Application of Deep Learning Methods for Stock Market Prediction

Vlad Savchenko
December 31, 2017

1. Background of the problem, motivation, project outline.

The stock market is a vital part of the US economy, attracting investors from around the world to participate in price discovery and profit from this activity. Therefore, stock market prediction has become important occupation of thousands of professionals, from hedge fund, pension fund and wealth managers to trading desks and individual investors.

In this project, we will apply deep learning methods, such as Long-Short Term Memory Recurrent Neural Networks (LSTM RNN), for stock market prediction and compare the results to the classical methods used for time series analysis, such as ARIMA and GARCH models.

Also, there is another group of deep learning methods, recurrent reinforcement learning (RRL), applied successfully to stock market trading, bypassing prediction stage altogether. This provides further motivation to this study: can application of deep learning for prediction help in better understanding and improving of these results?

One of the most famous theories about the market is Efficient Market Hypothesis (EMT). It states that, it is impossible to “beat” the market, because market price already incorporates all relevant information. There is a large volume of academic literature defending this theory, while a small army of successful hedge funds, traders and investors prove it wrong with their bank accounts in practice. However, perhaps, one of the most striking examples contradicting the theory is the buyout phenomena: buyout prices almost always much higher than pre-buyout share prices, quite often 50% to 100% higher! Why would the buying firms be willing to pay 100% more than the market price, if EMT was true? Obviously, they do not believe EMT, neither do I.

2. Potential clients.

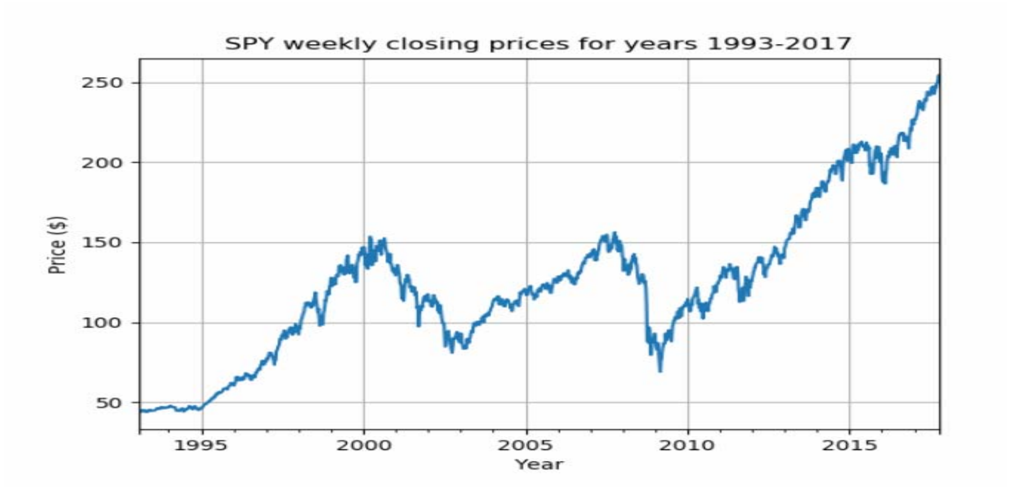
These methods can be applied to stock market indices, futures, individual stocks. Potential clients range from hedge fund, portfolio and wealth managers to trading desks and individual investors.

3. Data wrangling and initial exploration.

We are going to perform exploratory data analysis on the financial time series formed by SPY weekly closing prices. The goal of this analysis is to see if there are any identifiable patterns or other regularities in the time series data itself. SPY is an ETF (Exchange Traded Fund) which can be thought of as a proxy for S&P500 index. We focus only on the weekly prices excluding any other data which might be useful for price prediction.

Once a pattern is found, it can be modeled in some way, leading to better understanding as well as identification and analysis of money making and hedging opportunities (e.g. for portfolio management).

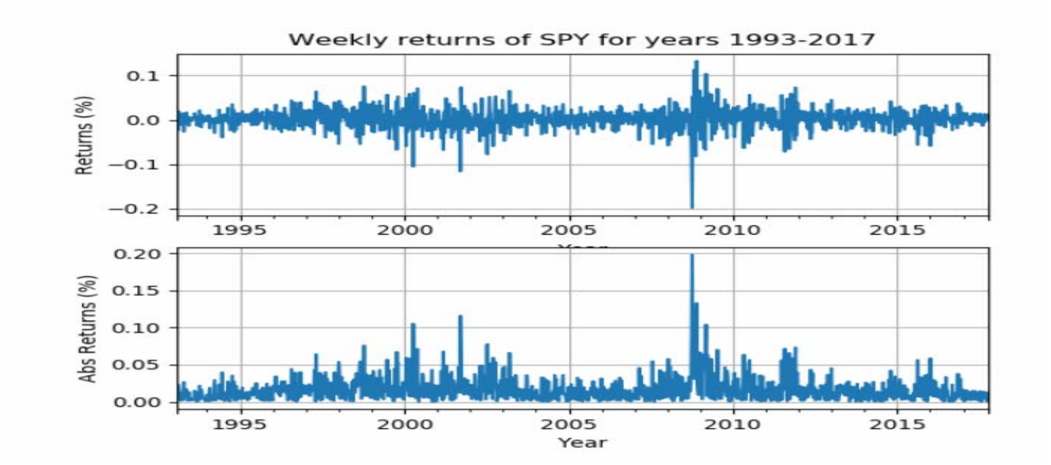
Let us plot a regular price vs time figure to understand the overall behavior and check for any anomalies.



From the figure above, one may notice that the time series is non-stationary. It exhibits trending behavior. One of the two requirements of weak stationarity is that the mean be independent of time. The plot above clearly shows that this is not the case. The second requirement says that covariance of returns

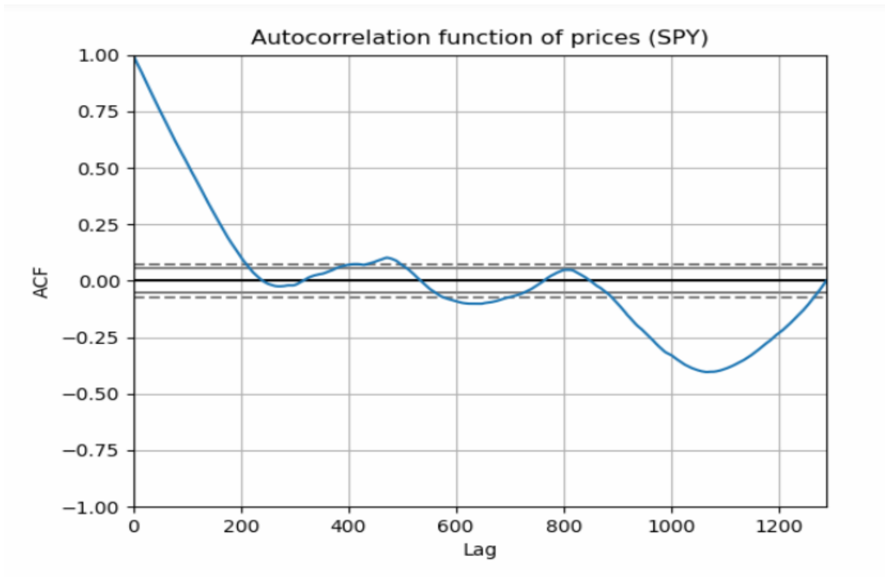
$$Cov(r_t, r_{t-l}) = \gamma_l$$

should depend only on the time difference in lag, *dt=l*, and not on the time itself. Let us examine this more closely by calculating and plotting a series of returns *r(t)*.

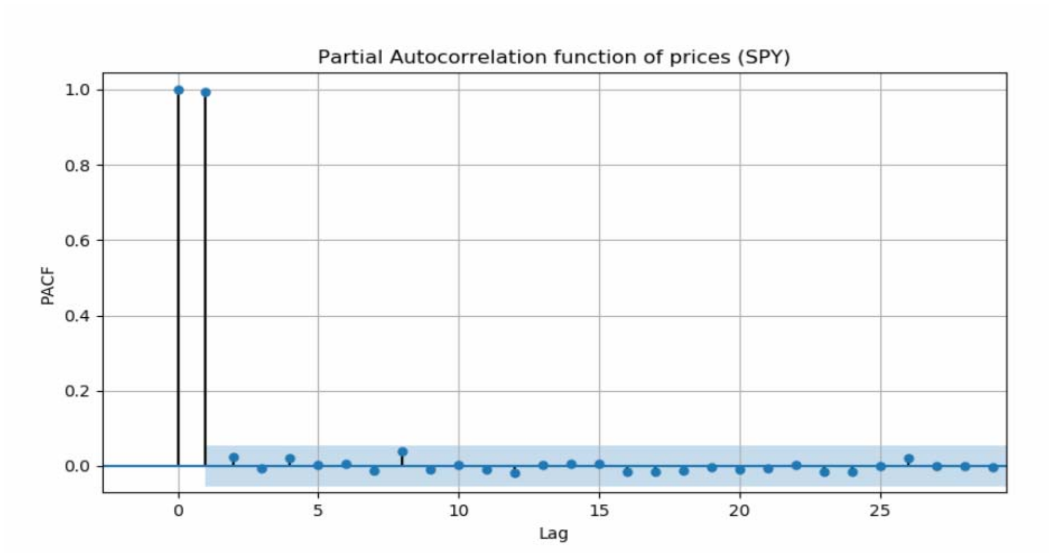


Let us examine the plots above. We see that the returns have a mean reverting behavior, since they oscillate around 0 mean. However, the amplitude of the oscillations is not constant, something we would expect for a stationary series. Instead, the amplitude exhibits clustering. Large amplitude spikes are followed by similarly large spikes, and short spikes follow short spikes. This can be better seen in the plot of absolute returns (second panel). We will come back to this important point later.

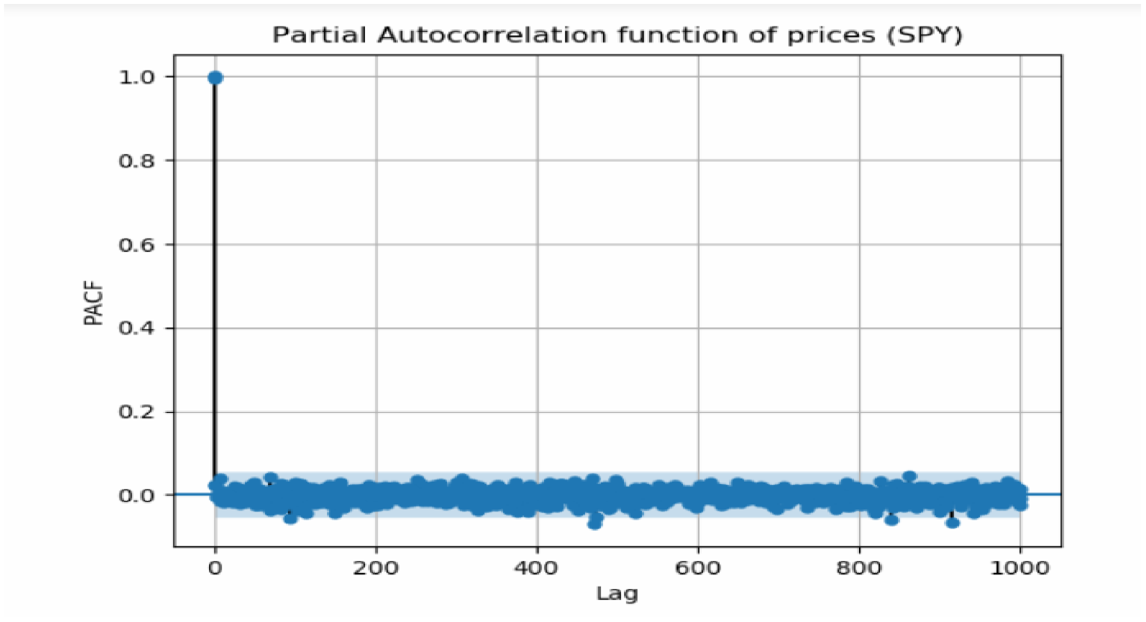
Now, let us turn our attention to the autocorrelation function (ACF). First, look at the ACF of prices.



One can see that, that ACF of prices exhibits non-zero auto-correlation for lags up to 200, therefore hinting at linear time dependence in the time series. However, if there is non-zero auto-correlation at lag 1, it will propagate through higher time lags, contaminating the specific influence of the other lags. So, we have to examine a partial auto-correlation function, PACF, which eliminates such influence and shows the effect of each specific lag. The plot of PACF is shown below, where the blue band around 0 indicates a 95% confidence interval.

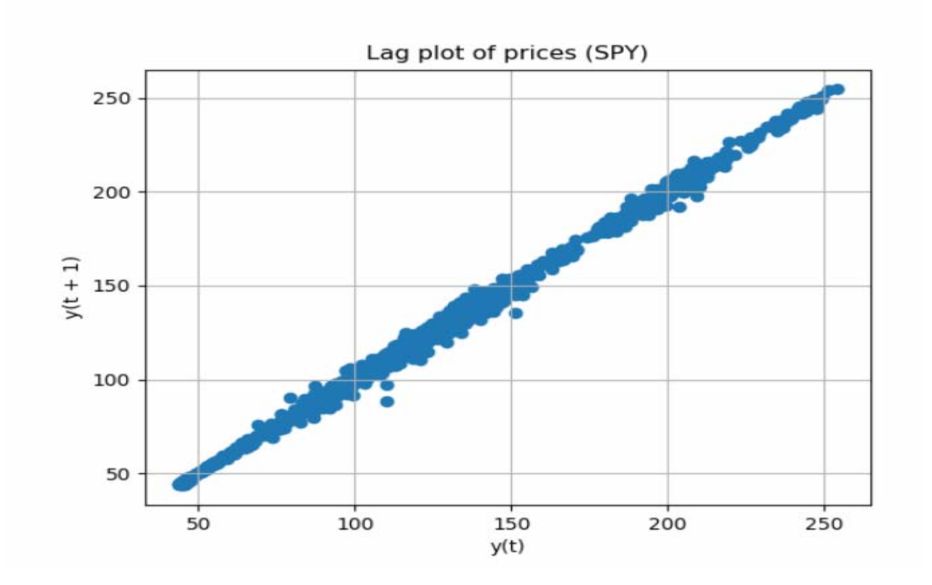


We see that the PACF shows only two non-zero auto-correlation coefficients, at lags 0 and 1, meaning that there is no linear time dependence in the series of returns (after differencing the price series). However, let us look at the coefficients at all lags more carefully.



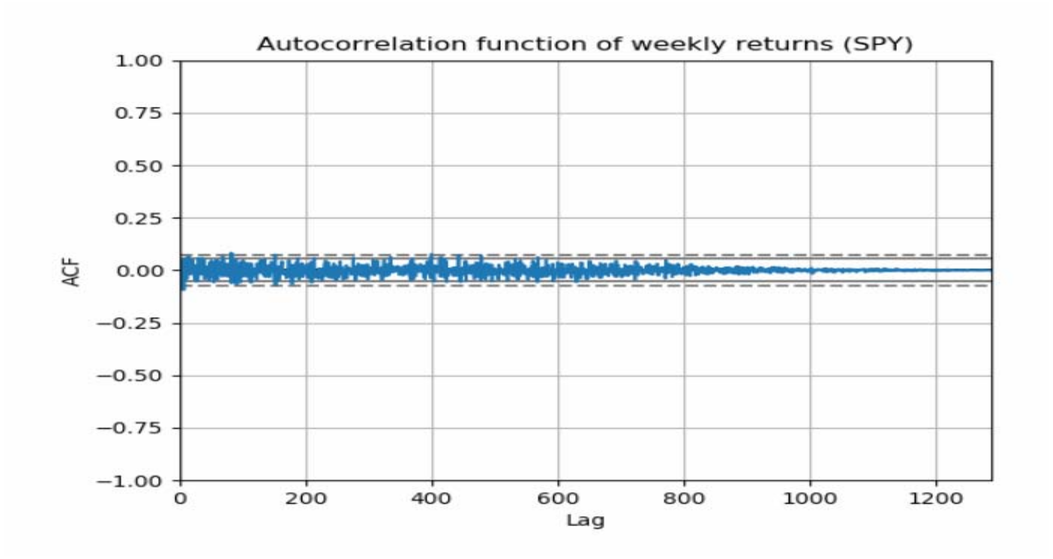
Interestingly, there seems to be non-zero (negative) coefficients at lags around 90, 470, 840, 920. They may indicate attraction/repulsion effects at certain price levels, perhaps, showing that proximity of current price to past price pivots (inflection points) is important. We will return to this point later.

For completeness, we also produce a lag plot of prices at lag 1 below.

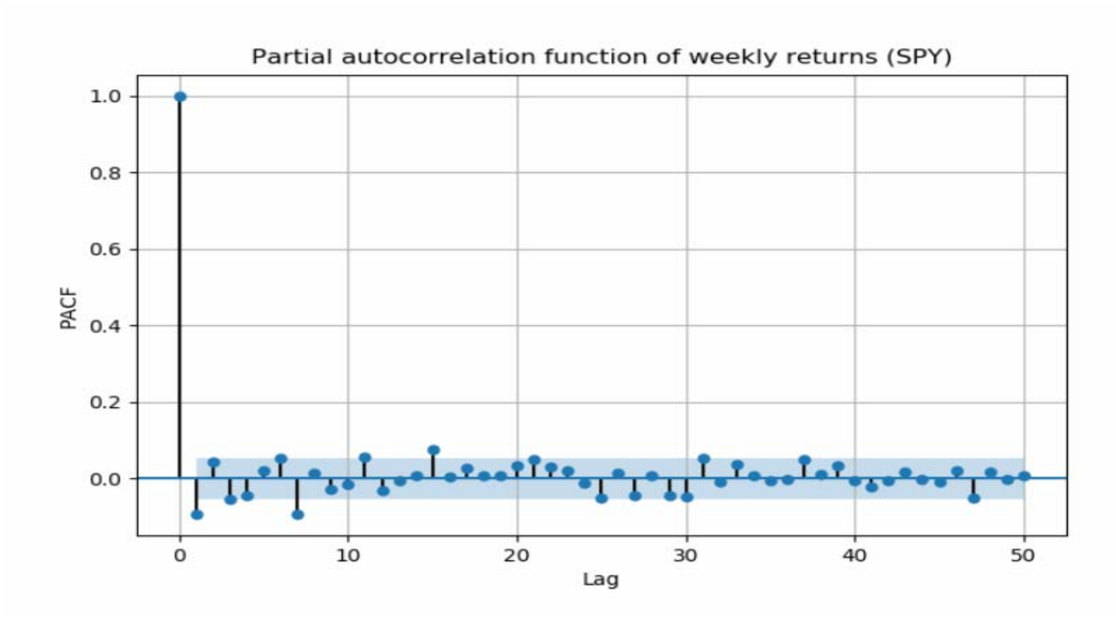


One interesting observation from this lag 1 plot is that the variance does not seem to be proportional to the mean (as some assertions in the financial literature would make us believe). We observe that the vertical spread does grow when *y*(*t*) increases from 50 to 150. However, it then oscillates when *y*(*t*) goes from 150 to 250: it shrinks from 160 to 180, then widens till 220, narrows to 240, then widens again. This observation is consistent with the current volatility regime in the S&P500 futures market. Due to prevalence of participants in the short volatility trade, the volatility index VIX has never been lower, indicating historically low volatility while prices are making new highs.

Since the final goal is to identify trading opportunities, we have to shift our attention to time series of returns. The ACF of returns is plotted below.

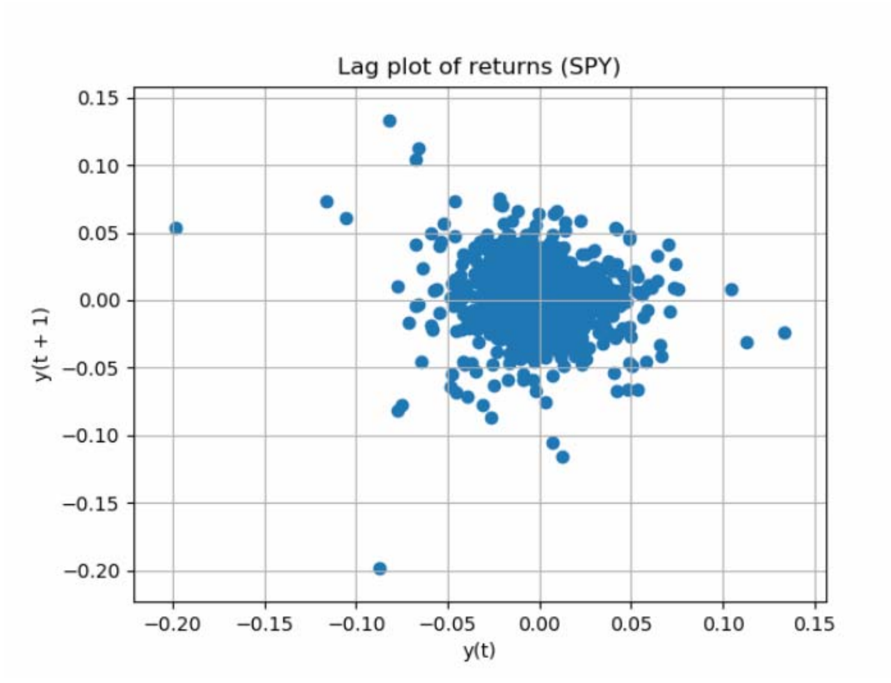


All of the auto correlation coefficients with lag *l*>0l>0 are zero with 95% confidence, implying lack of linear time dependence.

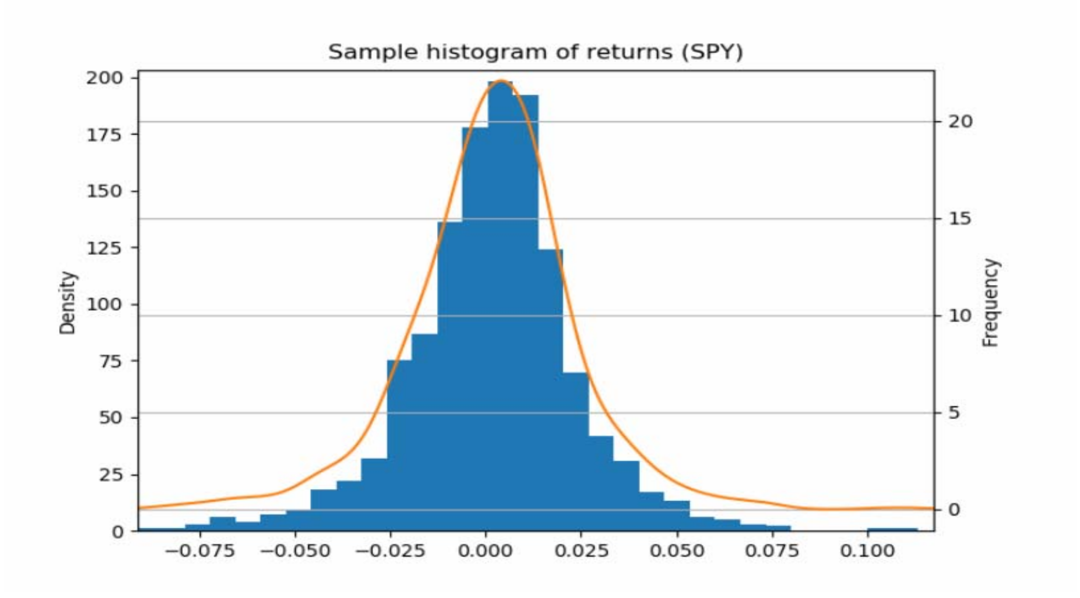


We can draw a similar conclusion by looking at the PACF function of returns above.

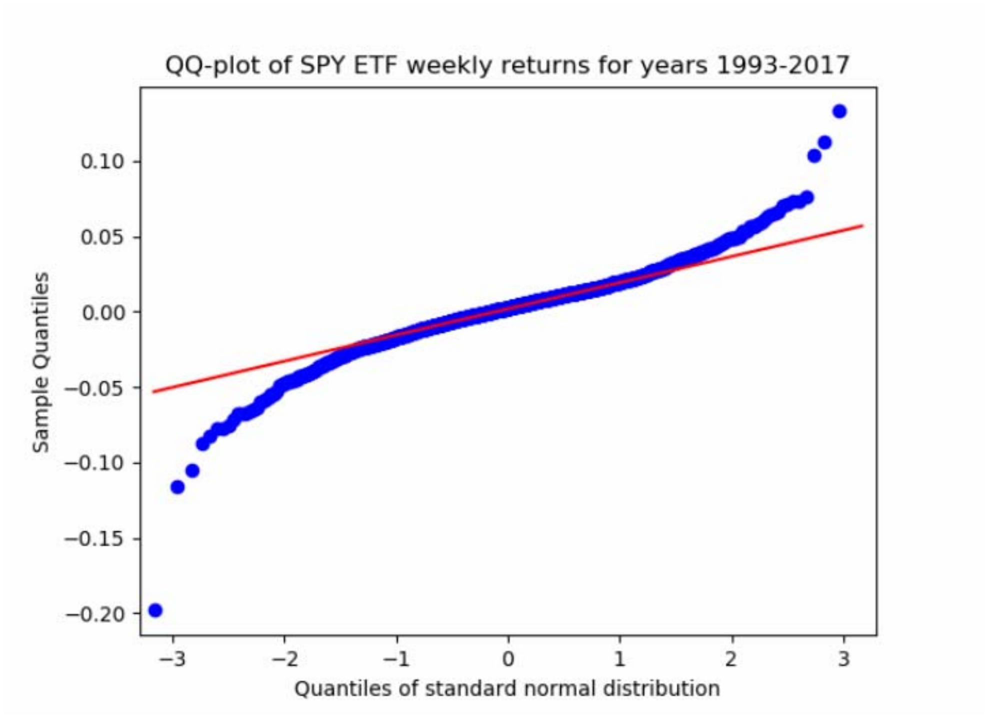
Now, let us check the lag plot of returns.



There is certainly some kind of non-linear structure in the lag 1 plot above. If the series were purely random, it would fill a circular blob. Instead, we see that the blob is elongated and has multiple outliers. So, we can examine the distribution of returns. First, let us look at the histogram.

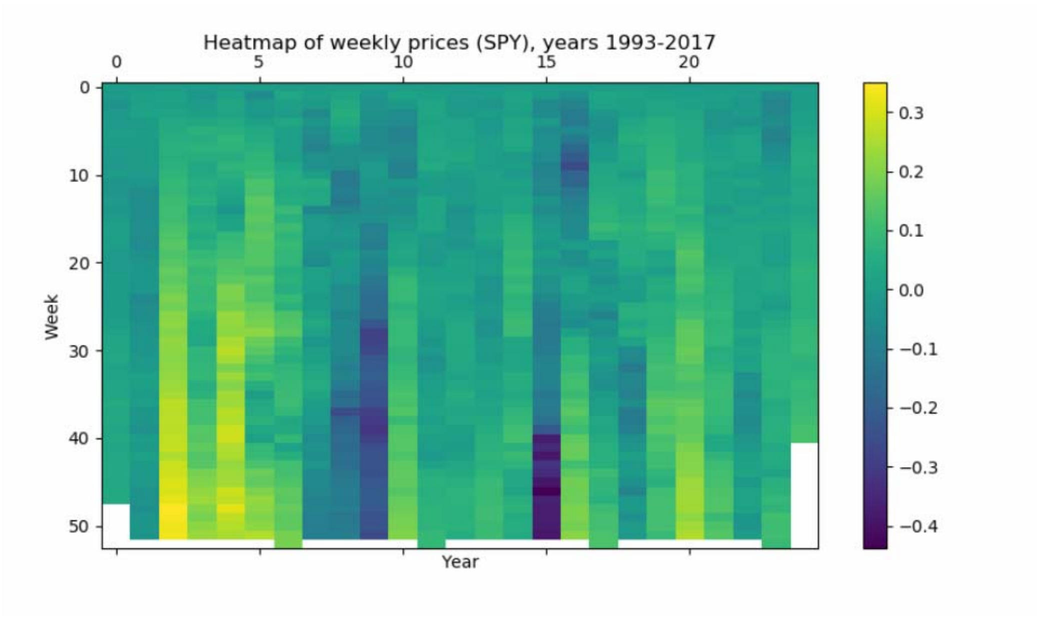


While it is clear that the histogram is not symmetric and has outliers sufficiently far in the tail, we will get a better understanding of the distribution function by looking at the Q-Q plot.



Blue dots are quantiles of our sample distribution; red line follows standard normal quantiles. We can see that there are substantial deviations in the tails, indicating that the distribution of returns is not normal and has fat tails. Fat tail events correspond to large returns in comparison to what a normal distribution would predict. This may imply some sort of nonlinear dependence in the time series.

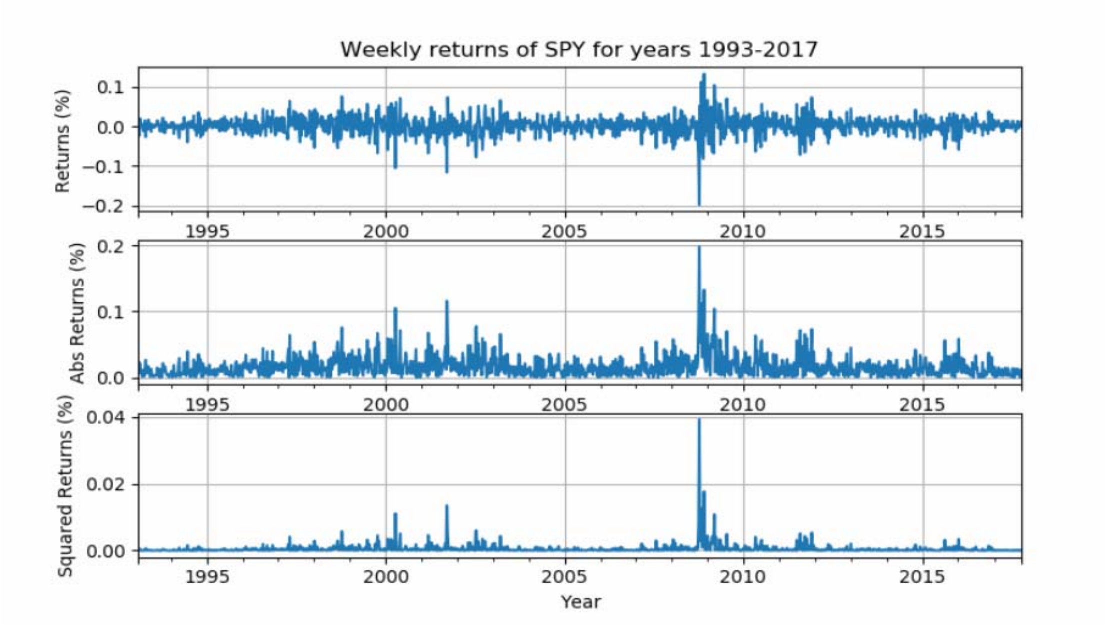
Some price series exhibit seasonality, a periodic oscillations repeating every year. Let us inspect our time series visually by constructing a heat map of yearly sub-series.



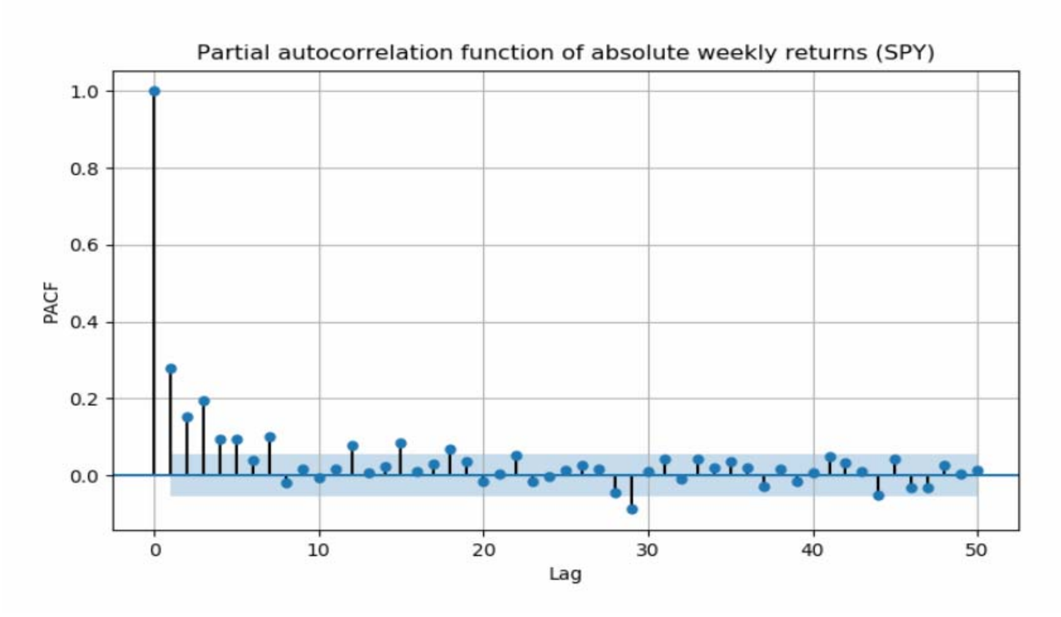
The heatmap Y-axis is weeks, X-axis - years. Every year starts at 0 corresponding to light green color. By examining the heatmap, it becomes apparent that there are no clear seasonal variations. However, we detect a few other interesting patterns. First, "yellow" year (outsized bull market) is not followed by a "deep blue" year (strong bear market). On the other hand, "deep blue" year is always followed by somewhat yellow year, indicating a recovery.

We can also analyze the January effect, which says "As S&P500 goes in January, so goes the year". We see that, there is some validity to it, since blue tinge in January leads to bluer year ends, while green tinge is followed by either lighter green or outright yellow.

Let us go back to analyzing clustering effects in the returns series.



We plotted absolute value of returns as well as squared values of returns to emphasize the clustering effect visually. Indeed, as the third panel shows, volatility clustering is very pronounced. We can plot the PACF of absolute returns to see the time dependence.



The above PACF of absolute returns clearly shows time dependence at multiple lags, strongly suggesting that such nonlinearity is important for prediction of future returns. This further implies that we are dealing with a financial time series which is not completely random and thus can be predicted to some extent, albeit by understanding its nonlinear nature. It is subject for future research.

Let us come back to potential long range correlations in the price series shown in the PACF plot above. Auto-correlation function of a time series is based upon notion of proximity in time. By construction, it silently implies that the most recent history is more important than the very distant past. However, it does not place correct emphasis on proximity in price, or rather, to pivot points at certain price levels. Such pivot points may reflect certain historical events, thus forming important milestones in collective perception of the fair value. Therefore, I propose to study a two dimensional price-pivot ACF (call it PR-ACF) based on price and time proximity. It may describe financial time series nonlinearities better than a standard ACF by incorporating both time and price lag coefficients together.

Therefore, long range correlations may be described by just a few price lag coefficients in PR-ACF (as opposed to many time lag coefficients of the standard ACF), thus leading to dimensionality reduction. It is important to note that, such correlations should not be thought of as waves with long wavelength, because they do not have to imply true time periodicity. Rather, they would show dependency on the price proximity, whenever such proximity occurs.

We may think of fat tails as being formed by multiple extreme events, each event being produced by unlikely coherency (accumulation) between sub-events. Under normality assumption, all sub-events are independent of each other; therefore, having a long string of sub-events of the same sign is very unlikely. However, we have to keep in mind that, financial time series is just a reflection of human, business affairs, and human collective psychology. Humans tend to have inertia in their way of acting, thinking and reacting to the outside world. Therefore, it seems likely that such inertia would lead to market inertia which would tend to produce long strings of sub-events of the same sign quite regularly. So, it is natural to expect fat tails in the distribution of financial time series. Mathematically, fat tail distributions have a power law dependency as opposed to exponential tails of normal distribution.

To understand better, what kinds of sub-events are important and are likely to produce a multiple sigma fat tail event, we have to examine financial history of bear markets and market crashes. For example, in the years 2003-2006 Federal Reserve started to raise interest rates aggressively, removing liquidity from the markets already addicted to liquidity. We may interpret this as one such aforementioned series of sub-events (clearly, not independent, all related to each other), culminating in the financial crash of 2008 (which is but one example of fat tail events).

By the same token, even the market reaction to the financial and business conditions in 2007-2008 may be described as a series of sub-events, each sub-event being the price movement over a certain time period. The early onset of the crisis was not apparent to most market participants, thus not resulting in the immediate crash at the very top. It took time for this realization to spread, bringing a series of news reflecting the situation over a period of several weeks and months, preceding Lehman Brothers collapse. Given the severity of the situation at that time, these news could not have been a mixture of equally good and bad news. They were all bad news, i.e. sub-events of the same sign, resulting in a significant (multiple sigma) price drop. At the same time, there was no regulatory response, since there was a substantial risk of error for regulators to act too early too strongly.

The initial price drop and subsequent Lehman collapse lead to a strong regulatory response, but it was too little, too late. Parade of bad news continued (partly reflecting the damage of the price collapses across the board) as well as price slides, resulting in a snow ball effect.

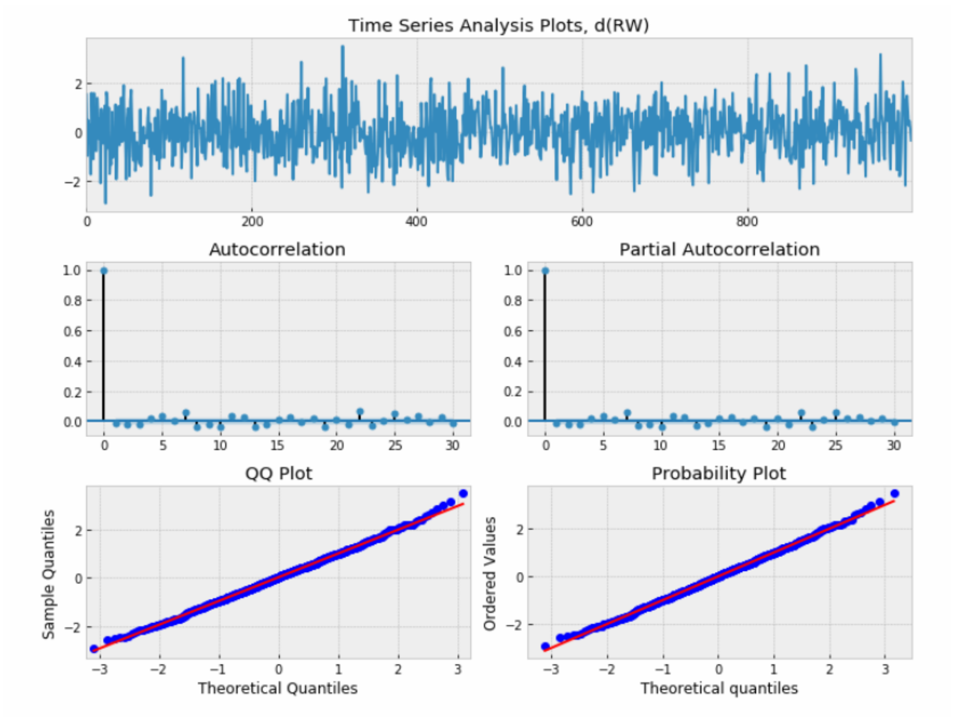
From this discussion, we may conclude that some type of non-linear dependency is the cause of fat tail events. Therefore, understanding and modeling such nonlinearities could be the foundation of successful prediction of multiple sigma events, which is an integral part of financial time series prediction in general.

In summary, we plotted and visually examined time series of weekly closing prices of S&P500 index, its returns, ACFs and PACFs. We discovered potential seasonal effects, volatility clustering, non-linear time dependency and fat tail characteristics of its pdf. We plan to model these effects with the help of latest advances in artificial intelligence.

4. Application of ARIMA and GARCH models

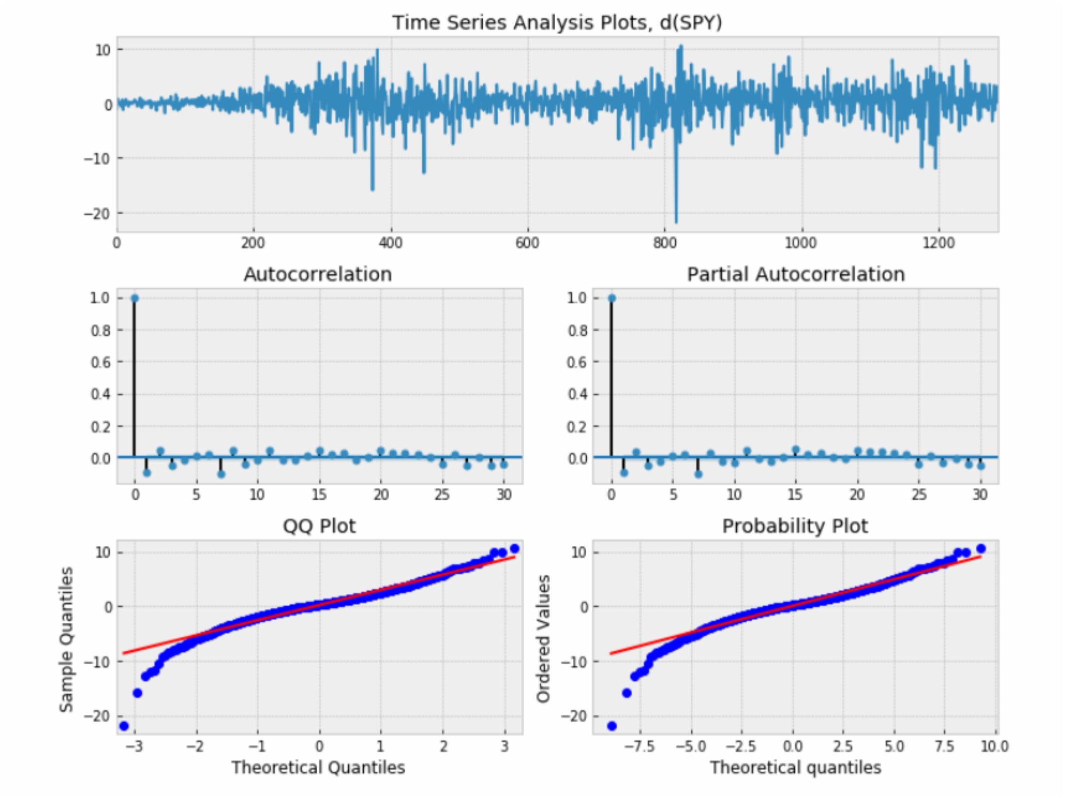
ARIMA and GARCH models have been applied to time series analysis successfully in the past. However, they have certain limitations, when it comes to financial time series. Let us examine them in details.

Let us look at the ACF/PACF and QQ-plots of a random walk process (RW) and contrast it with the SPY series. Here is the plots for RW process after taking a difference once.



From PACF we see that, autocorrelation coefficients are 0 at all non-zero time lags, as expected for white noise. QQ-plot is pretty much a straight line, indicating standard normal distribution.

Let us look at the similar plots for the SPY series.



First, we see that the time plot shows volatility clustering. We can think of volatility as simply the size of the up and down swings on the chart. Volatility clustering means that, the size of the swings continues growing beyond certain level. Second, there are non-zero autocorrelation coefficients just around -0.1 level. Third, QQ-plot deviates from straight line, hinting at existence of fat tails in the distribution. Let us investigate these effects with ARIMA models.

4.1 Autoregressive models - AR(p)

Here is the formula for an AR process:

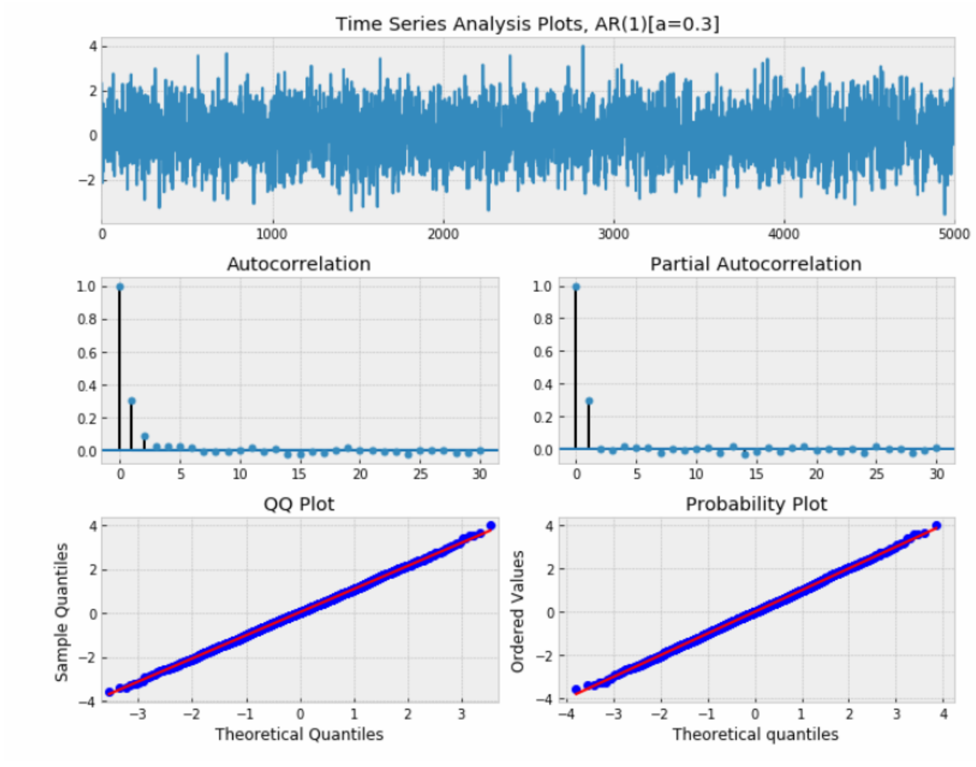
$$x_t = \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + \ldots + \alpha_p x_{t-p} + \omega_t = \\ = \sum_{i=1}^{p} \alpha_i x_{t-i} + \omega_t \tag{1}$$

Here α_i are the model coefficients and ω_t is a white noise term. The simplest models AR(1), AR(2) are as follows:

$$x_t = \alpha_1 x_{t-1} + \omega_t \tag{2}$$

$$x_t = \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + \omega_t \tag{3}$$

Let us generate AR(1) sample with $\alpha_1 = 0.3$ and then fit AR(p) model, recovering α_1 coefficient and the order of best model p.

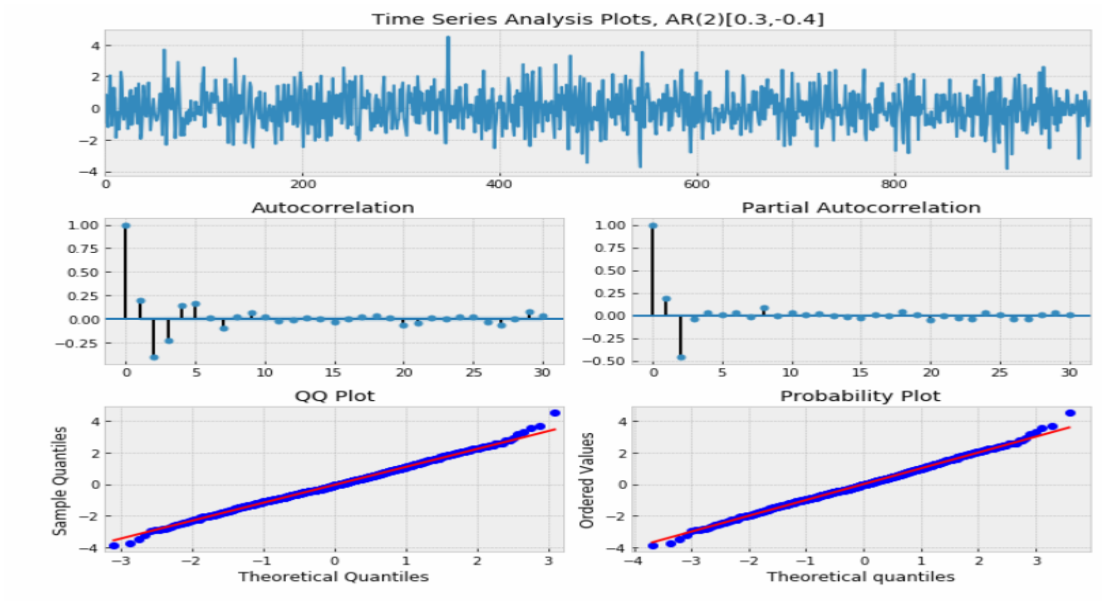


There is a significant correlation at lag=1, as seen in the PACF plot. This is, of course, what we would expect from an AR(1) process. Now, let us fit this data with AR(p) process from Python statsmodels. The output of the model is:

```
alpha-est: 0.30068, p-est = 1

alpha-true = 0.3, p-true = 1
```

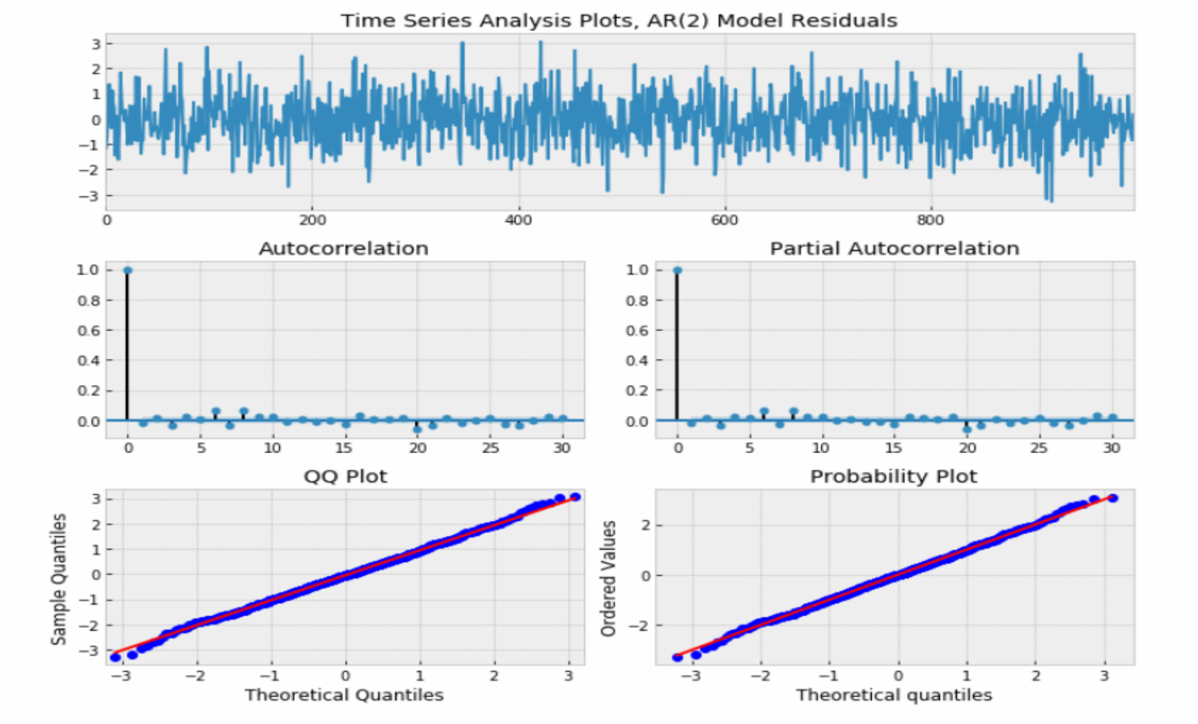
We see that, AR(1) parameters *α*1α1 and *p*p are found correctly! Let us continue with AR(2) in a more general fashion (to be useful for ARIMA modeling later).



Let us see if we can recover AR(2) process parameters with Python statsmodels. The output is as follows:

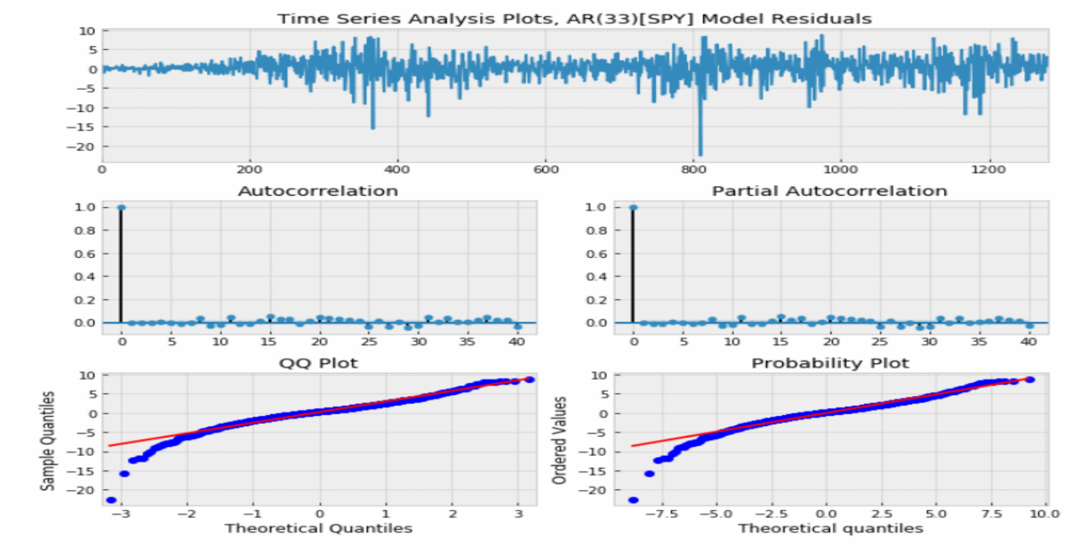
```
coef estimate: 0.28 -0.46 | best lag order = 8

true coefs = [0.3, -0.4] | true order = 2
```



From the plot of residuals above (and its ACF and PACF), we can see that they look like Gaussian noise. Pearson normality test with p-value being 0.27 confirms that, the data most likely came from the normal distribution.

We can try to fit SPY returns by the AR(p) model. Let us see what happens:



We see that, trying to fit SPY returns with AR(p) requires 33 lags. The model seems to be too complex, plus, it does not solve the volatility clustering problem.

4.2 Moving Average Models - MA(q)

Here is the formula for a Moving Average process:

$$\begin{aligned} x_t &= \beta_1 \omega_{t-1} + \beta_2 \omega_{t-2} + \ldots + \beta_p \omega_{t-p} + \omega_t = \\ &= \sum_{i=1}^{p} \beta_i \omega_{t-i} + \omega_t \end{aligned} \tag{4}$$

ARMA(p,q) model is just a merge between AR(p) and MA(q) models. The usefulness of ARMA modeling for finance lies in the following properties of AR and MA models:

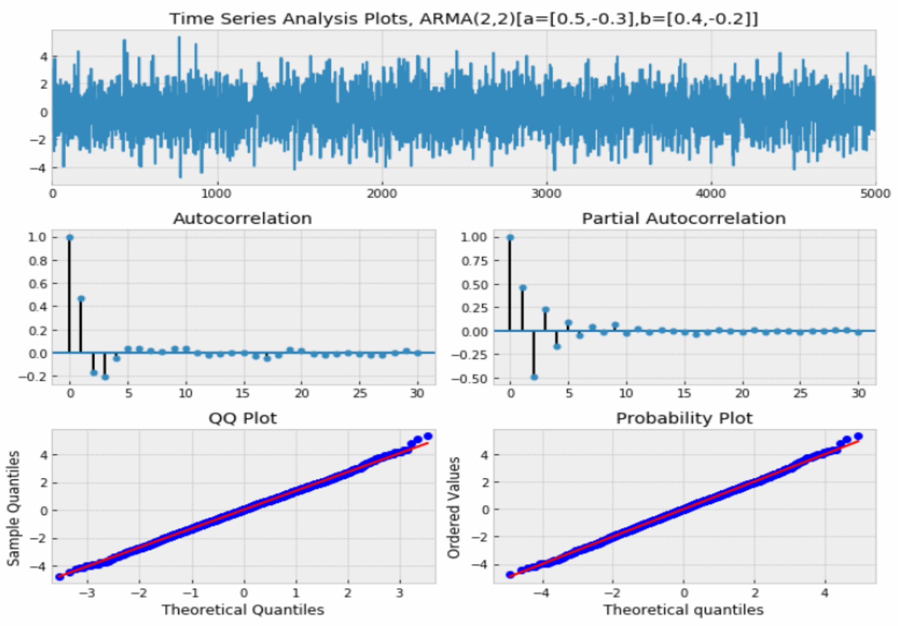
- AR(p) models explain the momentum and mean reversion effects often observed in trading markets
- MA(q) models explain the shock effects observed in the white noise terms. These shock effects could be thought of as unexpected events affecting the time series: e.g. earning surprises, buyout/mergers, interest rate hikes, etc.

However, the problem with ARIMA models is that, they do not capture volatility clustering effects.

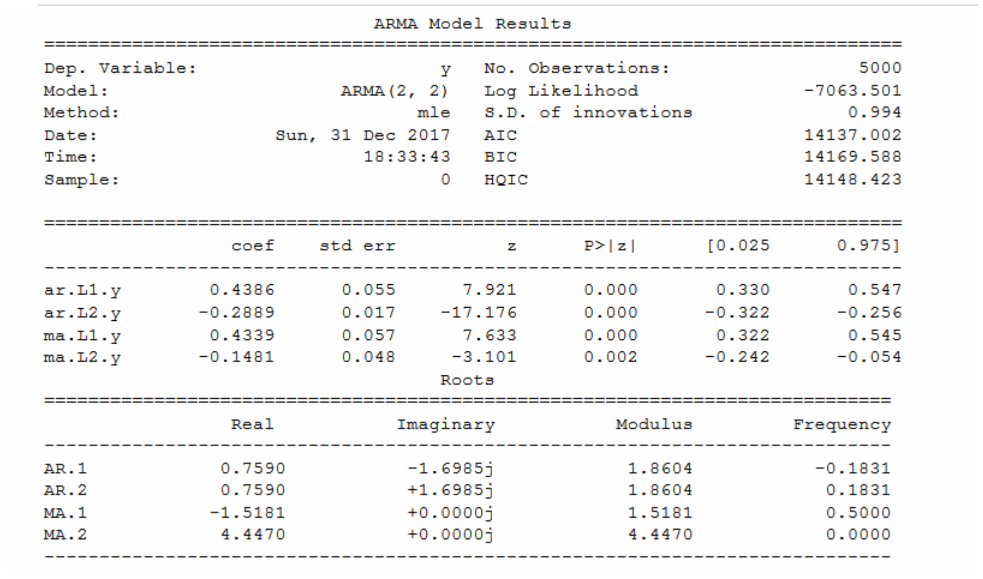
4.3 ARMA Models - ARMA(p,q)

$$\begin{aligned} x_t &= \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + \ldots + \alpha_p x_{t-p} + \omega_t + \beta_1 \omega_{t-1} + \beta_2 \omega_{t-2} + \ldots + \beta_q \omega_{t-q} = \\ &= \sum_{i=1}^{p} \alpha_i x_{t-i} + \omega_t + \sum_{i=1}^{q} \beta_i \omega_{t-i} \end{aligned} \tag{5}$$

Let us simulate an ARMA(2,2) process with parameters alphas=[0.5,-0.3], betas=[0.4,-0.2], and then fit an ARMA(2, 2) model and see if it can correctly estimate those parameters.



Here is the output from ARMA model:

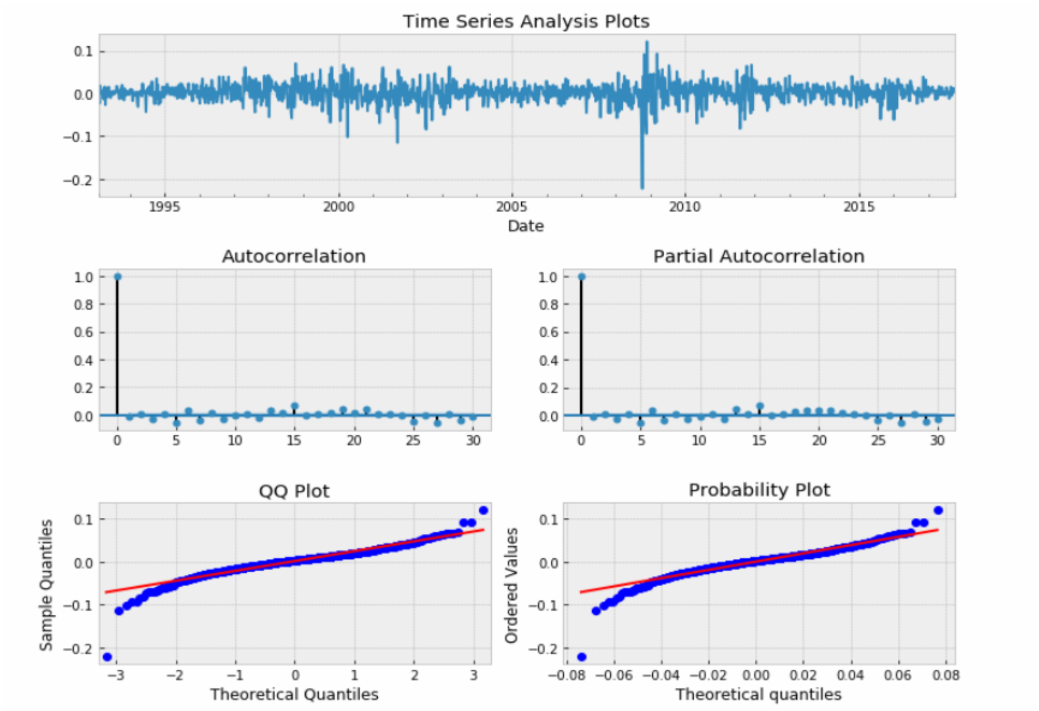


```
                              ARMA Model Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 5000
Model:                     ARMA(2, 2)   Log Likelihood               -7063.501
Method:                           mle   S.D. of innovations              0.994
Date:                Sun, 31 Dec 2017   AIC                          14137.002
Time:                        18:33:43   BIC                          14169.588
Sample:                             0   HQIC                         14148.423

==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1.y        0.4386      0.055      7.921      0.000       0.330       0.547
ar.L2.y       -0.2889      0.017    -17.176      0.000      -0.322      -0.256
ma.L1.y        0.4339      0.057      7.633      0.000       0.322       0.545
ma.L2.y       -0.1481      0.048     -3.101      0.002      -0.242      -0.054
                                    Roots
=============================================================================
                 Real           Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            0.7590           -1.6985j            1.8604           -0.1831
AR.2            0.7590           +1.6985j            1.8604            0.1831
MA.1           -1.5181           +0.0000j            1.5181            0.5000
MA.2            4.4470           +0.0000j            4.4470            0.0000
-----------------------------------------------------------------------------
```

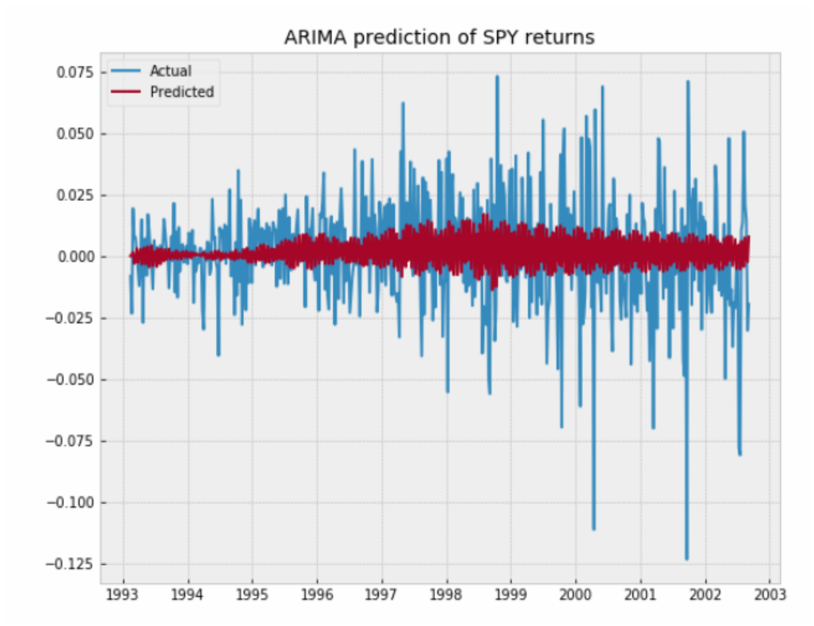
From the ARMA summary above, we see that our α, β parameters are recovered quite nicely. The true values are contained in the 95% confidence interval.

We can proceed now with fitting ARIMA(p,d,q) model to SPY returns. We iterate over the grid of potential parameters (p,d,q) and pick the best model based on the AIC information criteria. Based on the grid search, the best model was ARIMA(4,0,4) with the following plot of residuals:



We see that, volatility clustering is still present in the residuals. We will try to tackle it directly, by using ARIMA regression on the volatility itself with GARCH models.

Also, we can check how this model predicts the next step return. Here is a plot of actual and predicted returns:



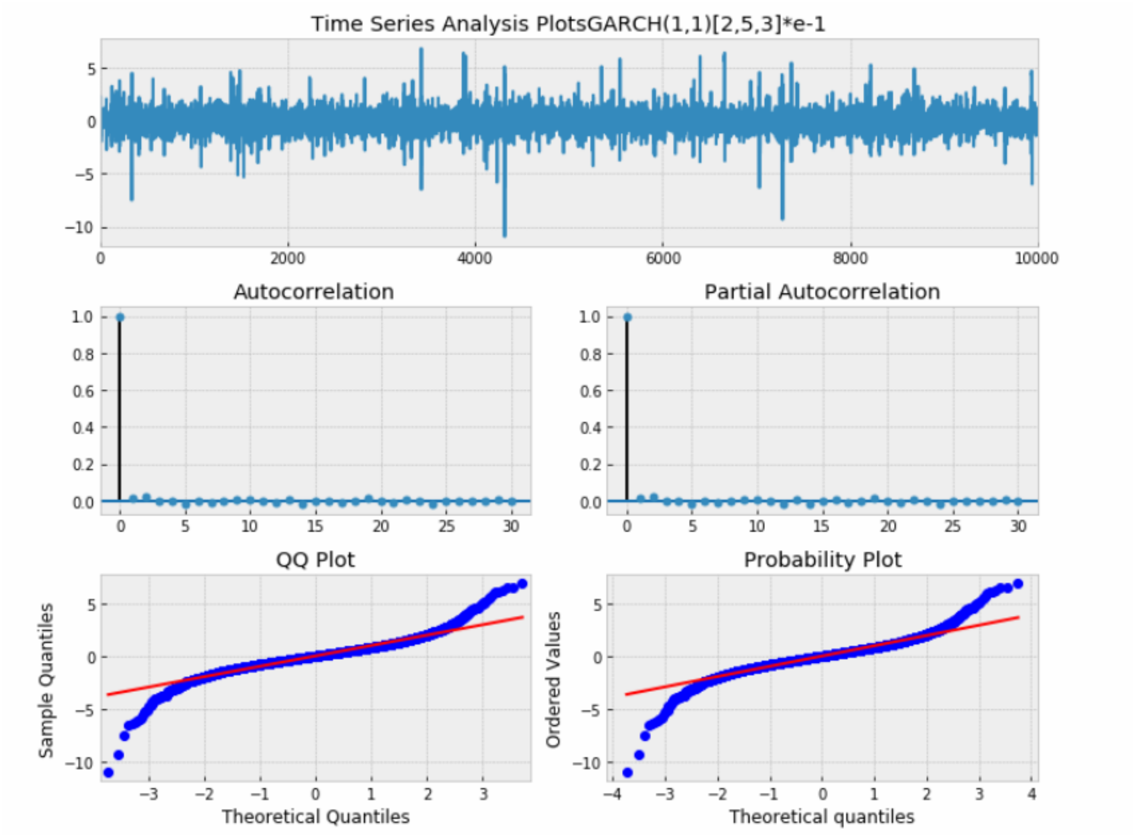
We see from this chart that, ARIMA predictions of returns are not very accurate. To summarize the accuracy of the model we calculate overall RMSE prediction error of prices, which turns out to be 35.49. We will compare this result with performance of deep learning models in Section 5.

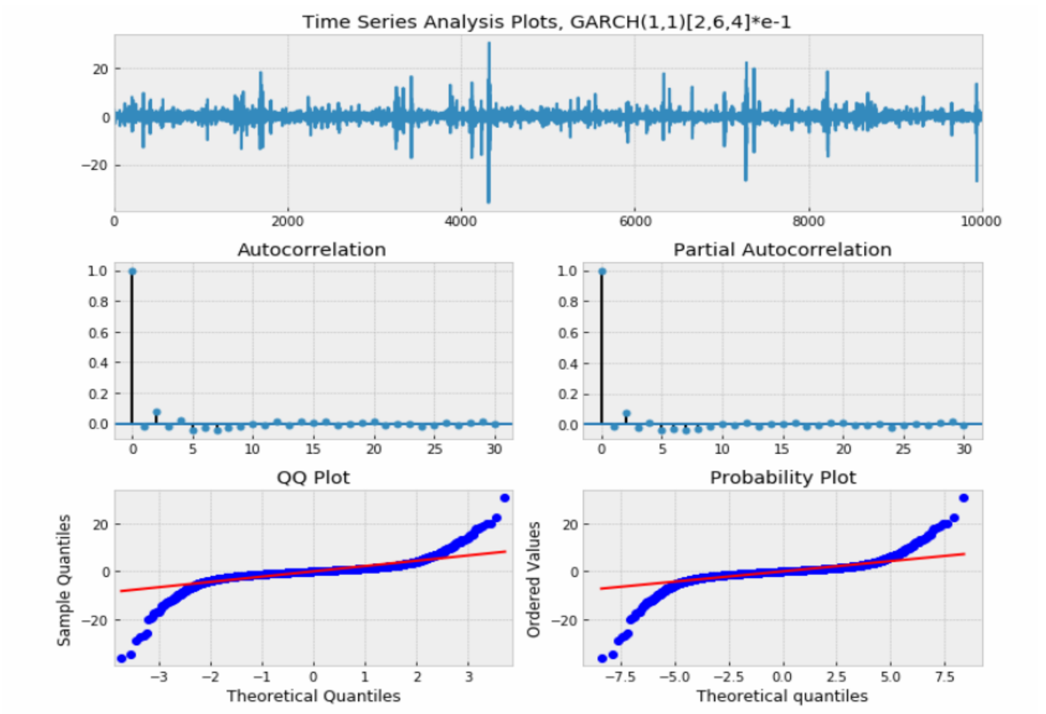
4.4 Generalized Autoregressive Conditionally Heteroskedastic Models - GARCH(p,q)

GARCH(p,q) model can be thought of as an ARMA model applied to the variance of a time series. The AR(p) part models the variance of the residuals (squared errors), while the MA(q) portion models the variance of the whole process. The basic GARCH(1, 1)[*α0*, *α1*, *β1*] formula is:

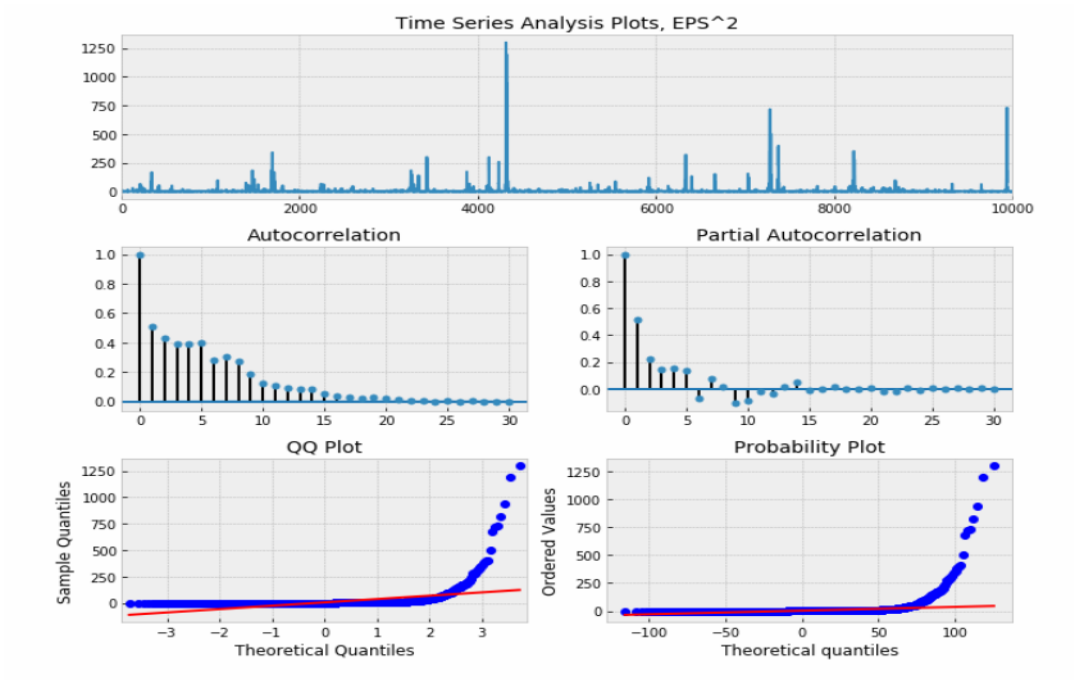
$$r_t = \mu + \epsilon_t \tag{6}$$
$$\epsilon_t = \sigma_t \omega_t \tag{7}$$
$$\sigma_t^2 = \alpha_0 + \alpha_1 \epsilon_{t-1}^2 + \beta_1 \sigma_{t-1}^2 \tag{8}$$

We generate data sample for the models GARCH(1,1)[.2,.5,.3] and GARCH(1,1)[.2,.5,.3] and plot the results:





From the figures above we see that higher $\alpha 1$, $\beta 1$ coefficients amplify volatility to a larger degree during periods of volatility clustering. Also, GARCH process generates fat tails (power law type decay rather than an exponential one), a feature ubiquitous in financial time series.



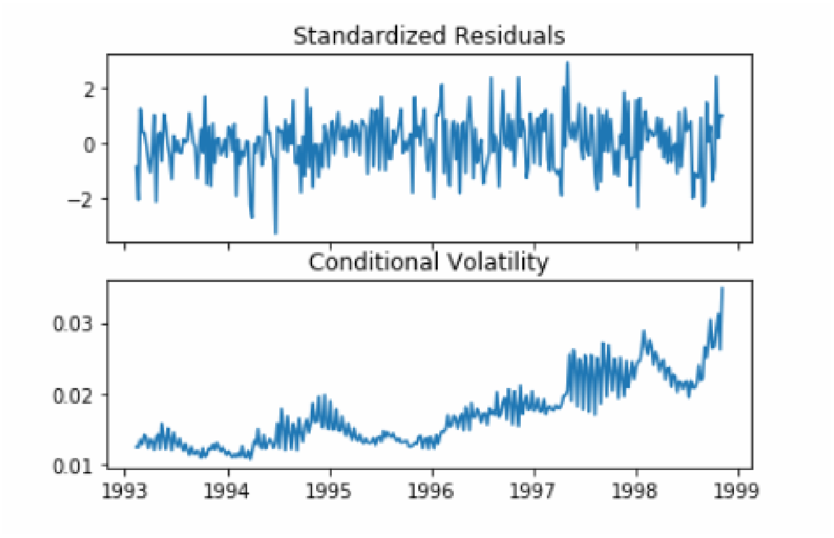
Looking at the ϵ^2 and its PACF, we conclude that we need both AR and MA terms. Let us try to recover them with ARCH module.

```
Iteration:      5,   Func. Count:     36,   Neg. LLF: 16519.17092086323
Iteration:     10,   Func. Count:     71,   Neg. LLF: 16389.34206067091
Optimization terminated successfully.    (Exit mode 0)
            Current function value: 16388.273514485376
            Iterations: 14
            Function evaluations: 95
            Gradient evaluations: 14
                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:                      y   R-squared:                      -0.000
Mean Model:             Constant Mean   Adj. R-squared:                 -0.000
Vol Model:                      GARCH   Log-Likelihood:               -16388.3
Distribution:                  Normal   AIC:                           32784.5
Method:            Maximum Likelihood   BIC:                           32813.4
                                        No. Observations:                10000
Date:                Sun, Dec 31 2017   Df Residuals:                     9996
Time:                        18:34:09   Df Model:                            4
                                 Mean Model
==============================================================================
                  coef    std err          t      P>|t|       95.0% Conf. Int.
------------------------------------------------------------------------------
mu         -8.7536e-04  8.762e-03 -9.991e-02      0.920 [-1.805e-02,1.630e-02]
                            Volatility Model
============================================================================
                 coef    std err          t      P>|t|  95.0% Conf. Int.
----------------------------------------------------------------------------
omega          0.2089  1.135e-02     18.410  1.086e-75 [  0.187,  0.231]
alpha[1]       0.5827  2.031e-02     28.685 5.843e-181 [  0.543,  0.622]
beta[1]        0.4012  1.359e-02     29.526 1.345e-191 [  0.375,  0.428]
============================================================================

Covariance estimator: robust
```

We successfully recovered $\alpha 0, \alpha 1, \beta 1$=[.2,.6,.4] coefficients for the GARCH(1,1) [.2,.6,.4] process! Now we are ready to use these tools on the real data of SPY returns. We pick the first 300 observations and attempt to find a suitable GARCH model.

After performing grid search for best parameters, we found that the best model was GARCH(3,0,3) with the following plot of residuals:



As we can see from the plot of residuals, they resemble white noise (without swings in volatility, i.e. the amplitude between local maxima and minima remains pretty stable), thus leading us to conclusion that the model was able to capture volatility clustering effects!

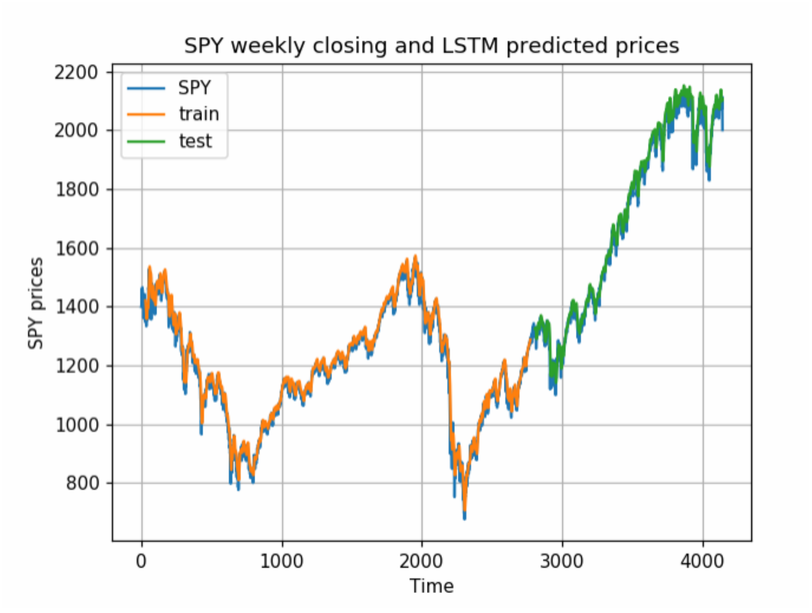
4.5 ARIMA and GARCH modeling conclusions

- ARIMA models can capture certain important effects (momentum, mean reversion), with the exception of volatility clustering;
- GARCH models are capable of explaining volatility clustering and fat tails phenomena in the financial time series;
- To achieve successful GARCH modeling, one has to do Walk Forward Optimization to adapt to changing market conditions;
- GARCH models can predict next day realized volatility (but not the sign) with high success percentages;
- Realized volatility is NOT implied volatility (which is part of option premiums). There is no easily available instrument to take advantage of this information, thus, no arbitrage opportunities, keeping the effect persistent over long time periods.

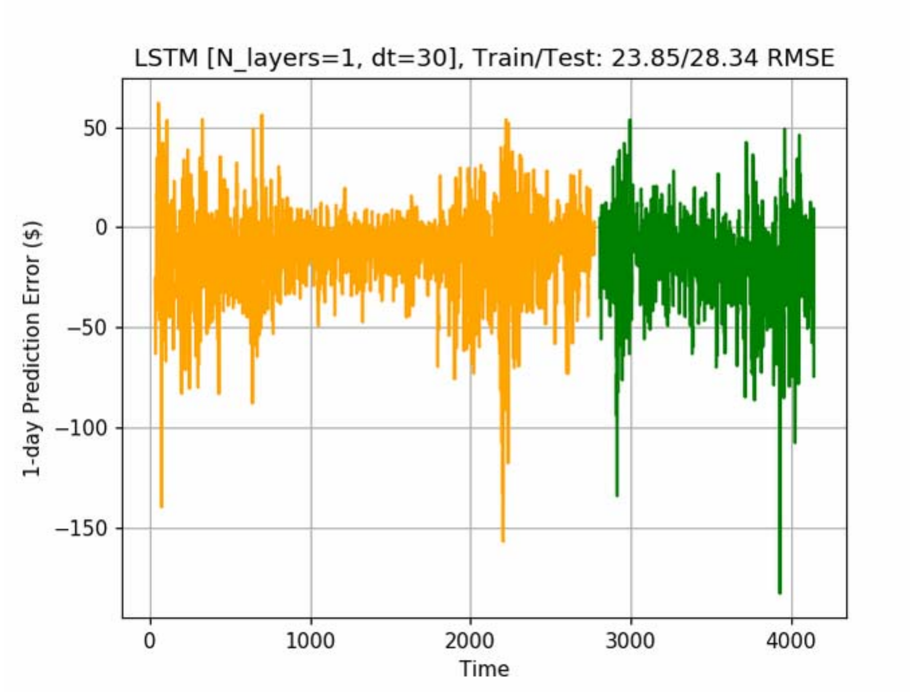
5. Application of deep learning methods for stock market prediction

In this Section we will apply one of the most successful deep learning techniques to the problem at hand: Long Short-Term Memory network. The Long Short-Term Memory network, or LSTM network, is a recurrent neural network (RNN), meaning that it has time loop connections (this feedback mechanism accounts for recurrence) applied to special kinds of neuron cells, called memory cells. Each cell contains gates which control input, output and memory flows. Such architecture enables learning long time sequences while overcoming a vanishing gradient problem which plagues standard RNNs.

We start with training a stateless, one layer LSTM network with 100 neurons and one dense output layer to predict prices 1-step ahead. We prepare our time series, SPY closing weekly prices, for training this network by reshaping the time series input in the form suitable for classification learning task. The look back parameter of the network determines the number of time series terms used for next step prediction. We vary it from around 30 to 100, while keeping internal cells stateless (without memory of prior sequences). Training the network was done via BPTT (Back Propagation Through Time) algorithm. Here is the result of training the network for 5 epochs with look back of 30 steps:

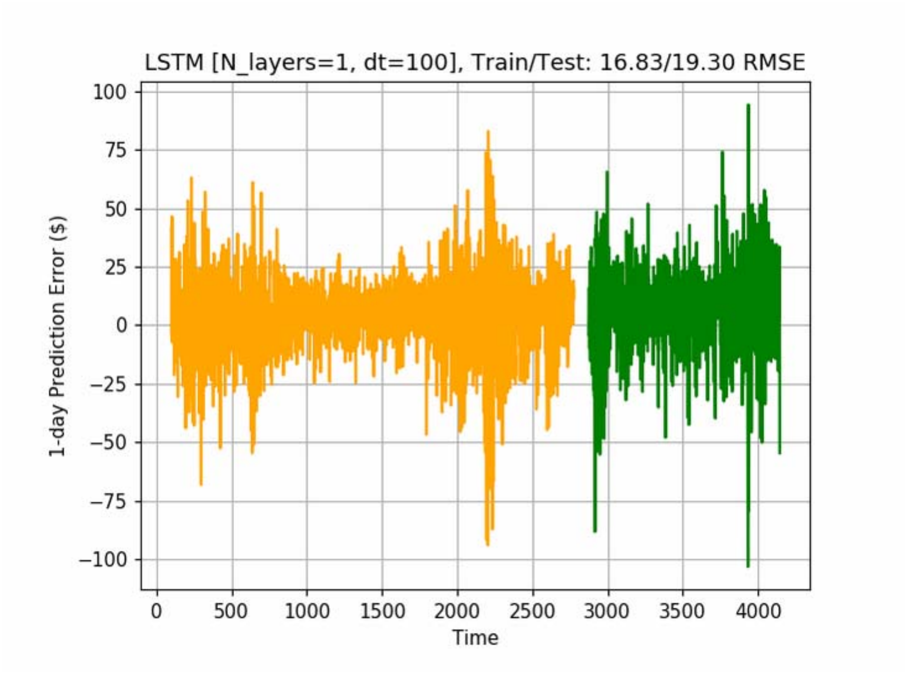


We assess performance by computing a root mean square error on training (orange color) and testing (green color) datasets (I adhere to this color scheme for the rest of this report). The result is: TrainScore = 23.85 RMSE, TestScore = 28.34 RMSE, with the 1-step prediction error plotted below:



Notice that while the network was able to learn and compensate for the non-stationarity of the series, some of it is visible in the plot above in the form of a systematic error visually recognizable as “drooping” of the test error. This artifact is corrected by longer training times and longer look-back periods (see below). However, even at this stage, this performance score is better than the score of 35.49 RMSE achieved with ARIMA modeling.

Training this stateless 1-layer network with longer look-back period of 100 led to a much better result with train/test scores of 16.83/19.30 RMSE:

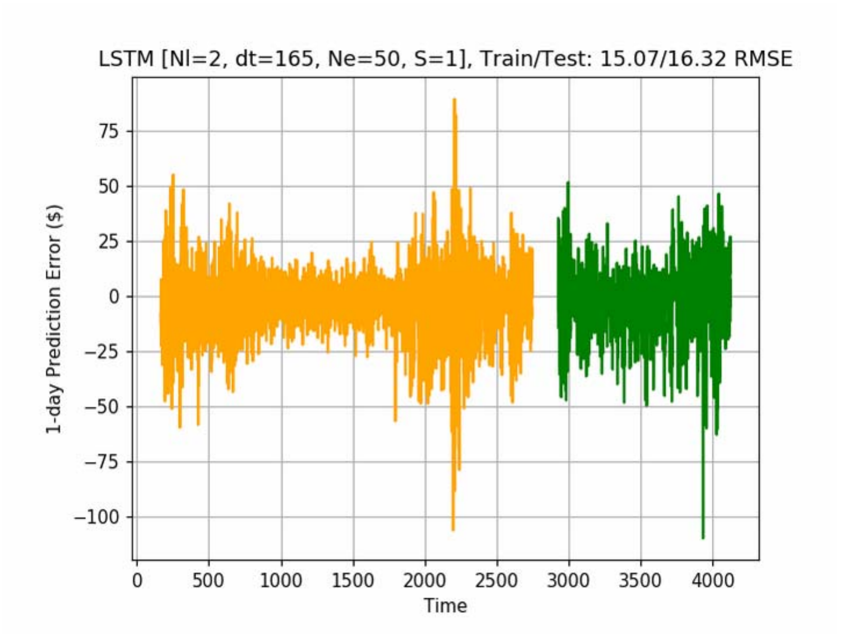


This is an interesting result. It implies that, the market has a memory, in contradiction to Market Efficiency Hypothesis (which is rejected by many practitioners); leading us to conclude that, longer past time sequences help predict future prices better.

To optimize our 1-layer stateless network we performed a hyper parameter search over the following parameters: 1) look back period, 2) number of epochs, 3) batch size. I already explained the meaning of a look back period. The number of epochs is the number of times the network is exposed to the data. Batch size determines the amount of input, before the weights are reset during training. I observed the following trends. Longer look back periods lead to better performance but require longer times to train (larger number of epochs). Increasing look back period by 30, requires correspondingly longer number of epochs to converge, roughly 5; varying batch size from 1 to 64 lead to a bell like curve in performance. First, performance improved up until size 11, and then it started to degrade. Therefore, we fixed batch size at 11 for all further experiments.

I also increased the depth of the network up to 5 layers, sandwiching Dropout layers of 20% for regularization. Such networks took too long to train, the most prominent error being inability to account for non-stationarity of the series. Recall, that 1 and 2 layer networks easily performed this task, after training sufficient number of epochs. Perhaps, this effect suggests that there might be better ways to train such a network, e.g. progressive training and stacking of layers (I have not investigated such approach).

Finally, I constructed statefull multi-layer networks, keeping track of learned internal states between training iterations. This required special data partitioning, ensuring that all tensor sizes are multiples of batch size. Again, training networks with more than 2 layers turned out to be too time consuming, therefore, we reach our conclusions by comparing 1 and 2 layer networks. One such conclusion is that, training a statefull network improves performance over a stateless one, especially if it has more than 1 layer. Here is the result of training a 2-layer statefull network with 165 look-back period:



6. Key findings

1) Application of deep learning methods to stock market prediction is a very promising endeavor, leading to results which surpass older techniques such as ARIMA.

2) Deep, stateless LSTM network with multiple stacked layers performs better than a one layer LSTM network.

3) Deep, statefull LSTM network with multiple stacked layers performs even better than a one layer LSTM network. This confirms my intuition that, a statefull network would keep track of its internal state, roughly corresponding to the state of the market (such as a trend), thus leading to better predictions.

4) One can feed the original price series without prior transformations (such as differencing) to the network; it is able to learn to account for the non-stationarity of the series. One of the goals of market analysis is to be able to understand trends and trend changes. Performing differencing leads to de-trending, thus, removing such information from further analysis. Such operation may be justified for other types of time series, but not for the study of trends in the financial series data.

5) The network learns to follow major trends, delivering good predictions on days/weeks when the market continues along. It also reacts pretty fast to trend changes. However, it cannot predict well counter-trend jumps. This is not entirely unexpected. To predict such counter-trend jumps well, it would need to learn more sophisticated patterns with more data, including data from higher time frames, and more complex network architecture.

6) Longer look back periods improved predictions. It also took longer to train such a network. Training for insufficient periods of time leads to non-stationarity of the time series to “seep back” into the prediction errors.

7) I found some evidence that, the network partially learned the effect of volatility clustering, predicted well by GARCH models. I see that, volatility clustering is still present in the error plots during both training and testing phases. Therefore, one needs to take special care to enable LSTM network to learn this effect.

7. Recommendations to clients:

1) If you do not use some form of trend analysis in your organization, or your in-house process is not reliable, it is time to incorporate such system for better hedging and overall future price predictions. My findings show that, deep learning methods can be successfully applied to stock market data.

2) In its current form, the model should not be used for volatility analysis. If the volatility modeling is desired, I recommend extending the model architecture to include volatility rolling window data as an additional input series with the t+1 volatility value being additional output training label.