Trading Volume Forecasting in the Equity Market using Machine Learning and Statistical Models

by Francois Le Dain

Trading Volume Forecasting in the Equity Market using Machine Learning and Statistical Models

Francois Le Dain (CID: 01565256)

Department of Mathematics
Imperial College London
London SW7 2AZ
United Kingdom

Thesis submitted as part of the requirements for the award of the
MSc in Mathematics and Finance, Imperial College London, 2018-2019

Declaration

The work contained in this thesis is my own work unless otherwise stated.

Signature and date: **September 4th 2019**

[illegible]

Acknowledgements

I would like to thank Deutsche Bank's electronic equity team for providing me constant helps and support, particularly Lars Boyde, Yuan Shen, Sacha Drevet and Torsten Schoeneborn have been a great help for insightful comments and discussions.

Also, I would like to express my gratitude to my project supervisor Mikko Pakkanen for providing guidance whenever needed.

Contents

1 Introduction

1.1 Motivation

Volume forecasting is an important topic for buy-side as well as sell-side firms that have a long-term and short-term vision for their trading strategies. Specifying the forecasting period is important when talking about volume forecasting: for instance, it is possible to forecast the volume that will be traded in the next minute (short time frame), or in the next hour (mid time frame). What interests us the most in this paper is to predict the daily volume *i.e* the volume that will be traded during the next day (long time frame). Due to the large number of different stocks in the equity market, some very simple methods have been developed to help forecast the daily volume. One of the easiest methods is to take the average daily volume of the last twenty days as a prediction for the next day. However, to be successful in an increasingly competitive market, firms have to improve their models to stay competitive. Therefore, volume forecasting using machine learning and sophisticated statistical techniques is of great interest and practical use for those companies.
Volume forecasting, as well as stock price forecasting, are part of a larger machine learning framework called time-series modelling. Time-series modelling is becoming increasingly important with more research being conducted in this specific area. Time-series have often been neglected in the past due to the time component that makes it so different from other kinds of data. Different techniques have been developed to handle time-series, from the simplest ones which only use the past values of a time series, to more complex ones such as random forest and stacking.
Concerning time-series analysis, it is essential to take into account the data's natural temporal ordering. This point is a key challenge for data scientists when they want to accurately train their model by taking into account that each data point holds a memory. Therefore, it is essential to ensure that only past informations is used to forecast the future.

1.2 Purpose and Research Question

In this thesis, different machine learning and statistical techniques will be used to predict the daily volume (long time frame). Furthermore, a different model will be used to update the initial prediction using hourly intra-day information (mid time frame). For both of these time domains, we address two main research questions:

- How can the seasonality and trend of a time-series be taken into account in the machine learning model?
- How can the machine learning model be trained efficiently, whilst taking into account the natural temporal ordering of the data?

1.3 Approach and Methodology

This thesis focuses on using different machine learning and statistical techniques in order to forecast the full-day volume that will be traded in the next day. There are three different parts that will be described within this thesis. Firstly, we examine volume prediction for only one stock: 'BP.L' (an oil and gas company) will be studied to find appropriate methods, models and metrics to use. Secondly, the results found in the first part will be generalized to 45 of the largest stocks in the FTSE 100 stock index (share index of the 100 companies listed on the London Stock Exchange with the highest market capitalisation). Finally, a new model will be created using the intra-day information of the current day in order to update the initial daily volume forecast of that day.

In this thesis, existing methods applied in time-series analysis and in machine learning will be used as well as very new methods. First, feature engineering techniques will be performed, meaning that new features will be created while others will be dropped. This step also contains the normalization of the features as well as investigating stationarity properties. Then, feature selection will be achieved, using different techniques to find the best subset of features to employ. Finally, we will compare different machine learning models with different cross-validation techniques to find the best models and hyper-parameters to use.

1.4 Scope and Limitation

In order to ensure that this thesis can be completed in time, some reasonable limitations were set, after some initial exploration. Thus, our work will focus on:

- This work focuses on linear regression, boosting and stacking machine learning methods. Other techniques such as Recurrent Neural Networks (LSTM) or Vector Autoregressive Moving-Average (VARMA) are beyond the scope of this project.
- This work will focus on typical trading days whereas special days (Half trading days, MSCI rebalancing days, Quarter results days, etc.) will be removed from the dataset and will not be studied.
- This work will focus only on 45 of the largest stocks in the FTSE 100.

1.5 Outline

The following chapter will address the theoretical background concerning this work, focusing on the different types of cross-validation techniques used in time series analysis, the machine learning methods and the metrics used to evaluate the models. After studying a single stock ('BP.L') in chapter 3, we will generalize the optimal techniques found for a single stock to the 45 largest stocks in the FTSE 100 in chapter 4. Chapter 5 will focus on updating the daily volume forecast with intra-day information. Chapter 6 will discuss the results, examine the reliability of our models and critically review the approaches taken. Finally, chapter 7 will summarize the results and give an outlook for possible future research to expand on topics discussed in this thesis.

2 Theoretical Background

In this section, we describe the relevant theoretical foundations in machine learning. The following questions will be answered:

- What is Cross-Validation and which kind of Cross-Validation techniques are used in Time-Series analysis?
- How to evaluate the stationarity of a time-series, and how to convert a non-stationary time-series into a stationary time-series?
- Which machine learning methods are widely used in time-series analysis and how do some of them work (Elastic Net, Random Forest, Stacking)?
- How to perform stock market clustering?
- Which metrics are used to best quantify the quality of predictions?

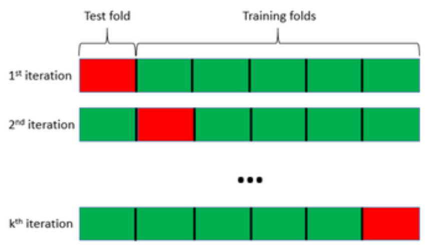
2.1 Time-Series Cross-Validation

2.1.1 Cross-Validation

The idea behind Cross-Validation is to evaluate the performance of any machine learning model and evaluate wetter the model under-fits or over-fits the data. In Cross-Validation, it is essential to keep aside a subset of data which is not used to train the model. A widely used algorithm for Cross-Validation is called K-Folds [1]. This algorithm follows the following steps:

1. Divide the sample data into k-parts.
2. Use k-1 of the parts for the training, and 1 for testing.
3. Repeat the procedure k times, rotating the test set.
4. Determine an expected performance metric based on the results across the iterations.

Figure 1: k-fold Cross-Validation.



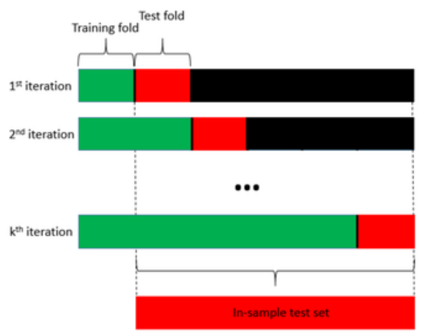
This cross-validation is always performed in-sample and the results is an average score over each test fold. The test folds have to be confront with the out-of-sample test set.

The main drawback of the Cross-Validation technique is that it does not take into account the natural temporal ordering of the data. As a result of this, some Cross-Validation techniques for Time-Series have been developed. The Nested Cross-Validation and the Walk-Forward Cross-Validation are two of them.

2.1.2 Nested Cross-Validation

Nested Cross-Validation uses an expanding window in order to train the machine learning model, and then it uses a test fold with a fixed window size. This algorithm can be pictured as follows:

Figure 2: Nested Cross-Validation.



This cross-validation is always performed in-sample and the results is the score obtained using the in-sample test set. The in-sample test set has to be confront with the out-of-sample test set.

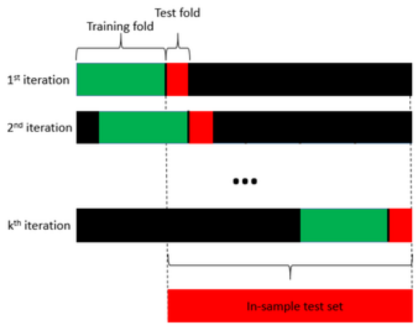
The main drawbacks of this method are:

- The data in the first training fold are not tested.
- In the first iteration, the data might be under-fitted and in the last iteration, the data might be over-fitted. This could lead to poor predictions in the test fold.

2.1.3 Walk-Forward Cross-Validation

The Walk-Forward Cross-Validation technique uses a rolling window with a fixed size of training fold and test fold. This method prevents under-fitting and over-fitting, however it could be difficult to find an optimal size of the training fold and test fold.

Figure 3: Walk-Forward Cross-Validation.



This cross-validation is always performed in-sample and the results is the score obtained using the in-sample test set. The in-sample test set has to be confront with the out-of-sample test set.

The Walk-Forward Cross-Validation will be used as a Cross-Validation technique in this paper because it is a dynamic technique that can avoid under-fitting and over-fitting, given a well chosen size of training and testing window.
Let us define the function $WF(features, training\ size, testing\ size, model, metric)$ which performs the Walk-Forward Cross-Validation using a list of features, a specific size for the training and testing windows, the machine learning model used (usually a linear regression) and the metric used. The main metric used in this paper is the R^2-score but the scores obtained with the mean absolute error (MAE) and the weighted asymmetrical logarithmic error (ALE) will also be exposed. The metrics used will be based on the predicted and true values across all the complete test fold.

2.2 Stationarity and Fractional Differentiation

2.2.1 Stationarity

Usually, time-series forecasting models require a certain time consistency named *stationarity*.
A time-series is stationary if its statistical properties such as mean, variance, covariance and dependence structure are invariant under translation in time. For example, a Time-Series should not exhibit a trend over time. There are numerous ways to check the stationarity of a Time-Series:

1. Visually inspect the line plots over time for a pronounced trend.
2. Inspect the autocorrelation plot: slow decay in the auto-correlation plot is often an indication of non-stationarity.
3. It applies test for unit roots. The most well known test for unit roots is the Augmented Dickey Fuller (ADF).

The part 2.2.2, we will focus on the last method, the Augmented Dickey Fuller test. It tests the null hypothesis: "existence of a unit root of some order" (which implies the non-stationarity of the time-series) against the alternative hypothesis of stationarity.
It is straight-forward to prove that the existence of unit root implies non-stationarity of the series with an AR(1) process. However, the general case is more complicated:

Example. The time-series y_t has a unit root test, that is: $y_t = y_{t-1} + \epsilon_t$ where $\epsilon_i \overset{i.i.d}{\sim} \mathcal{N}(0, \sigma^2)$. Then $y_t = y_0 + \sum_{i=0}^{t} \epsilon_i$ and $\text{Var}(y_t) = t \times \text{Var}(\epsilon_i) = t \times \text{const}$. Thus, the variance is time dependent and therefore the time-series is non-stationary. example

2.2.2 Fractional differencing

The classic transformation of non-stationary data to stationary data is done via integer order differencing, for example, modelling returns instead of absolute prices. This method eliminates memory in the data and therefore negatively affects the predictive power of the model. As a result of this, fractional differencing is a tool used to better balance a desire for stationarity and keeping meaningful memory [2].
Let $\mathbf{B}$ denote the lag operator, i.e $\mathbf{B}(X_t){=}X_{t-1}$ for $t > 1$ and a time-series $\mathbf{X} = \{X_1, X_2, ..., X_n\}$. Then, we can define the element-wise differencing of first order with the identity operator $\mathbf{I}$ as $(\mathbf{I} - \mathbf{B})(X_t) = X_t - X_{t-1}$

Remark 2.1. Polynomials of the Lag operator **B** are the repeated applications, e.g.
$\mathbf{B}^2(X_t) = X_{t-2}$

Therefore, it is possible to expand the time-series using Binomial coefficients and an order of differencing $d \in [0,1]$:

$$(1-B)^d = \sum_{k=0}^{\infty} \binom{d}{k} (-B)^k = 1 - dB + \frac{d(d-1)}{2!} B^2 - ...$$

From this derivation, we can read out the iterative formula for the weights of the lags:

$$\begin{aligned} & w_k = (-1)^k \binom{d}{k} \text{ and } w_{k-1} = (-1)^{k-1} \binom{d}{k-1} \\ & \text{so, } \frac{w_k}{w_{k-1}} = -\frac{\binom{d}{k}}{\binom{d}{k-1}} = -\frac{d!\ (k-1)!\ (d-k+1)!}{d!\ k!\ (d-k)!} \\ & \text{and, } w_k = -w_{k-1} \times \frac{d-k+1}{k} \end{aligned}$$

where w_k is the coefficient of the Lag operator $\mathbf{B}^k$ and $w_0 = 1$.
As a result of this the initial non stationary time-series $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ may potentially be transformed into a stationary time-series $\mathbf{X}' = \{X'_1, X'_2, \ldots, X'_n\}$ where:

$$\begin{aligned} X'_i &= w_0 \mathbf{B}^0(X_i) + w_1 \mathbf{B}^1(X_i) + \cdots + w_N \mathbf{B}^N(X_i) \\ &= w_0 X_i + w_1 X_{i-1} + \cdots + w_N X_{i-N}. \end{aligned}$$

Here, N is the number of lags that are kept. In theory, N can be very large, but in practice N is around 5 because $\forall j \geq 5, w_j \simeq 0$.
In this paper, fractional differencing will particularly be used to transform the VWAP [1] time-series data, which is a non-stationary time-series, into a stationary time-series (see Fig.11 for an example). The method used to discriminate between non-stationary and stationary time-series is the ADF test. This test does not guarantee stationarity but it gives strong indication concerning the stationarity or not of a time-series.

2.3 Machine Learning Models

This section describes the machine learning models that will be used in the thesis. Description of some models such as Ordinary Least Squares (OLS), LASSO, Ridge and Elastic Net are given in Appendix A.

2.3.1 Regression Tree and Random Forest

First, let us describe the regression tree algorithm which is the main algorithm used in the Random Forest algorithm (for regression). Basically, a regression tree is a tool that uses a tree-like model

[1] Volume-weighted average price (VWAP) is defined as: $VWAP = \frac{\sum_i^{allexec.} v_i p_{t_i}}{\sum_i v_i}$, where v_i is the volume traded during the execution at time t_i and p_{t_i} the associated price.

of decision, where: a node represents a "test" on a feature; each branch represents the outcome of the test; and each leaf node represents a value (for the regression).
The basic idea of the learning procedure is the following:

1. Recursively split the learning sample with a test based on the inputs, trying to minimize the variance of the output. The predictions at the leaf nodes are computed as the mean of the output of the leaf nodes.

2. Stop when the output is a constant leaf (or some other stopping criterion is met).

The idea behind the random forest algorithm is to use different regression trees and then to aggregate the results. In general, this reduces the variance of the predictions, and help prevent over-fitting. The procedure is the following :

Algorithm 1: Random Forest Algorithm

```
Input: F a list of features of size p; $S_{train}$ the training set; N the number of decision trees
  learned; and m instance randomly drawn with replacement for each tree.
for $i = 1$ to $N$ do
    $S_{train}^{(i)}$ = m instance randomly drawn with replacement from $S_{train}$;
    $h_{tree}^{(i)}$ = randomized decision tree learned from $S_{train}^{(i)}$. It is randomized because at each
      node, f features from F are selected (without replacement) with f $\ll$ p. Then the best
      split is chosen;
end
Result: $H^N = \frac{1}{N} \sum_i h_{tree}^{(i)}$
```

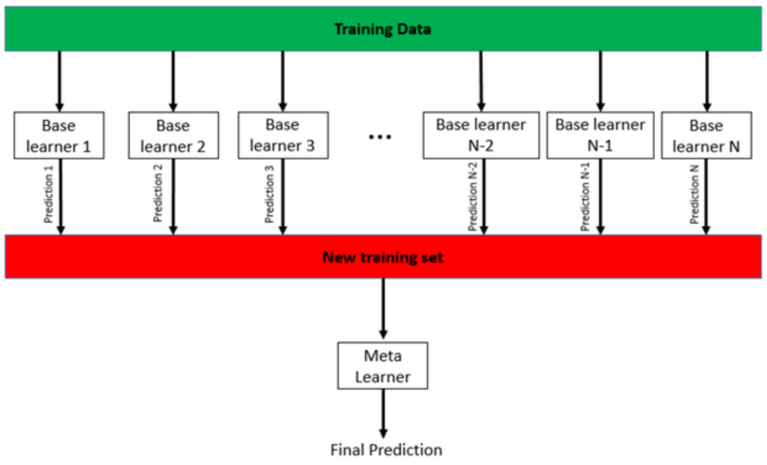
2.3.2 Stacking

Stacking is an ensemble machine learning technique that combines multiple regression models via a meta-regressor. The base level models are trained based on a complete training set, then the meta-regressor is trained on the outputs of the base level model as features [3]. The stacking algorithm is summarized below :

Algorithm 2: Stacking Algorithm

```
Input: Training data $D = \{x_i, y_i\}_{i=1}^{m}$;
Do:
 - Learn N base-level regressors $h_1, h_2, .., h_N$ based on $D$;
- Construct new data set based on predictions of the base models:
 $D_h = \{x'_i, y_i\}$, where $x'_i = \{h_1(x_i), ..., h_N(x_i)\}$;
- Learn a Meta-Regressor M based on $D_h$;
Result: M
```

Figure 4: One-level stacking example.



2.4 Visualization of High-Dimensional Datasets and Cluster Analysis

When studying different stocks, it could be useful to perform cluster analysis in order to use information from similar stocks to forecast the volume of a given stock. The most basic clustering technique would be to cluster the stocks with respect to their industry. In this paper, two other techniques are used:

1. Principal Component Analysis (PCA) to visualize the stocks in 2D and then perform a K-means algorithm.
2. t-Distributed Stochastic Neighbor Embedding (t-SNE) to visualize the stocks in 2D and then perform a K-means algorithm.

2.4.1 t-Distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE is a machine algorithm used for dimensionality reduction and visualization [4]. This non-linear reduction technique is well suited for embedding high-dimensional data for visualization in a low-dimensional space of two or three dimensions. This technique has three main computational steps:

1. Performs the fitting of a probability distribution over pairs of high-dimensional objects, such that similar objects have a high probability of being picked and dissimilar objects have a very small probability of being picked.

   Given a set of N high-dimensional objects $x_1, x_2, .., x_N$ the probabilities p_{ij} that are proportional to the similarity of objects x_i and x_j are computed as follow:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N},$$
$$\text{with } p_{j|i} = \frac{\exp(-\left\|x_i - x_j\right\|^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-\left\|x_i - x_k\right\|^2/2\sigma_i^2)},$$

2. Then, the algorithm computes a probability distribution over the points in the low-dimensional space.

   Consider a d-dimensional space with the objects $y_1, y_2, .., y_d$ with $d \ll N$. To reflect the similarities p_{ij} as well as possible, t-SNE measure the similarities q_{ij} between two objects y_i and y_j using a very analogous approach : [2]

$$q_{ij} = \frac{\left(1 + \left\|y_i - y_j\right\|^2\right)^{-1}}{\sum_{k \neq i} \left(1 + \left\|y_i - y_k\right\|^2\right)^{-1}},$$

3. Finally, the algorithm finds the location of points y_i in the d-dimensional space by minimizing the Kullback-Leibler divergence between the two distribution.

$$KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

2.4.2 Cluster Analysis with K-means Algorithm

In machine learning different types of clustering exist. Some of them are density based (DBSCAN), other are based on the distribution of the data (E-M algorithm) or the connectivity (Hierarchical clustering). In this section, we will describe an algorithm called the K-means algorithm, which aims to partition n observations into k clusters.

The K-means algorithm uses an iterative refinement technique to perform clustering, the algorithm

[2]On `https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding` the expression is given as $q_{ij} = \frac{\left(1+\left\|y_i - y_j\right\|^2\right)^{-1}}{\sum_{k \neq l}\left(1+\left\|y_l - y_k\right\|^2\right)^{-1}}$, I had a different interpretation from this expression and change the l index into i.

proceeds by alternating between two steps :

```
Algorithm 3: K-means Algorithm
  Input: Integer $K > 0$ which is the number of clusters, set $S$ of points in the euclidean
    space;
  Initialization: Select $K$ points in $S$ as the initial centroids;
  while The centroids change do
      Form $K$-clusters by assigning each point to its closest centroid;
      Recompute the centroid of each new cluster;
  end
  Result: Return the updated set $S$ with the label associated to each point
```

2.5 Regression metrics for evaluating predictions

When using a machine learning algorithm, it is necessary to choose a metric to evaluate its performance. The metric used is very important and it influences how the importance of different characteristics in the results is weighted and which machine learning algorithm performs "best". In this section we will talk about the three main metrics that are used in regression and their main advantages and drawbacks [5].

2.5.1 R^2-score

The R^2-score is a goodness-of-fit measure for linear regression models. It is defined by:

$$R^2 = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2}.$$

The R^2 lies in $[-\infty, 1]$, but if the R^2 is negative, then it means that the model predicts worse than $\bar{y}$. Hence, the R^2-score represents the proportion of the variance that is explained .

Advantages:

- The R^2 is easily interpretable if it is compared to a baseline score.

Drawbacks:

- Large values (outliers) can drive the R^2.
- R^2 does not indicate whether the coefficients estimates and predictions are biased.

2.5.2 Mean Absolute Error (MAE)

The MAE is defined by:

$$MAE = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|.$$

Where $\hat{y}_i$ is the prediction and y_i is the true value. As a result of this, the MAE describes the average magnitude of the residuals.

ADVANTAGES:

- Easily interpretable because the error is measured in the original unit of measurement
- Robust to outliers

DRAWBACKS:

- The absolute value of the residuals is not as desirable as squaring this difference because it does not bring more attention to outliers or downplay them.

2.5.3 Weighted Asymmetrical Logarithmic Error (ALE)

When predicting the trading volume, for an execution perception, it is important to take into account the asymmetric risk profile of execution. For example, overestimating the volume can lead to an increase in the participation rate and thus increase the slippage to the VWAP. As a result of this, our metric has to penalize more the over-predictions than the under-predictions. To do this, we can use the Weighted Asymmetrical Logarithmic Error (ALE) [6]:

$$ALE = \frac{1}{N}\sum_{i=1}^{N} w(X_i^{pred} - X_i^{true}) * |X_i^{pred} - X_i^{true}|,$$

where

$$w(x) = \begin{cases} 1, \ if \ x \leq 0 \\ 2, \ if \ x > 0 \end{cases}$$

Thus, ALE is an asymmetrical generalization of L_1 norm in the logarithmic space. This norm uses double weights for over-estimations of the volume and takes into account the asymmetric risk profile of execution.

2.5.4 Root-Mean-Square error (RMSE)

The RMSE is defined by:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}.$$

Where $\hat{y}_i$ is the prediction and y_i is the true value.

ADVANTAGES:

- It is differentiable, so it is easy to calculate or compute the solution.

- It is symmetric and quadratic, which is suitable for Gaussian noises.

DRAWBACKS:

- It is sensitive to outliers, thus an extreme value can completely change the estimate.

3 Study of a single stock : BP

This section will give an overview of the techniques used to perform time-series analysis and predictions over one stock. The results that are obtained in this section are of high importance because they will be generalized to 45 of the largest stocks in the FTSE 100. We will focus our attention on the results obtained for continuous trading. The same work has been done for the daily auction trading and the results obtained are very similar.
The data used in this section are: daily data between 01/01/2015 and 31/12/2018. The main features in the dataset are: daily highest and lowest prices, first and last prices, daily average and median prices, vwap, daily number of trades, daily traded volume (in number of shares), turnover, open and close prices.
In this section, the following questions will be answered in detail:

- Which features are adopted or removed to perform a linear regression?
- How to select the best features?
- Which machine learning model achieves the best results?

3.1 Data Cleaning

First of all, the data needs to be cleaned because some data points correspond to so-called "Special days". During those days, the amount of shares traded do not follow the same pattern as a normal day. Those points are essentially outliers and the date of those special days are well known. For the equity market, examples of special days are:

- Half trading days (e.g. around bank holidays)
- Holidays
- Dividend days
- Quarterly result announcement days
- MSCI rebalancing days

3.2 Feature Engineering

3.2.1 Initial Features

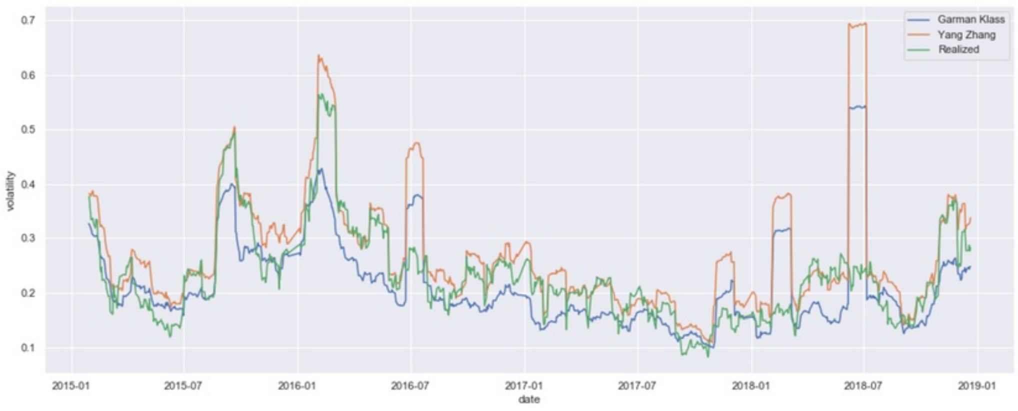
The initial subset of features that is chosen includes: highest and lowest daily price, average daily price, VWAP, average traded size, open and last traded size, open and close price. The daily traded volume in number of shares (which is the target of the machine learning model) is also kept.

3.2.2 Created Features

In this section, we will describe the different features that have been created in order to improve our model. In the next section concerning the volume forecasting with a large universe of stocks,it will be possible to analyse the best subset of features chosen for each stock. Thus, the following features are created:

- Difference between the highest and lowest price of each day.
- Historical volatility (close-to-close): Simplest and most common type of calculation of volatility only uses log-returns at the close and takes into account the number of trading days (see appendix B).
$$\sigma_{historical} = \sum_{i=1}^{N} \left[\log(\frac{c_i}{c_{i-1}}) - \overline{\log(\frac{c_i}{c_{i-1}})} \right]^2$$
- Garman-Klass volatility (see appendix B).
- Yang-Zhang volatility (see appendix B).
- Momemtum of the close price: difference between the close price two days ago and the close price twenty days ago.
- Log difference of the volume: $\log(V_t) - \log(V_{t-1})$.
- Log returns using the close prices: $\log(C_t) - \log_(C_{t-1})$.
- Lowest log returns on the 6 previous days.
- Volume at lag 1: V_{t-1}.
- Volume at lag 2: V_{t-2}.
- Volume at lag 3: V_{t-3}.

Figure 5: Different volatilities for BP.L.

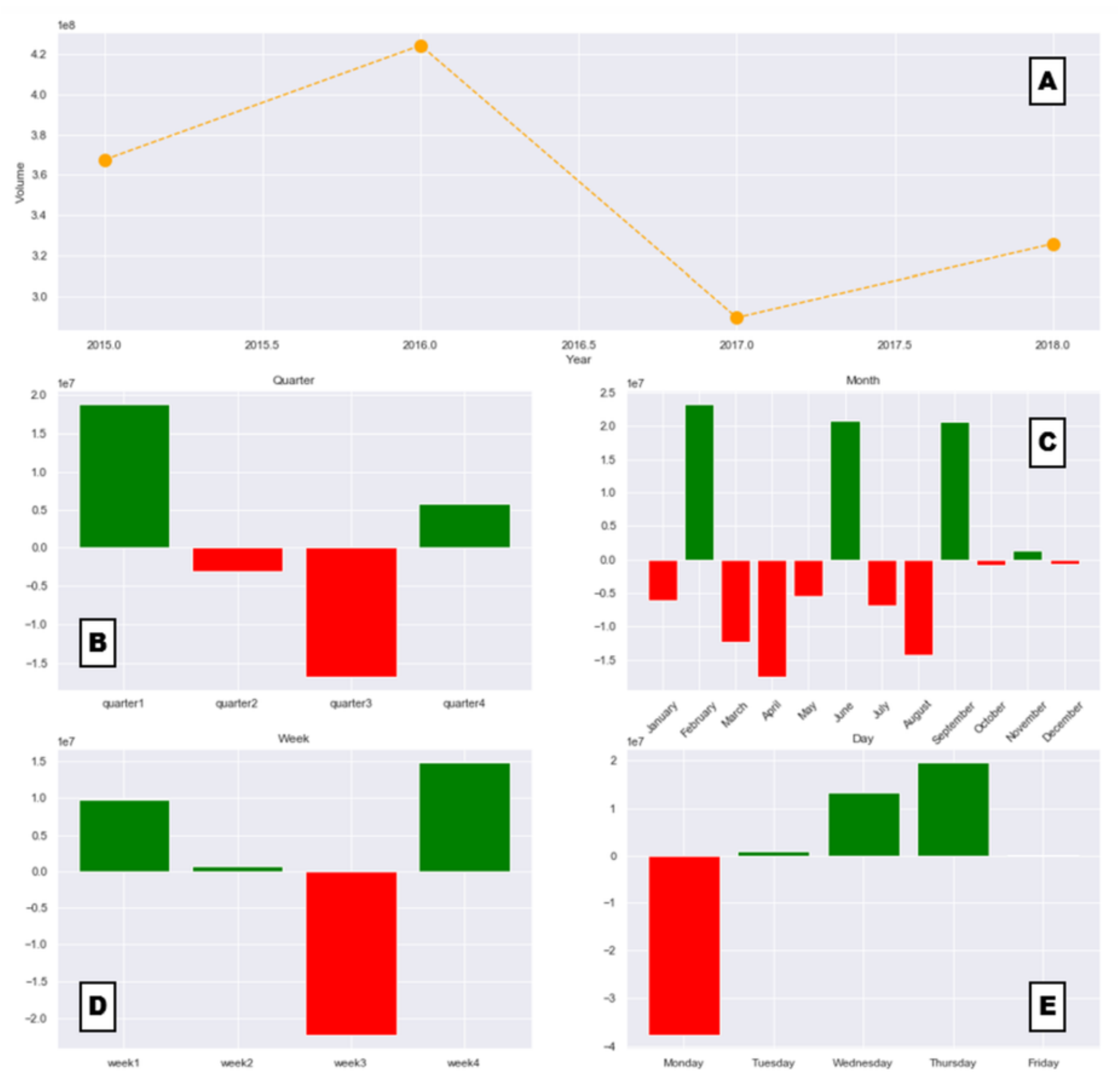


Comparison of the Garman Klass, Yang-Zhang and realized volatility sampled at a daily scale.

3.2.3 Seasonal Adjustment

In this section, we will analyse the impact of seasonality effects over time periods of years, quarters, months, weeks and days on the volume traded (see Fig.6). For example, in the third quarter, much less volume is traded compared with other quarters because there is a long holiday period. To see the impact of each time period, we will proceed as follow: first, the mean of the traded volume per year is computed. Then, for each year we subtract this mean, we are now able to study the impact of the quarter because we have removed the impact of the annual component. We then compute the mean of the volume traded for each quarter and remove this mean for each quarter. We are now able to see the seasonal impact of months, weeks and finally the day of the week. The idea is to remove the volume periods patterns to clearly analyse the impact of the quarters, months, and so on.

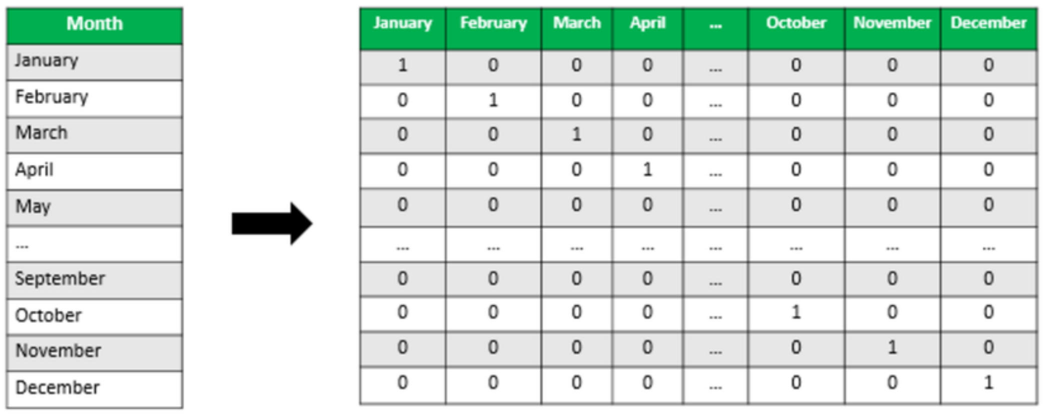
Figure 6: Seasonal adjustment of the volume.



A) Yearly volumes; B) Quarterly volumes; C) Monthly volumes; D) Weekly volumes; E) Daily volumes.

In the graphs above, we notice that some time periods have a large impact on the volume traded. As a result of this, the idea is to create features which can take this impact into account. To do that, we will use "one hot encoding"[7]. This process performs a "binarization" of the categories and includes it as a feature used to train the model.

Figure 7: One-hot encoding.



Month
January
February
March
April
May
...
September
October
November
December

January	February	March	April	...	October	November	December
1	0	0	0	...	0	0	0
0	1	0	0	...	0	0	0
0	0	1	0	...	0	0	0
0	0	0	1	...	0	0	0
0	0	0	0	...	0	0	0
...	...	...	...	...	...	...	...
0	0	0	0	...	0	0	0
0	0	0	0	...	1	0	0
0	0	0	0	...	0	1	0
0	0	0	0	...	0	0	1

example of one-hot-encoding using one feature which represents the months to twelve binary features.

However, it is essential to avoid creating too many features in our model. Thus the idea is to group the time period that have a similar impact on the volume. Therefore the features created are: Quarter 1; Quarter 3; [Quarter2 & Quarter4]; [February & June & September]; [March & April & August]; [January & May & July]; [Week 1 & Week 4]; Week 3; Monday; [Wednesday & Thursday].

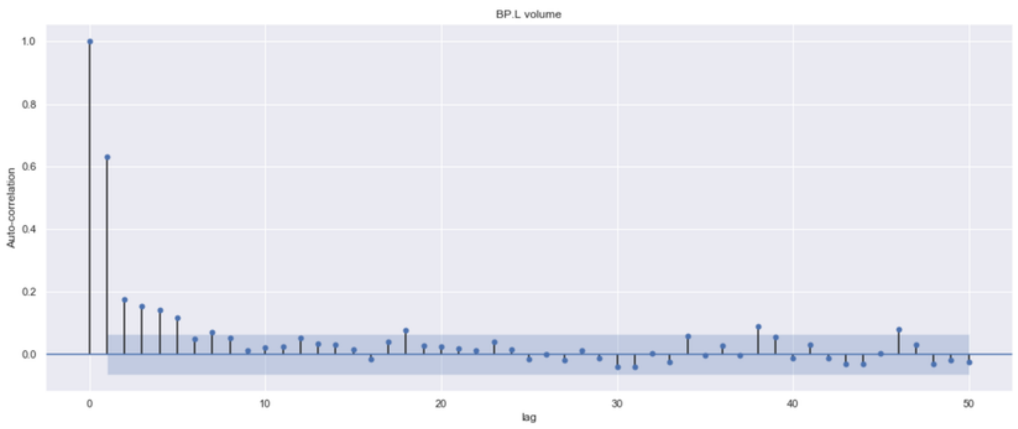
3.3 Stationarity and Normalization of the Features

Many machine learning models assume that data are stationary and normally distributed. This section will show some stationarity results obtained by using different methods presented in Section 2.2. It will also focus on the normality of features and the methods that are used to normalize time-series data.

3.3.1 Stationarity

Stationarity of a time-series is important because stationary processes are easier to analyse. In fact, a stationary time-series repeat some patterns and the results obtained can be reusable. Figure 8 highlights the autocorrelation of the volume. It shows that the volume time-series of BP.L is weakly-stationary because the autocorrelation decays exponentially with only one large autocorrelation value between the lag 0 and the lag 1 of the time-series.

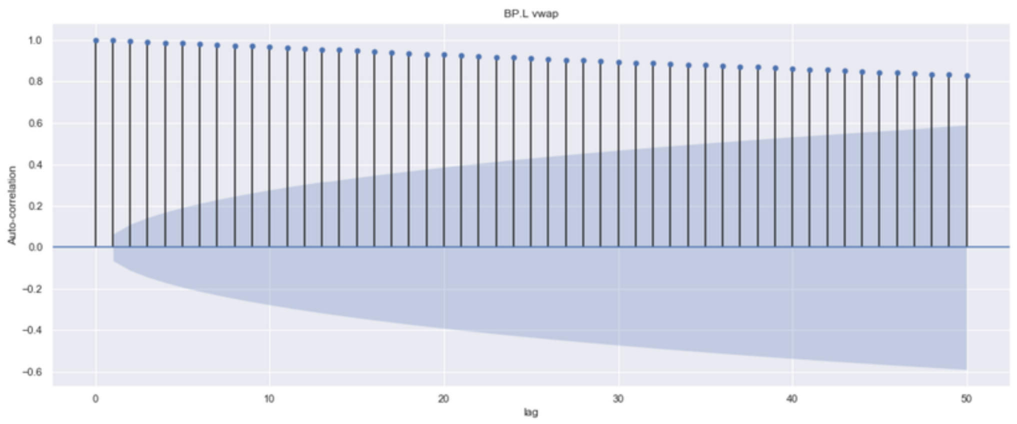
Figure 8: Auto-correlation for the volume of BP.L.



The volume time-series of BP.L appears to be weakly stationary because only the lag 0 and the lag 1 are highly correlated.

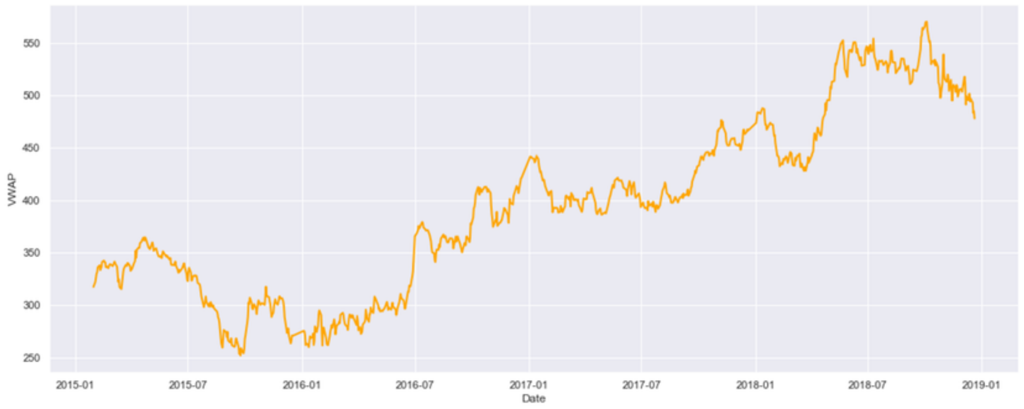
However, by studying the auto-correlation plot for the VWAP (see Fig.9), we can show that it is a non-stationary time-series. Furthermore, by inspecting the line plot over the time (see Figure 10), we can see a pronounced trend in the time-series which also shows that the time-series is not stationary.

Figure 9: Auto-correlation for the VWAP of BP.L.



The autocorrelation plot of the VWAP time-series of BP.L shows a decreasing pattern, it is an indication that the time-series is non-stationary.

Figure 10: VWAP of BP.L.

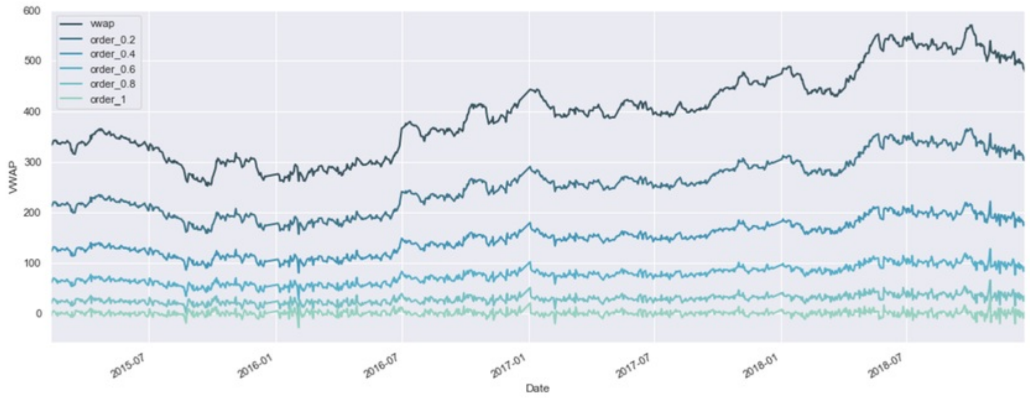


Daily VWAP time-series for BP.L between 2015-02-01 to 2018-20-12. The time-series shows a pronounced trend which suggests that this time-series is not stationary.

To transform the non-stationary VWAP time-series into a stationary one, we can use fractional differentiation. The graphs described in this section show two important aspects:

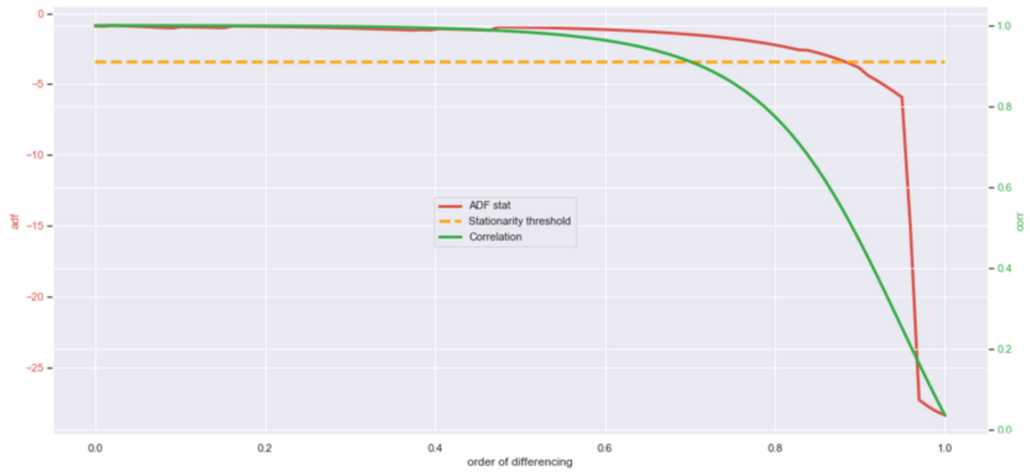
- Fig.11 shows that the higher the order of differentiation, the more stationary is the time-series. In fact, we can observe that the trend is decaying when the order of differentiation is increasing.
- Fig.12 shows both the Augmented Dickey-Fuller (ADF) statistic and the correlation between the initial time-series and the differentiated one with respect to the order of differencing. We can see that the higher the order of differencing, the lower the ADF statistic and the correlation. In other words, this means that the higher the order of differencing, the more stationary is the time-series but the more memory is lost (the lower the ADF statistic, the more stationary the time-series is). Again, the rejection of the null hypothesis in the ADF test is not a guarantee of stationarity but if is a strong indication that a time-series is stationary.

Figure 11: VWAP for different orders of differencing.



The higher the order of differencing, the lower is the trend is and consequently the more stationary is the time-series.

Figure 12: ADF test statistic and correlation for VWAP time series for different orders of differencing (fractional differentiation).



The correlation (green) between the original time-series and the differentiated one is decreasing with respect to the order of differencing. The ADF statistic (red) is also decreasing with respect to the order of differencing. The lower the ADF statistic is, the more stationary is the time-series.

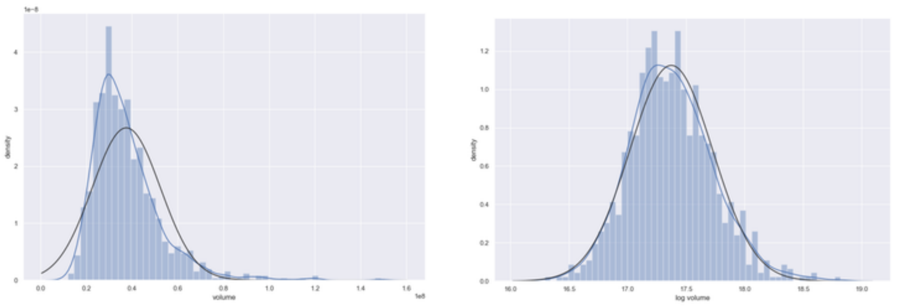
3.3.2 Normalization

Normalization consists of changing the values of numeric columns in the dataset to a common scale, without distorting the differences in the range of values. Data normalization is necessary for machine learning models because some models give more importance to features with higher values. Moreover, The dataset used in this project requires normalization because the features have different ranges. When normalizing a time-series, it is essential to not use future information to normalize previous data. Two main transformations can be used :

- Applying the log function to a time-series. The *log*-transformation is widely used in machine learning to deal with skewed data. In fact, *log*-transformation can decrease the variability of data and make data conform more closely to the normal distribution [8].
- Using a rolling window of size N to compute: (subtract the rolling mean and divide by the rolling standard-deviation.)

$$\tilde{X}_t = \frac{X_t - \frac{1}{N}\sum_{i=1}^{N} X_{t-i}}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(X_{t-i} - \bar{X})^2}}$$

Figure 13: Volume normalization with *log* function



On the left, the volume data are highly skewed. On the right, after having applied the *log* to the volume, we obtain data normally distributed.

3.4 Feature Selection

Feature selection is one of the most important steps in a machine learning project. It consists of choosing the most relevant features among all available features of the model. Different subsets of features can lead to very different prediction scores. A large number of techniques exist to select the best features. For example, it is possible to use a correlation heat-map and to select the features that are the most correlated to the target. It is also possible to use a random grid search for different subsets of features. If there are a very large number of features, a PCA can be more useful.

After comparing the Walk-forward R^2-score obtained for different feature selection methods (Correlation Heat-map, feature importance using a random forest regressor, selectKBest, random grid-search and PCA) the best method seems to be obtained using a bottom-up approach. The algorithm of the bottom-up approach is as follows:

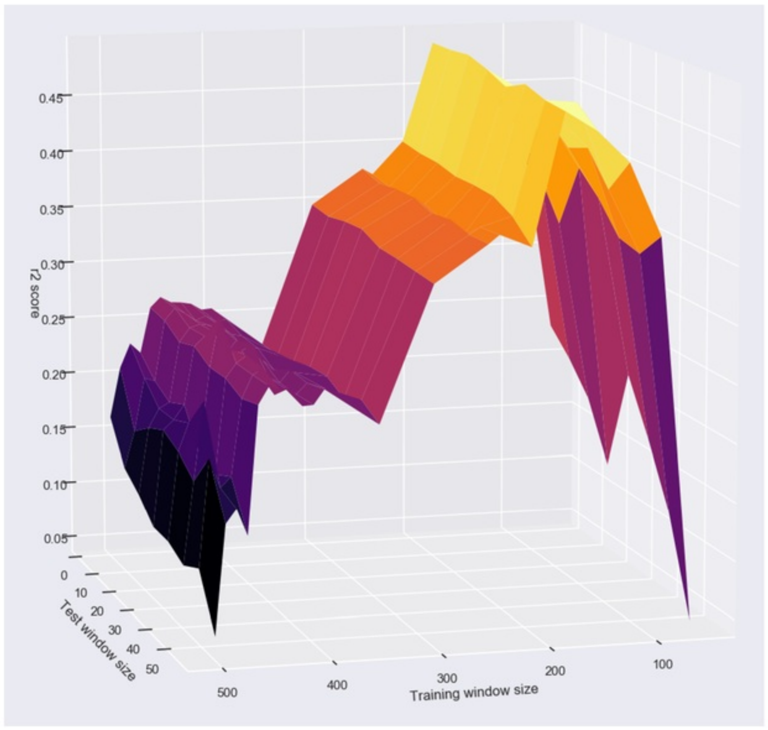
Algorithm 4: Bottom-Up Algorithm

```
Input : Let $F = [f_1, f_2, .., f_n]$ be a list of all the features except $V_{t-1}$ (The one-day
  volume lag) because we consider this feature as the most important one and it is essential
  to include it in the model;
Initialization : Suppose that $V_{t-1}$ is the most important feature;
$B$= $[V_{t-1}]$ is the best subset of features and
  $max_{R^2} = WF(B, N_{train}{}^{a} = 180, N_{test}{}^{b} = 5, linear\ regression, R^2)$ is the best $R^2$-score
  initially obtained;
for $i$ = 1 to $n$ do
    Compute the $R^2$ obtained using the subset of features $\{B\ ,\ f_i\}$;
    if $R^2$ increases then
        The best subset of features is updated : $B = \{B\ ,\ f_i\}$;
        The maximum $R^2$-score obtained is updated :
        $max_{R^2} = WF(B, N_{train} = 180, N_{test} = 5, linear\ regression, R^2)$;
    else
        The feature $f_i$ is dropped;
    end
end
Result: Return the best subset of features $B$ with the associated $R^2$-score
```

[a]Size of the training window
[b]Size of the testing window

Remark 3.1. To find the best training and testing windows sizes, we can compute the R^2-score with the Walk Forward function for a grid of varying training and testing windows sizes. In figure 14 (below), we can clearly see that when the training window size increases beyond a threshold, the R^2-score decreases, which is due to over-fitting. Likewise, when the training size decreases below a certain threshold, the R^2-score decreases, due to under-fitting. Consequently, the best window size for the training fold is around N_{train}=180, and the best window size for the testing fold is unclear so N_{test}=5 will be taken for easier computations.

Figure 14: Walk-Forward R^2-score for different sizes of training and testing windows (using linear regression).



This graph highlights that there exist an optimal size for the training window, but also that beyond or below a certain threshold the R^2-score drops sharply.

Results: The best features selected by the bottom-up algorithm are the following (not in order of importance, see Fig.20 for feature importance.):

- Volume at lag 1: V_{t-1}
- Volume at lag 2: V_{t-2}
- Volume at lag 3: V_{t-3}
- Close Price
- VWAP
- Garman-Klass volatility

- Momemtum of the close price
- Third week of the month
- Friday
- Monday

3.5 Machine Learning Models

In this section the results obtained with different machine learning algorithms will be presented. It is important to keep in mind that the selected features, the optimal windows size for training and testing and other manipulations are done in order to increase the R^2-score. However, the ALE score is more important for financial institutions (e.g Deutsche Bank) for the reasons explained.

3.5.1 In-Sample Test Score

The tab.1 shows the results obtained using the the in-sample test set. Concerning the metrics used, the higher the R^2-score, the better are the results. The lower the MAE and the ALE, the better are the results.

Table 1: In-sample test score for BP.L

Model	**R^2-score**	**MAE**	**ALE**
Avg 20 past days	0.396	0.228	0.341
ARMA(1,1)	0.525	0.207	0.298
OLS	0.554	0.202	0.299
Lasso	0.552	0.202	0.298
Ridge	0.545	0.202	0.309
Random Forest	0.469	0.221	0.317

In-sample test scores obtained for different metrics (R^2-score, MAE score & ALE score) using different models. The baseline scores are obtained using the average of the 20 past days as a prediction.

Concerning the R^2-score, the tab.1 shows that all the models perform better than a simple arithmetic average of the last 20 days. Moreover, among the different models used, the linear regressions such (OLS, LASSO and Ridge regressions) perform better than an ARMA(1,1) or a random forest model in terms of R^2-score.

Concerning the MAE, the arithmetic average of the last 20 days and the random forest models perform worse than other models. The other models such that ARMA(1,1) or the different linear regressions seem to be equivalent.
Finally, concerning the ALE, again the arithmetic average of the last 20 days and the random forest models perform worse than other models. As a result, the linear regressions (particularly the OLS) seem to provide the best results overall. We now need to compare the score on the out-of-sample test set to verify our results.

Remark 3.2. The hyper-parameters used in the different machine learning models (Lasso, Ridge, Random Forest) where obtained using a random grid-search.

3.5.2 Out-of-Sample Test score

The tab.2 exposes the results obtain with the in-sample test set. Again, the higher the R^2-score, the better are the results. The lower the MAE and the ALE, the better are the results.

Table 2: Out-of-sample test score for BP.L

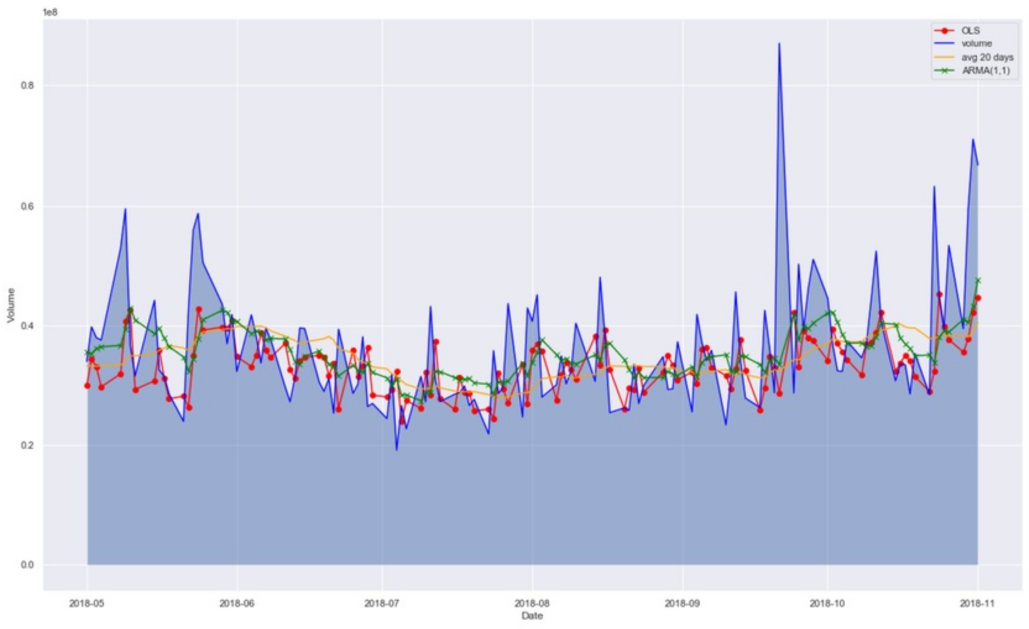
Model	**R^2-score**	**MAE**	**ALE**
Avg 20 past days	0.165	0.192	0.295
ARMA(1,1)	0.212	0.185	0.276
OLS	0.111	0.190	0.321
Lasso	0.089	0.192	0.329
Ridge	0.165	0.186	0.279
Random Forest	-0.016	0.199	0.358

Out-of-sample test scores obtained for different metrics (R^2-score, MAE score & ALE score) using different models. The baseline scores are obtained using the average of the 20 past days as a prediction.

First, the R^2-scores obtained for each model show that the random forest perform worst than taking the average of the time-series as a predictor model. Moreover, the results show that the ARMA(1,1) model provides the best R^2-score and a simple arithmetic average of the last 20 days perform as good as a Ridge regression.
Second, concerning the MAE, all the models seem to provide similar results, however, the ARMA(1,1) appears to be slightly better than the others.
Finally, the ALE shows that the OLS, LASSO and random forest have the worst score. As a result, those models seem to over-predict the daily volume in the out-of-sample test set. The ARMA(1,1)

provides the best scores overall in the out-of-sample test set and the arithmetic average of the last 20 days appears to be a strong model. The differences between the results obtained in the in-sample and in the out-of-sample test sets can be explained by the fact that the out-of-sample test period could be more difficult to forecast because of the volatility of the markets. Moreover, the results obtained in this section concern only one stock, thus we need to compare the scores for a larger universe of stocks. In Sec.4, the scores obtained for 45 of the largest stocks in the FTSE 100 will provide more colour about the best model to use and allow us to obtain more certitude concerning our conclusions.

Figure 15: Volume forecasting for the test set without the special days.



Comparison between the predictions obtained with OLS, ARMA(1,1) and the arithmetic average of the last 20 days.

3.6 Stacking Model

The tab.3 shows the results obtained by averaging the predictions from ARMA(1,1), OLS, LASSO and Ridge (Stacking).

Table 3: Stacking model scores for BP.L

Model	**R^2-score**	**MAE**	**ALE**
Stacking (in-sample test)	0.564	0.199	0.294
Stacking (out-of-sample test)	0.204	0.181	0.290

Scores obtained for different metrics (R^2-score, MAE score & ALE score) using a stacking model composed of ARMA(1,1) model, OLS, LASSO & Ridge regressions.

Concerning the R^2-score, the stacking method obtain very good results compared to other models for both the in-sample and the out-of-sample test sets. The score also obtained for the MAE and the ALE are promising, especially concerning the MAE because we obtained the lowest values of MAE for the two test sets.

Conclusion: This section showed the time-series analysis that has been performed on a single stock. Concerning the result obtained in Sec.3.5, it is difficult to draw general conclusions because only one stock have been studied. However, the ARMA(1,1), the OLS and the stacking models seem to have the best performances. The objective of the Sec.4 is to study a large universe of stocks, namely 45 of the largest stocks in the FTSE 100 stock index to draw clear conclusions concerning the different machine learning models.

4 Generalisation of Analysis from a Single Stock to a Larger Stock Universe

This section will extend the work done in the past section to 45 of the largest stocks in the FTSE 100. Moreover, a study about the clustering of stocks will be performed in order to use information from other stocks to predict the volume of a particular stocks. Again, we will focus our attention on the continuous trading period since the same work has been done for the daily auction trading resulting in very similar outcomes.
The data used in this section are daily data between 01/01/2015 and 12/31/2018 for 45 stocks in the FTSE 100. The main features in the dataset are: daily highest and lowest prices, first and last prices, daily average and median prices, vwap, daily number of trades, daily traded volume (in number of shares), turnover, open and close prices.
The following questions will be answered in detail:

- How to perform cluster analysis and why is it an important topic ?
- How to interpret and utilise the informations provided by the clustering analysis ?
- Which machine learning models achieve the best results for the stock universe ?

4.1 Data Processing

To begin this part, it is important to explain the feature engineering that has been applied for the stock universe explained. As in Sec.3, the data cleaning, the feature engineering and the normalization processes are the same applied for all the stocks studied. However, although the feature selection algorithm remains the same, we will see that each stock has its own best subset of features.
From this point, we consider that our data are processed in the same way as for a single stock (BP.L) in the previous section.

4.2 Cluster Analysis

When studying a large universe of stocks, there are two main approaches: the first one is to consider that all the stocks have the same behaviour. With this approach, cluster analysis is not perform. The second approach (that we will study in this section), is to consider that stocks within a same cluster have similar behaviour and they influencing each other. After performing a cluster analysis, it is important to understand why some stocks are in a same cluster. As a result, it could be useful to compare the stocks with respect to their industry and their liquidity. In fact, it seems reasonable to consider that stocks within the same industry or stocks that are similarly liquid share common behaviour.

Various metrics are used to measure the liquidity of a stock. In this section, we will consider the *Amihud* liquidity measure.

Definition 4.1. *Amihud* liquidity measure

The *Amihud* liquidity measure is defined as [12].

$$LIQ = \frac{1}{\frac{1}{N}\sum_{t=1}^{T}\frac{|r_t|}{\eta_t}}$$

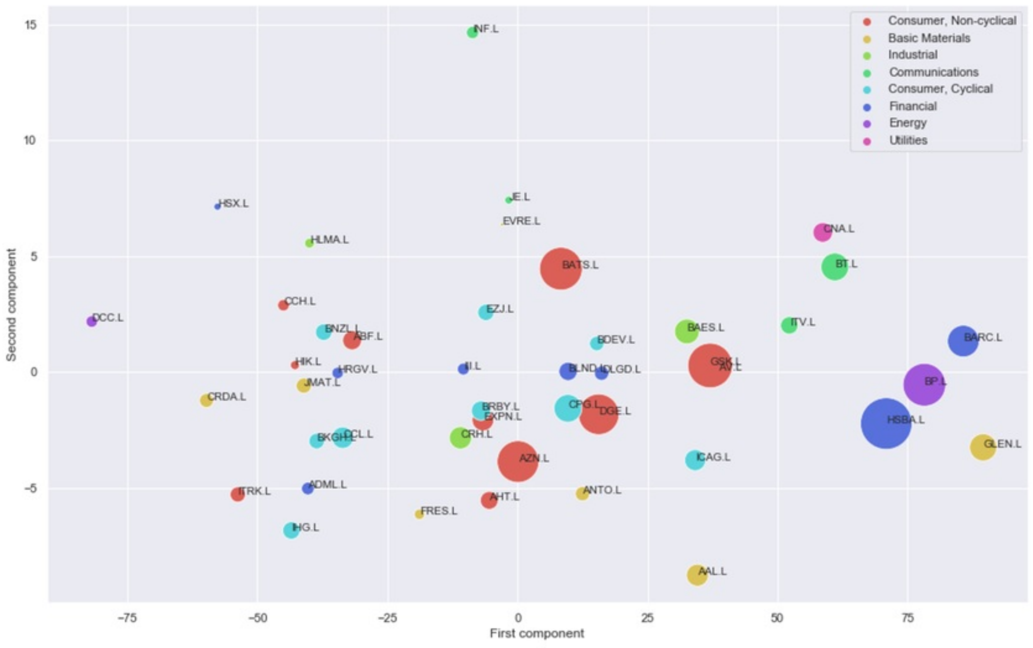
Where T is the number of days, $\eta_t = p_t V_t$ is the notional in pound (i.e price * volume) on day t, and r_t is the return on day t. This measure is mainly used to calculate the degree of liquidity of a stock.

4.2.1 Principal Component Analysis

Principal Component Analysis (PCA) is an unsupervised machine learning technique that performs dimensionality reduction while preserving as much variance as possible. PCA can be useful for visualizing high dimensional data (for example a universe of stocks). In this section, we will use PCA in order to visualize in 2D the 45 stocks studied.then, it will be possible to determine whether the PCA is performing clustering with respect to the industry of each stock or their liquidity, or something different.

Although it is difficult to see different clusters in the Fig.16, we notice that the larger dots tend to be on the right and the smaller ones further to the left. As a result, the first component of the PCA seems to discriminate the stocks regarding their liquidity.

Figure 16: Stocks visualization using PCA with respect to the volume.



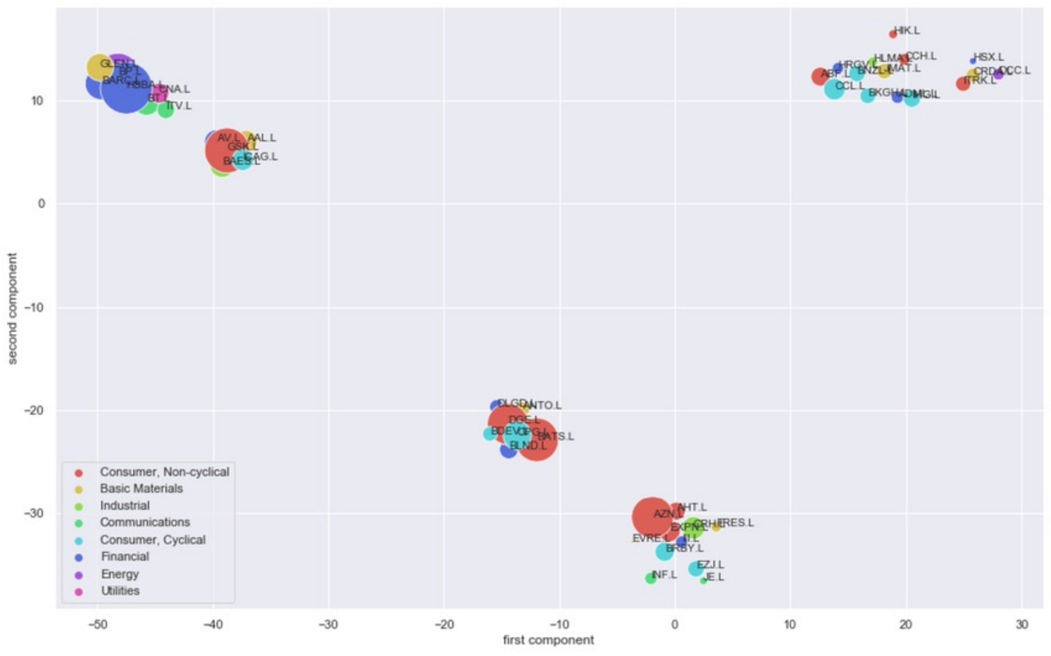
Loading on second component of the PCA with respect to the first component of the PCA. Each color represents an industry and the size of each dot represents the liquidity of the stock (The larger the dot, the more liquid the stock is). The first component of the PCA seems to discriminate the stocks regarding their liquidity.

4.2.2 t-distributed stochastic neighbour embedding and k-Means

As explained in Sec.2.4, t-distributed stochastic neighbour embedding (t-SNE) is a tool that can perform dimensionality reduction. As a result, as PCA, t-SNE can be useful to visualize in 2D the universe of stocks studied (see Fig.17). Then, we will be able to determine if the liquidity or the industry of a stock can have an impact on the cluster analysis. After performing t-SNE algorithm, to visualize the stocks, a k-means algorithm can be used to find the different clusters (See Fig.18). The t-SNE visualization will help us to determine which value of k choose in the k-means algorithm (i.e. the right number of clusters). Indeed, Fig.17 highlights four different clusters, thus the k-means algorithm will take $k = 4$ as parameter.

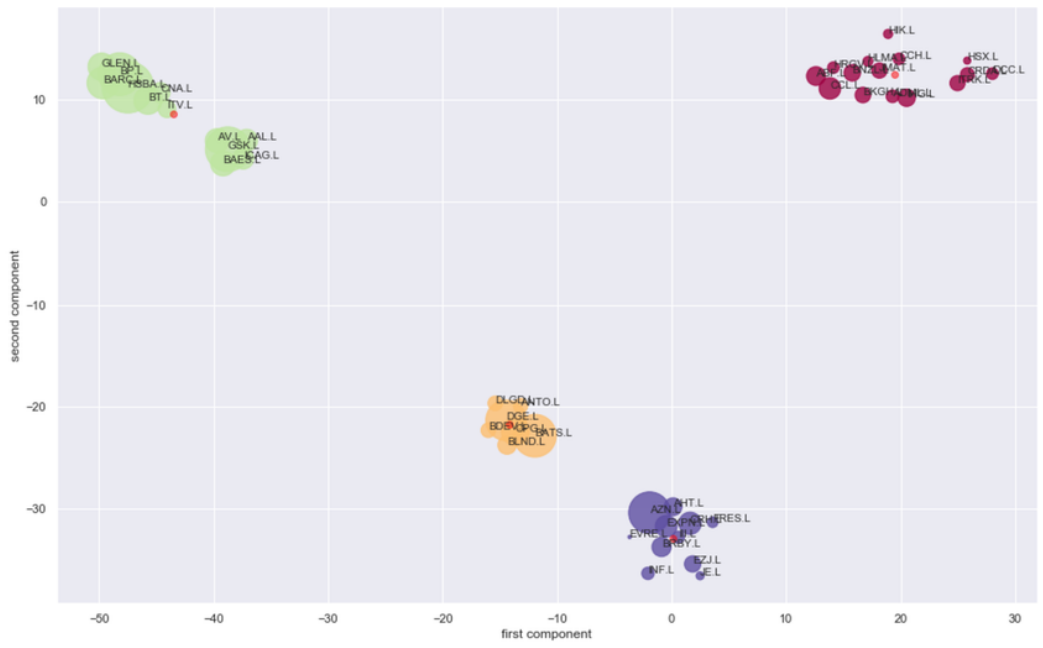
Fig.17 shows that the liquidity of a stock has a huge impact concerning the cluster analysis performed with t-SNE contrary to the industry. In fact, the less liquid stocks are in the top right corner whereas the most liquid ones are in the top left corner.

Figure 17: Stocks visualization using t-SNE.



Visualization of a large universe of stocks using t-SNE. Each color represents an industry and the size of a dot represents its liquidity (the larger the dot, the more liquid the stock is).

Figure 18: Clustering using k-Means.



k-Means algorithm performed after the t-SNE algorithm. The stocks with the same color are in the same cluster.

After having performed both PCA and t-SNE to visualize in 2D a large universe of stock, it is possible to compare those techniques and use the most powerful one. PCA is a stable algorithm that give consistent results and it is easily interpretable. However, it is difficult to see how many clusters are created and what are them. On the other hand, t-SNE is very powerful to create distinct clusters, but the outcome highly depends on the hyper parameters chosen for the t-SNE function.

4.2.3 FTSE 100 Visualization

Another way to visualize high dimensional data in 2D is to visualize the interactions within the data. For example, for a universe of stocks, we can visualize the correlation between the traded volumes of each stock and draw a 2D graph. In python, some tools are very useful to represent the interactions between objects, particularly NetworkX [3]. To obtain an aesthetically-pleasing graph, it is possible to use force-directed graph drawing algorithm [9]. Basically, the purpose of those

[3]NetworkX is a python package for the creation, manipulation and visualization of complex networks.

algorithms is to position the nodes of a graph in 2D (or 3D) so that there are few crossing edges as possible and that the weight of each edge is chosen to minimize the energy of the graph. One of those algorithm is called the *Kamada-Kawai* algorithm.

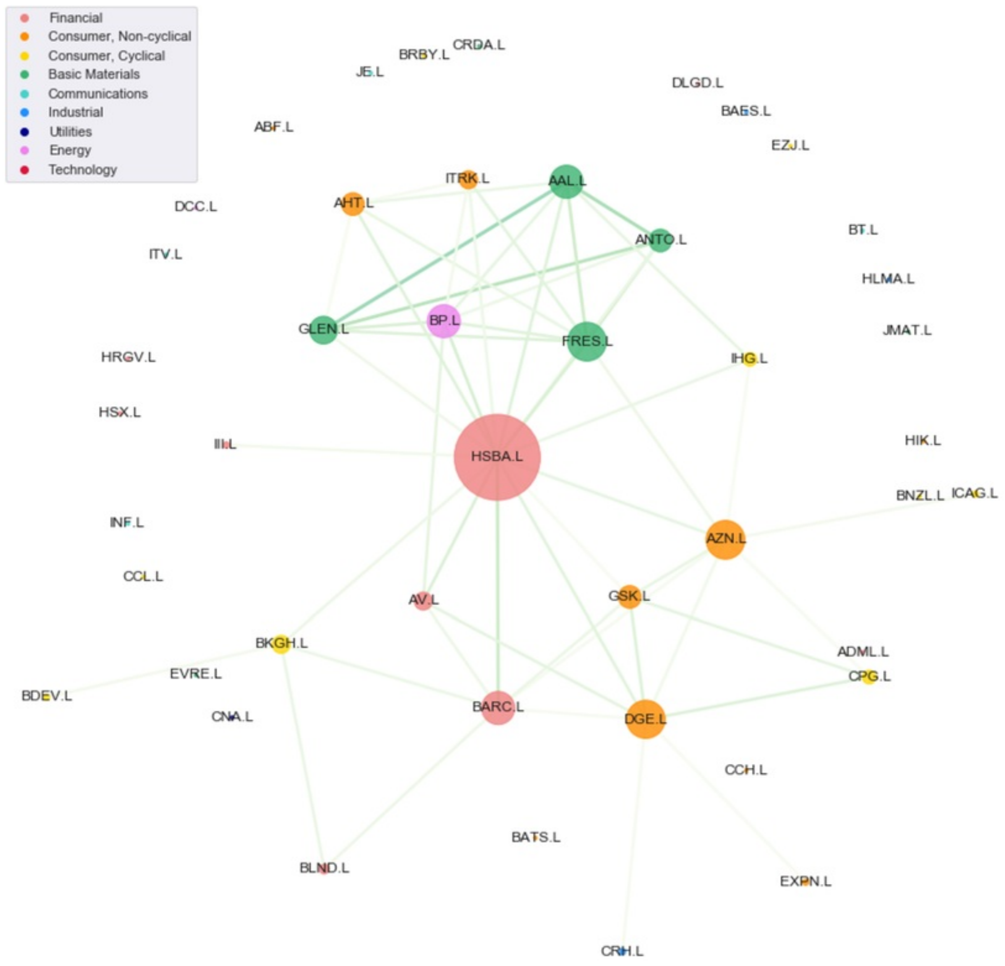
The *Kamada-Kawai* algorithm is used to draw general graphs. Let us imagine that springs are connecting **n** vertices in an on-screen graph. Then, the graph can be viewed as a dynamic system that tries to reach a minimum-energy state, where springs are not stretched or compressed too much from their "relaxed-state length". Then, the energy function to minimize is [10]:

$$
\begin{aligned}
Energy &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{2} k_{ij} (||y_i - y_j|| - l_{ij})^2 \\
\text{with } l_{ij} &= L * d_{ij}, \\
k_{ij} &= \frac{K}{d_{ij}^2}
\end{aligned}
$$

Where d_{ij} is the length of the shortest path from i to j, L is the desired on-screen length of a spring and K a constant. The energy is minimized using a gradient descent method with respect to the y_i.

The Fig.19 is a 2D visualization of the universe of stocks studied using *Kamada-Kawai* algorithm.

Figure 19: Visualization of a large universe of stocks using NetworkX.



This graph highlights the industry of each stock (the color of the dots) and the correlation between stocks (the stronger the color of an edge, the more correlated the connected stocks are). The size of a dot is proportional to sum of the correlations between the stock and the others. Some industries such that basic materials (in green) or non-cyclical consumer (in orange) seem to form very strong clusters in terms of correlation.

In this graph, the edges are weighted by the correlation between stocks. The stronger the color of an edge is, the more correlated the connected stocks are. Moreover, the larger a dot is, the more connected to other stocks the stock is. Thus, we notice that the basic materials companies (GLEN.L, FRES.L, AAL.L & ANTO.L) form a very strong cluster and they are highly correlated between each others. Other non-cyclical consumer stocks are also highly connected between them. Finally, the financial stocks, especially HSBA.L, seem to be highly correlated to all the other stocks of the studied universe. As a result of this, the industry of each stock seems to have an important impact in cluster analysis.

In Sec.4.2, we have seen different techniques to visualize a large universe of stocks in 2D and to perform clustering. The Sec.4.2.3 highlights that it could be very powerful to cluster stocks using their industries. This method is simpler, more consistent and more intuitive than performing a PCA or a t-SNE algorithms and then k-Means algorithm. We will now consider that the stocks are clustered using their industries.

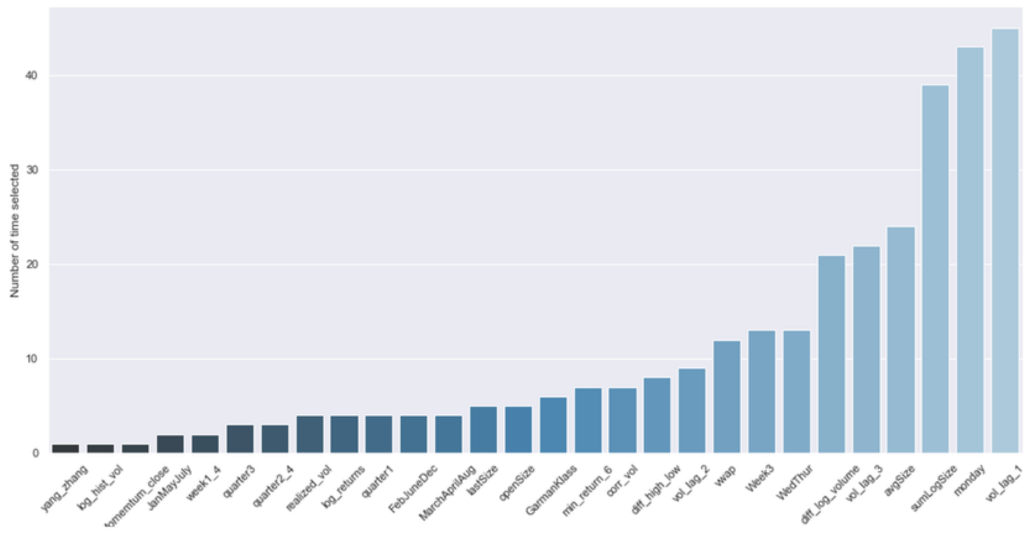
4.2.4 Feature Engineering

The goal of the previous parts (Sec.4.2.1, 4.2.2 & 4.2.3) was to cluster similar stocks to use that information in order to predict the traded volume of another related stock. The idea is to use the average of the traded volume from stocks in the same cluster as a new feature. Consequently, each stock will have as a new feature, the average of the traded volume of the stocks in the same industry as itself.

4.3 Feature Selection

The algorithm used for automatically perform feature selection over the stocks studied is the same as described in the Bottom-up approach (see Sec.3.4). As a result of this, each stock has a certain best subset of features and it is possible to analyse which features are picked most frequently. The features directly related to the volume (one-day lagged volume, cumulative sum of the log size, average traded size, etc.) have a more important impact than the features related to the price or volatility. Finally, the temporal features such weekdays (e.g. Monday or Wednesday & Thursday) are very important features and have a huge impact on the volume traded (as seen in Sec.3.2.3).

Figure 20: Number of times each feature is picked on 45 simulations.



The most picked features (consequently the most important ones) are related to the volume (e.g. one-lag daily volume, average size of the trades etc.) or related to the seasonality (e.g. Monday or Wednesday & Thursday).

After having performed a feature selection for each stock, we can apply a machine learning model to forecast the one-day ahead volume that will be trade for all the stocks in the universe studied.

4.4 Machine Learning Models

In this section, the results obtained with different machine learning algorithms will be presented. It is important to keep in mind that the selected features, the optimal windows size for training, and testing and other manipulations are done in order to increase the R^2-score (or similar relevant metrics).

4.4.1 In-sample Test Score

The table 4 displays the results obtained for the in-sample test set across the full stock universe:

Table 4: In-sample average test score for 45 of the largest stocks in the FTSE 100.

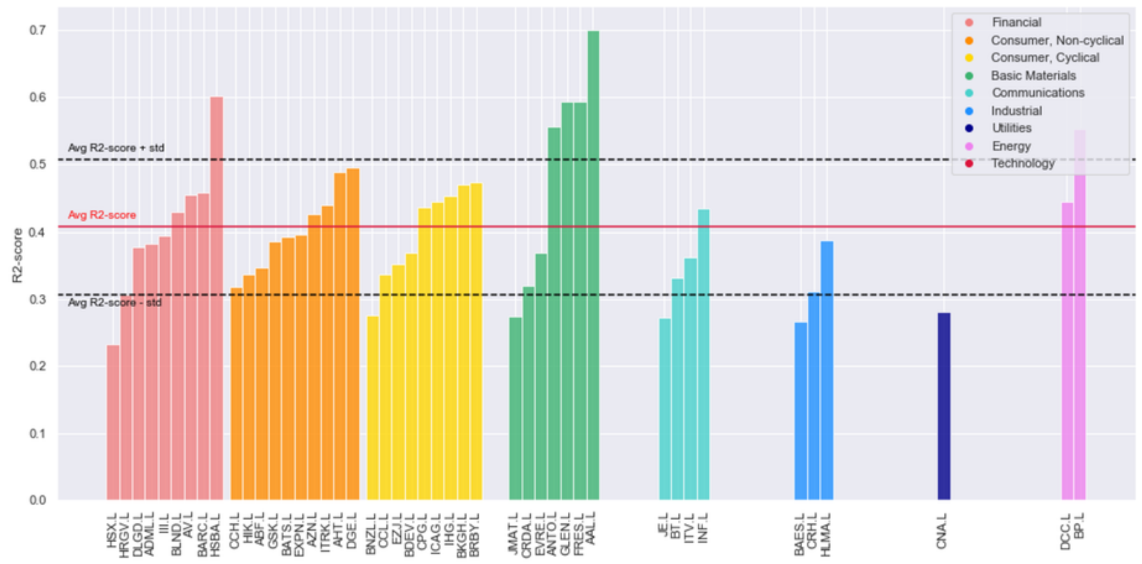
Model	**avg R^2-score**	**avg MAE**	**avg ALE**
Avg 20 past days	0.229	0.281	0.421
ARMA(1,1)	0.369	0.252	0.376
OLS	0.407	0.245	0.365
Lasso	0.407	0.245	0.364
Ridge	0.383	0.249	0.374
Random Forest	0.318	0.264	0.390

In-sample test scores obtained for different metrics (R^2-score, MAE score & ALE score) using different models. The baseline scores are obtained using the average of the 20 past days as a prediction.

Concerning the R^2-score obtained, the linear regressions and particularly the OLS and the LASSO perform better than the other models for the in-sample test set. Moreover, it is important to notice that the OLS highly improve the benchmark model (the arithmetic average of the 20 last days) in terms of R^2-score.

Then, the lower average MAE and ALE are also obtained using a linear regression and more particularly the OLS and the LASSO. Both scores indicate that the OLS and the LASSO are performing very well compared to the other models. Moreover, the arithmetic average of the last 20 days model obtains the worst score among all the other models. This show that, even if it is a simple model, it is not as strong as linear regression or a random forest algorithm.

Figure 21: R^2-score for each stock with OLS (in-sample test set).



This graph highlights that our model can predict very well some basic materials stocks (ANTO.L, GLEN.L, FRES.L and AAL.L) and HSBA.L compared to others. This can be due to the fact that they are part of a very strong cluster (for the basic materials) or highly correlated to others stocks in the FTSE 100 (for HSBA.L). (See Fig.19)

From Fig.21, we notice that the basic materials companies, especially ANTO.L, GLEN.L, FRES.L and AAL.L are very well predicted by our model. Moreover, those four stocks form a strong cluster when using a graph representation (see Sec.4.2.3). The same idea holds for HSBA.L which is a stock highly correlated with other stocks in the FTSE100 and which is highly predictable.

We can now compare the scores obtained for the out-of-sample test set to confirm that the linear regressions are the models that have the best performances.

4.4.2 Out-of-Sample Test Score

The table below lists the results obtained within the out-of-sample test set across all the studied universe of stocks:

Table 5: Out-of-sample average test score for 45 of the largest stocks in the FTSE 100.

Model	**avg R^2-score**	**avg MAE**	**avg ALE**
Avg 20 past days	0.173	0.264	0.404
ARMA(1,1)	0.324	0.250	0.373
OLS	0.293	0.241	0.378
Lasso	0.297	0.239	0.377
Ridge	0.276	0.245	0.369
Random Forest	0.224	0.254	0.401

Out-of-sample test scores obtained for different metrics (R^2-score, MAE score & ALE score) using different models. The baseline scores are obtained using the average of the 20 past days as a prediction.

First, concerning the R^2-score, the results show that the ARMA(1,1) model provides the best R^2-score and the OLS and LASSO models obtain the best scores for the linear regressions. Moreover, all the models used have better performance than the arithmetic average of the last 20 days (our baseline score).

Second, concerning the MAE, all the models seem to provide similar results, however, the LASSO and the OLS appear to be slightly better than the others.

Finally, the ALE shows that the arithmetic average of the last 20 days and the random forest models have the worst score. As a result, those models seem to over-predict the daily volume in the out-of-sample test set. The ARMA(1,1) and the linear regressions provide the best scores, even if the Ridge regression seems to be slightly better. The differences between the results obtain in the in-sample and in the out-of-sample test sets can be explained by the fact that the out-of-sample test period could be more difficult to forecast because of the volatility of the markets. The comparison between the scores obtained using the in-sample test set or the out-of-sample test set show that the linear regressions and particularly the OLS provide good and consistent results (compared to the baseline score). Moreover, the ARMA(1,1) model provides also strong results and could be a useful model. However, this model could not converge for some stocks if the time-series used is not stationary.

4.5 Stacking

The table 6 shows the results obtained by stacking the predictions from ARMA(1,1), OLS, LASSO and Ridge. We will simply take the arithmetic mean of the four different predictions.

Table 6: Stacking model average scores for 45 of the largest stocks in the FTSE 100.

Model	**R^2-score**	**MAE**	**ALE**
Stacking (in-sample test)	0.410	0.243	0.363
Stacking (out-of-sample test)	0.329	0.234	0.360

Scores obtained for different metrics (R^2-score, MAE score & ALE score) using a stacking model composed of ARMA(1,1) model, OLS, LASSO & Ridge regressions.

The stacking model obtains the best scores overall for both the in-sample test set and the out-of-sample test set. As a result, a stacking model seems to be a very strong model which provides consistent results.

Conclusion: This section showed the time-series and clustering analysis for a large universe of stocks. The results obtained for different stocks allow us to draw clear conclusions concerning the different machine learning models. The stacking model using results from ARMA(1,1) and OLS, LASSO & Ridge regressions provides the best scores overall. Moreover, using a simple OLS model can also provide strong and consistent results. Finally, this section showed only one-day ahead prediction for the traded volume, consequently there is a lack of intra-day volume dynamic. The following section will take into account the intra-day dynamic of the volume to update the initial prediction.

5 Updating Initial Prediction using Intra-Day Information

The main objective of this section is to update our initial prediction using additional intra-day information. In fact, in the previous section, we investigated methods of daily predictions to forecast the one-day ahead volume that will be traded. Update the initial prediction using intra-day information is essential because it takes into account the dynamic of the volume profile. The intra-day volume profile can be affected by market news or special events, that's why it is important to take into account additional intra-day information. In this section, we consider that we are in the following day and at each hour we will update the initial prediction with new intra-day information. For daily prediction, the best machine learning model to use is a Linear Regression associated with a Walk-Forward cross-validation (See Sec.4.4). In the following we enrich this analysis by considering new features which take into account intra-day information.

5.1 Feature Engineering

As for Sec.3.2, feature engineering is a crucial step in our goal to update the initial predictions. An initial subset of features will be chosen, and then some features will be added.

5.1.1 Initial Features

The initial subset of features to be chosen from the initial ones are : hourly first and last price, hourly traded volume, cumulative log volume, hourly open and last traded size, daily open and close price.

5.1.2 Created Features

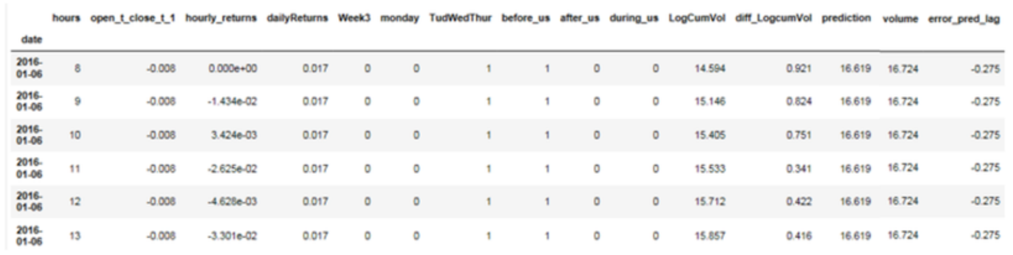
The different features that have been created in order to update our initial prediction are as follows:

- *log* difference between open price of the actual day and the close price of the previous day. This feature will capture the over-night information
- Hourly *log* returns with the *log* difference between the first price at time t and the first price at time t-1.
- Daily *log* returns of the previous day using the *log* difference between the close price of the previous day and the open price of the previous day.
- One-hot encoding with the hours that are: before, during, and after the opening of the US market (2.30pm UTC).
- Difference between the cumulative *log* volume of today and the cumulative *log* volume of the previous day for the same hour.

- Some temporal features such as: time-of-day (in hours), Week 3, Monday, Tuesday & Wednesday & Thursday.
- Lagged error term of the initial prediction obtained using our linear regression model.

The final data frame which contains all the features is presented in Fig.22 (below):

Figure 22: Data Frame used to update the initial prediction of the daily volume.



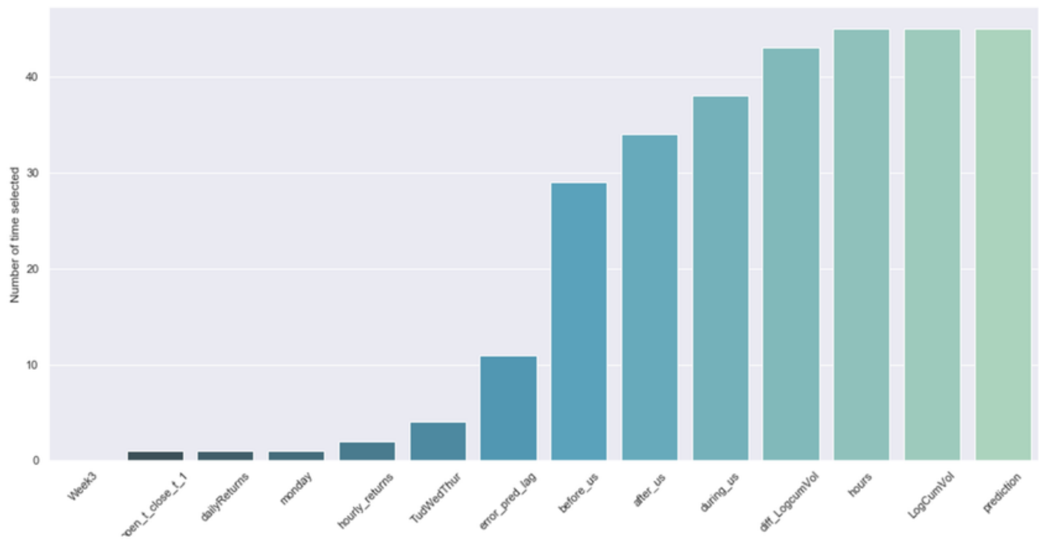
	hours	open_t_close_t_1	hourly_returns	dailyReturns	Week3	monday	TudWedThur	before_us	after_us	during_us	LogCumVol	diff_LogcumVol	prediction	volume	error_pred_lag
date															
2016-01-06	8	-0.008	0.000e+00	0.017	0	0	1	1	0	0	14.594	0.921	16.619	16.724	-0.275
2016-01-06	9	-0.008	-1.434e-02	0.017	0	0	1	1	0	0	15.146	0.824	16.619	16.724	-0.275
2016-01-06	10	-0.008	3.424e-03	0.017	0	0	1	1	0	0	15.405	0.751	16.619	16.724	-0.275
2016-01-06	11	-0.008	-2.625e-02	0.017	0	0	1	1	0	0	15.533	0.341	16.619	16.724	-0.275
2016-01-06	12	-0.008	-4.628e-03	0.017	0	0	1	1	0	0	15.712	0.422	16.619	16.724	-0.275
2016-01-06	13	-0.008	-3.301e-02	0.017	0	0	1	1	0	0	15.857	0.416	16.619	16.724	-0.275

This Data Frame contains all the features used to perform the chosen machine learning model.

5.2 Feature Selection

The algorithm used to automatically perform feature selection over the stock universe is the same as described in Sec.3.4 & 4.3 (Bottom-up approach). As in Sec.4.3, each stock has a certain subset of best features and it is possible to analyse which features are picked the most frequently. Again, the most important features are related to the volume (initial prediction, *log* cumulative volume, difference between the cumulative *log* volume of today and the cumulative *log* volume of the previous day) or the time (hours, hours related to the US market).

Figure 23: Number of time each feature is picked on 45 simulations.



This graph shows that the features that are the most frequently picked are relative to the volume traded or the time of the day (in hours). The features related to price returns or the seasonality (such as Monday or week 3) are not very useful.

Once the best subset of features is chosen for each stock, it is possible to perform the chosen machine learning model (in this section an OLS regression).

5.3 Results

This section summarizes some of the results obtained using a Linear Regression and a walk-forward cross-validation over the dataset presented in Fig.22.

5.3.1 Example of Updated Prediction

To show how the model reacts to new intra-day information, we can draw the updated predictions for one particular stock on a particular day and compare it with the actually realized volume on that day.

Figure 24: Volume update for 'AAL.L' on 2017/03/02.

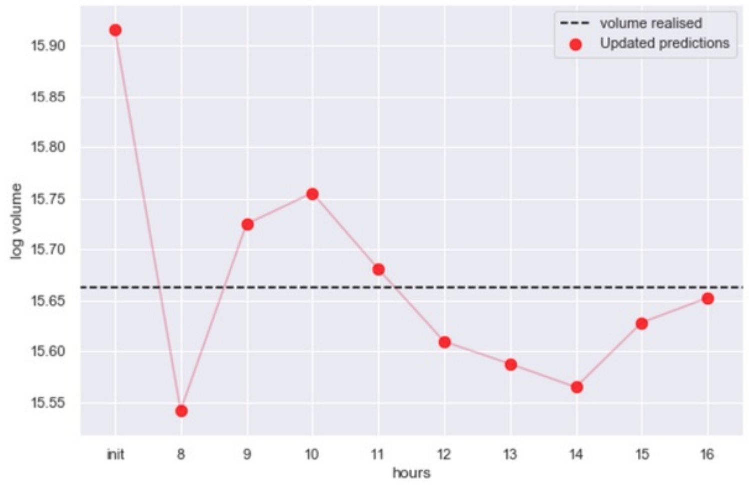
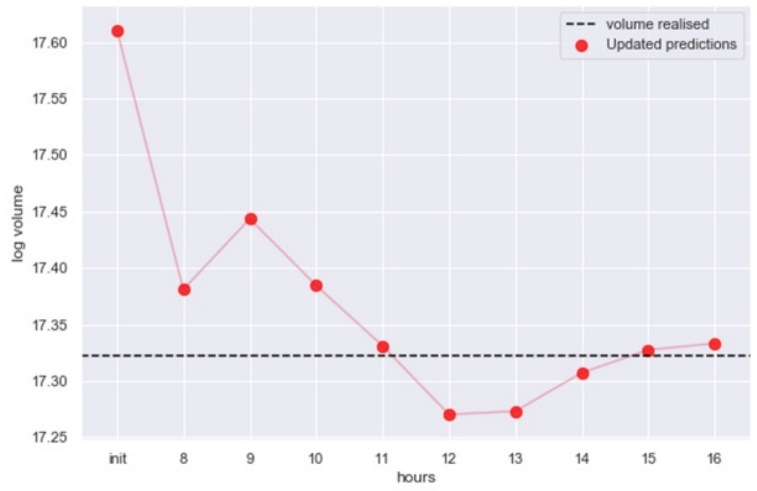


Figure 25: Volume update for 'BP.L' on 2017/03/02.



In both graphs, we can see that the the updated volume prediction converges to the daily realised volume.

In Fig.24 & 25, we can see that the more intra-day information becomes available, the easier it is to predict the daily traded volume. However, we can not draw conclusions with two examples, therefore we will study the distribution of the R^2-score, MAE and ALE error throughout the day for the full stock universe.

5.3.2 Distribution of the Metrics

In the following, we present the distribution of the metrics (R^2-score, MAE, ALE) for different times of the day sampled in hours. In fact, in each hour we can compute the different metrics for all of our stocks (45 stocks), consequently, we can estimate a distribution of them. To estimate a distribution of the metrics, a kernel density estimation (associated with a Gaussian kernel) will be used.

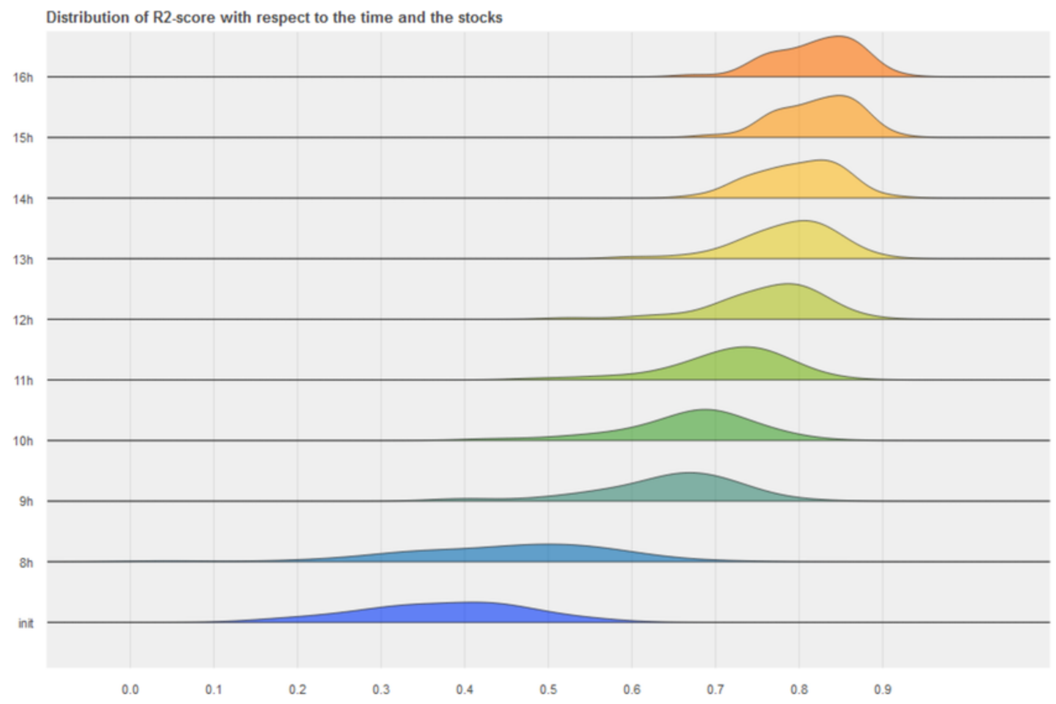
Remark 5.1. A kernel density estimation (KDE) is a non-parametric way to estimate an unknown density function f. Let $X = (X_1, \cdots, X_n)$ be a vector of n i.i.d variables drawn from some distribution with an unknown density f. We are interested in estimating the unknown density f. Let K be a kernel (a non-negative function) and $h > 0$ be a smoothing parameter called the bandwidth. Then, the kernel density estimator of f is defined as [13]:

$$\hat{f}_h(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right).$$

In this section, we will use the Gaussian kernel function, defined as: $K(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \mathbb{1}_{]-\infty,+\infty[}$.

The figures 26, 27 & 28 will show the distributions of each metric using a kernel density estimation for each hours.

Figure 26: Distribution of R^2-score for different times of the day.



This figure shows that the mean of the R^2-score distribution is shifting to the right (toward 0.9) this means that the updated predictions are shifting to the realised volume. Moreover, the standard deviation of the R^2-score distribution is decreasing with respect to the time. Consequently, the scores obtained are becoming more homogeneous across stocks.

As can be seen in Fig.26 the mean of the R^2-score distribution increases intra-day, which means that our predictions are becoming more and more accurate the closer we get to the closing auction. Moreover, we notice that there is a high increase in the mean of the distributions from 8 a.m to 9 a.m. Hence, the information collected between 8 a.m and 9 a.m seem to be crucial. Concerning the variance of the distributions, we can notice that it shrinks more and more, as we approach the closing auction. This proves that the scores obtained are becoming more and more homogeneous across the 45 stocks studied.

Figure 27: Distribution of ALE-score for different hours.

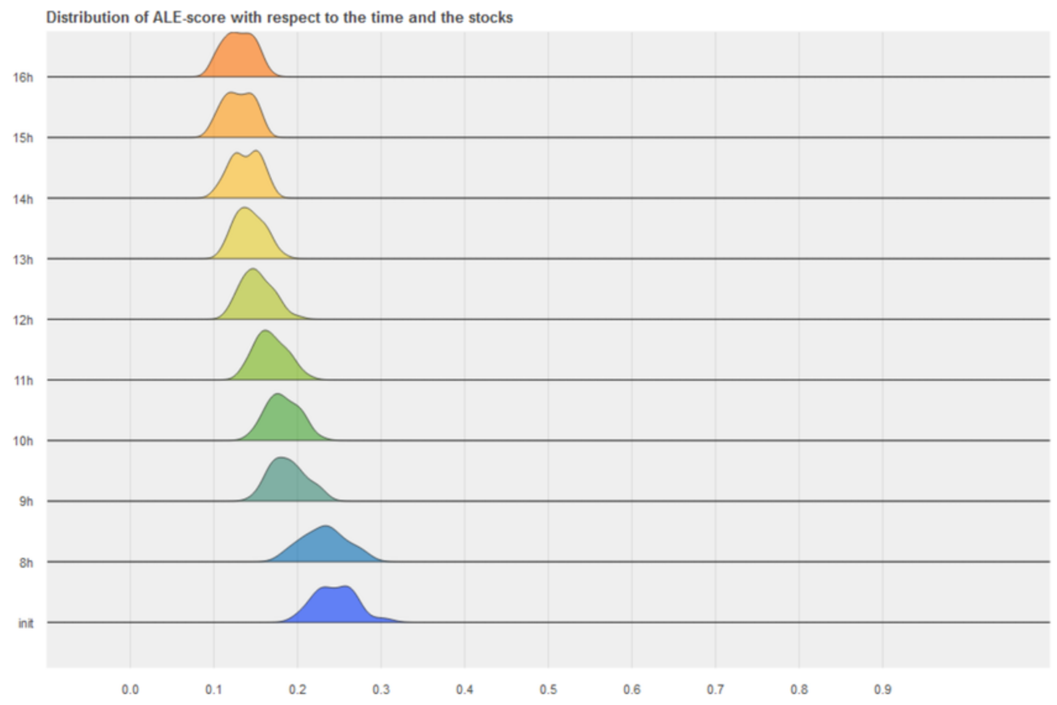
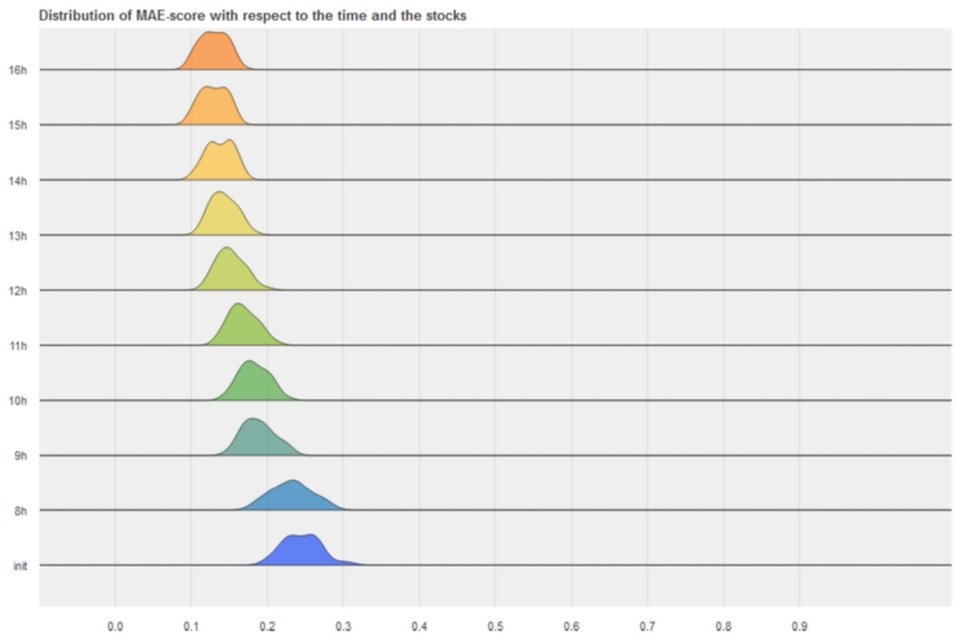


Figure 28: Distribution of MAE-score for different hours.



Both Fig.27 & 28 show that the mean of the MAE and the ALE distributions are shifting to the left, this means that the updated predictions are more and more accurate the closer we get to the closing auction. Moreover, the standard deviation of the MAE and the ALE distributions are decreasing with respect to the time. Consequently, the scores obtained are becoming more homogeneous across stocks.

Likewise, for both the mean of the ALE & MAE distributions is decreasing with respect to time. Again, this is a proof that the closer we get to the closing auction, the more accurately we can predict the daily volume. Moreover, as for the distribution of the R^2-score, the variance of both the ALE and the MAE distributions are shrinking more and more as we approach the closing auction. Again, this means that the scores obtained are becoming more homogeneous.

Conclusion: Updating the initial daily volume prediction is very important because it takes into account the volume profile dynamic and it can helps the buyers to update their trading strategies. In this section, we have seen that by creating few features and then using a OLS regression, it is possibly to greatly improve the initial daily volume prediction. In fact, the closer we are to the closing auction, the better are the R^2-score, ALE & MAE and the variances are narrowed more and more.

Conclusion

This thesis showed different techniques to analyse time-series data. A large part of the work concerned the feature engineering (i.e. creating and processing features). The results obtained in Sec.4 show that the features related to the volume and the seasonality have more predictive power than the features related to the price. The results presented in Sec.4 for an universe of 45 stocks in the FTSE 100 showed that an OLS regression over a selected subset of features provide on average a better score that the baseline score (obtained using the average traded volume over the 20 past days). Then, the Sec.5 explored a dynamic way to improve the initial volume prediction using intra-day new information. The results showed that the daily volume prediction is becoming more and more accurate the closer we get to the closing auction. In this thesis, the results obtained using the in-sample test set and the out-of-sample test set were systematically confronted. It showed that the MAE and ALE results where consistent whereas the R^2-scores were slightly different. Concerning the machine learning models and the statistical tools used, we have seen that the simplest methods such that an OLS regression perform very well compared to our baseline score. However, even if the model used is simple, it is necessary to work extensively on feature engineering.
The way that we have discriminated the stocks by selecting one best subset of features for each stock was an arbitrary decision that I have chosen because I wanted to look at the correlation between stocks and analyse them individually. However, other techniques are used, and one of them consists of considering that all the stocks have the same behaviour. As a result of this, a best subset of features can be selected across all the stocks. Moreover, at the beginning of the project, the special days (half-trading days, days before bank holidays etc.) were dropped because they have different behaviour than normal trading days. Thus, it is possible to study carefully those days and the behaviour of the volume for those days. Finally, different machine learning can be used to forecast time-series, such that recurrent neural network and more particularly LSTM neural networks. That kind of models where not studied in this thesis but they can be very useful to perform time-series forecasting.

A Linear Regressions

The general goal of regression analysis is to learn some relationship between a variable to predict $y \in \mathbb{R}$ and some covariates $x = (x_1, ..., x_p)^T \in \mathbb{R}^p$, with $p \geq 1$. This is done by a learning function that maps the input x to the output y. Linear regression is interested in modeling y using a linear link function of x, i.e., the variable y is modeled by $\theta_0 + \theta_1 x_1 + ... + \theta_p x_p$ where $(\theta_0, ..., \theta_p)$ are the parameters of the linear link function. To learn the values of these parameters, $(\theta_0, ..., \theta_p)$, we observe $n \geq 1$ pairs (x_i, y_i) that are supposed to come from the same generating mechanism. In what follows we introduce the ordinary least squares (OLS) approach which basically consists in minimizing the sum of squares of the distance between the observed values y_i and the predicted values at x_i under the linear model.

A.1 Ordinary Least Squares (OLS)

We focus on a regression problem with $n \geq 1$ observations and $p \geq 1$ covariates. For notational convenience, for $i = 1, ..., n$, we consider $y_i \in \mathbb{R}$ and $x_i = (x_{i,0}, ..., x_{i,p})^T \in \mathbb{R}^{p+1}$ with $x_{i,0} = 1$. This is only to include the intercept in the same way as the other coefficients. The OLS estimator is any coefficients vector $\hat{\boldsymbol{\theta}}_n = (\hat{\boldsymbol{\theta}}_{n,0}, ..., \hat{\boldsymbol{\theta}}_{n,p})^T \in \mathbb{R}^{p+1}$ such that

$$\hat{\boldsymbol{\theta}}_n \in \text{argmin}_{\boldsymbol{\theta} \in \mathbb{R}^{p+1}} \sum_{i=1}^{n} \left(y_i - x_i^T \boldsymbol{\theta}\right)^2 \tag{A.1}$$

It is useful to introduce the notations

$$X = \begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \end{pmatrix} = \begin{pmatrix} x_{1,0} & \dots & x_{1,p} \\ \vdots & & \vdots \\ x_{n,0} & \dots & x_{n,p} \end{pmatrix} \in \mathbb{R}^{n \times (p+1)}, \quad Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

The matrix X which contains the covariates is called the design matrix. With the previous notation, (A.1) becomes

$$\hat{\boldsymbol{\theta}}_n \in \text{argmin}_{\boldsymbol{\theta} \in \mathbb{R}^{p+1}} \|Y - X\boldsymbol{\theta}\|^2$$

where $\|.\|$ stand for the Euclidean norm.

With the above formulation, the OLS has a nice geometric interpretation : $\hat{Y} = \hat{\boldsymbol{\theta}}_n$ is the closest point to Y in the linear subspace $span(X) \subset \mathbb{R}^n$ (where $span(A)$ stands for the linear subspace generated by the columns of A). Using the Hilbert projection theorem ($\mathbb{R}^n$ is a Hilbert space, $span(X)$ is a (closed) linear subspace of $\mathbb{R}^n$), $\hat{Y}$ is unique and is characterized by the fact that the vector $Y - \hat{Y}$ is orthogonal to $span(X)$. This property is equivalent to the so-called normal equation:

$$X^T(Y - \hat{Y}) = 0.$$

Since $\hat{Y} = X\hat{\boldsymbol{\theta}}_n$, we obtain that the vector $\hat{\boldsymbol{\theta}}_n$ must verify

$$X^T X\hat{\boldsymbol{\theta}}_n = X^T Y.$$

Thus, the OLS estimator exists and is uniquely defined if and only if $X^T X$ is invertible. In this case, it has the following expression :

$$\hat{\boldsymbol{\theta}}_n = (X^T X)^{-1} X^T Y.$$

A.2 Ridge regularization

The ridge estimator is defined as a solution of the following minimization problem:

$$\hat{\boldsymbol{\theta}}_n \in \operatorname{argmin}_{\boldsymbol{\theta}\in\mathbb{R}^{p+1}} \|Y - X\boldsymbol{\theta}\|_2^2 + \lambda\|\boldsymbol{\theta}\|_2^2, \tag{A.2}$$

where $\lambda > 0$, called the regularization parameter, is fixed by the analyst. The minimizer of (A.2) exists and is unique. It is given by :

$$\hat{\boldsymbol{\theta}}_n^{(rdg)} = (X^T X + \lambda I_p)^{-1} X^T Y.$$

Some simple remarks can be stated concerning the ridge estimate :

- Intuitively, when $\lambda \to 0$, we obtain the OLS. When $\lambda \to \infty$, we estimate 0.
- Doing ridge is adding a regularization term to the square loss of OLS, aiming to penalize for large coefficient in θ. Other norms might be used such as $\sum_{k=1}^{p} |\boldsymbol{\theta}_k|$ (See the next section about LASSO).
- As the expression (A.2) is a Lagragian with constraint $\|\boldsymbol{\theta}\|^2 \leq c$, the ridge is an OLS under constraints. The link between c and λ is not explicit.

A.3 LASSO regularization

The lasso estimator is defined as a solution of the following minimization problem:

$$\hat{\boldsymbol{\theta}}_n \in \operatorname{argmin}_{\boldsymbol{\theta}\in\mathbb{R}^{p+1}} \|Y - X\boldsymbol{\theta}\|_2^2 + \lambda\|\boldsymbol{\theta}\|_1, \tag{A.3}$$

where $\lambda > 0$, called the regularization parameter, is fixed by the analyst. The LASSO estimator is not always unique for a fixed λ. The LASSO regularization has some properties:

- The LASSO is a convex problem.
- $\hat{\boldsymbol{\theta}}_n^{(lasso)}$ has potentially many zeroed coefficients. The λ parameter controls the sparsity level: if λ is large, solutions are very sparse.

A.4 Elastic Net regularization

The elastic net estimator is defined as a solution of the following minimization problem:

$$\hat{\boldsymbol{\theta}}_n \in \operatorname{argmin}_{\boldsymbol{\theta} \in \mathbb{R}^{p+1}} \|Y - X\boldsymbol{\theta}\|_2^2 + \lambda \left(\gamma |\boldsymbol{\theta}\|_1 + (1-\gamma)\frac{|\boldsymbol{\theta}\|_2^2}{2} \right), \tag{A.4}$$

where $\lambda > 0$, called the regularization parameter, is fixed by the analyst and γ is a constant that multiply the penalty terms. The motivations under elastic net are:

- It helps selecting all relevant but correlated variables (not only one as for the LASSO).
- Some coefficients are less shrunk compared to LASSO.

B Volatility

Historical volatility can be measured using different ways which can use the open (o), close (c), high (h) or low (l) prices. This appendix will describe the four most used volatility [11]:

- Historical volatility (close-to-close): Simplest and most common type of calculation of volatility only using log-returns at the close and taking into account the number of trading days.

$$\sigma_{historical} = \sum_{i=1}^{N} \left[\log\left(\frac{c_i}{c_{i-1}}\right) - \overline{\log\left(\frac{c_i}{c_{i-1}}\right)} \right]^2$$

- Garman-Klass volatility: it is a type of volatility that incorporates intra-day information such as the closing and opening price. (With open (o), close (c), high (h) and low (l)).

$$\sigma_{Garman-Klass} = \sqrt{\frac{N}{n} \sum_{i=1}^{N} \frac{1}{2} \left(\log\left(\frac{h_i}{l_i}\right)\right)^2 - (2\log(2) - 1)\left(\log\left(\frac{c_i}{o_i}\right)\right)^2}$$

  With N the number of trading days and n the sample size.

  In most cases, the Garman-Klass volatility is more efficient than just taking into account close-to-close information. However, this model does not include overnight information.

- Rogers-Satchell volatility: The previous historical volatilities assume that the average return is zero. As a reult of this, the Rogers-Satchell volatility can measure the volatility for securities with on-zero mean. However, this volatility does not handle jumps, hence it underestimates the volatility.

$$\sigma^2_{Rogers-Satchell} = \sqrt{\frac{1}{N}} \sqrt{\sum_{i=1}^{N} \log\left(\frac{h_i}{c_i}\right) \log\left(\frac{h_i}{o_i}\right) + \log\left(\frac{l_i}{c_i}\right) \log\left(\frac{l_i}{o_i}\right)}$$

- Yang-Zhang volatility: This volatility measure handles both intra day information and overnight volatility contrary to the Garman-Klass volatility. (With open (o), close (c), high (h) and low (l)).

$$\sigma_{Yang-Zhang} = \sqrt{\sigma^2_{overnight} + k\sigma^2_{open-to-close} + (1-k)\sigma^2_{Rogers-Satchell}}$$

$$\sigma^2_{overnight} = \frac{1}{N-1} \sum_{i=1}^{N} \left[\log\left(\frac{o_i}{c_{i-1}}\right) - \overline{\log\left(\frac{o_i}{c_{i-1}}\right)} \right]^2$$

$$\sigma^2_{open-to-close} = \frac{1}{N-1} \sum_{i=1}^{N} \left[\log\left(\frac{c_i}{o_i}\right) - \overline{\log\left(\frac{c_i}{o_i}\right)} \right]^2$$

$$\text{with } k = \frac{0.34}{1.34 + \frac{N+1}{N-1}} \text{ , with N the number of trading days.}$$

References

[1] Sanjay.M: Why and how to Cross Validate a Model?,
`https://towardsdatascience.com/why-and-how-to-cross-validate-a-model-d6424b45261f`

[2] Vasily E. Tarasov and Valentina V. Tarasova. *Long and Short Memory in Economics: Fractional-Order Difference and Differentiation.* IRA-International Journal of Management and Social Sciences. 2016. Vol.5. No.2. P.327-334.

[3] Vadim Smolyakov: Ensemble Learning to Improve Machine Learning Results,
`https://blog.statsbot.co/ensemble-learning-d1dcd548e936`

[4] L.J.P. van der Maaten and G.E. Hinton. *Visualizing High-Dimensional Data Using t-SNE.* Journal of Machine Learning Research 9(Nov):2579-2605, 2008.

[5] Christian Pascual: Tutorial: Understanding Regression Error Metrics in Python,
`https://www.dataquest.io/blog/understanding-regression-error-metrics/`

[6] Vladimir Markov, Olga Vilenskaia and Vlad Rashkovich. *Quintet Volume Projection.* Automated Trader Magazine, Issue 44, Q1, 2018.

[7] Rakshith Vasudev: What is One Hot Encoding? Why And When do you have to use it?,
`https://hackernoon.com/what-is-one-hot-encoding-why-and-when-do-you-have-to-use-it-e3c6186d008`

[8] Changyong FENG, Hongyue WANG, Naiji LU, Tian CHEN, Hua HE, Ying LU, and Xin M. *Log-tranformation and its implications for data analysis.* Shanghai Arch Psychiatry, 2014 Apr, 26(2): 105–109.

[9] Stephen G. Kobourov. *Force-Directed Drawing Algorithms.* University of Arizona.

[10] Jaako Peltonen, Francesco Corona and Manuel J.A.Eugster. *Dimensionality Reduction and Visualization.* Tampere University, MTTS1, Lecture 13, Spring, 2014.

[11] Colin Bennett and Miguel A.Gil *Measuring Historical Volatility.* Santander, February, 2018.

[12] Lou, Xiaoxia, and Tao Shu. *Price Impact or Trading Volume: Why Is the Amihud (2002) Measure Priced?.* The Review of Financial Studies 30.12 (2017): 4481-4520.

[13] Francois Roueff. *Introduction aux statistiques non-parametriques* [introduction to nonparametric statistics]. Telecom Paris-Tech, SD205, Janvier, 2017.

IMPERIAL COLLEGE LONDON

THESIS DECLARATION FORM

(for candidates whose degree will be awarded by **Imperial College**)

Declaration: I hereby confirm that the work contained in this thesis is my own work unless other wises stated.

Name : Francois Le Dain **CID** : 01565256

Title of Thesis : Trading Volume Forecasting in the Equity Market using Machine Learning and Statistical Models

Month and year of submission for examination: September 2019

Date : September 4th 2019

Signature of Candidate [illegible]

Trading Volume Forecasting in the Equity Market using Machine Learning and Statistical Models

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

PAGE 31

PAGE 32

PAGE 33

PAGE 34

PAGE 35

PAGE 36

PAGE 37

PAGE 38

PAGE 39

PAGE 40

PAGE 41

PAGE 42

PAGE 43

PAGE 44

PAGE 45

PAGE 46

PAGE 47

PAGE 48

PAGE 49

PAGE 50

PAGE 51

PAGE 52

PAGE 53

PAGE 54

PAGE 55

PAGE 56

PAGE 57

PAGE 58

PAGE 59

PAGE 60

PAGE 61

PAGE 62

PAGE 63