Project :
Prédiction des variations des indices boursiers grâce aux actualités

DI GIUSTO UGO

Sommaire :

I) Introduction

La Bourse est une forme de placement qui peut offrir une alternative aux différentes offres d'épargne des banques qui ne suivent généralement pas l'inflation.
Mais celle-ci peut s'avérer risquée et compliquée pour n'importe qui. Elle demande de se tenir à jour constamment pour s'assurer que le rendement de ses placements reste compétitif.

Ce projet a pour but d'amoindrir la charge du suivi des informations en utilisant l'intelligence artificielle afin de prévoir les variations futures des indices boursiers par leurs mouvements internes mais aussi par l'actualité qui peut avoir un impact important sur ceux-ci.

C'est à l'aide d'un dashboard que les différentes personnes s'intéressant à leurs placements pourront suivre les prédictions à long terme réalisées par l'IA, et ainsi faciliter leurs prises de décision.

II) Réalisation d'une IA

A. Données

1. Collecte

i) données textuelles

La réalisation de toute IA commence par la collecte de données. Pour ce projet, nous avons besoin de deux types de données :

- Des articles de journaux, de préférence liés à l'actualité boursière.
- Un historique des données boursières des indices.

C'est en cherchant sur www.boursorama.com des données boursières que je suis tombé sur un nombre important d'actualités relatives à la bourse, j'ai donc décidé d'utiliser le scraping afin d'y collecter les articles nécessaires à l'entraînement de mes modèles.

Pour ce faire et parce que j'avais déjà commencé à scraper l'historique des indices boursiers avec, j'ai utilisé Sélénium afin de créer un script de scraping des différents articles.

Sélénium n'étant pas optimisé pour scraper de telle quantité de données, j'ai dû réaliser du multiprocessing pour scraper plus vite, pour une mise à jour du code, j'utiliserais plutôt scrapy bien plus indiqué pour ce genre de tâches.

Voici quelques fonctions de scraping que j'ai réalisé :

```
def get_last_page(driver):

    wait = WebDriverWait(driver, 2)
    element = wait.until(
        EC.element_to_be_clickable((By.XPATH,
      "/html/body/main/div/div[1]/div[4]/div[1]/article/div[2]/div/div/div[2]/div[2
      ]/div/a[4]")))
    element.click()
    return driver.current_url.split('-')[1]
```

Cette fonction permet de trouver la dernière page contenant les urls des articles.

```
def extract_article_url(driver, page_url, urls : Queue):
    driver.get(page_url)
    wait = WebDriverWait(driver, 2)
    elements = wait.until(
        EC.presence_of_all_elements_located((By.XPATH, "//div[contains(@class,
      'c-latest-news')]//ul/li")))
    for element in elements:
        date = element.find_element(By.CLASS_NAME, "c-list-news__date").text
        source = element.find_element(By.CLASS_NAME,

      "c-source__name.c-source__name").text.replace('•', '')
        element = element.find_element(By.CLASS_NAME, "c-link.c-link--small")
        url = element.get_attribute('href')
        title = element.get_attribute('title')
        urls.put(Url_data(title, url, date, source, 0))
```

Celle-ci permet de récupérer tous les urls des articles ainsi que le titre et la date de parution de ceux-ci

```
def crawler(urls : Queue, meta_datas : Queue,no_data : Queue, progress_counter :
      Value,thread_name = 'pas de nom',download_path = output_data):

    with webdriver.Firefox(options=options) as driver:
        # go to page driver
        char_interdit = ["<", ">", ":", "\"", "/", "\\", '|', '?', '*']
        driver.get("https://www.boursorama.com/")
        driver.maximize_window()

        # pass cookie consent
        wait = WebDriverWait(driver, 2)
        element = wait.until(EC.element_to_be_clickable((By.CLASS_NAME,
      "didomi-continue-without-agreeing")))
```

```
    element.click()
    while not urls.empty():
        url = urls.get()
        title = url.title
        for char in char_interdit:
            title = title.replace(char, " ")
        date = url.date
        date_path = format_date(date)
        os.makedirs(download_path+"/"+date_path, exist_ok=True)
        source = url.source
        attempt = url.attempt
        url = url.url
        if random() < 0.01:
            driver.get(url)
            gc.collect()
        else:
            driver.get(url)
        as_content = False
        try:
            element =
wait.until(EC.presence_of_element_located((By.CLASS_NAME,
"c-news-detail__content")))
            with open(download_path+"/"+date_path+'/'+title + ".txt", 'w') as
file:
                file.write(element.text)
            as_content = True
        except:
            if attempt<3:
                urls.put(Url_data(title, url, date, source, attempt+1))
                add_counter(progress_counter, -1)
            else:
                no_data.put(Url_data(title, url, date, source, attempt+1))
        finally:
            add_counter(progress_counter)

        impact = set()
        if as_content:
            try:
                elements =
wait.until(EC.presence_of_all_elements_located((By.CLASS_NAME,

"c-table__cell.c-table__cell--dotted.c-table__cell--wrap")))
                for element in elements:
                    impact.add(element.text)

                meta_datas.put(Meta_data(title, url, date_path, source,
frozenset(impact)))
            except:
                meta_datas.put(Meta_data(title, url, date_path, source,
frozenset(impact)))
                gc.collect()
```

Cette dernière fonction extrait les articles des urls se trouvant dans une Queue et de les enregistrer dans un fichier par rapport à la date. Il collecte aussi quelques métadonnées qui seront enregistrées dans une Queue. Un premier traitement est appliqué au titre, si l'article est une vidéo ou n'a juste pas de contenu textuel alors il n'est pas sauvegardé.

ii) données boursières

Pour les données historiques boursières, j'ai au final jeté mon dévolu sur Yfinance une api qui permet d'importer directement les dernières valeurs des indices boursiers.
Pour cela, il m'a suffi de constituer un dictionnaire des symboles des indices boursiers qui m'intéressait.
voici avec lequel je suis partie.

{'^FCHI': "CAC 40", '^GSPC': "S&P 500", '^DJI': "Jones Industrial Average",
'^IXIC': "NASDAQ Composite", '^NYA': "NYSE COMPOSITE (DJ)",
"^XAX": "NYSE AMEX COMPOSITE", '^RUT': "Russell 2000", '^FTSE': "FTSE 100",
"^VIX": "CBOE Volatility Index", "^GDAXI": "DAX PERFORMANCE-INDEX"}

```
index_info = {}
columns = ["Open","High","Low","Close","Volume"]
for sym in symbol.key():
   tmp = tickers.tickers[sym].history(period="max")
   tmp.index= tmp.index.tz_convert(None)
   tmp.index = tmp.index.date
   index_info[sym]=tmp[columns]
```

Ce script me permettant ensuite de récupérer les données de ces indices avec l'api.

2. Analyse

i) données textuelles

Voici la forme qu'ont nos articles :

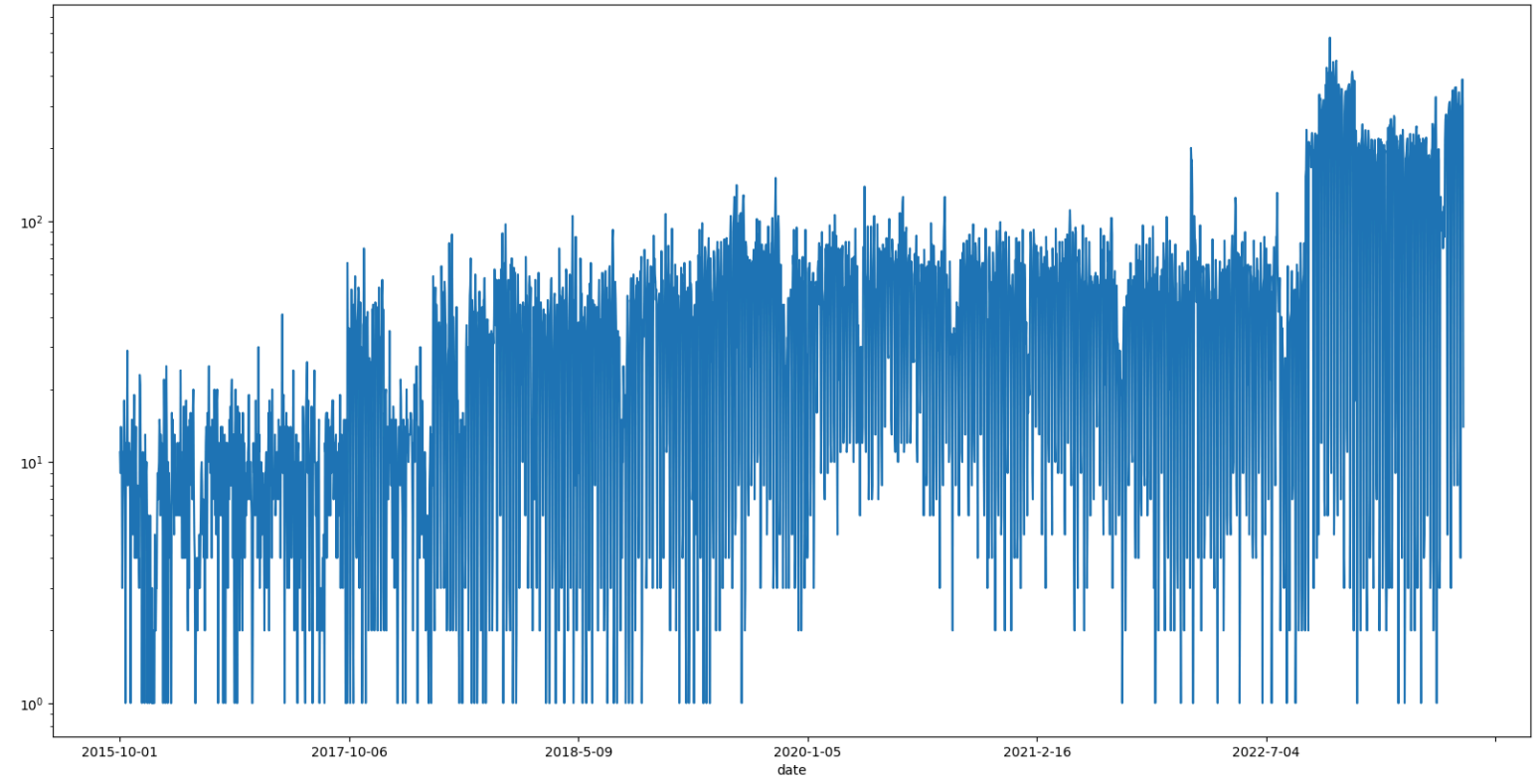
	date	title	url	impact	contenu
0	2023-12-19	Les pilotes de Southwest Airlines parviennent ...	https://www.boursorama.com/bourse/actualites/l...	(SOUTHWEST AIRLIN, UNITED AIRLINES, AMERICAN A...	(Ajoute la valeur de l'accord au paragraphe 1)...
1	2023-12-19	FedEx manque à l'appel, réduit ses prévisions ...	https://www.boursorama.com/bourse/actualites/f...	(FEDEX, UPS (UNITED PARCEL SER.) B)	(Nouveau dans l'ensemble, ajoute des détails, ...
2	2023-12-19	AAR Corp AIR.N devrait afficher un bénéfice de...	https://www.boursorama.com/bourse/actualites/a...	(AAR)	* AAR Corp AIR.N AIR devrait afficher ...
3	2023-12-19	Steelcase Inc. publie ses résultats pour le tr...	https://www.boursorama.com/bourse/actualites/s...	(STEELCASE RG-A)	* Steelcase Inc SCS.N a déclaré un bénéf...
4	2023-12-19	FedEx Corp. publie ses résultats pour le trime...	https://www.boursorama.com/bourse/actualites/f...	(FEDEX)	* FedEx Corp FDX.N a déclaré un bénéfice...
...	...	...	...	...	...
150264	2015-9-17	TxCell présentera les derniers développements ...	https://www.boursorama.com/bourse/actualites/t...	(TXCELL)	Si vous souhaitez recevoir en temps réel toute...
150265	2015-3-24	PAREF 2014 annual results	https://www.boursorama.com/bourse/actualites/p...	(PAREF)	Paris, 19 March 2015, 5.35 pm\n2014 Annual Res...
150266	2015-9-17	ESKER Résultats semestriels 2015	https://www.boursorama.com/bourse/actualites/e...	(ESKER)	Doublement du résultat\n\nCroissance à deux ch...
150267	2015-3-20	Petercam Avis aux actionnaires	https://www.boursorama.com/bourse/actualites/p...	(DPAM L BONDS EUR CORPORATE HIGH YIELD W)	Par la présente, le Conseil d'Administration i...
150268	2015-9-17	Remontée des taux « un facteur de soutien ma...	https://www.boursorama.com/bourse/actualites/r...	()	Pour H2O AM, la volatilité va redevenir plus é...

150269 rows × 5 columns

Avec le bout de code suivant on retire tous les articles de moins de 200 caractères

```
df_articles["lenght"] = df_articles["contenu"].apply(lambda x : len(x))
df_articles = df_articles[df_articles['lenght']>=200]
```

Nous nous retrouvons finalement avec 148 039 articles.



Voici la répartition du nombre d'articles par jour sur une échelle logarithmique, on remarque deux choses post 2022-08, on a une forte augmentation du nombre d'articles et avant 2018 il y a bien moins d'articles.
De plus, les pics que l'on voit correspondent en fait au week-end, la bourse étant fermée, le site ne propose que peu d'articles.

On a en moyenne 50 articles par jour.
Plus précisément après août 2022, on a une moyenne de 108 articles par jour.
Entre août 2022 et 2018 on a en moyenne 42.5 articles par jour et avant 2018, on tourne autour de 13.5 articles par jour.

ii) données boursières

Les données boursières se présentent sous cette forme :

Date	Open	High	Low	Close	Volume
2014-03-13 04:00:00	1869.060059	1874.400024	1841.859985	1846.339966	3670990000
2014-03-14 04:00:00	1845.069946	1852.439941	1839.569946	1841.130005	3285460000
2014-03-17 04:00:00	1842.810059	1862.300049	1842.810059	1858.829956	2860490000
2014-03-18 04:00:00	1858.920044	1873.760010	1858.920044	1872.250000	2930190000
2014-03-19 04:00:00	1872.250000	1874.140015	1850.349976	1860.770020	3289210000

La plus récente parmi celles qui nous intéresse commence en 1995, jusqu'à aujourd'hui

```
df.index= df.index.tz_convert(None)
df.index = df.index.date
```

J'utilise ces lignes de code pour retirer l'heure, car cela me permettra de joindre les données des différents indices étant donné qui sort a des heures différentes .

3. Traitement

i) embedding

Le premier modèle que je compte réaliser est un transformer qui prend directement les articles en entrée, j'ai donc besoin de les embedder.
Pour cela, j'utilise le modèle pré entraîné trouvable sur huggingface.co intfloat/multilingual-e5-small qui, au moment du choix de celui-ci, était le meilleur petit embedding disponible sur le site.

```
tokenizer = AutoTokenizer.from_pretrained('intfloat/multilingual-e5-small')
model = AutoModel.from_pretrained('intfloat/multilingual-e5-small')

input_texts = df_articles[0:10000]
input_texts = np.char.add ("query: ",input_texts.title.values).tolist()
batch_dict = tokenizer(input_texts, max_length=512, padding=True,
     truncation=True, return_tensors='pt')
outputs = model(**batch_dict)
embeddings_tmp = average_pool(outputs.last_hidden_state,
     batch_dict['attention_mask'])
del outputs, batch_dict
gc.collect()
embeddings = F.normalize(embeddings_tmp, p=2, dim=1).detach()
del embeddings_tmp
gc.collect()
for i in range(1, len(df_articles)//10000+1):
   input_texts = df_articles[10000*i:10000*(i+1)]
   input_texts = np.char.add ("query: ",input_texts.title.values).tolist()
   batch_dict = tokenizer(input_texts, max_length=512, padding=True,
     truncation=True, return_tensors='pt')
   outputs = model(**batch_dict)
   embeddings_tmp = average_pool(outputs.last_hidden_state,
     batch_dict['attention_mask'])
   del outputs, batch_dict
   gc.collect()
   embeddings_tmp = F.normalize(embeddings_tmp, p=2, dim=1).detach()
   embeddings = torch.cat((embeddings, embeddings_tmp),dim=0)
   del embeddings_tmp
   gc.collect()

torch.save(embeddings, 'embending.pt')
```

Ceci est le script qui importe le modèle d'embedding, convertit le titre des articles en tenseur Torch puis les sauvegarde.
Il suffit ensuite de faire correspondre la position dans le dataframe de l'article avec le premier index du tenseur pour retrouver la date du tenseur.

Après cette étape j'utilise les fonctions suivantes :

```
def get_plage_date(df, traget_date, distance_view, nb_max_jour,
    column_date_name='date'):
  date_filter = df[column_date_name] < (traget_date +
    datetime.timedelta(days=-distance_view))
  date_filter &= df[column_date_name] >= (traget_date +
    datetime.timedelta(days=-(distance_view + nb_max_jour)))
  return df[date_filter]


def format_train_data(embeddings, check_date, index_info, distance_view,
    nb_max_jour, nb_feature,
                  date_range=('2018-01-01', '2023-12-20'),
    histo_nb_feature=384):
  train_data = {}
  index_info.index = index_info.index
  date_filter = index_info.index <= datetime.datetime.strptime(date_range[1],
    '%Y-%m-%d')
  date_filter &= index_info.index >= datetime.datetime.strptime(date_range[0],
    '%Y-%m-%d')
  index_range = index_info[date_filter].copy()
  index_all = index_info.resample('D').mean().fillna(0).copy()
  index_all["Date"] = index_all.index
  for row in index_range.iterrows():
      plage_date = get_plage_date(check_date, row[0], distance_view,
    nb_max_jour)
      plage_date = plage_date.set_index("date")
      plage_date_index = get_plage_date(index_all, row[0], distance_view,
    histo_nb_feature, column_date_name='Date')
      min_nb_feature = min(len(plage_date), nb_feature)
      # tmp = pd.Series({date:1/(i) for date,i in
    zip(plage_date.index.unique(),range(1,nb_max_jour))})
      plage_date = plage_date.sample(n=min_nb_feature).sort_index()
      target = (get_plage_date(index_all, row[0], 0, distance_view,
    column_date_name='Date').drop(
          columns='Date') + 1).prod().copy()
      train_data[row[0]] = {'index': plage_date.old_index.values
          , "nb_padding": nb_feature - min_nb_feature
          , "open": target.Open
          , "high": target.High
          , "low": target.Low
          , "close": target.Close
          , "volume": target.Volume
          , "open_histo": plage_date_index["Open"].values
          , "high_histo": plage_date_index["High"].values
          , "low_histo": plage_date_index["Low"].values
          , "close_histo": plage_date_index["Close"].values
          , "volume_histo": plage_date_index["Volume"].values}

  return train_data
```

Elle me permet de rapidement créer des données d'entraînement du nombre d'articles max que je veux, dans une période donnée, de définir le padding nécessaire s'il manque des articles, d'y associer un historique des indices boursiers ainsi que de créer une target de la valeur de l'indice en temps +n à choisir.

ii) Finbert

Mon second modèle sera basé sur des timeseries plus traditionnelles, j'ai donc opté pour l'utilisation Finbert un modèle de sentiment analysis spécialisé dans la finance afin de résumer jour par jour le nombre d'articles positifs, négatifs, neutres ainsi que le score moyen de ces labels.
J'ai suivi approximativement le même procédé que pour l'embedding, la différence principale et que par manque de temps et vu que Finbert est optimisé pour des articles en anglais, j'ai dû utiliser un autre modèle pour traduire mes articles du français à l'anglais, cela peut jouer un rôle dans les performances finales du modèle.

Pour la traduction j'ai utilisé le modèle facebook/mbart-large-50-many-to-many-mmt et pour finbert j'ai utilisé la version suivante : ProsusAI/finbert tous les deux trouvables sur huggingface Co.

Voici le dataframe que j'obtiens:

	title	trad	score_label	score_label_value	score	score_positive	score_neutral	score_negative
9237	L'offre de cuivre de Codelco aux clients d'Asi...	Codelco's copper supply to customers in Southe...	negative	-1	0.972010	0.000000	0.000000	0.97201
102921	France Coronavirus-205 hospitalisations de moi...	France Coronavirus-205 hospitalizations less t...	neutral	0	0.805961	0.000000	0.805961	0.00000
1351	bp accord avec BWT Alpine F1 Team	bp agreement with BWT Alpine F1 Team	positive	1	0.805793	0.805793	0.000000	0.00000
35191	FL Entertainment nomination d'Alexia Laroche...	FL Entertainment appoints Alexia Laroche-Joube...	neutral	0	0.914649	0.000000	0.914649	0.00000
14599	Kone finalise la vente de ses activités en Ru...	Kone completes sale of its activities in Russia	neutral	0	0.859777	0.000000	0.859777	0.00000

Il me suffit ensuite d'utiliser le script suivant pour agréger les données par jour puis les joindre avec les indices boursiers.

```
score = score.copy()
score[date] = pd.to_datetime(tmp[date])
score2 = tmp[["date","score_label_value"]].copy()
score2['positive_label_value'] = score2["score_label_value"].apply(lambda x : 1
    if x>0 else 0)
score2['négative_label_value'] = score2["score_label_value"].apply(lambda x : 1
    if x<0 else 0)
score2 = score2.groupby(by=date).sum().copy()
score =
    score[["date","score_label_value","score_positive","score_neutral","score_neg
    ative"]].groupby(by = date,as_index= True).mean()
score['positive_label_total'] = score2['positive_label_value'].values
score['négative_label_total'] = score2['négative_label_value'].values
```

```
df_final_with_score = pd.merge(df_index, score,how='inner', left_index=True,
    right_index=True).copy()
df_final_with_score
```

iii) les indices

Les indices boursiers n'ont besoin que d'être scaler avant d'être envoyés à l'algorithme pour cela j'utilise la classe StandardScaler de sklearn.

J'ai tout de même essayé quelques traitements supplémentaires en plus de conserver les données telles quelles, par exemple j'ai utilisé la méthode pct_change des DataFrame Pandas afin d'obtenir le pourcentage de variation des indices.

J'ai aussi utilisé des techniques de lissage de courbe comme celle-ci pour simplifier les données et voir comment réagissent les algorithmes :

```
def exponential_moving_average(prices, period, weighting_factor=0.2):
   ema = np.zeros(len(prices))
   sma = np.mean(prices[:period])
   ema[period - 1] = sma
   for i in range(period, len(prices)):
       ema[i] = (prices[i] * weighting_factor) + (ema[i - 1] * (1 -
     weighting_factor))
   return ema
```

B. Création et entraînement des modèles

1. Transformers

Pour la création du transformer j'ai utilisé Pytorch, il existe plusieurs modules de transformer encoder et décoder. Seule la partie positional encoding n'est pas intégrée, il y a beaucoup de ressources sur comment la mettre en place.

Voici le modèle utilisé dans cette partie

```
class PositionalEncoding(nn.Module):
    def __init__(self, d_model, max_seq_length):
        super(PositionalEncoding, self).__init__()

        pe = torch.zeros(max_seq_length, d_model)
        position = torch.arange(0, max_seq_length, dtype=torch.float).unsqueeze(1)
        div_term = torch.exp(torch.arange(0, d_model, 2).float() *
      -(math.log(10000.0) / d_model))

        pe[:, 0::2] = torch.sin(position * div_term)
        pe[:, 1::2] = torch.cos(position * div_term)

        self.register_buffer('pe', pe.unsqueeze(0))

    def forward(self, x):
        return x + self.pe[:, :x.size(1)]



class Transformer(nn.Module):
    def __init__(self, output_size, d_model, num_heads, num_layers, d_ff,
      max_seq_length, dropout):
        super(Transformer, self).__init__()
        self.positional_encoding = PositionalEncoding(d_model, max_seq_length)

        encoder_layers = nn.TransformerEncoderLayer(d_model, num_heads,d_ff
      batch_first=True)
        self.encoder = nn.TransformerEncoder(encoder_layers, num_layers)

        decoder_layers = nn.TransformerDecoderLayer(d_model, num_heads,d_ff
      batch_first=True)
        self.decoder = nn.TransformerDecoder(decoder_layers, num_layers)
```

```
        self.fc = nn.Linear(d_model * max_seq_length, output_size)

        self.dropout = nn.Dropout(dropout)

    def generate_mask(self, src):
        src_mask = (src != 0)
        return src_mask

    def forward(self, src, tgt):
        src = self.dropout(self.positional_encoding(src))
        src = self.encoder(src)

        tgt = self.dropout(self.positional_encoding(tgt))
        src = self.decoder(src, tgt)

        src = torch.flatten(src, start_dim=1)
        src = self.fc(src)
        src = torch.squeeze(src, 1)
        return src
```

Dans ce modèle src, l'input de l'encoder était embedding, parce que la dimension de celle-ci est de 384 par conséquent c'est aussi la dimension du modèle, et du tgt l'input de l'encoder qui est l'historique de l'indice, je lui fournissais donc plus d'un an d'historiques.

Pour les hyperparamètres j'ai fait varier les couches de feedforward de 128 à 2048, le nombre de heads entre 4 à 16, le nombre de couche entre 1 à 3, le dropout de 0 à 0.5, et le learning rate de 1e-3 à 1e-7

J'ai testé cela pour un nombre variable d'actualités par jour, ainsi que sur des fenêtres de temps différents en utilisant pour loss la MSE et l'optimiseur Adam.

2. LSTM

Étant donné la nature des timeseries de mes données, il était important pour moi de voir comment celles-ci performeront avec et sans présence d'actualité sur des modèles LSTM.
Étant donné que ceux-ci sont connus pour fonctionner correctement avec ce genre de problème.

voici l'architecture de modèle que j'ai utilisé :

```
class LSTM(nn.Module):
    def __init__(self, input_dim, hidden_dim, num_layers, output_dim):
        super(LSTM, self).__init__()
```

```
        self.hidden_dim = hidden_dim
        self.num_layers = num_layers
        self.lstm = nn.LSTM(input_dim, hidden_dim, num_layers, batch_first=True)
        self.fc = nn.Linear(hidden_dim, output_dim)

    def forward(self, x):
        h0 = torch.zeros(self.num_layers, x.size(0),
      self.hidden_dim).requires_grad_()
        c0 = torch.zeros(self.num_layers, x.size(0),
      self.hidden_dim).requires_grad_()
        out, (hn, cn) = self.lstm(x, (h0.detach(), c0.detach()))
        out = self.fc(out[:, -1, :])

        return out
```

Tout comme précédemment j'ai fait varier les hyperparamètres, j'ai remarqué que lui donner trop de couche ou de dimension lui était plus nuisible, pas contre un nombre important d'epoch lui donnait de meilleurs résultats.

3. PatchTST

PatchTST est un modèle récent qui utilise du patching, une décomposition de la donnée des timeseries qui sert d'input a un transformer.
Celui-ci a fait beaucoup parler de lui car c'est le premier modèle de timeseries qui utilise uniquement les transformers qui obtiennent de très bonnes performances.

Le code de ce modèle a été mis à disposition par ces créateurs sur github : https://github.com/yuqinie98/PatchTST il est maintenant aussi sur huggingface.co

Pour lancer l'entraînement avec le repo il faut placer un dataset dans le fichier dateset/ et remplir les arguments shell suivants :

```
if [ ! -d "./logs" ]; then
    mkdir ./logs
fi

if [ ! -d "./logs/LongForecasting" ]; then
    mkdir ./logs/LongForecasting
fi
seq_len=30 #336
model_name=PatchTST

root_path_name=./dataset/
data_path_name=custom_dataset_wt_score_V4.csv
model_id_name=test_custom_dataset_score
```

```
data_name=custom

random_seed=2021
for pred_len in 1
do
   python3.12  run_longExp.py \
     --random_seed $random_seed \
     --is_training 1 \
     --root_path $root_path_name \
     --data_path $data_path_name \
     --model_id $model_id_name_$seq_len'_'$pred_len \
     --model $model_name \
     --data $data_name \
     --features MS \
     --seq_len $seq_len \
     --pred_len $pred_len \
     --enc_in 10 \
     --freq d \
     --label_len 0 \
     --e_layers 4 \
     --n_heads 32 \
     --use_gpu true\
     --batch_size 16\
     --d_model 128 \
     --d_ff 128 \
     --dropout 0.2\
     --fc_dropout 0\
     --head_dropout 0.1\
     --patch_len 32\
     --stride 2\
     --des 'Exp' \
     --train_epochs 100\
     --patience 5\
     --lradj 'TST'\
     --pct_start 0.2\
     --itr 1  --learning_rate 0.0001
done
```

C’est ici que l’on peut choisir les hyperparamètres que l’on souhaite.

C. Exploration des résultats

1. Comparaison des métriques

Le premier modèle orienté NLP n'a pas porté ses fruits, celui-ci converge systématiquement vers la moyenne haute.

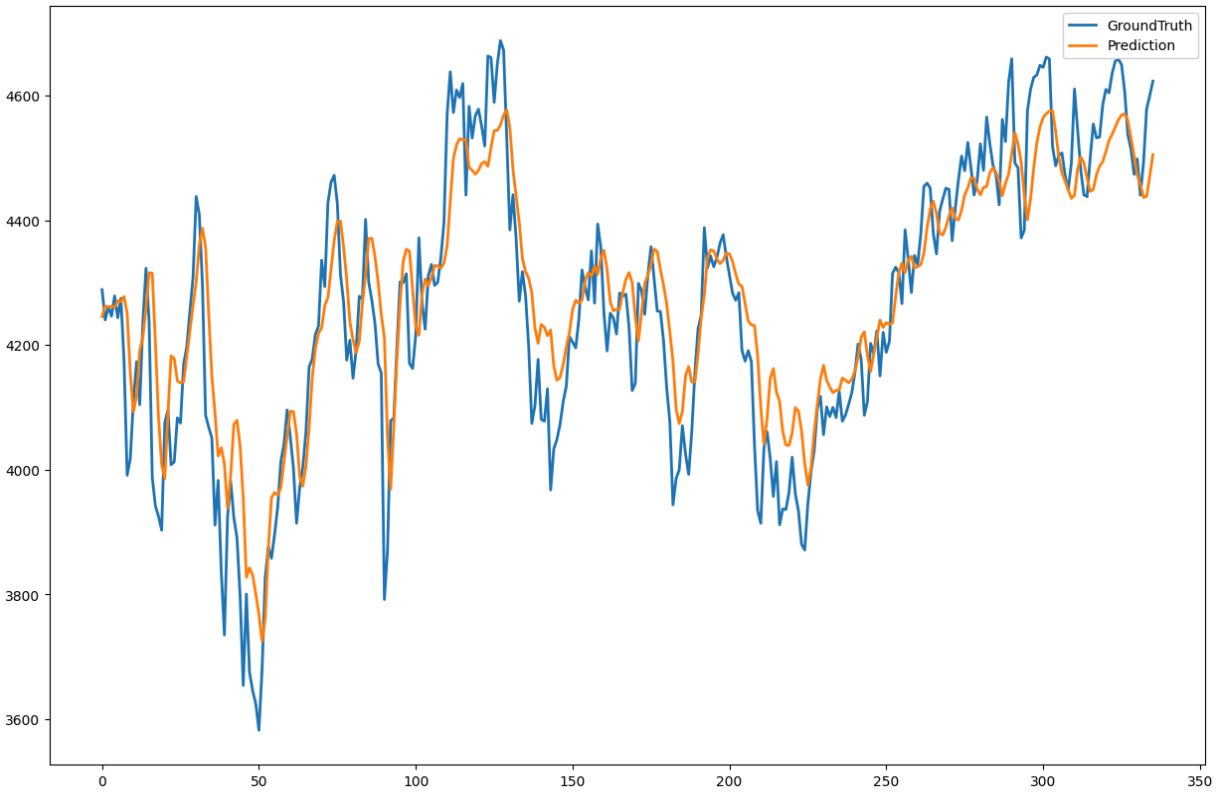
Ce comportement est probablement le résultat de l'incapacité pour le modèle de lier les variations des indices avec l'ensemble des articles qui représente une masse données immense comparées au nombre d'itération possible avec 6 ans historiques.

De plus la récence de PatchTST et l'utilisation du patching me fait croire que la forme de transformer que j'utilisais était incompatible avec la structure de données des timeseries.

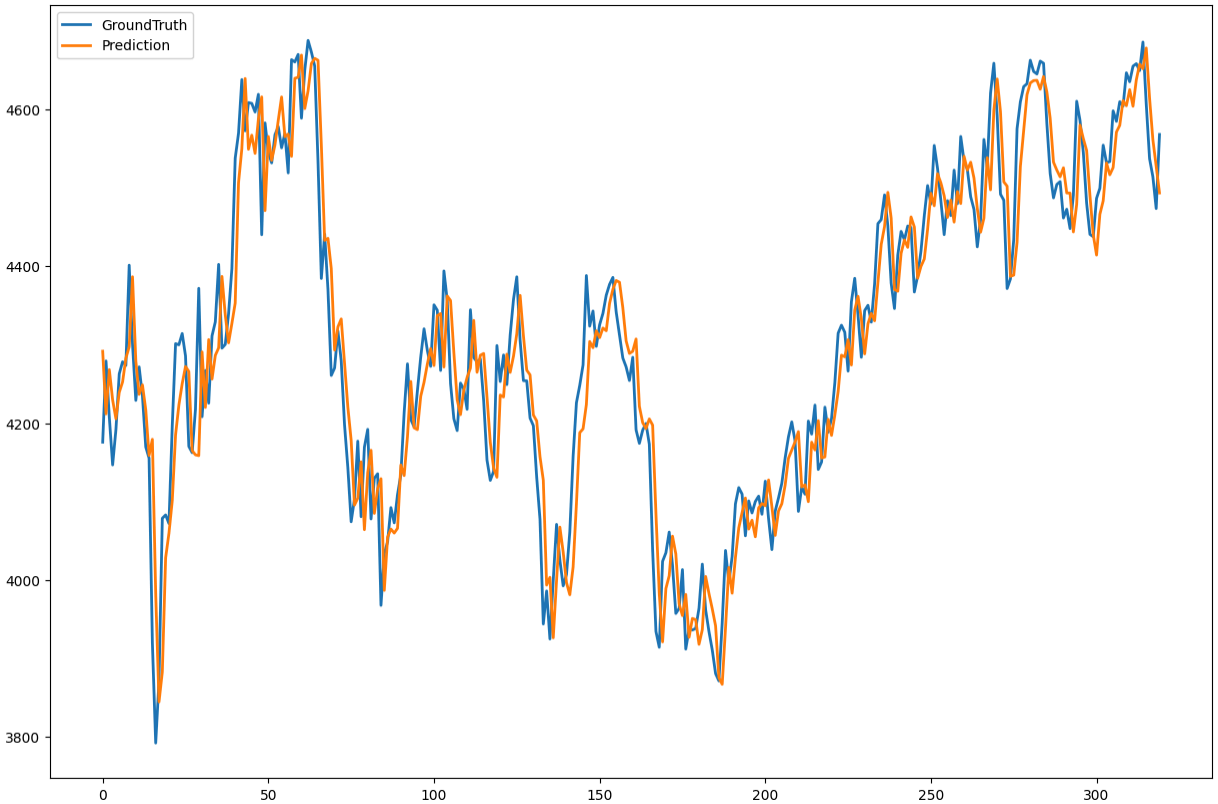
Intéressons-nous plutôt au résultat du LSTM et de PatchTST
Dans le cas où on aurait 30 jours historiques pour prédire+1 du CAC40.

Même si l'on vise plutôt des prédictions sur le long terme, on sait que les LSTM donnent des résultats intéressants dans ces cas.

	MSE	MAE	RMSE
PatchTST	4472	52	66,9
LSTM	10188	77,5	100,9



LSTM

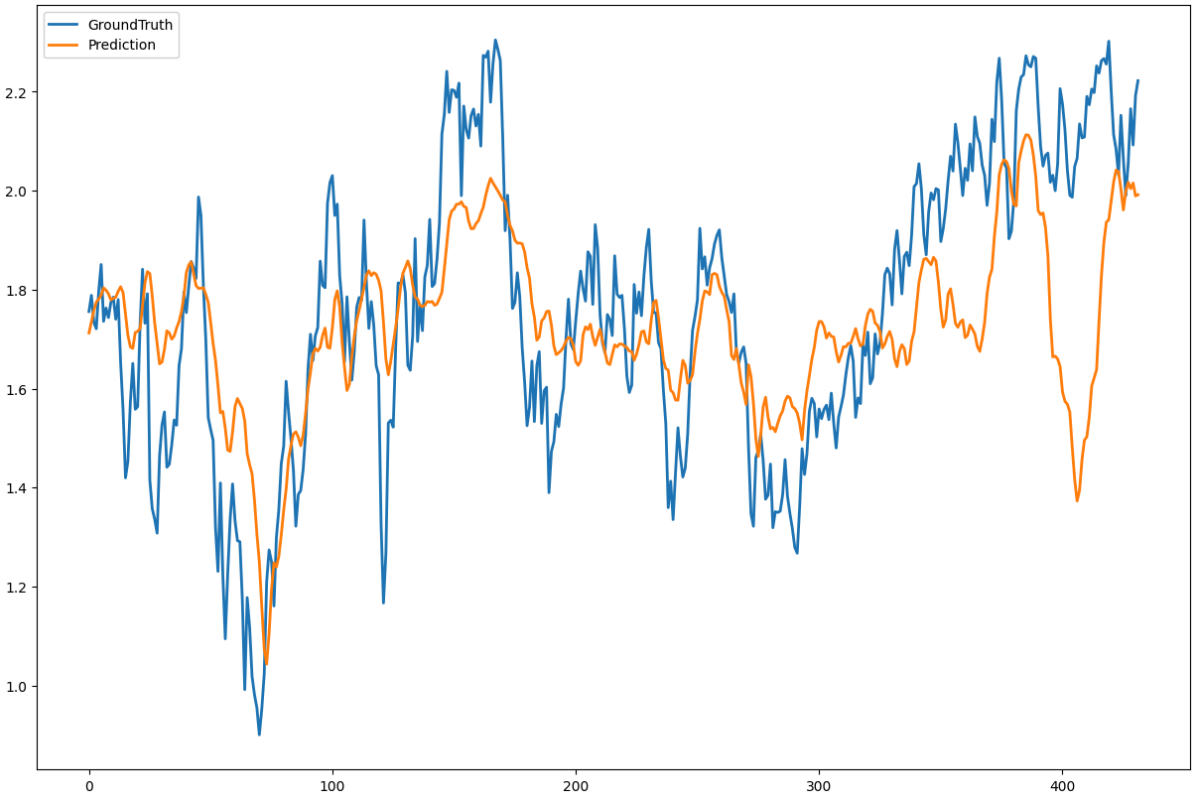


PatchTST

Sur ce premier jet, PatchTST est en tête, ajoutons maintenant les infos d'actualité, ici je vais garder les valeurs scaler car il est compliqué de déscaler le LSTM.

	MSE	MAE	RMSE
PatchTST	0,074	0,212	0,321
LSTM	0,014	0,095	0,118
PatchTST actu	0,068	0,202	0,262
LSTM actu	0,047	0,169	0,219

/!\ le scale n'est pas consistant entre les deux modèles, il faut regarder PatchTST avec PatchTST actu, et le LSTM avec LSTm actu.



LSTM

PatchTST

On se rend bien compte que l'ajout des actualités améliore un petit peu PatchTST alors que ça dessert le LSTM, c'est probablement dû au fait que PatchTST est fait pour prendre en compte plusieurs séquences là où le LSTM est plus fait pour une séquence à la fois

Étant donné que l'on souhaite du plus long terme, partons sur 30 jours d'historique pour avoir j+30 sans les actualités :

	MSE	MAE	RMSE
PatchTST	62524	194,3	250
LSTM	850244	862	922

Les LSTM performent moins bien, PatchTST restent donc au-dessus dans toutes les circonstances.
Essayons sa configuration qui permet de voir le plus loin sans perdre en pertinence.

sans actualité	avec actualité

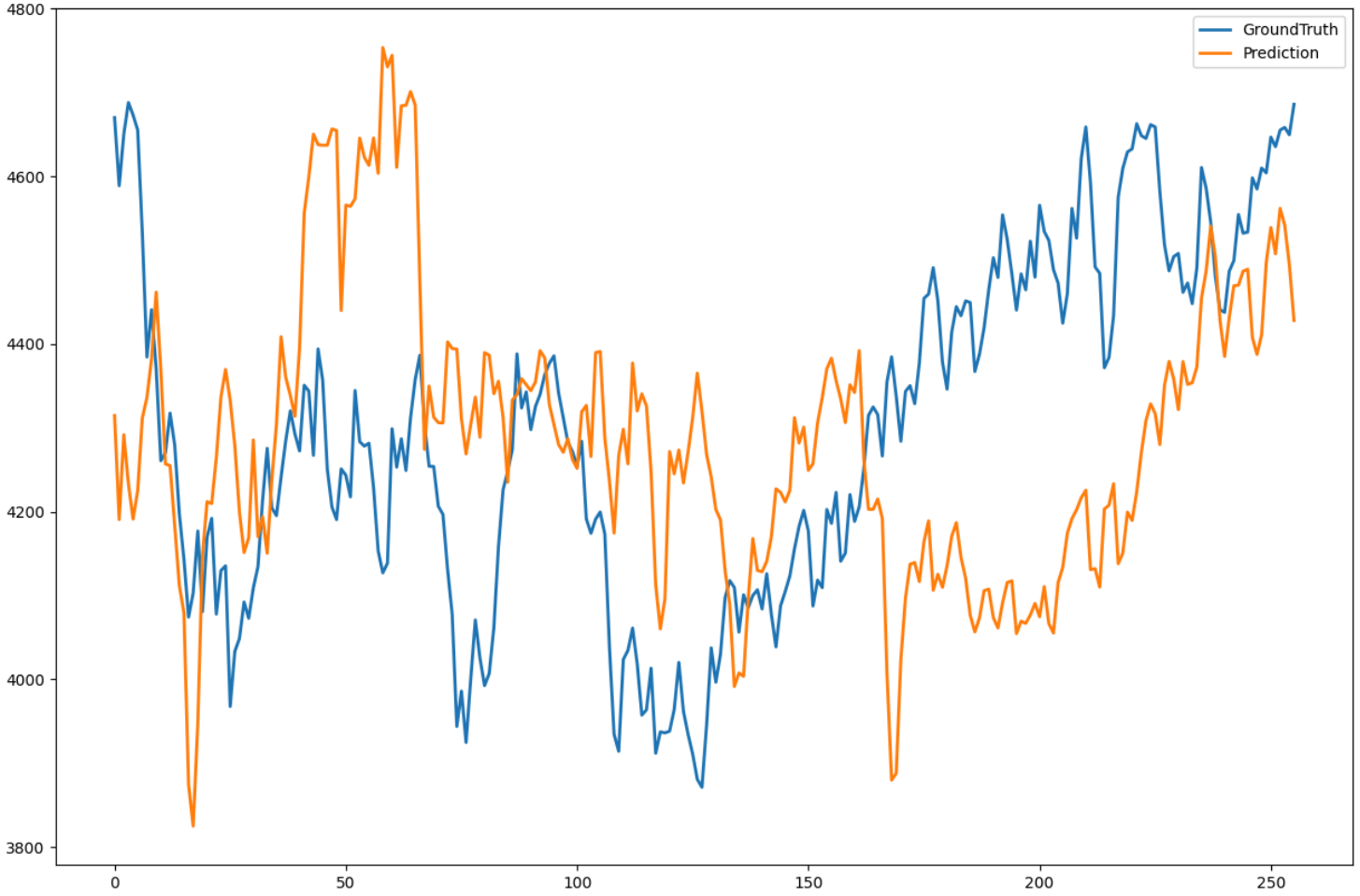
	MSE	MAE	RSE	MSE	MAE	RSE
30j j+1	4 472,00	52,00	0,32	4 119,00	49,00	0,30
30j j+15	22 693,00	114,00	0,72	23 876,00	116,00	0,74
30j j+30	38 143,00	150,00	0,95	39 407,00	150,00	0,96
30j j+60	57 169,00	191,00	1,19	59 083,00	190,00	1,21
30j j+90	74 232,00	218,00	1,41			
60j j+30	49 727,00	171,00	1,08			
60j j+60	69 916,00	210,00	1,31			
60j j+90	91 336,00	247,00	1,58			
15j j+1	4 062,00	49,00	0,30	4059	49	0,3
15j j+15	23 789,00	116,00	0,74	24128	117	0,75
15j j+30	38 461,00	151,00	0,95	39336	152	0,96
15j j+60	55 954,00	192,00	1,17	56588	192	1,18

2. Conclusion

Les résultats semblent clairs que PatchTST fonctionne assez bien, malheureusement l'ajout d'actualité n'a un impact que sur de courtes périodes, même s'il est intéressant de le noter, ce n'est pas ce que nous cherchons pour du placement, mais cela pourrait faire l'objet de plus de recherche pour du trading.

La configuration optimale semble être la suivante :

30 jours d'historique pour une vision à 60 j, ce qui si on remet les week-ends nous donne 84 j, soit presque 3 mois ce qui correspond à une vision à moyenne portée.



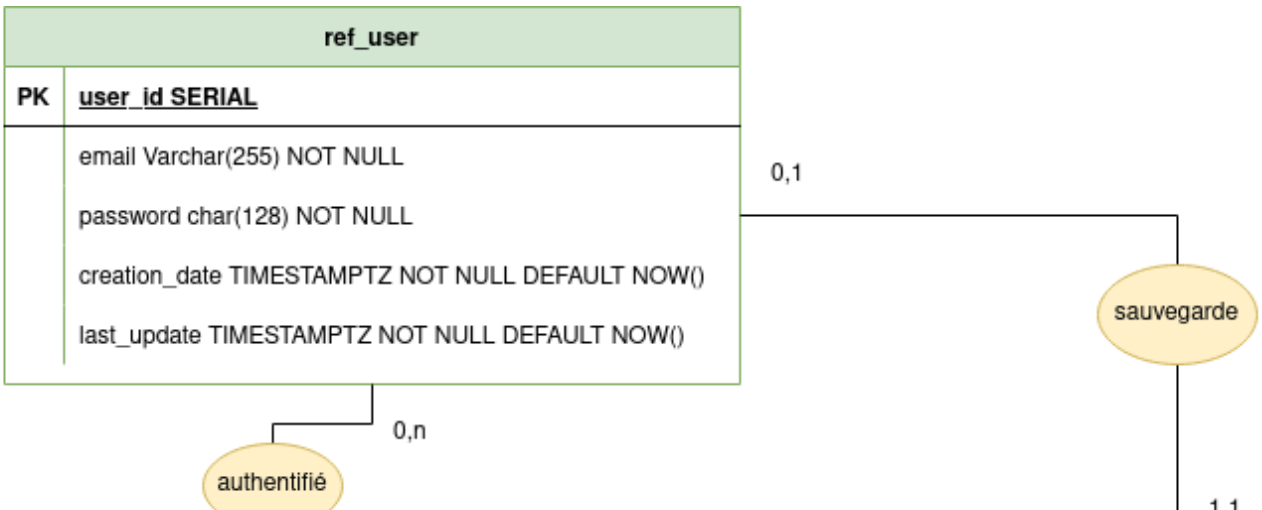
prévision à 60j

Le fait de ne pas utiliser l'actualité nous permet quand même d'utiliser plus de données historiques lors de l'entraînement.

III) Réalisation d'une app

A. Création d'une base de donnée

1. Schéma de la base



ref_user stocke les utilisateurs autorisés à accéder à l'application.
user_token stocke le token des utilisateurs afin qu'il puisse naviguer sur l'application après cette connexion.
user_favorite stocke l'index par défaut qui sera changé lorsque l'utilisateur se trouve sur le dashboard.

2. Requête de création

```
CREATE TABLE IF NOT EXISTS ref_user (
   user_id SERIAL PRIMARY KEY,
   email VARCHAR(255) UNIQUE NOT NULL,
   password VARCHAR(128) NOT NULL,
   creation_date TIMESTAMPTZ NOT NULL DEFAULT NOW(),
   last_update TIMESTAMPTZ NOT NULL DEFAULT NOW()
);



CREATE TABLE IF NOT EXISTS ref_user (
   user_id INT UNIQUE NOT NULL,
   token VARCHAR(128) NOT NULL,
   expire INT DEFAULT 3600,
   created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
   CONSTRAINT user_for_token
       FOREIGN KEY(user_id)
       REFERENCES ref_user(user_id)
);

CREATE TABLE IF NOT EXISTS user_favorite (
   fav_id SERIAL PRIMARY KEY,
   user_id INT UNIQUE NOT NULL,
   fav VARCHAR(5) NOT NULL,
   CONSTRAINT user_for_favorite
       FOREIGN KEY(user_id)
       REFERENCES ref_user(user_id)
)


INSERT into ref_user (email, password)
   VALUES  ('test@test.com',crypt('test@test.com',gen_salt('bf',8)));
```

Cette requête SQL permet de générer les tables de la base de données, les utilisateurs sont insérés hors app, car celle-ci n'est ouverte qu'à un nombre restreint de personnes.
Le mot de passe est tout de suite crypté lors de l'insertion d'un nouvel utilisateur.

3. Communication avec l'app

L'authentification, la création du token et la sauvegarde du favori sont gérées par un module appelé par le backend qui communique avec la base de données.

Ce module contient une classe permettant de se connecter à la base de donnée

```
class App_DB:
   def __init__(self):
      self.__dbname = 'app_project'
      self.__user = os.environ['DB_USER']
      self.__password = os.environ['DB_PWD']
      self.__host = 'localhost'
      self.__port = '5432'
      self.conn = psycopg2.connect(dbname=self.__dbname, user =self.__user,
                           password=self.__password, host=self.__host,
    port=self.__port)
```

Cette classe Classe contient plusieurs méthodes permettant de requêter la base de données

par exemple :

```
def check_user_password(self, email, password):
   with self.conn.cursor() as cur:
      cur.execute("""
          with is_confirm_user as (
          SELECT user_id
          ,(password = crypt(%s,password)) as responce
          ,email
          FROM ref_user
          WHERE email = %s
          )

      , create_token as (
          SELECT user_id
              ,CASE WHEN responce THEN
    crypt(md5(random()::text),gen_salt('bf',8)) ELSE null END as token
              ,email
          FROM is_confirm_user

      )
      , insert_token_if_valid as (
          INSERT into user_token (user_id,token)
          SELECT user_id,token
          FROM create_token
          WHERE token is not null
      )

      SELECT *
      FROM create_token
      """, (password, email))
```

```
        self.conn.commit()
        return cur.fetchone()
```

Cette méthode permet d'identifier la personne qui essaye de se connecter.
Elle répond à trois cas de figure, la personne n'existe pas en base, la personne existe mais n'a pas renseigné le bon mot de pas, la personne existe et le mot de passe est valide dans ce cas un token est généré en base et envoyé à l'app.

Les autres méthodes permettent de valider un token, supprimer les tokens de l'utilisateur, récupérer le favori de l'utilisateur, le modifier, ou le supprimer.

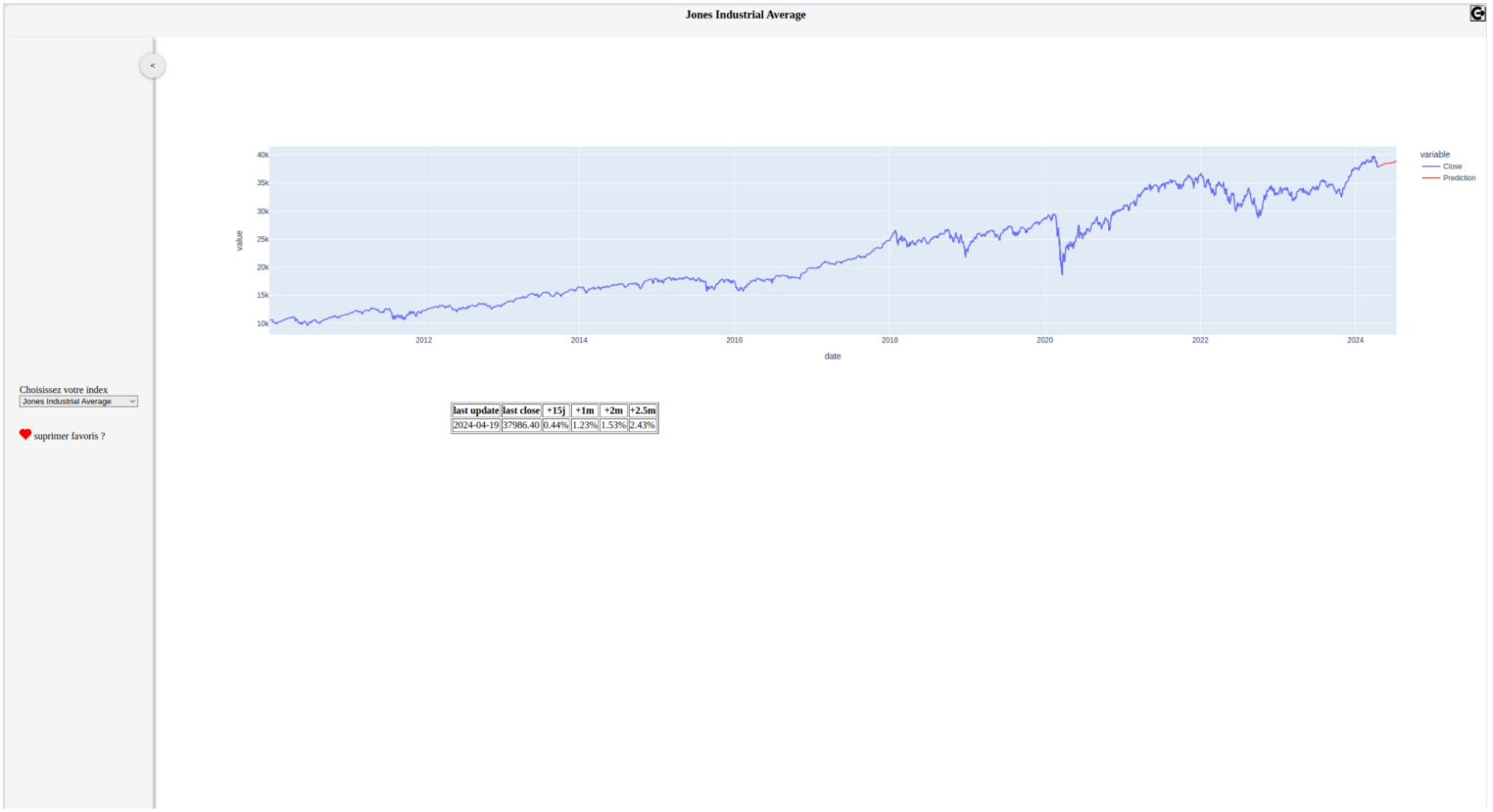
B. app
    1. Page de connection

Voici la page de connection

C’est le seul moyen d’entrer sur les urls importantes du site, si la personne n’est pas identifiée et qu'elle tente d'accéder à d'autres pages de l’app elle est directement redirigée vers cette page.

Les seules exceptions sont l’url root et cookie, qui me servent de test unitaire pour voir si l'application est lancée et si elle enregistre bien un token dans les cookies.

2. dashboard



Voici le dashboard, il contient un récapitulatif des prédictions sur les 2 prochains mois ainsi que l'historique des indices, ceux-ci sont stockés côté serveur sous la forme de HTML et sont appelés par du HTMX lorsque nécessaire.

IV) Synthèse

A. Conclusion

La prédiction des indices boursiers est loin d'être triviale et reste un problème complexe, bien que le fait de prendre l'actualité, n'est pas montré d'amélioration sur

le long terme, il est peut-être possible d’y arriver en modifiant la façon dont on présente les données à l'algorithme.

B. Piste d’amélioration

Il existe plusieurs pistes d’amélioration
En matière d’IA on pourrait voir comment se débrouille une IA génétique avec cette donnée si en la faisant parier sur la hausse ou la baisse des index.

Je pense qu’il est nécessaire de mettre en place une méthode de mesure des performances différentes de la MSE, MAE…
Le but serait par exemple de mesurer la capacité de l'algorithme à ranger les différents index, pour ce faire on pourrait imaginer un Gini à partir d'une courbe construite en prenant en compte le fait que le modèle ai eu raison sur la baisse ou la montée de l’index, ainsi que la proportion qu’il a donné à celle-ci