Nettoyage des Données :

Introduction

Ce rapport présente les étapes et les techniques employées pour le nettoyage des données dans le cadre de notre projet. L'objectif principal était de traiter les incohérences, les données manquantes et les colonnes inutiles afin d'obtenir un jeu de données propre et exploitable.

Étape 1 : Suppression des données non pertinentes

- Grâce à un script automatisé, nous avons supprimé toutes les colonnes qui contenaient plus de 100 000 lignes vides afin d'optimiser la qualité des données.
- De plus, nous avons éliminé les lignes correspondant aux années antérieures à 1990. Le jeu de données final se concentre donc sur la période de 1990 à 2017.

Étape 2 : Traitement des colonnes avec incohérences

Nous avons ensuite procédé au nettoyage des colonnes présentant des incohérences ou des valeurs non logiques :

- Colonne imonth : suppression des lignes avec le mois "0" (20 lignes impactées), car un mois "0" est non valide.
- Colonne iday : suppression des lignes avec un jour "0" (892 lignes impactées), également non valide.
- Colonnes attacktype1 et attacktype1_txt : suppression des lignes vides (30 116 lignes impactées).
- Colonnes targetype1 et targetype1_txt : suppression des lignes vides (30 116 lignes impactées).

PS : les lignes vides dans les colonnes atacktype1, attacktype1_txt, targetype1, targetype1_txt sont toutes mêmes, par exemple si une ligne dans la colonne atacktype1 est vide, alors elle le sera dans les autres colonnes (d'où le 30116 lignes impacter pour ces 4 colonnes).

Étape 3 : Suppression des colonnes non informatives

Certaines colonnes n'apportaient pas d'informations pertinentes pour notre analyse. Nous les avons donc supprimées :

- Colonnes INT_LOG, INT_IDEO, INT_MISC, INT_ANY
- Colonnes crit1, crit2, crit3

Étape 4 : Gestion des valeurs négatives et inconnues

- Colonne nperps : remplacement des valeurs négatives et vides par -9 pour indiquer un nombre de terroristes inconnu.
- Colonne claimed : remplacement des valeurs vides par -9 pour indiquer l'absence d'information sur la revendication de l'attaque.

- Colonne weatype : remplacement des lignes vides par 13, identifiant pour "Unknown".
- Colonne weatype_txt : remplacement des lignes vides par "Unknown".

Résultat du nettoyage

- Fichier de départ : 135 colonnes et 181 690 lignes.
- Fichier nettoyé : 47 colonnes et 110 266 lignes.

Conclusion

Grâce à ce processus rigoureux de nettoyage, nous avons obtenu un jeu de données plus cohérent et plus facile à analyser, permettant de garantir la fiabilité des résultats obtenus lors de l'analyse des données.

Intégration des Visualisations dans l'Application Web

Étape 1 : Exportation des graphiques depuis Google Colab

Après avoir effectué le nettoyage et la préparation des données, les différentes visualisations ont été réalisées à l'aide de la bibliothèque Plotly dans un environnement Google Colab. Chaque graphique a été généré de manière interactive, puis exporté au format HTML à l'aide de la méthode fig.write_html("graph_X.html"), où X correspond au numéro ou au thème du graphique.

Ce format d'exportation a été privilégié pour sa compatibilité avec le web, permettant une intégration simple et rapide sans nécessiter de traitement supplémentaire côté client.

Étape 2 : Organisation des fichiers dans l'architecture du projet

Les fichiers HTML obtenus ont été transférés dans l'arborescence du projet web, et plus précisément dans le répertoire frontend/public/plotly. Cette structure permet un accès direct aux fichiers via une URL relative, facilitant ainsi leur affichage dans les composants Vue.js via des balises <iframe>.

Étape 3 : Intégration dans le tableau de bord Vue.js

Chaque fichier HTML correspondant à une visualisation a ensuite été intégré dans un composant Vue (Dashboard.vue) à l'aide de balises <iframe>, encapsulées dans des sections dédiées à chaque graphique. Cela permet de structurer le tableau de bord de manière lisible et interactive, tout en maintenant une séparation claire entre les données, la logique métier (Django en backend) et la présentation (Vue.js en frontend).

Étape 4 : Communication avec le backend Django

Le backend de l'application est géré avec le framework Django, utilisé principalement pour gérer la logique métier, la sécurité et l'architecture de l'application. Bien que les visualisations soient statiques (sous forme de fichiers HTML), Django permet de maintenir une structure cohérente du projet et de potentiellement étendre la logique en y intégrant des appels dynamiques ou des API REST si nécessaire.

Ce processus a permis de relier efficacement la phase de préparation et d'exploration des données avec leur restitution visuelle au sein d'une interface web. L'exportation des graphiques sous format HTML et leur intégration dans l'interface Vue.js a facilité le déploiement rapide d'un tableau de bord interactif, sans recourir à des bibliothèques complexes côté client. Cette solution garantit une visualisation fluide, tout en respectant la séparation des responsabilités entre les différentes couches de l'application.

Analyse des visuels :

Pour rédiger les analyses associées à chaque graphique, nous nous sommes appuyés sur une approche croisée mêlant l'exploitation des données brutes, la mise en perspective historique et l'analyse des grands événements géopolitiques survenus depuis 1990. À travers chaque description, nous avons cherché à dégager les grandes tendances, formuler des hypothèses explicatives, et surtout replacer les données dans leur contexte réel. L'objectif était de ne pas se limiter à une lecture purement statistique, mais d'enrichir les visualisations par des éléments concrets, des exemples marquants et parfois des récits humains représentatifs. Cette démarche permet de donner du sens aux chiffres, d'illustrer les dynamiques complexes du terrorisme mondial, et de proposer une grille de lecture à la fois analytique et sensible.