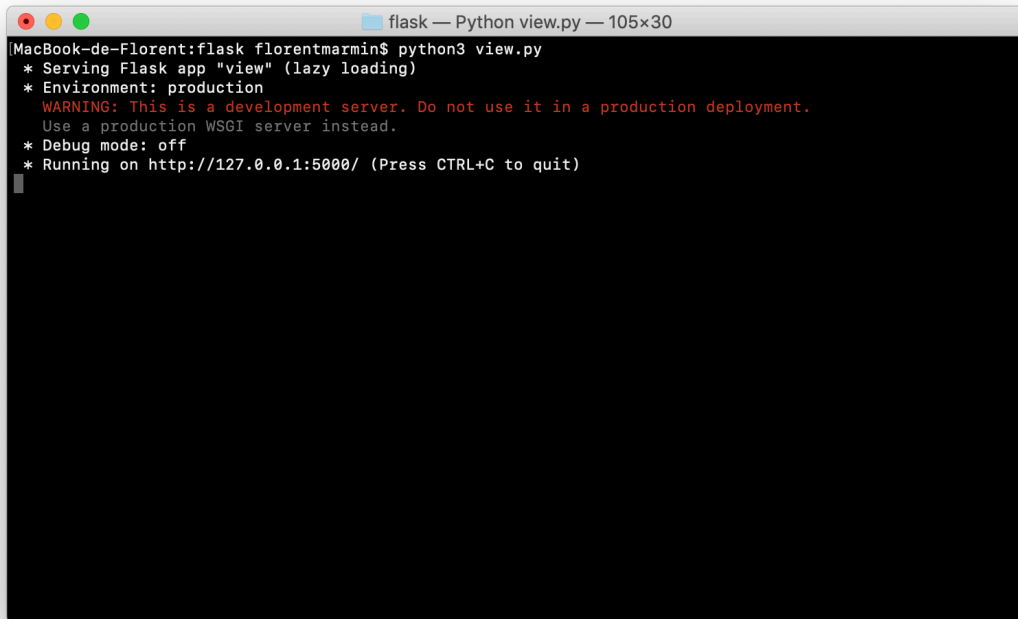


“If I had only known” website :

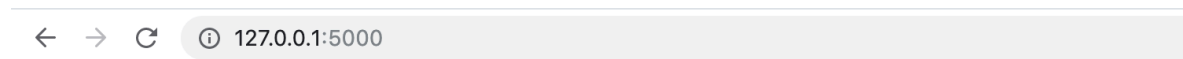
Nous avons utilisé Flask pour créer un site web local et nous avons récupéré les données boursiers depuis le site AlphaVantage (granularité de 5 minutes).

L’algorithme qui recherche les maximums de profits est lent (double boucle), il doit y avoir un moyen d’optimiser la méthode...

Exemple avec le 3 Juin 2019



```
flask — Python view.py — 105x30
MacBook-de-Florent:flask florentmarmin$ python3 view.py
* Serving Flask app "view" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```



Date Selection

Month

Day

Year

Show Top 10

Monday 03. June 2019

NKTR bought at 09:35AM and sold at 12:00PM would have made a 9.02% profit
GT bought at 09:35AM and sold at 02:20PM would have made a 5.63% profit
PVH bought at 10:05AM and sold at 02:15PM would have made a 5.49% profit
BWA bought at 09:35AM and sold at 02:15PM would have made a 5.23% profit
NWL bought at 09:35AM and sold at 02:10PM would have made a 5.05% profit
DVA bought at 09:35AM and sold at 02:15PM would have made a 4.68% profit
APTV bought at 09:35AM and sold at 02:15PM would have made a 4.55% profit
PKG bought at 09:45AM and sold at 02:15PM would have made a 4.46% profit
RRC bought at 10:00AM and sold at 03:45PM would have made a 4.38% profit
LYB bought at 09:35AM and sold at 03:30PM would have made a 4.23% profit

Prévision des données boursières :

Méthodes implémentées : moyenne glissante, régression linéaire, k plus proches voisins, auto ARIMA et Facebook Prophet.

Nous avons utilisé les données de Quandl (granularité de 1 jour)

Utilisation du module predict :

– Importer les données depuis un fichier CSV (il doit contenir une colonne Date et une colonne Close)

```
>>> data = predict.prepare_data(nom_du_fichier_csv)
```

– Appliquer une des méthodes de prédiction, les paramètres sont data (issue de prepare_data), startAt (début de la zone à prédire), stopAt (*facultatif*, fin de la zone à prédire), do_rms (*facultatif*, booléen affichant ou non l'erreur quadratique moyenne) et print_plot (*facultatif*, booléen affichant ou non le graphique des données)

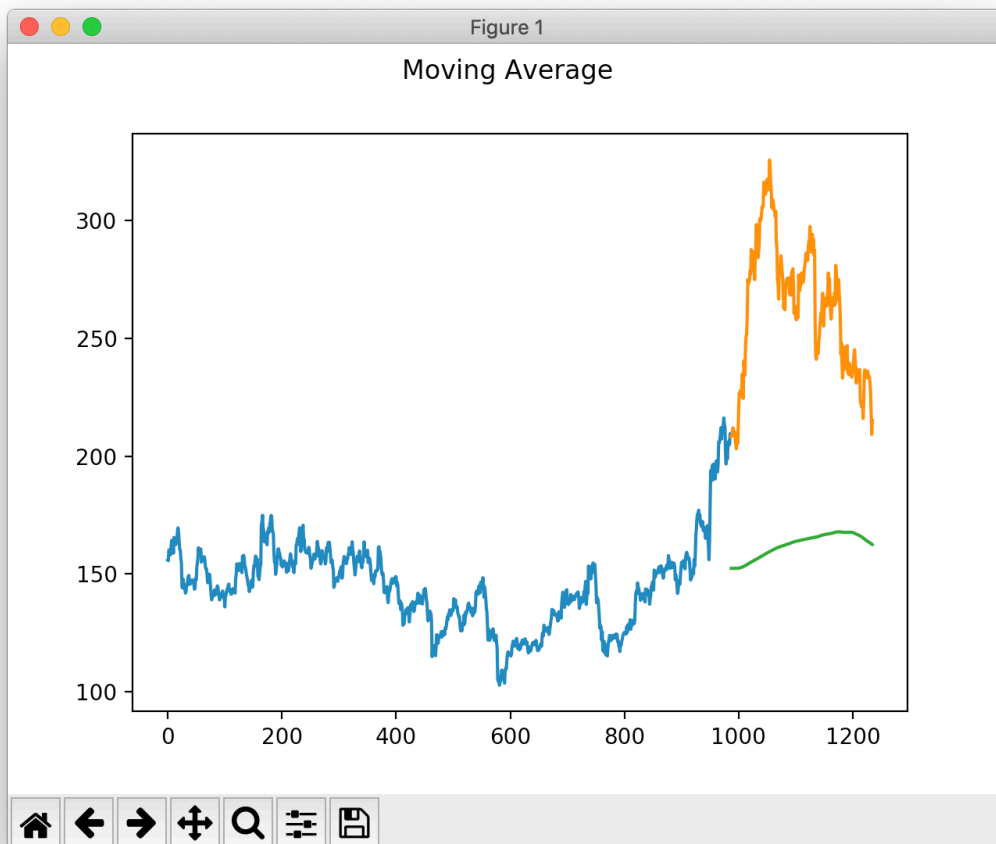
```
>>> predictions = predict.moving_average(data, startAt, stopAt, do_rm, print_plot)
>>> predictions = predict.linear_regression(data, startAt, stopAt, do_rm, print_plot)
>>> predictions = predict.knn(data, startAt, stopAt, do_rm, print_plot)
>>> predictions = predict.arima_auto(data, startAt, stopAt, do_rm, print_plot)
>>> predictions = predict.prophet(data, startAt, stopAt, do_rm, print_plot)
```

Dépendances : numpy, pandas, matplotlib, fastai.tabular, sklearn, pmdarima, fbprophet

Exemple avec la moyenne glissante :

```
MacBook-de-Florent:previsions florentmarmin$ python3
Python 3.7.3 (default, Mar 27 2019, 09:23:15)
[Clang 10.0.1 (clang-1001.0.46.3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import predict
>>> data = predict.prepare_data('NSE-TATAGLOBAL11.csv')
>>> predictions = predict.moving_average(data, 987, print_plot=True)
/Users/florentmarmin/Google Drive/stocks/previsions/predict.py:35: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  data['Predictions'][(len(data)-len(predictions):)] = predictions
```



Nous nous sommes aussi renseignés sur la M4 Competition et les méthodes des Slawek Smyl (mais nous n'avons pas réussi à installer DyNet) et quelques algorithmes de type Bandit Manchot.