

Projet de Machine Learning.



Réalisé par :

Soukaina BOUCHANE

Nassima EL JAZOULI

Encadré par :

Pr. Lotfi EL AACHAK

Remerciements :

L'année universitaire touchant à sa fin, nous tenions à vous dire que vous avez été un excellent professeur ! Nous vous remercions d'avoir partagé vos connaissances avec nous, d'avoir toujours été juste dans votre éducation monsieur Lotfi EL AACHAK et de nous avoir toujours soutenus et aidés. Merci encore de la part de tous vos étudiants.

Tables des matières :

I.	Introduction :	5
II.	Objectif de la compétition :	6
III.	Vue d'ensemble sur nos données :	6
1.	Dossiers :	6
2.	Fichiers :	7
3.	Stock_prices.csv :	8
IV.	Outils utilisés :	9
1.	Google Colab :	9
2.	Connexion entre Google Colab et Kaggle :	10
V.	JPX Tokyo Stock Exchange Prediction en utilisant la régression linéaire :	13
1.	Le traitement initial des données :	13
2.	Modèle d'entraînement :	19
VI.	Conclusion :	21

Table des figures :

Figure 1 Vue d'ensemble de notre hiérarchie de données.	6
Figure 2 Drapeau des titres sous surveillance et des titres à radier.....	9
Figure 3 Google Colab.	10
Figure 4 Connexion avec Kaggle.	10
Figure 5 Téléchargement des données de la compétition.	11
Figure 6 L'arborescence des données.	12
Figure 7 L'importation des bibliothèques.....	13
Figure 8 L'importation des données depuis stock_prices.csv.	13
Figure 9 Suppression des valeurs dupliquées.....	13
Figure 10 Le résumé statistique de nos données.	14
Figure 11 Valeurs nulles.	14
Figure 12 Les types de nos colonnes.....	14
Figure 13 Suppression de RowId.	14
Figure 14 Traitement sur la colonne SupervisionFlag.	15
Figure 15 Traitement des valeurs nulles.	15
Figure 16 Les données après le remplacement des valeurs nulles.....	15
Figure 17 BoxPlot des données.....	16
Figure 18 Histogramme des données.	16
Figure 19 Les prix des actions selon la date.	16
Figure 20 les prix des actions selon la date après le traitement.	17
Figure 21 Conversion du type de Date.....	17
Figure 22 Heatmap.	18
Figure 23 Suppression des colonnes à cause de la corrélation.	18
Figure 24 La normalisation des données.....	19
Figure 25 La division des données en entraînement et test.....	19
Figure 26 l'entraînement du modèle.....	19
Figure 27 Prédiction de la target.	20
Figure 28 Score du modèle.....	20
Figure 29 Les mesures de performances.	20

I. Introduction :

Selon la capitalisation boursière, le JPX TOKYO est considérée comme la quatrième bourse mondialement. Le covid19 a affecté tous les marchés financiers et afin que le marché financier peut ressentir la fraîcheur, cette compétition était mise à notre disposition afin de prédire les rendements futurs des actions.

Il existe de nombreux efforts de trading quantitatifs utilisés pour analyser les marchés financiers et formuler des stratégies d'investissement. Pour créer et exécuter une telle stratégie, il faut à la fois des données historiques et en temps réel, ce qui est difficile à obtenir, en particulier pour les investisseurs particuliers. Ce concours fournira des données financières pour le marché japonais, permettant aux investisseurs particuliers d'analyser le marché dans toute sa mesure.

Japan Exchange Group, Inc. (JPX) est une société holding exploitant l'une des plus grandes bourses du monde, la Bourse de Tokyo (TSE), et les bourses de dérivés Osaka Exchange (OSE) et Tokyo Commodity Exchange (TOCOM). JPX organise ce concours et est soutenu par la société de technologie AI AlpacaJapan Co.,Ltd.

II. Objectif de la compétition :

Cet ensemble de données contient des données historiques pour une variété d'actions et d'options japonaises. Notre objectif est de prédire les rendements futurs des actions.

III. Vue d'ensemble sur nos données :

Comme les cours historiques des actions ne sont pas confidentiels, il s'agira d'un concours de prévisions utilisant l'API de séries chronologiques. Les données de la période de classement public sont incluses dans l'ensemble de données de la compétition.

Voici une vue d'ensemble de notre hiérarchie de données :

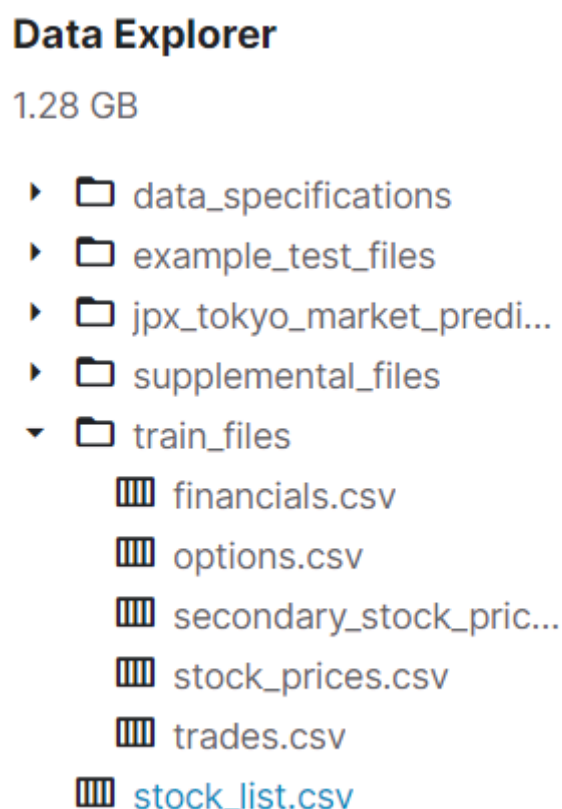


Figure 1 Vue d'ensemble de notre hiérarchie de données.

1. Dossiers :

data_specifications/ - Définitions pour les colonnes individuelles.

jpx_tokyo_market_prediction/ Fichiers qui activent l'API. Attendez-vous à ce que l'API fournisse toutes les lignes en moins de cinq minutes et réserve moins de 0,5 Go de mémoire.

Des copies des fichiers de données existent dans plusieurs dossiers qui couvrent différentes fenêtres temporelles et servent à des fins différentes.

train_files/ Dossier de données couvrant la période de formation principale.

supplemental_files/ Dossier de données contenant une fenêtre dynamique de données d'entraînement supplémentaires. Celui-ci sera mis à jour avec de nouvelles données pendant la phase principale du concours début mai, début juin et environ une semaine avant le verrouillage des soumissions.

example_test_files/ Dossier de données couvrant la période de test public. Destiné à faciliter les tests hors ligne. Inclut les mêmes colonnes fournies par l'API (c'est-à-dire aucune **Targetcolonne**). Vous pouvez calculer la **Targetcolonne** à partir de la **Closecolonne** ; c'est le rendement de l'achat d'une action le lendemain et de la vente le surlendemain. Ce dossier comprend également un exemple d'exemple de fichier de soumission qui sera fourni par l'API.

2. Fichiers :

stock_prices.csv Le fichier principal qui vous intéresse. Comprend le cours de clôture quotidien de chaque action et la colonne cible.

options.csv Données sur le statut d'une variété d'options basées sur le marché plus large. De nombreuses options incluent des prédictions implicites du prix futur du marché boursier et peuvent donc être intéressantes même si les options ne sont pas notées directement.

Secondary_stock_prices.csv L'ensemble de données de base contient les 2 000 actions les plus couramment négociées, mais de nombreux titres moins liquides sont également négociés sur le marché de Tokyo. Ce fichier contient des données pour ces titres, qui ne sont pas notés mais qui peuvent être intéressants pour évaluer le marché dans son ensemble.

trades.csv Résumé agrégé des volumes de transactions de la semaine ouvrée précédente.

financials.csv Résultats des rapports trimestriels sur les résultats.

stock_list.csv - Mappage entre les **SecuritiesCode** noms de société et, ainsi que des informations générales sur le secteur d'activité de la société.

Le fichier qui va nous intéresser le plus c'est le fichier ***stock_prices.csv*** qui est notre fichier principal dans cette compétition.

3. Stock_prices.csv :

Ce fichier est composé de 16 colonnes qui sont les suivant :

RowId, Date, SecuritiesCode, Open, High, Low, Close, Volume, AdjustmentFactor, ExpectedDividend, SupervisionFlag, Target.

RowId: ID unique des enregistrements de prix.

Date: Date de transaction.

SecuritiesCode: Code des valeurs mobilières local.

Open: Le premier prix négocié sur une journée.

High: Le prix négocié le plus élevé sur une journée.

Low: Le prix négocié le plus bas sur une journée.

Close: Le dernier prix négocié un jour.

Volume: Le nombre d'actions échangées sur une journée.

AdjustmentFactor: Un facteur utilisé pour calculer le prix/volume théorique en cas de fractionnement/split inversé (n'incluant PAS le dividende/l'attribution d'actions).

ExpectedDividend: La valeur attendue du dividende à la date ex. Cette valeur est constatée 2 jours ouvrés avant la date de détachement.

SupervisionFlag: Un drapeau des titres sous surveillance et des titres à radier.

Designation Date	Issue Name	Code	Market Segment*	Removal Date	Details	Remarks
May 20, 2022	Mutual Corporation	2773	Standard	-	Designation of Securities Under Supervision (Confirmation)	
May 18, 2022	Arte Salon Holdings, Inc.	2406	Standard	-	Delisting decision & designation of Securities to Be Delisted	
May 16, 2022	KK DI-NIKKO ENGINEERING	6635	Standard	-	Designation of Securities Under Supervision (Confirmation)	
May 16, 2022	KITO CORPORATION	6409	Prime	-	Designation of Securities Under Supervision (Confirmation)	
May 13, 2022	JALUX Inc.	2729	Standard	-	Delisting decision & designation of Securities to Be Delisted	

Figure 2 Drapeau des titres sous surveillance et des titres à radier.

Target: Le taux de variation du cours de clôture ajusté entre t+2 et t+1 où t+0 est la date de transaction.

IV. Outils utilisés :

1. Google Colab :

Google Colab ou Colaboratory est un service cloud, offert par Google (gratuit), basé sur Jupyter Notebook et destiné à la formation et à la recherche dans l'apprentissage automatique. Cette plateforme permet d'entraîner des modèles de Machine Learning directement dans le cloud. Sans donc avoir besoin d'installer quoi que ce soit sur notre ordinateur à l'exception d'un navigateur.

Google colab est un outil complet pour entraîner rapidement et tester rapidement des modèles d'apprentissage automatique sans avoir de contrainte matérielle. Il offre 107 Go de stockage et 12 Go de RAM.



Figure 3 Google Colab.

2. Connexion entre Google Colab et Kaggle :

Cette étape s'effectue en installant d'abord kaggle et en uploadant le fichier kaggle.json généré dans mon profile kaggle, puis on a créé le dossier .kaggle et on copie ce fichier en dedans de lui, afin de lui donner les droits et télécharger nos données en moins de 5 minutes et en allouant 0,5 Go de mémoire.

```
[1] ! pip install -q kaggle

[2] from google.colab import files
files.upload()

Select files: kaggle.json
• kaggle.json(application/json) - 65 bytes, last modified: 26/04/2022 - 100% done
Saving kaggle.json to kaggle.json
{'kaggle.json': b'{"username": "nasmaelja", "key": "e3f28c40ca303bd4fb783abc3f834545"}' }

[3] from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

[4] ! mkdir ~/.kaggle

[5] ! cp kaggle.json ~/.kaggle/

[6] ! chmod 600 ~/.kaggle/kaggle.json

[7] ! kaggle datasets list
```

Figure 4 Connexion avec Kaggle.

```
[8] ! kaggle competitions download -c jpx-tokyo-stock-exchange-prediction

Downloading jpx-tokyo-stock-exchange-prediction.zip to /content
97% 227M/233M [00:06<00:00, 33.5MB/s]
100% 233M/233M [00:06<00:00, 36.8MB/s]

[9] ! unzip jpx-tokyo-stock-exchange-prediction.zip

Archive:  jpx-tokyo-stock-exchange-prediction.zip
  inflating: data_specifications/options_spec.csv
  inflating: data_specifications/stock_fin_spec.csv
  inflating: data_specifications/stock_list_spec.csv
  inflating: data_specifications/stock_price_spec.csv
  inflating: data_specifications/trades_spec.csv
  inflating: example_test_files/financials.csv
  inflating: example_test_files/options.csv
  inflating: example_test_files/sample_submission.csv
  inflating: example_test_files/secondary_stock_prices.csv
  inflating: example_test_files/stock_prices.csv
  inflating: example_test_files/trades.csv
  inflating: jpx_tokyo_market_prediction/__init__.py
  inflating: jpx_tokyo_market_prediction/competition.cpython-37m-x86_64-linux-gnu.so
  inflating: stock_list.csv
  inflating: supplemental_files/financials.csv
  inflating: supplemental_files/options.csv
  inflating: supplemental_files/secondary_stock_prices.csv
  inflating: supplemental_files/stock_prices.csv
  inflating: supplemental_files/trades.csv
  inflating: train_files/financials.csv
  inflating: train_files/options.csv
  inflating: train_files/secondary_stock_prices.csv
  inflating: train_files/stock_prices.csv
```

Figure 5 Téléchargement des données de la compétition.

Voici l'arborescence des données après tous ces bouts de code :

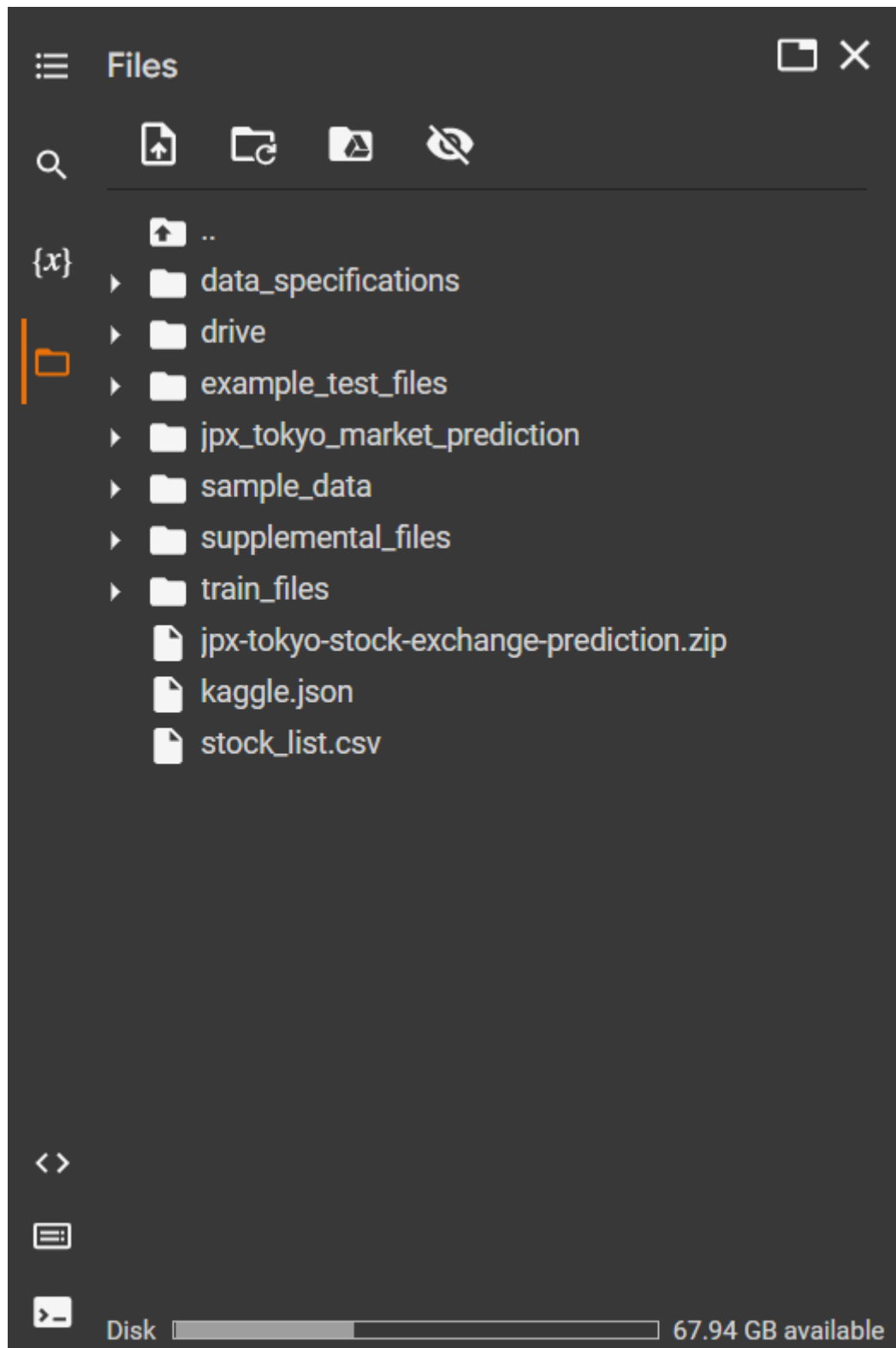


Figure 6 L'arborescence des données.

V. JPX Tokyo Stock Exchange Prediction en utilisant la régression linéaire :

1. Le traitement initial des données :

L'importation des bibliothèques dont on a besoin :

```
[10] import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
```

Figure 7 L'importation des bibliothèques.

Les données sur lesquelles on travaille sont de type timeseries, qui sont mis à jour pendant cette compétition 3 fois, la première c'était au début mai, la deuxième ça sera au début juin et la troisième, ça sera une semaine avant la fin de la compétition.

```
[11] url = '/content/train_files/stock_prices.csv'
url1 = '/content/supplemental_files/stock_prices.csv'
dataset = pd.read_csv(url)
dataset_supp = pd.read_csv(url1)
df = pd.concat([dataset, dataset_supp])
```

Figure 8 L'importation des données depuis stock_prices.csv.

D'abord on supprime les valeurs dupliquées par rapport la Date et le SecuritiesCode :

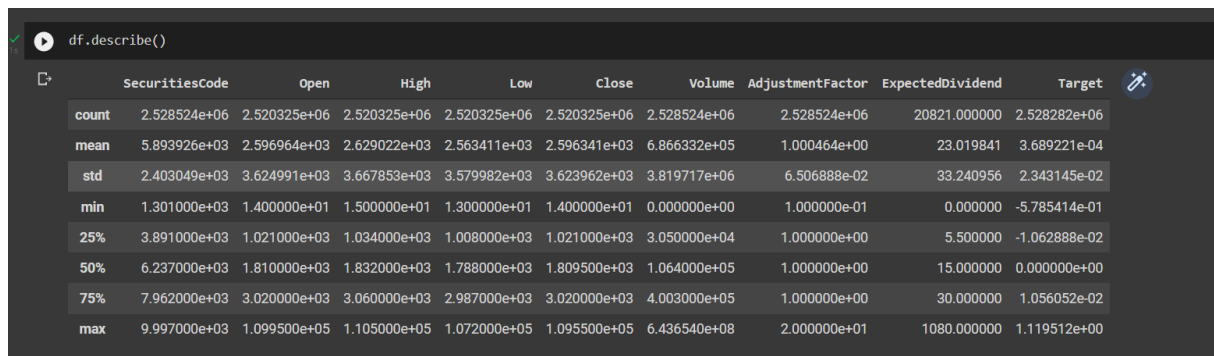
```
[12] df.drop_duplicates(['Date','SecuritiesCode'],keep='first',inplace=True)
[13] df
```

	RowId	Date	SecuritiesCode	Open	High	Low	Close	Volume	AdjustmentFactor	ExpectedDividend	SupervisionFlag	Target
0	20170104_1301	2017-01-04	1301	2734.0	2755.0	2730.0	2742.0	31400	1.0	NaN	False	0.000730
1	20170104_1332	2017-01-04	1332	568.0	576.0	563.0	571.0	2798500	1.0	NaN	False	0.012324
2	20170104_1333	2017-01-04	1333	3150.0	3210.0	3140.0	3210.0	270800	1.0	NaN	False	0.006154
3	20170104_1376	2017-01-04	1376	1510.0	1550.0	1510.0	1550.0	11300	1.0	NaN	False	0.011053
4	20170104_1377	2017-01-04	1377	3270.0	3350.0	3270.0	3330.0	150800	1.0	NaN	False	0.003026
...
195988	20220428_9990	2022-04-28	9990	497.0	504.0	494.0	504.0	51900	1.0	NaN	False	0.024048
195989	20220428_9991	2022-04-28	9991	776.0	786.0	774.0	784.0	23400	1.0	NaN	False	0.011378
195990	20220428_9993	2022-04-28	9993	1520.0	1530.0	1510.0	1517.0	8100	1.0	NaN	False	-0.005941
195991	20220428_9994	2022-04-28	9994	2471.0	2489.0	2457.0	2485.0	7800	1.0	NaN	False	0.005602
195992	20220428_9997	2022-04-28	9997	717.0	731.0	714.0	731.0	149800	1.0	NaN	False	-0.013680

2528524 rows x 12 columns

Figure 9 Suppression des valeurs dupliquées.

On passe pour afficher le résumé statistique de nos données :



```
df.describe()
```

	SecuritiesCode	Open	High	Low	Close	Volume	AdjustmentFactor	ExpectedDividend	Target
count	2.528524e+06	2.520325e+06	2.520325e+06	2.520325e+06	2.520325e+06	2.528524e+06	2.528524e+06	20821.000000	2.528282e+06
mean	5.893926e+03	2.596964e+03	2.629022e+03	2.563411e+03	2.596341e+03	6.866332e+05	1.000464e+00	23.019841	3.689221e-04
std	2.403049e+03	3.624991e+03	3.667853e+03	3.579982e+03	3.623962e+03	3.819717e+06	6.506888e-02	33.240956	2.343145e-02
min	1.301000e+03	1.400000e+01	1.500000e+01	1.300000e+01	1.400000e+01	0.000000e+00	1.000000e-01	0.000000	-5.785414e-01
25%	3.891000e+03	1.021000e+03	1.034000e+03	1.008000e+03	1.021000e+03	3.050000e+04	1.000000e+00	5.500000	-1.062888e-02
50%	6.237000e+03	1.810000e+03	1.832000e+03	1.788000e+03	1.809500e+03	1.064000e+05	1.000000e+00	15.000000	0.000000e+00
75%	7.962000e+03	3.020000e+03	3.060000e+03	2.987000e+03	3.020000e+03	4.003000e+05	1.000000e+00	30.000000	1.056052e-02
max	9.997000e+03	1.099500e+05	1.105000e+05	1.072000e+05	1.095500e+05	6.436540e+08	2.000000e+01	1080.000000	1.119512e+00

Figure 10 Le résumé statistique de nos données.

Passant maintenant pour afficher s'il y a des valeurs nulles dans chaque colonne :



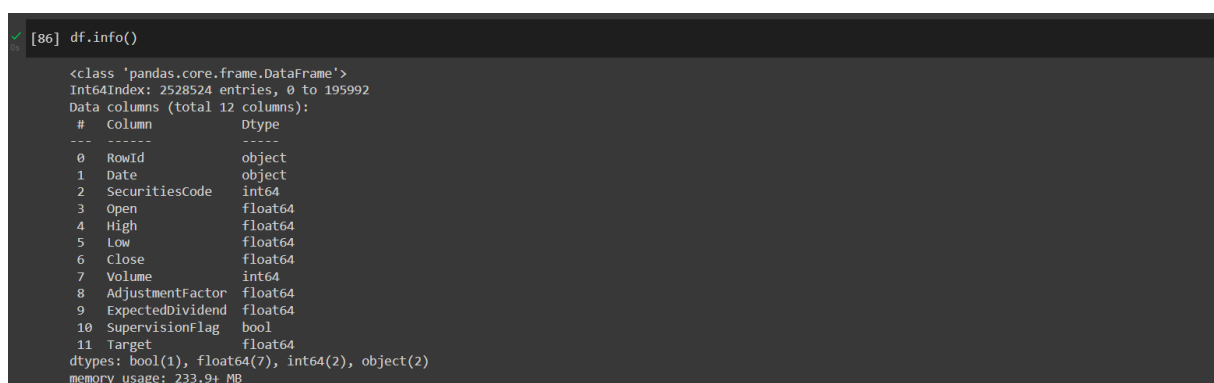
```
[85] df.isnull().sum()
```

RowId	0
Date	0
SecuritiesCode	0
Open	8199
High	8199
Low	8199
Close	8199
Volume	0
AdjustmentFactor	0
ExpectedDividend	2507703
SupervisionFlag	0
Target	242
dtype:	int64

Figure 11 Valeurs nulles.

On constate qu'il y a des valeurs nulles au niveau de Open, High, Low, Close, Target qu'on va les remédier en les remplaçant avec la moyenne de chaque feature d'entre eux. Par contre, la propriété ExpectedDividend, qui signifie le taux de profit, on va remplacer ses valeurs nulles par des zéros.

On va remédier le problème des valeurs nulles toute de suite, mais avant on va afficher les types de nos colonnes :



```
[86] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2528524 entries, 0 to 195992
Data columns (total 12 columns):
#   Column          Dtype
---  -
0   RowId           object
1   Date            object
2   SecuritiesCode  int64
3   Open            float64
4   High            float64
5   Low             float64
6   Close           float64
7   Volume          int64
8   AdjustmentFactor float64
9   ExpectedDividend float64
10  SupervisionFlag  bool
11  Target           float64
dtypes: bool(1), float64(7), int64(2), object(2)
memory usage: 233.9+ MB
```

Figure 12 Les types de nos colonnes.

On va supprimer la colonne RowId qui ne sera pas utile dans notre traitement :



```
[87] df = df.drop(['RowId'], axis = 1)
```

Figure 13 Suppression de RowId.

Comme on a déjà cité lors de l'explication des données, SupervisionFlag représente un drapeau des titres sous surveillance et des titres à radier. On va se débarrasser des données qui ont un SupervisionFlag égal à True (960 enregistrements), et on va supprimer toute la colonne.

```
[88] df = df[df['SupervisionFlag'] == False]
[89] df = df.drop(['SupervisionFlag'], axis = 1)
```

Figure 14 Traitement sur la colonne SupervisionFlag.

C'est le tour des valeurs nulles :

```
[90] df['ExpectedDividend'] = df['ExpectedDividend'].replace(np.nan, 0)
[91] df['Open'] = df['Open'].replace(np.nan, df['Open'].mean())
[92] df['Close'] = df['Close'].replace(np.nan, df['Close'].mean())
[93] df['High'] = df['High'].replace(np.nan, df['High'].mean())
[94] df['Low'] = df['Low'].replace(np.nan, df['Low'].mean())
[95] df['Target'] = df['Target'].replace(np.nan, df['Target'].mean())
```

Figure 15 Traitement des valeurs nulles.

Les données s'affichent comment cela après cette partie :

	Date	SecuritiesCode	Open	High	Low	Close	Volume	AdjustmentFactor	ExpectedDividend	Target
0	2017-01-04	1301	2734.0	2755.0	2730.0	2742.0	31400	1.0	0.0	0.000730
1	2017-01-04	1332	568.0	576.0	563.0	571.0	2798500	1.0	0.0	0.012324
2	2017-01-04	1333	3150.0	3210.0	3140.0	3210.0	270800	1.0	0.0	0.006154
3	2017-01-04	1376	1510.0	1550.0	1510.0	1550.0	11300	1.0	0.0	0.011053
4	2017-01-04	1377	3270.0	3350.0	3270.0	3330.0	150800	1.0	0.0	0.003026
...
195988	2022-04-28	9990	497.0	504.0	494.0	504.0	51900	1.0	0.0	0.024048
195989	2022-04-28	9991	776.0	786.0	774.0	784.0	23400	1.0	0.0	0.011378
195990	2022-04-28	9993	1520.0	1530.0	1510.0	1517.0	8100	1.0	0.0	-0.005941
195991	2022-04-28	9994	2471.0	2489.0	2457.0	2485.0	7800	1.0	0.0	0.005602
195992	2022-04-28	9997	717.0	731.0	714.0	731.0	149800	1.0	0.0	-0.013680

2527551 rows x 10 columns

Figure 16 Les données après le remplacement des valeurs nulles.

On dessine notre boîte à moustache pour afficher nos valeurs aberrantes (outliers) et un histogramme pour chaque colonne pour mieux visualiser la distribution des données :

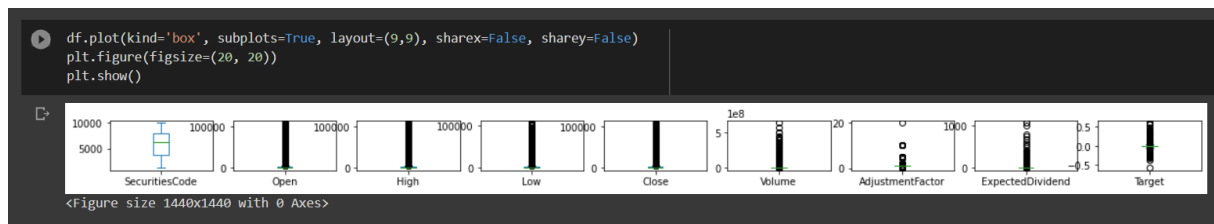


Figure 17 BoxPlot des données.



Figure 18 Histogramme des données.

Il est bien clair que les valeurs aberrantes sont énormes, mais c'est normal dans le cas d'un marché financier. Pour cela, on ne va rien faire à ce niveau.

On visualise maintenant les prix de nos actions selon la date :

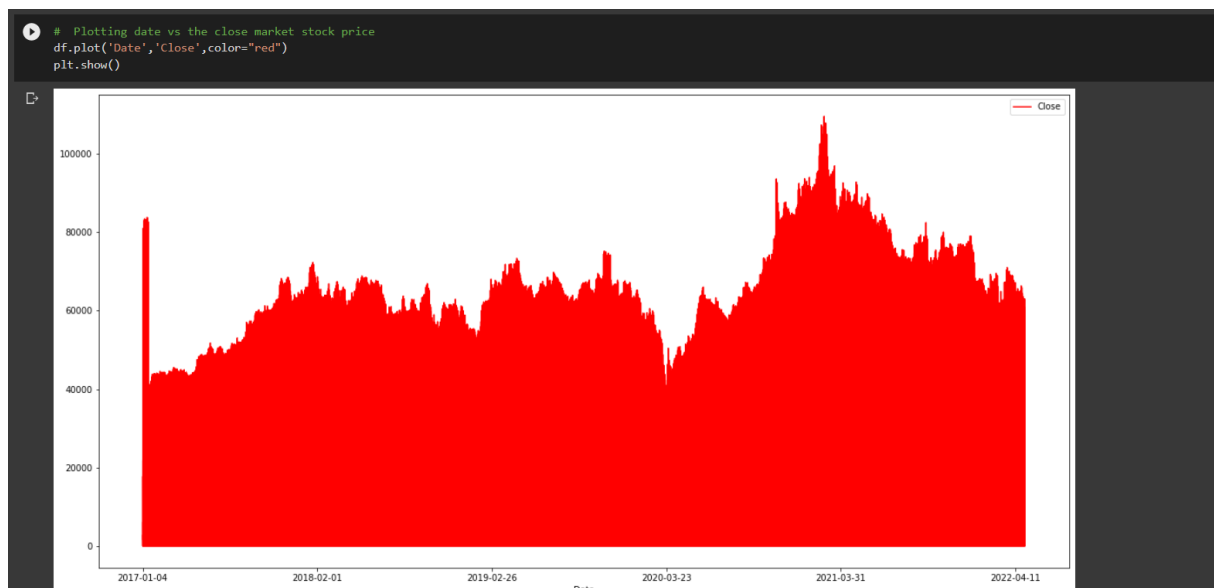


Figure 19 Les prix des actions selon la date.

Comme vous remarquez, pendant la période du confinement au Japon, les actions ont subi une grande diminution dans leurs prix (généralement il y a des secteurs qui ont connu des grandes pertes et d'autres qui ont connu un succès incontournable), dans notre cas on va supprimer les enregistrements de 01-03-2020 jusqu'à 01-09-2020 afin de ne pas affecter les résultats finaux.

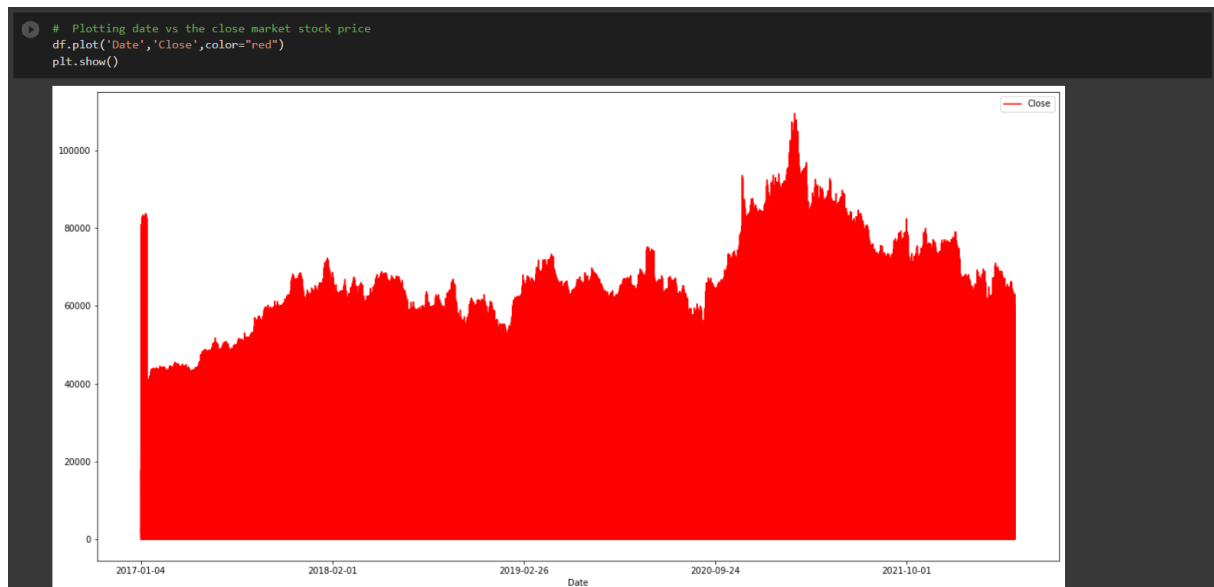


Figure 20 les prix des actions selon la date après le traitement.

Afin d'éviter les problèmes par la suite, on va procéder par convertir notre feature Date en datetime puis en int :

```
[102] df['Date'] = pd.to_datetime(df['Date'])
      df['Date'] = df['Date'].dt.strftime("%Y%m%d").astype(int)
```

Figure 21 Conversion du type de Date.

L'affichage du Heatmap, afin de visualiser les données corrélées et les traiter :



Figure 22 Heatmap.

On constate que les features Open, High, Low et Close sont corrélés entre eux, d'où la suppression des features qui nous donnent les mêmes informations, on va procéder par supprimer les features Open, High et Low et garder juste Close et voir ce que le modèle qu'on va adopter va produire par la suite :

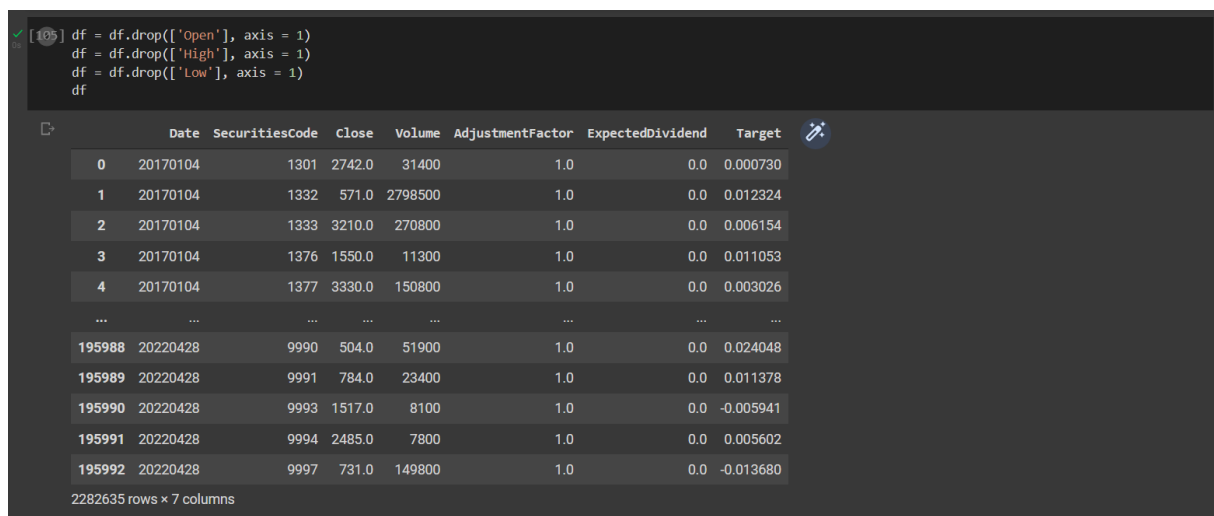


Figure 23 Suppression des colonnes à cause de la corrélation.

La remarque est bien claire que les features Close et Volume ont des valeurs grandes tant que les valeurs de la feature Target par exemple est très petite, pour cela, on va normaliser nos données en utilisant la méthode MinMaxScaler() :

```
[107] from sklearn.preprocessing import MinMaxScaler

[107] scaler = MinMaxScaler()
print(scaler.fit(df))
MinMaxScaler(copy=True, feature_range=(0, 1))
print(scaler.data_max_)
print(scaler.transform(df))

MinMaxScaler()
[[2.02204280e+07  9.99700000e+03  1.09550000e+05  6.43654000e+08
  2.00000000e+01  1.08000000e+03  6.18238022e-01]
 [[0.         0.         0.02490505 ... 0.04522613 0.         0.48402557]
 [0.         0.00356486 0.00508509 ... 0.04522613 0.         0.49371281]
 [0.         0.00367985 0.02917762 ... 0.04522613 0.         0.48855722]
 ...
 [1.         0.99954002 0.01372152 ... 0.04522613 0.         0.4784514 ]
 [1.         0.99965501 0.02255879 ... 0.04522613 0.         0.48809632]
 [1.         1.         0.00654579 ... 0.04522613 0.         0.47198463]]
```

Figure 24 La normalisation des données.

2. Modèle d'entraînement :

On passe maintenant pour diviser notre data en train et test afin d'appliquer notre modèle.

```
[108] from sklearn.model_selection import train_test_split

[109] features = ['SecuritiesCode', 'Close', 'Volume', 'AdjustmentFactor']

[110] X = df[features]
     y = df['Target']

[111] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

Figure 25 La division des données en entraînement et test.

Passant maintenant à l'entraînement de notre modèle en utilisant la régression linéaire :

```
[112] lm = LinearRegression()
     lm.fit(X_train, y_train)

LinearRegression()
```

Figure 26 l'entraînement du modèle.

Prédictions les valeurs de notre target :

```
[113] y_pred = lm.predict(X_test)
      df_ = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})

[114] df_1 = df_.head(25)
      print(df_1)
```

	Actual	Predicted
767188	0.003856	0.000322
860393	0.017790	0.000346
2185396	-0.016626	0.000308
1379402	0.000000	0.000272
691183	-0.033099	0.000165
417760	0.032662	0.000242
309873	-0.004467	0.000310
850015	0.004145	0.000237
741869	-0.037037	0.000363
142676	0.026128	0.000241
126475	-0.004049	0.000266
428470	0.013744	0.000294
419678	0.000000	0.000256
43796	-0.010471	0.000152
1286034	-0.005038	0.000176
1222098	-0.025758	0.000389
134586	0.005299	0.000316
281650	0.026362	0.000295
274545	0.011673	0.000271
79769	0.006728	0.000268
2146802	0.008785	0.000291
1775726	0.020378	0.000258
146996	-0.021053	0.000303
554238	-0.004573	0.000214
2062885	-0.021579	0.000336

Figure 27 Prédiction de la target.

Calculant le score de notre modèle :

```
[115] lm.score(X_test,y_test)

2.7521391326001954e-05
```

Figure 28 Score du modèle.

Utilisant les mesures de performances pour évaluer notre modèle (MAE, MSE, RMSE) :

```
[117] print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
      print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
      print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Mean Absolute Error: 0.014789498980545913
Mean Squared Error: 0.0004904753013745926
Root Mean Squared Error: 0.022146676982667007
```

Figure 29 Les mesures de performances.

VI. Conclusion :

L'utilisation de la régression linéaire pour prédire les cours des actions est une tâche simple en Python lorsque l'on exploite la puissance des bibliothèques d'apprentissage automatique telles que scikit-learn. La commodité de pandas, la bibliothèque ne peut pas non plus être surestimée, ce qui permet d'ajouter l'un des dizaines d'indicateurs techniques dans une seule ligne de code.

Dans ce projet, nous avons vu comment charger des données, tester et diviser les données, ajouter des indicateurs, former un modèle linéaire et enfin appliquer ce modèle pour prédire les cours boursiers futurs de JPX Tokyo, avec un certain succès !