Volo-Wizard

April 8, 2022

1 Introduction

Usage of machine learning methods is among hot topics in Financial Engineering. Its applications ranges from clustering of different stocks according to their idiosyncratic differences, to modeling time-series financial data using neural networks. This report aims to give a detailed information about the Volo-Wizard, a machine learning software for predicting the volatility of S&P500 index.

2 Motivation

One of the major research fields in quantitative finance is to understand the nature of financial market returns: their direction, their magnitude and their relation with other financial indicators. After long years of financial research, it has been found that although direction of market returns are almost random, their magnitude, which is also called volatility is not. More explicitly, there is a considerable auto-correlation between daily volatility of most risky assets (stock, index, options and etc.). S&P500 index being the weighted sum of top 500 stocks also follows this phenomenon of autocorrelation between consequetive days. This finding has been the main motivation of this project. The aim is to learn the underlying pattern of volatility autocorrelation using machine learning methods and use it to make forecasts.

3 Dataset

S&P 500 index data from 1967 to 2022 has been used for training and testing the model. The source of the data is a website [1]. The dataset is composed of 13631 data points and 5 columns. Here are the columns and their descriptions:

1) `Close` - price level at the end of each day.
2) `Open` - price level at the start of each day.
3) `High` - the highest price level for each day.
4) `Low` - the lowest price level for each day.
5) `Volume` - trade volume for each day

The specific information that we used for training and testing is the return data (in percentage). It has been extracted from the `Close` column of the original dataset. The formula for the return in time frame t is $R_t = 100 * \frac{P_t - P_{t-1}}{P_{t-1}}$ where P_t is the price in time frame t and R_t is the return in time frame t.

The dataset has been divided into 2 sections: Train and Test. 80% (10891 data oints) of the data was used for training while rest (2725 data points) was used for testing. In addition, 10% of the training data was used as the validation set for hyperparameter tuning.

4 The Model

The underlying machine learning model is an LSTM (Long Short Term Memory) model, a special kind of RNN (Recurrent Neural Network) to learn complex time-series patterns. The model is composed 3 layers. Type and features of these layers are as following:

1) `Layer1 - LSTM` Layer with 14 nodes. This layer has 952 parameters (all trainable) in total.
2) `Layer2 - Dense` Layer with 8 nodes. This layer has 120 parameters (all trainable) in total. The activation function is the `linear` activation function.
3) `Layer3 - Dense` Layer with a single node. This layer has 8 parameters (all trainable) in total. The activation function is the `relu` activation function.

For learning process, Adam optimizer has been used with parameters of $b_1 = 0.9$, $b_2 = 0.999$ and $\epsilon = 1e - 8$ (default value). As a loss function, mean squared error has been used. The model has been trained for 25 epochs with the batch size of 500. Epoch size is relatively low since many training epochs cause overfitting. This decision has been given using validation dataset.

5 Cloud Architecture

The project uses AWS lambda function as its backend computing device. Lambda function has access to 2 resources:

1) `project.py` which is python script for evaluating the result and sending it back to the front end side. This script is stored in the lambda function itself.
2) `model.h5` which stores the coefficients of the trained LSTM model. This file is also stored in the lambda function itself since it is small in size.

The script utilizes python packages of `tensorflow` and `keras` for rebuilding the trained model and `numpy` for making calculations. Since aws Lambda functions do not have an access to these packages, I added already created layer to my lambda function for utilizing specified packages. Here is the steps of how the project functions:

1) Only predictor utilizes aws lambda function. When there is a request from the predictor, the lambda function recieves the input data as a string. $\rightarrow$
2) The lambda function splits the string to numbers (using parsing). $\rightarrow$
3) Then it rebuilds the model using `model.h5` file. $\rightarrow$
4) Then, generated numbers from step 2 are given to the LSTM model as an input to make a prediction. $\rightarrow$
5) After predtion is made, the result is sent to front-end side. $\rightarrow$
6) The graph in the predictor page is built in front-end side using the result from the aws lambda function.

6 Evaluation

Testing set has been used for evaluating the data. Metrics that were used for evaluation are mean squared error (MSE) and mean absolute error (MAE). The reason for using MSE and MAE is see how close the predictions are to real values. The value for these metrics are 0.46 and 0.44 respectively. According to this result, we can say that, the difference between real and predictions values are not very high.

More detailed analysis show that, high true values (volatility) are mostly associated with high prediction values. This aspect of the model can be useful in that, high prediction for volatility usually matches with high true volatilty.

7 How to use the software?

In order to use the software, user should first go to Predictor page from the navigation bar. Then, user should give the return data of previous days as an input to the input field. The format of the input is the comma seperated return values (in percentage) of consequentive days in ascending order $(R_1, R_2, R_3...)$.

Here is the example input: `1.16, 1.12, -2.11, 0.1234, 3.44` (length is 5)

The software is designed to accept the data from last 14 days when the market was open. So, it is recommended to input the data of the last 14 days. If the length of the data is higher than 14, then last 14 days (last 14 values) will be used. If the length of input data is lower than 14, then initial missing values will be filled with the mean daily return of S&P500 index which is 0.02978%. The visualization will be shown as a line plot of previous volatility and predicted volatility (the last data point in the line plot)

To see an example of how predictor works, `Demo` button in the predictor page can be clicked. This button will run a demo with pre-defined input (`0.64, -0.16, -0.13, -0.2, -0.24, 0.19, -0.11, -0.08, -0.1, 0.73, 0.11, 0.29, -0.23, -0.16`).

8 Conclusion

In conclusion, there are some additional suggestion for further improvement of the model:

1) design the model for distribution forecasting instead of point forecasting. By that way, user can evaluate the credibility of the estimate.
2) Add continuous learning component for the model so that it would update itself.
3) More financial instruments can be added to the website.[1]

9 References

[1] - Online resource: https://stooq.com/db/h/