2019

SHADOW INDEX DOCUMENTATION

STABLE GROUP LTD

DATA SCIENCE TEAM

STABLE | Data Science Team | 3 Whitehall Court

Documentation for shadow index

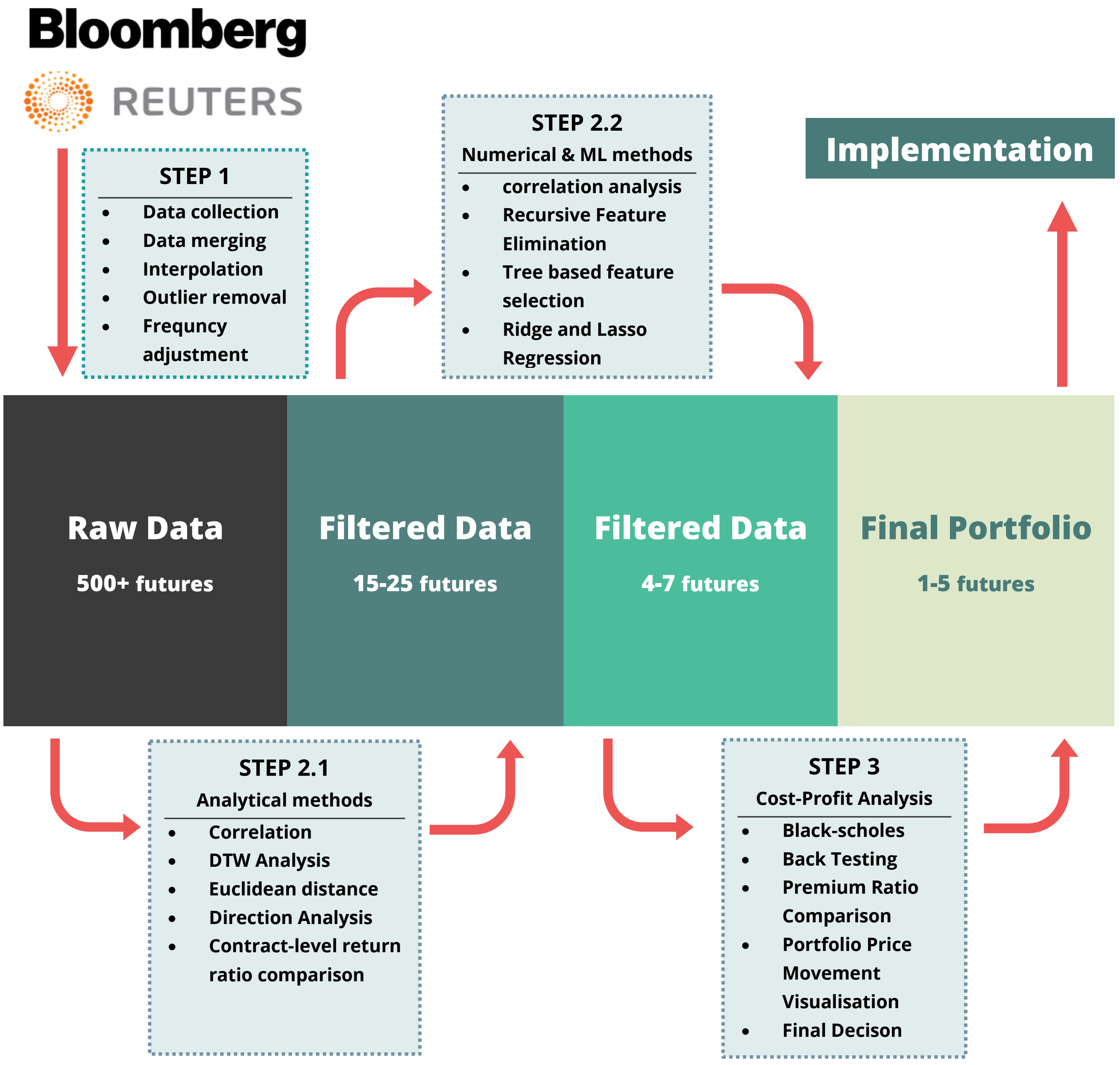
A Brief Intro:

Stable Group has always aimed for risk minimisation and dynamic hedging for each contract to reduce total risk as well as to lower price of the product. When we sell a product, we decide to find some similar future indexes which will replicate the price movement of that product. Hence, the selection of portfolio and the calculation of corresponding weights is the main challenge.

The hedging process compose of 3 main sections:

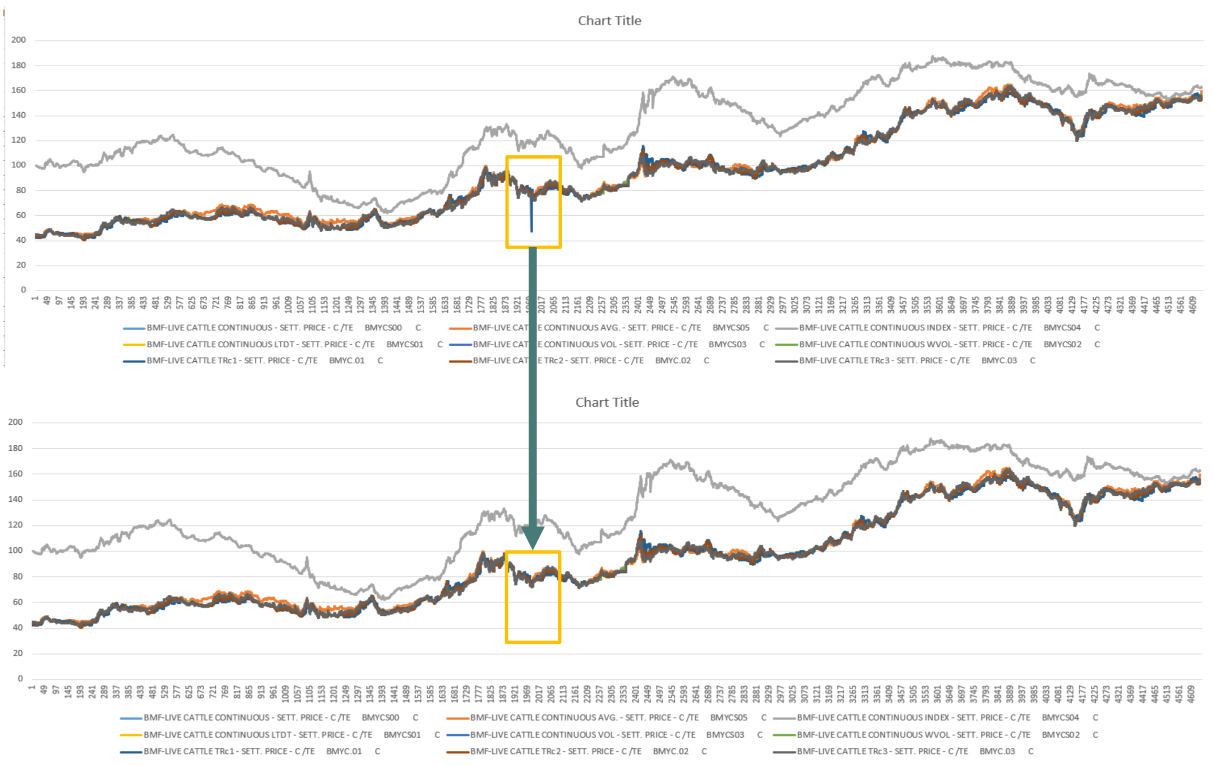
1. Perform Data collection & cleaning, which involves algorithms in Python Pandas DataFrame
2. Determine Choice of initial portfolios & weights, which involves various feature selection algorithms and Ridge regression.
3. Analysis Cost effectiveness, which involves implied volatility calculation, Black-Scholes and Monte Carlo methods for price simulation as well as implementation of back-testing

Workflow of Shadow Index Project



Section 1: Data Preparation

For the first section, we downloaded and merged 500+ futures' daily close prices from Reuters & Bloomberg and adjusted the data frequency according to target index's movement frequency. Then we converted daily price data to log return ratio over fixed period where length of the period is same to the maturity. We also adjusted the extreme price movement as it would create large bias in linear regression as well as correlation calculation.



Section 2: Main Challenge

The second section was the major work area where I needed to select the minimum number of futures out of 500+ futures to replicate index movement. The main challenges:

- The index we worked on to replicated is a monthly updating index with only 247 data points available. Any direct machine learning and feature selection algorithm will make the model significantly overfitting.
- The weights of the components in hedging portfolio should not be large as we need to consider cost of purchasing future contracts.

In order to shrink 500+ choices to less than 20 choices, I have implemented explicit and informative analysis including DTW (Dynamic Time Warping) analysis, Correlation analysis, Euclidean Distance calculation and price movement direction analysis on log return rates of different futures in order to finalise 20 most closely related futures to our target index.

Then I used various feature selection algorithms in python including correlation analysis, Recursive Feature Elimination, Tree based feature selection, Ridge and Lasso Regression to further reduce the number of commodity futures from 20 to 3-5.

At the very last, I used ridge regression to confirm the final weights of the portfolio. Ridge regression is similar to linear regression, but it gives an extra penalty on the size of the fitting coefficients. This complies to the second challenge: we need to consider the price of future contracts. Though we want the replicating portfolio simulates the movement of target index as accurate as possible, we also need to minimise the size of portfolio.

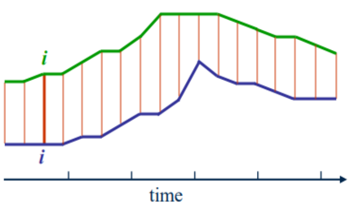
Now, let me briefly explain all methods used in this section.

Section 2.1: Analytical similarity methods

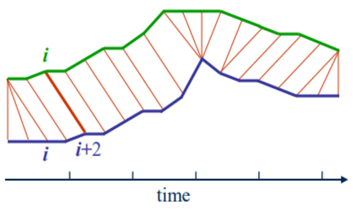
Dynamic Time Warping:

Dynamic time warping is an algorithm used to measure similarity between two sequences which may vary in time or speed. It works as follows:

- Divide the two series into equal points.
- Calculate the Euclidean distance between the first point in the first series and every point in the second series. Store the minimum distance calculated. (this is the 'time warp' stage)
- Move to the second point and repeat 2. Move step by step along points and repeat 2 till all points are exhausted.
- Repeat 2 and 3 but with the second series as a reference point.
- Add up all the minimum distances that were stored and this is a true measure of similarity between the two series.



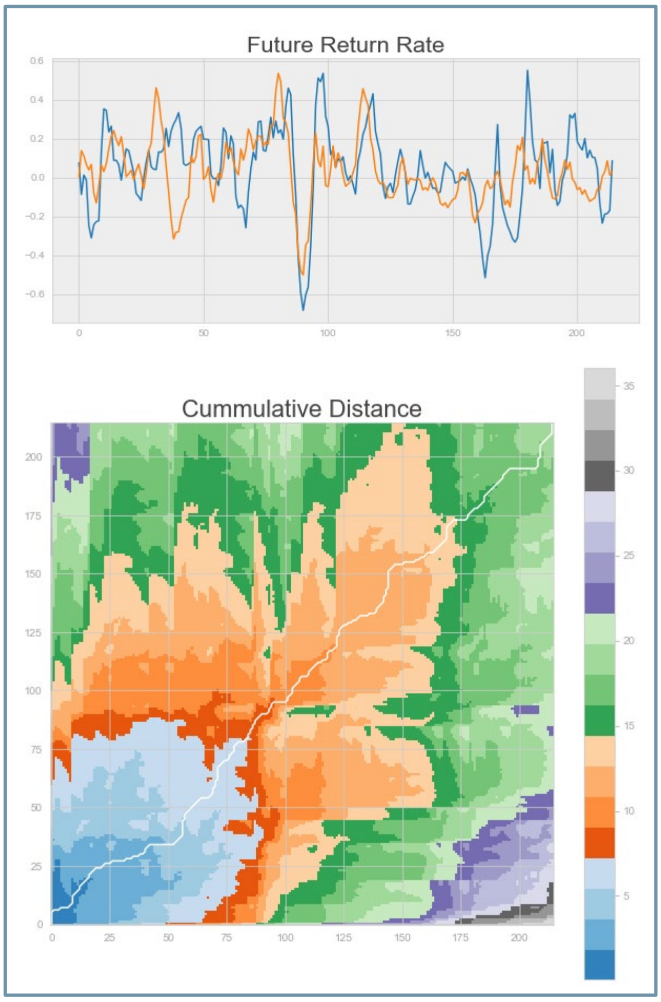
Any distance (Euclidean, Manhattan, …) which aligns the *i*-th point on one time series with the *i*-th point on the other will produce a poor similarity score.

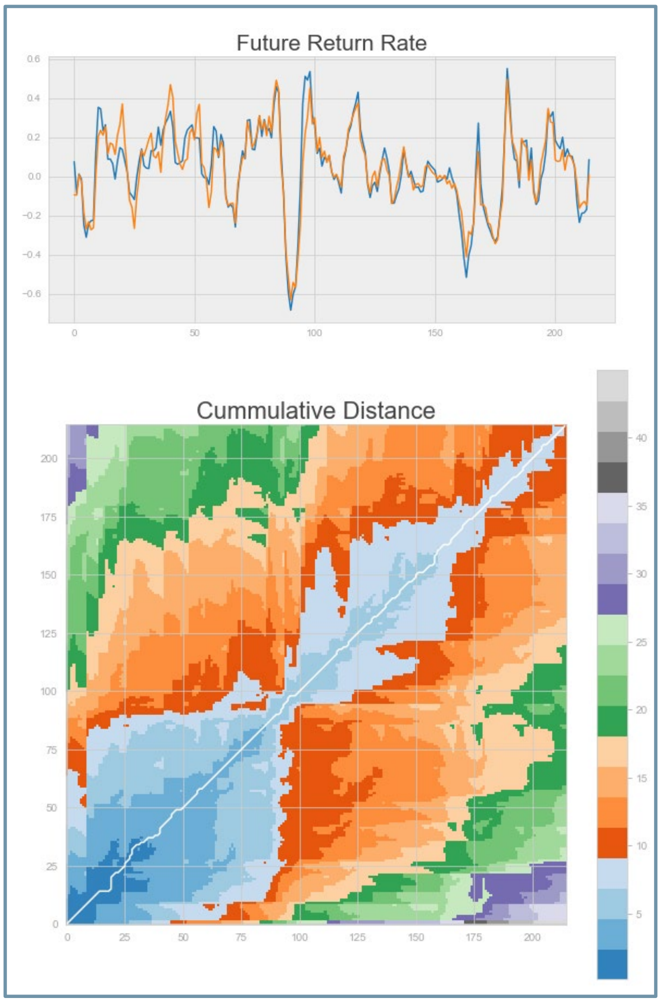


A non-linear (elastic) alignment produces a more intuitive similarity measure, allowing similar shapes to match even if they are out of phase in the time axis.

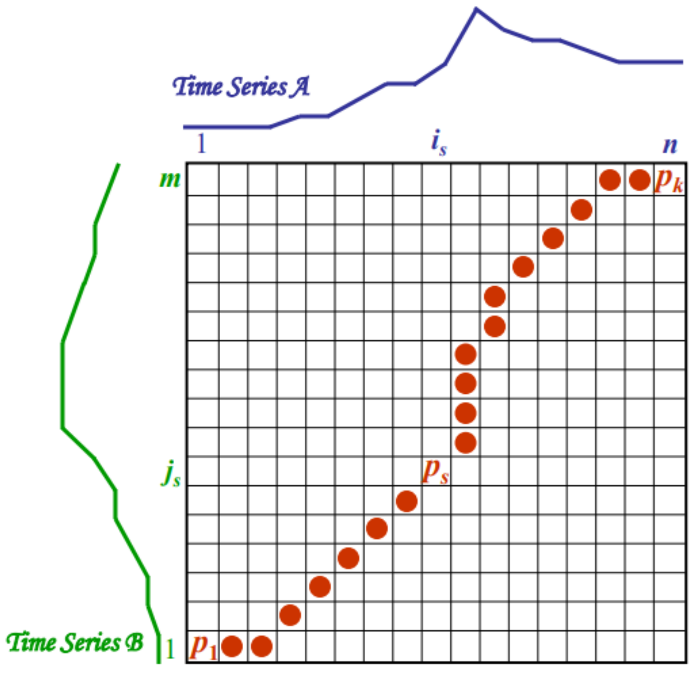
The steps above have followed some constraints in order to optimize the solution found. DTW is a better method than direct Euclidian and correlation method as it will find similar pattern regardless of time scale

Now take an example of two future index:





The diagonal line is just the cumulative Euclidian distance without time scaling, the second upper or lower diagonal is the cumulative Euclidian distance for comparing price with lag 1, etc. And the value of number in position [n,m] is the Euclidian distance between timeseries1 at time n and timeseries2 at time m.



Time-normalized distance between $\mathcal{A}$ and $\mathcal{B}$:

$$D(\mathcal{A}, \mathcal{B}) = \left[\frac{\sum_{s=1}^{k} d(p_s) \cdot w_s}{\sum_{s=1}^{k} w_s} \right]$$

$d(p_s)$: distance between i_s and j_s

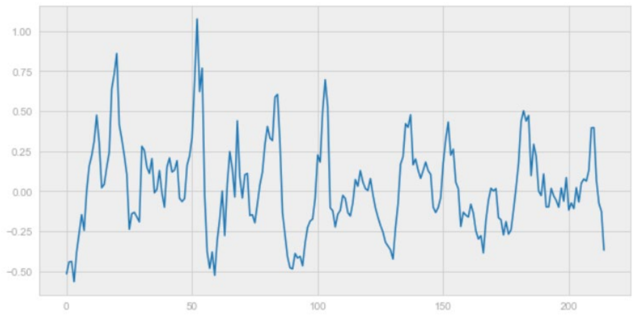
$w_s > 0$: weighting coefficient.

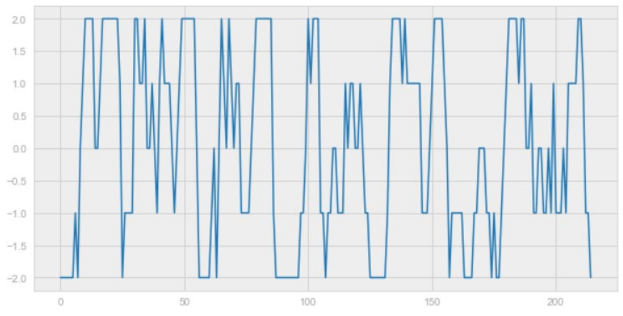
Best alignment path between $\mathcal{A}$ and $\mathcal{B}$:

$$P_0 = \arg\min_{P} (D(\mathcal{A}, \mathcal{B})).$$

Other Similarity Metrics:

Euclidian distance is just the square sum of distance between two timeseries in every datapoint. Correlation is just the Pearson correlation between two timeseries. Direction distance is more complicated. I define the return rate higher than 0.2 or lower to -0.2 as fast change, rate between 0.05 and -0.05 as unchanged and the rest as moderate change. Hence a graph of price change can be converted as below:



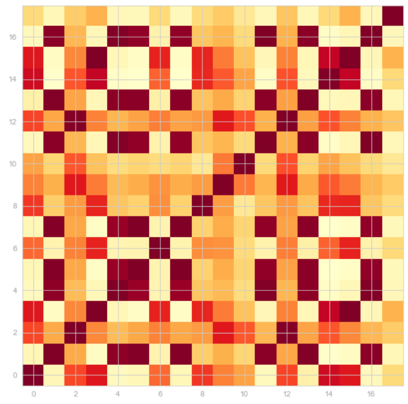


To compare target index with other futures indexes, we used DTW distance, Euclidian distance, Correlation Distance and direction distance. The sorted result gives the most closed few futures to go into the next step.

Section 2.2: Numerical & ML methods

Correlation Selection:

If the futures selected has very high correlation, then we can assume the extra one future will not improve hedging accuracy too much as they are providing replicating information. Hence for the example we used here, we need only to keep one of future 1,4,5,7,11,13,16.



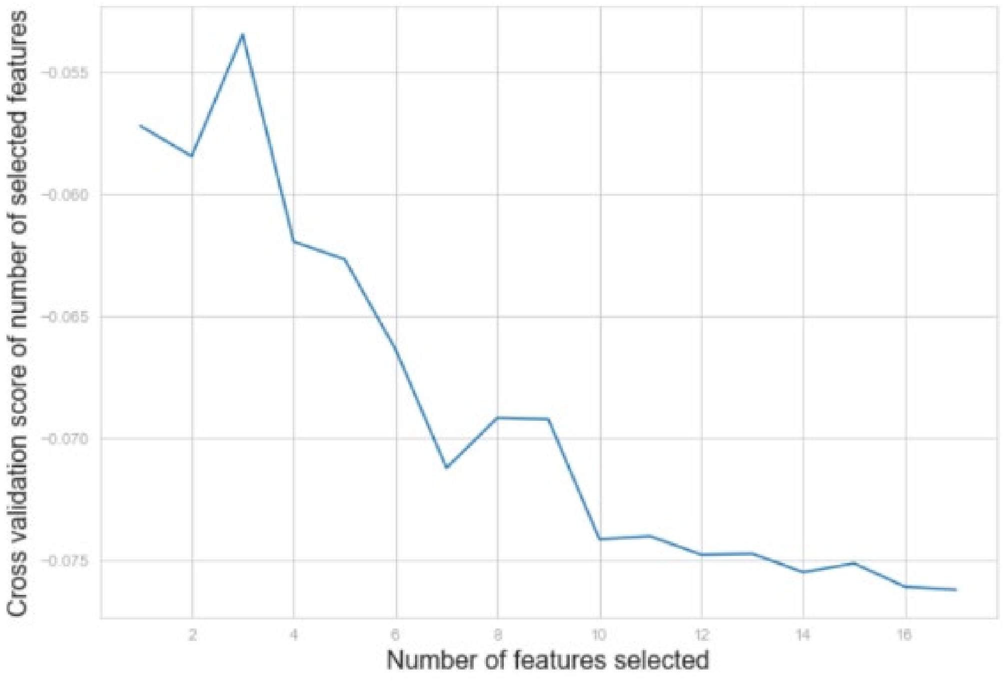
Recursive Feature Elimination:

Recursive Feature Elimination (RFE) as its title suggests recursively removes features, builds a model using the remaining attributes and calculates model accuracy. It fits a model and removes the weakest feature (or features) until the specified number of features is reached. Features are ranked by the model's coef_ or feature_importances_ attributes, and by recursively eliminating a small number of features per loop, RFE attempts to eliminate dependencies and collinearity that may exist in the model.

RFE is able to work out the combination of attributes that contribute to the prediction on the target variable (or class). Scikit-Learn does most of the heavy lifting just import RFE from sklearn.feature_selection and pass any classifier model to the RFE() method with the number of features to select.

In our example, the algorithm recommends keeping only 3 variable which is future 1,3,13

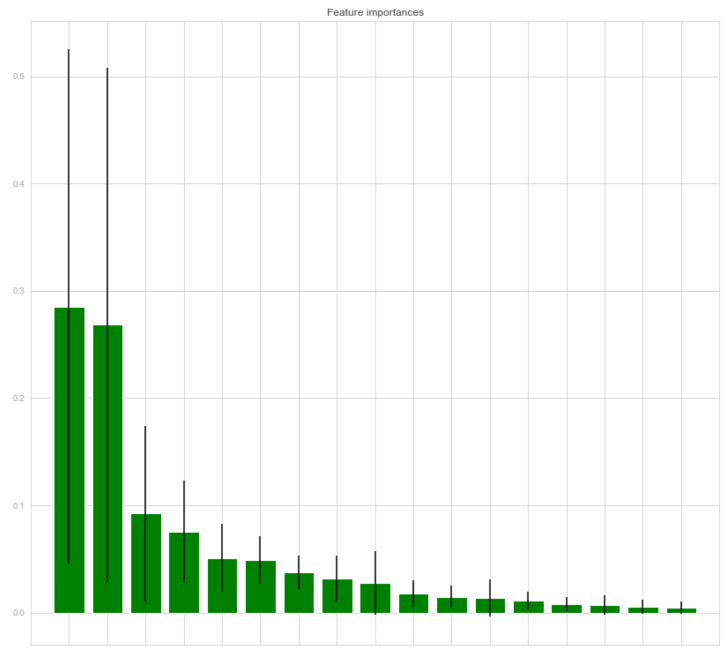
```
3
[False  True False  True False False False False False False False False
 False  True False False False]
[10  1  3  1  8 15  9 14 13  7  5 12  2  1  6  4 11]
```



Random forest feature importance

Random forests are among the most popular machine learning methods thanks to their relatively good accuracy, robustness and ease of use. They also provide two straightforward methods for feature selection: mean decrease impurity and mean decrease accuracy.

Random forest consists of a number of decision trees. Every node in the decision trees is a condition on a single feature, designed to split the dataset into two so that similar response values end up in the same set. The measure based on which the (locally) optimal condition is chosen is called impurity. For classification, it is typically either Gini impurity or information gain/entropy and for regression trees it is variance. Thus, when training a tree, it can be computed how much each feature decreases the weighted impurity in a tree. For a forest, the impurity decrease from each feature can be averaged and the features are ranked according to this measure.



```
Feature ranking:
1. feature 15 (0.285119)
2. feature 3 (0.268286)
3. feature 8 (0.092197)
4. feature 13 (0.075183)
5. feature 14 (0.050717)
6. feature 10 (0.048827)
7. feature 6 (0.037779)
8. feature 9 (0.031538)
9. feature 1 (0.027840)
10. feature 0 (0.017622)
11. feature 2 (0.014626)
12. feature 4 (0.013943)
13. feature 12 (0.011219)
14. feature 7 (0.007503)
15. feature 16 (0.007189)
16. feature 11 (0.005600)
17. feature 5 (0.004812)
```

In our example, this algorithm recommends future 15,3,8,13,14 to be the final portfolio

Lasso Feature Selection Algorithm:

The LASSO (Least Absolute Shrinkage and Selection Operator) is a method of automatic variable selection which can be used to select predictors X* of a target variable Y from a larger set of potential or candidate predictors X.

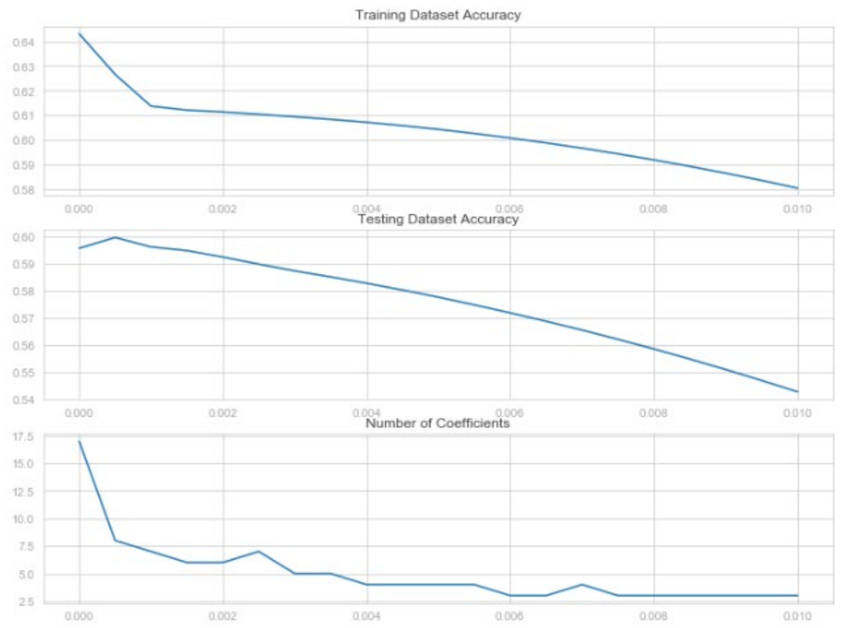
Developed in 1996 by Tibshirani, the LASSO formulates curve fitting as a quadratic programming problem, where the objective function penalizes the absolute size of the regression coefficients, based on the value of a tuning parameter λ. In doing so, the LASSO can drive the coefficients of irrelevant variables to zero, thus performing automatic variable selection.

The cost function for Lasso (least absolute shrinkage and selection operator) regression can be written as:

$$\sum_{i=1}^{M}(y_i-\hat{y}_i)^2=\sum_{i=1}^{M}\left(y_i-\sum_{j=0}^{p}w_j\times x_{ij}\right)^2+\lambda\sum_{j=0}^{p}|w_j| \qquad \text{For some } t>0,\ \sum_{j=0}^{p}|w_j|<t$$

Lasso regression not only helps in reducing over-fitting, but it can help us in feature selection. The difference of lasso and ridge regression is that some of the coefficients can be zero i.e. some of the features are completely neglected

In this example, the algorithm gives the choices for future 1,3,4,9,12,15

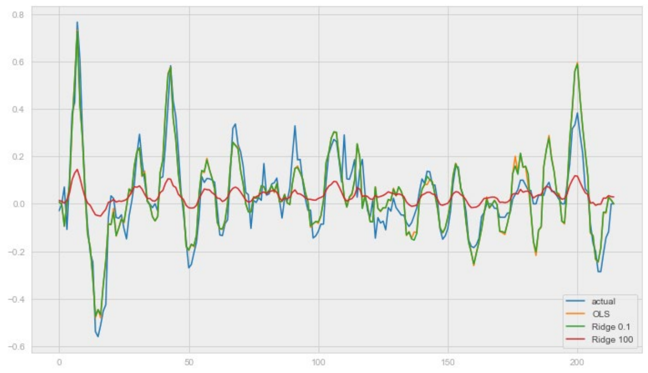


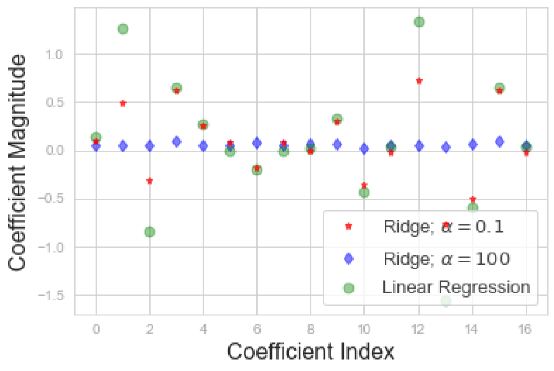
```
[ 0.          0.05630997  0.          0.79030419  0.          0.09936026
  0.          0.          0.          0.1694753  -0.          0.
  0.03622296  0.          0.          0.04046697  0.        ]
```

Ridge Regression:

In ridge regression, the cost function is altered by adding a penalty equivalent to square of the magnitude of the coefficients. The purpose here is not for figure selection, but to find a lambda that will minimise the magnitude of coefficients without significantly affecting acuuracy.

$$\sum_{i=1}^{M}(y_i-\hat{y}_i)^2=\sum_{i=1}^{M}\left(y_i-\sum_{j=0}^{p}w_j\times x_{ij}\right)^2+\lambda\sum_{j=0}^{p}w_j^2 \qquad (1.3)$$





As we can see, ridge regression with alpha=0.1 did not change the fitting result but significantly reduced the size of coefficients of several variables. This means we just need to purchase less amount of options which reduce the total cost.

In concluision, from step 2, we have chosen 1,3,8,9,13

Section 3: Cost-profit analysis

In this section, I aim to investigate whether the implementation of hedging portfolio will reduce total payoff of our product. I will also calculate payoff of all combination of possible hedging indexes and then find the best hedging portfolio. Before go into the details, I will explain how to simulate the price of future option using Black-Scholes model.

Payoff_total = Payoff_RU3 - Payoff_Matif + Cost_matif_option

Black-Scholes Pricing for futures options:

The original Black–Scholes model has undergone several theoretical developments. One such development for the valuation of futures options is introduced by Black (1976). Black proposed a formula for options under the assumption that investors generate risk less hedges between options and the futures or forward contracts. The problem of negative cost of carry was addressed by using 'forward prices' in the option pricing model instead of 'spot prices'. Black observed that actual forward prices not only incorporate cost of carry but also takes into account other irregularities in the market.

In his proposed model, he substituted spot price (S) by the discounted value of future price (F.e-rt) in the original Black-Scholes model. Black's model found application in valuing options on physical commodities where future price is a better alternative input for valuing options. The Call options prices as per Black's formula can be observed solving following equation:

$$C = Fe^{-rt}.N(d_1) - X.e^{-rt}.N(d_2)$$

$$= e^{-rt}\left[F.N(d_1) - X.N(d_2)\right]$$

$$\text{Where d1 } = \frac{\ln\left(F/X\right)+\left(\sigma^2/2\right)t}{\sigma\sqrt{t}}$$

$$\text{d2 } = \frac{\ln\left(F/X\right)-\left(\sigma^2/2\right)t}{\sigma\sqrt{t}} = d_1 - \sigma\sqrt{t}$$

In the formula F is the future price of the asset and other input parameters are similar to the inputs used in the Black-Scholes model.

Volatility Calculation for Black-Scholes model:

The past volatility of a security can be estimated as the standard deviation of a stock's returns over a predetermined number of days. Choosing the appropriate number of days is complicated. Longer period of observation has an averaging effect and as volatility varies with time and very old data may not be relevant for the current situation and can not be used for predicting the future. In absence of an agreed method to estimate volatility to be used in options pricing models, a simple method of estimating standard deviation using **past three months return** was used in the study.

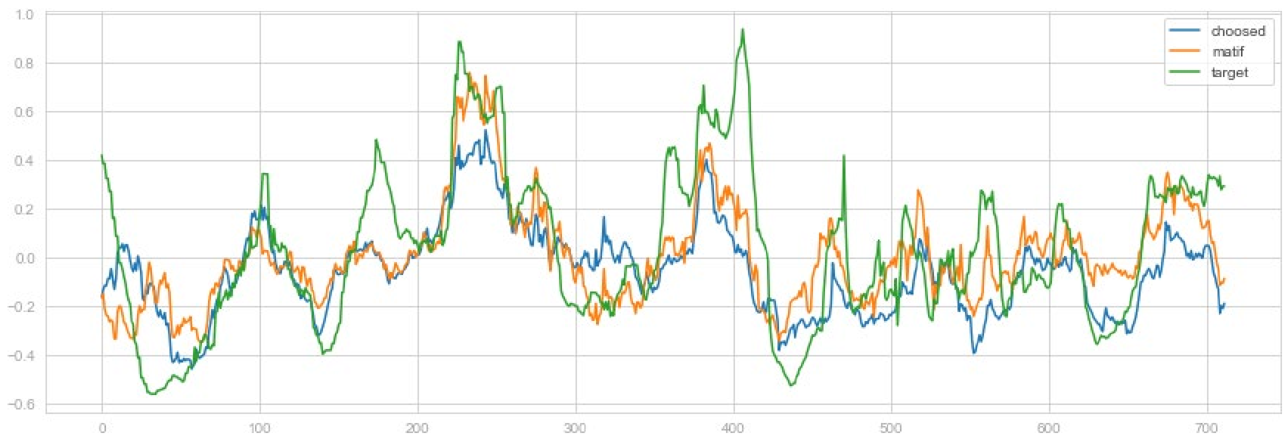
$$\sigma_{annualised} = \sigma_{daily}\sqrt{Trading\ days\ per\ annum.}$$

In our example, time is in monthly unit, so I need to calculate monthly volatility. $\sigma_{month} = \sigma_{weekly} \times \sqrt{8.5}$

Final Portfolio:

By find all the combination of chosen products and calculate the average premium ratio. Then I choose the one which gives the minimum premium ratio.

```
[('[3]', 0.04110108180387846),
 ('NA', 0.04188748031725115),
 ('[1, 3]', 0.06953862009586523),
 ('[1]', 0.08135003605383152),
 ('[3, 4]', 0.09215848142381769),
 ('[0, 3, 4]', 0.09253025959088221),
 ('[0, 3]', 0.09271179688964812),
 ('[1, 3, 4]', 0.09766739113207719),
 ('[1, 2, 3, 4]', 0.09869217918305152),
 ('[1, 2, 3]', 0.09899943819611287),
 ('[0, 1, 3]', 0.10110276070601187),
 ('[0, 1, 2, 3]', 0.10129101788902166),
 ('[0, 1, 3, 4]', 0.10147845145616168),
 ('[0, 1, 2, 3, 4]', 0.10540110653945797)
 ('[1, 2, 4]', 0.11716565062397222),
 ('[1, 4]', 0.11884280256432304),
 ('[0, 1, 2]', 0.12025085466694804),
 ('[0, 1]', 0.12194221155225175),
 ('[0, 1, 4]', 0.1224333856325576),
 ('[0, 1, 2, 4]', 0.1240931905866737),
 ('[1, 2]', 0.1277013032705797),
 ('[2, 3]', 0.17605940820420835),
 ('[2, 3, 4]', 0.17835937250562128),
 ('[0, 2, 3]', 0.1803118497451987),
 ('[0, 2, 3, 4]', 0.1842531655432294),
 ('[0, 4]', 0.18766010313450743),
 ('[4]', 0.18867104213401306),
 ('[0]', 0.19351651289123387),
 ('[2, 4]', 0.2875559183905271),
 ('[0, 2]', 0.2913851171549114),
 ('[0, 2, 4]', 0.29415173594036725),
 ('[2]', 0.3022840833869181)]
```



So we choose the blue line to hedge rather than matif.