Capstone Project
Machine Learning Engineer Nanodegree

Michal Jacek Kozinski
January 30th, 2020

Definition

Project Overview

Equity options are one of the most popular derivate products being traded worldwide on financial markets. With the advance of mathematics, computer science and computing power, traders moved from pricing options based on their experience, towards using mathematical models towards using machine learning and statistical models.
In this project I tried to predict future realized volatility of Spanish stocks based on past realized volatility, one of the key factors in option pricing. The model uses a machine learning model using dataset downloaded from Kaggle for 27 Spanish stocks' daily returns from 2000 to 2019.

Problem Statement

The goal of this project is to create a realized volatility predictor based on past realized volatility. The process of creating it involves:

1. Download and import Kaggle dataset
2. Look at missing data and overall quality
3. Clean feature extraction
4. Dividing data into train, validation and test sets
5. Training various models
6. Checking accuracy of the model on validation data
7. Choosing the best model and hyperparameter tuning on validation data
8. Look at the final model accuracy.

Metrics

So the metric that I think is relevant here to look at is RMSE, aka root mean square error (Figure 1):

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

Figure 1

This metric is a good measure of how accurate our model is at it is sensitive to outliers which are very important in trading. Second metric that I plan to look at is the distribution of our results vs the actual numbers, its skewness and standard deviation. Besides that, I will also look at MAD (mean absolute deviation, Fig 2), which is very similar to RMSE, but is not sensitive to outliers, is "equally" sensitive.

$$MAD = \sum_{i=1}^{n} \frac{|y_i - y_i|}{n}$$

Figure 2

Analysis

Data Exploration

The data analyzed was taken from Kaggle[1] and consist of daily stock returns for 27 companies from 2009-02-11 to 2019-06-07, measured in typical fashion for financial returns where each stock has recorded 4 prices for each day, open price, highest price of the day, lowest price of the day and closing price (OHLC). There is also one extra metric, which is volume, which is the total volume of stock that changed ownership on that day. The data is divided into 27 csv files, one for each stock, as in Fig 1, that shows example of first 5 entries for stock Ibedrola, a Spanish multinational electric utility company.

	Date	Close	High	Low	Open	Volume
0	2009-02-11	5.59	5.59	5.52	5.57	0
1	2009-02-12	5.43	5.55	5.43	5.53	0
2	2009-02-13	5.50	5.62	5.50	5.55	0
3	2009-02-16	5.35	5.50	5.35	5.48	0
4	2009-02-17	5.66	5.66	5.34	5.35	0

Figure 3

We can already see that data must have missing values as values reported for volume are 0. However, OHLC we can see that OHLC data seem to be accurate, however we need to look also at all data. I decided to use only closing pricing of the day as it is industry standard as well as traders usually hedge their option portfolio towards the end of the day, therefore the close pricing is the one that mostly reflect

[1] Bartolome, Alvaro. "Spanish Stocks Historical Data from 2000 to 2019." *Kaggle*, www.kaggle.com/alvarob96/spanish-stocks-historical-data.

reality. The most important information concerning the data set is summarized below:

1. The maximum closing data exist for 2604 days for 19 stock, minimum exist 2242 for one stock and average is equal to 2584.3, therefore we have in total 69,777 data points of daily stock prices (what equals to 69,750 stock returns as we need 2 days to calculate one day of stock returns).
2. The minimum price for any stock is not 0 but 0.0106, what actually tells us that we do not have bad data (as stock cannot fall below 0 as well as it is practically impossible to be 0). Furthermore, after quick online search, that stock still has similar price as reported in our data which only assures that we do not have bad data. [2]

Exploratory Visualization

The plot bellow (Figure 2) shows a random selection of 6 stocks out of 27 daily prices. We can see that stock data seem to be accurate, with only one stock (Enags) showing some extreme behavior. However, it should not be discredited as such behavior might happen in case there is a late afternoon, just before market close rumor of company being taken out at multiple premiums to current stock price, therefore as such behavior might be uncommon it should not be discredited.

[2] "Abengoa." *Wikipedia*, Wikimedia Foundation, 27 Dec. 2019, en.wikipedia.org/wiki/Abengoa.

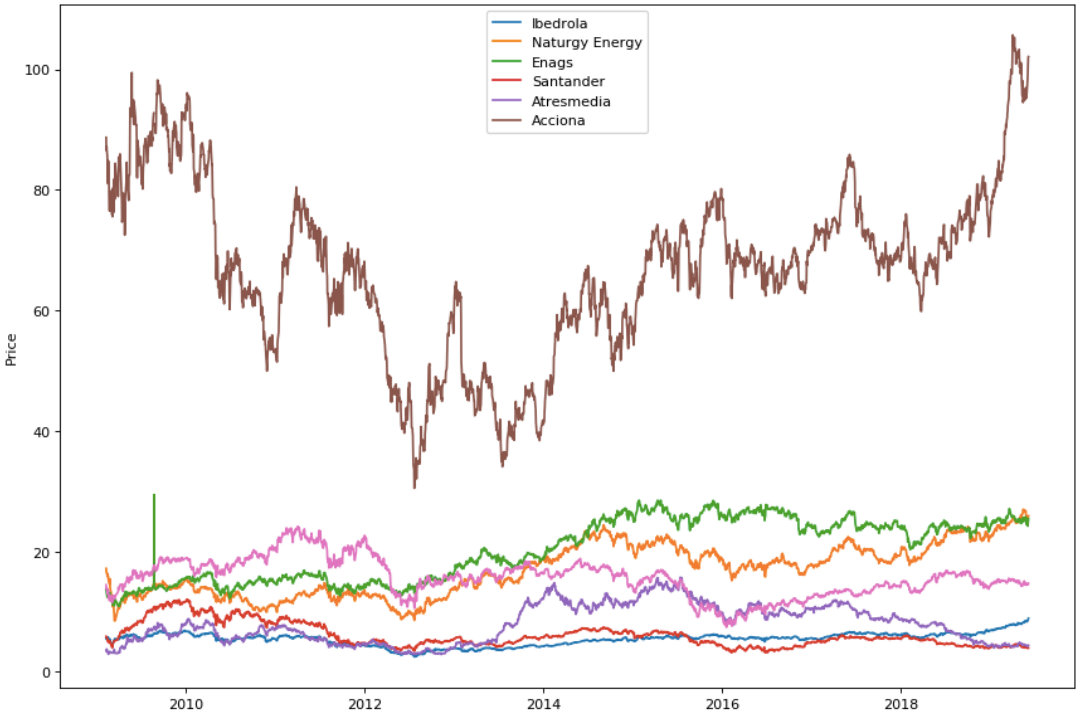

Figure 4

In the plot below (Figure 3) we can see daily stock returns for same stocks as previously. We can see that daily stock changes are around zero, what we should expect, especially as we can see that chosen stocks barely had any positive returns over that period and average return across all stocks and all available data from this data set is 10% over this 10 year period.

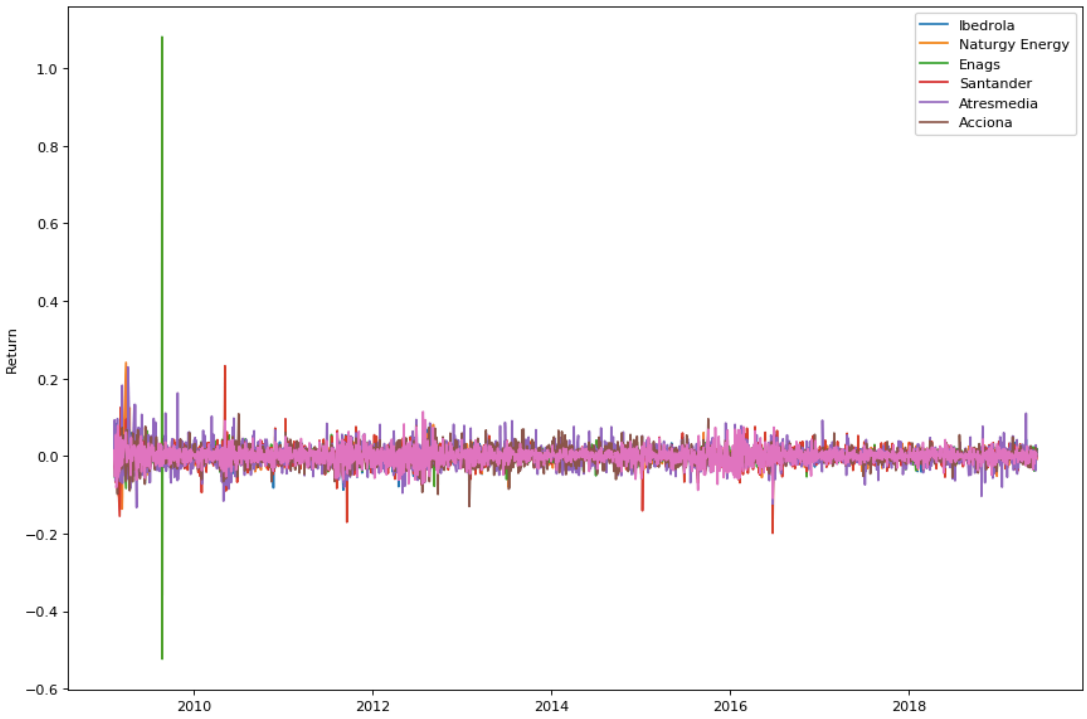

Figure 5

In the plot (Figure 4) below we can see that daily stock returns are centered around 0 and "resembles" normal distribution, as previous plots showed. However upon calculating typical statistical metrics, the data is highly skewed with skew = 200 and excess kurtosis also much bigger than normal distribution has.

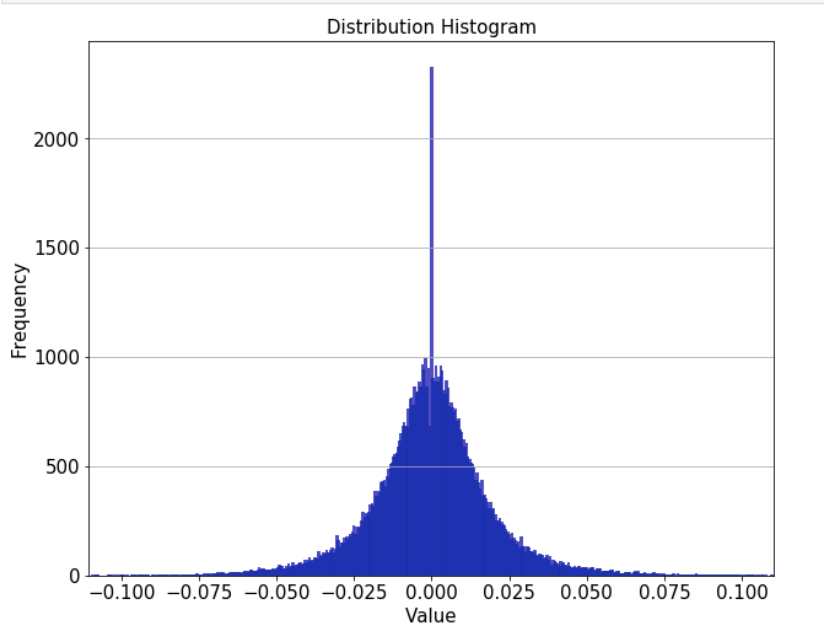

Figure 6

Algorithms and Techniques

I plan on using Amazon XGBoost algorithm as it is the most suitable for this task. I originally planned also using Amazon's DeepAR forecasting algorithm, however I realized it is not appropriate for my task as DeepAR tries to forecast "the same thing" as it was inputted, while here we have granular, daily volatility data combined with the direction of move and we are trying to predict "average" realized vol over a period of days. XGBoost fits here as it is state of art, highly efficient implementation of the gradient boosted trees algorithm[3]. It fits our purpose perfectly as we need supervised learning algorithm as we have input and out which we are trying to predict. XGBoost is especially good as it won multiple machine learning competitions and proves to be a great tool. The version used in this project is 0.90-1, latest available one. The amount of possible hyperparameters is too big to describe here everyone of them, so I will do only ones that I change from default ones.

Benchmark

Usually such models are implemented as proprietary tools by hedge funds and prop shops therefore there is very limited data. Furthermore, in most cases the models use much more data than is available to me here, therefore I will just use "naive" model where I will use "lookback" realized vol as prediction for the look forward period (so if I am using past 20 days data to predict next 5 days realized vol I will just assume next 5 days realized vol is equal to past 20 days realized vol). Benchmark calculations are created in a very similar fashion to preprocessing, thus both functions look very similarly as a result.

Methodology

Data Preprocessing

The preprocessing is divided into a number of functions:

[3] Mishra, Abhishek. "Machine Learning in the AWS Cloud: Add Intellegence to Applications with Amazon SageMaker and Amazon Rekognition." *Amazon*, John Wiley & Sons, Inc., 2019, docs.aws.amazon.com/sagemaker/latest/dg/xgboost.html.

1. Import each .csv file, copy the "Close" column add it to the main dataframe (df) with the name of file as the column name.
2. Replace column names for just stock names (delete .csv at the end).
3. Create log-diff-power dataframe by first applying log function to each entry, then difference between rows, and then squaring each value. This dataframe saves a lot of computing time in next steps.
4. Create direction dataframe, dataframe that only has 3 possible values for each day {0,-1,1}, -1 if the return was negative, 1 if it was positive and 0 if stock closed at the same price as yesterday's close, which also will be used in next steps.
5. Create a dataframe that has in its first column the thing we are trying to predict, which is realized vol over period of days, and in next columns has features, which are daily realized vol multiplied by the corresponding time vector taken from direction dataframe.
6. Drop all rows with at least one nan (therefore where data is missing or it was not possible to calculate as you need at least 2 days of observations to get daily return).
7. Combine all possible combinations into one dataframe.

The following formula (Figure 5[4]) shows the calculation for realized vol adjusted to be in yearly vol numbers where n in the number of days observations are taken and P_t is the daily price (I changed the order as I first took logarithms of each number and then subtracted using laws of logarithms for efficiency).

$$Vol = 100 * \sqrt{\frac{252}{n} \sum_{t=1}^{n} R_t^2}$$

$$R_t = Ln \frac{P_t}{P_{t-1}}$$

Figure 7

Now, I feed all the dataframe, regardless of stock into the function and it returns one dataframe with all possible model inputs. In next step I shuffle order of all features-ys pairs (but not order inside of them as that would not make any sense) and I divide all data into 3 sets, 70% for training, 21% for testing and 9 % for validation. Next step is just simple uploading to S3 so algorithm can actually use it.

Implementation

[4] "RealVol Daily Formula (Realized Volatility Formulas)." *Calculating Realized Volatility*, www.realvol.com/VolFormula.htm.

First problem I encountered is while testing if the XGBoost algorithm runs on a much smaller dataset, I realized that even running it on 70% of 3 stocks it takes a bit of time, therefore I decided to do hyperparameter tuning on 3 random stocks and then train the whole model according to these hyperparameters. I chose to predict next 5 days (trading week) avg realized vol to be based on last 20 days realized vol on my own personal experience.

After first run of XGBoost algorithm, with XGBoost set for:

```
max_depth=5,
 eta=0.15,
gamma=3,
min_child_weight=5,
subsample=0.8,
objective='reg:linear',
early_stopping_rounds=10,
num_round=250
```

On the validation set I obtained values of RMSE = 71.7 and MAD = 22.3. These values are definitely quite high, as the average volatility is actually equal to 28 from this data set (which equals to about average 1.5% per day movement of stocks). However, the much higher value of RMSE should not be suprising as stocks tend to have returns with much higher excess kurtosis than if they came from Gaussian distribution as I showed previously while calculating excess kurtosis (aka have much fatter tails). As we can see on the graph below (Fig 8), our model tend to overestimate predicted vol. It should not be especially surprising as usually big vol days come because of some unexpected event, or event that could not be predicted by the algorithm just from past data, such as a day when company issues earnings or there is premiere of their new product (as Tesla introducing new car model).

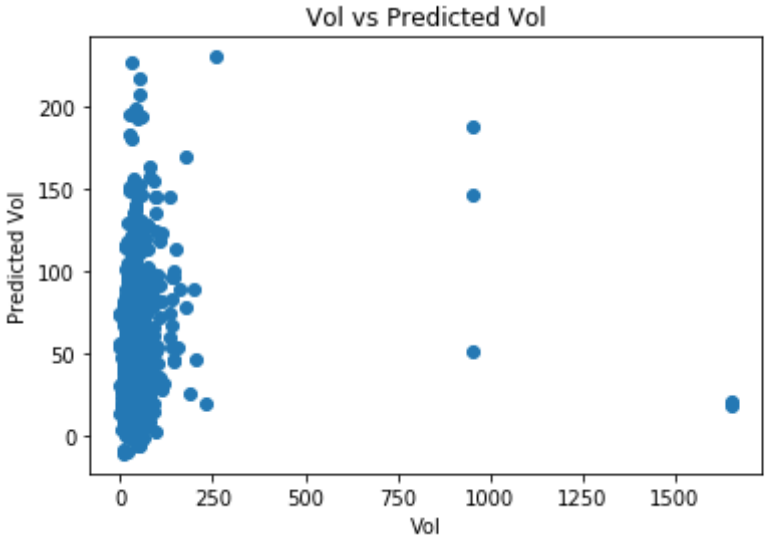

Figure 8

Refinement

In order to improve my model, I decided to run a hyperparameter tuning job, with following variables to test:

```
objective_metric_name = 'validation:rmse',
objective_type = 'Minimize',
max_jobs = 40,
max_parallel_jobs = 4,
hyperparameter_ranges = {
        'max_depth': IntegerParameter(3, 12),
        'eta'     : ContinuousParameter(0.05, 0.5),
        'min_child_weight': IntegerParameter(2, 8),
        'subsample': ContinuousParameter(0.5, 0.9),
        'gamma': ContinuousParameter(0, 10),}
```

After quite a long time, the best model from the hyperparameter tuning job obtained samll improvement, with RMSE = 69.8 and MAD = 17.2. As we can see the MAD increased much more than RMSE, which should not be of a surprise as I wrote previously, the very big vol days are very hard to predict without extra information, however “regular” vol days should be easier to predict. Next, with hyperparameters set to the same that gave the best tuning job results, I trained the model on the whole dataset of 27 stocks. The hyperparameters were:

```
(max_depth=3,
 eta=0.06166,
 gamma=5.0833,
 min_child_weight=3,
 subsample=0.8855,
 objective='reg:linear',
 early_stopping_rounds=10,
 num_round=550)
```

After the model trained (and what is important it stopped just after 100 epochs because there were no improvements), the RMSE = 38.9 and MAD 12.5.

Results

Model Evaluation and Validation

The final model parameters chosen by hyperparameter tuner job indicate that model is prone to overfitting and thus the model is not deep (max_depth = 3) however all other values seem to be within "normal" range to what is usually used. The model seem to be robust as it is especially hard to predict vol based and looking only at daily returns we should not expect huge precision and accuracy.

Justification

The benchmark model, which "predicts" realized vol of next period to be equal to realized vol a look-back period have RMSE = 46 and MAD=14 when using all data. First thing that is important here to note is that it is MUCH more accurate than the model trained only on 3 stocks. One of the reasons might be because the 3 stocks that were chosen were not a good representation of general patterns across stocks. Furthermore, the model might just needed more data as it over fitted for the specific training data it was provided and much more data greatly improved the results. On the graph below we can see how the new model predicts vol.

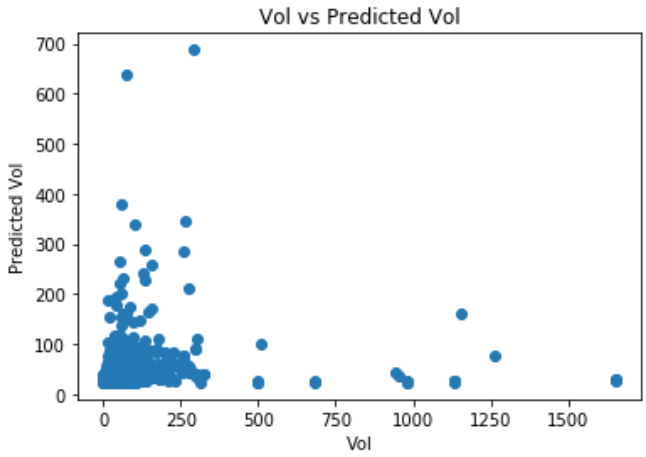

Figure 9

I tried also training a fully connected PyTorch Neural Network however it did not want to work as it gave nan as the error function result. One area of improvement would be making PyTorch NN to work, however I doubt it would have much bigger accuracy.

The model is definitely not enough for traders to be able to accurately predict vol for options trading, however it shows how with very little data we are able of get within about 13vol points accuracy assuming we consider MAD. In order to improve the

model we would have to use MUCH more data and consider features besides just past period returns. Of course my arbitrary choice to use past 20 days to predict next 5 days realized vol is completely arbitrary, however from my experience the more you look in backwards the less relevant it is. Of course I could also tune these hyperparameters but I doubt it would have huge effect on the performance as a "general" rule of thumb is to predict next x days of vol on last x day of vol. Of course ML model should be able to discount past vol if it was true, therefore in theory we could use past 100 days to predict next 5 day. Furthermore, our model does not account for (as it does not have data) for special volatility days such as when company releases earnings or some other important market-wide thing happens (such as FED changing interest rates). The model did the most it could, and using more features, if they were available would definitely improve accuracy but based on given data it is the best that could be obtained.