DOCUMENTATION

1. Présentation

Notre pipeline DataOps sert à récupérer, nettoyer données quantitatives et qualitatives d'un dataset et ensuite créer un modèle à partir de ces données pour enfin le convertir en format json.

Le dataset utilisé (titanic.csv) contient les données sur les passagers du Titanic le jour de son naufrage.

Pour la mise en place de cette pipeline plusieurs technologies ont été nécessaires :

- Python (langage informatique)
- Pandas (Pandas est une bibliothèque écrite pour le langage de programmation Python permettant la manipulation et l'analyse des données)
- urllib.request (urllib est un paquet qui collecte plusieurs modules travaillant avec les URLs : urllib.request pour ouvrir et lire des URLs)

2. Documentation du code

Les fonctions du pipeline sont écrites dans le fichier pipeline.py et ces fonctions sont appelées et exécutées dans main.py.

NB : La documentation se trouve dans les commentaires du code

Pipeline.py

```
"""
Documentation du Code

Ce code est une série de fonctions qui permettent de charger des données à partir d'une URL, de nettoyer les données,
d'extraire un modèle de données spécifique et de les convertir en format JSON.

"""

import numpy as np
import urllib.request
import pandas as pd
import json

# Fonction pour charger les données depuis une URL
def requestUrl(url):
    """
    Charge les données à partir d'une URL CSV.

    Parameters:
    url (str): L'URL du fichier CSV.

    Returns:
    pd.DataFrame or None: Un DataFrame pandas contenant les données CSV si le chargement réussit, sinon None.
    """
    try:
        urldata =  urllib.request.urlopen(url)
        data = pd.read_csv(urldata)
        return data
    except Exception as e:
        print(f"Erreur de chargement de données : {e}")
        return None
```

```
  # Fonction pour nettoyer les données

def transform(data):
    """
    Nettoie les données en remplaçant les valeurs manquantes par des valeurs pardéfaut.

    Parameters:
    data (pd.DataFrame): Le DataFrame pandas contenant les données brutes.

    Returns:
    pd.DataFrame: Un DataFrame pandas contenant les données nettoyées.
    """
```

```
# Copiez les données pour éviter de modifier les données originales
    cleaned_data = data.copy()

    # Remplacez les valeurs manquantes par la médiane de la colonne 'Age'
    cleaned_data['Age'].fillna(cleaned_data['Age'].median(), inplace=True)

    #Remplacer les valeurs manquantes par la valeur la plus fréquemment observée dans
la dataset
    Most_frequent_port = cleaned_data['Embarked'].mode()[0]
    cleaned_data['Embarked'].fillna(Most_frequent_port, inplace=True)
```

```
    return cleaned_data

# Fonction pour extraire les données en fonction d'un modèle prédéfini
def extract_model(data):
    """
    Extrait un modèle de données spécifique à partir du DataFrame.

    Parameters:
    data (pd.DataFrame): Le DataFrame pandas contenant les données brutes.

    Returns:
    list of dict: Une liste de dictionnaires représentant les données extraites.
    """
    selected_columns = ["Sex", "Pclass", "Age", "Survived", "Fare", "Embarked"]
    data_subset = data[selected_columns]

    # Renommez la colonne "Fare" en "price" et "Pclass" en "Class"
    data_subset.columns = ['sex', 'class', 'age', 'survived', 'price', 'embarked']
    donnees_passager = data_subset.to_dict(orient="records")
    return donnees_passager
```

```
# Fonction pour convertir les données en JSON
def load(data):
    """
    Convertit les données en format JSON et les enregistre dans un fichier.

    Parameters:
    data: Les données à convertir en JSON.

    Returns:
    None
    """
    try:
```

```
        with open('passenger.json', 'w') as file:
            json.dump(data,file)
        print('Création du fichier réussi !')
    except Exception as e:
        print(f"Erreur : {e}")
        return None
```

Main.py

```
from PipelineData import requestUrl , transform , load , extract_model
```

```
# Chargement de la donnée
Data=requestUrl("https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv")

# Néttoyage de données
data_cleaned = transform(data)

# Extraire le model
donnees_passager = extract_model(data_cleaned)

# Création du fichier json
load(donnees_passager)
```