Documentation Test Pratique- Novalitix AI Lab

Table des matières

I. Description du projet

Ce projet est un projet de test fourni par NOVALITIX dans le but d'évaluer les compétences d'un candidat nommé X en intelligence artificielle (apprentissage automatique, science des données, etc.). Le projet comprend deux tests : un test de prédiction des prix des maisons en fonction de certaines caractéristiques, et un second test pour l'analyse des critiques de films afin de déterminer si une critique est positive ou non.

Le projet est divisé en trois principales parties : une partie web appelée "HousePricePredictor", une API nommée "api", et la création des modèles dans la partie "Jupyter_code".

1. La partie Web(HousePricePredictor)

La partie Web de ce projet correspond à l'interface utilisateur accessible via un navigateur web. Elle permet aux utilisateurs d'interagir avec les différents modèles et fonctionnalités du projet. Sur la page d'accueil de l'interface Web, vous trouverez une description des deux tests proposés ainsi que des liens vers les projets correspondants. Par exemple, le premier projet concerne la prédiction des prix des appartements, tandis que le second projet traite de l'analyse des critiques de films. Cette partie est développée en utilisant la technologie Vue.js.

2. L'API(api)

L'API est le composant qui expose les endpoints permettant à l'interface Web de communiquer avec les différentes fonctionnalités du projet. Elle est développée en utilisant Django L'API permet à l'interface Web de consommer les données et les fonctionnalités

nécessaires pour les tests de prédiction des prix des appartements et d'analyse des critiques de films.

3. Code des modèles (Jupyter_code)

Cette partie du projet inclut des scripts contenant les algorithmes et les étapes de création et d'entraînement des modèles. Le code des modèles est écrit en Python.

II. Création des modèles

1. Processus de création du modèle de régression linéaire

Après avoir effectué l'étape exploratoire des données et sélectionné les attributs pertinents (RM, LSTAT et PTRATIO), nous subdivisons le jeu de données en deux parties : les données d'entraînement (80 %) et les données de test (20 %).

1. Les données d'entraînement sont utilisées pour entraîner le modèle, tandis que les données de test sont utilisées pour évaluer le modèle.
2. Ensuite, nous instancions un objet de la classe LinearRegression de scikit-learn et nous entraînons cette instance avec les 80 % de notre jeu de données.
3. Nous évaluons les performances du modèle en utilisant les 20 % restants de données de test.
4. Une fois satisfait des performances de notre modèle, nous sauvegardons cette instance à l'aide de la bibliothèque joblib.

2. Création du modèle d’analyse des sentiments

Ce modèle est créé en utilisant l'architecture LSTM Il comprend plusieurs couches :

1. La première couche est une couche d'embedding qui permet de représenter les mots ou les tokens sous forme de vecteurs denses.
2. Ensuite, une couche LSTM est ajoutée pour capturer les dépendances à long terme dans la séquence d'entrée.
3. Une couche Dense avec une fonction d'activation sigmoïde est utilisée pour obtenir une sortie binaire (0 ou 1) pour la classification.
4. Pour réduire le risque de surapprentissage, une couche Dropout est ajoutée pour désactiver aléatoirement certains neurones pendant l'entraînement.
5. Enfin, le modèle est compilé avec l'optimiseur Adam, la fonction de perte binaire_crossentropy.

Une fois le modèle créé et les données prétraitées, nous divisons le jeu de données en deux parties. Dans un premier temps, nous utilisons une répartition de 80 % pour l'entraînement

et de 20 % pour les tests. Ensuite, nous subdivisons à nouveau le jeu de données d'entraînement en un ensemble d'entraînement proprement dit et un ensemble de validation. L'ensemble de validation est utilisé pour évaluer si le modèle souffre de surapprentissage ou de sous-apprentissage, et permet d'ajuster les poids des neurones pendant l'entraînement. Une fois que le modèle est bien entraîné, nous le sauvegardons et l'exposons via l'API.

III. Description des processus déploiement en local

Pour lancer le backend (projet django), créer un environnement virtuel , activer cet environnement virtuel et installer les dépendances du projet se trouvant dans le requirements.txt avec la commande pip install -r requirements.txt

Pour lancer le frontend (projet vuejs +vite), se positionner dans **HousePricePredictor,** exécuter la commande ***npm install*** pour installer toutes les dépendances du projet et ensuite exécuter ***npm run dev*** pour lancer le projet.