Facebook Prophet appliqué au prix du pétrole

Youssef Saile
Maître de stage : Hajar Fariane

L’objectif de l’application était dans un premier temps de récupérer les données avec un scraper puis de réaliser une prédiction des prix. Pour arriver à notre fin, nous avons utilisé Facebook Prophet. Prophet est un procédé pour prédire sur la base de séries temporelles, il est résistant au manque de données, supporte les exceptions statistiques, et a l’avantage d'être open source.

Ce projet était assez ardu notamment à cause du fait qu'il fallait construire plusieurs outils indépendants pour que l’application fonctionne.

Ainsi, je présenterais mon application dans le rapport suivant.

Scraper

J’ai pu construire un scraper assez court et simple qui enregistre le cours du pétrole Brent. J’ai opté pour cette solution mais il était aussi possible d'émuler une session Firefox ou Chrome pour scraper la page html du site, mais c'était inutilement complexe.

In [ ]:
```
import numpy as np
import pandas as pd
from fbprophet import Prophet
import matplotlib.pyplot as plt



import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/brent-oil-prices/BrentOilPrices.csv
```

In [ ]:
```
filepath='../input/brent-oil-prices/BrentOilPrices.csv'
brent=pd.read_csv(filepath)
```

Ici, donc on importe les différents packages nécessaires et je récupere les données à travers un fichier csv.

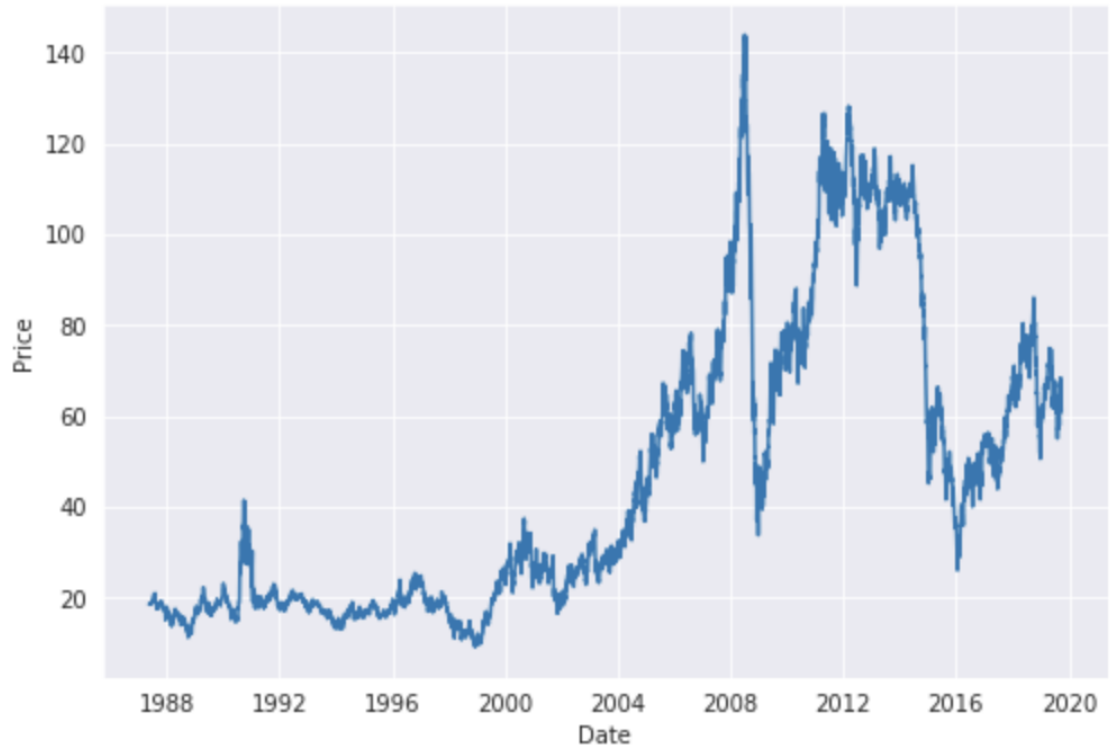
Out[37]:

	Date	Price
0	May 20, 1987	18.63
1	May 21, 1987	18.45
2	May 22, 1987	18.55
3	May 25, 1987	18.60
4	May 26, 1987	18.63
5	May 27, 1987	18.60
6	May 28, 1987	18.60
7	May 29, 1987	18.58
8	Jun 01, 1987	18.65
9	Jun 02, 1987	18.68

Nous avions donc un sample de data de 1987 jusqu'en 2021. Mais pour utiliser ces données il faut les convertir dans un format standard que Prophet peut comprendre

```
df['Date'] = pd.to_datetime(df['Date'], format="%b %d, %Y")
df.head()
```

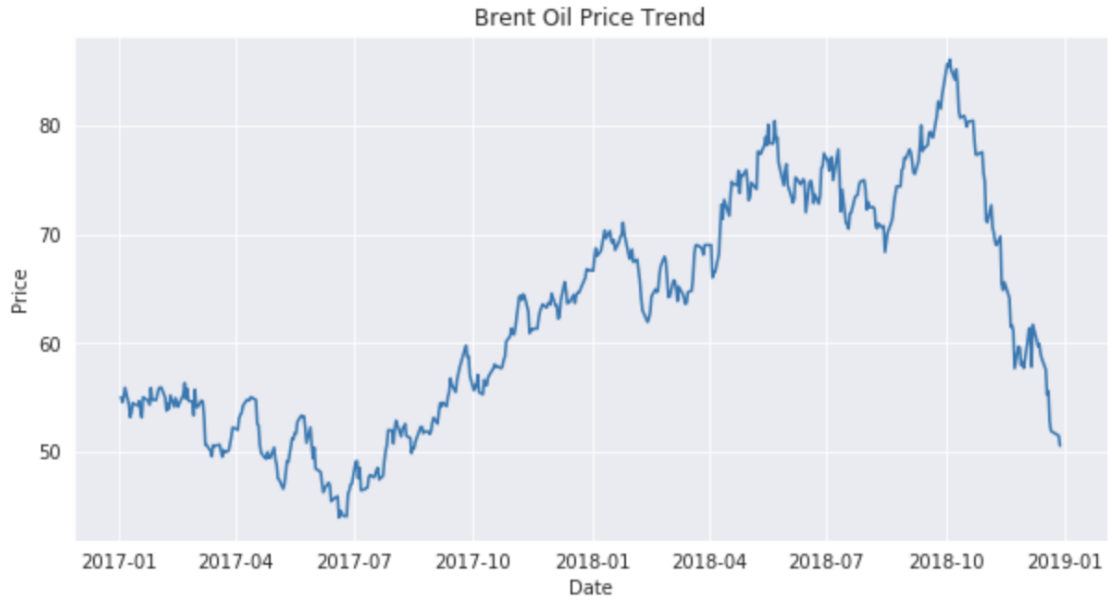
J’ai donc pu faire un graphique de la tendance de prix du pétrole qui allait nous servir plus tard comme fondation, étant donné que c’est sur la base de ce graphique que Prophet construit toute sa prédiction.



J’ai aussi fait un graphique des tendances de prix par période temporelle

```
In [ ]: def plot_price_trend(df, start_date, end_date):

            mask = (df['Date'] > start_date) & (df['Date'] <= end_date)
            sdf = df.loc[mask]
            plt.figure(figsize = (10,5))
            chart = sns.lineplot(x='Date',y='Price',data = sdf)
            #chart.set_xticklabels(chart.get_xticklabels(), rotation=45)
            plt.title("Brent Oil Price Trend")
```



On note donc que le prix était en constante progression jusqu'en octobre 2018 ou il a commencé a brutalement descendre.

PROPHET

On commence d'abord par importer Prophet et on crée une instance.

```
from fbprophet import Prophet
m = Prophet()
```

Sachant que Prophet requiert que la colonne date soit nommée "ds" et que la variable donc ici le prix soit nommée "y", on procède aux changements nécessaires.

In [44]:
```
pro_df = df
pro_df.columns = ['ds','y']
pro_df.head()
```

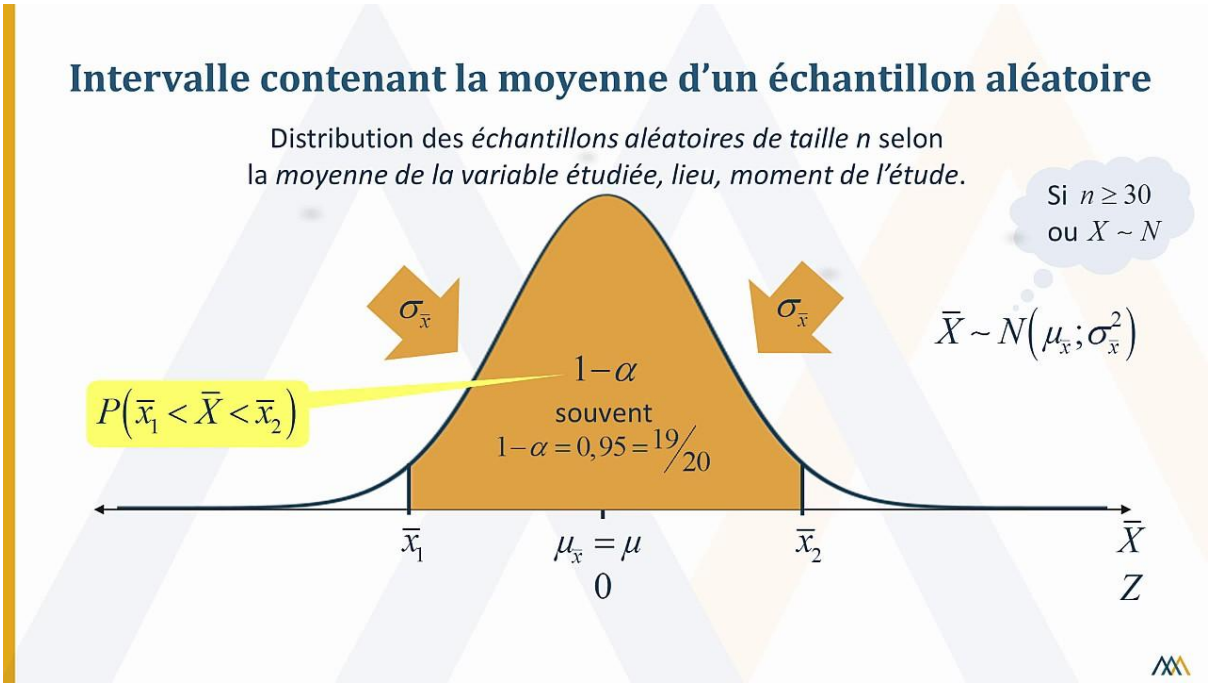
Out[44]:

	ds	y
0	1987-05-20	18.63
1	1987-05-21	18.45
2	1987-05-22	18.55
3	1987-05-25	18.60
4	1987-05-26	18.63

Après, il faut faire correspondre ce dataframe au modèle créé, puis on débute la prédiction dans une période de 90 jours après le dernier prix.

```
m.fit(pro_df)
future = m.make_future_dataframe(periods = 90)
forecast = m.predict(future)
```

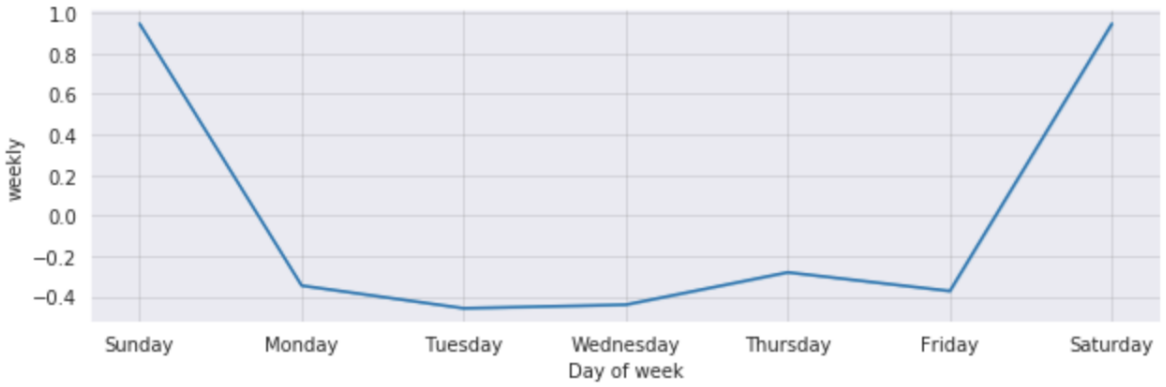
La prédiction comporte plusieurs éléments, les tendances, la saisonnalité journalière, hebdomadaire et annuelle, et pour chacun de ses éléments nous avons les bornes supérieures et inférieures de l'intervalle de confiance

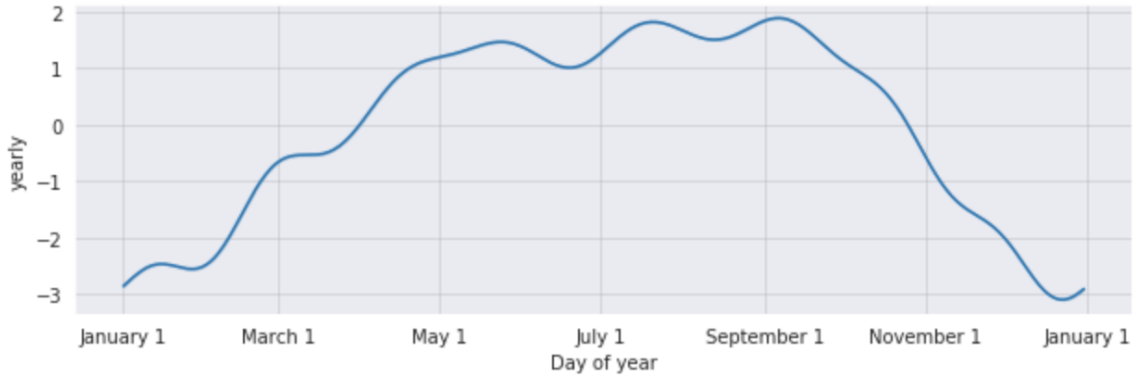


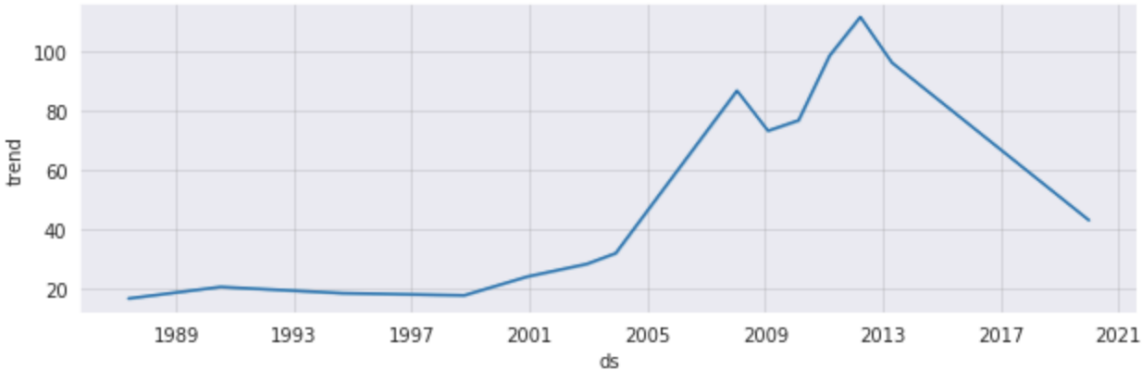
	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_terms	additive_terms_lower	additive_terms_upper	weekly	weekly_low
8935	2021-11-18	36.216149	17.689843	52.261755	27.795212	44.886225	-1.333485	-1.333485	-1.333485	-0.217625	-0.217625
8936	2021-11-19	36.198735	16.665284	52.834296	27.741800	44.884855	-1.447730	-1.447730	-1.447730	-0.304014	-0.304014
8937	2021-11-20	36.181321	18.552477	53.423873	27.692233	44.883484	-0.400028	-0.400028	-0.400028	0.771495	0.771495
8938	2021-11-21	36.163907	19.453588	52.590659	27.642666	44.882114	-0.428232	-0.428232	-0.428232	0.771495	0.771495
8939	2021-11-22	36.146494	16.550788	50.950006	27.593099	44.880743	-1.488850	-1.488850	-1.488850	-0.260096	-0.260096

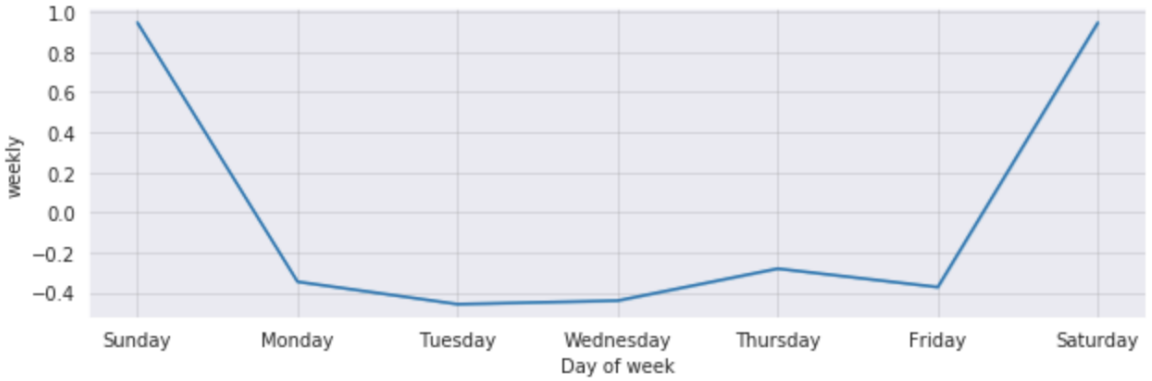
weekly_lower	weekly_upper	yearly	yearly_lower	yearly_upper	multiplicative_terms	multiplicative_terms_lower	multiplicative_terms_upper	yhat
-0.217625	-0.217625	-1.115860	-1.115860	-1.115860	0.0	0.0	0.0	34.882664
-0.304014	-0.304014	-1.143717	-1.143717	-1.143717	0.0	0.0	0.0	34.751005
0.771495	0.771495	-1.171523	-1.171523	-1.171523	0.0	0.0	0.0	35.781293
0.771495	0.771495	-1.199727	-1.199727	-1.199727	0.0	0.0	0.0	35.735675
-0.260096	-0.260096	-1.228754	-1.228754	-1.228754	0.0	0.0	0.0	34.657644

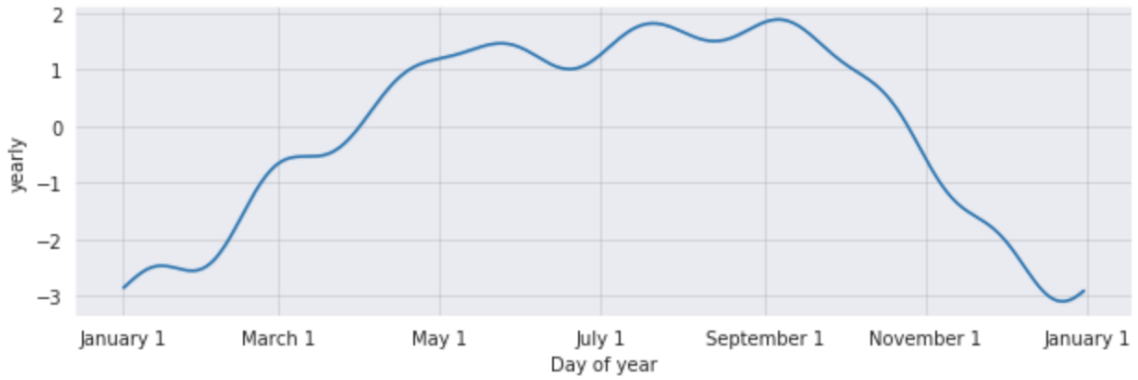
On graphe toutes ces elements du modèle de prédiction


1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
weekly
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Day of week

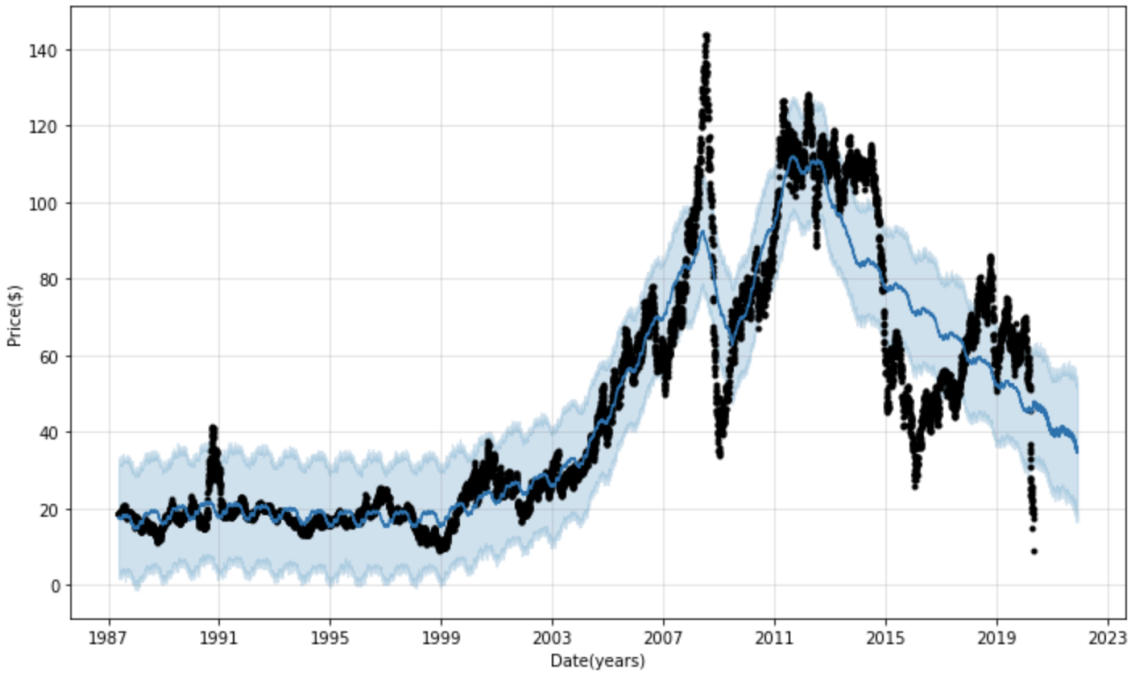

2
1
0
−1
−2
−3
yearly
January 1
March 1
May 1
July 1
September 1
November 1
January 1
Day of year


100
80
60
40
20
trend
1989
1993
1997
2001
2005
2009
2013
2017
2021
ds


1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
weekly
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Day of week



On construit la version finale de la prédiction grâce à tous ces éléments



J’ai pensé que ce serait une bonne idée de visualiser côte-à-côte la prédiction avec les prix réels. Pour cela, on fusionne les données originales avec celles projetées.

```
cmp_df = forecast.set_index('ds')[['yhat','yhat_lower','yhat_upper']].join(pro_df.set_index('ds'))
```

Puis, on visualise les données.

```
plt.figure(figsize=(17,8))
plt.plot(cmp_df['y'],color='red')
plt.plot(cmp_df['yhat_lower'],color='#ff7a7d')
plt.plot(cmp_df['yhat_upper'],color='#6aaaf7')
plt.plot(cmp_df['yhat'],color='#1be038')
plt.legend()
plt.show()
```

On constate donc que Prophete est plutôt juste et les prix sont dans les intervalles de confiance.

Fonctionnalités à ajouter

-On pourrait appliquer le modèle à d'autres données, comme l'or, la bourse

-Créer une application web avec une interface utilisateur (nécessite des serveurs)

-Proposer d'autres modèles comme du calcul stochastique, des régressions linéaires