RÉPUBLIQUE TUNISIENNE

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET

DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ SESAME

CYCLE INGENIEUR EN INFORMATIQUE
PROJET DE MODULE FOUILLE DE DONNEES

Analyses et prévisions du marché boursier Tunisien

Présenté par :

Azer Ltifi

Hichem Ben Hamed

Seddik Klaa

Année universitaire : 2020-2021

Table des matières

1 Présentation Générale **4**

1.1 Cadre académique du projet 4

1.1.1 Problématique 4

1.1.2 Solution Proposée 5

1.2 Source des données 5

2 Collecte et traitement des données **6**

2.1 Collecte des données 6

2.1.1 Packages 6

2.1.2 La collecte 6

2.2 Traitement des données 7

3 Études et analyse des Données **12**

3.1 Graphes généraux et Analyses 12

4 Problématique Décisionnelle **16**

4.1 Prédiction avec KPP 16

4.1.1 L'algorithme KPP 16

4.1.2 Implémention 16

4.1.3 Conclusion 19

4.2 Prédiction avec XGboost 19

4.2.1 L'algorithme XGboost 19
4.2.2 Implémention 19
4.2.3 Conclusion 21
4.3 Prédiction avec L'auto-régression 21
4.3.1 L'algorithme d'auto-régression 21
4.3.2 Implémentation 22
4.3.3 Conclusion 23
4.4 Prédiction avec Orange DataMining 24
4.4.1 Présentation d'Orange DataMining 24
4.4.2 Implémention 24
4.4.3 Conclusion 25

Liste des figures

2.1 downTse.py & downloaderBvmt.py 7

2.2 convertPDFtoCsvTable.py et exctractFromCSV.py 8

2.3 Capture de pd.set_index 8

2.4 capture d'une des fonctions utilisées pour centraliser les données 9

2.5 Capture de la concaténation de toutes les données du groupe 11 9

2.6 Capture 2 de la concaténation 10

2.7 Capture 3 de la concaténation 10

2.8 Capture de l'application de pd.dropna() 11

2.9 Capture de la fonction pd.fillna() 11

3.1 TunIndex vs SFBT 12

3.2 étude de corrélation 13

3.3 Lag Plot 14

3.4 AutoCorrélation Plot 14

4.1 Importation de Packages 17

4.2 Test/Train Split 17

4.3 Instanciation du KPP 17

4.4 Capture pour K=1 17

4.5 Graphe pour K = 1 18

4.6 Variation de l'erreur par rapport à K 18

4.7 Prédiction du TunIndex pour K = 2 19

4.8 Capture d'Importation des packets 20

4.9 Application de pandas.shift() sur les valeur d'ouverture de TunIndex 20

4.10 Capture de Train/Test split 20

4.11 Fonction validate pour effectuer des prédictions 20

4.12 Capture de la prédiction de Xgboost vs les valeurs reels de TunIndex 21

4.13 Capture de l'importation des paquets 22

4.14 Application de pdřead_csv pour charger les valeurs du TunIndex 22

4.15 Découpage en Train/Test Set 22

4.16 Entraînement du modèle d'auto-régression 23

4.17 Capture du Test 23

4.18 Capture des valeurs prédites vs valeurs réelles 23

4.19 Orange DataMining1 24

4.20 Orange DataMining2 25

Introduction Générale

La fouille de données aussi connue sous le terme data Mining, est une discipline qui a pour objet l'extraction d'un savoir, modèle, ou bien d'une connaissance à partir des données. Certainement l'efficacité des algorithmes et des pratiques de fouille de données implique que les donnés soient volumineuse.

En réalité la fouille de données, ou bien l'extraction des connaissances à partir de données n'est pas résante car certaines pratiques dans l'histoire se conforment à cette définition. Sachant qu'on peut citer des exemples préhistoriques (allant jusqu'à 2238 av .j.c), l'analyse réelle et supportée par un modèle mathématique provient originalement du livre *«Natural and Political Observations Made upon the Bills of Mortality»* de *John Graunt* publiée en 1662 dans le quel il analyse la mortalité due à la peste noir à Londres, et essaye de prédire son évolution. Depuis plusieurs oeuvres se sont succédé on en cite les travaux de *Thomas Bayes*, *Laplace* et *Legendre* qui ont mis les piliers mathématiques pour cette discipline.

Ce n'est que dans les années 1950 que la fouille de données tel que nous la connaissons a commencé à apparaître, plusieurs expérimentations et recherches sur des ordinateurs ont vu le jour et avec eux la méthode *bayésienne*, les réseaux de neurones... Encore plus récemment avec l'évolution exponentielle des ordinateurs sont venues les méthodes d'apprentissage automatique, encore plus avec Internet et la communauté Open Source tout est devenu à disposition du grand public pour extraire, modéliser, analyser des phénomènes à partir d'un grand nombre de données, des bibliothèques dédiées ont vu le jour, des plates-formes sociales et surtout des compétitions qui ont servi à l'évolution de ce domaine et à renforcer son aspect multidisciplinaire.

La fouille de données de nos jours est devenue indiscernable des activités industrielles, financière ou de recherche, car ces secteurs adoptent de plus en plus une partie de fouille de données que ce soit dans la partie de validation, optimisation, test, vu leur efficacité.

Le mot « bourse » désigne le lieu public où s'assemblent, à certaines heures, les négociants, les banquiers, les agents de change, les courtiers, pour traiter d'affaires. Son origine vient du nom

du lieu où les échanges de créances et de titres divers entre banquiers s'effectuaient au XIVème siècle. Dans la ville de Bruges, qui était à l'époque un très important centre du commerce d'argent, les transactions avaient lieu devant la maison de la famille Ter Beurse, d'où le nom donné au lieu où s'effectuaient les échanges de créances et de titres divers entre banquiers.

Et depuis, des milliers de trillions de dollars ont apparu en échange, en 2020 seulement la capitalisation du marcher boursière mondial est estimé à 85 trillions de dollars. En réalité l'année 2020 à marquer une hausse de 320% qui est la plus importante depuis l'année 2009.

De nos jours l'activité boursière n'est plus conduite par des humains, tout est automatisé, l'ère où des intermédiaires en bourse se déchirent pour l'achat/vente d'un bien boursier n'est plus. Il suffit de remarquer que les plus grandes fermes d'intermédiaires en bourse(dans les États-Unis Amérique) ont cumulé des centaines et des centaines de Billions de dollars grâce à des systèmes automatisés qui effectuent toute l'activité bourse.

La fouille de données étant omniprésente dans ce domaine, les techniques et les différents algorithmes sont utilisés principalement pour :

- L'analyse de l'évolution des biens boursiers.
- La détermination des signales Alpha.
- Validation de modèle de prévision.
- Découverte de modèle décrivant l'évolution d'une quote.

Le travail proposée aborde la rencontre du domaine boursier et celui de la fouille de données, et se propose de l'appliquer sur le marché boursier Tunisien. Avant d'aborder les détails du travail réalisée il se doit de citer quelque caractéristiques du marché boursier Tunisien, car il diffère du reste du monde, donc l'application de certaines techniques n'est pas possible vu :

- L'asymétrie de l'information : Certaines information qui sont impératifs pour quelque analyses quantitatives et qualitatives ne sont pas à la disposition du grand public, ce qui rend plus difficile l'analyse et la prévision de la majorité des cotas
- La bourse tunisienne n'est pas aussi sophistiquée que celles du reste du monde, en effet devenir intermédiaire en bourse est un presque un luxe, vu les lois compliquées qui organisent cette activité
- Le marchée boursier souffre de la fraude et du blanchiment d'argent, plusieurs phénomènes de blanchiment d'argents sont présents, malgré les lois présentes, la bureaucratie et le manque de numérisation de l'activité on fait que la fraude soit présente

Afin de réalisée le travail demandée, toute tache de collecte, pré-traitement, analyse et prédiction à été effectuée moyennant le langage de programmation Python. En effet Python est un langage complet et riche en ressources . Il faut bien noter que dans ce projet on a adopter le workflow classique du traitement d'un problème de fouille de données :

1. Collecte de données
2. Pré-traitement
3. Analyse
4. Modélisation
5. Conclusion

Chapitre 1

Présentation Générale

Introduction

Dans ce chapitre une description détaillée du synopsis du projet, et de la solution proposée sera présenté.

1.1 Cadre académique du projet

Dans le cadre du module de fouille de donnée, assuré par Dr.Tarek Hamrouni, il nous est demander de traiter une problématique décisionnelle, après avoir causer et échanger sur les différents idées de projets nous avons opter pour la prévision et l'analyse du marché boursier Tunisiens vu que les données sont abondantes et ouvertes au grand publique mais aussi les ressources et les application de la fouille de données sur la bourse est une discipline a part entière et les ressources académique et de recherche sont abondantes.

1.1.1 Problématique

Prédire au mieux l'évolution du TunIndex, à travers :

- L'historique du TunIndex
- L'historique des sociétés du Groupe 11, qui constituent le TunIndex
- Des variable socio-économique

1.1.2 Solution Proposée

L'implémentation des différents algorithmes de fouille de données avec en entrée les variables les plus corrélées et les plus significatives afin de prédire l'évolution du prix d'ouverture du TunIndex.

1.2 Source des données

Une tache de fouille de données n'est complète qu'avec des données, diverses et multiples. En Tunisie les données boursière sont à la fois centralisée et dispersée. Par exemple pour avoir l'historique complète du groupe 11 il faut se rendre aux cites : http ://www.tse.tn et http ://www.bvmt.com.tn/, pour avoir de différents informations socio-économique il faut se rendre sur https ://www.bct.gov.tn/ , le site officiel de la banque centrale de Tunisie, ainsi qu'au site de l'institut national de statistique http ://ins.tn/.

Conclusion

Dans le prochain chapitre on couvera en détail le processus de collecte de données des différentes sources cité, ainsi que le processus du traitement des données collectées.

Chapitre 2

Collecte et traitement des données

Introduction

Dans ce chapitre on couvera en un premier temps le processus de collecte adoptée et en second temps le traitement des données.

2.1 Collecte des données

2.1.1 Packages

Python est un langage riche en bibliothèque permettant le téléchargement automatisée des données, dans ce processus Les bibliothèques suivantes ont été utilisées

- Le package python *Beautiful Soup* : Qui permet d'extraire des données à partir des documents HTML et XML, il permet la navigation facile et souple des documents HTML et XML, c'est package reconnu pour faciliter et automatiser des tache de collecte de données.
- Le package *Requests* : Il permet l'envoi de requêtes HTTP facilement et rapidement afin de télécharger tout fichier.

2.1.2 La collecte

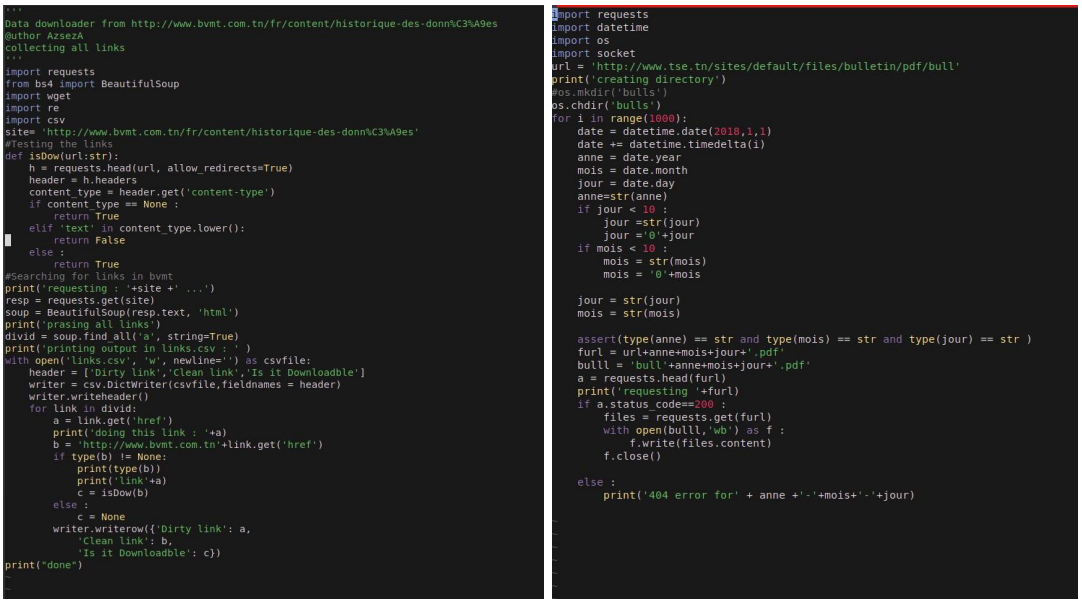
Chaque source de données à été traité séparément, un fichier *.py à été crée pour automatiser le téléchargement des données. On a crée 4 fichiers

- downloaderBvmt.py : permet l'extraction des donnés de http ://www.bvmt.com.tn/
- downTse.py : permet l'extraction des données de http ://www.tse.tn

— ins.cs : permet l'extraction des données de http ://ins.tn/ (L'institut national de statistique comporte un API qui n'est compatible qu'avec des téléchargement en C#)

— doingit.py : permet le téléchargement de données.

Ci-dessous deux captures des fichier utilisées pour le téléchargement des données.



```
'''
Data downloader from http://www.bvmt.com.tn/fr/content/historique-des-donn%C3%A9es
@uthor AzsezA
collecting all links
'''
import requests
from bs4 import BeautifulSoup
import wget
import re
import csv
site= 'http://www.bvmt.com.tn/fr/content/historique-des-donn%C3%A9es'
#Testing the links
def isDow(url,str):
    h = requests.head(url, allow_redirects=True)
    header = h.headers
    content_type = header.get('content-type')
    if content_type == None :
        return True
    elif 'text' in content_type.lower():
        return False
    else :
        return True
#Searching for links in bvmt
print('requesting : '+site +' ...')
resp = requests.get(site)
soup = BeautifulSoup(resp.text, 'html')
print('prasing all links')
divid = soup.find_all('a', string=True)
print('printing output in links.csv : ' )
with open('links.csv', 'w', newline='') as csvfile:
    header = ['Dirty link','Clean link','Is it Downloadble']
    writer = csv.DictWriter(csvfile,fieldnames = header)
    writer.writeheader()
    for link in divid:
        a = link.get('href')
        print('doing this link : '+a)
        b = 'http://www.bvmt.com.tn'+link.get('href')
        if type(b) != None:
            print(type(b))
            print('link'+a)
            c = isDow(b)
        else :
            c = None
        writer.writerow({'Dirty link': a,
            'Clean link': b,
            'Is it Downloadble': c})
print("done")
```

```
import requests
import datetime
import os
import socket
url = 'http://www.tse.tn/sites/default/files/bulletin/pdf/bull'
print('creating directory')
#os.mkdir('bulls')
os.chdir('bulls')
for i in range(1000):
    date = datetime.date(2018,1,1)
    date += datetime.timedelta(i)
    anne = date.year
    mois = date.month
    jour = date.day
    anne=str(anne)
    if jour < 10 :
        jour =str(jour)
        jour ='0'+jour
    if mois < 10 :
        mois = str(mois)
        mois = '0'+mois

    jour = str(jour)
    mois = str(mois)

    assert(type(anne) == str and type(mois) == str and type(jour) == str )
    furl = url+anne+mois+jour+'.pdf'
    bulll = 'bull'+anne+mois+jour+'.pdf'
    a = requests.head(furl)
    print('requesting '+furl)
    if a.status_code==200 :
        files = requests.get(furl)
        with open(bulll,'wb') as f :
            f.write(files.content)
        f.close()

    else :
        print('404 error for' + anne +'-'+mois+'-'+jour)
```

FIGURE 2.1 – downTse.py & downloaderBvmt.py

2.2 Traitement des données

Le téléchargement automatisée des fichier à aboutit à des centaines de fichiers (800) en format PDF, avant de centraliser les données dans un seul fichier, il faut tout d'abord convertir ces derniers en des fichier tabulaires (csv), pour se faire on a utiliser les packages suivants

— tabula : un package python qui convertit tout tableau en format pdf en fichier csv

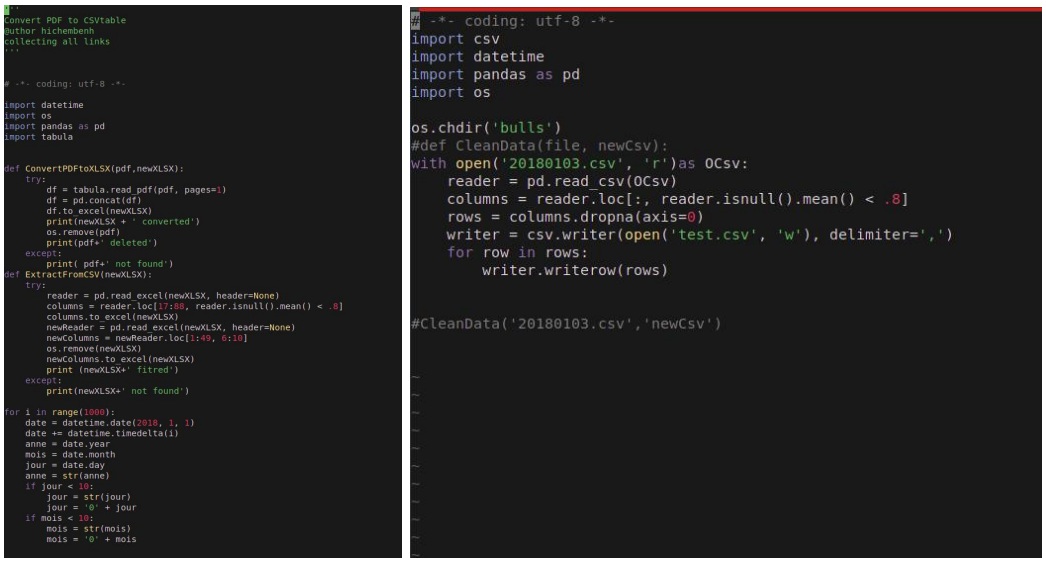
— csv : un package dédiée au traitement des fichier csv pour reformater les fichier convertit

Pour se faire deux fichier ont été crée, toujours en Python

— convertPDFtoCsvTable.py : permet de convertir les tableau dans les fichier *.pdf en fichier *.csv

— exctractFromCSV.py : permet de formater les fichier csv obtenu en des fichier plus lisibles et facile à manipuler

Ci-dessous deux captures des fichiers utilisées



```
'''
Convert PDF to CSVtable
@author hichembenh
collecting all links
'''


# -*- coding: utf-8 -*-

import datetime
import os
import pandas as pd
import tabula


def ConvertPDFtoXLSX(pdf,newXLSX):
    try:
        df = tabula.read_pdf(pdf, pages=1)
        df = pd.concat(df)
        df.to_excel(newXLSX)
        print(newXLSX + ' converted')
        os.remove(pdf)
        print(pdf+' deleted')
    except:
        print( pdf+' not found')
def ExtractFromCSV(newXLSX):
    try:
        reader = pd.read_excel(newXLSX, header=None)
        columns = reader.loc[17:88, reader.isnull().mean() < .8]
        columns.to_excel(newXLSX)
        newReader = pd.read_excel(newXLSX, header=None)
        newColumns = newReader.loc[1:49, 6:10]
        os.remove(newXLSX)
        newColumns.to_excel(newXLSX)
        print (newXLSX+' fitred')
    except:
        print(newXLSX+' not found')

for i in range(1000):
    date = datetime.date(2018, 1, 1)
    date += datetime.timedelta(i)
    anne = date.year
    mois = date.month
    jour = date.day
    anne = str(anne)
    if jour < 10:
        jour = str(jour)
        jour = '0' + jour
    if mois < 10:
        mois = str(mois)
        mois = '0' + mois
```

```
# -*- coding: utf-8 -*-
import csv
import datetime
import pandas as pd
import os

os.chdir('bulls')
#def CleanData(file, newCsv):
with open('20180103.csv', 'r')as OCsv:
    reader = pd.read_csv(OCsv)
    columns = reader.loc[:, reader.isnull().mean() < .8]
    rows = columns.dropna(axis=0)
    writer = csv.writer(open('test.csv', 'w'), delimiter=',')
    for row in rows:
        writer.writerow(rows)


#CleanData('20180103.csv','newCsv')
```

FIGURE 2.2 – convertPDFtoCsvTable.py et exctractFromCSV.py

Après la conversion des fichiers téléchargés en *.csv,il es temps de centraliser les données et de construire la base de données. Ces opérations ont été effectuées dans un seul Kernel Python 'Feature Engineering.ipynb'. Dans ce fichier les opérations suivantes ont été effectuées

- Finalisation et centralisation des données
- Suppression des colonnes en NaN
- Remplacement des valeurs en NaN par la dernière valeur non nulle

Pour ce faire seulement *Pandas* comme package à été utilisée, c'est le package référence en Python pour la manipulation et l'analyse de données tabulaires.

1. Centralisation de données :
   (a) Indexer L'historique des valeurs du TunIndex par date

```
: tunInx.set_index("Date")
```

:

Date	Price	Open	High	Low	Vol.	Change %
2020-12-11	6,854.55	6,888.83	6,888.83	6,839.19	599.03K	-0.49%
2020-12-10	6,888.59	6,895.04	6,921.28	6,876.05	1.39M	-0.07%
2020-12-09	6,893.34	6,874.89	6,907.70	6,864.23	1.68M	0.29%
2020-12-08	6,873.25	6,857.74	6,883.17	6,837.52	2.74M	0.26%
2020-12-07	6,855.60	6,830.68	6,866.40	6,822.13	1.81M	0.39%
...	...	...	...	...	...	...
2018-01-08	6,309.85	6,265.08	6,325.98	6,264.27	632.10K	0.72%
2018-01-05	6,265.03	6,240.59	6,265.83	6,226.92	710.57K	0.26%
2018-01-04	6,248.93	6,258.01	6,280.60	6,228.69	547.88K	-0.13%
2018-01-03	6,257.17	6,191.54	6,257.24	6,190.56	516.34K	0.87%
2018-01-02	6,203.27	6,287.26	6,287.26	6,175.95	368.26K	-1.25%

FIGURE 2.3 – Capture de pd.set_index

(b) Centraliser les Données des différentes Sociétés cotées à la bourse en un seul fichier *.csv (CleaningCSV.ipnb)

i. Centraliser les bulletins d'activité boursière par société :

SFBT

```
[7]: sfbt = pd.DataFrame(columns=['SFBT', 'ref', 'ouverture', 'cloture'],
                         index=pd.date_range(start='2018-01-01', end='2020-12-31', freq='D'))
```

```
[12]: def sfbtCollect(inDf, outDf, index):
          index=str(index)
          try:
              x=inDf.loc[inDf[6]=='SFBT - Société de Fabrication des Boissons de Tunisie']
              x=x.reset_index(drop=True)
              outDf.at[index,"ref"] = x.at[0,7]
              outDf.at[index,"ouverture"] = x.at[0,8]
              outDf.at[index,"cloture"] = x.at[0,9]
          except KeyError :
              x = inDf.loc[inDf[7]=='SFBT - Société de Fabrication des Boissons de Tunisie']
              x = x.reset_index(drop=True)
              outDf.at[index,"ref"] = x.at[0,8]
              outDf.at[index,"ouverture"] = x.at[0,9]
              outDf.at[index,"cloture"] = x.at[0,10]

          print(" did this "+index+" moving on ")

      #boucle trés malformé :p

      for index in range(20180101,20201231,1):
          try:
              filee=str(index)+'.xlsx'
              x = pd.read_excel(filee)
              sfbtCollect(x,sfbt,str(index))
          except FileNotFoundError :
              print("miving on with  "+ str(index) )
```

FIGURE 2.4 – capture d'une des fonctions utilisées pour centraliser les données

ii. Concaténer l'historique de chaque société par Index :

concatinaning all resulted files in one big fat one :p

```
In [161]: allofIt=pd.concat([deliced, udahd, sotid, mbpsd, sahd, cellcomd, ecd, ccd, othd, telnetd, enakld, tunired, hanid,
                             artesd, sopatd, tprd, adwyad, esoknad, wifakd, sitsd, gifd, sotrapild, electrostard, tild, sotumagd,
                             sotetеld, stard, pghd, atld, cild, simpard, uibd, atbd, amend, bnad, stbd, btd, tld, bhd, biatd,
                               tijarid, tunisAird, sfbt], axis=1)
```

```
In [162]: allofIt
```

Out[162]:

	Delice	ref	ouverture	cloture	UDAH	ref	ouverture	cloture	SOTIPAPIER	ref	...	ouverture	cloture	tunisAir	ref	ouverture	cloture
2018-01-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
2018-01-02	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
2018-01-03	NaN	16,500	16,690	16,500	NaN	2,450	2,450	2,460	NaN	3,520	...	36,000	36,150	NaN	0,430	0,430	0,42(
2018-01-04	NaN	16,500	16,980	16,980	NaN	2,460	2,540	2,600	NaN	3,500	...	36,160	36,470	NaN	0,420	0,420	0,43(
2018-01-05	NaN	16,980	16,900	16,900	NaN	2,600	2,690	2,690	NaN	3,490	...	36,410	36,860	NaN	0,430	0,420	0,42(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
2020-12-27	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
2020-12-28	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
2020-12-29	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
2020-12-30	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
2020-12-31	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN

1096 rows × 172 columns

```
In [163]: allofIt.to_csv('corpData.csv')
```

FIGURE 2.5 – Capture de la concaténation de toutes les données du groupe 11

(c) Indexer Les variables socio-économique par Date

(d) Concaténer le tout Dans un seul tableau par index de Date

```
sndC=soc.join(corp, lsuffix='_Unnamed: 0', rsuffix= '_Unnamed: 0')#pandas.DataFrame.join() is the best way
```

```
sndC
```

	Unnamed: 0_Unnamed: 0	IPI	MTT	Import	Export	BalCom	Pib1	Pib2	Unnamed: 0_Unnamed: 0	Delice	...	ouverture.40	cloture.40	tunisAir	ref.41	ouverture.41	clo
0	2018-01-01	92.8	5.53	3129.5	4339.0	NaN	1694.2	18158.6	2018-01-01	NaN	...	NaN	NaN	NaN	NaN	NaN	
1	2018-01-02	92.8	5.53	3129.5	4339.0	NaN	1694.2	18158.6	2018-01-02	NaN	...	NaN	NaN	NaN	NaN	NaN	
2	2018-01-03	92.8	5.53	3129.5	4339.0	NaN	1694.2	18158.6	2018-01-03	NaN	...	36,000	36,150	NaN	0,430	0,430	
3	2018-01-04	92.8	5.53	3129.5	4339.0	NaN	1694.2	18158.6	2018-01-04	NaN	...	36,160	36,470	NaN	0,420	0,420	
4	2018-01-05	92.8	5.53	3129.5	4339.0	NaN	1694.2	18158.6	2018-01-05	NaN	...	36,410	36,860	NaN	0,430	0,420	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1091	2020-12-27	NaN	NaN	NaN	NaN	-1187.8	NaN	NaN	2020-12-27	NaN	...	NaN	NaN	NaN	NaN	NaN	
1092	2020-12-28	NaN	NaN	NaN	NaN	-1187.8	NaN	NaN	2020-12-28	NaN	...	NaN	NaN	NaN	NaN	NaN	
1093	2020-12-29	NaN	NaN	NaN	NaN	-1187.8	NaN	NaN	2020-12-29	NaN	...	NaN	NaN	NaN	NaN	NaN	
1094	2020-12-30	NaN	NaN	NaN	NaN	-1187.8	NaN	NaN	2020-12-30	NaN	...	NaN	NaN	NaN	NaN	NaN	
1095	2020-12-31	NaN	NaN	NaN	NaN	-1187.8	NaN	NaN	2020-12-31	NaN	...	NaN	NaN	NaN	NaN	NaN	

1096 rows × 181 columns

FIGURE 2.6 – Capture 2 de la concaténation

(e) Concaténer les différents fichiers par index de date (TunIndex, Variables Socio-économique, Sociétés du Groupe 11)

```
In [85]: b=soc.set_index('Unnamed: 0').join(corp.set_index('Unnamed: 0'))
         b
```

Out[85]:

Unnamed: 0	IPI	MTT	Import	Export	BalCom	Pib1	Pib2	Delice	ref	ouverture	...	ouverture.40	cloture.40	tunisAir	ref.41	ouverture.41	cloture
2018-01-01	92.8	5.53	3129.5	4339.0	NaN	1694.2	18158.6	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	
2018-01-02	92.8	5.53	3129.5	4339.0	NaN	1694.2	18158.6	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	
2018-01-03	92.8	5.53	3129.5	4339.0	NaN	1694.2	18158.6	NaN	16,500	16,690	...	36,000	36,150	NaN	0,430	0,430	0,
2018-01-04	92.8	5.53	3129.5	4339.0	NaN	1694.2	18158.6	NaN	16,500	16,980	...	36,160	36,470	NaN	0,420	0,420	0,
2018-01-05	92.8	5.53	3129.5	4339.0	NaN	1694.2	18158.6	NaN	16,980	16,900	...	36,410	36,860	NaN	0,430	0,420	0,
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2020-12-27	NaN	NaN	NaN	NaN	-1187.8	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	
2020-12-28	NaN	NaN	NaN	NaN	-1187.8	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	
2020-12-29	NaN	NaN	NaN	NaN	-1187.8	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	
2020-12-30	NaN	NaN	NaN	NaN	-1187.8	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	
2020-12-31	NaN	NaN	NaN	NaN	-1187.8	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	

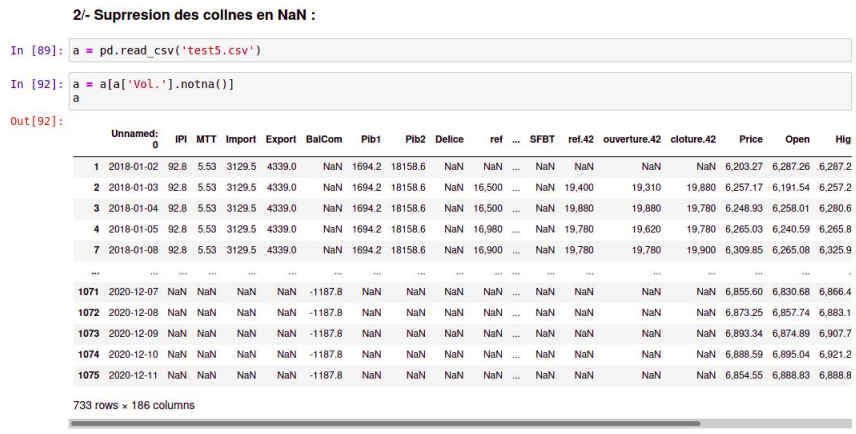
1096 rows × 179 columns

```
In [86]: c=b.join(tunInx)
```

```
In [87]: c.to_csv('test5.csv') #saving
```

FIGURE 2.7 – Capture 3 de la concaténation

2. Suppression des valeurs nulles et remplacement des valeurs en NaN par la dernière valeur non nulle



2/- Suprresion des colnes en NaN :

```
In [89]: a = pd.read_csv('test5.csv')

In [92]: a = a[a['Vol.'].notna()]
         a
```

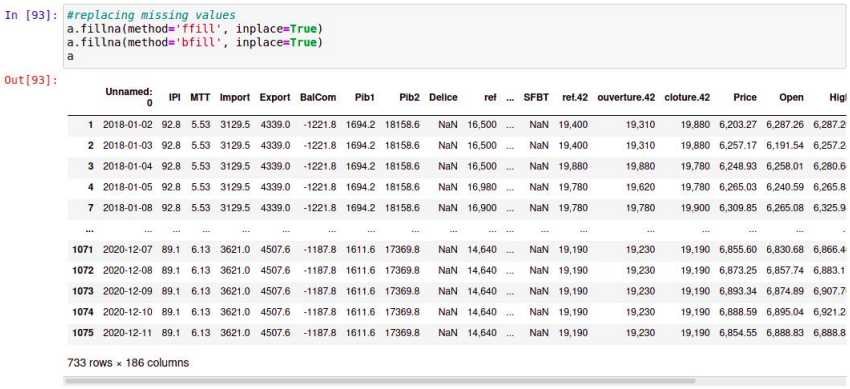
Out[92]:

	Unnamed: 0	IPI	MTT	Import	Export	BalCom	Pib1	Pib2	Delice	ref	...	SFBT	ref.42	ouverture.42	cloture.42	Price	Open	Hig
1	2018-01-02	92.8	5.53	3129.5	4339.0	NaN	1694.2	18158.6	NaN	NaN	...	NaN	NaN	NaN	NaN	6,203.27	6,287.26	6,287.2
2	2018-01-03	92.8	5.53	3129.5	4339.0	NaN	1694.2	18158.6	NaN	16,500	...	NaN	19,400	19,310	19,880	6,257.17	6,191.54	6,257.2
3	2018-01-04	92.8	5.53	3129.5	4339.0	NaN	1694.2	18158.6	NaN	16,500	...	NaN	19,880	19,880	19,780	6,248.93	6,258.01	6,280.6
4	2018-01-05	92.8	5.53	3129.5	4339.0	NaN	1694.2	18158.6	NaN	16,980	...	NaN	19,780	19,620	19,780	6,265.03	6,240.59	6,265.8
7	2018-01-08	92.8	5.53	3129.5	4339.0	NaN	1694.2	18158.6	NaN	16,900	...	NaN	19,780	19,780	19,900	6,309.85	6,265.08	6,325.9
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
1071	2020-12-07	NaN	NaN	NaN	NaN	-1187.8	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	6,855.60	6,830.68	6,866.4
1072	2020-12-08	NaN	NaN	NaN	NaN	-1187.8	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	6,873.25	6,857.74	6,883.1
1073	2020-12-09	NaN	NaN	NaN	NaN	-1187.8	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	6,893.34	6,874.89	6,907.7
1074	2020-12-10	NaN	NaN	NaN	NaN	-1187.8	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	6,888.59	6,895.04	6,921.2
1075	2020-12-11	NaN	NaN	NaN	NaN	-1187.8	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	6,854.55	6,888.83	6,888.8

733 rows × 186 columns

FIGURE 2.8 – Capture de l'application de pd.dropna()

3. Remplacement des valeurs en NaN par la dernière valeur non nulle



```
In [93]: #replacing missing values
         a.fillna(method='ffill', inplace=True)
         a.fillna(method='bfill', inplace=True)
         a
```

Out[93]:

	Unnamed: 0	IPI	MTT	Import	Export	BalCom	Pib1	Pib2	Delice	ref	...	SFBT	ref.42	ouverture.42	cloture.42	Price	Open	Higl
1	2018-01-02	92.8	5.53	3129.5	4339.0	-1221.8	1694.2	18158.6	NaN	16,500	...	NaN	19,400	19,310	19,880	6,203.27	6,287.26	6,287.2
2	2018-01-03	92.8	5.53	3129.5	4339.0	-1221.8	1694.2	18158.6	NaN	16,500	...	NaN	19,400	19,310	19,880	6,257.17	6,191.54	6,257.2
3	2018-01-04	92.8	5.53	3129.5	4339.0	-1221.8	1694.2	18158.6	NaN	16,500	...	NaN	19,880	19,880	19,780	6,248.93	6,258.01	6,280.6
4	2018-01-05	92.8	5.53	3129.5	4339.0	-1221.8	1694.2	18158.6	NaN	16,980	...	NaN	19,780	19,620	19,780	6,265.03	6,240.59	6,265.8
7	2018-01-08	92.8	5.53	3129.5	4339.0	-1221.8	1694.2	18158.6	NaN	16,900	...	NaN	19,780	19,780	19,900	6,309.85	6,265.08	6,325.9
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
1071	2020-12-07	89.1	6.13	3621.0	4507.6	-1187.8	1611.6	17369.8	NaN	14,640	...	NaN	19,190	19,230	19,190	6,855.60	6,830.68	6,866.4
1072	2020-12-08	89.1	6.13	3621.0	4507.6	-1187.8	1611.6	17369.8	NaN	14,640	...	NaN	19,190	19,230	19,190	6,873.25	6,857.74	6,883.1
1073	2020-12-09	89.1	6.13	3621.0	4507.6	-1187.8	1611.6	17369.8	NaN	14,640	...	NaN	19,190	19,230	19,190	6,893.34	6,874.89	6,907.7
1074	2020-12-10	89.1	6.13	3621.0	4507.6	-1187.8	1611.6	17369.8	NaN	14,640	...	NaN	19,190	19,230	19,190	6,888.59	6,895.04	6,921.2
1075	2020-12-11	89.1	6.13	3621.0	4507.6	-1187.8	1611.6	17369.8	NaN	14,640	...	NaN	19,190	19,230	19,190	6,854.55	6,888.83	6,888.8

733 rows × 186 columns

FIGURE 2.9 – Capture de la fonction pd.fillna()

Conclusion

Dans ce chapitre toutes les opération de collecte et de centralisation de données ont été couvertes. Dans le prochain chapitre l'analyse des données et l'exploration sera couverte en détail.

Chapitre 3

Études et analyse des Données

Introduction

Toute tache de fouille de données doit être précède par une phase d'exploration et d'analyse, pour bien mener les taches prédicatives et assurer un taux d'erreur minimal.

3.1 Graphes généraux et Analyses

Tout traitement de donnée induit une perte d'information, cette perte d'information est claire lorsqu'on effectue un graphe de l'évolution du TunIndex par l'une des société du groupe 11 par exemple :

FIGURE 3.1 – TunIndex vs SFBT

Ce graphe montre que l'évolution du cota de l'SFBT par rapport à celle du TunIndex n'est

pas continu, cet exemple montre que la suppression des valeurs en NaN nous a fait perdre des informations sur l'évolution de la quote de l'SFBT.

Mais aussi la partie droite du graphe en noir (SFBT) décrit un plateau ce qui ne convient pour aboutir à une prédiction précise. On conclu que les variables financières ne sont pas appropriées pour une bonne prédiction. Pour avoir une une idée précise sur la corrélation du TunIndex avec toutes les variables proposée on a recours à la bibliothèque Seaborn de Python qui offre une fonction pour calculer la corrélation deux à deux, la figure ci-dessous montre l'implémention de cette fonction et son résultat :

```
In [29]: mask = np.zeros_like(anal_corr)
         mask[np.triu_indices_from(mask)] = True
         plt.figure(figsize=(15,8))
         seaborn.heatmap(anal_corr, cmap='RdYlGn', vmax=1.0, vmin=-1.0 , mask = mask, linewidths=0.01, cbar=True)
         plt.yticks(rotation=0)
         plt.xticks(rotation=90)
         plt.show()
         plt.savefig('corr.png')
```

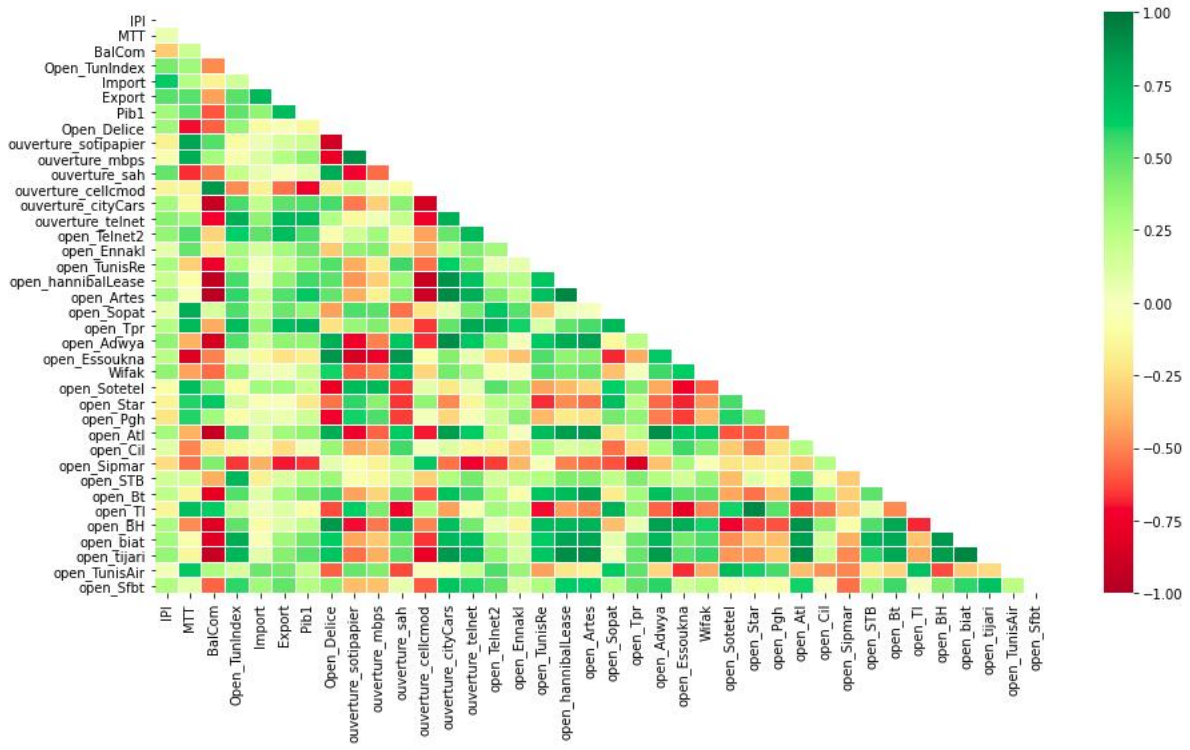


FIGURE 3.2 – étude de corrélation

On peut voir que la plus part des variables socio-économique ne sont pas du tout corrélée ni avec les sociétés du groupe 11 ni avec le TunIdex.

Si on ne peut pas prédire la variation du TunIndex avec les valeurs du groupe 11 et les variables socio-économique peut on le prédire en se basant seulement sur le TunIndex ?

En effectuant un 'Lag Plot' , un graphe indicateur de la variation de la valeur n+1 par rapport à la valeur n du TunIndex , on peut aisément dire que la prédiction du TunIdex basée sur son historique seulement est acceptable et pourra nous mener à un résultat acceptable Mais

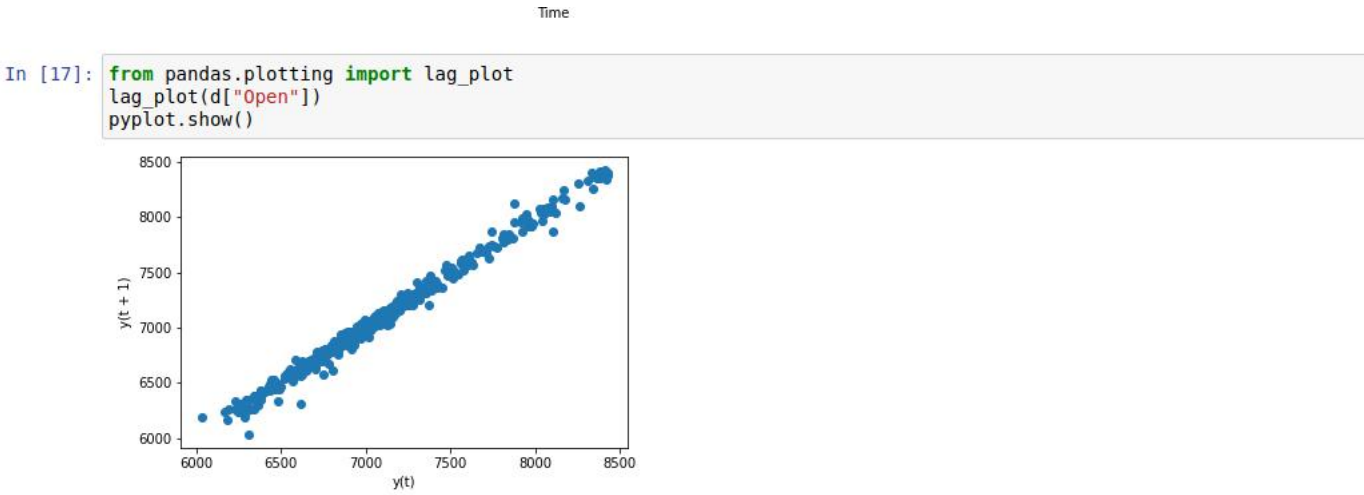


FIGURE 3.3 – Lag Plot

aussi si on effectue un graphe indicateur de l'auto-corrélation *'auto-corrélation-plot'* du package *'pandas.plotting'*, on peut clairement voir que le quotas du TunIndex est auto-corrélé ainsi tout ajout de variable extérieur induira en erreur les algorithmes de prédiction.

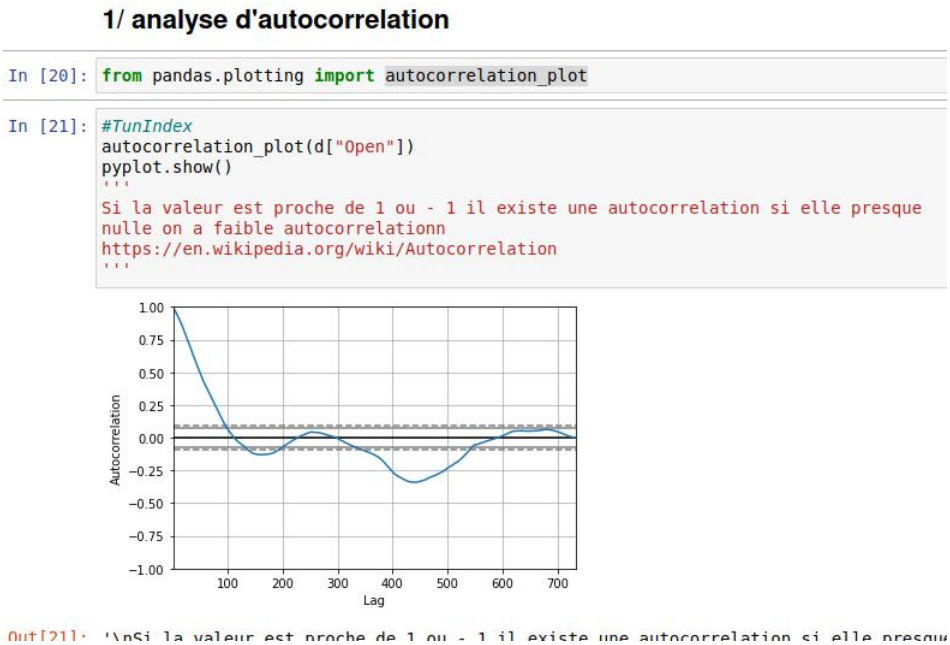


FIGURE 3.4 – AutoCorrélation Plot

Conclusion

Dans ce chapitre, on a effectuée des analyses qui vont nous guider pour mener à bien les taches de prédiction, on a découvert que

- TunIndex est auto-corrélé
- Seulement les valeurs historique du TunIndex suffisent pour prédire le TunIndex.
- Contrairement à ce qu'on a cru le TunIndex n'est pas corrélé avec la majorité des sociétés du groupe 11
- Contrairement à ce qu'on a cru le TunIndex n'est pas corrélée avec les indicateur socio-économique principaux

Chapitre 4

Problématique Décisionnelle

Introduction

Dans ce chapitre on va effectuer les différentes taches de prédiction, en se basant sur les observations du chapitre suivant.

4.1 Prédiction avec KPP

4.1.1 L'algorithme KPP

L'algorithme des k plus proches voisins est un algorithme d'apprentissage supervisée, dans le cadre de la prévision de l'évolution du TunIndex on dispose d'une base de données d'apprentissage *Data.csv* et de 733 couple entré-sortie qui sera divisée en 70% pour l'apprentissage et 30% pour le test. On aura recours à la variante de régression de cet algorithme qui est offert par le package *sklearn* du langage *Python*

4.1.2 Implémention

1. Importations des packages

```
In [1]: import pandas as pd
        import numpy as np
        import math
        import seaborn as sns; sns.set()
        import matplotlib.pyplot as plt
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.metrics import accuracy_score
        from sklearn.metrics import confusion_matrix
        from statistics import *
```

FIGURE 4.1 – Importation de Packages

2. Découpage en base d'apprentissage et test

```
In [2]: #Definition de la variable de prediction (X)
        data = pd.read_csv('Data.csv')
        data.set_index(["Unnamed: 0.1"], inplace=True)
        ti = pd.DataFrame(index = data.index)
        ti["Open-Close"] = data["Open"] - data["Price"]
        ti["High-Low "] = data["High"] - data["Low"]
        #definition La variablle cible (Y)
        tto = np.where(data["Price"].shift(-1)>data["Price"],1,-1)
```

FIGURE 4.2 – Test/Train Split

3. Instanciation de l'objet KPP

```
In [4]: #instancier le model KPP
        knn= KNeighborsClassifier(n_neighbors=1)

        # fit the model
        fitted_model = knn.fit(X_train, Y_train)
        predictions = fitted_model.predict(X_test)
```

FIGURE 4.3 – Instanciation du KPP

4. Implémentation pour K = 1

```
In [7]: #prediction
        ti.dropna()
        ti['Predicted_Signal'] = knn.predict(ti)
        ti.dropna()
        #calcul et analyse de la stratégie de prédiction
        ti["retour du tunindx"] = np.log(data["Price"]/data["Price"].shift(1))
        retour_cumulée_tunindex = ti[split:]["retour du tunindx"].cumsum()*100

        ti["retour stratégie"] = ti["retour du tunindx"] * ti['Predicted_Signal'].shift(1)
        retour_cummulé_stratégie =   ti[split:]["retour stratégie"].cumsum()*100
        ti
```

Out[7]:

Unnamed: 0.1	Open-Close	High-Low	Predicted_Signal	retour du tunindx	retour stratégie
2018-01-02	83.99	111.31	1	NaN	NaN
2018-01-03	-65.63	66.68	-1	0.008651	0.008651
2018-01-04	9.08	51.91	1	-0.001318	0.001318
2018-01-05	-24.44	38.91	1	0.002573	0.002573
2018-01-08	-44.77	61.71	-1	0.007129	0.007129

FIGURE 4.4 – Capture pour K=1

5. Graphe

```
In [8]: #Plotting
newdf = pd.concat([retour_cumulée_tunindex, retour_cummulé_stratégie], axis=1)
#Graph plotting

plt.figure(figsize=(25, 13))
sns.set(font_scale=1.4)
ax = sns.lineplot(data=newdf)
plt.setp(ax.get_legend().get_texts(), fontsize='19')
plt.savefig('returns.png', dpi=300)
```

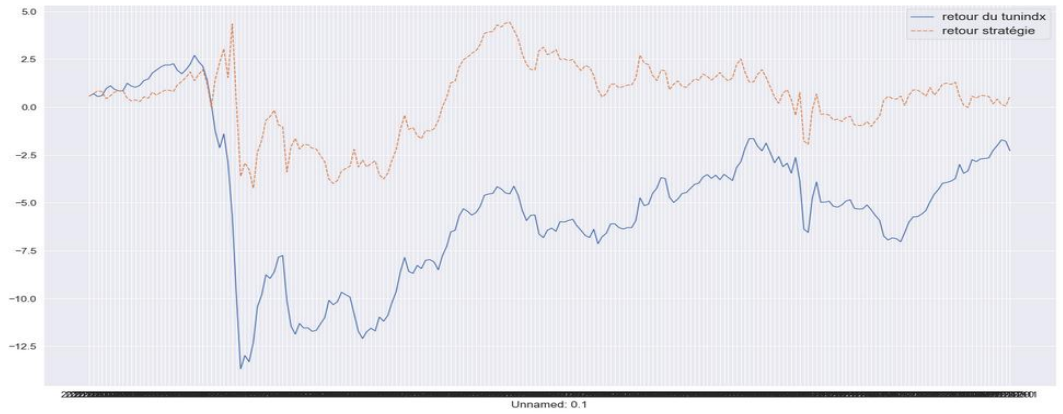


FIGURE 4.5 – Graphe pour K = 1

6. Recherche de la meilleur valeur de K

```
n [9]: #recherche de la valeur de K la plus optimale
diff_sumul= []
k_range = range(1,100)
topred = pd.DataFrame(index = ti.index)
topred = pd.concat([ti['Open-Close'], ti[ 'High-Low ']], axis=1)
for k in k_range:
    ti.dropna()
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, Y_train)
    ti[f'Predicted_Signal{k}'] = knn.predict(topred)
    ti[f"retour du tunindx{k}"] = np.log(data["Price"]/data["Price"].shift(1))
    retour_cumulée_tunindex = ti[split:][f"retour du tunindx{k}"].cumsum()*100
    ti[f"retour stratégie{k}"] = - ti[f"retour du tunindx{k}"] * ti[f'Predicted_Signal{k}'].shift(1)
    retour_cummulé_stratégie =  - ti[split:][f"retour stratégie{k}"].cumsum()*100
    diff_sumul.append(mean(int(x - y) for x, y in zip(retour_cummulé_stratégie, retour_cumulée_tunindex)))
```

FIGURE 4.6 – Variation de l'erreur par rapport à K

7. Prédiction optimale pour k = 2

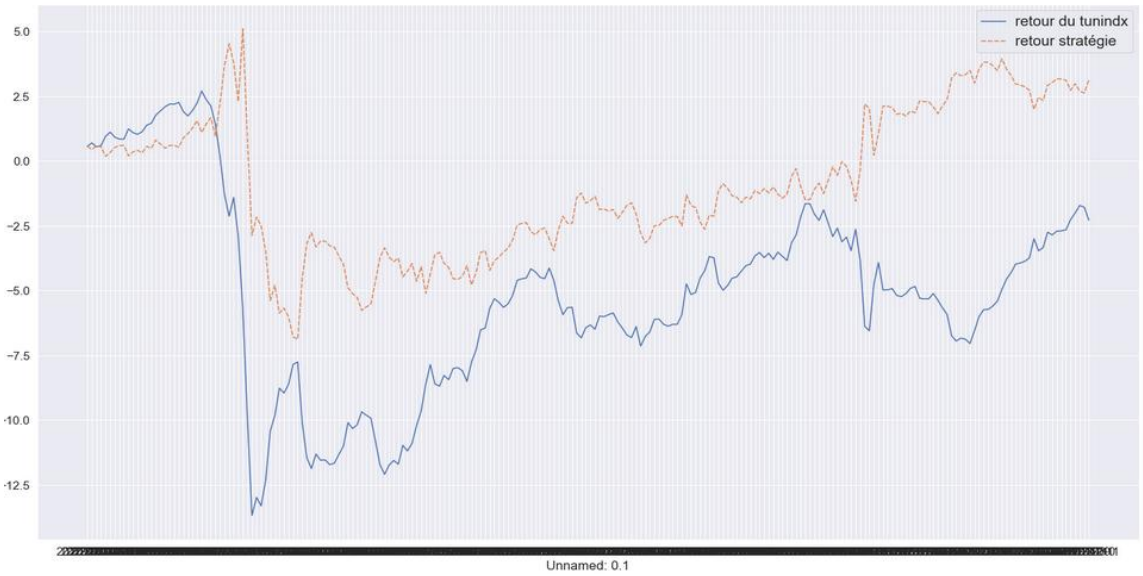


FIGURE 4.7 – Prédiction du TunIndex pour K = 2

4.1.3 Conclusion

Pour une valeur de k=2 (la plus optimale) le taux d'erreur dans les données d'entraînement est de 22% alors que pour le données de test le taux d'erreur est de 50%.

4.2 Prédiction avec XGboost

4.2.1 L'algorithme XGboost

XGBoost (comme eXtreme Gradient Boosting) est une implémentation open source optimisée de l'algorithme d'arbres de boosting de gradient. Le Boosting de Gradient est un algorithme d'apprentissage supervisé dont le principe et de combiner les résultats d'un ensemble de modèles plus simple et plus faibles afin de fournir une meilleur prédiction. Voyons ce que cet algorithme va donner.

4.2.2 Implémention

1. Importation des packages
2. Mise en forme des variables cibles
3. Découpage en données de test et d'entraînement

```
:  from xgboost import XGBRegressor as xgb
   from sklearn.metrics import mean_squared_error
```

FIGURE 4.8 – Capture d'Importation des packets

```
In [16]: def xgbt_predict(train, val):
             train = np.array(train)
             X , Y = train[: , :-1], train[: , -1]
             model = xgb(obective="reg:squarederror", n_esitmators=1000)
             model.fit(X, Y)

             val = np.array(val).reshape(1, -1)
             pred = model.predict(val)
             return pred
```

FIGURE 4.9 – Application de pandas.shift() sur les valeur d'ouverture de TunIndex

Train/Test Set Split

```
def train_test_split(data, pourcent):
    data = data.values
    n = int(len(data) * (1 - pourcent))
    print(data[n])
    return data[:n], data[n:]
```

FIGURE 4.10 – Capture de Train/Test split

4. Implémentation

```
In [17]: def validate(data, perc):
             prediction = []

             train, test = train_test_split(data, perc)
             historique = [x for x in train]

             for i in range(len(test)):
                 test_x, test_y = test[i , :-1], test[i , -1]
                 print(type(test_x))
                 pred = xgbt_predict(historique, test_x[0:])
                 prediction.append(pred)

                 historique.append(test[i])

             erreur = mean_squared_error(test[:,-1], prediction, squared=False)

             return  erreur ,test[:,-1], prediction
```

FIGURE 4.11 – Fonction validate pour effectuer des prédictions

5. visualisation

```
In [43]: #Graph de comparaison entre valuers prédite et valeurs réels pour TunIndex
         plt.figure(figsize=(12,5))
         plt.xlabel('Valuers prédites/réels TunIndex')

         ax1 = valeursTI['vals_tunIndex'].plot(color='blue', grid=True, label='valuers réels')
         ax2 = predictionsTI['pred_tunIndex'].plot(color='red', grid=True, secondary_y=True, label='

         ax1.legend(loc=1)
         ax2.legend(loc=2)

         plt.show()
```

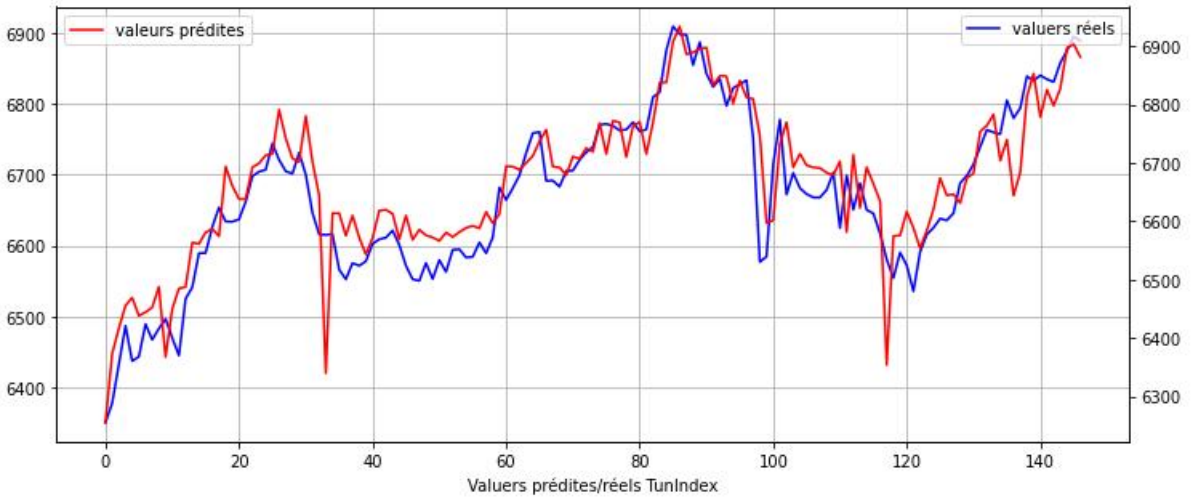


FIGURE 4.12 – Capture de la prédiction de Xgboost vs les valeurs reels de TunIndex

4.2.3 Conclusion

L'algorithme Xgboost est le plus précis jusqu'à présent avec un taux d'erreur de 12% .

4.3 Prédiction avec L'auto-régression

4.3.1 L'algorithme d'auto-régression

Un processus auto-régressif est un modèle de régression pour séries temporelles dans lequel la série est expliquée par ses valeurs passées plutôt que par d'autres variables. Le package *sklearn* offre une implémentation complète de cet algorithme, voyons ce que ça donne

4.3.2 Implémentation

1. Importation des packages

```
import pandas as pd
from matplotlib import pyplot
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
from statsmodels.tsa.ar_model import AR
from math import sqrt
%matplotlib inline
```

FIGURE 4.13 – Capture de l'importation des paquets

2. Chargement des valeurs cibles

```
: #chargement des valeur d'ouverture du TunIndex
  d = pd.read_csv('Data.csv')
  X = d["Open"]
  X.reset_index()
  len(X)

: 733
```

FIGURE 4.14 – Application de pdŕead_csv pour charger les valeurs du TunIndex

3. Découpage en Train/Test set

```
train, test = X[1:len(X)-15], X[len(X)-15:] #Découpage des valeur en Dest et train Data
train
```

```
1      6191.54
2      6258.01
3      6240.59
4      6265.08
5      6313.80
        ...
713    6645.79
714    6688.35
715    6698.63
716    6714.88
717    6739.41
Name: Open, Length: 717, dtype: float64
```

FIGURE 4.15 – Découpage en Train/Test Set

4. Entraînement

5. Test

```
: #entrainement du modéle de régression :
  model = AR(train)
  model_fit = model.fit()
  window = model_fit.k_ar
  coef = model_fit.params
```

FIGURE 4.16 – Entraînement du modèle d'auto-régression

```
#test du modéle autoregressif
history = train[len(train)-window:]
history.reset_index()
print(str(len(history)))
print(history)
history = [history[i] for i in range(699,717)]
predictions = list()
for t in range(718,732):
    length = len(history)
    lag = [history[i] for i in range(length-window,length)]
    yhat = coef[0]
    for d in range(window):
        yhat += coef[d+1] * lag[window-d-1]
    obs = test[t]
    predictions.append(yhat)
    history.append(obs)
    print( ' predicted=%f, expected=%f ' % (yhat, obs))
rmse = sqrt(mean_squared_error(test[:-1], predictions))
print( ' Test RMSE: %.3f ' % rmse)
```

FIGURE 4.17 – Capture du Test

4.3.3 Conclusion

Le modèle auto régressif est bien plus performant que celui du KPP mais moins performant avec un taux d'erreur de 24%

```
716    6714.88
717    6739.41
Name: Open, dtype: float64
 predicted=6719.387581, expected=6763.230000
 predicted=6769.953536, expected=6759.860000
 predicted=6758.280720, expected=6757.300000
 predicted=6756.870316, expected=6804.990000
 predicted=6815.415697, expected=6779.770000
 predicted=6784.331335, expected=6794.030000
 predicted=6799.620903, expected=6838.810000
 predicted=6846.208130, expected=6832.610000
 predicted=6843.621555, expected=6840.070000
 predicted=6847.933795, expected=6834.660000
 predicted=6837.813528, expected=6830.680000
 predicted=6841.432644, expected=6857.740000
 predicted=6869.198526, expected=6874.890000
 predicted=6884.617368, expected=6895.040000
 Test RMSE: 24.013
```

FIGURE 4.18 – Capture des valeurs prédites vs valeurs réelles

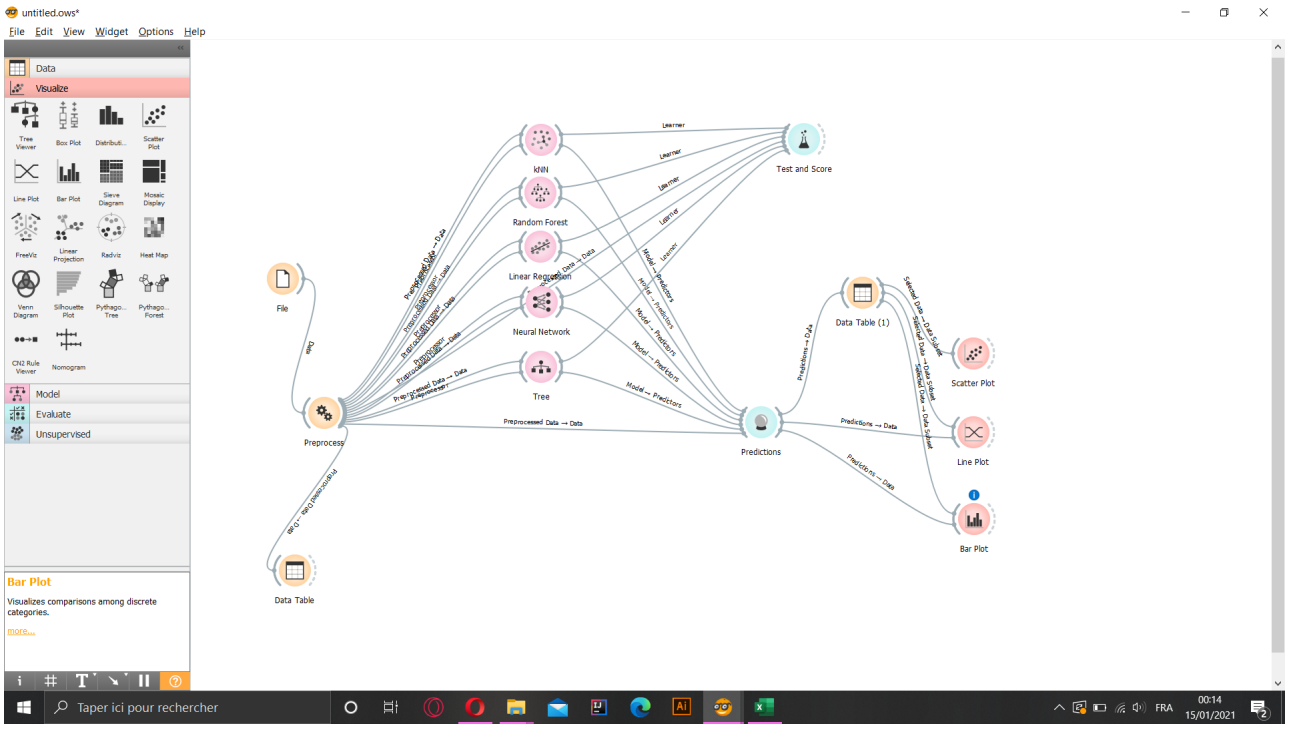


FIGURE 4.19 – Orange DataMining1

4.4 Prédiction avec Orange DataMining

4.4.1 Présentation d'Orange DataMining

Orange est un logiciel libre d'exploration de données (data mining). Il propose des fonctionnalités de modélisation à travers une interface visuelle, une grande variété de modalités de visualisation et des affichages variés dynamiques2. Développé en Python, il existe des versions Windows, Mac et Linux. Il sera utilisée pour valider la comparaison entre les différents algorithmes prédit

4.4.2 Implémention

Orange Data Mining offre une variété d'algorithmes prédicatifs l'un des plus importants du domaine boursier étant l'SVM, mais l'utilisation d'orange ne sera utile que pour comparer les différents algorithmes. ci-dessous une capture d'écran qui montre le paramétrage

Voici une comparaison entre les différents algorithmes prédicatifs

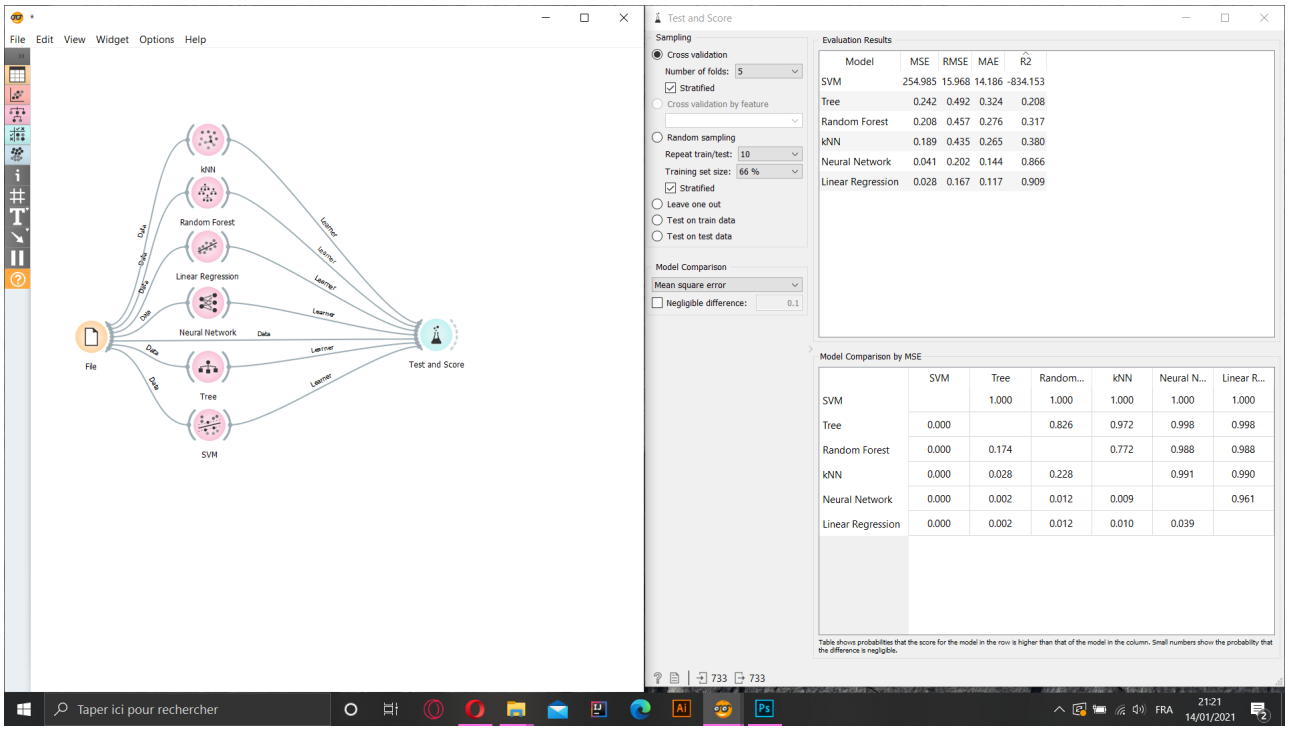


FIGURE 4.20 – Orange DataMining2

4.4.3 Conclusion

Le résultat d'orange DataMining est conforme à ce qu'on a obtenu lors de la prédiction du TunIndex en utilisant les algorithmes précédent, en effet Xgboost est le plus optimal, après viens le modèle auto-régressif finalement l'algorithme des KPP est le moins optimal

Conclusion

Ce projet incarne une découverte dans le monde financier, et une découverte du monde du Machine Learning appliquée au domaine financier. Tout les essais présents dans ce document peuvent certainement être utilisée pour aboutir à des applications plus concrètes et plus complexe et certainement plus lucratives. L'une des applications les plus reconnues étant celle du *Trading Bot*, qui n'est d'autre qu'un programme qui effectue l'analyse , la prévision et surtout effectue l'échange des valeurs boursiers d'une façon automatique le travail présentée peut faire l'oeuvre d'une introduction. Mais aussi les analyses quantitatives et qualitatives des valeurs boursière est devenu de plus en plus automatisée, mais aussi de plus en plus d'applications de Machine Learning et de Data Mining sont présentes dans le marché financier, qui sont certainement aussi lucratives d'intéressantes.