# 

# 

# PROJET DE FIN D’ETUDE 5IAE

Flight Booking Chatbot

Ce chatbot est utilisé pour fournir des prix de vol précis en fonction des informations fournies par l'utilisateur.

### Contributeurs :

* Tientcheu Ngaleu Larry Jordan
* Kameni Kameni Olivier
* Ej-Jaidi Aymane

Table des matières

[I. Structure du Projet 3](#_Toc167628790)

[II. Backend 3](#_Toc167628791)

[a. Modèle d'IA 3](#_Toc167628792)

[b. Flight\_prices\_prediction.ipynb (détails) : 3](#_Toc167628793)

[c. Model.ipynb 5](#_Toc167628794)

[d. Model\_eval.ipynb 5](#_Toc167628795)

[III. API Backend 5](#_Toc167628796)

[IV. Frontend 7](#_Toc167628804)

[V. Instructions pour exécuter 7](#_Toc167628805)

# Structure du Projet

Le projet se compose de deux parties principales : un backend et un frontend.

# Backend

Le backend est l'épine dorsale de ce projet. Il comprend le modèle d'IA utilisé pour prédire les prix des vols et une API utilisée pour interagir avec ce modèle et enregistrer les résultats dans une base de données.

# Modèle d'IA

Pour construire le modèle d'IA, nous avons suivi trois processus majeurs. Chacun de ces processus est bien défini dans ses fichiers `.ipynb` respectifs.

##### Flight\_prices\_prediction.ipynb :

##### Ce fichier contient tout le travail de traitement des données que nous avons effectué. Nous y réalisons l'évaluation des données, le nettoyage des données et les visualisations pour obtenir des insights précieux.

##### Model.ipynb :

##### Ici, nous entraînons notre modèle d'IA sur les données prétraitées obtenues du fichier précédent. Nous entraînons trois modèles et, en fonction des résultats, nous choisissons le meilleur des trois.

##### Model\_eval.ipynb :

##### Ce notebook permet de tester notre modèle pour s'assurer que tout fonctionne comme prévu.

# Flight\_prices\_prediction.ipynb (détails) :

#### L'objectif de cette section est de générer des prix de vols basés sur les informations fournies dans le jeu de données flight.csv. Nous traitons et analysons le jeu de données, et fournissons à la fin un nouveau jeu de données avec des prix de vol précis basés sur divers facteurs.

Évaluation des données :

* Vérification des problèmes de qualité et de structure.
* Observation que le jeu de données ne contient que des vols pour l'année 2015 et uniquement pour le territoire américain (vols domestiques).
* Identification des informations nécessaires : numéro d'identification unique de la compagnie aérienne, date du vol, aéroport d'origine, aéroport de destination, distance du vol, départ prévu.

Nettoyage des données :

* Suppression des informations inutiles.
* Formatage de date/l'heure de départ/prévue et des différentes dates dans une seule entité AAAA-MM-JJ et HH:mm:ss.
* Conversion des distances de miles en kilomètres.
* Calcul du coût de chaque vol basé sur le prix moyen des aéroports d'origine et le prix par kilomètre.

# Model.ipynb

#### Nous y entraînons trois modèles : régression linéaire, forêt aléatoire, réseau de neurones. Entre ces trois modèles, nous choisissons le meilleur, qui est le réseau de neurones en raison de sa meilleure performance. Le modèle est nommé `nn\_model.pkl` et est sauvegardé au format pickle dans le fichier `tmodels`.

# Model\_eval.ipynb

#### Ce notebook évalue notre modèle de réseau de neurones entraîné. Ce notebook teste le modèle sur une entrée personnalisée pour s'assurer que tout fonctionne bien.

#### Le modèle d'IA est complété par quelques fonctions auxiliaires dans le fichier `helpers/helpers.py`.

#### Ces fonctions sont utilisées pour interagir avec le modèle d'IA.

# API Backend

### C'est une API Flask qui fournit des routes pour interagir avec le modèle en fournissant des prompts et en retournant une réponse.

### Cette API comporte des fonctionnalités notables comme :

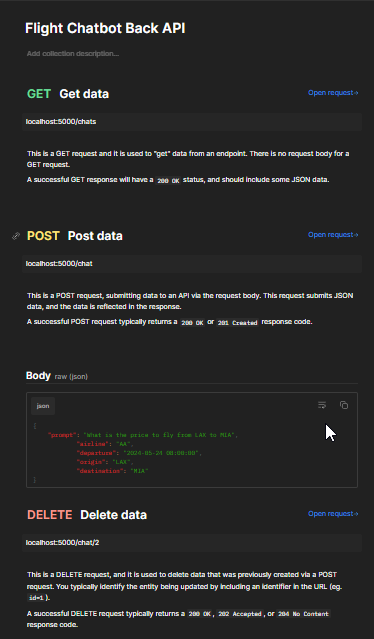
### L'API est construite en utilisant Flask, un framework de développement web en Python.

### PostgreSQL comme système de gestion de base de données.

### Le code peut être trouvé dans le dossier `backend`, dans le fichier `program.py`.

### Cette API prend en compte la migration de la base de données et utilise SQLAlchemy comme mapper objet-relationnel.

### L'API se compose de trois routes :



La réponse pour la requête POST est un JSON comme suit :

{

    "price": "660.51324",

    "prompt": {

        "airline": "AA",

        "destination\_airport": "MIA",

        "id": 8,

        "origin\_airport": "LAX",

        "prompt": "I want to fly from LAX to MIA on 24 May at 8 am",

        "scheduled\_departure": "Fri, 24 May 2024 08:00:00 GMT"

    },

    "success": true

}

Le chargement du modèle pour la prédiction est mis en cache. Le chargement réel n'est effectué que la première fois que la requête est traitée.

Des instructions précises sur la manière d'exécuter l'API backend se trouvent dans le fichier `Readme.md`.

# Frontend

Inclut les fichiers HTML, CSS et JavaScript nécessaires pour l'interface utilisateur.

**Fonctionnement du Chatbot**

* **Chargement des Données** : Les prompts et les réponses sont chargés à partir d'un fichier JSON.
* **Prétraitement des Données** : Les textes des prompts et des réponses sont tokenisés et les séquences sont rembourrées pour les rendre compatibles avec le modèle de deep learning.
* **Vérification des Étiquettes** : Les étiquettes des réponses sont vérifiées pour s'assurer qu'elles sont dans la plage valide et corrigées si nécessaire.
* **Modèle de Deep Learning** : Un modèle séquence à séquence (seq2seq) est défini et entraîné. Il utilise une couche d'embedding pour représenter les mots et une couche LSTM pour traiter les séquences.
* **Entraînement du Modèle** : Le modèle est entraîné sur les séquences de prompts et de réponses.
* **Génération de Réponses** : Lorsque l'utilisateur envoie une question, le modèle génère une réponse basée sur les prompts appris.

**Interface Utilisateur**

* **HTML et CSS** : La page HTML utilise Bootstrap pour le style et une barre latérale pour afficher les prompts.
* **JavaScript** : Gère les interactions utilisateur, envoie les messages au serveur Flask et affiche les réponses du chatbot dans la fenêtre de chat.

**Routes Flask**

* **Page d'accueil (/)** : Affiche la page HTML principale avec l'interface utilisateur.
* **Obtenir une réponse (/get\_response)** : Reçoit la question de l'utilisateur, utilise le modèle de deep learning pour générer une réponse, et renvoie cette réponse à l'utilisateur.

# Instructions pour exécuter (confère le Readme)