

RAPPORT MA0752



Master Statistique pour l'Événement et la Prévision

Léa Pimpernelle

Introduction	2
CAC40	2
L'oréal	2
Collecte des Données	2
Crawler	2
Launcher	3
Modélisation des Modèles Prédictifs	4
Régression Linéaire	4
Régression Polynomiale	5
Analyse des Résultats	6
Les Données	6
Régression Linéaire	7
Régression Polynomiale	8
Perspective	9
Moyenne Mobile	9
Conclusion	9
Sources	10
Annexes	10

Introduction

Ce projet a pour objectif de programmer un crawler sous Python afin de récupérer les cotations du CAC40 sur le site de Boursorama. Les données récupérées permettront de suivre l'évolution du cours boursier d'une entreprise parmi celles du CAC40. Puis, les données collectées seront analysées afin de prédire les valeurs futures du cours boursier de la société choisie avec différentes méthodes qui seront détaillées par la suite.

CAC40

Le CAC40 a été créé en 1987, c'est le principal indice de la Bourse de Paris. Il est composé de 40 valeurs de sociétés françaises représentant l'ensemble des secteurs d'activité dont les volumes d'échanges de titres sont les plus importants.

- CAC signifie Cotation Assisté en Continu. Sa valeur varie en permanence tous les jours ouvrés de 9h00 à 17h30.
- 40, c'est le nombre de valeurs parmi les 100 premières capitalisations françaises.

L'oréal

L'Oréal est présent depuis la création du CAC40. C'est le premier groupe cosmétique mondial. Le chiffre d'affaires du groupe se compose comme suit :

- 40,2% des produits de soins de la peau
- 20,5% des produits de maquillage
- 15,1% des produits de soins capillaires
- 10,9% des parfums
- 9,3% des produits de coloration
- 4% autres

Le groupe se compose de Garnier, L'Oréal Paris, Cacharel, Yves Saint Laurent, La Roche-Posay, Diesel et bien d'autres. La commercialisation des produits se fait au travers de la grande distribution, de la vente à distance, des salons de coiffure, des pharmacies,... En 2021, L'Oréal dispose de 39 sites de production dans le monde. Ainsi le chiffre d'affaires dans l'entreprise se répartit comme suit : Europe (31,5%), Asie du Nord (30,5%), Amérique du Nord (25,3%), Asie-Pacifique-Moyen Orient-Afrique (7,2%) et Amérique latine (5,5%).

Collecte des Données

Crawler

Le `crawler_lea.py` est fait dans une class afin de l'utiliser dans un launcher expliqué après. Le Crawler utilise cinq packages :

- `datetime` : manipuler les dates et les heures
- `time` : utiliser des fonctions liées au temps
- `pandas` : manipuler et analyser des données
- `requests` : faire des requêtes à une page web
- `BeautifulSoup` : extraire des données

Le crawler commence par récupérer les données entrées par l'utilisateur ayant lancé le launcher (date de début, date de fin, temps de pause, les deux urls et le nom de fichier de sortie). Avant que les crawls commencent, la date de début et la date de fin sont contrôlées par la fonction `check_time`. Une fois les deux dates vérifiées, la fonction `crawl` peut commencer. Dans un premier temps, la fonction récupère tous les libellés du CAC40 sur Boursorama grâce à l'url. Dans un second temps, elle récupère les cotations toujours sur le même site. Les cotations sont récupérées à un intervalle de temps rentré par l'utilisateur et concaténées à la suite des

libellés. Une fois la date de fin arrivée, le crawl s'arrête, un fichier .csv est enregistré dans l'ordinateur de l'utilisateur avec toutes les données recueillies.

EXEMPLE DE RENDU :

```
In [3]: data = pd.read_csv("un.csv", sep=';', index_col = "Libellé") # importation du fichier
...: print(data)
    10:00   10:01   10:02   10:03   10:04
Libellé
AIR LIQUIDE      132.920  132.800  132.860  132.820  132.800
AIRBUS          113.220  113.240  113.240  113.360  113.300
ALSTOM           22.180   22.180   22.180   22.200   22.200
AXA              26.450   26.430   26.435   26.450   26.445
BNP PARIBAS     50.500   50.470   50.470   50.490   50.510
BOUYGUES        29.760   29.740   29.750   29.740   29.740
CAPGEMINI       161.400  161.350  161.350  161.100  161.000
CARREFOUR       16.290   16.305   16.305   16.315   16.305
CREDIT AGRICOLE SA  9.707   9.700   9.701   9.704   9.710
DANONE           49.135   49.150   49.145   49.165   49.160
DASSAULT SYSTEMES 33.450   33.425   33.430   33.375   33.320
ENGIE            13.546   13.544   13.556   13.562   13.548
ESSILORLUXOTTICA 164.400  164.350  164.350  164.350  164.250
EUROFINS SCIENTIFIC 66.840   66.860   66.860   66.800   66.780
HERMES INTL     1335.500 1336.000 1336.000 1335.000 1335.000
KERING            503.100  503.100  503.000  502.800  503.000
L'OREAL           322.950  323.000  322.950  322.750  322.850
LEGRAND          73.360   73.320   73.340   73.300   73.280
LVMH             648.700  648.800  648.800  648.700  648.700
MICHELIN         26.180   26.170   26.150   26.145   26.140
ORANGE           9.804    9.805    9.804    9.809    9.804
PERNOD RICARD   179.850  179.900  180.000  179.950  180.050
PUBLICIS GROUPE 59.380   59.340   59.320   59.320   59.340
RENAULT          30.425   30.440   30.405   30.445   30.460
SAFRAN           113.040  113.140  113.180  113.180  113.160
ARCELORMITTAL   24.215   24.230   24.235   24.190   24.225
UNIBATL-RODAMCO-WESTFIELD 48.605  48.565  48.555  48.540  48.585
SAINT-GOBAIN     41.760   41.740   41.750   41.750   41.780
SANOFI            86.390  86.340  86.340  86.380  86.320
SCHNEIDER ELECTRIC 131.940 131.880 131.860 131.860 131.840
SOCIETE GENERALE 23.755   23.750   23.780   23.800   23.805
STELLANTIS        13.600   13.598   13.584   13.574   13.572
STMICROELECTRONICS 32.500  32.510  32.515  32.475  32.475
TELEPERFORMANCE  255.700  255.600  255.800  255.600  255.300
THALES            124.650  124.950  124.900  125.050  125.000
TOTALENERGIES    57.600   57.580   57.590   57.570   57.510
VEOLIA            23.560   23.550   23.560   23.570   23.570
VINCI             93.270  93.270  93.320  93.330  93.230
VIVENDI          8.442    8.442    8.442    8.442    8.442
WORLDLINE        44.070  44.070  44.080  44.030  44.030
```

Le crawler_lea.py est commenté , il se trouve en annexe et dans le dossier rendu.

Launcher

Le launcher.py sert à récupérer les données utiles à l'exécution du crawler. C'est ce programme qu'il faut lancer. Il permet de demander à l'utilisateur qui le lance la date de début, la date de fin, le temps de pause entre chaque crawl et le nom du fichier csv de sortie. Pour ce programme, il y a besoin d'un package : datetime pour manipuler les dates et les heures et il faut également importer le crawler_lea.

EXEMPLE DE RENDU :

```
-- Réglage de la récupération de données :
-- Donne moi les infos concernant la date et heure de DEBUT de ton crawl (format : JJ,MM,AAAA,HH,MM) : 16,11,2022,15,00
=====
Début du crawler le : 16/11/2022 à 15:00

Donne moi les infos concernant la date et heure de FIN de ton crawl (format : JJ,MM,AAAA,HH,MM) : 16,11,2022,17,00
=====
Fin du crawler le : 16/11/2022 à 17:00

Donne moi le temps entre chaque crawl (en minutes) : 2
L'intervalle entre chaque crawl est de 2 minutes.

Le nom du fichier sera : cac40.csv

-- Voici un récapitulatif de ce que tu as demandé :
Début : 16/11/2022 à 15:00
Fin : 16/11/2022 à 17:00
Intervalle : 2 minute(s) entre chaque crawl
URLs : ['https://www.boursorama.com/bourse/actions/cotations/?quotation_az_filter%5Bmarket%5D=1rPCAC', 'https://www.boursorama.com/bourse/actions/cotations/page-2?quotation_az_filter%5Bmarket%5D=1rPCAC']
Nom du fichier : cac40.csv
```

Le launcher_lea.py est commenté, il se trouve en annexe et dans le dossier rendu.

Modélisation des Modèles Prédictifs

Début des programmes : Après avoir importé le fichier .csv à étudier, il faut récupérer l'action choisie (L'Oréal). Les cotations sont mises dans une liste (Y). Il faut convertir les cotations de la liste qui sont en chaîne de caractères (str) en nombre flottant (float). Une deuxième liste est créée (X) de 1 jusqu'à la taille de Y afin de s'en servir en ordonnée.

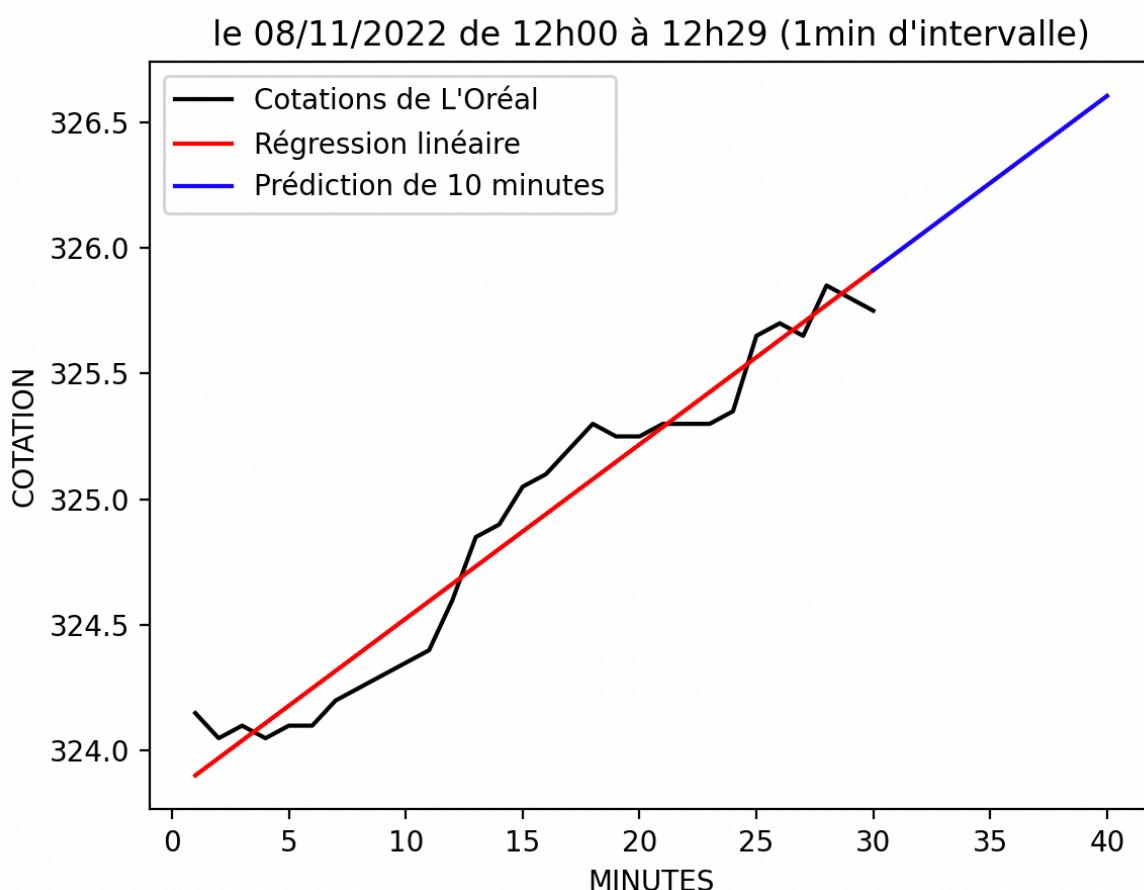
Régression Linéaire

Pour la régression linéaire, le programme utilise le module `from scipy import stat` afin de récupérer la pente de la régression linéaire, l'ordonnée à l'origine de la régression linéaire, le coefficient de corrélation de Pearson, la p-value et l'erreur standard de la pente. Enfin, dans une fonction `predict(x)`, on calcule la valeur de la pente fois la liste x plus l'ordonnée à l'origine ($y = ax + b$ avec a le coefficient et b l'ordonnée à l'origine). Pour prédire la suite de la droite de régression linéaire, on réutilise la fonction `predict(x)` avec une liste de valeurs commençant par le dernier chiffre de la liste x.

Dans ce programme, il y a trois packages utilisés en plus du module `scipy` :

- `pandas` : manipuler et analyser des données
- `numpy` : effectuer des calculs numériques et gestion des tableaux de nombres
- `matplotlib.pyplot` : création de graphique

EXEMPLE DE RENDU :



Le programme `regression_lineaire.py` est commenté, il se trouve en [annexe](#) et dans le dossier rendu.

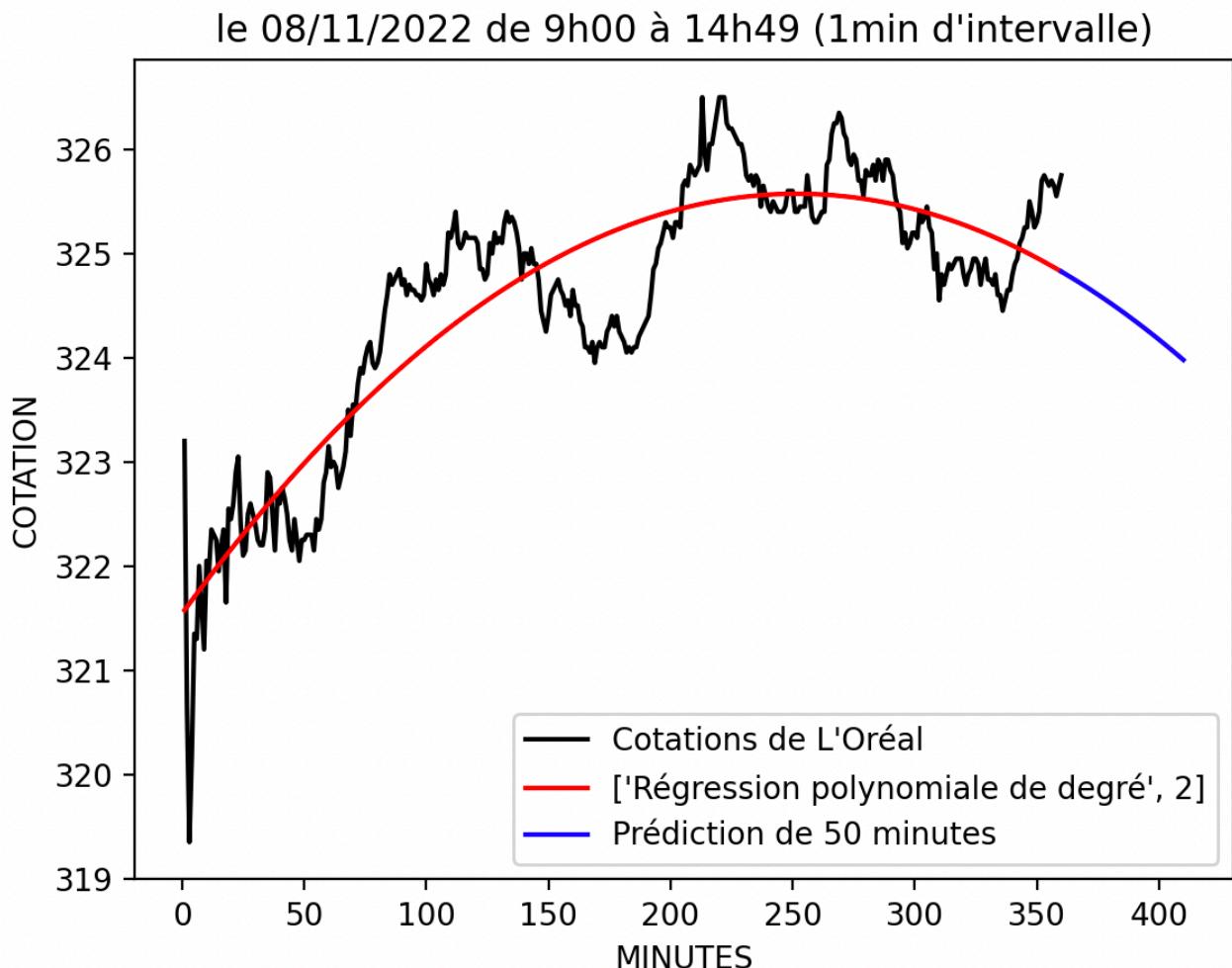
Régression Polynomiale

Pour la régression polynomiale, le programme utilise les modules `from sklearn.linear_model import LinearRegression` et `from sklearn.preprocessing import PolynomialFeatures` afin de faire une régression linéaire et une régression polynomiale de degré choisi. Par exemple, si on choisit de faire une régression polynomiale de degré n , on obtiendrait : $y = d_0 + d_1x + d_2x^2 + \dots + d_nx^n$ avec y la variable de réponse que nous voulons prédire, x les variables, d_0 l'ordonnée à l'origine de y et les autres d_1, \dots, n les coefficients. Pour prédire la suite de la régression polynomiale, on réutilise les coefficients et l'ordonnée à l'origine avec une liste de valeurs commençant par le dernier chiffre de la liste x dans une boucle `for` pour calculer chaque valeur de la liste x ($y = d_0 + d_1x + d_2x^2 + \dots + d_nx^n$).

Dans ce programme, il y a trois packages utilisés en plus des modules `sklearn.preprocessing` et `sklearn.linear_model` :

- `pandas` : manipuler et analyser des données
- `numpy` : effectuer des calculs numériques et gestion des tableaux de nombres
- `matplotlib.pyplot` : création de graphique

EXEMPLE DE RENDU :



Le programme `regression_polynomiale.py` est commenté, il se trouve en annexe et dans le dossier rendu.

Analyse des Résultats

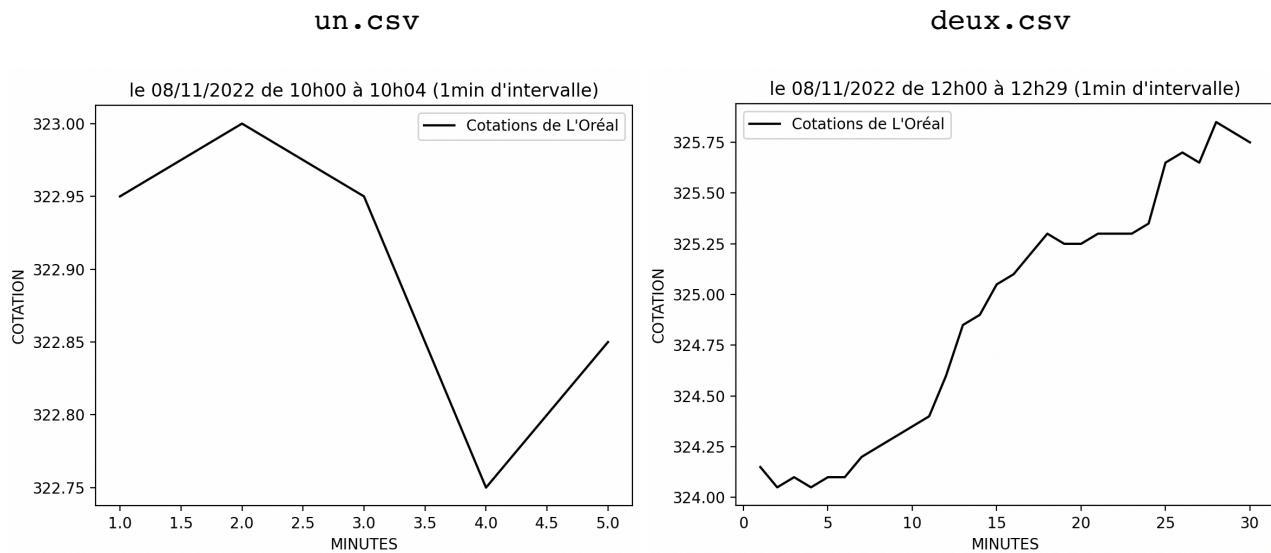
Les Données

Les cinq fichiers date du 08/11/2022 :

Nom du fichier	Heure de début	Heure de fin	Intervalle	Nombre de cotations
un.csv	10h00	10h04	1 min	5
deux.csv	12h00	12h29	1 min	30
trois.csv	15h00	16h58	2 min	60
quatre.csv	9h00	14h59	1 min	360
cinq.csv	9h00	17h28	2 min	255

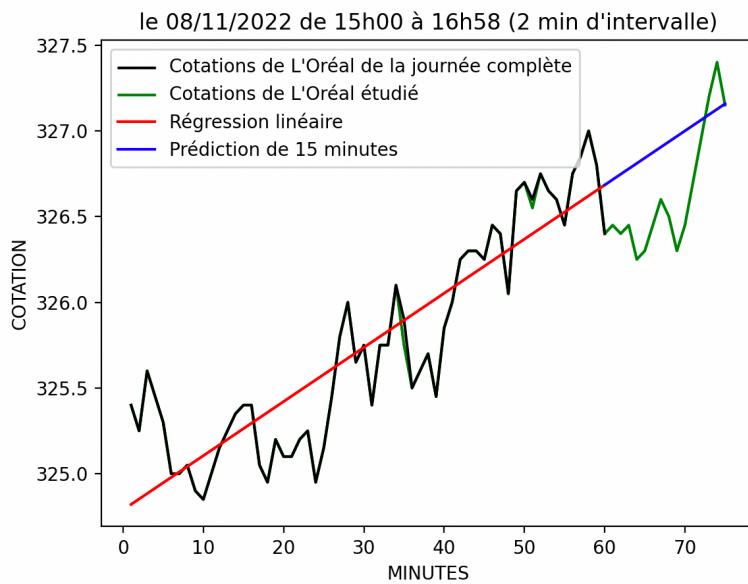
Observations des fichiers de données :

1. L'intervalle n'a jamais dépassé les 2 minutes pour garder assez de précision sur les graphiques.
2. Les fichiers un.csv et deux.csv n'ont pas assez d'historique de baisse et de hausse pour les modèles prédictifs compte tenu du nombre de cotations recueillies.
3. Le fichier cinq.csv pourra servir de comparaison avec les prédictions sur les autres fichiers.

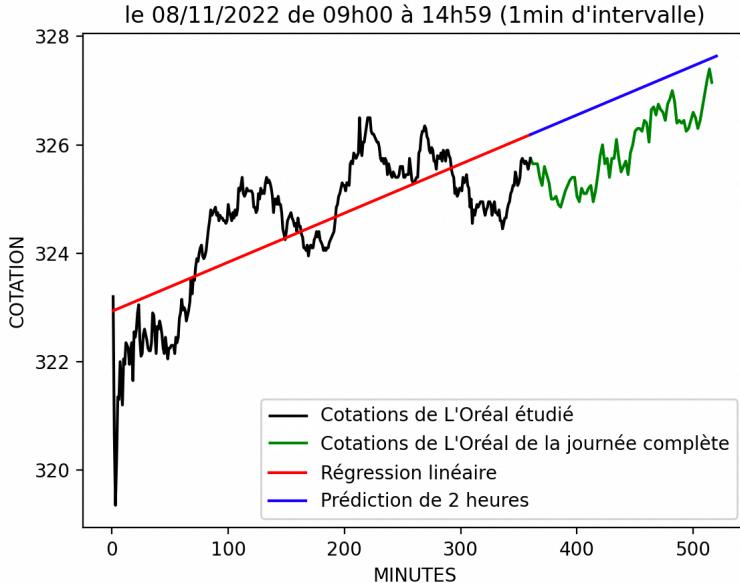


Régression Linéaire

Appliquons la régression linéaire ($y = ax + b$) sur les fichiers `trois.csv` et `quatre.csv` et comparons avec le fichier de la journée complète `cinq.csv`.



`trois.csv` : La courbe de prédition est strictement croissante alors on en conclut qu'il ne faudrait jamais prédire de vendre l'action. Cependant, on peut voir que la courbe verte continue de fluctuer.

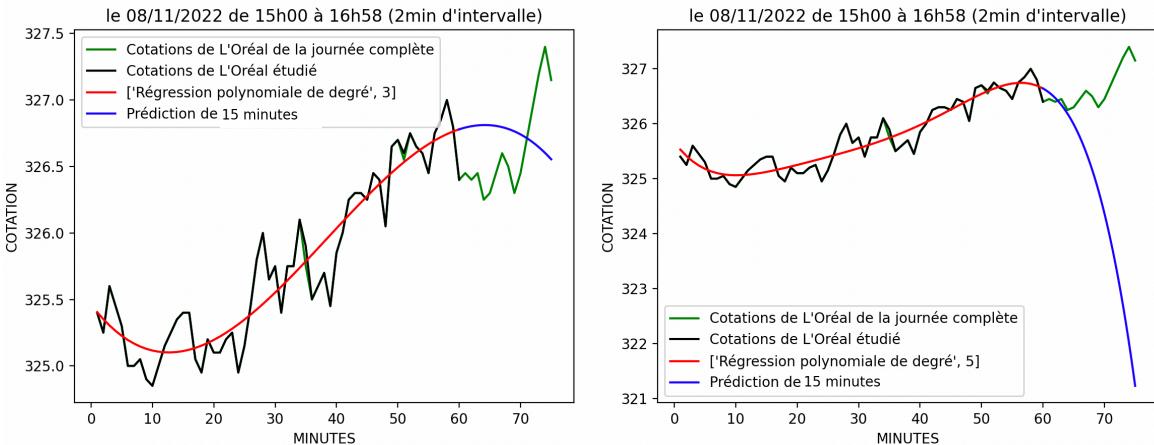


`quatre.csv` : La courbe de prédition est également strictement croissante; on en conclut la même chose : il ne faudrait jamais prédire de vendre l'action. Cependant, on peut voir que la courbe verte continue de fluctuer.

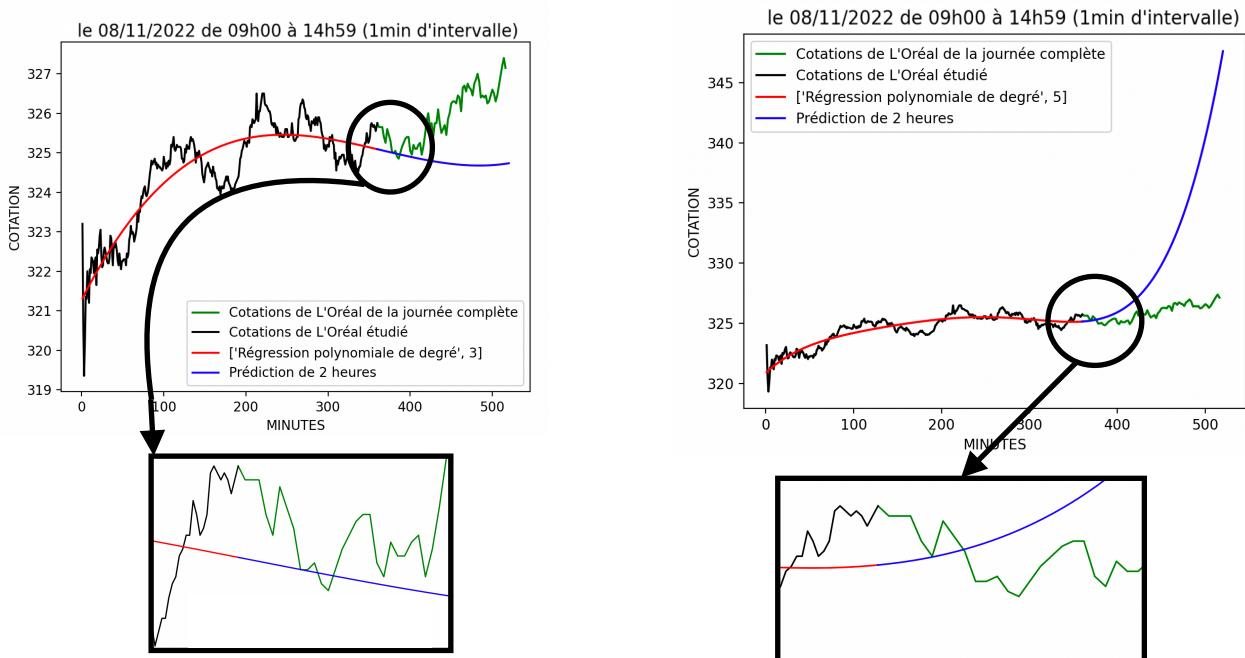
Ainsi, la régression linéaire n'est pas un bon modèle prédictif. En effet, si le coefficient est positif comme ce qui est le cas dans ces deux graphiques, la droite de régression linéaire sera en croissance constante.

Régression Polynomiale

Appliquons la régression polynomiale de degré 3 ($y = d_0 + d_1x + d_2x^2 + d_3x^3$) puis de degré 5 ($y = d_0 + d_1x + d_2x^2 + d_3x^3 + d_4x^4 + d_5x^5$) sur les fichiers `trois.csv` et `quatre.csv` et comparons avec le fichier de la journée complète `cinq.csv`.



`trois.csv` : La courbe de prédiction pour le degré 3 et 5 est meilleure que la droite de régression linéaire. On peut mieux prédire quand planifier l'achat et la vente d'actions mais n'est pas encore très fiable. Cependant pour le degré 5, il faut regarder à quelques minutes du début de la prédiction. Je remarque qu'il n'y a peut-être pas assez de hausse et de baisse pour ce jeu de données et l'utilisation de cette méthode.



`quatre.csv` : La courbe de prédiction pour le degré 3 et 5 est également meilleure que la droite de régression linéaire. On peut mieux prédire quand planifier l'achat et la vente d'actions mais n'est pas encore très fiable. Cependant, une prédiction de 2 heures était trop ambitieuse, quelques minutes du début de la prédiction seraient suffisantes.

Ainsi, la régression polynomiale est mieux que la régression linéaire mais ne reste pas un bon modèle. En effet, plus le degré augmente et plus la courbe de prédiction converge vite, comme on peut le voir sur le degré 5.

Perspective

Moyenne Mobile

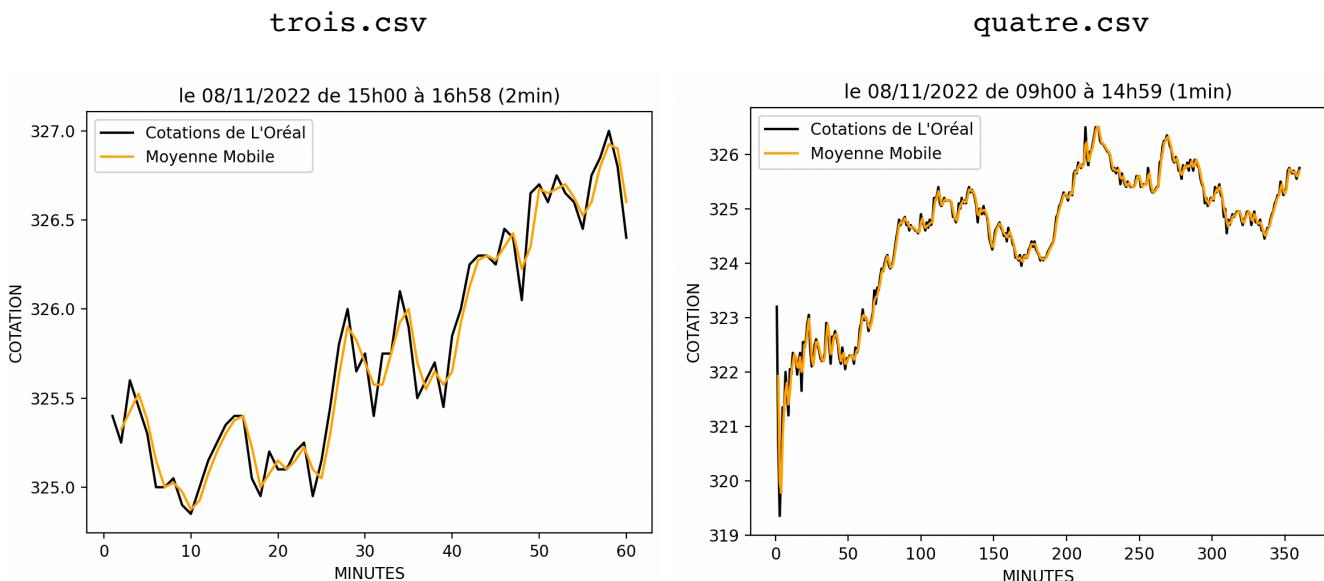
Cette moyenne est dite mobile car elle est calculé en continue. Dans ce programme, il y a trois packages utilisés :

- pandas : manipuler et analyser des données
- numpy : effectuer des calculs numériques et gestion des tableaux de nombres
- matplotlib.pyplot : création de graphique

Ainsi, dans ce programme les points sont calculés comme suit :

$$\frac{x_1 + x_2}{2}; \frac{x_2 + x_3}{2}; \dots; \frac{x_{n-1} + x_n}{2}; \frac{x_n + x_{n+1}}{2}$$

EXEMPLE DE RENDU :



La moyenne mobile serait un bon modèle de prédiction. Or, je n'ai pas réussi à prolonger la courbe. Cependant, si la courbe est actualisée en temps réel, on pourrait très bien prédire quand planifier la vente ou l'achat d'actions. Par exemple, si on regarde le graphique du fichier `trois.csv`, lorsque la courbe orange croise la courbe noire, ça serait le signe d'une vente ou d'un achat. Cependant, l'achat ou la vente devrait se faire dans la seconde pour ne pas être perdant.

Le programme `moyenne_mobile.py` est commenté, il se trouve en [annexe](#) et dans le dossier `rendu`.

Conclusion

Ce projet a permis de comprendre et de réaliser un crawler afin de récupérer les données du CAC40 sur le site de Boursorama. Il a également permis de programmer des modèles prédictifs (régression linéaire, régression polynomiale et moyenne mobile) et de juger si les modèles sont capables de planifier ou non l'achat / la vente d'actions. La planification de l'achat / vente par des modèles de prédiction ne sera jamais fiable, inclure l'actualité de l'entreprise L'Oréal et le contexte géopolitique seraient un plus pour l'analyse.

Sources

CAC40 : <https://www.economie.gouv.fr/facileco/cac-40>

L'Oréal : [https://fr.wikipedia.org/wiki/L%27Oréal](https://fr.wikipedia.org/wiki/L%27Or%C3%A9al)

L'Oréal : <https://www.boursorama.com/cours/1rPOR/>

Régression Linéaire : <https://mrmint.fr/regression-lineaire-python-pratique>

Régression Linéaire : <https://www.stat4decision.com/fr/faire-regression-lineaire-r-python/>

Régression polynomiale : <https://mrmint.fr/regression-polynomiale>

Régression polynomiale : <https://data36.com/polynomial-regression-python-scikit-learn/>

Moyenne Mobile : <https://www.broker-forex.fr/moyenne-mobile-macd.php>

Annexes

CRAWLER

```

1  # -*- coding: utf-8 -*-
2  """
3  @author: Lea Pimpernelle
4  """
5
6  import datetime # manipuler les dates et les heures
7  import time # fonctions liées au temps
8  import pandas as pd # manipuler et l'analyser des données
9  import requests # requête à une page web
10 from bs4 import BeautifulSoup # extraire des données
11
12 # Création d'une class
13 class Crawler():
14     def __init__(self, start_date, end_date, pace, urls, file_name):
15         # Données
16         self.start_date = start_date
17         self.end_date = end_date
18         self.pace = pace
19         self.urls = urls
20         self.output_file = file_name
21         self.dataframe = pd.DataFrame()
22         self.check_time()
23
24     def check_time(self):
25         # fixer le timer d'attente à un nombre rond
26         print("En attente de l'heure de début...")
27         x = datetime.datetime.now()
28         if x.second != 0:
29             time.sleep(60 - x.second)
30
31         while datetime.datetime.now() < self.start_date:
32             time.sleep(60)
33
34         print("Début du crawl")
35         self.crawl()
36
37     def crawl(self):
38         # Crawl les 2 urls
39         # Les libellés
40         soups = self.get_soups()
41         libs = []
42
43         for soup in soups:
44             lib = soup.find_all("div", class_="o-pack__item u-ellipsis u-color-cerulean") # Données récupérées (libellés)
45             for i in range(len(lib)):
46                 lib[i] = lib[i].get_text(strip=True)
47
48             libs += lib
49
50         self.dataframe["Libellé"] = libs # colonne des libellés
51
52         # La data
53         current_time = datetime.datetime.now()
54         while current_time < self.end_date:
55             soups = self.get_soups()
56             act = []
57             for soup in soups:
58                 act_soup = soup.find_all("td", class_="c-table__cell c-table__cell--dotted u-text-right u-text-medium") # Données récupérées (cotations)
59                 act_soup = [act_soup[i] for i in range(len(act_soup)) if i % 5 == 0]
60                 for i in range(len(act_soup)):
61                     act_soup[i] = act_soup[i].get_text(strip=False).replace('(c)', '').replace(' ', '')
62
63             act += act_soup
64
65             temp_pd_df = pd.DataFrame()
66             temp_pd_df[current_time.strftime("%H:%M")] = act # Nom des colonnes
67             self.dataframe = pd.concat([self.dataframe, temp_pd_df], axis=1) # concaténation des colonnes
68
69             time.sleep(self.pace * 60 - datetime.datetime.now().second) # temps de pause
70             print(f"Il reste {self.end_date - current_time}.seconds // 60 minutes") # information du temps restant
71             current_time = datetime.datetime.now()
72
73         self.dataframe.to_csv(self.output_file, sep=';', index=False) # ajout au fichier csv
74
75     # Requête des 2 pages web
76     def get_soups(self):
77         soups = []
78         for url in self.urls:
79             page = requests.get(url)
80             soup = BeautifulSoup(page.text, 'html.parser')
81             soups.append(soup)
82
83         return soups

```

LAUNCHER

```

1  # -*- coding: utf-8 -*-
2  """
3  @author: Lea Pimpernelle
4  """
5
6  import datetime # manipuler les dates et les heures
7  from crawler_lea import Crawler # crawler_lea.py
8  import time # fonctions liées au temps
9
10 # URLs utilisées
11 urls = ['https://www.boursorama.com/bourse/actions/cotations/?quotation_az_filter%5Bmarket%5D=lrPCAC', 'https://www.boursorama.com/bourse/actions/cotations/page-2?quotation_az_filter%5Bmarket%5D=lrPCAC']
12
13 # -----
14 print("-----")
15 print("Réglage de la récupération de données : ")
16 print("-----")
17 # -----
18
19 # Date du début du crawler
20 day, month, year, hour, minute = input("Donne moi les infos concernant la date et heure de DEBUT de ton crawl (format : JJ,MM,AAAA,HH,MM) : ".split(','))
21 # Vérification du format input
22 try:
23     start_date = datetime.datetime(int(year), int(month), int(day), int(hour), int(minute))
24 except ValueError:
25     print("Erreur, le programme va s'arrêter dans 10 secondes")
26     time.sleep(10)
27     exit()
28
29 # -----
30 print("-----")
31 print("Début du crawler le : ", start_date.strftime("%d/%m/%Y à %H:%M"), "\n")
32 # -----
33
34 # Date de la fin du crawler
35 day, month, year, hour, minute = input("Donne moi les infos concernant la date et heure de FIN de ton crawl (format : JJ,MM,AAAA,HH,MM) : ".split(','))
36 # Vérification du format input
37 try:
38     end_date = datetime.datetime(int(year), int(month), int(day), int(hour), int(minute))
39 except ValueError:
40     print("Erreur, le programme va s'arrêter dans 5 secondes")
41     time.sleep(5)
42     exit()
43
44 # -----
45 print("-----")
46 print("Fin du crawler le : ", end_date.strftime("%d/%m/%Y à %H:%M"), "\n")
47 # -----
48
49 # Intervalle de temps entre chaque crawl
50 pace = int(input("Donne moi le temps entre chaque crawl (en minutes) : "))
51
52 # -----
53 print("L'intervalle entre chaque crawl est de ", pace, "minutes.")
54 file_name = 'cac40'
55 print("\nLe nom du fichier sera : cac40.csv\n")
56 # -----
57
58 # Récapitulatif
59 print("—Voici un récapitulatif de ce que tu as demandé :")
60 print("Début : ", start_date.strftime("%d/%m/%Y à %H:%M"))
61 print("Fin : ", end_date.strftime("%d/%m/%Y à %H:%M"))
62 print("Interval : ", pace, "minute(s) entre chaque crawl")
63 print("URLs : ", urls)
64 print("Nom du fichier : cac40.csv\n—")
65
66 # launch du crawler
67 crawler = Crawler(start_date, end_date, pace, urls, file_name)

```

RÉGRESSION LINÉAIRE

```

1  # -*- coding: utf-8 -*-
2  """
3  @author: Lea Pimpernelle
4  """
5
6  import pandas as pd # manipuler et l'analyser des données
7  import numpy as np # effectuer des calculs numériques et gestion des tableaux de nombres
8  import matplotlib.pyplot as plt # création de graphiques
9  from scipy import stats # Réaliser une régression linéaire (stats.linregress)
10
11 """importation du fichier csv"""
12 data = pd.read_csv("deux.csv", sep=';', index_col = "Libellé") # importation du fichier
13 #print(data)
14
15 """Listes"""
16 lib = "L'OREAL" # libellé étudié
17 Y=data.loc[lib] # Création d'une liste des valeurs du libellé choisis
18 Y=[float(x) for x in Y] # str -> float
19 X = np.arange(1, len(Y)+1, 1, dtype=int) # liste avec nombre de valeurs prises
20
21 """Régression linéaire"""
22 slope, intercept, r_value, p_value, std_err = stats.linregress(X,Y)
23
24 def predict(x):
25     return slope * x + intercept # fonction : ax+b
26 Y_reg = predict(X)
27
28 """GRAPHIQUE"""
29 plt.plot(X,Y,c='k',label="Cotations de L'Oréal") # courbe du libellé étudié
30 #plt.scatter(X,Y,c='k',marker='+',label='cotation') # points des cotations
31 plt.plot(X, Y_reg, c='r',label='Régression linéaire') # régression linéaire
32
33 """Prédiction"""
34 XP = np.linspace(len(X),len(X)+10,100)
35 YP = predict(XP)
36 plt.plot(XP,YP , c='b',label='Prédiction de 10 minutes') # prédition de la régression linéaire
37
38 """Habillage du graphique"""
39 plt.title("le 08/11/2022 de 12h00 à 12h29 (1min d'intervalle)")
40 plt.xlabel("MINUTES")
41 plt.ylabel("COTATION")
42 plt.legend()
43 plt.show()

```

RÉGRESSION POLYNOMIALE

```

1  # -*- coding: utf-8 -*-
2  """
3  @author: Lea Pimpernelle
4  """
5
6  import pandas as pd # manipuler et l'analyser des données
7  import numpy as np # effectuer des calculs numériques et gestion des tableaux de nombres
8  import matplotlib.pyplot as plt # création de graphiques
9  from sklearn.preprocessing import PolynomialFeatures # Réaliser une régression polynomiale
10 from sklearn.linear_model import LinearRegression # Réaliser une régression linéaire
11
12 """importation du fichier csv"""
13 data = pd.read_csv("quatre.csv",sep=';', index_col = "Libellé") # imparatition du fichier
14 #print(data)
15
16 """Listes"""
17 lib = "L'ORÉAL" # libellé étudié
18 Y=data.loc[lib] # Création d'une liste des valeurs du libellé choisis
19 Y=[float(x) for x in Y] # str -> float
20 X = np.arange(1, len(Y)+1, 1, dtype=int) # liste avec nombre de valeurs prises
21
22 """Régression polynomiale"""
23 d=2 # degré
24 poly = PolynomialFeatures(degree=d, include_bias=False)
25 poly_features = poly.fit_transform(X.reshape(-1, 1)) # transforme notre tableau numpy X d'un tableau 1D en un tableau 2D
26 poly_reg_model = LinearRegression()
27 poly_reg_model.fit(poly_features, Y) # coefficients + intercept
28 y_reg = poly_reg_model.predict(poly_features) # résultats régression polynomiale
29
30
31 """GRAPHIQUE"""
32 plt.plot(X,Y,c='k',label="Cotations de L'Oréal") # courbe du libellé étudié
33 #plt.scatter(X,Y,c='k',marker='+',label='cotation') # points des cotations
34
35 plt.plot(X,y_reg,'r',label=["Régression polynomiale de degré",d]) # régression polynomiale
36
37 """Prediction"""
38 XP = np.linspace(len(X),len(X)+50,100) #ajout de 50min de prédition
39 YP = np.linspace(1,1,100)
40
41 # calcul des prédition avec les coefficients de poly_reg_model
42 for i in range (len(XP)):
43     YP[i] = (XP[i]**d*poly_reg_model.coef_[1])+(XP[i]*poly_reg_model.coef_[0])+poly_reg_model.intercept_
44
45 plt.plot(XP,YP,'b',label='Prédiction de 50 minutes') # prédition des futures cotations
46
47 """Habillage du graphique"""
48 plt.title("le 08/11/2022 de 9h00 à 14h49 (1min d'intervalle")
49 plt.xlabel("MINUTES")
50 plt.ylabel("COTATION")
51 plt.legend()
52 plt.show()

```

MOYENNE MOBILE

```
1  # -*- coding: utf-8 -*-
2  """
3  @author: Lea Pimpernelle
4  """
5
6  import pandas as pd
7  import numpy as np
8  import matplotlib.pyplot as plt
9
10 '''importation du fichier csv'''
11 data = pd.read_csv("quatre.csv",sep=';', index_col = "Libellé") # imparatition du fichier
12 #print(data)
13
14 '''Listes'''
15 lib = "L'OREAL" # libellé étudié
16 Y=data.loc[lib] # Création d'une liste des valeurs du libellé choisis
17 Y=[float(x) for x in Y] # str -> float
18 X = np.arange(1, len(Y)+1, 1, dtype=int) # liste avec nombre de valeurs prises
19
20 '''Moyenne mobile'''
21 def moving_average(x, w):
22     return np.convolve(x, np.ones(w), 'valid') / w #calcul des moyennen en continue
23
24 '''GRAPHIQUE'''
25 plt.plot(X,Y,c='k',label="Cotations de L'Oréal") # courbe du libellé étudié
26 #plt.scatter(X,Y,c='k',marker='+',label='cotation') # points des cotations
27
28 plt.plot(X[1:],moving_average(Y,2),'orange',label='Moyenne Mobile') # moyenne mobile
29
30
31 '''Habillage du graphique'''
32 plt.title("le 08/11/2022 de 09h00 à 14h59 (1min)")
33 plt.xlabel("MINUTES")
34 plt.ylabel("COTATION")
35 plt.legend()
36 plt.show()
```