Machine Learning Engineer Nanodegree

Capstone Project

Gregorio Polito
May 15th, 2020

I. Definition

Project Overview

The current project proposal aims to solve one of the most interesting problems in finance: Stock Price prediction. Such problem is assuming a central role in finance for both trading and risk-management purposes in order to ensure that no future crises would be unpredictable. The use of the Machine Learning is providing a way to success. Several hedge funds and banks are using Machine Learning for stock price prediction and are generating higher earnings. The use of Machine Learning helps finding patterns in the dataset which may help in a more accurate prediction of the future prices. In addition, such predictions are commonly applied for the portfolio optimization (whose aim is to find the best way to combine stocks in order to maximize a metric of interest, e.g. utility function, log-term return, Sharpe Ratio, etc.).

Sometimes only large corporates and hedge funds can account for Machine Learning algorithms for investment purposes, but everyone can be interested in good stock prices predictors in order to assure the profitability of the own investments: such solutions could help people in finding the best strategy.

This project will provide an answer to the portfolio optimization problem, focusing on a “forward optimization” since the optimal solution will be assessed on the simulated portfolio value. The portfolio is assumed made of the main European Stock Indices and the optimal strategy consists in the maximization of the Sharpe Ratio (Wikipedia, Sharpe ratio, 2020) which represents a trade-off between Portfolio returns and volatility.

Problem Statement

Taking into account a portfolio of Stocks, defined through the couples (S_i, w_i) where at each stock S_i is assigned a weight w_i such that the sum of the weights is 1 (each weight can be either positive or zero), the problem is to find the best weights combination in order to assure higher annualized Sharpe ratio over a given period. In particular, the problem is divided into the following components

1- Stock prices forecasting: for each stock in perimeter the Adjusted Closing Stock price is predicted on a future date T
2- Taking into account the predictions computed in 1., the portfolio composition (i.e. the stock weights w_i) is optimized in order to maximize the annualized Sharpe Ratio.

Once the portfolio is predicted on a future date, obtaining different paths, the Sharpe Ratio is computed as a function of the weights and the optimal value is found across the different weights combination.

Metrics

The assessment of both performance and results needs to the divided into the prediction and optimization components.

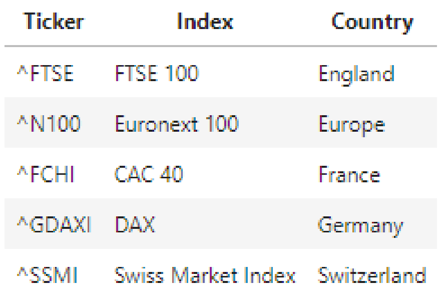
For the evaluation of the prediction component, the daily prediction is performed and the predicted quantiles are compared with the actual stock value observed during the prediction horizon. If the actual stock price systematically exceeds the predicted quantiles, the performance is assumed bad, otherwise the performance is good. For this kind of problems, it is difficult to assess the goodness of the prediction performance in a rigorous quantitative way: in case of crises, the actual prices may reach levels which have never be observed in the history. On the other hand, the performance of the training part of the prediction process will be assessed in terms of the Root mean squared error (RMSE) in order to access the error between the prediction and the actual price.

The performance of the optimization is assessed in terms of the output of the optimization function which highlights if the optimization algorithm succeeds.

II. Analysis

Data Exploration

The dataset used for the prediction component consists in the historical closing prices over the training time window for the main European Stock Indices. The whole dataset is retrieved from Yahoo! Finance APIs (i.e. *yfinance* python module). In particular, the dataset is made of the following stocks



Ticker	Index	Country
^FTSE	FTSE 100	England
^N100	Euronext 100	Europe
^FCHI	CAC 40	France
^GDAXI	DAX	Germany
^SSMI	Swiss Market Index	Switzerland

For each of the tickers, the dataset is made of the daily stock values such as Open, High, Low, Close, Volume and Adjusted Close and only the last value is used for the calibration and prediction (in the table below an example of the extraction for *^FTSE*

	Open	High	Low	Close	Adj Close	Volume
Date						
2010-01-04	5412.899902	5500.299805	5410.799805	5500.299805	5500.299805	750942000
2010-01-05	5500.299805	5536.399902	5480.700195	5522.500000	5522.500000	1149301200
2010-01-06	5522.500000	5536.500000	5497.700195	5530.000000	5530.000000	998295300
2010-01-07	5530.000000	5551.700195	5499.799805	5526.700195	5526.700195	1162933700
2010-01-08	5526.700195	5549.299805	5494.799805	5534.200195	5534.200195	1006420600

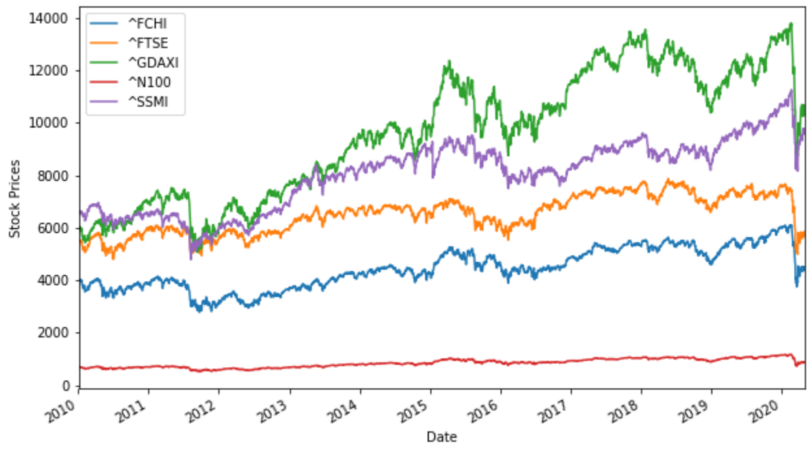
Taking into account the whole dataset, the table below shows some statistics

	count	mean	std	min	25%	50%	75%	max
^FCHI	2635.0	4439.090338	769.757224	2781.679932	3838.660034	4391.500000	5117.479980	6111.240234
^FTSE	2600.0	6517.021535	724.781606	4805.799805	5885.274902	6605.099854	7164.175049	7877.500000
^GDAXI	2612.0	9601.207649	2400.171326	5072.330078	7225.294922	9795.264648	11829.317627	13789.000000
^N100	2635.0	847.870132	162.153685	529.500000	699.154999	849.950012	995.795013	1182.099976
^SSMI	2590.0	8038.267254	1306.321163	4791.959961	6627.824951	8308.995117	8984.087158	11263.009766

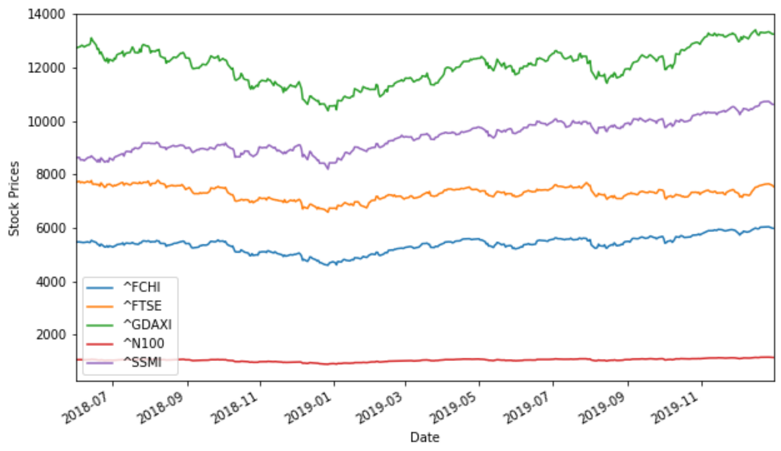
It can be notices that some data are missing due to the difference in business days over the different Indices which will be addressed in the data preprocessing phase.

Exploratory Visualization

The whole dataset is represented in the chart below



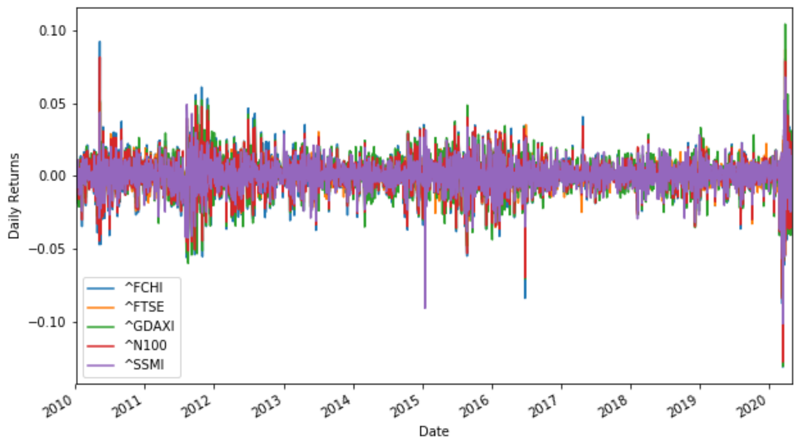
with a deep focus on the last years (since June 2018, the dataset used for the training)



It can be noticed that

- no periodic patterns can be highlighted due to the fact that usually stock indices are not affected by cyclical events
- the general trend is increasing with some downward movements
- the stock prices refer to different scales (e.g. ^GDAXI is about 12,000 whereas ^N100 is less than 2,000)

In particular, since the optimization problem will be solved in terms of the daily returns of the portfolio made of the above-mentioned indices, the chart below focus on the daily log returns



for the plot above, the following considerations can be made:

- strong correlation between the different returns
- presence of volatility clustering (periods of high (respectively low) volatility are followed by period of high (low) volatility)
- the returns are approximatively normal (hypothesis used in the definition of the Sharpe Ratio)

Algorithms and Techniques

The Stock prices prediction uses the Amazon SageMaker DeepAR (AWS, 2020) which is a supervised learning algorithm based on recurrent neural networks (RNN).

On the other hand, the portfolio optimization is performed computing the optimal weights composition in order to maximize the annualized Sharpe Ratio on a given future date. Such metrics takes as input the Expected Value and the Standard Deviation of the daily Excess Return computed at portfolio level. Considering the simulated Stock prices S_i^T in T, the current spot price S_i observed in t_0 and the portfolio weights w_i, the Sharpe Ratio is computed according the following steps (for sake of simplicity, the risk-free rate is assumed 0%)

1. Daily Excess Return of the predicted portfolio on a single predicted path
$$\frac{1}{T - t_0} \cdot \sum_i w_i \cdot ret_i^T$$
2. Expected Value $\mathbb{E}$ of the predicted Excess Return (computed over the whole set of simulated values)
$$\mathbb{E} = E_{Sim}\left[\frac{1}{T - t_0} \cdot \sum_i w_i \cdot ret_i^T\right] = \sum_i w_i \cdot E_{Sim}\left[\frac{1}{T - t_0} \cdot ret_i^T\right]$$
3. Standard Deviation σ of the predicted Excess Return (computed over the whole set of simulated values)
$$\sigma = \sqrt{Variance_{Sim}\left[\sum_i ret_i^T \cdot \frac{1}{T - t_0} \cdot w_i\right]}$$
4. Annualized Sharpe Ratio (considering a year made of 252 business days)
$$\frac{\mathbb{E}}{\sigma}$$

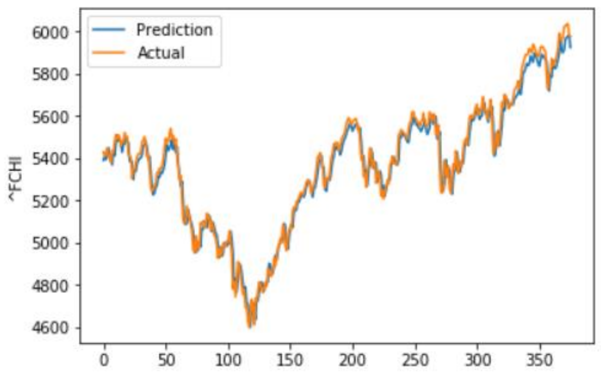
where $ret_i^T = \log\left(\frac{S_i^T}{S_i}\right)$ is the log-return of the i-th index observed over the period between t_0 and T. The optimal portfolio composition is obtained maximizing the Sharpe Ratio over the different weights combination.

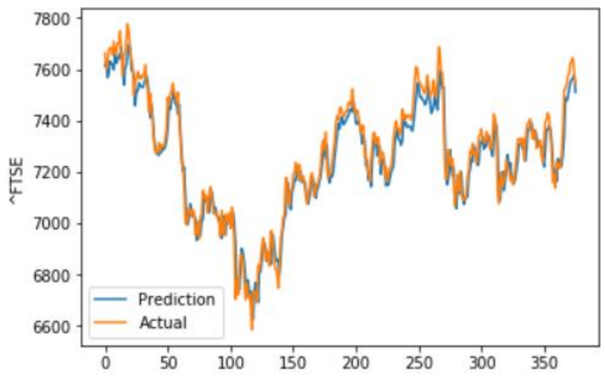
Benchmark

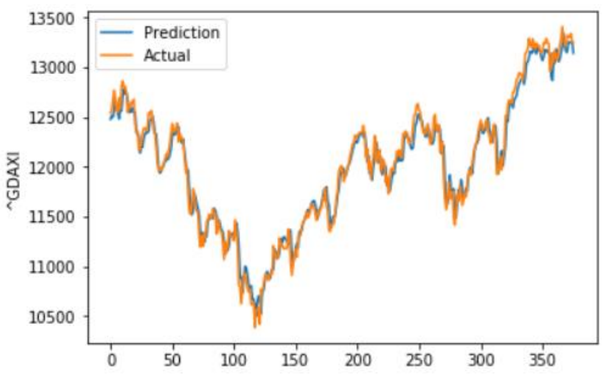
The benchmark will be applied only to the prediction part of the problem. The DeepAR outcome will be compared with the one performed through a different network, in particular with another recurrent neural network which is based on the Long short-term Memory (Wikipedia, Long short-term memory, 2020) architecture. According the Keras documentation (Keras, 2020), the network is defined concatenating 3 layers: LSTM, Dropout and Dense.

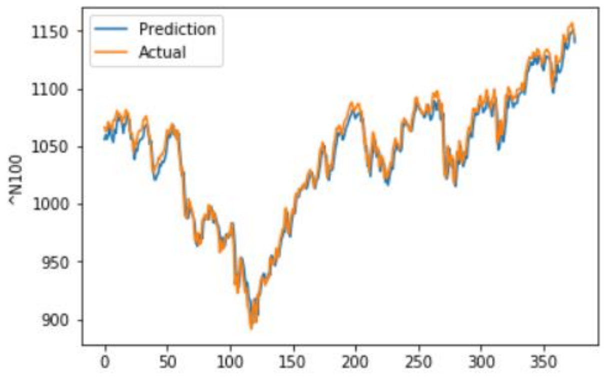
The benchmark is trained on the same dataset and the metric used for the evaluation of the performance is RMSE as well.

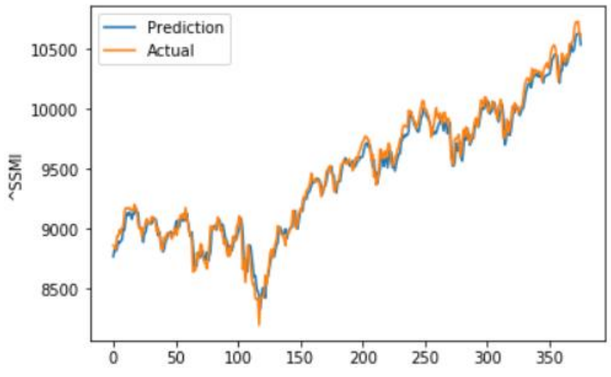
The outcome of the training process is reported below. In each chart, the predicted stock price is compared with the actual one on the whole dataset used in the training process and highlights a good performance of the model.











III. Methodology

Data Preprocessing

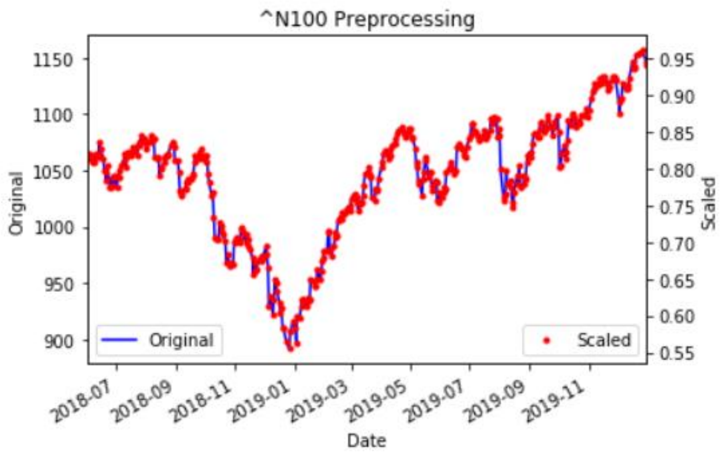
As previously highlighted in the Data Exploration section, the dataset used has some missing data and the different scale of the indices. Both issues can generate an impact on both training and prediction activities, in particular using a common model for the different stock prediction these issues could generate problems in the predictions.

According the data missing issue, the data prices which are not available are filled using a “forward method”:

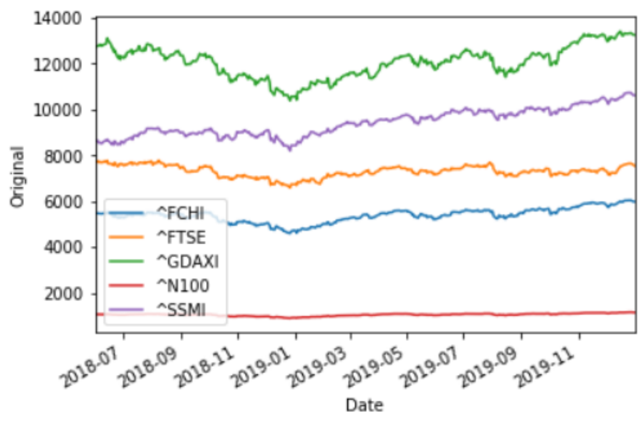
- the missing data is substituted with the last available data
- otherwise, if the first data point is missing, the second value is considered

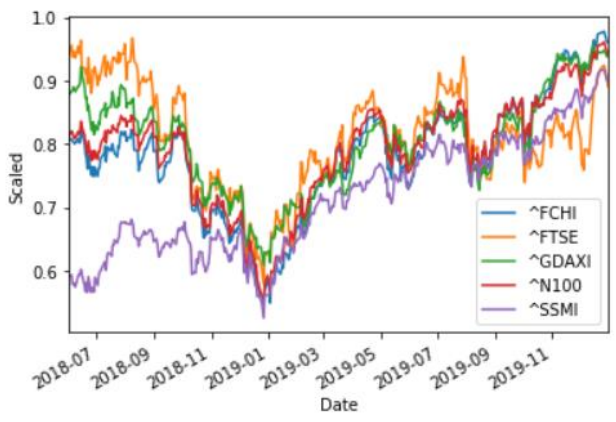
Regarding the different scale, this could lead to an underestimation of stocks with the lower scale. In particular the scaler used is the sklearn.preprocessing MinMaxScaler class, which scales the dataset according the minimum and maximum values.

Please find below an example of the application of the preprocessing part for the Index with the lowest scale (^N100)



and the effect of the scaling process to the whole dataset is represented in the charts below (without (on the left) and with (on the right) the scaler).





Implementation

The solution of the optimization problem is made of two components: the prediction and the optimization.

The prediction part is performed using the DeepAR model which predicts the index prices on a given horizon. In particular the model is configured to perform daily predictions over a month. To this end, the whole dataset is divided into the training and testing sets starting from June 2018 up to end of 2019. For the training set, the last month is omitted since such period will be predicted by the model, whereas the testing set is made of the whole dataset (including the last month of observation). The two datasets previously defined are uploaded to the S3 storage, once converted in the JSON format as defined in the DeepAR documentation. In order to take into account the simulation of the different stocks, the *cat* variable is used in order to label each different stock.

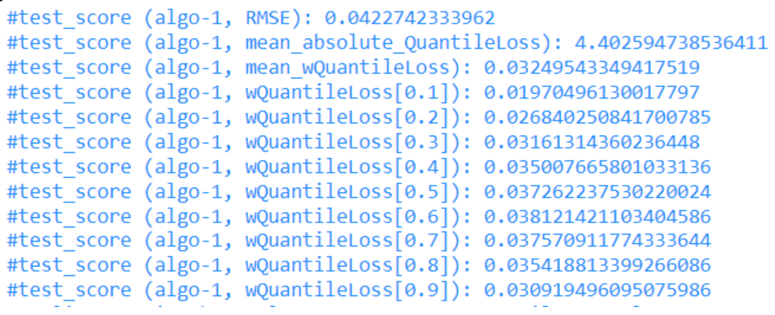
Taking into account the DeepAR hyperparameters, the daily frequency and the prediction 1-month prediction horizon are respectively encoded through the variables *time_freq*, *prediction_length* (and *context_length* set equal to 1 month in order to maximize the prediction performance).

Once the model is trained and deployed, the prediction is performed. According the DeepAR documentation, the prediction is performed through a JSON file containing the instances and configurations of the prediction. The instances has 3 keys fiels: *start* (start date of the prediction), *target* (input time series, if available) and *cat* (equity to be predicted). Regarding the configuration, *quantiles*, *mean* or *samples* are computed over a *num_samples* paths. The output of the prediction request is in JSON format and contains the requested prediction starting from the start date up to 1 month later with daily observations.

Using the predictions previously described, the portfolio optimization takes place. As defined in the Algorithms and Techniques section, the daily Excess Return is computed (taking into account the log-returns observed over 1-month of prediction for each sample). Then the Sharpe Ratio is computed and the function is optimized taking into account the different weights compositions.

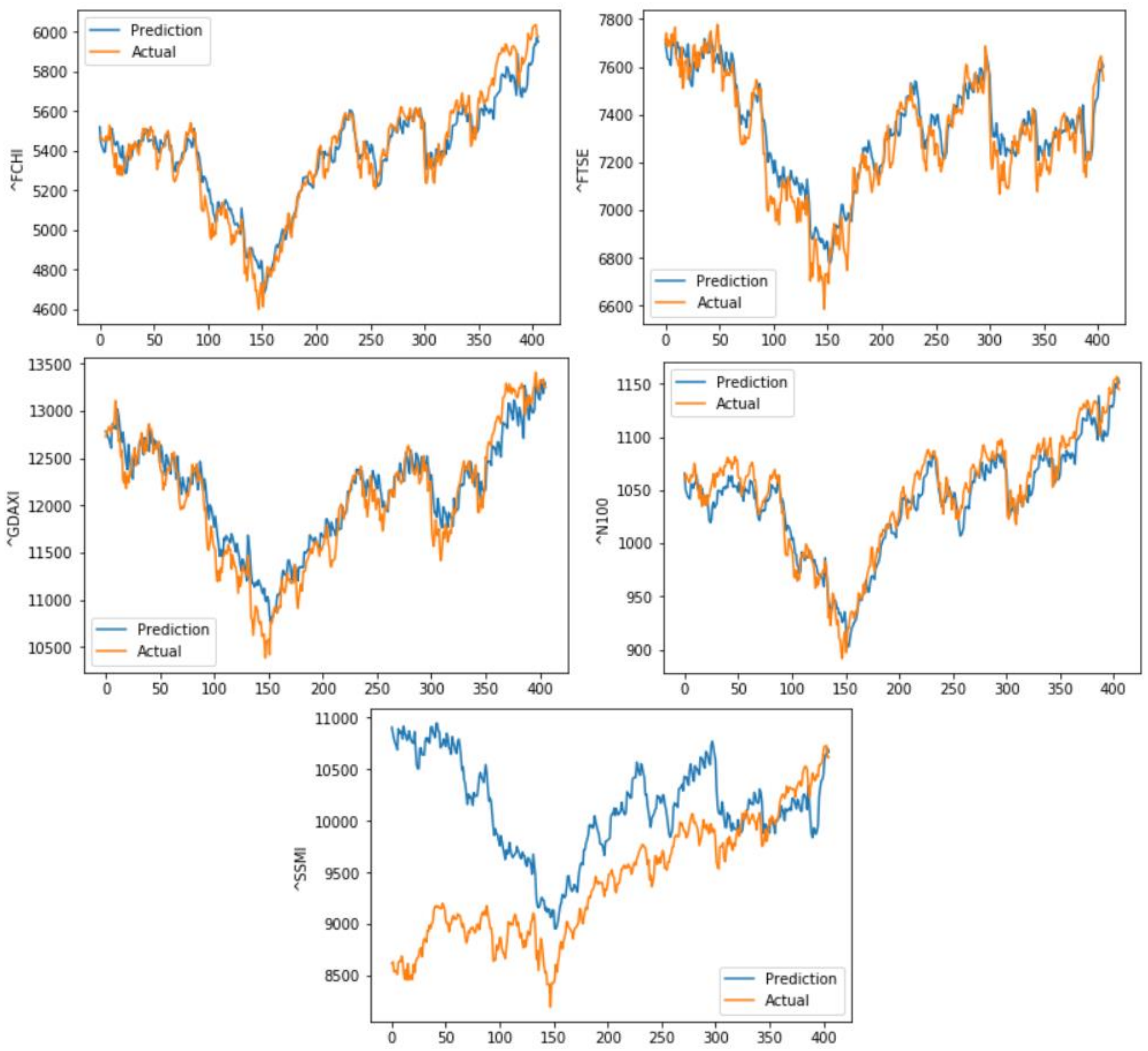
Refinement

Apart from the indices scaling (using the MinMaxScaler), which was straightforward, the hyperparameters used in the DeepAR model resulted relevant for the performance of the model. The model parameters have been set up according the DeepAR documentation. Please find below the outcome of the training job (which is identified by name *forecasting-deepar-2020-05-14-11-55-16-363*)



```
#test_score (algo-1, RMSE): 0.0422742333962
#test_score (algo-1, mean_absolute_QuantileLoss): 4.402594738536411
#test_score (algo-1, mean_wQuantileLoss): 0.03249543349417519
#test_score (algo-1, wQuantileLoss[0.1]): 0.01970496130017797
#test_score (algo-1, wQuantileLoss[0.2]): 0.026840250841700785
#test_score (algo-1, wQuantileLoss[0.3]): 0.03161314360236448
#test_score (algo-1, wQuantileLoss[0.4]): 0.035007665801033136
#test_score (algo-1, wQuantileLoss[0.5]): 0.037262237530220024
#test_score (algo-1, wQuantileLoss[0.6]): 0.038121421103404586
#test_score (algo-1, wQuantileLoss[0.7]): 0.037570911774333644
#test_score (algo-1, wQuantileLoss[0.8]): 0.035418813399266086
#test_score (algo-1, wQuantileLoss[0.9]): 0.030919496095075986
```

and the qualitative outcome of the prediction



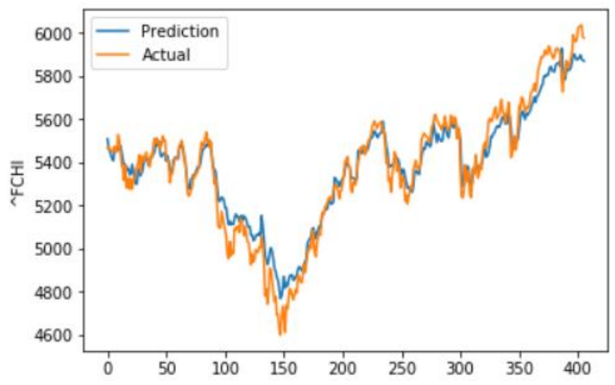
which clearly shows some issues on the training process (which can be easily seen in the comparison between Prediction and Actual for the ^SSMI index).
Hence the Hyperparameters needed some adjustments in order to increase the prediction performance. This is achieved through the Hyperparameter Tuner focusing the attention on parameter ranges suggested in the DeepAR documentation. The best solution is found training the Tuner on the same dataset previously used for the DeepAR training. Then the model is deployed taking in the account the best tuning solution (which corresponds to *forecasting-deepar-200514-1301-005-9687d922*).
The updated qualitative comparison between Prediction and Actual is reported in the next session.

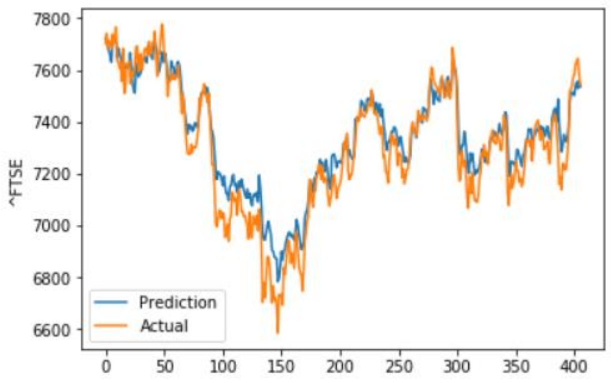
IV. Results

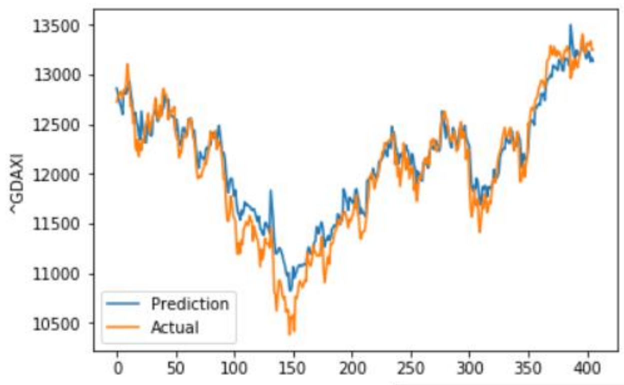
Model Evaluation and Validation

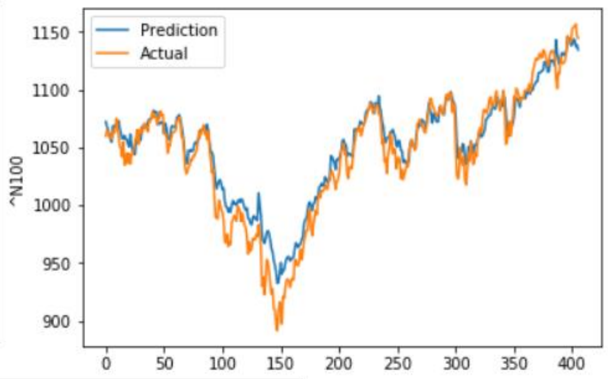
In this section, the model is assessed in terms of the prediction performance and optimization. The use of the DeepAR model allows to predict a distribution rather than only one sample. Consequently, a forward looking portfolio optimization can be obtained.

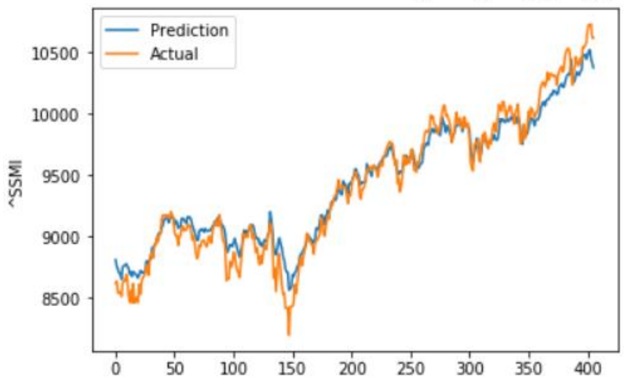
Regarding the prediction part, the charts below show the model performance in qualitative terms, the predicted index (1-month later the prediction trigger date) is compared with the one obtained through the expected value simulated through the DeepAR model (the whole perimeter of indices is reported)





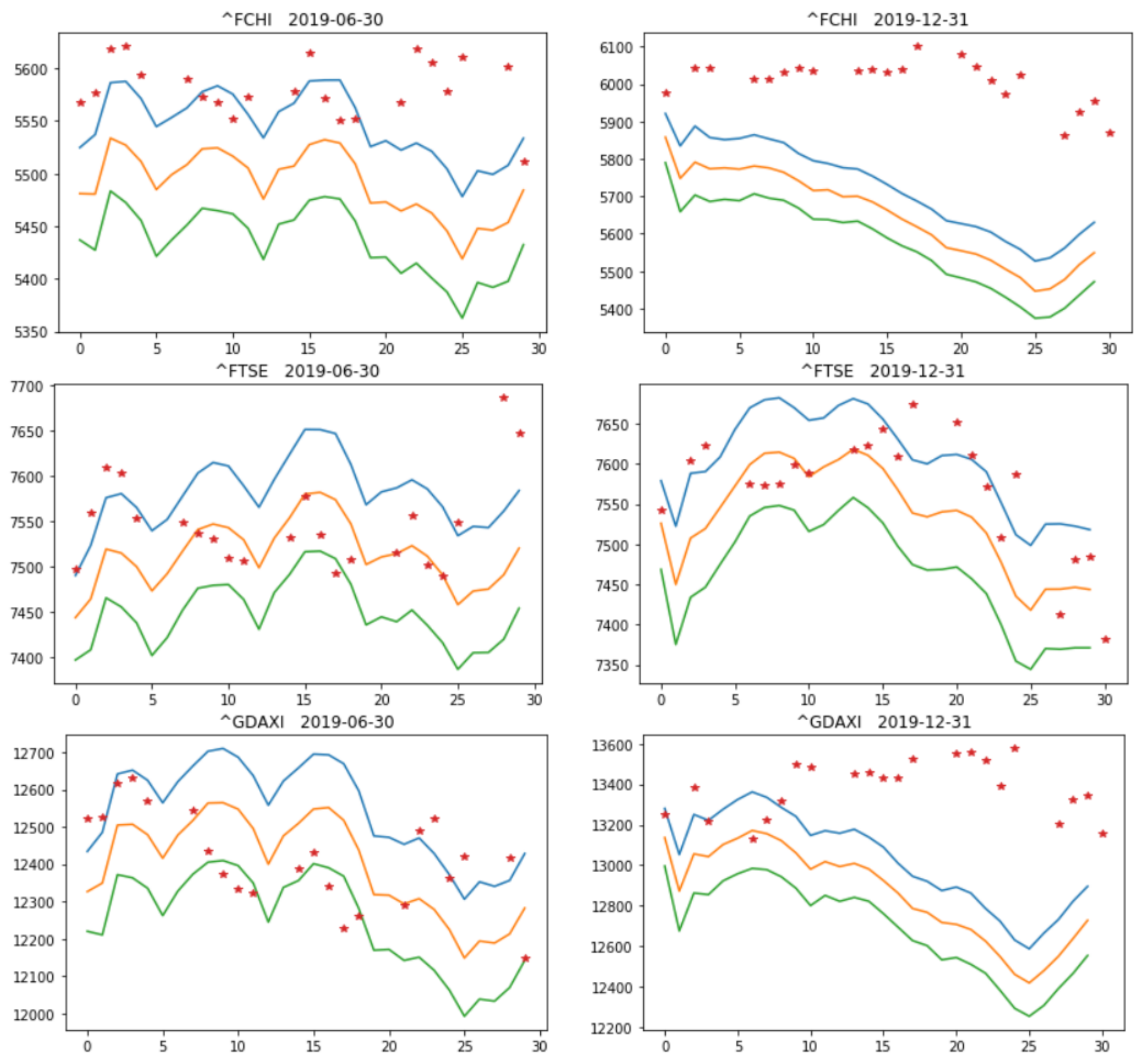


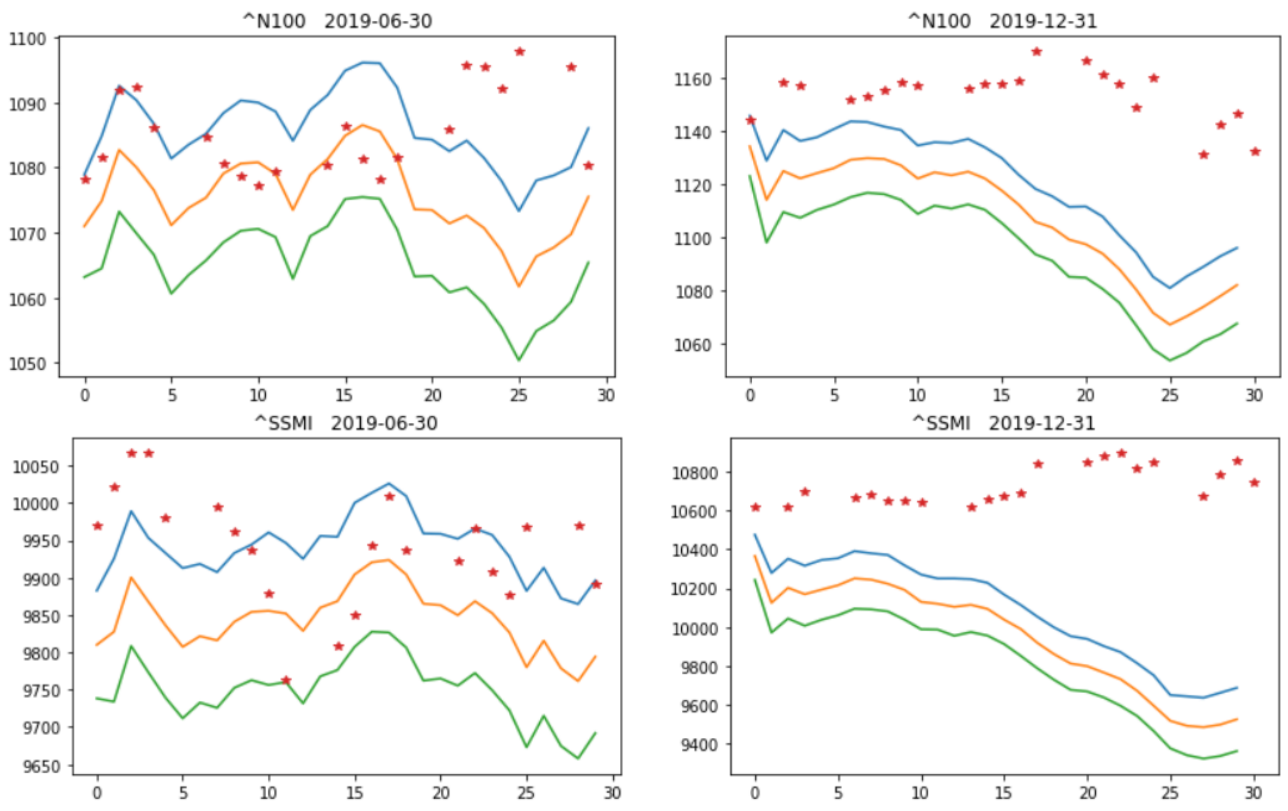




Please note that the ^SSMI plot shows a significant improvement in the first part, thanks to the Hyperparameter tuning.

In addition one of the key features of the DeepAR model is the ability to simulate a distribution of samples which allows to compare the prediction performance in a wider way: the simulation of a distribution, allows to check the model performance in terms of the percentiles. Taking into account such percentiles, the model allows to check the degree of error from a probabilistic perspective. The charts below show the ability of the model to perform both in case of prediction start date inside the training set and just after the set: this are two good examples to assess to performance ability of the model in the training set and outside. In case of a prediction start date inside the training set (30th June 2019), the predicted distributions (5th, 95th percentiles and median, respectively in green, blue and orange) well represents the actual values (red stars). Concerning the predicted profile outside the training set, the predicted quantiles do not highlight anomalous distributions (e.g. spikes or median which reaches zero or infinity) and in some cases are in line with the actual prices.





Regarding the optimization part of the problem, the maximization of the Sharpe Ratio is performed on the simulated distribution and this allows to have a forward-looking approach in the portfolio optimization (this aspect will be detailed in the Conclusion chapter).

Justification

As previously defined in the Benchmark section, only the prediction component of the problem can be subject to benchmark. Regarding the optimization part, the positive outcome of the optimization problem (with *success* status) assures that this kind of problem is correctly solved.

Concerning the prediction part, the use of a benchmark is needed in order to assess the performance of the network compared with another simpler case. In particular the table below reports the Root Mean Squared Error computed in the training process

	RMSE
LSTM	0.0138
DeepAR	0.0411

It can be noticed that both statistical results are low and the Benchmark seems performing better than the DeepAR model. The reason behind such difference is how the RMSE is computed in the different cases:

- Benchmark (LSTM network): the RMSE is defined according the Keras documentation as “the mean of squares of errors between labels and predictions”
- DeepAR network: according the documentation DeepAR RMSE, the model performance is assessed also in terms of the Daily prediction over 1-month rather then only on the simulated 1-month point. In addition, the DeepAR RMSE uses as prediction the mean of the prediction distribution

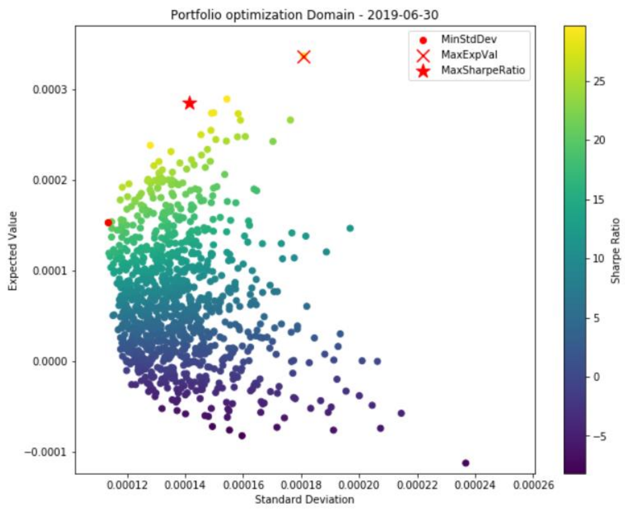
Consequently even if the DeepAR model shows a RMSE higher than the benchmark, the outcome of the model fulfills the expectations and gives a stronger solution due to the nature of the model (see the Conclusion chapter for further details).

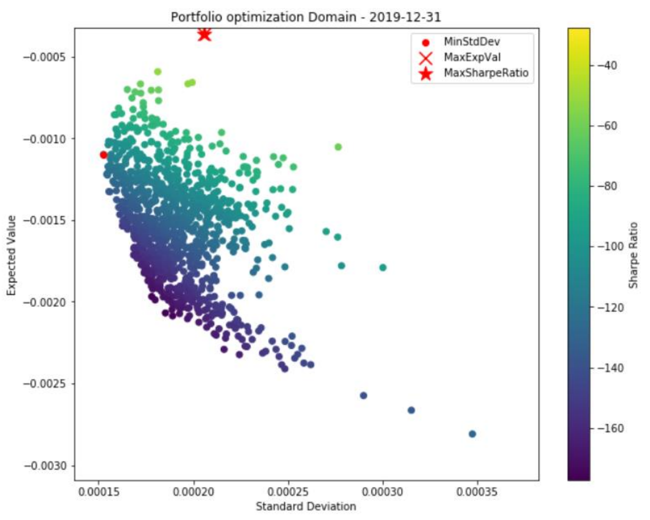
V. Conclusion

Free-Form Visualization

The following section focuses the attention on the result of the optimization problem: the Sharpe Ratio optimization. Simulating the portfolio on a given future date, thanks to DeepAR, it is possible to estimate the Expected Value and the Standard Deviation computed on the Excess Returns over the predicted distribution. The portfolio composition is randomly sampled in order to have a flavor of the optimization problem.

The chart below shows from a graphical point of view the domain of the optimization problem (for both the simulation dates)





each point represents the Sharpe Ratio of the portfolio given the Expected Value and Standard deviation of the portfolio. In addition the red star highlights the optimal Sharpe Ratio. This is an interesting result if compared with other possible optimization strategies. The most simple ways of optimization are represented by the maximization of the Portfolio Expected Return, highlighted with a red cross (numerator of the Sharpe Ratio function), and the minimization of the Portfolio Standard Deviation, highlighted with a red point (denominator of the Sharpe Ratio function). It can be noticed that the Sharpe Ratio maximization represents a trade-off between the above mentioned strategies. In addition, comparing the two charts, it can be noticed how the optimal solution is found even in different market conditions (i.e. positive (chart on the left) and negative (chart on the right) expected returns): in case of negative returns, which is the worst case scenario, the maximization of the Sharpe Ratio approximatively corresponds to the maximization of the expected returns (even if they are negative).

Reflection

The aim of this project is to solve an optimization problem, in particular the portfolio allocation in order to maximize the Sharpe Ratio. Typically, the portfolio optimization is performed analyzing the portfolio historical time series in order to include the historical variability in the estimation of the future portfolio. On one hand, such approach is “model-free” since there is no assumption on the portfolio evolution (since it is based on the history) but, on the other hand, in this case it is not possible to optimize the portfolio in the future.

The DeepAR network provides a valid solution in the forward-looking portfolio optimization. It represents a trade-off between the portfolio prediction on a future date and the use of the available information observed in the history. This is a powerful tool which allows to estimate in time the portfolio value without assuming a predefined dynamic (as it typically happens in case of stochastic models which assign a predefined dynamic to the portfolio evolution through time), consequently such approach can be thought “model-free” like the pure historical approach.

This project provides a way to solve the hard problem of the portfolio optimization starting from the daily Indices prices, up to the definition of an optimal strategy of asset allocation.

Improvement

According the powerful result, this project represents a starting point of the asset allocation in the real-world. To this hand several improvements could be taken into account

- use of a larger set of Indices
- introduction of fees related to changes in the portfolio weights which impacts the optimization function
- use of a risk-free rate different than zero in the definition of the Excess Return
- optimization of different metrics rather than the only Sharpe Ratio

Regarding the last bullet, different optimization function could be used. For example, each user could define the own risk profile in order to choose a custom optimization function: for risky users, the optimization could focus on selecting the most volatile stocks with the higher return, on the contrary for users which do not prefer risky strategies, the model could focus on the selection of less volatile stocks with significant returns.

References

AWS. (2020). *DeepAR Forecasting Algorithm*. Retrieved from https://docs.aws.amazon.com/sagemaker/latest/dg/deepar.html

Keras. (2020). *LSTM layer*. Retrieved from https://keras.io/api/layers/recurrent_layers/lstm/

Wikipedia. (2020). *Long short-term memory*. Retrieved from https://en.wikipedia.org/wiki/Long_short-term_memory

Wikipedia. (2020). *Sharpe ratio*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Sharpe_ratio