Projet MLops: Predict Future Sales

EL HARRAK Safouane

PIRO Younes

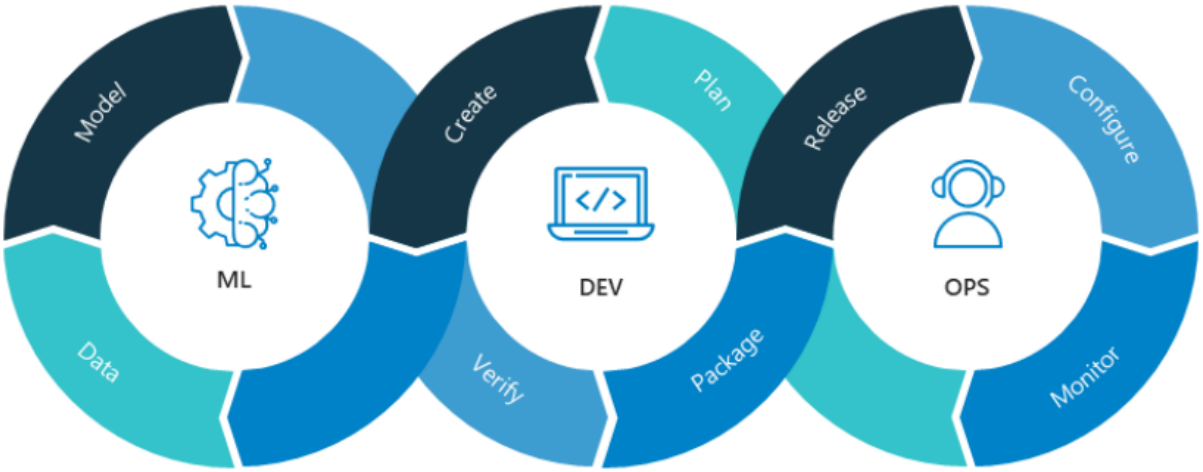
Encadré par : Mr Lotfi EL AACHAK

MLOps :Machine Learning Operations

1- Définition :

MLOps est un ensemble de pratiques visant à intégrer et automatiser la création, le déploiement et la maintenance de modèles de machine learning en production. Cela implique une collaboration entre les scientifiques de données et les professionnels de l'informatique et l'utilisation d'outils tels que le contrôle de version, l'intégration continue et les pipelines de déploiement. L'objectif de MLOps est d'améliorer l'efficacité du processus de machine learning et de faciliter la collaboration entre les différentes équipes.

2- Architecture MLOps :



- **ML**

ML (abréviation de machine learning) est un type d'intelligence artificielle qui permet aux ordinateurs d'apprendre et d'améliorer leurs performances sur une tâche spécifique sans être explicitement programmés. Les algorithmes d'apprentissage automatique analysent les données et utilisent les informations obtenues à partir de ces données pour faire des prédictions ou prendre des mesures.

- **DEV**

Dev (abréviation de développement) fait référence au processus de création ou de conception de quelque chose de nouveau, comme une application logicielle ou un modèle d'apprentissage automatique..

- **OPS**

Ops (abréviation d'opérations) fait référence aux processus et activités impliqués dans la gestion et la maintenance des systèmes et de l'infrastructure. Dans le contexte de MLOps, les opérations font référence aux processus et activités impliqués dans la gestion et la maintenance de l'infrastructure et des systèmes nécessaires pour exécuter des modèles d'apprentissage automatique en production.

3- Pourquoi utilisé Mlops ?

- Efficacité améliorée : les pratiques et les outils MLOps peuvent aider à automatiser et à rationaliser les différentes étapes du cycle de vie de l'apprentissage automatique, ce qui facilite et accélère la création, le déploiement et la maintenance des modèles d'apprentissage automatique.

- Collaboration renforcée : MLOps implique une collaboration entre les scientifiques des données et les professionnels de l'informatique, ce qui peut aider à améliorer la communication et la coordination entre ces équipes et faciliter le partage des connaissances et des ressources.

- Meilleures performances des modèles : en intégrant la surveillance et les tests dans le processus d'apprentissage automatique, les MLOps peuvent aider à garantir que les modèles fonctionnent comme prévu et peuvent être rapidement mis à jour ou améliorés si nécessaire.

- Sécurité renforcée : les pratiques MLOps peuvent aider à garantir que les modèles d'apprentissage automatique et l'infrastructure sur laquelle ils s'exécutent sont sécurisés et conformes aux normes et réglementations de l'industrie.

- Évolutivité améliorée : les pratiques et outils MLOps peuvent aider les organisations à faire évoluer plus facilement leurs efforts d'apprentissage automatique, leur permettant de déployer et de gérer davantage de modèles dans des environnements de production.


predict future sales

Introduction :

L'objectif de ce projet est de développer un modèle d'apprentissage automatique capable de prédire avec précision les ventes futures d'un magasin, sur la base de données de ventes historiques quotidiennes. La tâche consiste à prévoir la quantité totale de produits vendus dans chaque magasin pour l'ensemble de test. Notez que la liste des boutiques et des produits change légèrement chaque mois. Créer un modèle robuste capable de gérer de telles situations fait partie du défi. En prédisant avec précision les ventes futures, les magasins peuvent mieux planifier et allouer les ressources, optimiser les stratégies de tarification et de marketing et améliorer les revenus globaux.

Objectif du Projet :

Les principaux objectifs de ce projet sont de :

- ✓ Développez un modèle d'apprentissage automatique capable de prédire avec précision les ventes futures en fonction des données historiques et d'autres facteurs pertinents.
- ✓ Automatisez le processus de création, de déploiement et de maintenance du modèle d'apprentissage automatique dans un environnement de production.
- ✓ Surveillez les performances du modèle et effectuez les ajustements nécessaires pour vous assurer qu'il continue de fonctionner avec précision au fil du temps.
- ✓ Collaborez avec des scientifiques des données et des professionnels de l'informatique pour vous assurer que le modèle est intégré de manière transparente dans les systèmes et processus existants de l'entreprise.

Outils et Technologies :

Python est un langage de programmation interprété, multi-paradigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet.

Keras est une bibliothèque open source puissante et facile à utiliser pour créer et former des modèles d'apprentissage en profondeur en Python. Il est conçu pour être convivial et modulaire, ce qui facilite le prototypage, la construction et la formation de modèles à l'aide de divers backends (tels que TensorFlow, Microsoft Cognitive Toolkit et Theano).

Pandas est une bibliothèque écrite pour le langage de programmation Python permettant la manipulation et l'analyse des données. Elle propose enparticulier des structures de données et des opérations de manipulation de tableaux numériques et de séries temporelles.

NumPy est une bibliothèque pour langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi quedes fonctions mathématiques opérant sur ces tableaux.

FastAPI est un framework Web moderne, rapide et facile à utiliser pour créer des API avec Python. Il est basé sur des conseils de type Python standard et génère automatiquement la documentation OpenAPI. FastAPI est efficace et léger, et il prend en charge WebSockets pour une communication en temps réel.

Docker est un outil qui utilise des conteneurs pour faciliter le déploiement et l'exécution d'applications dans divers environnements.

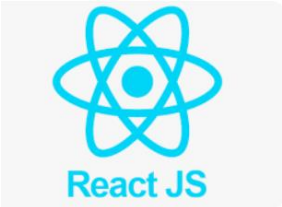
MongoDB est un système de gestion de base de données orienté documents, répartissable sur un nombre quelconque d'ordinateurs et nenécessitant pas de

schéma prédéfini des données.

Kubernetes est un système open source permettant d'automatiser le déploiement, la mise à l'échelle et la gestion des applications conteneurisées. Il regroupe les conteneurs en unités logiques et facilite la gestion et la découverte de ces applications.

Uvicorn est une implémentation de serveur ASGI hautes performances à utiliser avec les frameworks ASGI tels que FastAPI. Il dispose d'une interface de ligne de commande et prend en charge le rechargement à chaud.

TensorFlow est une bibliothèque de logiciels open source pour l'apprentissage automatique et l'intelligence artificielle. Il est largement utilisé pour la formation et le déploiement de modèles d'apprentissage automatique et dispose d'API disponibles dans plusieurs langues.

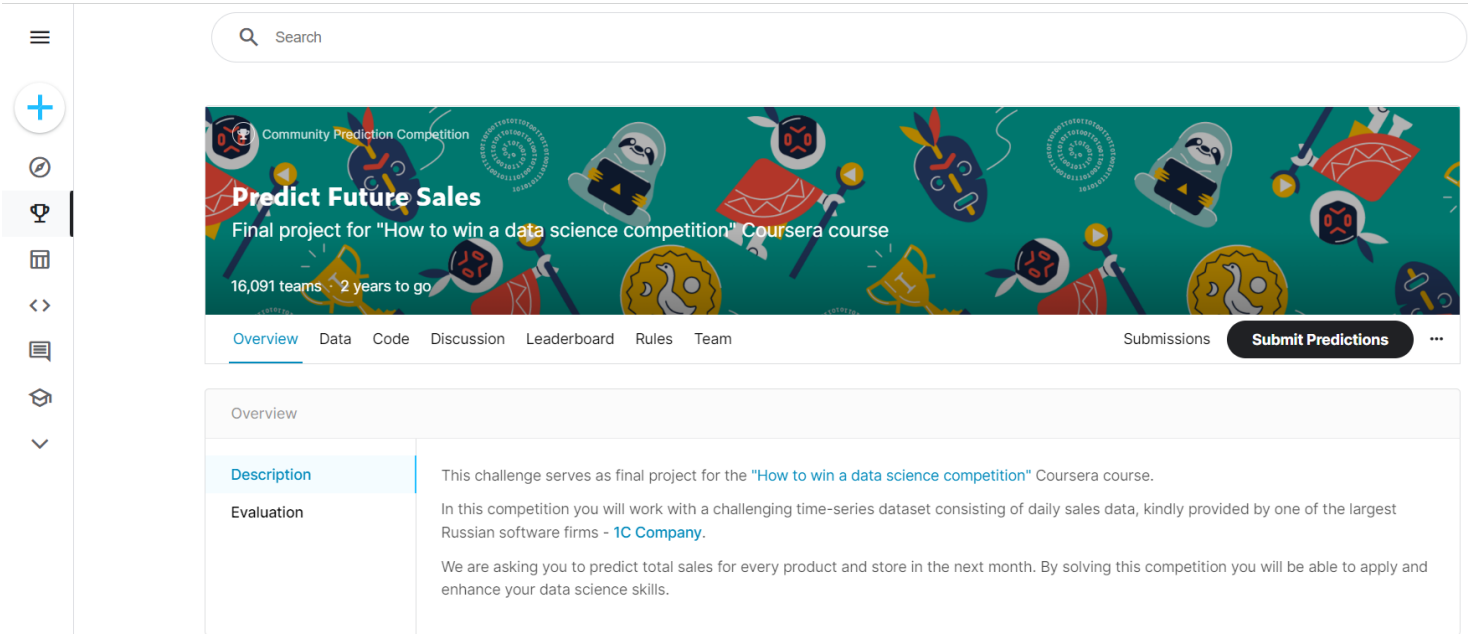


React est une bibliothèque JavaScript pour créer des interfaces utilisateur. Il a été développé par Facebook et est souvent utilisé pour créer des applications d'une seule page et des applications mobiles.

1- Kaggle Compétition : Predict Future Sales

Predict Future Sales est un concours d'apprentissage automatique hébergé sur Kaggle qui vise à prédire le montant total des ventes d'une entreprise pour chaque produit et magasin au cours du mois prochain. Le concours utilise un ensemble de données fourni par l'entreprise qui comprend des données de ventes quotidiennes pour un certain nombre de produits et de magasins, ainsi que des informations supplémentaires telles que le prix, les promotions et la publicité.

Site : https://www.kaggle.com/competitions/competitive-data-science-predict-future-sales/overview



Description :

Ce défi sert de projet final pour le cours Coursera "Comment gagner un concours de science des données".

Dans ce concours, vous travaillerez avec un ensemble de données chronologiques complexe composé de données de ventes quotidiennes, aimablement fournies par l'une des plus grandes sociétés de logiciels russes - 1C Company.

Nous vous demandons de prédire les ventes totales de chaque produit et magasin au cours du mois prochain. En résolvant ce concours, vous pourrez appliquer et améliorer vos compétences en science des données.

Description de l'ensemble de données :

Vous recevez des données historiques quotidiennes sur les ventes. La tâche consiste à prévoir la quantité totale de produits vendus dans chaque magasin pour l'ensemble de test. Notez que la liste des boutiques et des produits change légèrement chaque mois. Créer un modèle robuste capable de gérer de telles situations fait partie du défi.

- Descriptions de fichiers
  - sales_train.csv - l'ensemble de formation. Données historiques quotidiennes de janvier 2013 à octobre 2015.
  - test.csv - le jeu de test. Vous devez prévoir les ventes de ces magasins et produits pour novembre 2015.
  - sample_submission.csv - un exemple de fichier de soumission au format correct.
  - items.csv - informations supplémentaires sur les articles/produits.

- item_categories.csv - informations supplémentaires sur les catégories d'articles.
- shops.csv - informations supplémentaires sur les magasins.

➢ Champs de données

- ID - un identifiant qui représente un tuple (boutique, article) dans le jeu de test
- shop_id - identifiant unique d'une boutique
- item_id - identifiant unique d'un produit
- item_category_id - identifiant unique de la catégorie d'article
- item_cnt_day - nombre de produits vendus. Vous prédisez un montant mensuel de cette mesure
- item_price - prix actuel d'un article
- date - date au format jj/mm/aaaa
- date_block_num - un numéro de mois consécutif, utilisé par commodité. janvier 2013 est 0, février 2013 est 1,..., octobre 2015 est 33
- item_name - nom de l'élément
- shop_name - nom de la boutique
- item_category_name - nom de la catégorie d'articles

Cet ensemble de données peut être utilisé à toutes fins, y compris à des fins commerciales.

2- Model :

Importer certaines bibliothèques qui seront nécessaires pour un projet d'apprentissage automatique. Plus précisément, le code importe les bibliothèques suivantes :

- os : cette bibliothèque fournit des fonctions permettant d'interagir avec le système d'exploitation, telles que la lecture et l'écriture de fichiers.
- pandas : cette bibliothèque fournit des structures de données et des outils d'analyse de données pour travailler avec des données structurées.
- numpy : cette bibliothèque prend en charge les grands tableaux et matrices multidimensionnels de données numériques, ainsi que les fonctions mathématiques permettant de les utiliser.
- datetime : ce module fournit des classes pour manipuler les dates et les heures.
- Sequential : il s'agit d'une classe de modèle de la bibliothèque keras, qui est utilisée pour définir une pile linéaire de couches dans un réseau de neurones.
- LSTM : il s'agit d'un type de couche de la bibliothèque keras, qui signifie "Long Short-Term Memory". Les couches LSTM sont couramment utilisées dans les réseaux de neurones récurrents pour le traitement des séquences.
- Dense : il s'agit d'un type de couche de la bibliothèque keras, qui représente une couche entièrement connectée dans un réseau de neurones.
- Dropout : il s'agit d'un type de couche de la bibliothèque keras, qui est utilisé pour appliquer la régularisation de l'abandon à un réseau de neurones.

```python
#load modules
import os
import pandas as pd
import numpy as np
from pandas import read_csv
from pandas import datetime
from keras.models import Sequential
from keras.layers import LSTM,Dense,Dropout
```

Python

Charger des données de plusieurs fichiers CSV dans des dataframes Pandas. La fonction os.listdir est utilisée pour répertorier le contenu du répertoire d'entrée, qui est supposé être situé un niveau au-dessus du répertoire courant (indiqué par '../input').
La fonction pd.read_csv est ensuite utilisée pour lire les fichiers CSV suivants dans les dataframes Pandas :

- item_categories.csv : ce fichier contient des informations sur les différentes catégories d'articles en vente.
- items.csv : ce fichier contient des informations sur les articles individuels qui sont vendus, tels que leur nom et leur catégorie.
- shops.csv : Ce fichier contient des informations sur les différents magasins où les articles sont vendus.
- sample_submission.csv : ce fichier contient un exemple de soumission dans le format correct pour le concours.

```python
#loading data
os.listdir('../input')
item_cat = pd.read_csv('/content/item_categories.csv')
items = pd.read_csv('/content/items.csv')
shops = pd.read_csv('/content/shops.csv')
sample_submission = pd.read_csv('/content/sample_submission.csv')
```

Python

Lire le fichier sales_train.csv dans une trame de données Pandas, puis imprimer la trame de données sur la console pour l'affichage.

```python
df_train = pd.read_csv('/content/sales_train.csv')
df_train
```

Python

	date	date_block_num	shop_id	item_id	item_price	item_cnt_day
0	02.01.2013	0	59	22154	999.00	1.0
1	03.01.2013	0	25	2552	899.00	1.0
2	05.01.2013	0	25	2552	899.00	-1.0
3	06.01.2013	0	25	2554	1709.05	1.0
4	15.01.2013	0	25	2555	1099.00	1.0
...	...	...	...	...	...	...
2935844	10.10.2015	33	25	7409	299.00	1.0
2935845	09.10.2015	33	25	7460	299.00	1.0
2935846	14.10.2015	33	25	7459	349.00	1.0
2935847	22.10.2015	33	25	7440	299.00	1.0
2935848	03.10.2015	33	25	7460	299.00	1.0

2935849 rows × 6 columns

Data description

```python
df_train.info()
```

[10] Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2935849 entries, 0 to 2935848
Data columns (total 6 columns):
 #   Column          Dtype
---  ------          -----
 0   date            object
 1   date_block_num  int64
 2   shop_id         int64
 3   item_id         int64
 4   item_price      float64
 5   item_cnt_day    float64
dtypes: float64(2), int64(3), object(1)
memory usage: 134.4+ MB
```

Convertir la date au format datetime

```python
#convert date to datetime format
df_train['date'] = pd.to_datetime(df_train['date'],format = '%d.%m.%Y')
```

[14] Python

Créer un tableau croisé dynamique

```python
#create pivot table
dataset = df_train.pivot_table(index = ['shop_id','item_id'],values = ['item_cnt_day'],columns = ['date_block_num'])
dataset.reset_index(inplace = True)
dataset
```

[15] Python

	shop_id	item_id	item_cnt_day								...									
date_block_num			0	1	2	3	4	5	6	7	...	24	25	26	27	28	29	30	31	32	33
0	0	30	0	31	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	31	0	11	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	32	6	10	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	33	3	3	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	35	1	14	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
424119	59	22154	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
424120	59	22155	0	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0
424121	59	22162	0	0	0	0	0	0	0	0	...	0	9	4	1	1	0	0	1	0	0
424122	59	22164	0	0	0	0	0	0	0	0	...	0	2	1	2	0	0	1	0	0	0
424123	59	22167	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

424124 rows × 36 columns

Vérifiez les valeurs nulles et remplissez toutes les valeurs NaN avec 0

```python
#check for any null values
dataset.isnull().sum().sum()
```

[18] Python

```
3495064
```

```python
#fill all NaN values with 0
dataset.fillna(0,inplace = True)
dataset.isnull().sum().sum()
```

[19] Python

```
0
```

Diviser l'ensemble de données en données de train et en données de test :

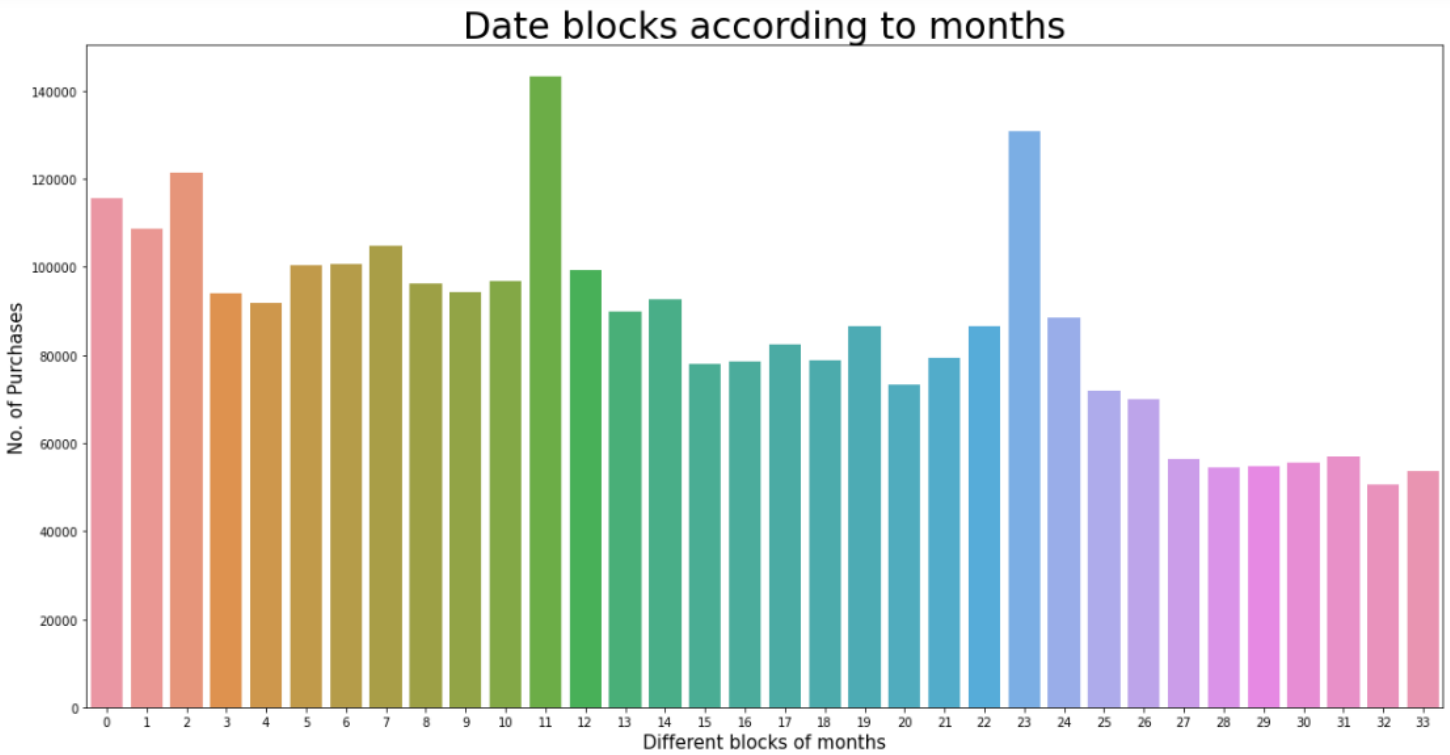
```python
#split the dataset in two
#keep all columns execpt the last one
X_train = np.expand_dims(dataset.values[:,:-1],axis = 2)
# the last column is our label
y_train = dataset.values[:,-1:]
# for test we keep all the columns execpt the first one
X_test = np.expand_dims(dataset.values[:,1:],axis = 2)
# lets have a look on the shape
print(X_train.shape,y_train.shape,X_test.shape)
```

Python

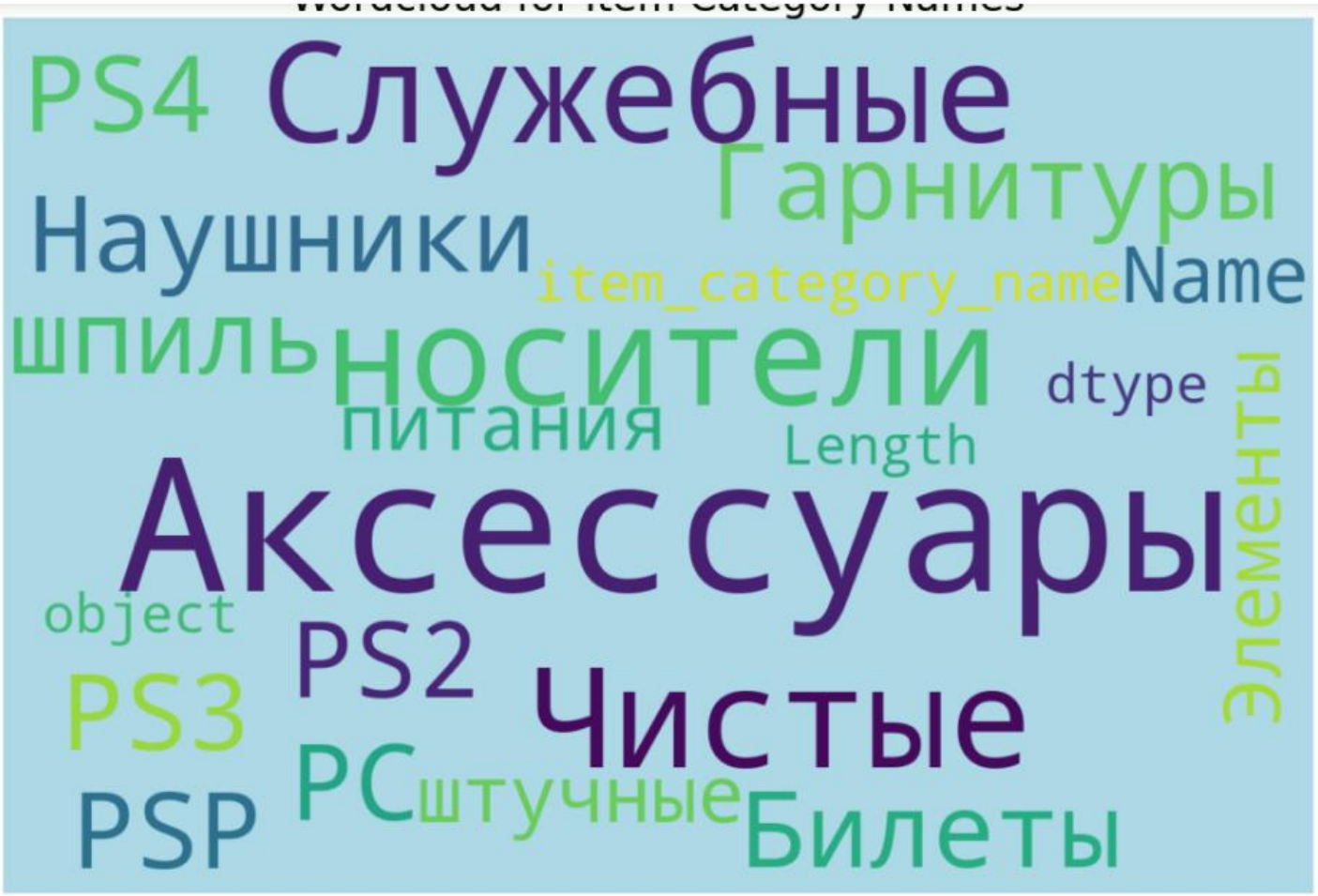
```
(214200, 33, 1) (214200, 1) (214200, 33, 1)
```

Analyse des données :

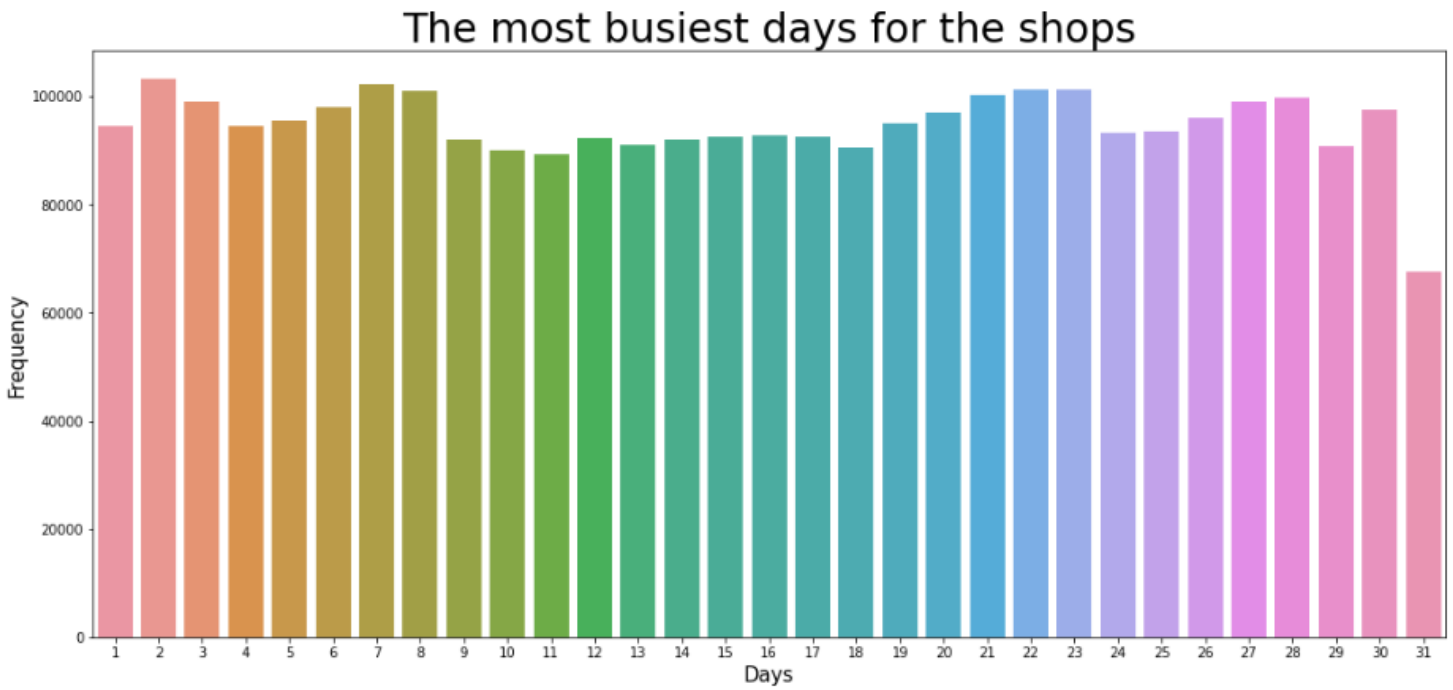
Pour bien comprendre les données on a utilisé EDA (Explorary Data analysis) et voilà quelque résultat que on a obtenue

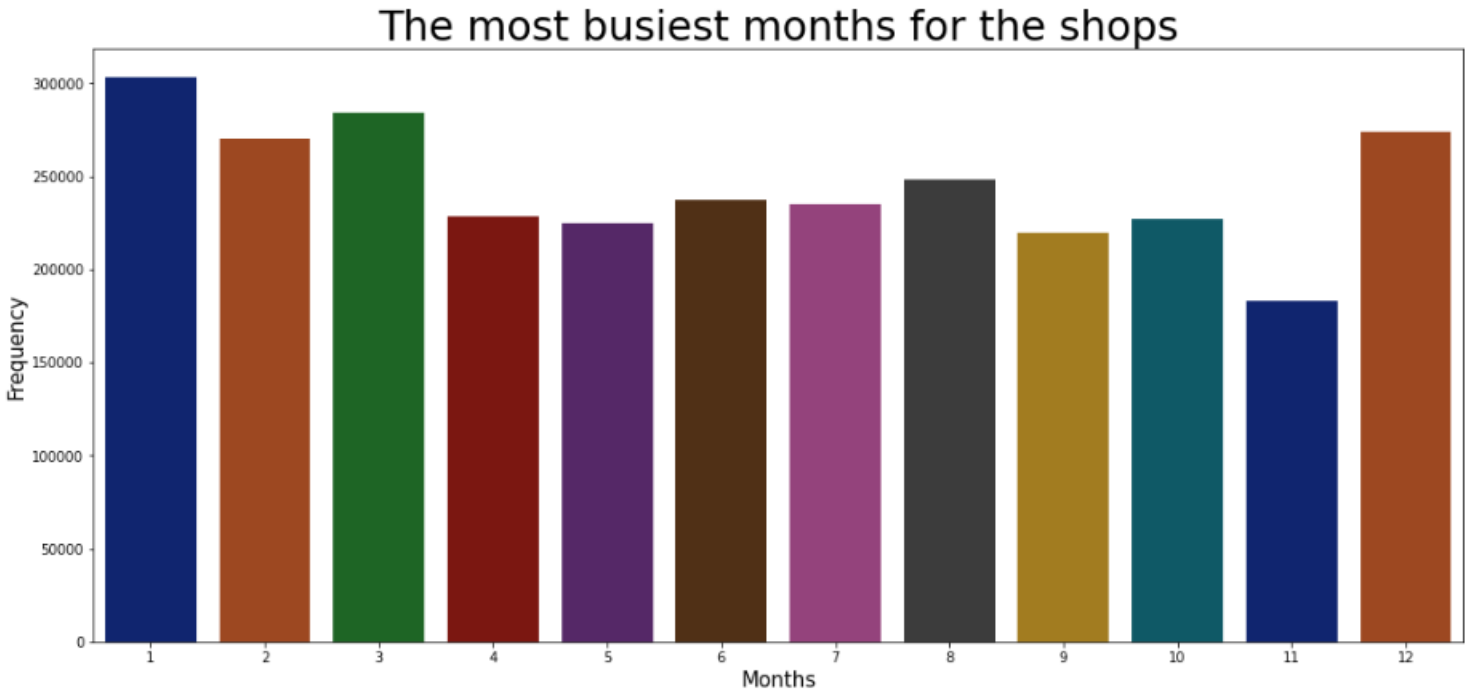


Numéros des purchases par rapport au mois on remarque que les mois 11 et 23 on les plus de vente ce qui est équivalent au mois décembre ce qui est naturel due au solde et black Friday qu'il existe toujours à la fin de l'année



Wordcloud des categories existant





Le mois 1 et le début de mois on la plus de fréquence

Créer un modèle séquentiel

```python
# create sequential model
my_model = Sequential()
my_model.add(LSTM(units = 64,input_shape = (33,1)))
my_model.add(Dropout(0.4))
my_model.add(Dense(1))
my_model.compile(loss = 'mse',optimizer = 'adam', metrics = ['mean_squared_error'])
my_model.summary()
```

Python

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 64)                16896

 dropout (Dropout)           (None, 64)                0

 dense (Dense)               (None, 1)                 65

=================================================================
Total params: 16,961
Trainable params: 16,961
Non-trainable params: 0
_________________________________________________________________
```

Entraîner le modèle

```python
#fit the model
my_model.fit(X_train,y_train,batch_size = 4096,epochs = 10)
```

Python

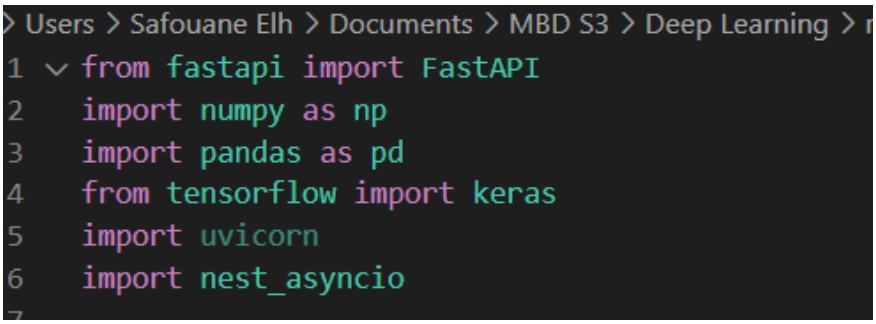
```
Epoch 1/10
53/53 [==============================] - 7s 16ms/step - loss: 30.6590 - mean_squared_error: 30.6590
Epoch 2/10
53/53 [==============================] - 1s 15ms/step - loss: 30.2758 - mean_squared_error: 30.2758
Epoch 3/10
53/53 [==============================] - 1s 15ms/step - loss: 30.0886 - mean_squared_error: 30.0886
Epoch 4/10
53/53 [==============================] - 1s 15ms/step - loss: 29.9602 - mean_squared_error: 29.9602
Epoch 5/10
53/53 [==============================] - 1s 15ms/step - loss: 29.8511 - mean_squared_error: 29.8511
Epoch 6/10
53/53 [==============================] - 1s 14ms/step - loss: 29.7825 - mean_squared_error: 29.7825
Epoch 7/10
53/53 [==============================] - 1s 15ms/step - loss: 29.7755 - mean_squared_error: 29.7755
Epoch 8/10
53/53 [==============================] - 1s 14ms/step - loss: 29.7670 - mean_squared_error: 29.7670
Epoch 9/10
53/53 [==============================] - 1s 14ms/step - loss: 29.7278 - mean_squared_error: 29.7278
Epoch 10/10
53/53 [==============================] - 1s 14ms/step - loss: 29.7156 - mean_squared_error: 29.7156

<keras.callbacks.History at 0x7f8400346d90>
```

3- FastApi

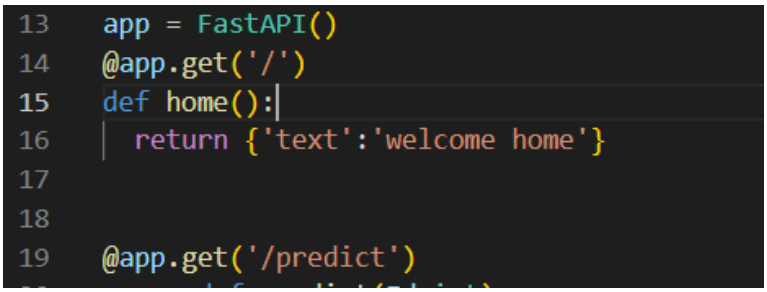
FastAPI est un framework Web moderne permettant de créer des API avec Python. Il n'est pas directement lié à l'apprentissage automatique ou à MLOps, mais il peut être utilisé comme backend pour un pipeline MLOps afin de servir des modèles d'apprentissage automatique en tant qu'API. FastAPI fournit des fonctionnalités telles que la génération automatique de documentation et la prise en charge de la communication en temps réel, ce qui peut faciliter le déploiement et la mise à l'échelle des modèles d'apprentissage automatique.

Création d'une API pour faire des prédictions à l'aide d'un modèle d'apprentissage automatique. L'API a deux points de terminaison :

- Le code importe d'abord plusieurs bibliothèques qui seront utilisées dans l'API, notamment numpy, pandas, keras (de TensorFlow) et uvicorn (pour exécuter l'API). Il importe également la bibliothèque nest_asyncio, qui est utilisée pour permettre aux bibliothèques basées sur asyncio (telles que FastAPI) de fonctionner correctement dans des environnements imbriqués.



- Le code charge ensuite une application FastAPI. L'application a deux routes : /, qui renvoie un message de bienvenue, et /predict, qui prend un paramètre Id et renvoie une prédiction du nombre de produits qui seront vendus.



- Pour effectuer la prédiction, le code charge un modèle d'apprentissage automatique formé à partir d'un fichier, l'utilise pour faire des prédictions sur certaines données de test, puis renvoie la valeur prédite pour l'ID demandé. Les valeurs prédites sont également enregistrées dans un fichier de soumission dans un format spécifique.

```python
@app.get('/predict')
async def predict(Id:int):
    loaded_model = keras.models.load_model('app\LSTM_model.h5')
  # creating submission file
    xtest = np.load('app\parrot.npy')
    submission_file = loaded_model.predict(xtest)
  # we will keep every value between 0 and 20
    submission_file = submission_file.clip(0,20)
  # creating dataframe with required columns
    submission_trp = pd.DataFrame({'ID':df_test['ID'],'item_cnt_month':submission_file.ravel()})
    val = submission_trp['item_cnt_month'].values[Id]
    val = float(val)
   # Return val as a dictionary
    return {'number of products sold':val}
```

- Enfin, le code renvoie la valeur prédite sous forme de dictionnaire, avec la clé 'nombre de produits vendus'.

main.py

C: > Users > Safouane Elh > Documents > MBD S3 > Deep Learning > mlops3 > app > main.py > ...

```python
from fastapi import FastAPI
import numpy as np
import pandas as pd
from tensorflow import keras
import uvicorn
import nest_asyncio

nest_asyncio.apply()


df_test = pd.read_csv('C:/Users/Safouane Elh/Documents/MBD S3/Deep Learning/MLOPS/test.csv')

app = FastAPI()
@app.get('/')
def home():
  return {'text':'welcome home'}


@app.get('/predict')
async def predict(Id:int):
    loaded_model = keras.models.load_model('app\LSTM_model.h5')
  # creating submission file
    xtest = np.load('app\parrot.npy')
    submission_file = loaded_model.predict(xtest)
  # we will keep every value between 0 and 20
    submission_file = submission_file.clip(0,20)
  # creating dataframe with required columns
    submission_trp = pd.DataFrame({'ID':df_test['ID'],'item_cnt_month':submission_file.ravel()})
    val = submission_trp['item_cnt_month'].values[Id]
    val = float(val)
   # Return val as a dictionary
    return {'number of products sold':val}
```

```python
@app.get("/shops")
def all_shops():
    # Access the "shops" collection
    shops_collection = db["Shops"]

    shops = list(shops_collection.find())


    return [Shop(**shop) for shop in shops]

@app.get("/items")
def all_items():
    # Access the "shops" collection
    items_collection = db["Items"]

    items = list(items_collection.find())


    return [Item(**item) for item in items]
```

Deux endpoints pour retourner les shops et items pour predicter

4- MongoDb & React js

4-1 / MongoDb

La partie mongoDb pour passer les donnees de format .csv vers mongodb pour les assumes dans notre application spa

Exemple pour enregistrer tous les donnees des shops :

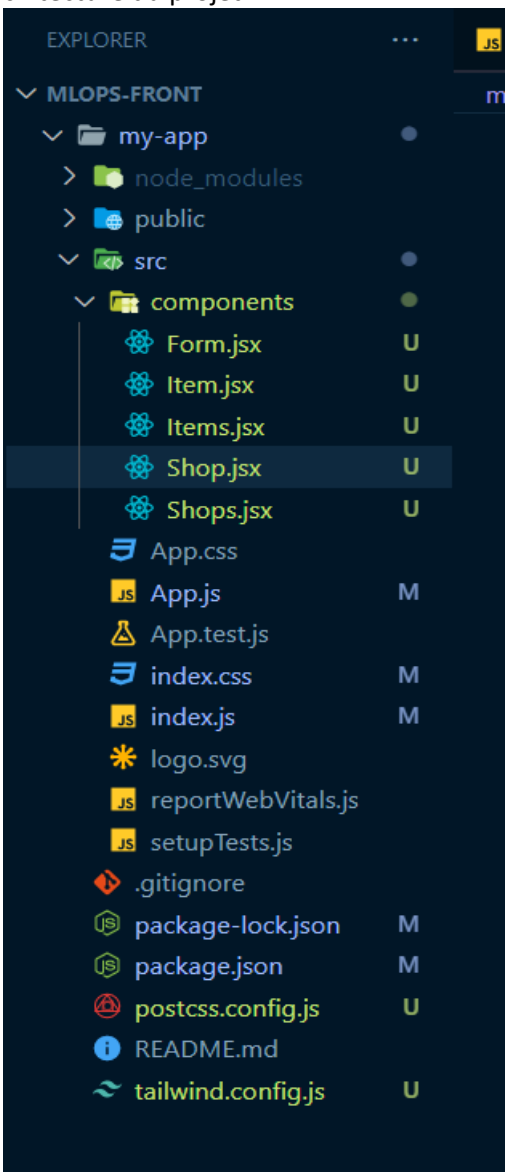
```python
# Making a Connection with MongoClient
client = MongoClient("mongodb://localhost:27017/")
# database
db = client["MLOPS"]
# collection
shops= db["Shops"]
```

```python
df_shops = pd.read_csv('shops.csv')


data_shops = df_shops.to_dict("records")


for data in data_shops:
    shops.insert_one({"id":data['shop_id'],"shop_name":data['shop_name']})
```

1. Connection vers mongoDb et créations d'une database et une collection dans celle-ci
2. Lire les données du fichier csv
3. Itérer sur les données et enregistrer dans notre collection

4-2 / React

Architecture du projet :



App.js qui render notre component Form qui contient notre formulaire

```
my-app > src > App.js > App
1  import "./App.css";
2  import Form from "./components/Form";
3
4  function App() {
5    return (
6      <>
7        <Form></Form>
8      </>
9    );
10 }
11
12 export default App;
13
```

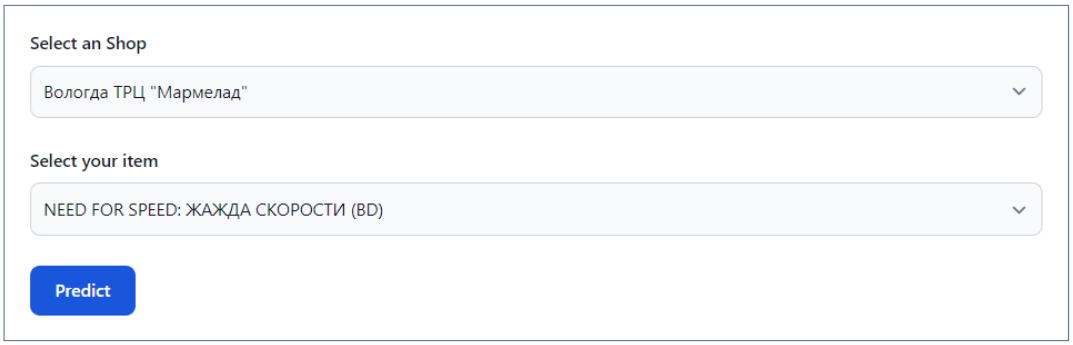
Notre Form component qui contient notre formulaire

```
return (
  <>
    <div className="w-2/4 m-auto mt-5 border-separate border-spacing-2 border border-slate-500 p-5">
      <form onSubmit={handleSubmit}>
        <label class="block mb-2 text-sm font-medium text-gray-900 dark:text-white">
        </label>
        <select
          name="id_shop"
          id="id_shop"
          onChange={handleChange}
          value={formData.id_shop}
          className="bg-gray-50 border border-gray-300 text-gray-900 text-sm rounded-lg focus:ring-blue-500 focus:border-blue-500 block w-full p-2.5 dark:bg-gray-700 dark:border-gray-600 dark:placeholder-
        >
          <option selected>Choose a shop</option>
          <Shops></Shops>
        </select>
        <br />

        <label class="block mb-2 text-sm font-medium text-gray-900 dark:text-white">
          Select your item
        </label>
        <select
          name="id_item"
          id="id_item"
          onChange={handleChange}
          value={formData.id_item}
          className="bg-gray-50 border border-gray-300 text-gray-900 text-sm rounded-lg focus:ring-blue-500 focus:border-blue-500 block w-full p-2.5 dark:bg-gray-700 dark:border-gray-600 dark:placeholder-
        >
          <option selected>Choose an item</option>

          <Items></Items>
        </select>
        <br />

        <button
          type="submit"
          class="text-white bg-blue-700 hover:bg-blue-800 focus:ring-4 focus:outline-none focus:ring-blue-300 font-medium rounded-lg text-sm w-full sm:w-auto px-5 py-2.5 text-center dark:bg-blue-600 dark:hover
        >
          Predict
        </button>
      </form>
    </div>

    {responseData && (
      <p class="w-1/4 m-auto mt-9">
        result of prediction is{" "}
        <span className="text-blue-700">{responseData}</span>
      </p>
    )}
  </>
```

Ce formulaire contient tous les shops et items et on submit permet de faire une requête vers notre backend pour prédire la valeur

```
const handleSubmit = (event) => {
  event.preventDefault();
  // Submit the form data to the server
  const { id_shop, id_item } = event.target;

  console.log(id_shop.value);

  axios
    .get(
      `http://127.0.0.1:5000/predict?id_shop=${id_shop.value}&id_item=${id_item.value}`
    )
    .then((response) => {
      const { Result } = response.data;
      setResponseData(Result);
    })
    .catch((error) => {
      console.error(error);
    });
};
```

Resultat :



result of prediction is 0.22228911519050598

5- Docker

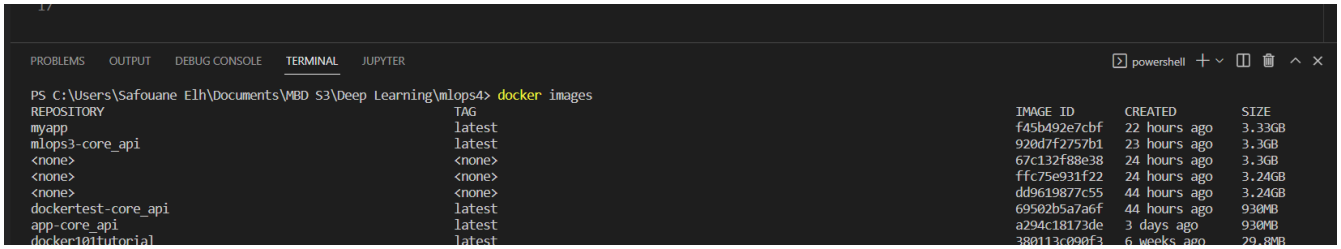
Docker est un outil couramment utilisé dans les pipelines MLOps pour empaqueter et déployer des modèles d'apprentissage automatique. Il permet de conteneuriser les modèles et leurs dépendances, de créer des versions reproductibles, de simplifier le processus de déploiement et d'isoler les modèles du système hôte. En utilisant Docker dans un pipeline MLOps, vous pouvez rationaliser le processus de création, de test et de déploiement de modèles d'apprentissage automatique.

Dockerfile

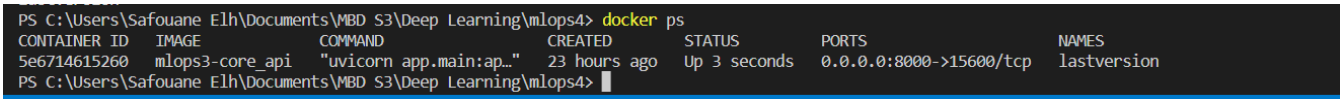
C: > Users > Safouane Elh > Documents > MBD S3 > Deep Learning > mlops4 > Dockerfile > ...

```
1   FROM python3.7
2
3   COPY . /app
4
5   WORKDIR /app
6
7   RUN pip install fastapi
8   RUN pip install uvicorn
9   RUN pip install numpy
10  RUN pip install pandas
11  RUN pip install tensorflow
12  RUN pip install nest-asyncio
13
14  EXPOSE 8000
15
16  CMD ["uvicorn", "main:app", "--reload", "--host", "0.0.0.0", "--port", "8000"]
17
```

- ✓ Définit une image Docker pour une application qui utilise le framework Web FastAPI pour créer une API. L'image est construite en copiant le contenu du répertoire actuel dans l'image, en installant plusieurs bibliothèques Python, en exposant le port 8000 lors de l'exécution et en définissant la commande par défaut pour démarrer le serveur Web Uvicorn avec les arguments spécifiés.



Déploiement réussi :



6- Kubernetes

Kubernetes est un système open source permettant d'automatiser le déploiement, la mise à l'échelle et la gestion des applications conteneurisées, y compris les modèles d'apprentissage automatique. Dans un pipeline MLOps, Kubernetes peut être utilisé pour automatiser le déploiement de modèles d'apprentissage automatique, augmenter ou réduire le nombre de répliques en fonction de la demande, équilibrer la charge du trafic vers plusieurs répliques, déployer des mises à jour de modèles de manière contrôlée et surveiller le santé des modèles. En utilisant Kubernetes dans un pipeline MLOps, vous pouvez automatiser le déploiement et la gestion des modèles d'apprentissage automatique, ce qui facilite la mise à l'échelle et l'amélioration de vos modèles au fil du temps.

Dockerfile ● ! kubernetes.yml ×

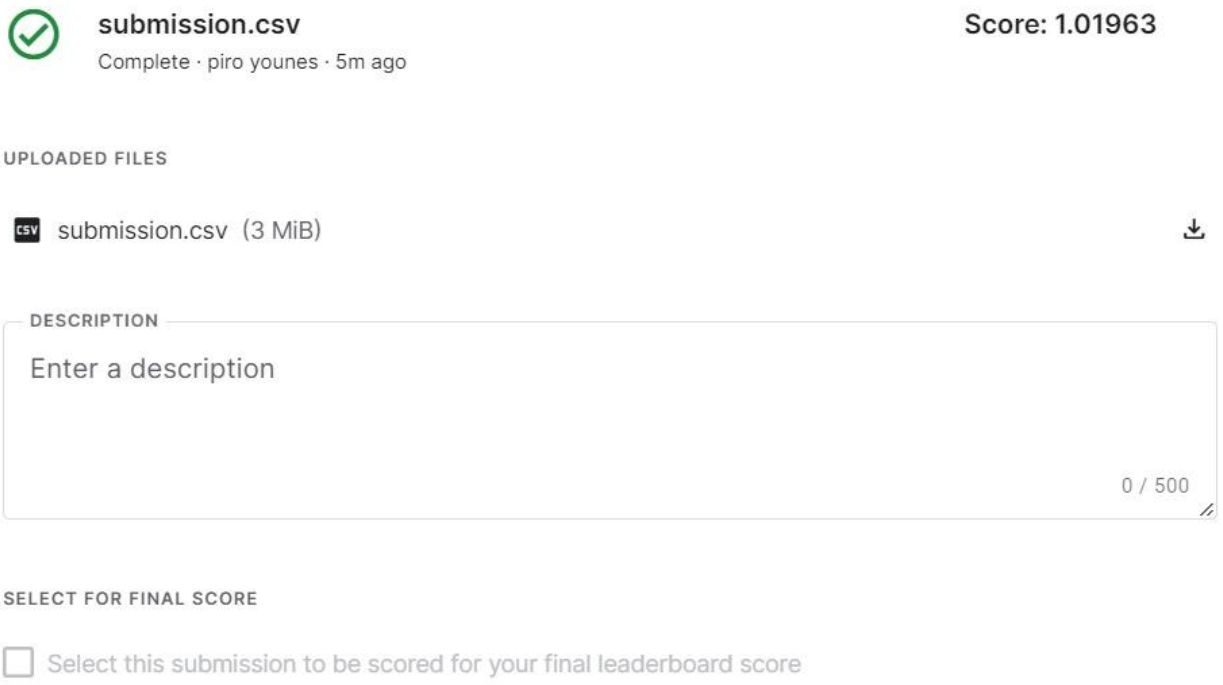
C: > Users > Safouane Elh > Documents > MBD S3 > Deep Learning > mlops4 > ! kubernetes.yml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
      - name: myapp
        image: myapp:latest
        ports:
        - containerPort: 8000

```

- Il s'agit d'un manifeste de déploiement Kubernetes écrit en YAML. Il spécifie un déploiement appelé "myapp-deployment" qui gère 3 répliques d'un pod avec un seul conteneur exécutant une image docker appelée "myapp:latest". Le conteneur écoute sur le port 8000 et est étiqueté avec "app : myapp". Le déploiement créera des pods avec cette étiquette, et le champ de sélection dans la spécification du déploiement garantira que le déploiement gère ces pods.

```
PS C:\Users\Safouane Elh\Documents\MBD S3\Deep Learning\mlops4> kubectl apply -f kubernetes.yml
deployment.apps/myapp-deployment configured
PS C:\Users\Safouane Elh\Documents\MBD S3\Deep Learning\mlops4>
```

7- Le score du la compétition



8- Conclusion

La conduite de ce projet fut une expérience à la fois inédite et enrichissante pour nous. Autant sur le plan théorique que pratique. Nous en sortons ragaillardis, riche de nouvelles compétences techniques à savoir déploiement d'un modèle de deep learning utilisant fast api, docker et kubernetes ….

9- Lien GitHub :

https://github.com/safouane-elharrak/Projet-MLOps