Faire du voyage en train un lieu de rencontre

Gauthier Meffe
Noa Chaze
Félix Duron
Thomas Dos
Lisa Escaron

Un **Voyage Presque Parfait** est une option applicable sur vos billets de train, ou autres moyens de transport en commun longue durée, vous permettant de rencontrer quelqu'un durant votre trajet. Après un court questionnaire sur vos centres d'intérêt et vos attentes pour votre trajet, vous serez placé à côté d'une personne remplissant vos critères, que ce soit pour une rencontre professionnelle, amicale voire amoureuse. Notre objectif : créer du lien social, rendre vos trajets plus interactifs et vous permettre de rencontrer de nouveaux profils.

Notre équipe a conçu un **pipeline ETL** (Extract, Transform, Load) récupérant les données en sortie du questionnaire utilisateurs, les transformant pour ne conserver que les informations pertinentes et les réponses complètes, et les chargeant efficacement dans une base de données. Ce rapport détaillera les choix et étapes du processus d'Extraction, de Transformation et de Chargement des données, en justifiant nos décisions.

Extraction

Nous allons nous concentrer sur une unique base de données, créée à partir de données utilisateurs recueillies via un questionnaire proposé aux clients SNCF. À l'avenir, nous pouvons imaginer l'ajout d'un formulaire de satisfaction pour améliorer le modèle d'attribution des places et ainsi perfectionner le service.

1. Données utilisateurs via un questionnaire

Dans le cadre du projet *Un Voyage Presque Parfait (VPP)*, nous avons choisi de recueillir les données directement auprès des utilisateurs via un **questionnaire rempli au moment de la réservation du billet de train**. Cette approche permet de collecter des informations précises, structurées et directement exploitables, tout en évitant les difficultés liées à l'extraction de données issues de réseaux sociaux (souvent hétérogènes, incomplètes ou peu représentatives pour certaines personnes). Si nous devions développer VPP dans le futur, suite à une première phase de fonctionnement via le questionnaire, nous essayerions de mettre un place un système de collecte de données de centres d'intérêts en demandant des accès aux réseaux sociaux pour simplifier au maximum l'expérience utilisateur. Cette deuxième phase de fonctionnement ne sera pas développée dans ce rapport.

Le questionnaire proposé comprend plusieurs dimensions essentielles pour notre objectif d'optimisation du placement des passagers :

<u>Informations demandées à chaque voyage</u> :

- **Motifs de voyage** : travail, vacances, retour au domicile, etc.
- **Attentes vis-à-vis du voisin** : recherche d'amitié, d'une relation amoureuse, d'un collaborateur, ou simplement passer un bon moment.
- **Préférences de communication** : discussions courtes et légères, échanges approfondis, ou absence de discussion.
- **Niveau d'ouverture aux nouvelles rencontres** : mesuré sur une échelle de 1 à 4.

<u>Informations enregistrées dans l'application</u> :

- **Centres d'intérêts** : sport, musique, technologies, entrepreneuriat, art.
- **Types de personnalités recherchées et auto-définition** : introverti, drôle, etc.

- **Sujets à éviter** : politique, religion, sport.
- **Langues parlées** : espagnol, français, anglais, chinois.
- **Données sociodémographiques** : âge, genre, orientation sexuelle.

Ces données, fournies volontairement par l'utilisateur au moment de sa réservation, présentent plusieurs avantages :

1. **Fiabilité et exhaustivité** : les réponses couvrent directement les dimensions sociales nécessaires au placement intelligent des passagers, contrairement aux réseaux sociaux où l'activité est variable et les informations parfois absentes.
2. **Structuration facilitée** : les questions sont fermées ou semi-fermées (choix multiples, échelles de valeur), ce qui simplifie la transformation et la modélisation des données.
3. **Respect de la vie privée** : l'utilisateur choisit explicitement ce qu'il partage, ce qui réduit les risques de collecte intrusive et augmente la confiance envers la plateforme.

En résumé, l'extraction se fait via un **formulaire intégré au processus de réservation SNCF**, séparé en deux, une partie demandée à chaque voyage, et une autre pré-enregistrée, qui alimente notre pipeline de données en informations déjà adaptées à notre cas d'usage. Les réponses aux formulaires arrivent combinées dans un fichier CSV qui sera ensuite traité dans le suite de notre processus ETL.

Nous avons décidé de nous concentrer pour le moment uniquement sur les trajets en train proposés par la SNCF. Si le projet devait grandir et la fonctionnalité s'ouvrait à d'autres compagnies de transport, nous pourrions conserver la base de données utilisateurs et juste créer une base de données opérationnelles par compagnie pour suivre les différents trajets et placement.

2. Historique et feedback

Comme évoqué précédemment, afin d'améliorer en continu la pertinence de notre modèle, nous pourrions intégrer des données issues de l'**historique des voyages effectués** :

- **Association de passagers** ayant déjà voyagé ensemble (s'ils confirment avoir apprécié leur trajet à la fin, il est évidemment possible de demander à ne plus être placé à côté d'une personne.

- **Évaluations et retours d'expérience** : chaque utilisateur peut noter son trajet et indiquer s'il a apprécié voyager aux côtés de telle ou telle personne.

- Ces feedbacks permettront d'ajuster le calcul du **score de compatibilité** pour les voyages futurs, en intégrant une boucle d'apprentissage dans notre système.

Ces données pourraient ouvrir de nouveaux champs pour enrichir notre base de données et permettre d'améliorer les duos créés lors des placements dans le train. Cette option ne sera pas développée plus dans ce rapport car elle ne concerne pas la base de données initiale.

Transformation

1. Cleaning

Nous commençons par **nettoyer** les données. Comme notre fichier CSV de données utilisateurs provient d'un questionnaire dont nous avons choisi nous même les différents champs, nous n'avons pas besoin de **sélectionner** les données car elles nous intéressent toutes.

Pour tester notre processus, nous avons créé une fonction de génération de 1000 réponses à notre formulaire avec des profils différents. Nous séparons nos données en deux fichiers différents : un fichier contenant les données spécifiques au voyage concerné (motif du voyage, attentes, …) et un fichier contenant les données générales de l'utilisateur (nom, prénom, …) que l'on veut conserver pour de prochains trajets.
Nos données sont stockées dans les fichiers *users_travels.csv* et *users_profiles.csv.*

Nous voulons nous assurer que nos données ont le bon format, nous vérifions alors :

- Le format de l'identifiant de l'utilisateur : *UUI (xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx)*
- Le format des Noms, Prénoms : *string*
- Pour la plupart des questions, le questionnaire propose plusieurs options de réponses.Le format est alors incorrect si la réponse ne rentre pas dans les options possibles.
- L'âge : on convertit la réponse au format numérique.
- Le genre : on standardise les valeurs.

Une fois les données nettoyées, nous obtenons de nouveaux fichiers de données *data_profiles_clean.csv* et *data_travels_clean.csv* que nous allons utiliser pour les étapes suivantes.

2. Aggregation

Après le nettoyage des données, il est essentiel de regrouper les informations pré-enregistrées et les données demandées à chaque nouveau voyage.
Pour ce faire nous fusionnerons simplement les bases de données sur la colonne de l'identifiant client lors de la requête de notre application.
Cette étape sera effectuée après chargement.

Chargement

Le script load_data a pour objectif d'intégrer automatiquement les données utilisateurs, préalablement nettoyées et structurées au format CSV, dans une **base de données PostgreSQL**. Plus précisément, le script lit deux fichiers : users_profiles, qui contient les informations personnelles et les préférences des utilisateurs, et users_travels, qui regroupe les données relatives à leurs voyages et attentes. Pour chaque fichier, le script établit une connexion sécurisée à la base de données grâce à **SQLAlchemy**, puis crée (ou remplace) les tables users_profiles et users_travels selon la structure des CSV. Les données sont ensuite insérées dans les tables correspondantes.

Le choix d'un **stockage local** des données utilisateurs a été privilégié afin de renforcer la sécurité et la confidentialité des informations collectées. En effet, une fois les données utilisées pour placer les personnes, celles-ci sont effacées de l'organisme de calcul. Ce mode de stockage permet de limiter la circulation des données sensibles en les conservant directement sur le terminal de l'utilisateur, réduisant ainsi les risques de fuite, de piratage ou d'accès non autorisé.
À l'inverse, une approche centralisée, telle qu'un stockage sur un serveur cloud sécurisé (par exemple via Amazon Web Services (AWS) ou Google Cloud Platform), présente l'avantage de faciliter la collecte, la synchronisation et l'analyse en temps réel d'un grand volume de données. Cette solution permet notamment de mettre à jour les modèles prédictifs de manière continue et de partager les résultats d'apprentissage entre les utilisateurs, améliorant ainsi la performance globale du système. Toutefois, elle implique une exposition accrue aux risques de cyberattaques et une dépendance à des prestataires tiers pour la protection et la gestion des données. Le choix du stockage local constitue donc un compromis orienté vers la sécurité, la confiance et la maîtrise des données personnelles,

tout en reconnaissant la pertinence du stockage cloud pour des besoins analytiques étendus et collaboratifs. À terme, une approche hybride, combinant traitement local et synchronisation sécurisée via le cloud, pourrait offrir un équilibre optimal entre confidentialité et performance.

Application

Une fois la collecte de données terminée, le service VPP analyse les données pour prédire les meilleures rencontres à encourager. Pour ce faire, il utilise non seulement les données renseignées par les utilisateurs mais aussi les informations issues des feedbacks recueillis. Un algorithme permettra enfin de quantifier le taux de compatibilité entre deux utilisateurs, qui sera la métrique utilisée pour l'attribution des sièges. Une fois le match créé, l'application se charge de réserver les places assises des utilisateurs pour qu'ils soient à côtés.

Code du projet

Lien du dossier github contenant le projet : https://githubf.com/Rasalghulsslayer/VPP

Des détails concernant l'utilisation des scripts et notre processus ETL sont présents dans le README du projet. Pour lancer le code complet avec tous les prérequis remplis, suivre le README.