



RAPPORT PROJET DE DATA ENGINEERING

Prédiction du prix du pétrole

Réalisé par :

AABBAR Adnane
ER-RACHIDY Najwa
KLIMINA Mariia

Encadré par :

Pr. Amine Ferdjaoui

Année : 2022/2023

Table des matières

Introduction	3
Réalisation	4
1 Étapes du projet	4
2 Partage des tâches	8
3 Repository	9
3.1 Github	9
3.2 Docker Hub	9

Table des figures

Figure 1	Résultats pour 10 time step	6
Figure 2	Résultats pour 20 time step	6
Figure 3	Résultats pour 30 time step	6

Introduction

L'objectif de ce projet est de travailler sur un projet concernant la prédiction de séries temporelles, principalement la prédiction des prix du pétrole. Nous utiliserons Git et Github pour le contrôle de version, et nous collaborerons les uns avec les autres dans ce dépôt. Ainsi nous utiliserons de nombreux conteneurs responsables de différentes tâches et pipelines.

Réalisation

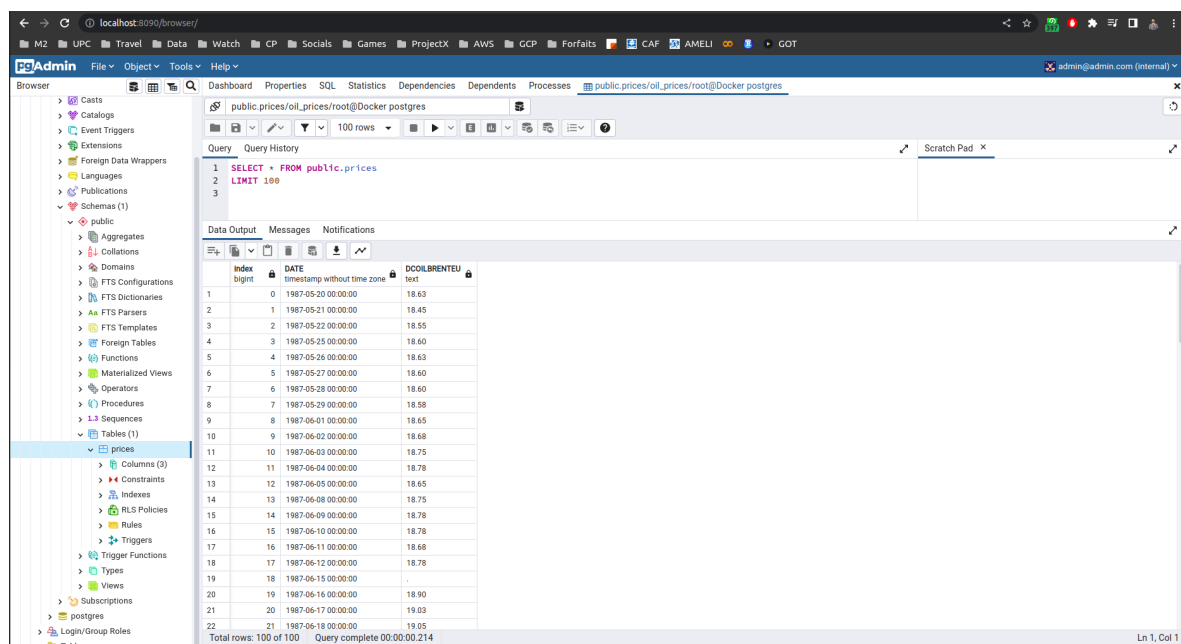
1 Étapes du projet

- Récupération dynamique des données à l'aide de l'URL donné en entrée au container de Docker

- Ingestion de données dans Postgres

Pour cette partie, l'idée est d'éviter de copier manuellement des ensembles de données de l'intérieur du conteneur vers l'hôte afin d'effectuer des tâches d'apprentissage automatique, nous avons donc décidé de construire un petit flux ETL qui ingère des données dans une base de données Postgres qui peut être administrée à l'aide du logiciel pgAdmin.

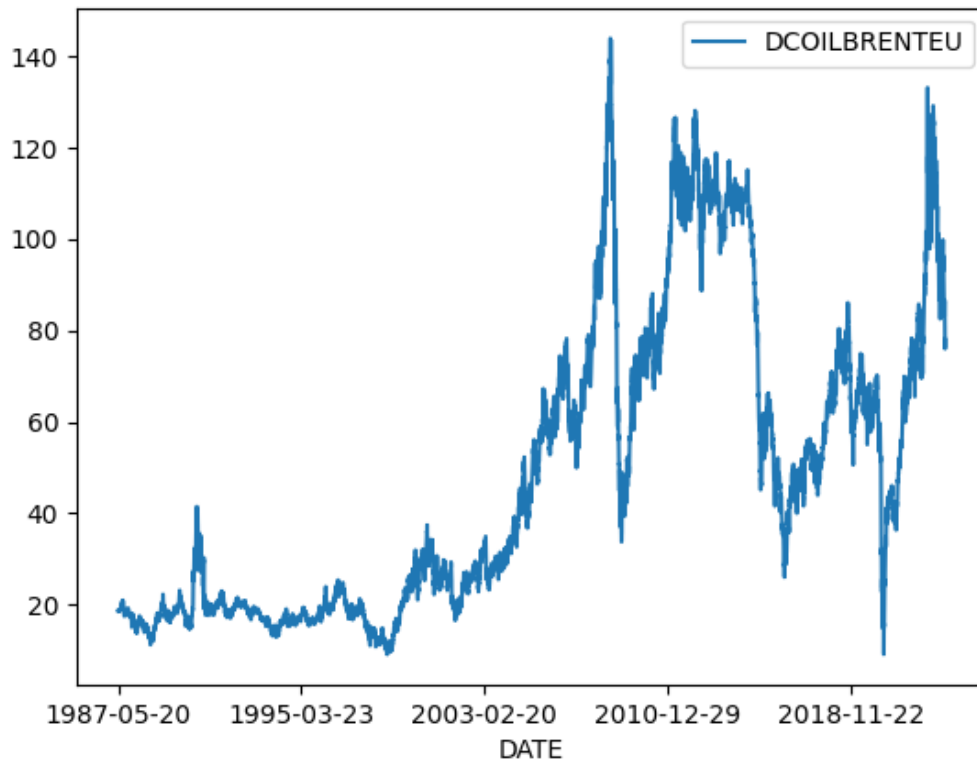
Pour créer et exécuter l'instance de la base de données Postgres et pgAdmin, nous avons essayé de construire des images séparées et d'exécuter les conteneurs séparément dans le même réseau, cela a fonctionné mais c'était aussi une chance pour nous d'explorer docker-compose qui facilite l'exécution de plusieurs conteneurs, nous avons également utilisé des volumes pour conserver la configuration de la dernière exécution du conteneur afin de ne pas perdre les données après l'ingestion à chaque fois. (Consultez le fichier docker-compose.yaml pour en savoir plus).



index	DATE	DOUILBREITEU
0	1987-05-20 00:00:00	18.63
1	1987-05-21 00:00:00	18.45
2	1987-05-22 00:00:00	18.55
3	1987-05-23 00:00:00	18.60
4	1987-05-24 00:00:00	18.63
5	1987-05-25 00:00:00	18.60
6	1987-05-26 00:00:00	18.60
7	1987-05-27 00:00:00	18.60
8	1987-05-28 00:00:00	18.58
9	1987-06-01 00:00:00	18.65
10	1987-06-02 00:00:00	18.68
11	1987-06-03 00:00:00	18.75
12	1987-06-04 00:00:00	18.78
13	1987-06-05 00:00:00	18.65
14	1987-06-08 00:00:00	18.75
15	1987-06-09 00:00:00	18.78
16	1987-06-10 00:00:00	18.78
17	1987-06-11 00:00:00	18.68
18	1987-06-12 00:00:00	18.78
19	1987-06-15 00:00:00	.
20	1987-06-16 00:00:00	18.90
21	1987-06-17 00:00:00	19.03
22	1987-06-18 00:00:00	19.05

- Entraînement d'un modèle de série temporelle

1. Lire les données et les tracer,



2. Diviser les données en ensembles d'entraînement et de test,
3. Feature scaling et reshaping des données d'entraînement et de test.
4. Entraîner un modèle LSTM localement,
5. Effectuer les prédictions sur les données de test (Avec différents time step pour améliorer le modèle) et générer les graphiques et le modèle.

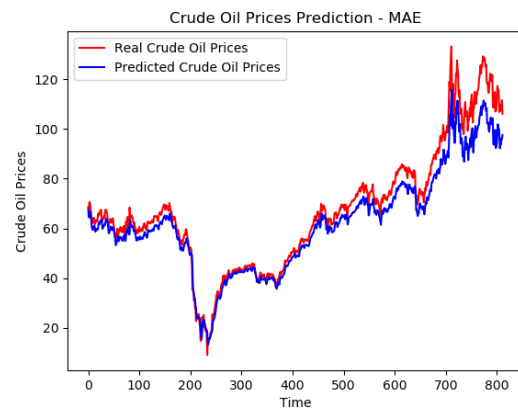
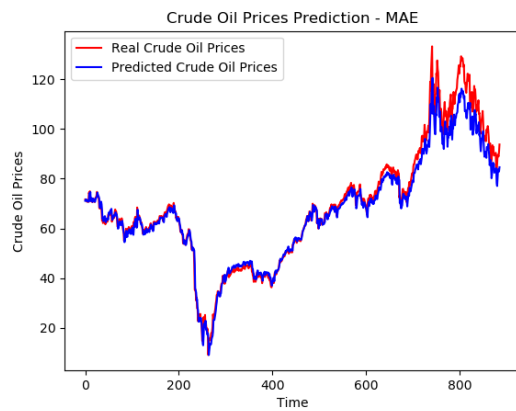


FIGURE 1 – Résultats pour 10 time step FIGURE 2 – Résultats pour 20 time step

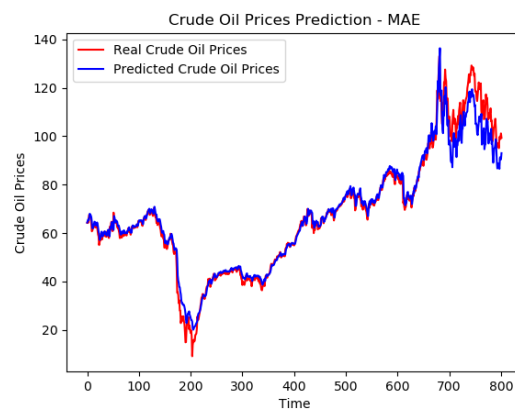


FIGURE 3 – Résultats pour 30 time step

- Dockerisation du modèle
- Déploiement

Modèles disponibles

Choisissez un modèle:

Choisissez une date à prédire:

```
python app.py
(base) adnane@hp-adnane: flask_app git:(flask_app) X python app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
* Use a production WSGI server instead.
* Debug mode: on
* Running on all addresses.
WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://10.25.3.115:5000/ (Press CTRL+C to quit)
* Restarting with watchdog (inotify)
* Debugger is active!
* Debugger PIN: 104-102-372
10.25.3.115 - - [20/Jan/2023 23:21:29] "GET / HTTP/1.1" 200 -
[20/Jan/2023 23:21:29] "POST /predict HTTP/1.1" 200 -
```

```
{
  "Day": "2023-01-25",
  "Predicted Price": [
    83.05342177114441
  ]
}
```

```
python app.py
(base) adnane@hp-adnane: flask_app git:(flask_app) X python app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
* Use a production WSGI server instead.
* Debug mode: on
* Running on all addresses.
WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://10.25.3.115:5000/ (Press CTRL+C to quit)
* Restarting with watchdog (inotify)
* Debugger is active!
* Debugger PIN: 104-102-372
10.25.3.115 - - [20/Jan/2023 23:21:29] "GET / HTTP/1.1" 200 -
10.25.3.115 - - [20/Jan/2023 23:23:12] "POST /predict HTTP/1.1" 200 -
```

- Dockerisation du backend


```
{
  "Day": "2023-01-25",
  "Predicted Price": {
    83.85342177114441
  }
}
```

```
docker run -it --name=flask-app --mount source=share,destination=/app/share 101x51
(base) adnane@hp-adnane: flask_app git:(flask_app) # docker run -it --name=flask-app --mount source=share,destination=/app/share flask-app
* Serving Flask app 'app' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.17.0.2:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 190-983-210
172.17.0.1 - - [21/Jan/2023 00:27:27] "GET / HTTP/1.1" 200 -
Importing plotly failed. Interactive plots will not work.
172.17.0.1 - - [21/Jan/2023 00:27:55] "POST /predict HTTP/1.1" 200 -
```

Modèles disponibles

Choisissez un modèle: figraphnet

Choisissez une date à prédire:

```
docker run -it --name=flask-app --mount source=share,destination=/app/share 101x51
(base) adnane@hp-adnane: flask_app git:(flask_app) # docker run -it --name=flask-app --mount source=share,destination=/app/share flask-app
* Serving Flask app 'app' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.17.0.2:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 190-983-210
172.17.0.1 - - [21/Jan/2023 00:27:27] "GET / HTTP/1.1" 200 -
```

2 Partage des taches

- Aabbar Adnane :
 - Dockerisation du process de téléchargement de données dynamiquement à l'aide d'un URL sur la branche '*data_fetching*'.
 - Dockerisation du process d'ingestion de données dans une BDD **Postgres**, et son administration avec **PgAdmin** à l'aide de **Docker Compose** sur la branche '*data_ingesting*'.
 - Dockerisation et déploiement du modèle sur une branche '*flask_app*' à l'aide de flask comme backend.
- Er-rachidy Najwa :

- Entraîner un modèle **LSTM** et le tester sur 10 step sur une branche *'model_training'*.
- Récupérer un plus grand dataset de pétrole, entraîner un modèle **Fbprophet** sur une branche *'flask_app'* et le sauvegarder pour déploiement.
- Klimina Mariia :
 - Tester le modèle sur **20** et **30 timesteps** sur une branche *'model_training_timesteps'*.
 - Dockerisation sur la branche *'model_training_timesteps'* de l'entraînement sur différentes timesteps et récupération des modèles et images depuis le container.

3 Repository

3.1 Github

<https://github.com/adnaneaabbar/oil-price-prediction>

3.2 Docker Hub

<https://hub.docker.com/r/adnaneaabbar/flask-app>