**Compte rendu TP OC02 : Gouvernance de données**

Le projet « **Gouvernance des données - Mini-plateforme de véhicule connecté** » avait pour objectif de reproduire les grands principes d'une architecture cloud AWS (IoT Core, Lambda, Timestream, SageMaker, QuickSight) en utilisant uniquement Python et des outils open source.  
Nous avons mis en œuvre un pipeline complet de gestion de données de véhicules connectés, comprenant :

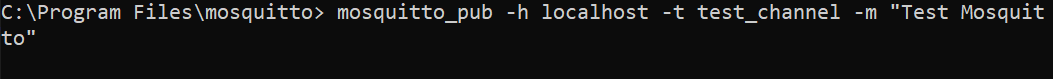
* **La récupération de données** issues de la plateforme EnviroCar, simulant les véhicules connectés.
* **La publication et la réception asynchrone** de ces données via MQTT, à l’aide de Mosquitto et Paho-MQTT.
* **Le stockage et le traitement** de ces messages dans des fichiers CSV avec Pandas.
* **Le filtrage et l’enrichissement** des données pour isoler des comportements spécifiques (par exemple, des vitesses moyennes élevées).
* **La construction d’un modèle de machine learning** simple pour prédire la consommation de carburant en fonction de la vitesse moyenne (régression linéaire avec Scikit-learn).
* **La visualisation cartographique** des trajets via Folium.
* **La création d’une interface web** interactive à l’aide de Flask pour accéder aux données et aux cartes.
* Chaque composant a été conçu pour être modulaire, réutilisable et aisément testable de manière indépendante.
* Etape 1 : Simulation de la génération de données depuis enviroCar via MQTT

Test rapide du broker

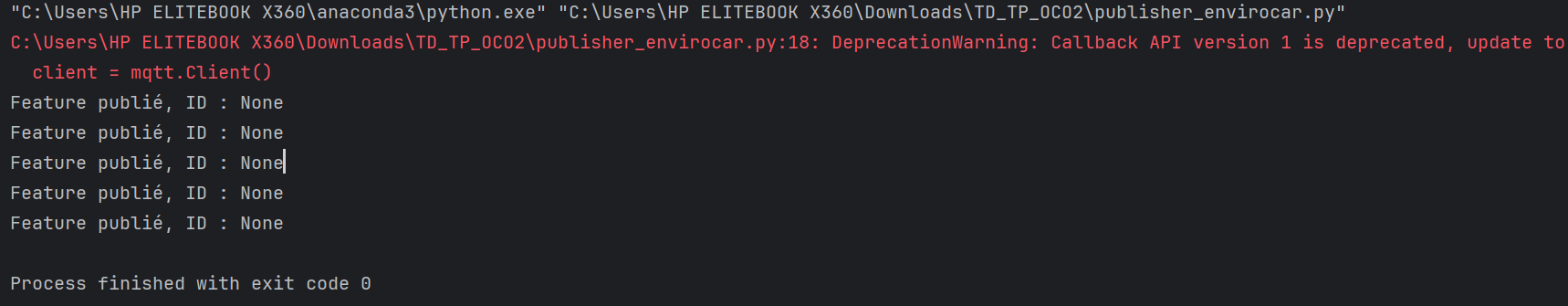
1. Client abonné au topic test\_channel



1. Publication du message

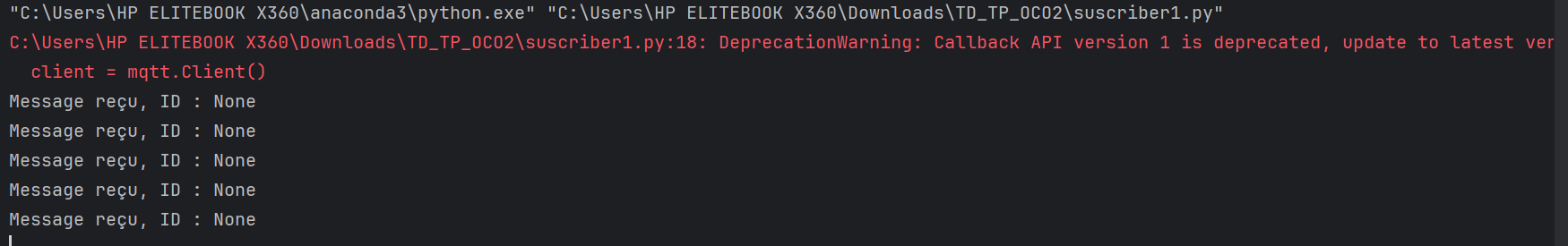


1. Script publisher\_envirocar.py



* Etape 2) Réception et traitement asynchrone

Suscriber.py

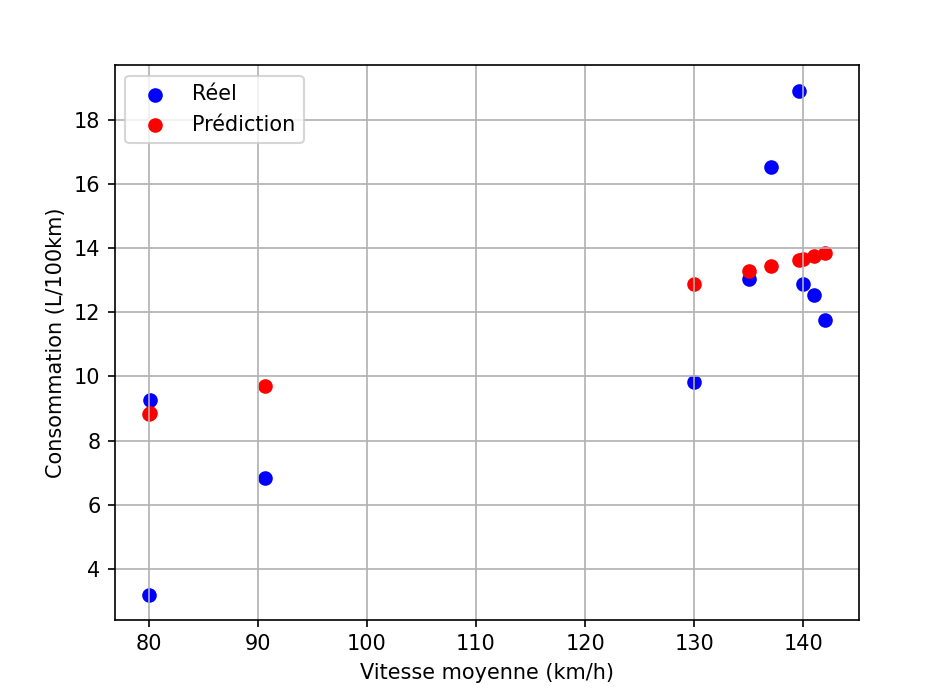


* Etape 3 : filtrage et enrichissement des données

Le script enrichement.py permet de générer un fichier csv qui est issu d’une filtration du ficher précédemment obtenu avec le suscriber.py de sorte à ne garder que les enregistrements dont la vitesse moyenne est supériure à 80km/h.

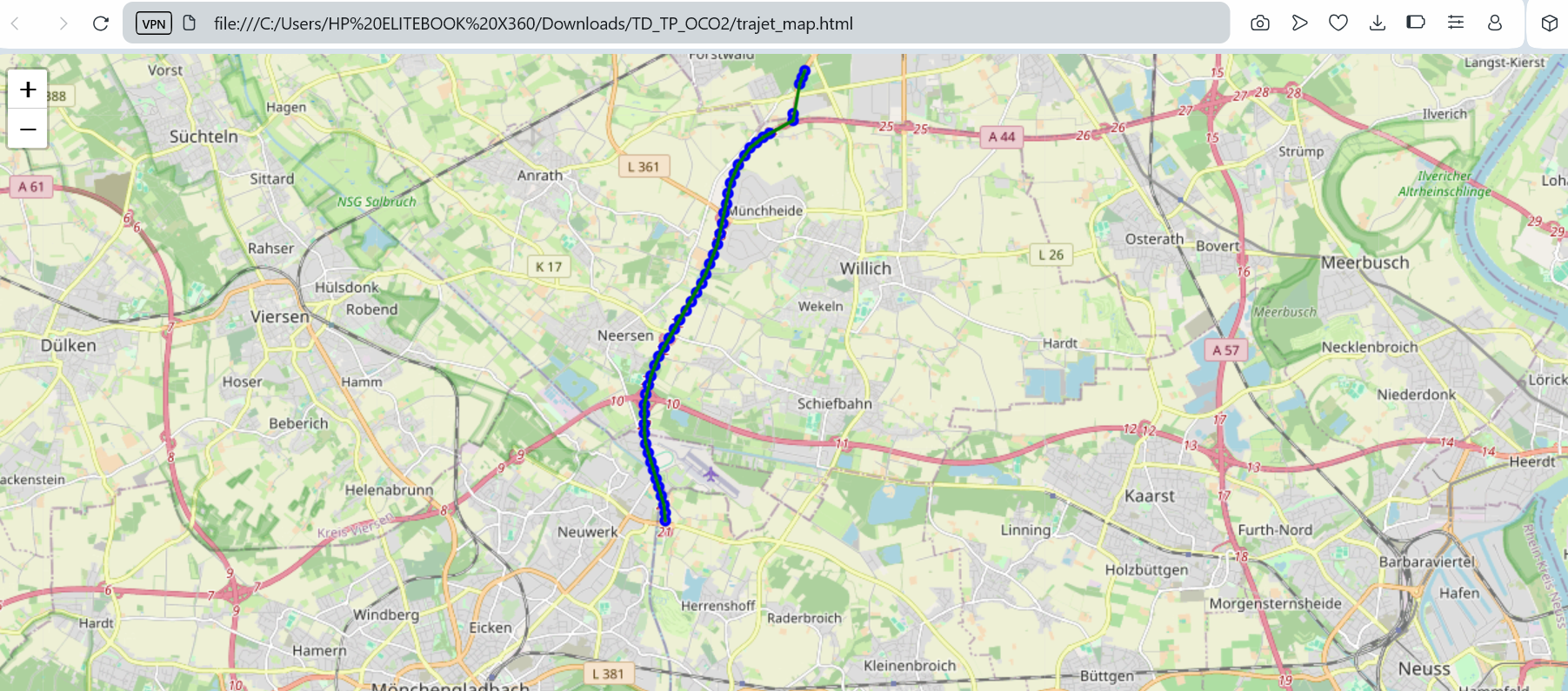
* Etape 4) Analyse prédictive et machine learning

Graphique obtenu après execution du code ml\_analysis



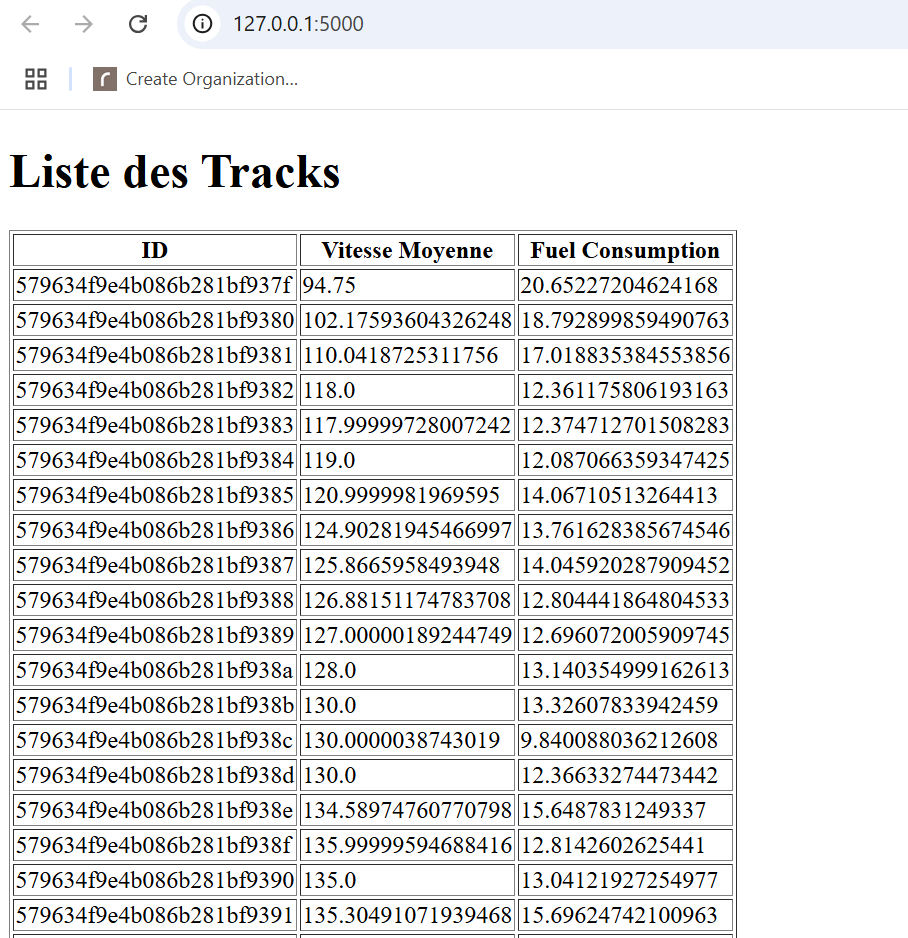
* Etape 5) Visualisation cartographique interactive

La courbe de trajet\_map.html obtenu



* Etape 6) interface web avec Flask

Après execution de flask\_app.py on obtient ce tableau



* Questions de reflexion
* **Avantages de l'architecture adoptée**

L’architecture basée sur la messagerie et le traitement asynchrone présente de nombreux avantages, notamment :

* **Découplage fonctionnel** entre les producteurs et les consommateurs de données, offrant une grande flexibilité.
* **Scalabilité facilitée**, chaque service pouvant être étendu indépendamment.
* **Robustesse accrue** face aux interruptions temporaires grâce à l’asynchronisme.
* **Facilité d’intégration** de nouveaux modules (nouveaux traitements, nouveaux visualiseurs).

Ce modèle correspond aux meilleures pratiques modernes dans les systèmes IoT et les applications cloud natives.

* **Limites par rapport à une vraie architecture cloud**

Malgré sa valeur pédagogique, cette simulation présente plusieurs limitations :

* **Pas d’autoscaling** : la charge ne s’ajuste pas dynamiquement.
* **Pas de mécanismes avancés de résilience** (ex : redondance, reprise après panne automatique).
* **Sécurité simplifiée** : l'authentification, le chiffrement des données, et les bonnes pratiques de cybersécurité ne sont que partiellement implémentés.
* **Absence de monitoring et d'alertes** : contrairement aux solutions cloud qui offrent un suivi temps réel et des alertes sur incidents.

Ainsi, la plateforme n’est pas conçue pour la production mais sert d’excellent prototype éducatif.

* **Pistes d'amélioration possibles**

Pour rendre la plateforme plus robuste et plus proche des standards industriels, plusieurs évolutions peuvent être envisagées :

* **Ajout de monitoring** avec Prometheus et Grafana.
* **Mise en place d'alertes** pour détecter rapidement les anomalies.
* **Sécurisation des communications MQTT** (certificats, TLS).
* **Utilisation d'un orchestrateur** comme Apache Airflow pour automatiser et planifier les étapes du pipeline.
* **Gestion des erreurs améliorée** (retries, dead-letter queues).
* **Enrichissement du dashboard** avec des visualisations avancées et des options multi-trajets.
* **Extension du modèle de machine learning** avec d’autres variables (ex : analyse d'efficacité énergétique, émissions CO2).