéristiques physiques simples** : la **longueur**, la **largeur** et le **tirant d’eau**.

L’objectif est de proposer un outil de classification automatique, fiable et rapide, à partir de données disponibles dans les systèmes de suivi maritime. Après une phase de **préparation des données**, plusieurs **modèles supervisés** ont été évalués afin d’identifier celui offrant les meilleures performances.

Parmi eux, le **modèle Random Forest** a été retenu pour sa capacité à bien généraliser sur de nouvelles données, avec une **accuracy moyenne de 75,29 %** obtenue via validation croisée. Un **script Python interactif** a ensuite été développé pour permettre la prédiction directe du type de navire à partir de simples entrées utilisateur.

1. **Visualisation des navires**
2. Préparation des données

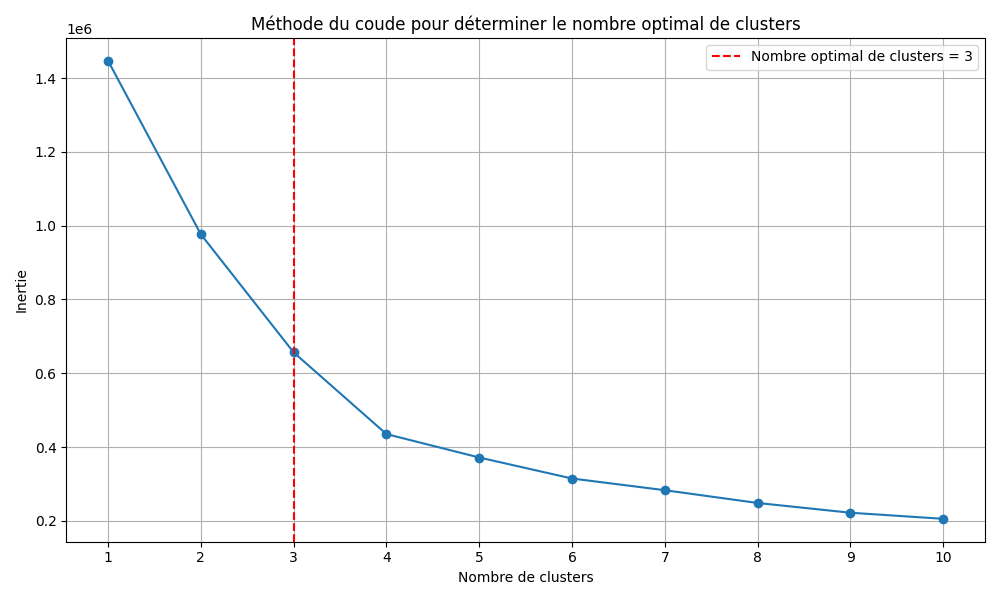
Dans l’optique de répondre aux besoins du client 1, nous avons jugé que les données les plus pertinentes pour l’analyse sont :

* COG, Cap du navire par rapport au fond. Un changement abrupt ou un cap incohérent peut indiquer un comportement suspect.
* SOG, Vitesse du navire par rapport au fond. Une vitesse anormalement élevée ou basse pour une zone donnée ou un type de navire peut être une anomalie.
* Heading, Angle de cap réel du navire. Similaire au COG, il peut révéler des changements de direction inattendus.
* LAT, Coordonnée géographique. Essentielle pour situer le navire et identifier des regroupements spatiaux ou des déviations de trajectoire.
* LON, Coordonnée géographique. Complémentaire à la latitude pour la position du navire.

Aucune des données utilisées ci-dessus ne nécessitaient d’encodage ou avait déjà été encodé au préalable lors du nettoyage de la partie Big Data, du coup aucun encodage des données catégorielles n’a été effectué.

1. Apprentissage non-supervisé

L’algorithme sélectionné est celui du **KMeans** car moins gourmand en mémoire dans l’environnement VSCode. Ensuite, une fois l’algorithme sélectionné, il fallait déterminer le coefficient K optimal pour regrouper les navires selon les schémas de navigation similaires, pour cela nous avons utilisé la méthode de coude, dont le résultat est la courbe ci-dessous :

%

il faut choisir la valeur de K où l'inertie (WCSS - Within-Cluster Sum of Squares) commence à diminuer de manière moins significative, formant un "coude" ou un "genou".

En observant le graphique :

* De K=1 à K=2, la diminution de l'inertie est très forte.
* De K=2 à K=3, la diminution est encore significative.
* À partir de **K=3**, la courbe semble s'aplatir considérablement. La réduction de l'inertie en passant de K=3 à K=4 est moins prononcée que les réductions précédentes.

Par conséquent, la valeur de **K=3** est le point où le "coude" est le plus apparent sur ce graphique. C'est la valeur qui représente le meilleur compromis entre la réduction de la variance intra-cluster et la complexité du modèle (le nombre de clusters).

**Donc, l’application de la méthode de Coude, la valeur de K à choisir est 3.**

Analyse des résultats du clustering :

SOG COG Heading LON LAT

Cluster

0 1.516896 21.345807 18.431666 -92.794316 29.274832

1 13.886652 250.693679 323.077409 -91.529376 29.028200

2 6.067320 105.342988 101.969619 -80.851723 26.059252

Taille des clusters :

Cluster

0 159248

1 78453

2 51710

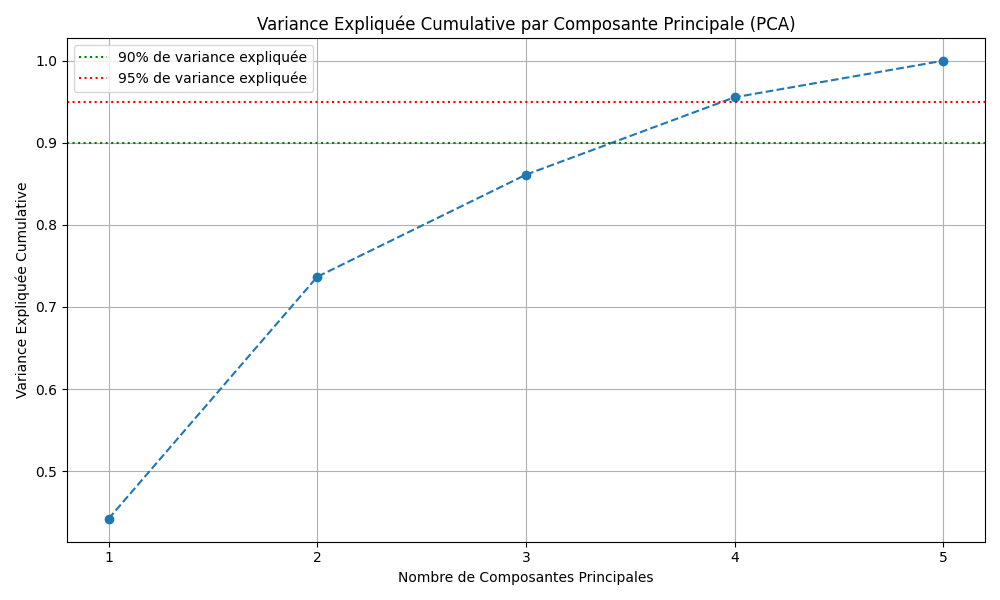
Name: count, dtype: int64

**Nous obtenons la répartition des données suivantes.**

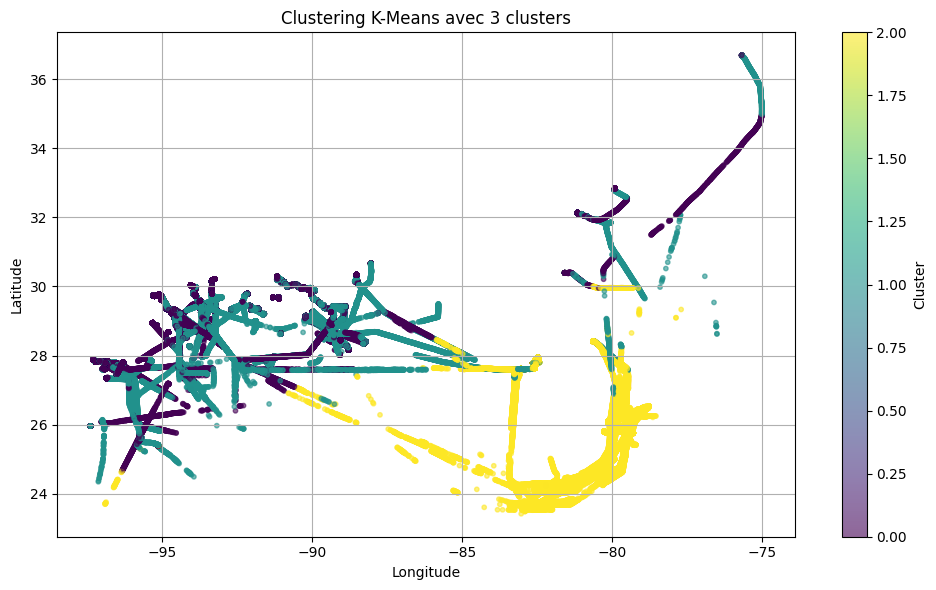
**REDUCTION DE LA DIMENSIONNALITE**

Pour réduire les données à étudier, ainsi réduire le temps de traitement des données, nous avons cherché à réduire la dimensionnalité de la base de données standardisée des colonnes sélectionnés, en cherchant dans un premier temps le « pca\_component », qui fait référence aux nouvelles dimensions créées par l’algorithme PCA après la transformation des données nécessaires pour obtenir la meilleure variance des données originales (au moins 95%).

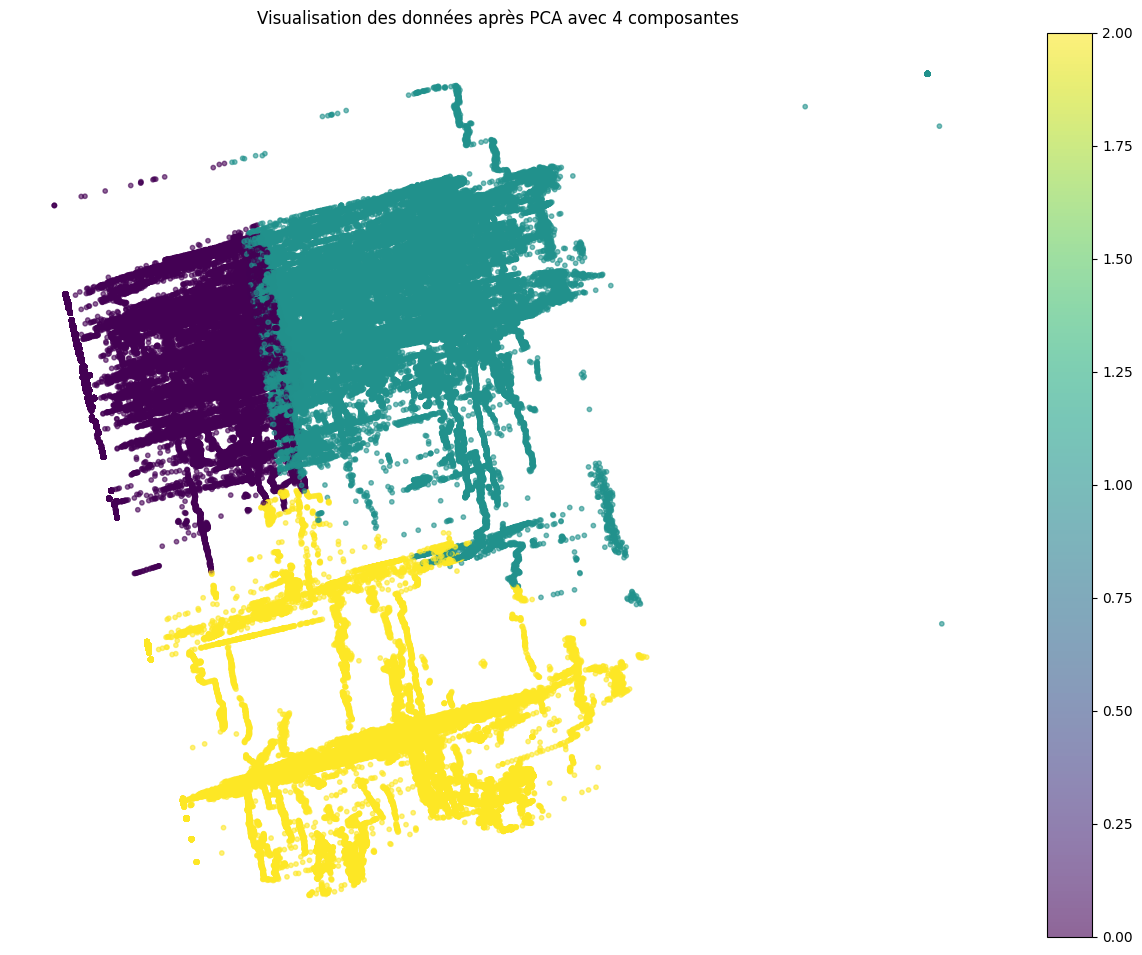
Comme l’indique cette courbe, nous atteignons le pic de Variance expliquée à **4** composants principaux, par conséquent **pca\_component = 4.**



**Représentation sans PCA**

****

**Représentation avec PCA**

****

Nous pouvons constater que contrairement à la représentation sans PCA, les données sont relativement plus digestes et distinctes, ce qui facilitera l’interprétation.

Analyse des résultats du clustering :

SOG COG Heading LON LAT

Cluster

0 6.064203 105.320748 101.951491 -80.852512 26.057100

1 1.526401 21.712739 18.638466 -92.790564 29.274633

2 13.923076 250.955777 323.980864 -91.526758 29.027798

D’après les résultats du clustering ci-dessus, On constate les critères utilisés par l’algorithme pour catégoriser les données d’entrée dans un cluster sont les suivantes.

**REMARQUE**

Nous avons essayé d’utiliser l’algorithme DBSCAN mais suite à des problèmes d’utilisation excessive de la mémoire de cette dernière, nous avons décidé de nous tenir uniquement à l’utilisation de l’algorithme KMeans.

1. Métriques pour l’apprentissage non-supervisé

Nous avons utilisé les métriques pour définir la qualité du clustering kMeans Évaluation des métriques de clustering :

Nous avons fait cette étude pour chaque valeur de k.

> k=1: Un seul cluster ou aucun cluster formé. Scores d'évaluation non calculables.

> k=2: Inertie (PCA)=912200.46, Silhouette=0.4087, CH=139401.05, DB=1.2355

> k=3: Inertie (PCA)=592161.07, Silhouette=0.4901, CH=174224.06, DB=0.9481

> k=4: Inertie (PCA)=370724.67, Silhouette=0.5080, CH=224370.22, DB=0.7183

> k=5: Inertie (PCA)=314115.37, Silhouette=0.5025, CH=204594.52, DB=0.7635

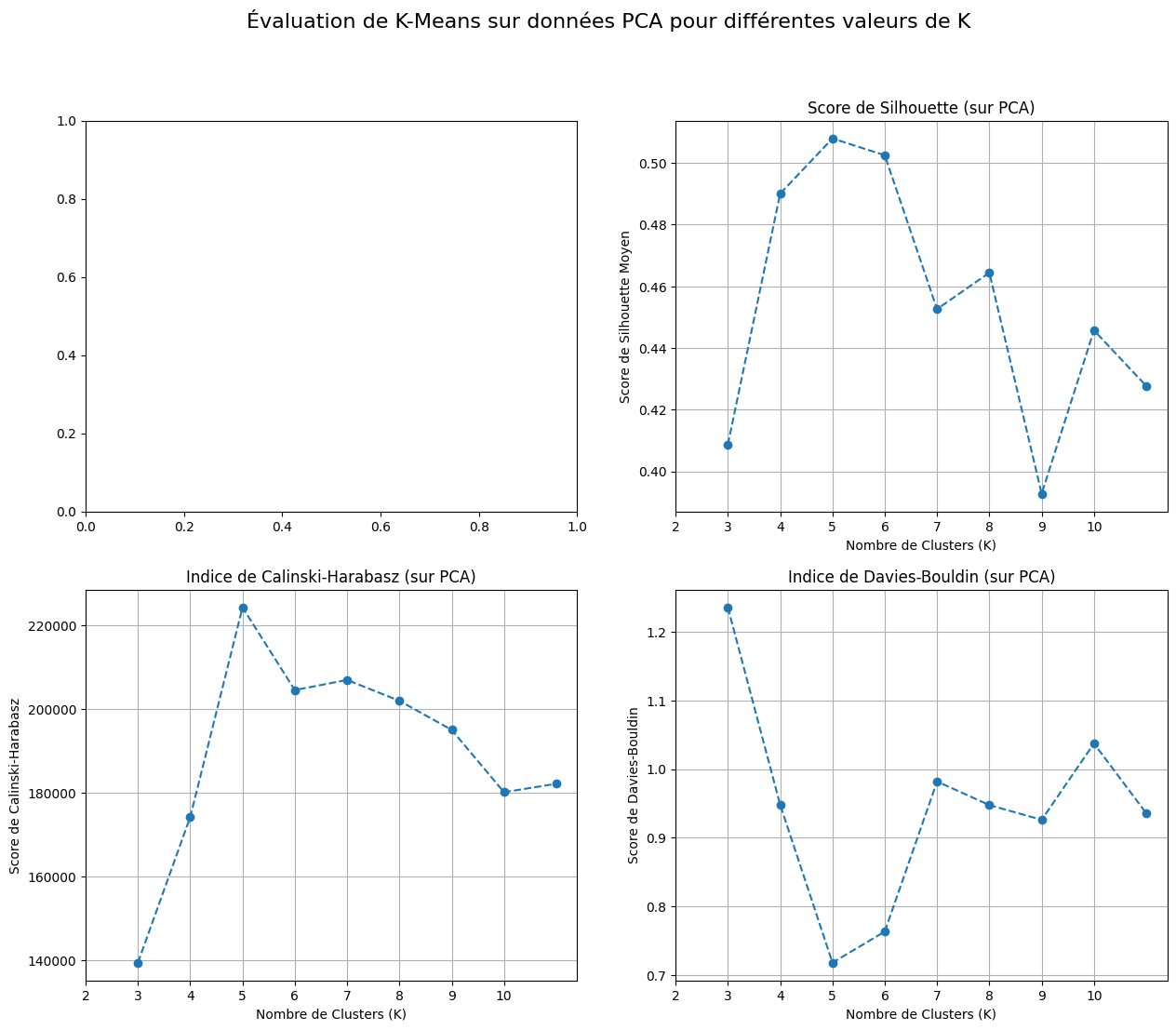
> k=6: Inertie (PCA)=258370.80, Silhouette=0.4526, CH=207014.87, DB=0.9820

> k=7: Inertie (PCA)=222978.78, Silhouette=0.4644, CH=202028.39, DB=0.9477

> k=8: Inertie (PCA)=197358.28, Silhouette=0.3927, CH=195066.87, DB=0.9263

> k=9: Inertie (PCA)=184916.89, Silhouette=0.4457, CH=180165.59, DB=1.0374

> k=10: Inertie (PCA)=160504.51, Silhouette=0.4277, CH=182192.69, DB=0.9358



On sait que :

David Bouldin : Un score plus **faible** est meilleur, mesure le rapport entre la dispersion intra-cluster et la séparation inter-cluster pour chaque cluster et moyenné.

Silhouette :

* **+1 :** Le point est bien regroupé avec son propre cluster et bien séparé des clusters voisins.
* **0 :** Le point est très proche de la frontière entre deux clusters.
* -**1 :** Le point est probablement assigné au mauvais cluster.

Calinski-Harabasz : Un score plus **élevé** indique des clusters bien définis, Mesure le rapport entre la variance inter-cluster (entre les clusters) et la variance intra-cluster (au sein des clusters.

Tout semble nous indiqué que le nombre de clusters idéal est de 5.

Cependant les graphiques eux affirment le contraire, par exemple prenons le cas où k=5, on obtient :

Analyse des résultats du clustering :

SOG COG Heading LON LAT

Cluster

0 7.655193 232.950913 289.828477 -91.891991 28.996518

1 1.253112 13.128933 12.271747 -92.794123 29.297841

2 3.649488 25.976604 22.141114 -80.879829 26.490698

3 102.300000 360.000000 510.896348 -91.096333 29.689839

4 9.692925 220.785555 218.455751 -80.899879 25.634871

Taille des clusters :

Cluster

0 80858

1 150276

2 29997

3 4901

4 23379

Name: count, dtype: int64

Une analyse des résultats de Clustering disproportionnés, comme le confirme la visualisation des données après le PCA :

 **On ne remarque véritablement que 4 couleurs, ce qui est problématique, donc les clusters ne sont pas assez bien répartis.**

**De même pour k=4.**

Taille des clusters :

Cluster

0 81979

1 51714

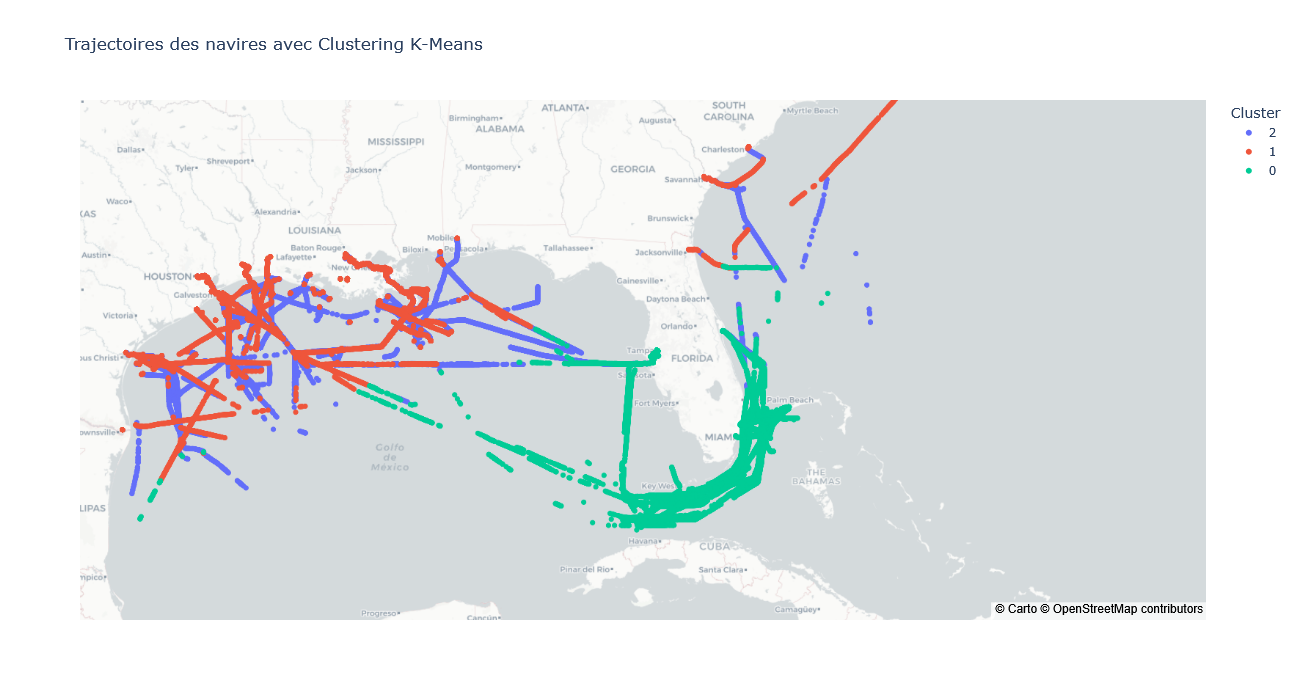
2 150817

3 4901

Name: count, dtype: int64

Dans ce cas ci aussi on constate une disproportion de la taille des clusters, donc malgré l’indication des métriques et par élimination nous avons conclut que le choix le plus judicieux est celui des 03 clusters, **K = 3**.

1. Visualisation sur une carte



**Visualisation directe des points de notre cluster et des directions en fonction du cluster**

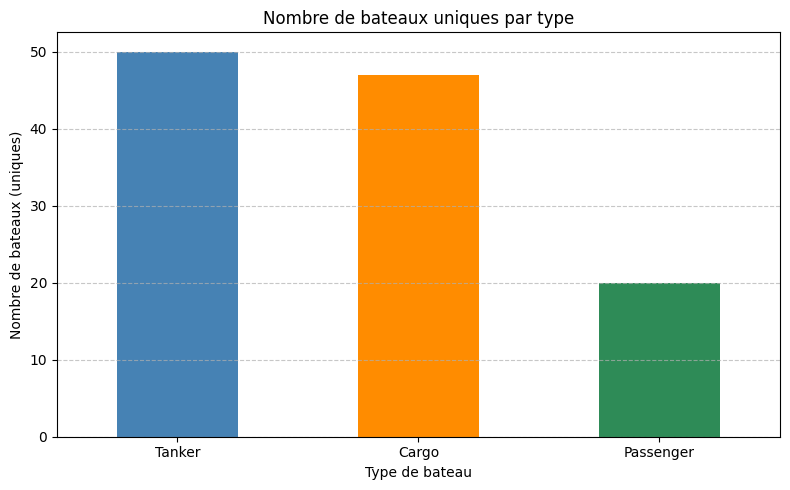
Cette carte nous permet d’avoir une représentation visuelle des trajectoires de chaque navire en fonction de son cluster (Couleur différente). On constate dans un premier temps que la carte nous indique bien le golfe du Mexique ce qui justifie sa fiabilité.

1. **Modèle de prédiction du type de navire**

1. **Préparation des données**

Le fichier CSV nettoyé, contenant les informations sur les navires a été chargé, puis filtré pour ne conserver qu’un enregistrement unique par navire (via le champ MMSI).

On obtient donc cette repartition sur 117 bateaux :



Le type de navire (VesselType) a été regroupé en quatre catégories :

* **Passenger** (60–69),
* **Cargo** (70–79),
* **Tanker** (80–89),
* **Other** (toutes les autres valeurs).

Trois variables numériques ont été sélectionnées comme les plus pertinentes : Length, Width, Draft, car ce sont les variables qui sont les plus corrélées au type de bateau (Voir matrice)



**NB**: Cargo n’a pas été choisi car il donne énormément d’information sur le type de bateau et avec lui la prédiction perdra tout son sens car elle sera parfaite.

Ces variables ont été normalisées à l’aide de StandardScaler afin d’homogénéiser les échelles des différentes caractéristiques. Les données ont ensuite été divisées en jeu d’entraînement (80 % dont 93 bateaux) et jeu de test (20 % dont 24 bateaux), La séparation a été réalisée avec un random\_state fixe pour assurer la reproductibilité.

2. **Modélisation supervisée**

Quatre modèles de classification ont été testés :

* Régression Logistique
* SVM (Support Vector Machine)
* Random Forest
* Arbre de Décision.

Chaque modèle a été évalué à l’aide de **GridSearchCV** pour optimiser les hyperparamètres. Les performances ont été mesurées à l’aide des métriques suivantes : **Accuracy**, **F1-score** , **Precision**, **Recall**, et **matrice de confusion**.

Le **F1-score** est la **moyenne harmonique** entre la **précision** (*precision*) et le **rappel** (*recall*).  
Il donne une mesure **équilibrée** de la performance du modèle, surtout quand les classes sont **déséquilibrées** comme dans notre cas (, peu de navires "Passenger" comparés aux "Cargo" et Tanker)

**Précision** est la proportion de vraies prédictions positives parmi les cas prédits positifs.

Par exemple : combien de bateaux prédits *Tanker* sont réellement *Tanker* ?

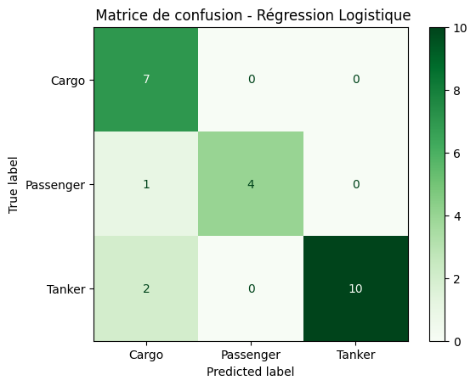
**Rappel** est la proportion de vraies prédictions positives parmi tous les cas réellement positifs.

Par exemple : combien de *Tanker* réels ont été bien détectés ?

Les résultats obtenus sont les suivants :

* **Régression Logistique**

Régression Logistique - Meilleurs paramètres : {'C': 10, 'solver': 'lbfgs'}



Rapport Régression Logistique :

precision recall f1-score support

Cargo 0.70 1.00 0.82 7

Passenger 1.00 0.80 0.89 5

Tanker 1.00 0.83 0.91 12

accuracy 0.88 24

macro avg 0.90 0.88 0.87 24

weighted avg 0.91 0.88 0.88 24

* **SVM**

SVM - Meilleurs paramètres : {'C': 10, 'gamma': 'scale', 'kernel': 'linear'}

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

Le contenu généré par l’IA peut être incorrect.

Rapport SVM :

precision recall f1-score support

Cargo 0.54 1.00 0.70 7

Passenger 1.00 0.80 0.89 5

Tanker 1.00 0.58 0.74 12

accuracy 0.75 24

macro avg 0.85 0.79 0.78 24

weighted avg 0.87 0.75 0.76 24

* **Random Forest**

Random Forest - Meilleurs paramètres : {'criterion': 'entropy', 'max\_depth':20,'n\_estimators': 50}

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

Le contenu généré par l’IA peut être incorrect.

Rapport Random Forest :

precision recall f1-score support

Cargo 0.55 0.86 0.67 7

Passenger 1.00 0.80 0.89 5

Tanker 0.89 0.67 0.76 12

accuracy 0.75 24

macro avg 0.81 0.77 0.77 24

weighted avg 0.81 0.75 0.76 24

* **Une image contenant texte, capture d’écran, diagramme, Rectangle

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

Rapport Decision Tree :

precision recall f1-score support

Cargo 0.55 0.86 0.67 7

Passenger 1.00 0.80 0.89 5

Tanker 0.89 0.67 0.76 12

accuracy 0.75 24

macro avg 0.81 0.77 0.77 24

weighted avg 0.81 0.75 0.76 24

Decision Tree - Meilleurs paramètres : {'criterion': 'entropy', 'max\_depth': None}

Après comparaison des performances, on obtient :

Comparaison des performances :

SVM: Accuracy=0.7500, F1=0.7578

Decision Tree: Accuracy=0.7500, F1=0.7606

Logistic Regression: Accuracy=0.8750, F1=0.8799

Random Forest: Accuracy=0.7500, F1=0.7606

Meilleur modèle : Logistic Regression avec une accuracy de 0.8750

**3-** **Choix du meilleur modèle grâce à la validation croisée avec 5 folds**

Après avoir appliqué le cross val-score sur chaque modèle , on obtient les résultats suivants :

=== Validation croisée (Random Forest) ===

Scores par fold: [0.875 0.54166667 0.86956522 0.7826087 0.69565217]

Score moyen: 0.753 ± 0.124

=== Validation croisée SVM ===

Scores par fold: [0.75 0.54166667 0.69565217 0.69565217 0.7826087 ]

Score moyen: 0.693 ± 0.083

=== Validation croisée (Decision Tree) ===

Scores par fold: [0.75 0.54166667 0.86956522 0.69565217 0.69565217]

Score moyen: 0.711 ± 0.106

=== Validation croisée (accuracy) ===

Scores par fold: [0.70833333 0.54166667 0.69565217 0.60869565 0.91304348]Score moyen: 0.693 ± 0.125

Cette méthode donnant plus de crédibilité sur la généralisation du modèle nous a donc permis de choisir le meilleur modèle que nous avons utilisé pour la suite. Le résultat du choix du meilleur modèle basé sur le f1\_score est donc le suivant :

Résumé des scores de validation croisée (accuracy) :

Random Forest: 0.7529 ± 0.1243

SVM: 0.6931 ± 0.0827

Decision Tree: 0.7105 ± 0.1056

Logistic Regression: 0.6935 ± 0.1254

Meilleur modèle selon la validation croisée : **Random Forest avec une accuracy moyenne de 0.7529**

Ce modèle généralise donc le mieux sur de nouvelles données

**Entraînement du modèle Random Forest**

Le modèle :

* Crée **plusieurs arbres de décision** (par exemple, 100 arbres),
* Chaque arbre est entraîné sur un **échantillon aléatoire** du jeu d'entraînement,
* À chaque nœud, il choisit une **caractéristique aléatoire** (ex. Length) et cherche la **meilleure coupure** (ex. Length > 200) pour séparer les classes.

**4. Réalisation du script de prédiction**

**📄 Objectif**

Le but de ce script Python est de permettre à un utilisateur de **prédire le type d’un navire** (Passenger, Cargo, Tanker ou Other) à partir de **trois caractéristiques physiques** :

* **Longueur** (Length)
* **Largeur** (Width)
* **Tirant d’eau** (Draft)

Ce script repose sur un **modèle Random Forest** entraîné précédemment et sauvegardé dans un fichier (best\_random\_forest\_model.pkl), ainsi qu’un **scaler** utilisé pour la normalisation (scaler.pkl)

**Fonctionnement du script**

Le modèle et l’objet de normalisation sont chargés à l’aide de joblib.

**Fonction de prédiction** :

* Les données saisies sont mises sous forme de tableau.
* Elles sont **normalisées** avec le scaler précédemment entraîné.
* La prédiction est faite par le modèle Random Forest.
* Le type de bateau prédit est retourné.

**NB :** Un fichier .txt (Readme.txt) est mis à disposition pour expliquer le fonctionnement du script.

Quelques prédictions

| **Length (m)** | **Width (m)** | **Draft (m)** | **VesselType** |
| --- | --- | --- | --- |

|  |  |  |  |
| --- | --- | --- | --- |
| 45 | 12 | 5 | Cargo |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 250 |  | 44 | 12 | Tanker |

|  |  |  |  |
| --- | --- | --- | --- |
| 47 | 9.5 | 2.5 | Passenger |

1. **Prédiction de la trajectoire des navires**

Dans l’optique de résoudre un problème temporel, nous allons entraîner et évaluer un modèle de prédiction de trajectoire.

1. **Préparation des données**

Le fichier CSV nettoyé, contenant les informations sur les navires est chargé, puis filtré (via le champ MMSI et BaseDateTime). Ensuite, on va sélectionner les données (colonnes) pertinentes pour la prédiction de la trajectoire des bateaux. Ensuite, nous allons créer les nouvelles variables qui vont nous aider pour prédire la trajectoire de chaque bateau à t={t+5, t+10, t+15} (en minutes).

Nous avons jugé que les données les plus pertinentes pour prédire la trajectoire sont :

* COG, Cap du navire par rapport au fond.
* SOG, Vitesse du navire par rapport au fond.
* Heading, Angle de cap réel du navire.
* LAT, Coordonnée géographique.
* LON, Coordonnée géographique.

Ce sont les features. LAT et LON sont les targets.

Ensuite, on crée les tableaux qui contiendront la position des navires à 5, 10 et 15 minutes après l’instant t de départ.

1. **Apprentissage supervisé**

**2-1- Entraînement du modèle**

Nous avons commencé par diviser la base de données en base d’entraînementet en base de test avec un ratio 80-20. Afin d’avoir les meilleurs résultats possibles, 04 modèles ont été testés lors de l’entraînement, avec leur meilleurs paramètres, grâce à RandomizedSearchCV. Ce sont les modèles :

* Random Forest Regressor
* Random Forest Regressor avec Multi Output Regressor
* Decision Tree Regressor
* Linear Regression

Les performances ont été mesurées à l’aide des métriques suivantes :

* RMSE (Root Mean Square Error) ou Racine carrée des erreurs
* Cross Val Score ou validation croisée

Après d’entraînement, on obtient les résultats suivants :

|  |  |  |
| --- | --- | --- |
| Modèle | RMSE | Cros |
| Random Forest Regressor | 0.004790157245282562 | 0.0074527720629739376 |
| Random Forest Regressor avec Multi Output Regressor | 0.012133151963541526 | 0.022481851682332018 |
| Decision Tree Regressor | 0.006472452322404998 | 0.022481851682332018 |
| Linear Regression | 2.4151419090324252e-14 | 0.011901803120324742 |

Meilleur modèle : Random Forest. Ce modèle généralise donc le mieux sur de nouvelles données.

**2-2- Vérification de la précision du modèle et enregistrement**

Afin de voir la qualité globale du modèle, on a utilisé le coefficient de détermination R². Il mesure la proportion de variance expliquée par le modèle Selon ses valeurs, on peut conclure sur la perfection des résultats obtenus. En effet :

R²= 1 : Prédiction parfaite

R²=0 : Le modèle ne fait pas mieux que la moyenne

R²<0 : Le modèle n’est pas bon

Nous avons calculé R² avec les données réelles et les données prédites et on a obtenu R²=1. La prédiction peut donc être considérée comme parfaite.

Pour finir, ce modèle est enregistré avec l’extension .pkl afin de le charger dans le script. Il nous sera également utile pour la suite du projet outils numériques.

1. **Utilisation du modèle dans un script**

**3-1- Objectif du script**

Le but de ce script Python est de permettre à un utilisateur de prédire la trajectoire d’un navire (latitude et longitude) à t+ 5, 10 et 15 minutes, à partir de 7 caractéristiques, celles des positions précédentes:

* Longueur (Length)
* Tirant d’eau (Draft)
* L’angle du cap réel du navire (Heading)
* Le cap du navire (COG)
* La vitesse du navire (SOG)
* Longitude (LAT)
* Latitude (LON)

Ce script repose sur un modèle Random Forest entraîné précédemment et sauvegardé dans un fichier (model\_random\_forest.pkl).

**3-2- Fonctionnement du script**

Après contrôle de saisie, le modèle est chargé à l’aide de joblib. Les données saisies sont mises sous forme de tableau, puis la prédiction est faite par le modèle Random Forest. Les latitudes et longitudes futures prédites sont retournées.

## **Conclusion**

Le modèle de prédiction développé dans ce projet permet de **classifier efficacement les navires** à partir de données physiques de base. Grâce à une approche rigoureuse combinant **prétraitement, modélisation, évaluation et optimisation**, nous avons mis en place un **système robuste et opérationnel** basé sur Random Forest.

Le **script final**, simple d'utilisation, permet à tout utilisateur de saisir les dimensions d’un navire et d’obtenir une prédiction fiable de son type. Ce travail constitue une première étape vers une solution plus complète d’analyse automatisée du trafic maritime, qui pourrait à terme intégrer des données temporelles ou comportementales pour aller plus loin dans la détection de profils ou d’anomalies.

**REFERENCE**

**GitHub :** [**https://github.com/Rollintayi/Project\_ISEN\_IA.git**](https://github.com/Rollintayi/Project_ISEN_IA.git)

Nous avons utilisé GitHub pour coordonner les travaux et le GANT, nous avons créer des branches, chacune correspondant à un besoin client avant de « push » les codes de chacun dans le fichier main principal.

