RUSH 3 — Satisfaction client (compagnie aérienne)
Analyse exploratoire et modélisation prédictive

Rapport final — Python / Google Colab

Équipe (3 personnes)	Membre 1 : **Lamiae Zriouali** Membre 2 : **Abdessalam Lahlou** Membre 3 : **Yassine Thyfa Filali**
Promo / Cours	**MSC RENTREE DECALEE**
Date	10.02.2026
Environnement	Google Colab (Python)
Livrables	Notebook (.ipynb) + .Rapport + Figures

Sommaire

Résumé exécutif

L'objectif du projet est de prédire la satisfaction des passagers (satisfait vs non satisfait) et d'identifier les facteurs qui influencent le plus cette satisfaction, afin d'en déduire des recommandations concrètes. Les données sont fournies sous forme de deux fichiers : un jeu d'entraînement (103 904 lignes, 25 colonnes) et un jeu de test (25 976 lignes, 25 colonnes).

L'analyse exploratoire montre des écarts marqués de satisfaction selon le type de voyage, la classe et la fidélité. Côté prédiction, une régression logistique intégrée dans un pipeline scikit-learn obtient des performances solides sur le test (accuracy 0,87 ; AUC 0,9256). L'interprétation des coefficients confirme que le contexte « business/loyal » et la qualité de service augmentent la probabilité d'être satisfait, tandis que le voyage personnel, l'absence de fidélité, la classe économique et les retards à l'arrivée la diminuent.

Conformité aux attendus

D'après le brief du Rush 3, les livrables attendus sont : (i) un code produisant une prédiction via une méthode de régression ou de KNN, et (ii) un rapport écrit justifiant les choix d'EDA et les variables explicatives. Ce projet répond à ces attendus :

- Modèle : régression logistique (classification binaire) avec pipeline de préparation.
- EDA : analyses de distribution, valeurs manquantes, segments, drivers (écarts de moyennes).
- Rapport : synthèse + résultats + interprétation + recommandations + annexes.

1. Introduction

Ce projet vise à analyser la satisfaction client dans le transport aérien et à construire un modèle de classification capable d'anticiper cette satisfaction à partir d'informations passager, de contexte de voyage et de notes de service. L'intérêt est double : produire un modèle performant et dégager des leviers d'amélioration de l'expérience client.

2. Environnement de travail

Langage : Python.

Environnement : Google Colab.

Bibliothèques principales : pandas, numpy, matplotlib, scikit-learn.

3. Données

Train : 103 904 lignes, 25 colonnes. Test : 25 976 lignes, 25 colonnes.

Variable cible : satisfaction (satisfied / neutral or dissatisfied).

Encodage binaire : 1 = satisfied ; 0 = neutral or dissatisfied.

Répartition (train) : 0 ≈ 56,67 % ; 1 ≈ 43,33 %.

4. Préparation et nettoyage

4.1 Chargement robuste et contrôles

Lors de l'import, une première ligne invalide a perturbé la lecture standard de train.csv. La lecture a été rendue robuste en sautant la première ligne (skiprows=1), en imposant les noms des colonnes et en tolérant les lignes problématiques (engine='python', on_bad_lines='skip'). Après import, on vérifie systématiquement train.shape et test.shape.

4.2 Nettoyage

Colonnes supprimées : id et Unnamed: 0 (identifiants, pas de valeur explicative).

Valeurs manquantes : présence de manquants dans Arrival Delay in Minutes. Traitement via imputation dans le pipeline (médiane).

5. Analyse exploratoire (EDA)

5.1 Distribution de la cible

Objectif : vérifier l'équilibre des classes afin d'interpréter correctement les résultats. La distribution est relativement équilibrée (≈ 56,7 % non satisfaits, ≈ 43,3 % satisfaits).

5.2 Valeurs manquantes

Objectif : repérer les colonnes à traiter avant modélisation. Seule Arrival Delay in Minutes présente des manquants significatifs ; les autres colonnes sont complètes.

5.3 Satisfaction par segments

Méthode : taux de satisfaction (%) = (nombre de satisfaits / total) × 100.

- Type of Travel : Business travel ≈ 58,3 % vs Personal Travel ≈ 10,2 % (Voir Figure 1);
- Class : Business ≈ 69,4 % vs Eco Plus ≈ 24,6 % vs Eco ≈ 18,6 % (Voir Figure 2).
- Customer Type : Loyal ≈ 47,7 % vs disloyal ≈ 23,7 % (Voir Figure 3).

Figure 1 : satisfaction (%) par Type of Travel, Class, Customer Type.

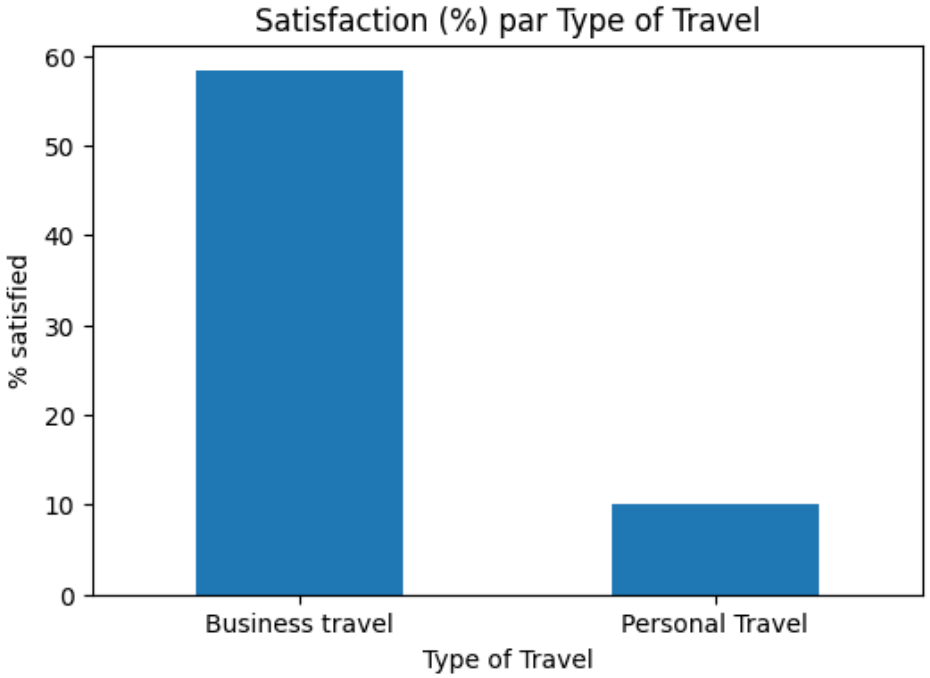


Figure 2 : Satisfaction (%) par Classe.

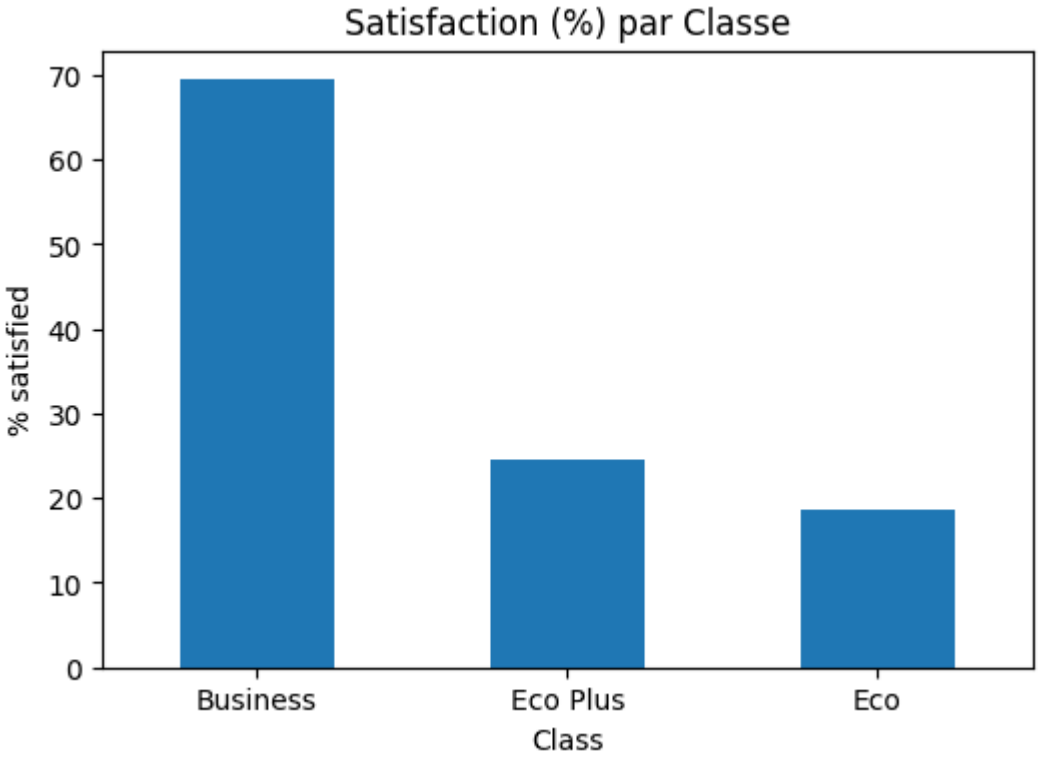
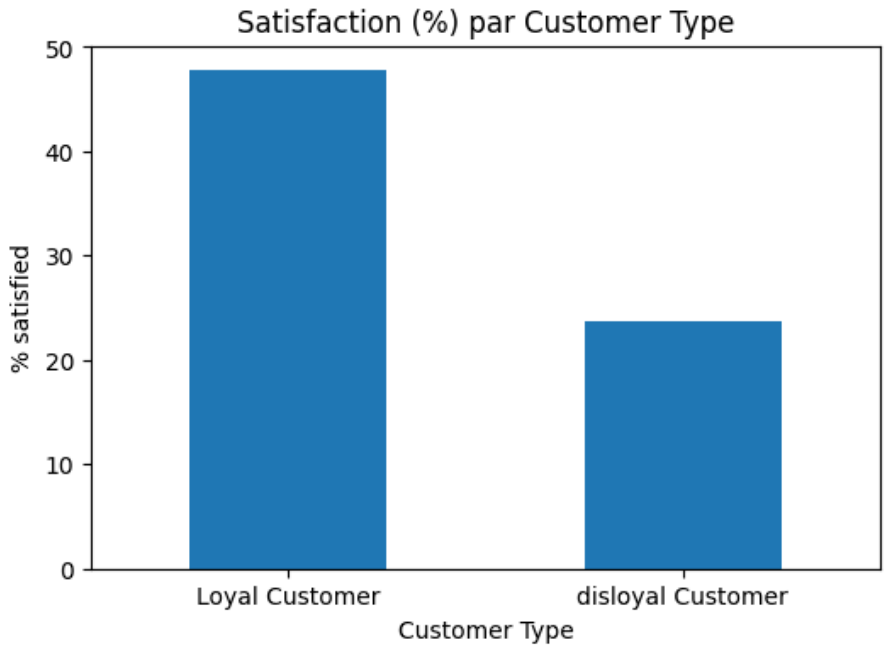


Figure 3 : Satisfaction (%) par Custumer Type.

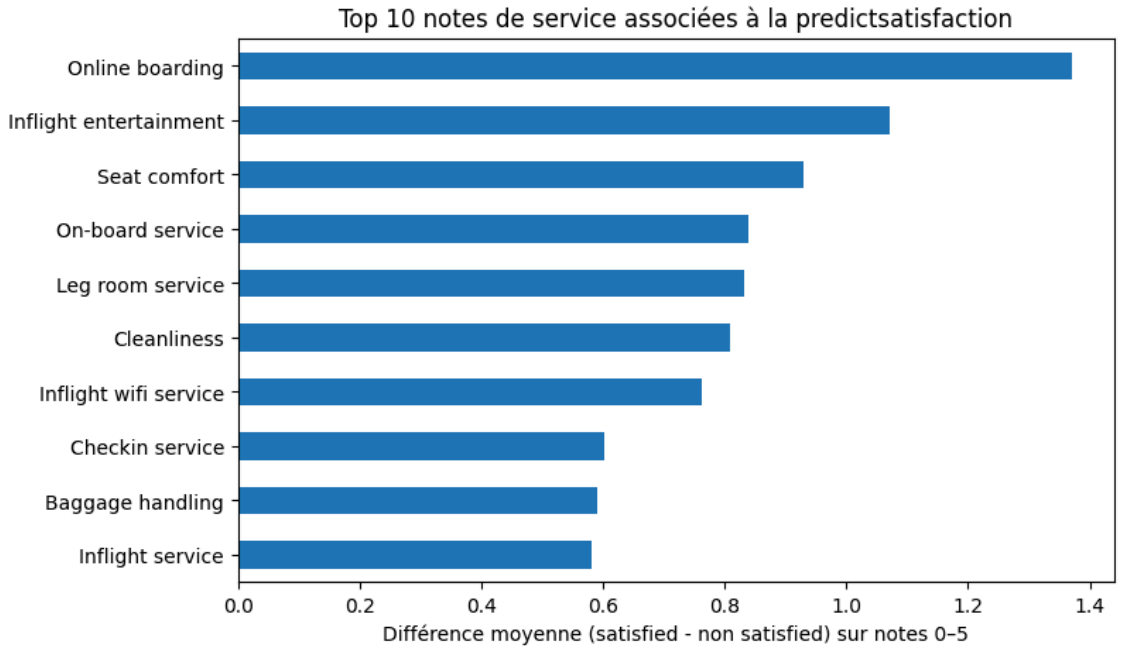


5.4 Drivers sur les notes de service (0–5)

Méthode : comparer la moyenne des notes entre satisfaits et non satisfaits pour chaque variable de service, via Δ = moyenne(satisfaits) − moyenne(non satisfaits), puis trier les écarts.

Les plus grands écarts concernent : Online boarding, Inflight entertainment, Seat comfort, On-board service, Leg room service, Cleanliness, Inflight wifi service, Checkin service, Baggage handling, Inflight service (Voir Figure 4).

Figure 4 : Top 10 des écarts sur les notes de service.



6. Modélisation

6.1 Choix du modèle

Une régression logistique est utilisée comme modèle de référence : adaptée à la classification binaire, rapide, et interprétable.

6.2 Pipeline de préparation (scikit-learn)

Un pipeline garantit un traitement identique sur train et test et évite les erreurs liées aux valeurs manquantes.

- Numériques : SimpleImputer(médiane) + StandardScaler.
- Catégorielles : SimpleImputer(valeur la plus fréquente) + OneHotEncoder.
- Modèle : LogisticRegression(max_iter=2000).

7. Résultats

Accuracy : 0,87. AUC : 0,9256.

Matrice de confusion : [[13145 1428], [1901 9502]].

Interprétation : bonne détection des non satisfaits ; davantage de faux négatifs (satisfaits prédits non satisfaits), donc un modèle légèrement prudent.

8. Interprétation des coefficients

En régression logistique, un coefficient positif augmente la probabilité d'être satisfait (à variables constantes), un coefficient négatif la diminue.

8.1 Top facteurs négatifs

- Type of Travel = Personal Travel : -1,677
- Customer Type = disloyal Customer : -1,336
- Class = Eco Plus : -0,539
- Class = Eco : -0,418
- Gender = Female : -0,339
- Arrival Delay in Minutes : -0,336
- Ease of Online booking : -0,200
- Departure/Arrival time convenient : -0,190
- Age : -0,126

8.2 Top facteurs positifs

- Type of Travel = Business travel : +1,040
- Customer Type = Loyal Customer : +0,700
- Online boarding : +0,826
- Inflight wifi service : +0,524
- Checkin service : +0,410
- On-board service : +0,388

- Leg room service : +0,333
- Class = Business : +0,321
- Cleanliness : +0,290
- Baggage handling : +0,158

9. Recommandations

Optimiser l'embarquement en ligne — Simplifier le parcours, réduire les frictions, stabiliser l'expérience digitale.

Renforcer l'expérience à bord — Divertissement, confort siège, espace jambes, propreté et service à bord.

Fiabiliser le wifi — La connectivité ressort comme un facteur significatif de satisfaction.

Réduire les retards à l'arrivée — Les retards augmentent fortement le risque d'insatisfaction.

Renforcer la fidélisation — Avantages ciblés et parcours client privilégié pour les clients fidèles.

10. Limites et pistes d'amélioration

- Tester d'autres modèles (KNN, arbres, ensembles) pour comparaison.
- Ajouter une validation croisée (cross-validation) pour renforcer la robustesse.
- Étudier les corrélations entre notes de service (variables potentiellement redondantes).

Conclusion

Le modèle atteint une accuracy de 0,87 et une AUC de 0,9256, avec une interprétation cohérente avec l'EDA. Les leviers majeurs sont le contexte de voyage (business vs personal), la fidélité, la classe, et la qualité de service (online boarding, wifi, confort, propreté, check-in).

Annexes

Annexe A — Explication simple du notebook

Cette annexe explique, en termes simples, ce que fait chaque bloc de code et pourquoi il existe. Elle permet de refaire le travail sans IA, en suivant la logique.

Imports : On importe os (vérifier les fichiers), pandas/numpy (données), matplotlib (graphiques), scikit-learn (modèle).

Lecture des CSV : pd.read_csv charge train.csv et test.csv. On vérifie immédiatement train.shape et test.shape.

Correction d'import (train) : Si une première ligne est invalide, on force header=None, names=cols, skiprows=1 et on_bad_lines='skip'.

Nettoyage : On retire id et Unnamed: 0, car ce sont des identifiants non explicatifs.

Cible binaire : On transforme satisfaction en 0/1 : 1 si 'satisfied', sinon 0.

EDA — distribution : value_counts et value_counts(normalize=True) pour comprendre l'équilibre des classes.

EDA — valeurs manquantes : isna().sum() pour localiser les colonnes avec NaN.

EDA — segments : crosstab(..., normalize='index') pour obtenir des % de satisfaits par catégorie.

EDA — drivers notes : On calcule Δ moyenne(satisfaits) − moyenne(non satisfaits) pour les notes 0–5, puis on trace un barplot.

Pipeline & entraînement : ColumnTransformer applique des traitements différents aux colonnes numériques et catégorielles, puis LogisticRegression apprend.

Prédictions & métriques : predict / predict_proba, classification_report, confusion_matrix, roc_auc_score.

Coefficients : On récupère les noms OneHot + les coefficients, puis on trie pour obtenir les plus grands positifs/négatifs.

Annexe B — Formules et définitions

- Taux de satisfaction (%) = (nombre de satisfaits / total) × 100.
- Driver notes (0–5) : Δ = moyenne(satisfaits) − moyenne(non satisfaits).
- Accuracy = (TN + TP) / total.
- AUC = capacité à séparer les classes via les probabilités (0,5 hasard → 1 parfait).
- Régression logistique : coefficients interprétables (positif → augmente la probabilité d'être satisfait).

Annexe C — Librairies

- pandas : lecture et manipulation (DataFrames), calculs (crosstab, value_counts).
- numpy : opérations numériques (tableaux, tri), utile pour assembler noms/coefficients.
- matplotlib : graphiques (bar charts).
- scikit-learn : preprocessing (imputation, scaling, OneHot), pipeline, modèle, métriques.