Rapport d'Évaluation Sommatif UA3 : Analyse de Données Massives avec PySpark

Projet : Analyse de la performance et des retards des vols nationaux américains en 2020.

Ce rapport détaille la démarche de modélisation, de transformation et d'analyse exploratoire réalisée sur un jeu de données massif de vols, en utilisant le framework Apache Spark via son API Python (PySpark).

1. Présentation et Nettoyage du Jeu de Données

1.1 Titre et Source

Métrique	Détail
Titre	US National Flight Data 2020
Source	Kaggle (Données initialement issues du Bureau of Transportation Statistics – BTS)

1.2 Description des Variables du Dataset

Variable	Description Simple	Utilisation Clé
YEAR	Année du vol (2020).	Contexte temporel.
MONTH	Mois du vol (1 à 12).	Analyse de la saisonnalité.
DAY_OF_MONTH	Jour du mois (1 à 31).	Analyse quotidienne.
DAY_OF_WEEK	Jour de la semaine (1 = Lundi, 7 = Dimanche).	Analyse du trafic hebdomadaire (Q1).
OP_UNIQUE_CARRIER	Code unique du transporteur aérien (ex: OO, YV).	Identifier la performance des compagnies (Q1).
ORIGIN_CITY_NAME	Nom de la ville de départ (ex: New York).	Analyse d'itinéraire.

ORIGIN_STATE_ABR	Abréviation de l'État de départ (ex: NY).	Analyse régionale.
DEST_CITY_NAME	Nom de la ville d'arrivée.	Analyse d'itinéraire.
DEST_STATE_ABR	Abréviation de l'État d'arrivée.	Analyse régionale.
CRS_DEP_TIME	Heure de départ prévue (format HH:MM:SS ou HHMM).	Base pour l'analyse du retard par créneau horaire (Q2).
DEP_DELAY_NEW	Nouveau retard au départ en minutes (0 si vol à l'heure/en avance).	Calcul de la ponctualité.
CRS_ARR_TIME	Heure d'arrivée prévue.	Comparaison avec l'heure réelle pour le calcul du retard à l'arrivée.
ARR_DELAY_NEW	Nouveau retard à l'arrivée en minutes (0 si vol à l'heure/en avance).	Métrique de performance principale.
CANCELLED	Indicateur d'annulation (1 = Oui, 0 = Non).	Calcul du taux d'annulation (Q1).
CANCELLATION_CODE	Code de la raison de l'annulation (A, B, C, D).	Analyse des causes (Q3).
AIR_TIME	Temps de vol réel en minutes.	Calcul de la vitesse moyenne.
DISTANCE	Distance entre l'origine et la destination en miles.	Analyse d'itinéraire et de la typologie des vols.

1.3 Contenu, Format et Résumé Statistique

Le jeu de données se présente sous forme de fichier CSV unique. L'analyse est basée sur **4 316 997 lignes de données** couvrant l'intégralité de l'**année 2020**.

Métrique	Valeur (Réelle)	Observation Clé (Basée sur l'output summary)
Nombre total de lignes	4 316 997	Volume important, nécessitant une architecture distribuée.
Période Couverte	2020	Contexte atypique (pandémie), impactant le trafic.

Retard Moyen au Départ (DEP_DELAY_NEW)	**6.62 minutes**	Retard moyen relativement bas.
Taux d'Annulation (CANCELLED)	**6.42%**	Calculé à partir de la moyenne de la colonne binaire (0.0642).
Distance Moyenne	**777.68 miles**	Confirme une prédominance des vols court-moyen-courriers.

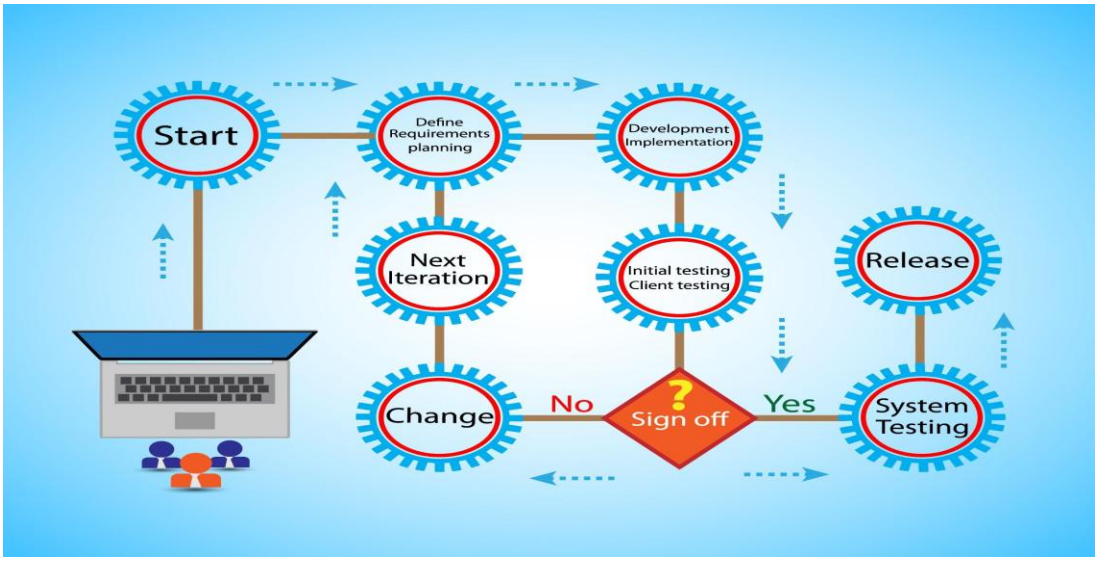
1.4 Nettoyage de Base Appliqué

L'étape de nettoyage a été vitale pour assurer la qualité du DataFrame :

- **Valeurs Manquantes :** Le dénombrement initial a confirmé **0 valeur manquante** pour toutes les colonnes après le processus d'ingestion (gestion des chaînes "NULL").
- **Doublons : 20 doublons exacts** ont été identifiés dans le jeu de données.
- **Action :** Ces 20 doublons ont été **supprimés**, ramenant le nombre de lignes distinctes à **4 316 977**.
- **Correction des Types :** Des transformations ont été appliquées pour convertir les colonnes de retard en types numériques et pour extraire l'heure de départ prévue (CRS_DEP_TIME - StringType) vers un format numérique (CRS_DEP_HOUR_CLEAN - IntegerType), essentiel pour l'analyse temporelle.

2. Explication de l'Architecture Spark Utilisée

Apache Spark s'appuie sur une architecture distribuée



1. **Driver Program (Programme Pilote) :** Contient la logique de l'application (le code PySpark). Il crée le SparkSession et le plan d'exécution (DAG), puis coordonne la distribution des tâches.
2. **Cluster Manager (Gestionnaire de Cluster) :** Alloue les ressources physiques (mémoire, CPU) sur les nœuds de travail.
3. **Executors (Exécuteurs) :** S'exécutent sur les nœuds esclaves. Ils effectuent les tâches de traitement des données, stockent les résultats intermédiaires en mémoire et communiquent les résultats au Driver.

Cette architecture permet l'exécution massivement parallèle des traitements, assurant une performance optimale sur les grands jeux de données.

3. Choix de Modélisation et Justification

3.1 Utilisation Principale des DataFrames

Le projet a utilisé **majoritairement les DataFrames PySpark** pour la modélisation et la manipulation des données.

Justification	Description Détaillée
Abstraction de Haut Niveau	L'API est intuitive (similaire à SQL ou Pandas), ce qui facilite le développement rapide et la maintenance du code.
Optimisation Automatique	**Avantage Décisif :** Le **Catalyst Optimizer** et l'exécution **Tungsten** optimisent le plan de requête (par exemple, en poussant les filtres avant la lecture des données), garantissant une **performance supérieure** aux RDD.
Schéma Structuré	Le DataFrame permet une gestion plus efficace des types de données et une détection des erreurs au moment de la planification de la requête.

3.2 Comparaison Partielle avec les RDD

L'utilisation des RDD (Resilient Distributed Datasets) a été écartée pour les données tabulaires de ce projet, car ils offrent un niveau d'abstraction plus faible, exigent une gestion manuelle de la sérialisation, et ne bénéficient pas de l'optimisation automatique du Catalyst Optimizer. Les RDD ne restent pertinents que pour des scénarios avancés (données non structurées, bibliothèques externes).

4. Transformations et Actions Appliquées

Question Métier Illustrative :

Retard Moyen par Transporteur et Jour de la Semaine

L'analyse de cette question requiert un minimum de 5 transformations et 3 actions, démontrant le concept d'exécution paresseuse (lazy execution) de Spark.

Type	#	Transformation/ Action	Description
Transformation	1	filter	Sélectionner uniquement les vols ayant un retard réel (DEP_DELAY_NEW_CLEAN > 0).
Transformation	2	select	Réduire le DataFrame aux colonnes Transporteur, Jour de la semaine, et Retard.
Transformation	3	groupBy	Regrouper les vols par Transporteur et Jour de la semaine.
Transformation	4	agg(avg(...), round(...))	Calculer le retard moyen au départ pour chaque groupe et l'arrondir.
Transformation	5	orderBy	Trier le résultat final par retard moyen décroissant.
Action	1	show()	Afficher les 10 premières lignes du résultat (force l'exécution).

Action	2	count()	Compter le nombre total de lignes dans le DataFrame résultant.
Action	3	take(5)	Récupérer les 5 premières lignes comme objets Python (pour traitement local).

Résultat de l'Analyse Illustrative (Top 3 des Pires Retards)

Transporteur	Jour de la Semaine	Retard Moyen (Minutes)	Total Vols Retardés
YV	4 (Jeudi)	**62.82**	3 542
G4	7 (Dimanche)	**59.79**	4 068
OO	6 (Samedi)	**55.01**	10 224

5. Résultats et Interprétation des Analyses Métier

Q1 : Performance Globale des Transporteurs (Fiabilité vs. Retard)

Cette analyse compare la performance des transporteurs sous deux angles : la **Ponctualité** (mesurée par le Retard Moyen) et la **Fiabilité** (mesurée par le Taux d'Annulation).

Indicateur	Transporteur	Valeur Obtenue	Interprétation Clé
Moins Ponctuel (Retard Max)	**OO (SkyWest)**	**53.68 min**	Ce transporteur régional subit le retard moyen le plus élevé, probablement dû à un effet de cascade dans son réseau de correspondances, affectant un très grand nombre de vols (82 485).

Plus Ponctuel (Retard Min)	**9E**	**46.87 min**	Parmi les 5 principaux transporteurs en retard, 9E (Endeavor Air) affiche la meilleure gestion des retards moyens.
Plus Fiable (Annulation Min)	**NK (Spirit)**	**2.31%**	Avec le taux d'annulation le plus bas, Spirit (NK) se distingue par une excellente fiabilité structurelle, bien en deçà de la moyenne du dataset (6.42%).
Moins Fiable (Annulation Max)	**YX (Republic)**	**5.14%**	YX (Republic) est classé comme le moins fiable en termes d'annulations parmi les Top 5, bien que son score reste raisonnable.

Q2 : Distribution du Retard par Heure de Départ

L'analyse de la distribution des retards par heure (CRS_DEP_HOUR_CLEAN) révèle un profil de dégradation de la ponctualité au cours de la journée.

Heure de Départ (HH)	Retard Moyen (Minutes)	Tendance Observée
05h	**51.88**	**Pic Inattendu.** Contrairement à l'attente, les vols très matinaux (souvent des vols de nuit en retard de la veille ou des vols sensibles aux vérifications techniques précoces) affichent les retards les plus élevés.
09h - 14h	31.71 à 33.44	Période de relative stabilité et meilleure ponctualité de la journée.
18h - 22h	36.03 à 38.92	Remontée des retards en fin de journée (heures de pointe).

Note : Le retard moyen général des vols en retard (environ 30-50 minutes) est plus élevé que le retard moyen global du dataset (6.62 minutes), car cette analyse filtre uniquement les vols **qui sont réellement en retard**.

Q3 : Analyse des Causes d'Annulation

L'analyse de la colonne CANCELLATION_CODE_CLEAN révèle la répartition des 277 176 vols annulés :

Code	Nom de la Cause	Nombre d'Annulations	Proportion (%)
D	**Sécurité (Security)**	**236 777**	**85.42%**
A	Transporteur (Carrier)	18 291	6.60%
B	Météo (Weather)	17 979	6.49%
C	Système National Aérien (NAS)	4 129	1.49%

Interprétation :

- **Prédominance du Code D :** Le code **D (Sécurité)** est la cause écrasante des annulations en 2020.
- **Contexte 2020 :** Ce résultat est une anomalie statistique due au contexte de la pandémie. Il est probable que la catégorie "Sécurité" ait été utilisée par le Bureau of Transportation Statistics (BTS) pour englober toutes les annulations liées aux **restrictions de voyage, aux fermetures de routes aériennes, ou à la faible demande** résultant de la crise sanitaire, des causes non traditionnellement codées A, B ou C. Les causes opérationnelles (A, B, C) ne représentent ensemble moins de 15% des annulations.

6. Limites du Projet et Pistes d'Amélioration

Limites Actuelles du Projet

1. **Biais Temporel (Année 2020) :** Les conclusions sur la performance des vols sont fortement influencées par la faible demande due à la pandémie et ne sont pas généralisables aux années standards.
2. **Granularité Temporelle :** L'extraction de l'heure de départ prévue (HH) simplifie l'analyse (Q2). Une modélisation plus précise impliquerait de convertir le temps prévu en secondes pour une métrique plus fine.
3. **Format de Stockage Non-Optimisé :** Le travail a été réalisé sur des fichiers CSV. L'utilisation de ce format ne permet pas de bénéficier des optimisations de lecture/écriture, des schémas d'indexation et de la compression offertes par des formats natifs à Spark comme Parquet ou ORC.

Pistes d'Amélioration et d'Extension

1. **Stockage Optimisé (Delta Lake) :** Convertir les données en format **Delta Lake** pour exploiter le potentiel de Spark dans un environnement de production (gestion des transactions ACID, versioning des données).
2. **Analyse Prédictive :** Utiliser la librairie **MLlib de Spark** pour développer un modèle de classification prédictive (ex: Régression Logistique ou Forêt Aléatoire) capable de prédire la probabilité de retard ou d'annulation d'un vol donné.
3. **Exploitation des RDD :** Introduire un scénario pertinent pour les RDD, tel que le traitement du langage naturel (NLP) sur les notes des codes d'annulation (si disponibles) pour démontrer leur flexibilité avec les données non structurées.
4. **Analyse Géo-Spatiale :** Corréler les données de vol avec des données météorologiques externes (fichiers JSON ou Parquet) pour analyser plus précisément l'impact de la météo sur la cause d'annulation "B", en utilisant les coordonnées des villes de départ et d'arrivée.