











## Hackaton Des 3M → Deuxième Place

 Technologies	Python
 Category	Data Cleaning Hackathon Project No-Code Automation
 Description	Hackathon autour d'un business case pour l'entreprise Malt, avec la participation de Mistral AI et Make pour mener à bien le projet !
 Tools Used	AI Canva Make MistralAI Pandas Python
 Analysis Methods	
 Results Summary	Real-time urban mobility optimization reducing traffic congestion by 30%
 GitHub Repository	<a href="https://github.com/Akilinoxx/Hackaton-Make-Malt">https://github.com/Akilinoxx/Hackaton-Make-Malt</a>
 Completion Date	@8 avril 2025

★ Featured	<input checked="" type="checkbox"/>
☰ Type de projet	

## Hackathon des 3M chez EugeniaSchool pour l'entreprise Malt (2ème place) coup de cœur mistral



### Contexte :

Eugenia School a organisé un hackathon en réunissant trois acteurs majeurs : **Malt**, **Mistral AI** et **Make**, avec un objectif commun : challenger les étudiants sur un business case réel de haut niveau.

Le sujet proposé par **Malt** ? **L'hyperpersonnalisation de leurs campagnes marketing.**

En combinant les outils d'IA de **Mistral AI**, les automatisations de **Make** et nos propres compétences techniques, nous avons conçu une solution complète, de

la data brute à l'activation marketing.

## Le projet en 8 étapes :

- **Nettoyage des données**
- **Scoring des contacts**
- **Développement d'une application Python**
- **Connexion à Make via Webhooks**
- **Création de scénarios Make pour l'hypermersonnalisation**
- **Connexion à une base vectorielle (RAG)**
- **Présentation orale (pitch final)**
- **Retour d'expérience & publication LinkedIn !**

## Le nettoyage des données

Malt disposait de plusieurs sources de données (site web, LinkedIn, etc.), mais celles-ci n'étaient ni uniformisées ni prêtes à être exploitées.

Nous avons donc développé un **script Python d'automatisation du nettoyage**, afin d'obtenir un fichier structuré, prêt pour des campagnes.

**Fini les Excel à la main, place à l'automatisation !**

## Le scoring des données

Une fois les données nettoyées, il fallait amorcer une **personnalisation intelligente**.

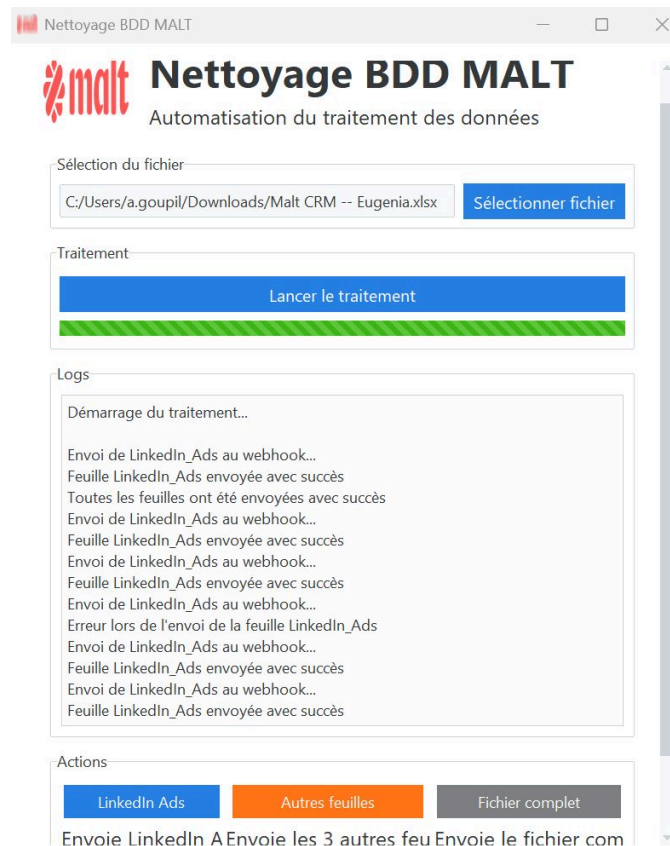
Nous avons mis en place un **système de scoring**, basé sur des critères spécifiques, pour hiérarchiser les contacts et initier une segmentation pertinente.

## L'application Python

Lancer un script via terminal, c'est pratique pour un développeur... mais pas pour un marketeur.

Nous avons donc développé une **application avec interface graphique**, permettant de glisser-déposer un fichier Excel.

➡ Résultat : le fichier est automatiquement nettoyé, scoré, puis transmis à Make pour lancer la campagne.



*Application qui exécute le nettoyage des fichiers Excel de malt. (Exécute le précédent Code Python)*

## L'activation Make

Pas besoin d'API complexe ! Nous avons utilisé des **Webhooks Make** pour connecter notre app à la plateforme.

Un simple envoi de fichier via webhook déclenche automatiquement le scénario d'hyperpersonnalisation.

(C'est cadeau 🎁)

```
def send_full_excel(self):
    # URL du webhook (service web donc scénario make) où le fichier Excel se
    webhook_url = "https://hook.eu2.make.com/5icnd5w45mm28y2rmbnp1c8t"

    try:
```

```

# Ouvre le fichier Excel en mode lecture binaire ('rb')
# Le bloc 'with' assure que le fichier sera correctement fermé après utilis
with open("Malt_DATA_processed_new.xlsx", 'rb') as excel_file:
    # Lit tout le contenu du fichier et le stocke dans la variable excel_data
    excel_data = excel_file.read()

# Prépare les en-têtes (headers) pour indiquer que c'est un fichier Exc
# Ces en-têtes aident le service web à identifier correctement le type c
headers = {
    'Content-Type': 'application/vnd.openxmlformats-officedocument.sp
    'Content-Disposition': 'attachment; filename=Malt_DATA_processed.
}

# Envoie le fichier au webhook en utilisant une requête HTTP POST
# La bibliothèque requests gère l'envoi des données
response = requests.post(
    webhook_url,
    data=excel_data,
    headers=headers
)

# Vérifie si l'envoi a réussi (code 200 = succès)
if response.status_code == 200:
    # Affiche une fenêtre de confirmation à l'utilisateur
    messagebox.showinfo("Succès", "Fichier Excel envoyé avec succès")
    # Ajoute un message de succès dans le journal de l'application
    self.log_text.insert("end", "\nFichier Excel complet envoyé avec succ
else:
    # Affiche une fenêtre d'erreur si l'envoi a échoué
    messagebox.showerror("Erreur", "Erreur lors de l'envoi du fichier Ex
    # Ajoute un message d'erreur dans le journal
    self.log_text.insert("end", "\nErreur lors de l'envoi du fichier Excel")

# Gestion des erreurs potentielles (problème d'ouverture de fichier, connex
except Exception as e:
    # Affiche une fenêtre avec le détail de l'erreur
    messagebox.showerror("Erreur", f"Erreur : {str(e)}")

```

```
# Ajoute le détail de l'erreur dans le journal
self.log_text.insert("end", f"\nErreur : {str(e)}")
```

La conversion en binaire **est essentielle** avant l'envoi au webhook pour plusieurs raisons clés :

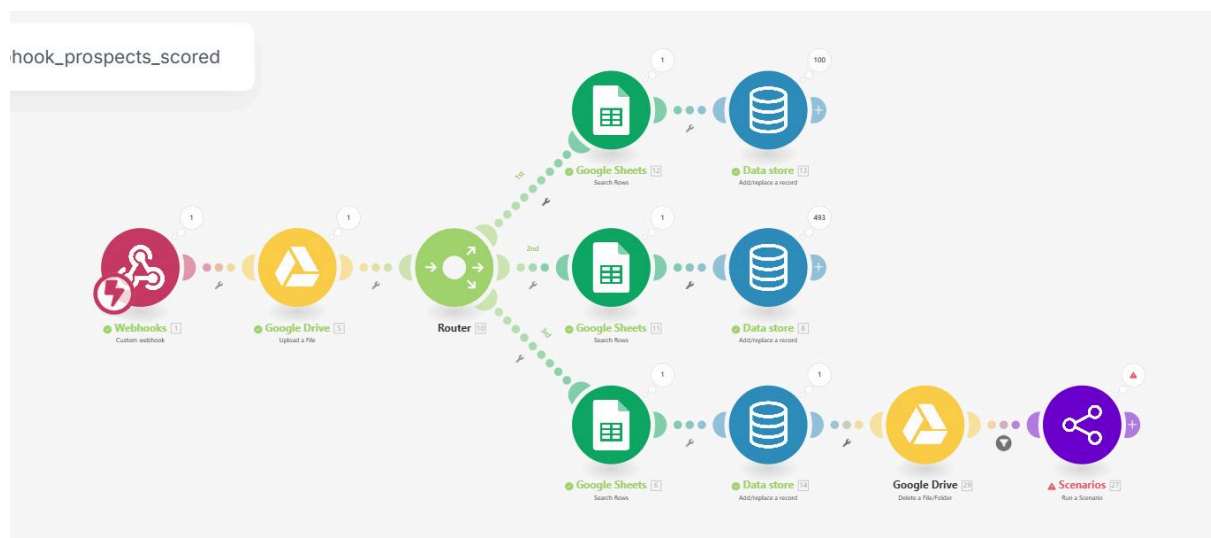
1. Un fichier Excel (.xlsx) contient des données non textuelles (compression, formats spéciaux) qui ne peuvent pas être correctement représentées en texte simple.
2. La lecture binaire ( `'rb'` ) préserve l'intégrité exacte du fichier original, incluant tous ses octets sans altération.
3. Sans cette conversion binaire, le fichier serait corrompu pendant la transmission et deviendrait inutilisable à l'arrivée.
4. Le webhook s'attend à recevoir un fichier dans son format d'origine - la lecture binaire garantit que le fichier reste identique de bout en bout du processus.

C'est comme si vous envoyiez un colis scellé plutôt que de tenter de décrire son contenu par téléphone - le destinataire reçoit exactement ce que vous avez envoyé, sans perte ni déformation.

## Les scénarios Make

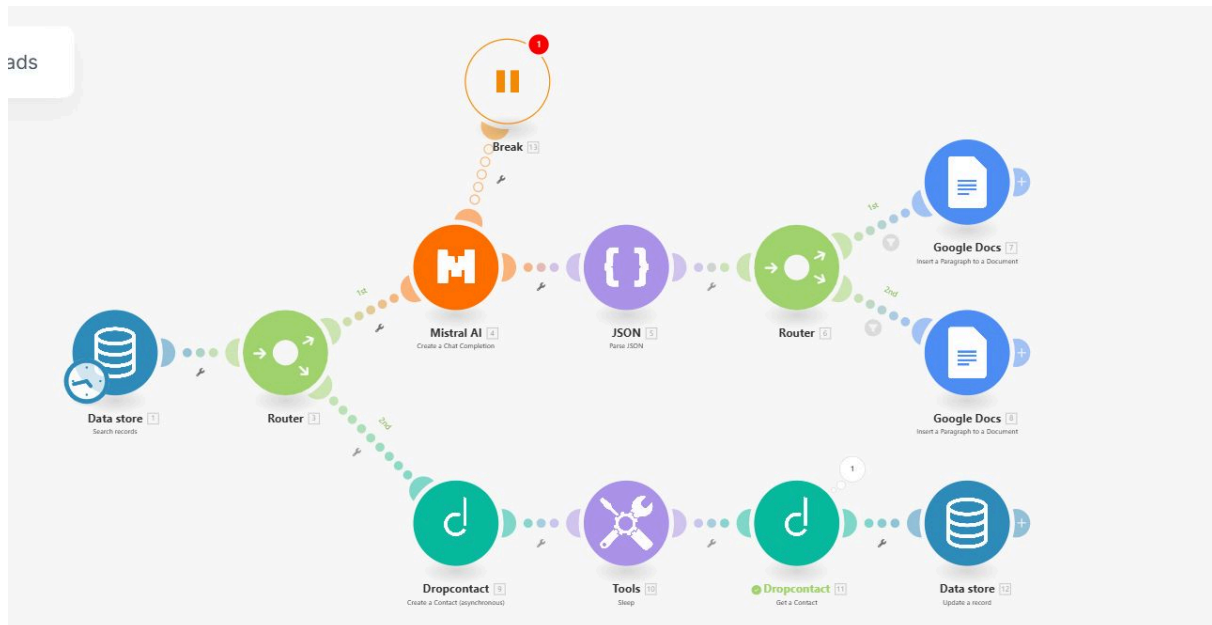
Une fois le fichier reçu, plusieurs actions sont déclenchées automatiquement :

- **Stockage et enrichissement des données**



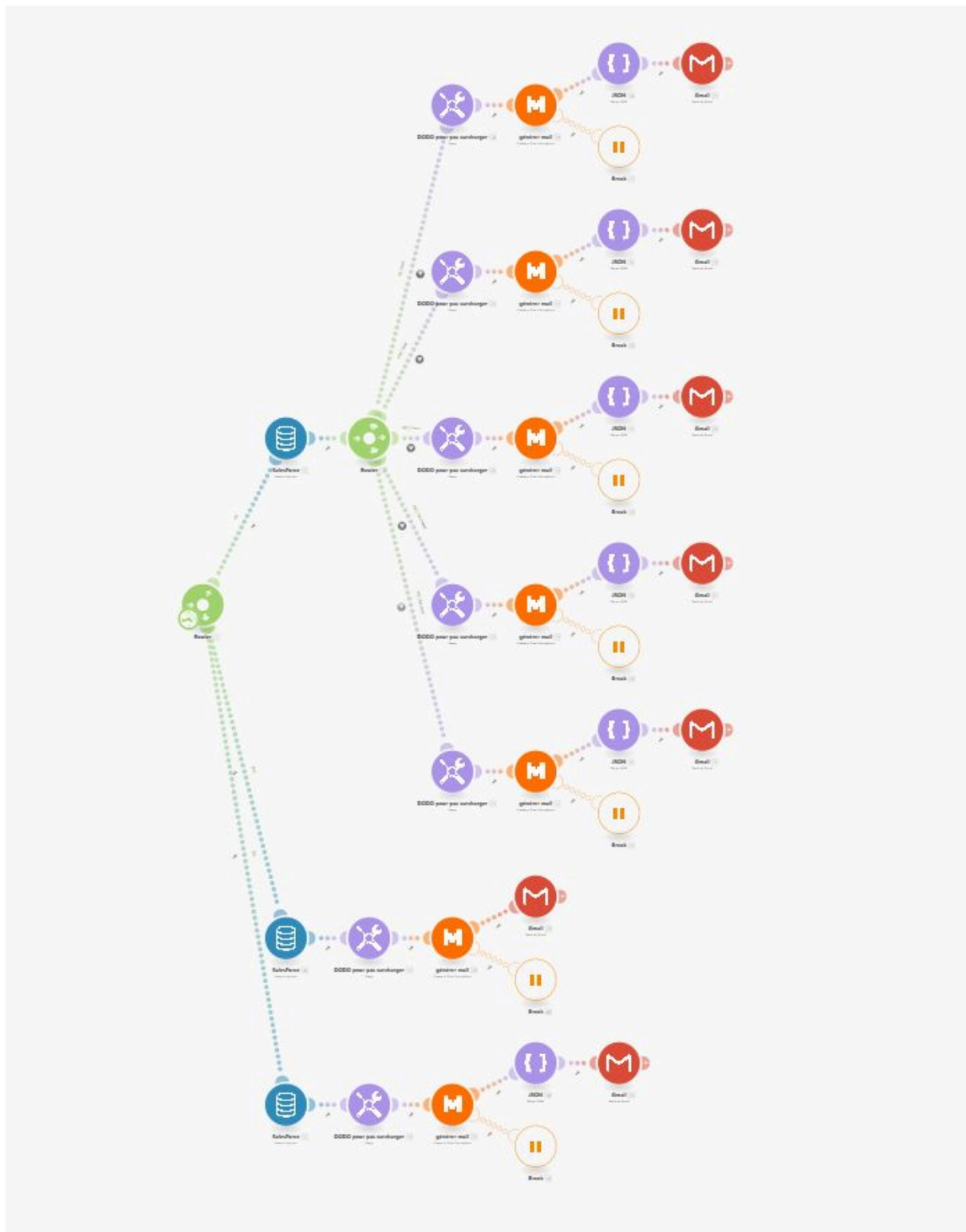
*Stockage des données nettoyées & scorées dans un CRM*

- **Campagnes LinkedIn ultra personnalisées via IA + Enrichissement**



*Nous récupérons les Urls Linkedin des campagnes & enrichissons la base grâce à des outils d'enrichissements, et nous envoyons des messages personnalisés sur linkedin (Google docs → remplaçable par linkedin)*

- **Campagnes emailing avec IA + base vectorielle (RAG)**



*Scénario Make qui effectue la campagne marketing en fonction du scoring, l'IA (MistralAI) se charge de l'hyperpersonnalisation*

## La base vectorielle RAG

Pourquoi intégrer une base vectorielle ?



Malt met en relation des freelances et des entreprises : ils disposent donc de milliers de CV.

Nous avons imaginé un moteur de recommandation intelligent qui, **selon le projet d'une entreprise**, propose automatiquement les **freelances les plus pertinents**.

Une vraie avancée pour améliorer le matching client-freelance !

## Le rendu final

Présentation du projet devant le jury : pitch technique, démonstration live et retours d'expérience.

## Ce qu'on retient

- Des apprentissages concrets (dev,data, IA, no-code, APIs...)
- Une vraie gestion de projet en équipe
- Quelques erreurs (et beaucoup d'enseignements !)
- Et un **post LinkedIn pour partager tout ça** 😊