Flight Catcher

Groupe D

Massil Kouchit   
massil.kouchit@etu.u-pec.fr

Yanis Ben Aied  
yanis.ben-aied@etu.u-pec.frRayane Slimani  
aissa-rayane.slimani@etu.u-pec.fr

Rayan Dansou  
yao-rayan.dansou@etu.u-pec.frAito Meignan  
aito-florentin.meignan@etu.u-pec.fr

# Introduction

Dans un monde où le transport aérien joue un rôle clé dans la connectivité mondiale, la gestion des retards de vols et l'optimisation des opérations aéroportuaires sont des enjeux majeurs. Le projet Flight Catcher vise à répondre à ces défis en développant une interface utilisateur dédiée à l’analyse des données issues du jeu de données Flight Status Prediction de Kaggle.

Ce jeu de données, qui regroupe des informations sur les vols commerciaux entre 2018 et 2022, offre une mine d'informations précieuses : retards, compagnies aériennes, aéroports, horaires, temps de vol et distances.

Le fichier a un volume total de 2 Go sous le format Parquet. Parquet grâce à sa structure orientée colonnes et ses capacités de compression avancées, permet de réduire considérablement la taille des fichiers tout en améliorant les performances d'accès aux données. Cela en fait un atout majeur pour traiter des ensembles de données volumineux tout en minimisant les ressources nécessaires.

# Mise en place d’un backend "minimal" et premières requêtes

## Choix techniques

Afin de développer l’interface d’analyse des données, nous avons opté pour l’utilisation de FASTAPI en tant que framework pour le développement de l’API côté serveur. Ce choix repose sur plusieurs avantages clés de FASTAPI, notamment sa légèreté, sa performance élevée et sa capacité à générer automatiquement une documentation des endpoints.

Pour la partie base de données, DuckDB a été choisi comme moteur analytique principal. DuckDB est particulièrement adapté pour travailler sur des fichiers volumineux, en particulier ceux au format Parquet. Son intégration native avec ce format permet d’effectuer des requêtes SQL directement sur les fichiers Parquet sans avoir à les importer dans une base de données intermédiaire. Cette approche est idéale pour notre projet, car elle réduit significativement le temps de prétraitement et limite l’utilisation de ressources mémoire.

## Chargement des données

Les données du jeu Flight Status Prediction sont stockées au format Parquet, ce qui offre des avantages significatifs en termes de compression et de performance de lecture. Ce format a été choisi en raison de sa capacité à réduire la taille des fichiers tout en maintenant un accès rapide aux données, ce qui est essentiel pour traiter les 2 Go d’informations contenues dans ce jeu de données.

Grâce à DuckDB, le fichier Parquet peut être chargé directement dans une table temporaire et interagie avec des requêtes SQL classiques. Cette méthode supprime le besoin de transformations complexes et permet de tirer parti de la simplicité des requêtes SQL pour récupérer les informations pertinentes.

## Premiers objectifs d'analyse

Une fois les données accessibles via DuckDB, nous avons défini plusieurs types de requêtes pour répondre aux principaux besoins analytiques. Ces requêtes constituent les fondations de notre backend et permettent d’extraire des statistiques utiles sur les vols commerciaux. Voici les premiers objectifs couverts par ces analyses :

Identification des compagnies aériennes : L’objectif est de récupérer la liste des compagnies présentes dans le jeu de données afin d’avoir un aperçu des acteurs impliqués.

Analyse des annulations : Le calcul du pourcentage de vols annulés a été réalisé selon deux approches :

Par année pour observer l’évolution des annulations au fil du temps.

Depuis une date spécifique pour analyser les tendances plus récentes.

Analyse des retards : De manière similaire aux annulations, l’étude des vols retardés a été menée par année et depuis une date donnée. Ces statistiques permettent d’identifier les périodes critiques où les retards sont les plus fréquents.

Classement des compagnies aériennes : Deux classements ont été mis en place :

Les 10 compagnies aériennes les plus utilisées, en fonction du nombre total de vols enregistrés.

Les 10 compagnies aériennes les plus performantes, en tenant compte à la fois du taux d’annulation et des délais moyens de départ.

Performance par État : Une analyse a été réalisée pour classer les États en fonction de leurs performances, notamment en évaluant le taux d’annulations et les retards moyens des vols.

Analyse des vols détournés : Enfin, le pourcentage de vols détournés a été étudié pour fournir une vision complète des perturbations.

Les requêtes définies avec DuckDB ont été intégrées dans le backend via FASTAPI. Chaque requête est exposée sous forme d’endpoint API, permettant ainsi à l’interface utilisateur de récupérer les données analytiques en temps réel. Cette architecture garantit une communication fluide entre le backend et l’interface utilisateur, tout en tirant parti des performances offertes par DuckDB.

# Remplacement de DuckDB par Spark

À la suite des premières phases d’analyse, certaines limites de DuckDB ont été identifiées, notamment pour le Machine Learning. Pour surmonter ces limites et répondre aux besoins en matière de scalabilité et d’efficacité, le choix a été fait de remplacer DuckDB par Apache Spark.

## Apache Spark

Apache Spark est un framework de traitement de données distribué qui offre des performances exceptionnelles pour les analyses de grandes volumétries. Contrairement à DuckDB, qui opère principalement en local, Spark permet de distribuer les calculs sur plusieurs nœuds, rendant le traitement plus rapide et plus efficace.

De plus, Spark s’avère particulièrement adapté si l’on souhaite intégrer des traitements de Machine Learning dans le futur. Grâce à sa bibliothèque MLlib, Spark propose des outils performants pour l’entraînement et l’exécution de modèles prédictifs sur des données massives.

L’un des principaux avantages de Spark réside dans son support natif des fichiers au format Parquet, tout comme DuckDB. Cette compatibilité a facilité la migration des requêtes existantes vers Spark sans nécessiter de transformations des données. De plus, Spark dispose d’une API SQL très performante, qui nous a permis de conserver une approche basée sur des requêtes structurées tout en bénéficiant de temps de réponse améliorés sur les analyses complexes.

## Nouvelles analyses réalisées avec Spark

Avec Spark, nous avons pu enrichir notre ensemble de requêtes analytiques en introduisant de nouvelles analyses plus avancées. Parmi celles-ci, on retrouve :

L’analyse des retards par heure de départ : Cette étude permet de déterminer les heures où les retards sont les plus fréquents, offrant ainsi une meilleure compréhension des périodes critiques pour les vols.

Le classement des aéroports en fonction de leur performance : Cette analyse classe les aéroports en tenant compte de leur taux de retards et d’annulations. Elle permet d’identifier les infrastructures les plus performantes et celles qui nécessitent des améliorations.

L’étude de la corrélation entre les retards au départ et à l’arrivée : Cette analyse met en lumière les relations entre les retards enregistrés au départ des vols et les retards constatés à l’arrivée, afin de mieux comprendre les facteurs d’impact.

La distribution des vols par distance parcourue : Cette analyse segmente les vols en plusieurs catégories (courts, moyens et longs trajets) afin d’observer la répartition des distances dans le jeu de données.

La prédiction des retards futurs : Grâce aux capacités d’apprentissage automatique intégrées à Spark, nous avons posé les bases d’un modèle prédictif qui prend en compte des variables comme l’heure de départ, l’origine, la destination et la distance parcourue pour anticiper les retards.

Identify applicable funding agency here. If none, delete this text box.

## Adaptation du backend avec Spark

L’intégration d’Apache Spark dans le backend développé avec FASTAPI a nécessité plusieurs ajustements. Une configuration spécifique de Spark a été mise en place pour assurer une lecture optimale des fichiers Parquet et permettre l’exécution rapide des nouvelles requêtes. Chaque analyse a ensuite été exposée sous forme d’endpoint API, afin de garantir une communication fluide entre le backend et l’interface utilisateur.

# Développement du frontend avec Dash Plotly

Pour le développement du frontend, nous avons sélectionné le framework Dash Plotly, qui constitue une surcouche de React.js pour Python. Dash facilite la création d’applications web interactives pour la visualisation de données, tout en permettant d’intégrer des librairies analytiques comme Pandas pour le traitement des données.

L’interface utilisateur a été conçue pour interagir directement avec le backend FASTAPI, offrant ainsi un tableau de bord interactif capable d’afficher les résultats des requêtes sous forme de tableaux ou de graphiques.

L’application repose sur une structure simple où chaque bouton permet de lancer une requête API spécifique et d’afficher les résultats dans un composant graphique ou tabulaire. Par exemple :

Affichage de la liste des compagnies aériennes : Lorsqu’un utilisateur clique sur le bouton correspondant, l’application récupère les données via un endpoint API et génère un tableau interactif grâce aux composants de Dash.

Calendrier des vols annulés : Une visualisation sous forme de carte thermique (heatmap) est générée pour représenter le pourcentage de vols annulés par mois et par année.

Enfin, Dash Bootstrap Components a été intégré pour assurer un design soigné et une expérience utilisateur optimale grâce à des styles modernes basés sur Bootstrap.

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors’ names; do not use “et al.”. Papers that have not been published, even if they have been submitted for publication, should be cited as “unpublished” [4]. Papers that have been accepted for publication should be cited as “in press” [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

# tests de performances

# scénarios d'application

Bibliographie

1. https://www.kaggle.com/datasets/robikscube/flight-delay-dataset-20182022
2. https://fastapi.tiangolo.com/
3. https://duckdb.org/
4. https://spark.apache.org/
5. https://dash.plotly.com/