National Research University Higher School of Economics
Faculty of Computer Science
Programme 'Master of Data Science'

Comparison of Time series forecasting methods

Master's Thesis

Student: Volkov, Leonid Y.

Supervisor: Chankin, Andrew

Moscow, 2021

Abstract

The idea of this work is to improve the performance of predictions of Time series by comparing AR class models with new generation models based on NeuroNets. The result of this analysis is intended to use in real world applications - Risk management and Portfolio construction of Asset management company.

Contents

List of Figures

Chapter 1

Introduction

Time series forecasting is a corner stone of any PM and RM systems. The importance of quality of Time series models is hard to overestimate here. The reliance on highly performed ones is the key competitive advantage and the way to success.

The main purpose to build a model which allow to simulate data with some widely know properties:

- Financial markets returns distributions demonstrate so called 'heavy tails' and more picked, because of long periods of markets inactivity changing by sharp prices moves. That means the returns are not Gaussian.
- Returns are uncorrelated but not independent, which means autocorrelation should be insignificant.
- They demonstrate so called 'Volatility clustering' - periods of low volatility is changed by high volatility periods.
- Returns are suffered by 'Leverage effect' - the negative correlation between returns and volatility.

The dataset used for this research is daily market capitalisation of companies were included in Dow Jones Industrial Average index accompanied with exchange traded volumes of their equities. The choice of dataset is made in such a way that is supposed convenient in the Assets Management Industry. Most of analysis below is made with market returns as they represent necessary stationarity, so Equities prices could be used exactly the same way. However the market caps allow to compare companies for the purpose of portfolio construction applications of the models (which is is beyond of the topic of current work), so they are so called 'market conventional' data. The dataset is split for the training part and test part. The dataset consists of 6742 observations (Market Caps and volumes). The original Market caps dataset is significantly larger, but it suffers of significant data missing before June 07, 1999. The dataset is split to train and test set (returns) so train set has 5000 observations (rich enough to train NeuroNet) and the 629 observations.

First part of research consists of the overview of AR models used here. Then the plan of their analysis is presented. Then the brief result of models performance is provided together with some quality metrics. For models building and

performance demonstration purpose 5 Equity names were chosen by the latest sizes of Market caps among the 30 presented in dataset. The rest of Equities were avoid for not to overwhelm the research overload. Market caps sorted in descending order are are presented in the Figure 1.

Figure 1.1: Market capitalisations of DJIA as of March 21, 2021

Dates	2021-05-21
Ticker	
AAPL	2.116158e+06
MSFT	1.861956e+06
JPM	4.909396e+05
JNJ	4.542081e+05
WMT	4.019059e+05
UNH	3.938777e+05
HD	3.428888e+05
PG	3.389578e+05
DIS	3.128030e+05
XOM	2.522343e+05
KO	2.374011e+05
VZ	2.370189e+05
INTC	2.279047e+05
PFE	2.265946e+05
CSCO	2.223614e+05
NKE	2.107913e+05
MRK	2.026406e+05
CVX	2.019598e+05
MCD	1.740228e+05
C	1.591153e+05
LIN	1.567415e+05
BA	1.375473e+05
RTX	1.311765e+05
IBM	1.296412e+05
CAT	1.295133e+05
GS	1.287084e+05
AXP	1.265202e+05
MMM	1.178363e+05
DOW	1.158781e+05
TRV	4.030737e+04

Chapter 2

Pure AR models description

2.1 Formal AR models

As Financial assets prices and thus Market caps of companies are non stationary processes (Figure 2.1), the first step to analysis is to move to returns r_t. :

$$r_t = \frac{MCap_t}{MCap_{t-1}} - 1$$

where $Mcap_t$ is Market cap of company at the point of time t.

The stochastic processes modelling old days are represented by wide family of AR models like ARIMA and ARCH. Among all I've chosen ARCH models such GARCH (Bollerslev, 1986 [1]) with Gaussian noise Standard AR-GARCH model is represented by:

$$r_t = \mu + \sum_{i=1}^{s} \psi_{L_i} r_{t-L-i} + \epsilon_t$$

$$\epsilon_t = e_t \sigma_t$$

$$\sigma_t^2 = \omega + \sum_{k=1}^{q} \beta_k \sigma_{t-k}^2$$

Where r_t is returns, $L_i \in N$ - are chosen time lags of AR process and $e_t \sim \mathcal{N}(0, 1)$.

GARCH model popularity inspired the fast development of stochastic volatility models as EGARCH, COGARCH, HARCH and many others with Gaussian and non-Gaussian noise. These models are very well described in paperwork and widely used in practice. For the purpose of current research the ARCH package by [2] Kevin Sheppard is used.

The natural choice of GARCH as background AR model is supposed to be natural. Then the choice of EGARCH is based on the undermined inclusion of fat tails effect Leverage effect.

AR-EGARCH models are represented by:

$$r_t = \mu + \sum_{i=1} {}_s\psi_{L_i} r_{t-L-i} + \epsilon_t$$

$$\epsilon_t = e_t \sigma_t$$

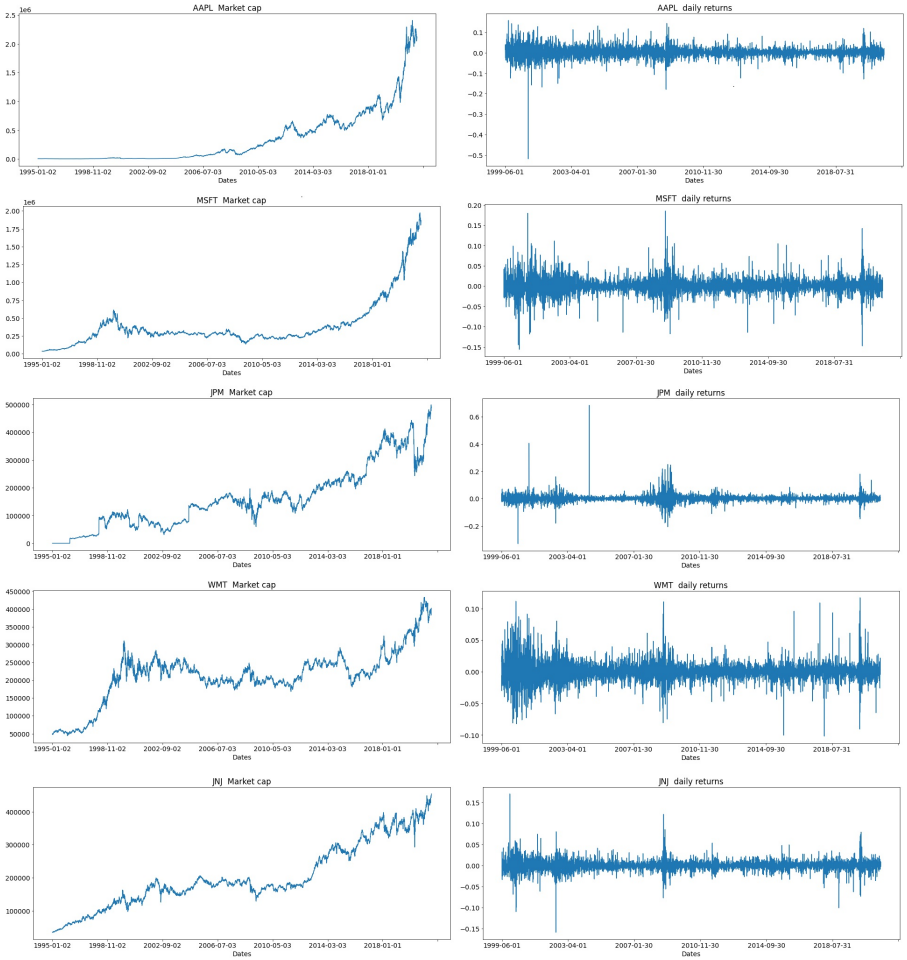
$$\ln \sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i \left(|e_{t-i}| - E[e_i] \right) + \sum_{j=1}^{o} \gamma_j e_{t-j} + \sum_{k=1}^{q} \beta_k \ln \sigma_{t-k}^2$$

Where r_t is returns, $L_i \in N$ - are chosen time lags of AR process and $e_t \sim GED(\nu)$ or $e_t \sim SkewStudent(,\lambda)$.

The choice of AR lags for return process is based on autocorrelation analysis and building of autocorrelation charts and common sense logic that main lags should be 1, 5, 21, 22, 250 working days which is consistent with daily, weekly, monthly and yearly cycles.

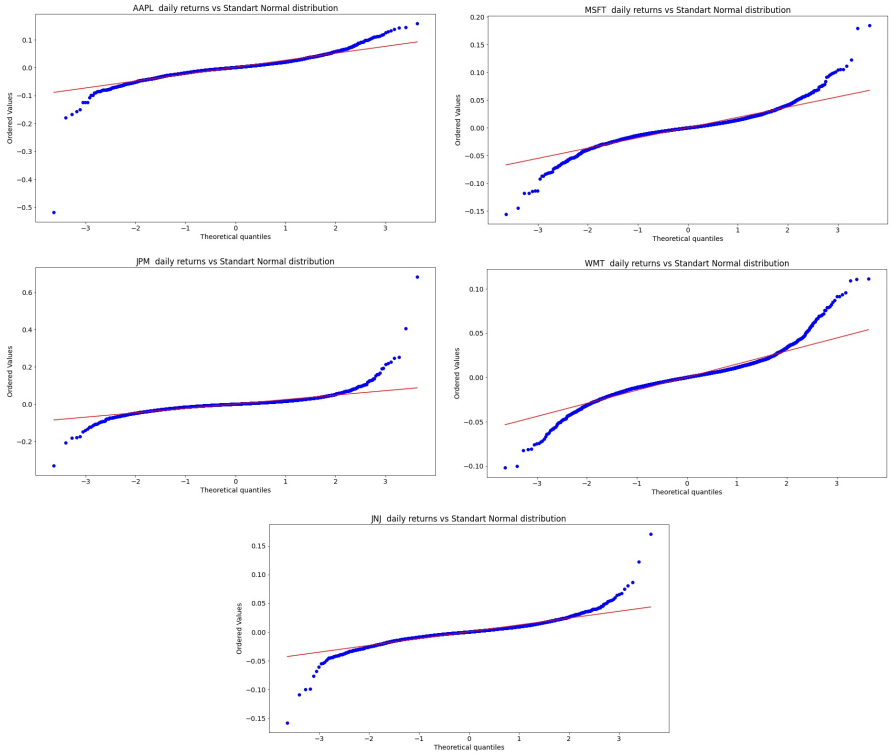
The $GED(\nu)$ distribution is the natural choice to model fat tails for the case it is not significantly considered by the term $\sum_{j=1}^{o} \gamma_j e_{t-j}$ in volatility process equation and in the case $\nu = 2$ $GED(\nu)$ is equivalent to Gaussian distribution.

Figure 2.1: Market caps and returns



Skew Student distribution is chosen for the same purpose is the case of leverage effect captured by the term $\sum_{i=1}^{p} \alpha_i \left(|e_{t-i}| - E[e_i]\right)$

Figure 2.2: Returns distribution VS Normal distribution



First let's show that returns are uncorrelated and non-Gaussian. As it can bee seen in the Figure 2.2, the tails of returns are away from the levels they should be in the case of Normal distribution. And on the autocorrelation charts in the Figure 2.3 the autocorrelation is not significant and it's absolute value does not exceed 0.25 (with only one exception). So, as it is seen we could suppose they uncorrelated, but as it will be shown below there is some interdependency and calendar cycles which could be useful to build meaningful predictive models of returns and volatility.

The choice of time lags for models was made based on calendar cycles (1 week, 1 month and 1 year). Also 1 day lag was added. As dataset presents only working days, lags of 1 day, 5 days, 21 days 22 days and 250 days were taken. All models were built using the Statmodels and ARCH Python packages. (!). For numerical stability returns were multiplied by 100. First model built for this research is AR[1, 5, 21, 22, 250]-GARCH(1,1). Then based on it's result using P-value of t-statistic one lag among 21 days or 22 days is chosen for followed research. Then 2 more models built: AR-EGARCH(3, 3, 3) with GED distribution of random process and with Skew Student distribution of random process. Then again, using P-value of t-statistic the meaningful AR-lags and ARCH lags were chosen and the same models with less parameters were built. The choice

of best one were made based on max Log Likelihood (which was equivalent to the choice based on Akaike information criterion (AIC)). Then the best model was used to simulate forecasts and compare results with the test set which then were checked for desirable properties presence.

First the forecasted returns were checked for the presence of significant autocorrelation and compared with real ones. Then I checked the Volatility clustering by autocorrelation analysis of squared simulated returns and compared with the autocorrelation of squared real returns. After that the leverage effect also was analyzed by plotting autocorrelation charts between returns and squared lagged returns. We should see significant negative autocorrelation there. And after that the real returns and simulated returns were compared with Gaussian distribution.

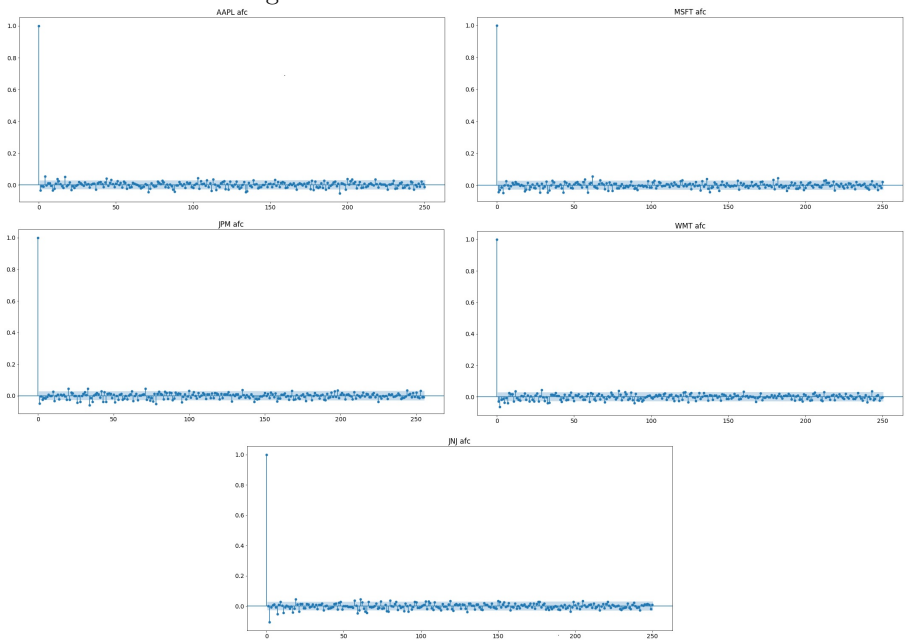
Autocorrelation is a correlation of time series with themselves with some time lag τ:

$$C(x, \tau) = Corr(x_{\tau:t}, x_{1:t-\tau})$$

For the autocorrelation of returns a series of the autocorrelations are used $Corr(r_{\tau:t}, r_{1:t-\tau})$ where $\tau \in 1..L$. For Volatility clustering it is an autocorrelation of squared returns $Corr(r^2_{\tau:t}, r^2_{1:t-\tau})$ where $\tau \in 1..L$. And Leverage effect is an autocorrelation of returns with the squared returns $Corr(r_{\tau:t}, r^2_{1:t-\tau})$ where $\tau \in 1..L$. L is number of lags used. Here 255 lags were used.

Finally the difference between real and simulated data distributions would be measured by Wasserstein distance (or Earth Mover Distance - EMD):

Figure 2.3: Autocorrelation of returns



$$EMD(P,Q) = \inf_{\gamma \in \pi} E_{x_1, x_2 \sim \gamma} ||x_1 - x_2||_2$$

where $x_1 \sim P$, $x_2 \sim Q$.

2.2 Findings

Figure 2.4: Log Likelihoods of models

AAPL		MSFT		JPM		WMT		JNJ	
Simple GARCH	-10357.46	Simple GARCH	-9059.22	Simple GARCH	-10583.63	Simple GARCH	-7835.32	Simple GARCH	-6727.99
ARCH 1-5-22-250 EGARCH 3-3-3 GED	-10074.98	ARCH 1-5-21-250 EGARCH 3-3-3 GED	-8673.94	ARCH 1-5-21-250 EGARCH 3-3-3 GED	-9305.10	ARCH 1-5-21-250 EGARCH 3-3-3 GED	-7563.98	ARCH 1-5-22-250 EGARCH 3-3-3 GED	-6513.05
ARCH 1-250 EGARCH 1-3-1 GED	-10082.74	ARCH 1-5-21-250 EGARCH 1-3-1 GED	-8679.38	ARCH 1-21-250 EGARCH 1-1-1 GED	-9312.01	ARCH 1-5 EGARCH 1-1-2 GED	-8186.54	ARCH 1-22 EGARCH 1-1-2 GED	-7039.04
ARCH 1-5-22-250 EGARCH 3-3-3 SkewSt	-10038.71	ARCH 1-5-21-250 EGARCH 3-3-3 SkewSt	-8621.73	ARCH 1-5-21-250 EGARCH 3-3-3 SkewSt	-21337.35	ARCH 1-5-21-250 EGARCH 3-3-3 SkewSt	-7510.73	ARCH 1-5-21-250 EGARCH 3-3-3 SkewSt	-6488.05
ARCH 1-5 EGARCH 1-3-1 SkewSt	-10750.86	ARCH 1-5 EGARCH 1-1-2 SkewSt	-8628.87	ARCH 1-21-250 EGARCH 1-1-1 SkewSt	-9187.2	ARCH 1-5 EGARCH 1-1-1 SkewSt	-8134.02	ARCH 1-22 EGARCH 1-1-2 SkewSt	-6994.60

On the Figure 2.4 the names of built models with some their parameters are presented with their Log Likelihoods. After choosing the model it was used to build 100 predictions, each of the size of test dataset. Then the average path was taken as the forecast.

The Figure 2.5 shows that the autocorrelation of real data and autocorrelation of forecasts are quite similar in diapasons. However the values of autocorrelation itself are quite small and look random. So we can say that neither real data nor simulated data are correlated.

The Figure 2.6 shows that Volatility clustering is significant in real returns but not significant in simulated returns. However such poor dynamics of predictions is the consequence of the averaging simulated paths. Taking several random paths can easily show prominent Volatility clustering. The stylized fact of high autocorrelation with close lags is also persists. The evidences of such dynamics could be found in additional materials.

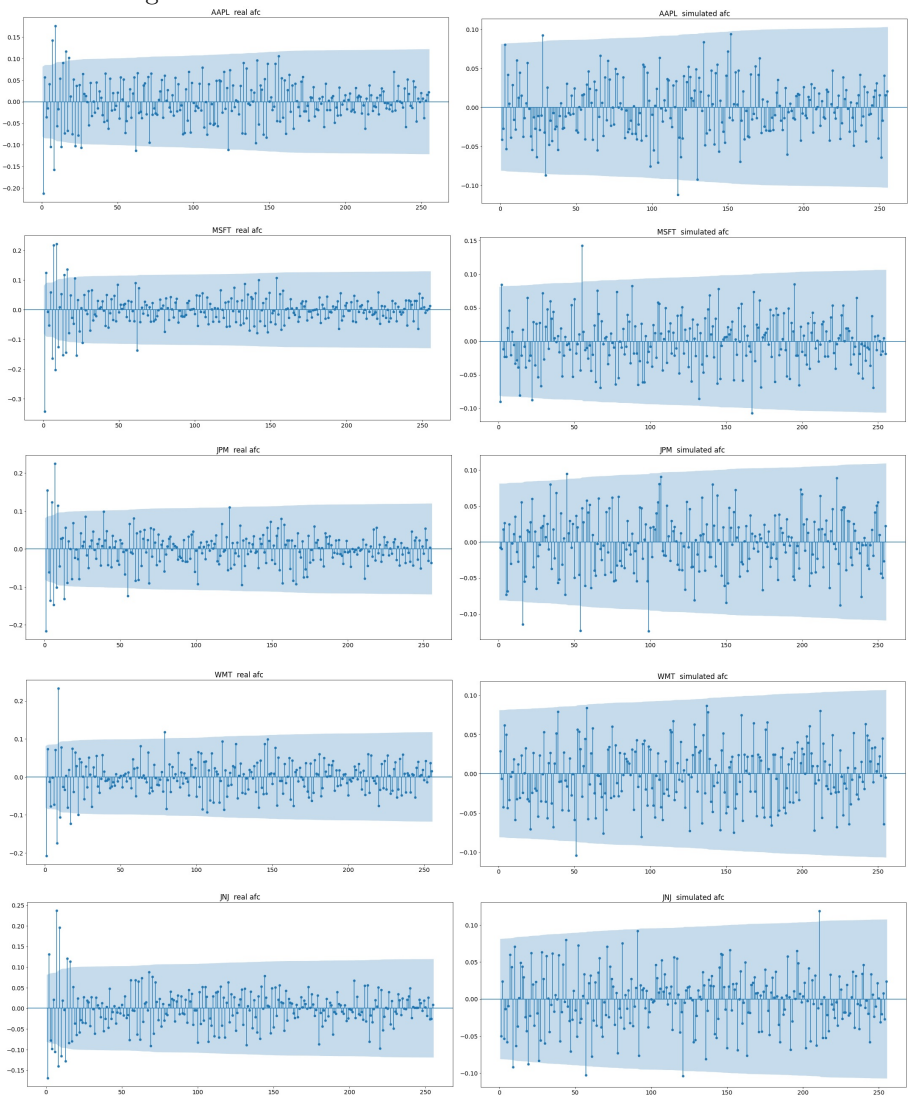
The Figure 2.7 shows that the Leverage not as prominent as Volatility clustering neither in real returns nor in simulated ones. This might be the consequence of random presence of this effect and long periods of lack of it. So it is local phenomena and probably highly individual one thus could be the topic of separate research itself. To catch this effect we need to pick short periods of high volatility of returns manually in each individual equity and measure its persistence there. So it is not possible to provide any consistent methodology that could easily used for the purpose of this research.

The Figure 2.8 shows the distributions of randomly generated individual forecasts. As we can see the distributions of forecasts are close to the distributions of real data. Also some of the distributions of returns are quite similar with Normal distribution but others have large divergence. However these plots

are very sensitive to the samples size and seems that with bigger samples the distributions of returns and forecasts would be closer to Normal one. That makes sense as with larger horizon of forecasts we are able to build more precise predictions then in the short run.

After several rebuilding of forecasts it became obvious that the ARCH models forecasts are quite unstable. In the Figure 2.9 the statistics of Wasserstein differences between test set and 100 different forecast simulations is presented. The difference between minimal and maximal Wasserstein distances is significant and fluctuate from 5 to 10 times. However it is not easy to see how different the forecasts are and what would be the difference when we turn back from re-

Figure 2.5: Autocorrelation of real data vs model forecast

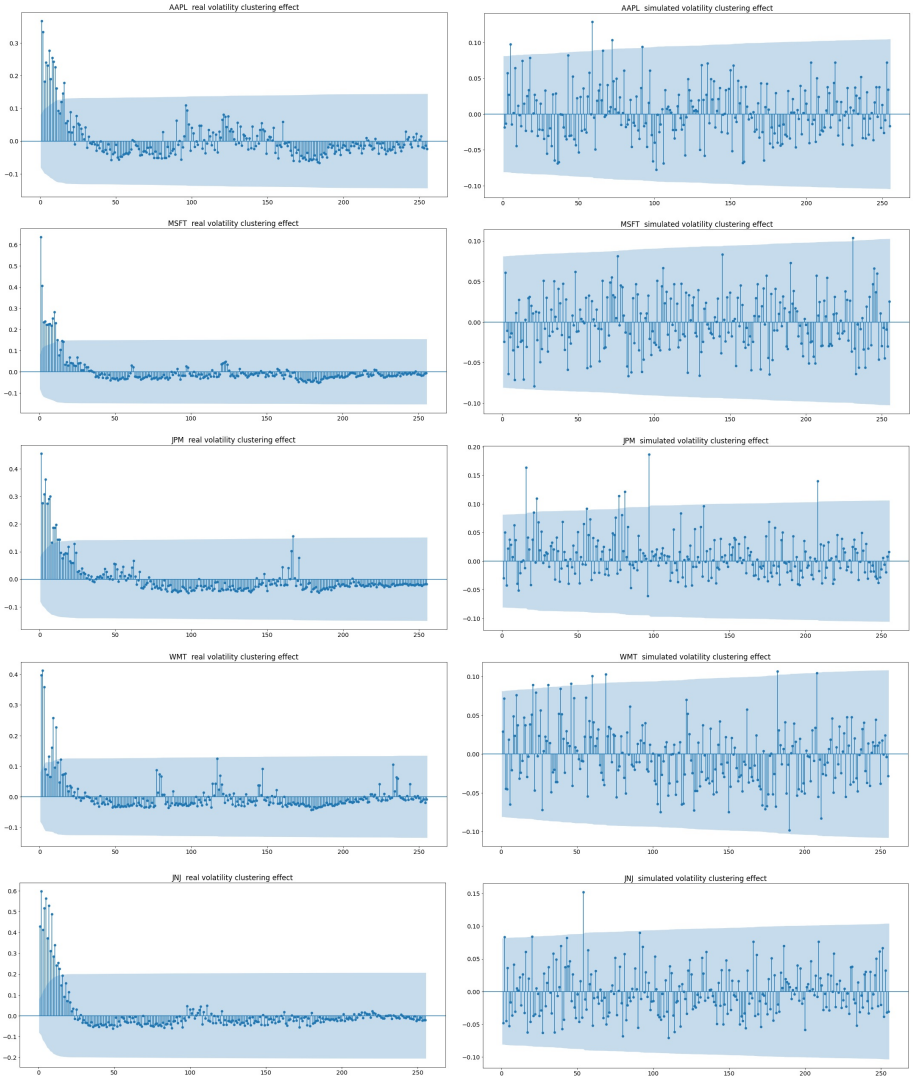


turns to Market caps.

In the Figure 2.10 we can see differences between real Market caps and simulated market caps for AAPL. It was simulated 100 different paths and the charts for that with minimal Wasserstein distance (upper charts) maximal Wasserstein distance (lower charts) and average path (middle charts) were built. We can clearly see the significant differences between them, which demonstrates instability of predictions caused by the nature of AR models.

The same is on the figure 2.11 built for MSFT. We see that depending on the particular model how fast and strong could the distribution of forecasts erode.

Figure 2.6: Volatility clustering of real data vs model forecast



Even mean path is far away from the actual shape of distribution of returns. The rest of charts and more experiments are available in supplement materials (Jupyter notebooks).

Figure 2.7: Leverage of real data vs model forecast

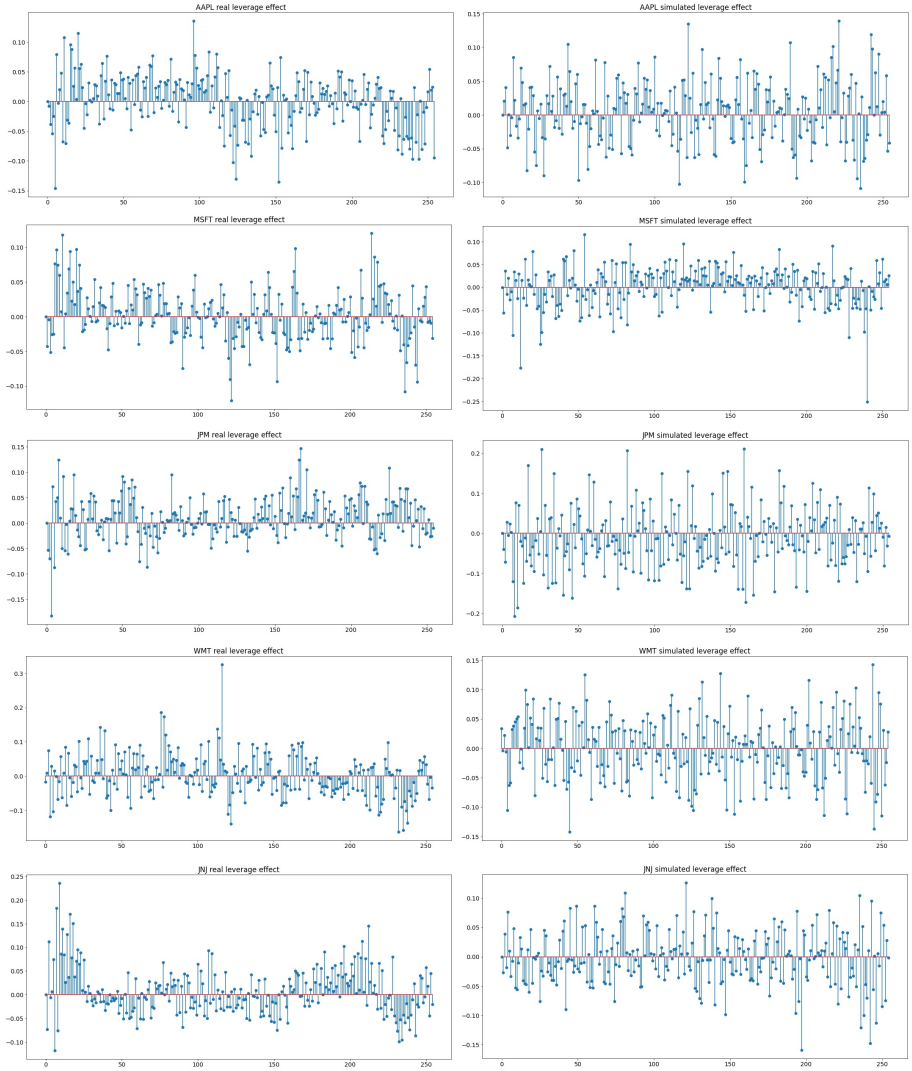


Figure 2.8: Distributions of forecasts vs real data, and theoretical distributions

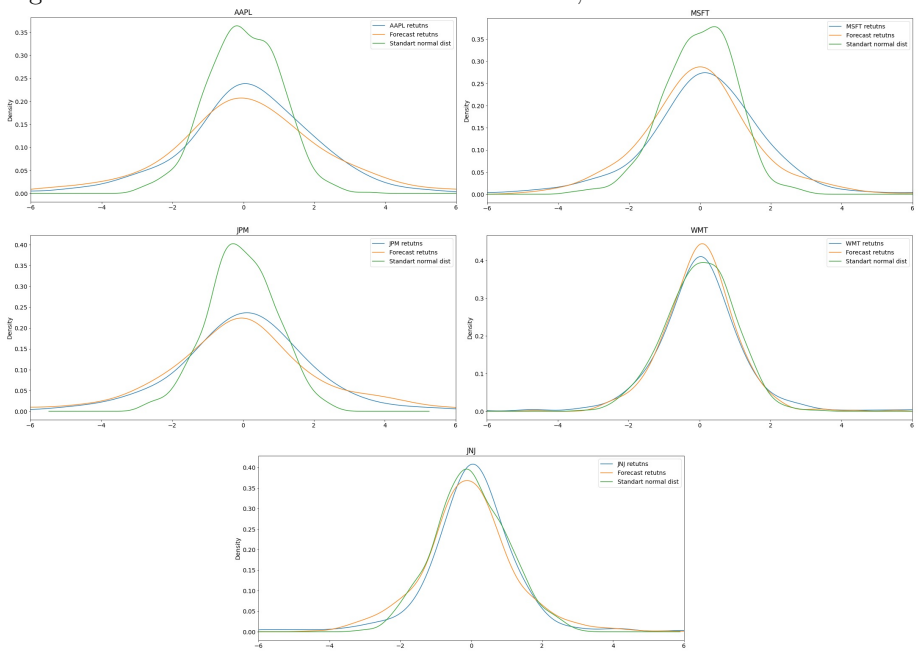


Figure 2.9: Wasserstein distances

	AAPL	MSFT	JPM	WMT	JNJ
mean	0.004198	0.004003	0.004658	0.002193	0.002255
std	0.002375	0.001540	0.002667	0.000849	0.000674
min	0.001256	0.001461	0.001677	0.001080	0.000972
25%	0.002481	0.002798	0.002682	0.001663	0.001777
50%	0.003777	0.003683	0.004086	0.002005	0.002090
75%	0.005301	0.005118	0.005961	0.002460	0.002728
max	0.018667	0.007953	0.020005	0.006292	0.003962

Figure 2.10: AAPL real Market caps VS forecasts

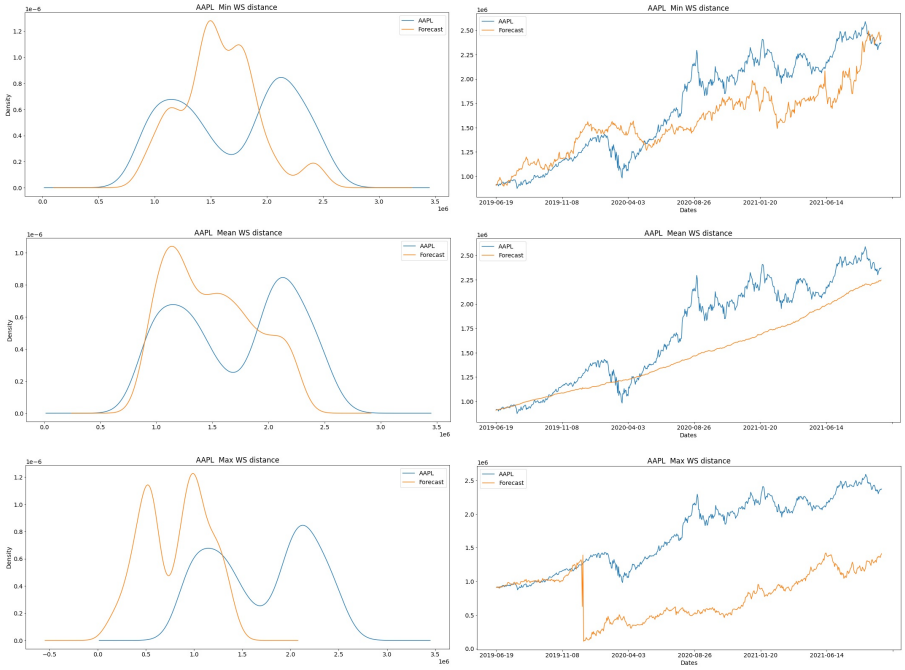
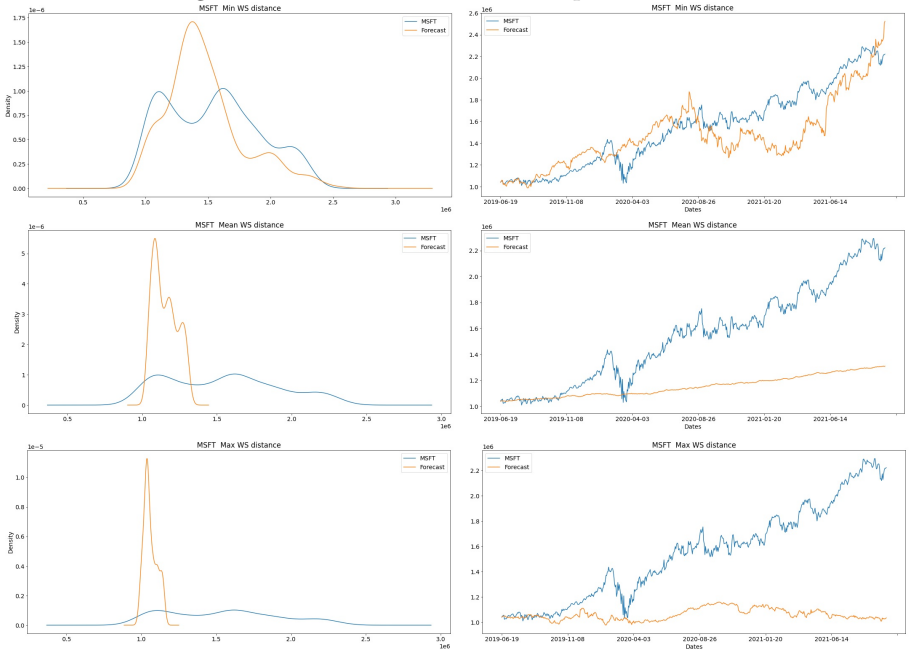


Figure 2.11: MSFT real Market caps VS forecasts



Chapter 3

AR models with exogenous variable

3.1 Model description

In this chapter I present the results of AR models with exogenous variable. Usually any variable or, speaking in terms of Data science, feature except the modelled process itself could be referred as exogenous variable in AR models.

GARCH model then gets the form:

$$r_t = \mu + \sum_{i=1}^{s} \psi_{L_i} r_{t-L-i} + \phi x_{t-1} + \epsilon_t$$
$$\epsilon_t = e_t \sigma_t$$
$$\sigma_t^2 = \omega + \sum_{k=1}^{q} \beta_k \sigma_{t-k}^2$$

EGARCH model are represented by:

$$r_t = \mu + \sum_{i=1} {}_s\psi_{L_i} r_{t-L-i} + \phi x_{t-1} + \epsilon_t$$
$$\epsilon_t = e_t \sigma_t$$
$$\ln \sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i \left(|e_{t-i}| - E[e_i] \right) + \sum_{j=1}^{o} \gamma_j e_{t-j} + \sum_{k=1}^{q} \beta_k \ln \sigma_{t-k}^2$$

Where r_t is returns, $L_i \in N$ - are chosen time lags of AR process and $e_t \sim GED(\nu)$ or $e_t \sim SkewStudent(, \lambda)$ and new addition ϕx_{t-1} is usually the process which time step is the same and ended before our targeted process. The last requirement is just supposed to ensure that we build 'fair' model, which is not trying to leak in the future.

Here and after as exogenous variable the changes in stock exchanges daily volumes in each Equity are used:

$$x_t = \frac{Volumes_t}{Volumes_{t-1}} - 1$$

3.2 Findings

Here following exactly the same logic as in Chapter 2 I start from presenting the Log Likelihood table with some models parameters. The models structures, lags and methods of selecting models and their parameters repeat the frame used in previous ones.

Figure 3.1: Log Likelihoods of models with exogenous variable

	AAPL		MSFT		JPM		WMT		JNJ
Simple GARCH	-10354.7	Simple GARCH	-9054.9	Simple GARCH	-10575.4	Simple GARCH	-7832.6	Simple GARCH	-6726
ARCH 1-5-22-250 EGARCH 3-3-3 GED	-10072.1	ARCH 1-5-21-250 EGARCH 3-3-3 GED	-8668.85	ARCH 1-5-21-250 EGARCH 3-3-3 GED	-9302.54	ARCH 1-5-21-250 EGARCH 3-3-3 GED	-7561.21	ARCH 1-5-21-250 EGARCH 3-3-3 GED	-6511.45
ARCH 1-250 EGARCH 1-3-1 GED	-10079.7	ARCH 1-5-21-250 EGARCH 1-1-1 GED	-8678.14	ARCH 1-5 EGARCH 1-1-1 GED	-9311.68	ARCH 1-5 EGARCH 1-1-2 GED	-8183.68	ARCH 1 EGARCH 1-1-2 GED	-7082.77
ARCH 1-5-22-250 EGARCH 3-3-3 SkewSt	-10087.4	ARCH 1-5-21-250 EGARCH 3-3-3 SkewSt	-83081.3	ARCH 1-5-21-250 EGARCH 3-3-3 SkewSt	-62751.1	ARCH 1-5-21-250 EGARCH 3-3-3 SkewSt	-7508.1	ARCH 1-5-21-250 EGARCH 3-3-3 SkewSt	-6487.46
ARCH 1-250 EGARCH 2-2-3 SkewSt	-10753.9	ARCH 1-5-21-250 EGARCH 1-1-1 SkewSt	-8623.74	ARCH 1-5-21-250 EGARCH 1-1-1 SkewSt	-143374	ARCH 1-5 EGARCH 1-1-1 SkewSt	-8131.08	ARCH 1 EGARCH 1-1-2 SkewSt	-7038.81

Charts on the Figures 3.2 - 3.4 show the same performance of forecasts compared to same ones in the Figures 2.5 - 2.7. Autocorrelations of returns show no prominent structure. The Volatility clustering hasn't been captured by the models. Leverage effects also doesn't look prominent. However distribution charts in the Figure 2.8 look better then in the Figure 3.5 as the forecasted returns distributions much closer to real returns distributions in the models in previous chapter then in current ones. But we can't rely on this results as they are based on randomly single forecasts (which are quite volatile itself).

The table on Figure 3.6 also shows no significant improvement in Wasserstein distances statistics for forecasted returns compared to Figure 2.9. And so Figures 3.7 and 3.8 show quite poor performance of forecasts compared to real data.

Figure 3.2: Autocorrelation of real data vs model forecast

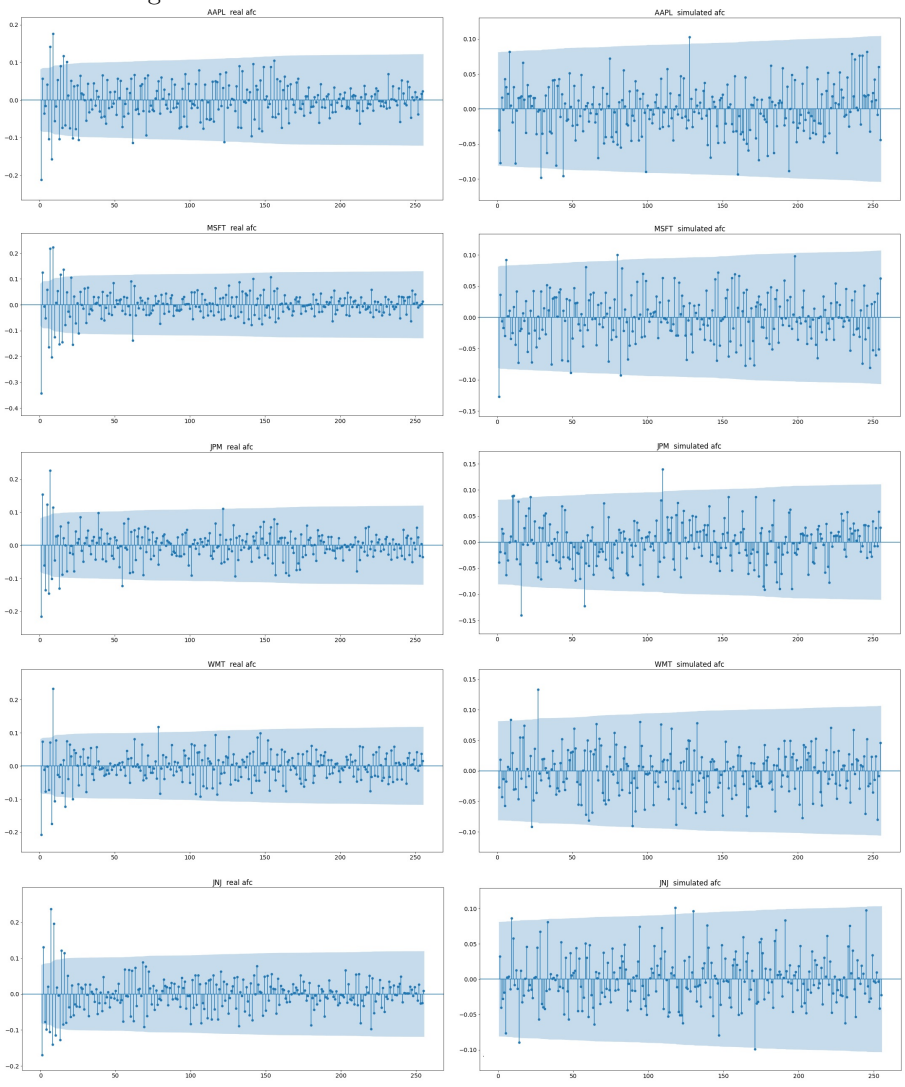


Figure 3.3: Volatility clustering of real data vs model forecast

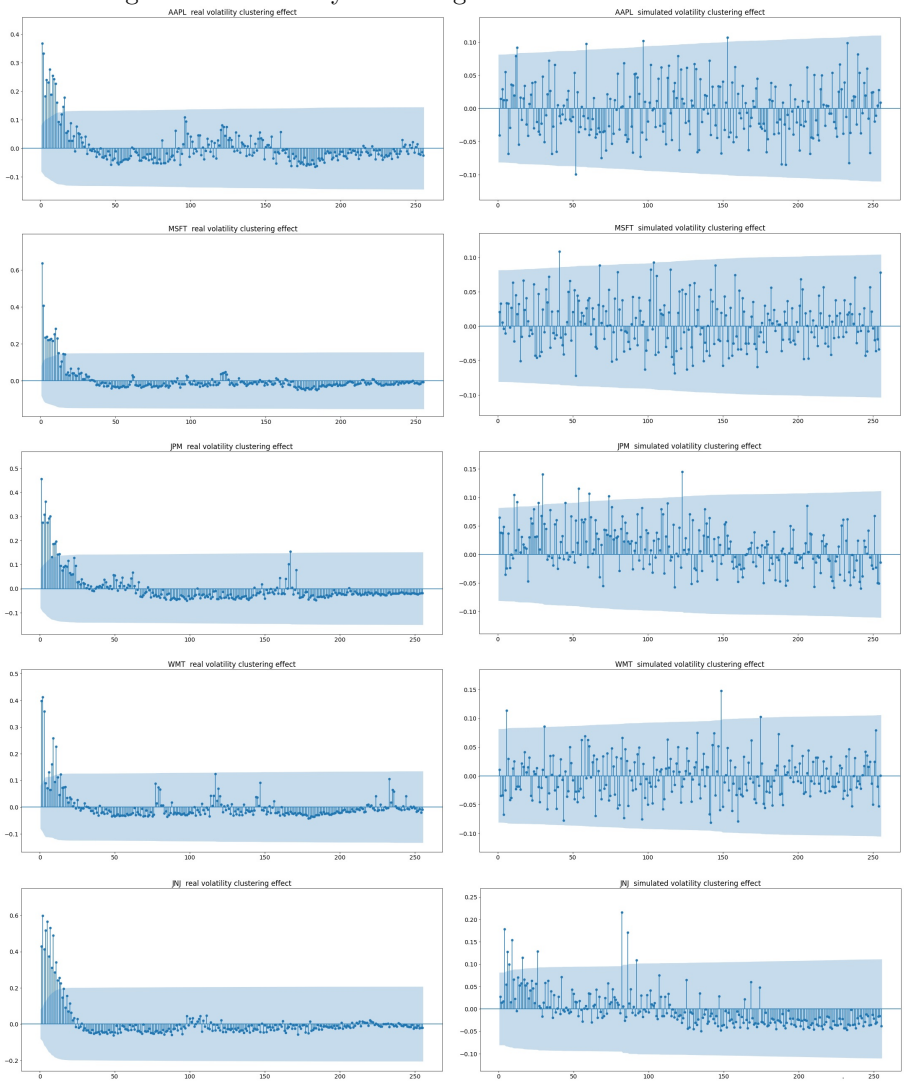


Figure 3.4: Leverage of real data vs model forecast

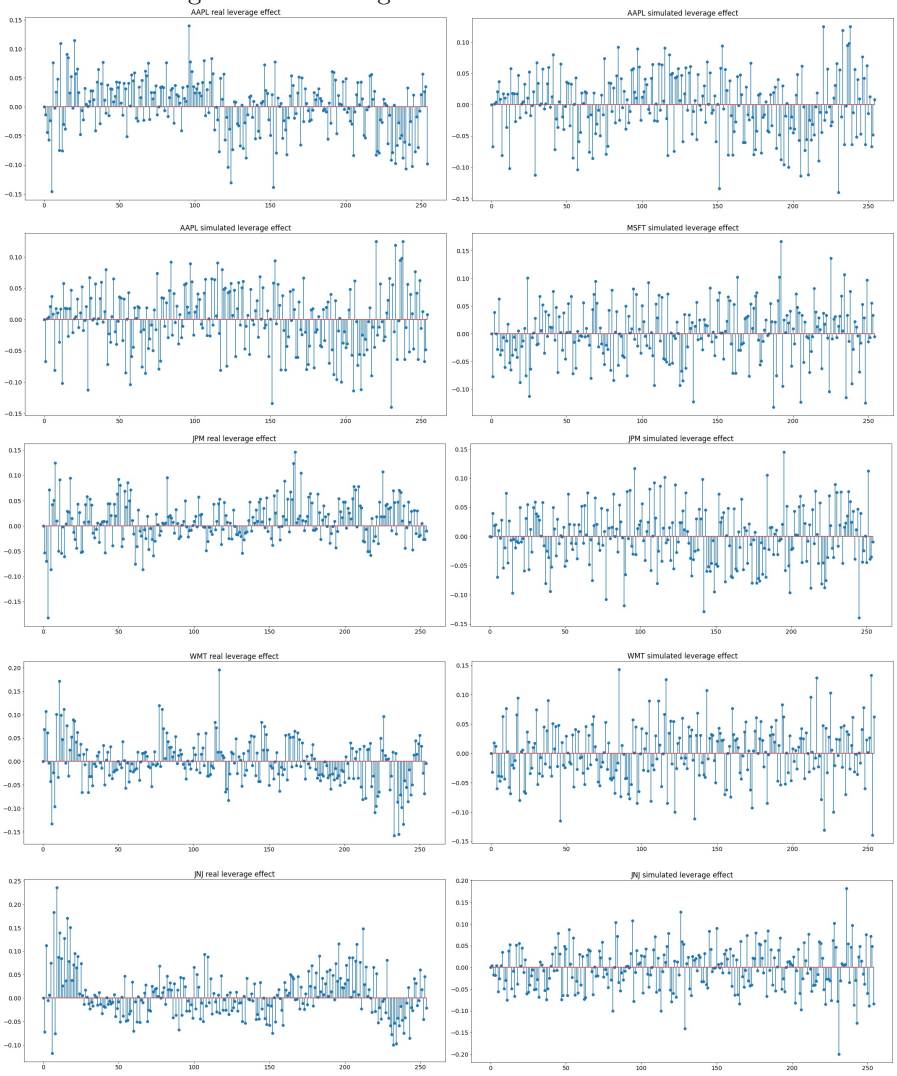


Figure 3.5: Distributions of forecasts vs real data, and theoretical distributions

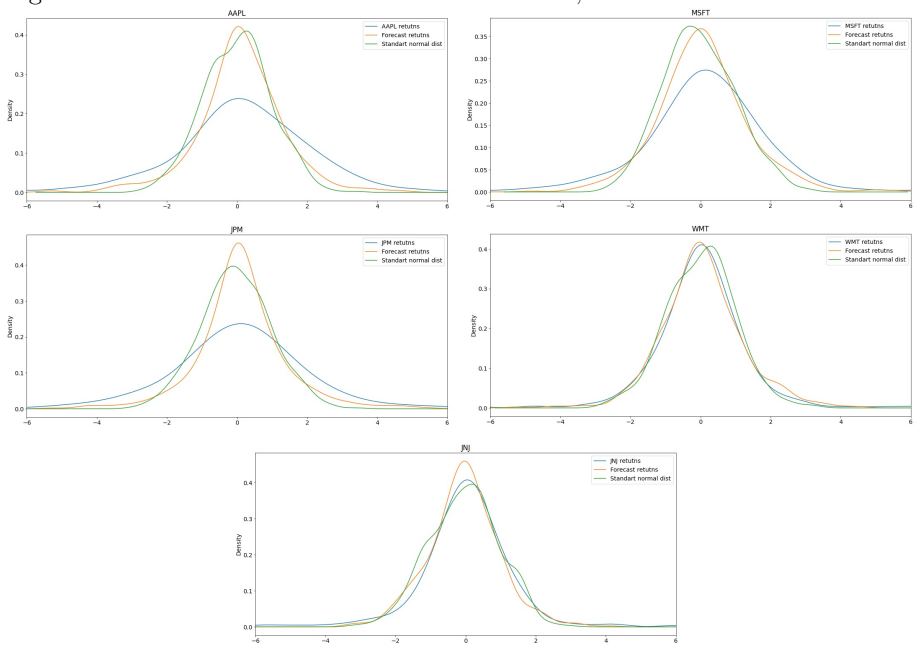


Figure 3.6: Wasserstein distances

	AAPL	MSFT	JPM	WMT	JNJ
mean	0.003531	0.004001	0.004777	0.002254	0.002158
std	0.001652	0.001662	0.002579	0.000819	0.000756
min	0.001081	0.001725	0.001562	0.001169	0.000700
25%	0.002281	0.002841	0.002680	0.001655	0.001629
50%	0.003242	0.003748	0.004575	0.002040	0.001947
75%	0.004576	0.004907	0.006424	0.002700	0.002560
max	0.008982	0.011197	0.017208	0.005443	0.004238

Figure 3.7: AAPL real Market caps VS forecasts

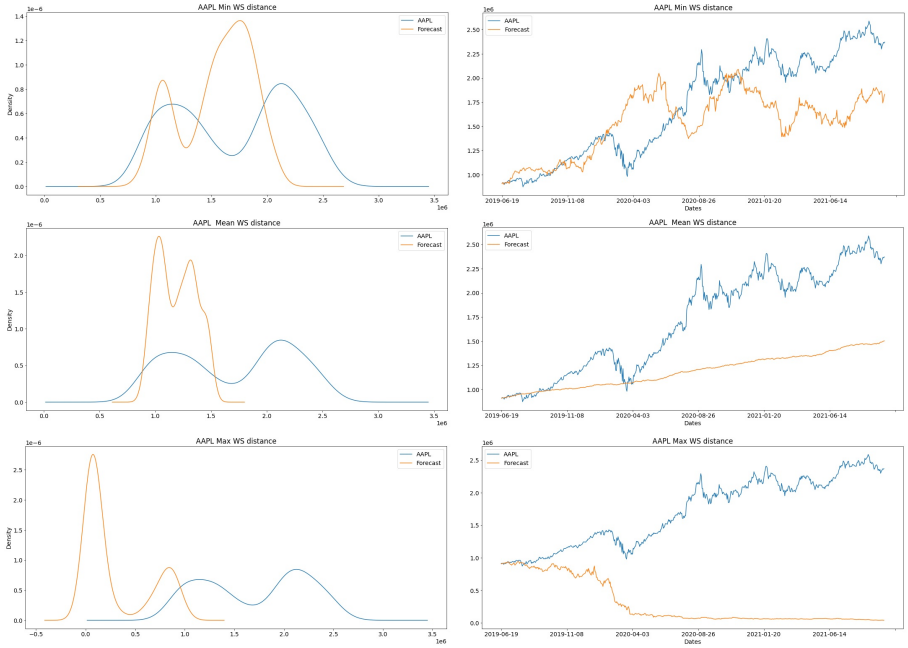
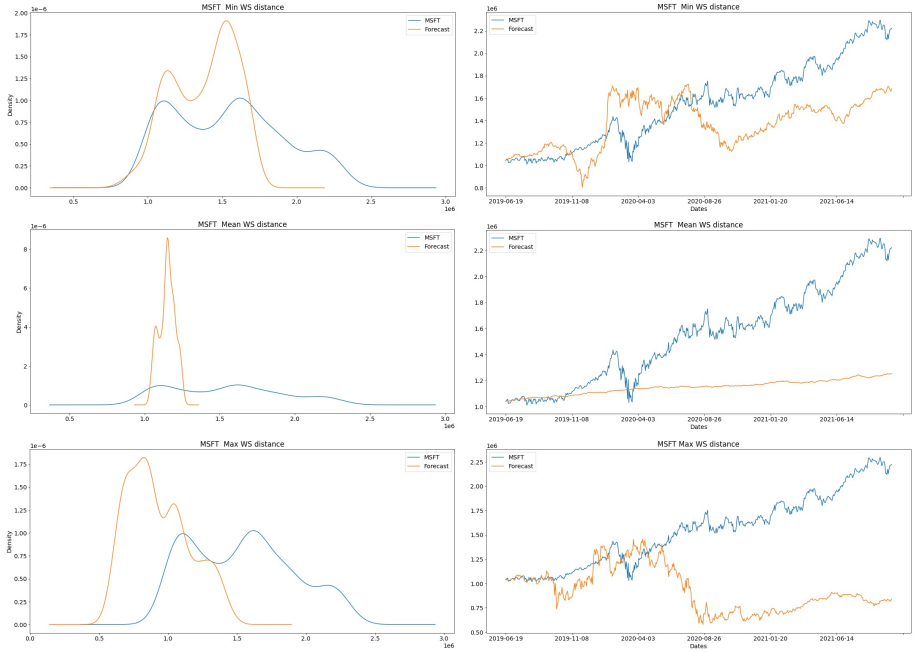


Figure 3.8: MSFT real Market caps VS forecasts



Chapter 4

Variational Auto Encoder (VAE)

4.1 Introduction

Original Variational Auto Encoder VAE is family of Neuro Nets predictive models based on idea of modelling distributions of random variables and sampling from them using two NeuroNets - Encoder and Decoder, trained simultaneously. It was introduced by Diederik P Kingma and Max Welling in 'Auto-Encoding Variational Bayes' [3]. The purpose of Encoder is to recognise the patterns (vector of Means and vector on Standard deviations in data) and the purpose of Decoder is to reconstruct the sample from the approximately closed distribution based on the Encoder proceeds. Here is more complicated NeuroNets structure is used. Such structure was introduced by Manuel Watter at all in 'Embed to Control' [4] article for the reason of modelling transition probabilities and restoring the possible paths for such tasks as Learning swing-up for an inverted pendulum task, or Balancing a cart-pole and controlling a simulated robot arm. This work was improved by Ershad Banijamali et all in 'Robust Locally-Linear Controllable Embedding' [5] for the same pool of tasks. Here in this work I adopted it for the Financial time series modelling.

4.2 Model description

VAE consists of 3 different Neuro Nets:

- $h_{\phi}^{enc}(x_t)$ - encoder part
- $h_{\theta}^{dec}(z_t)$ - decoder part
- $h_{\psi}^{trans}(z_t, u_t)$ - transition part

4.2.1 Encoder and Decoder

Encoder and Decoder: $z_t \in R^{10}$ is hidden state variable which is supposed to be low dimensional representation of x_t

$$z_t \sim Q_\phi(z_t|x_t) = \mathcal{N}(\mu_t, \Sigma_t)$$

where:

$$\mu_t = W_\mu h_\phi^{enc}(x_t) + b_\mu$$
$$\Sigma_t = diag(\sigma_t^2)$$
$$log(\sigma_t^2) = W_\sigma h_\phi^{enc}(x_t) + b_\sigma$$

for encoder-decoder training, we need to generate $\tilde{x}_t$ and $\tilde{x}_{t+1}$:

$$\tilde{x}_t, \tilde{x}_{t+1} = W_p h_\theta^{dec}(z_t) + b_p$$

So, the first part is the construction encoder and decoder and then train them simultaneously.

4.2.2 Transition network

Transition network: The purpose of generative model is to produce $\hat{z}_{t+1}$ based on z_t.

$$\hat{z}_{t+1} \sim \hat{Q}_\psi(\hat{z}_{t+1}|z_t, u_t) = \mathcal{N}(A_t\mu_t + B_t u_t + o_t, C_t)$$

where:

$$C_t = A_t \Sigma_t A_t^T + E_t$$

where E_t is the system noise dispersion matrix which is taken as $I \times \epsilon$:

and $A_T \in R^{10\times 10}$ $B_T \in R^{10\times n_u}$ and $o_t \in R^{10}$. To circumvent estimating of matrix A_t of size 10×10 matrix A_t could be chosen to be a perturbation matrix of identity matrix $A_T = (I + v_t r_t^T)$ which reduces the computational complexity of matrix A_t to 2×10.

$$vec[A_t] = W_A h_\psi^{trans}(z_t) + b_A$$
$$vec[B_t] = W_B h_\psi^{trans}(z_t) + b_B$$
$$o_t = W_o h_\psi^{trans}(z_t) + b_o$$

Now we have z_t as a result of encoder of x_t and z_{t+1} as a result of encoder of x_{t+1}. Here we train h_ψ^{trans}

4.2.3 Loss function

The loss function has a shape of:

$$\mathcal{L}(\mathcal{D}) = \sum_{(x_t, u_t, x_{t+1})\in\mathcal{D}} \mathcal{L}^{bound}(x_t, u_t, x_{t+1}) +$$
$$+\lambda KL(\hat{Q}_\psi(\hat{Z}|\mu_t, u_t)||Q_\phi(Z|x_{t+1})) + H(Q_\phi(\hat{z}_{t+1}|x_{t+1}))$$

where:

$$\mathcal{L}^{bound}(x_t, u_t, x_{t+1}) =$$

$$E_{(z_t \sim Q_\phi)(\hat{z}_{t+1} \sim \hat{Q}_\psi)}[-logP_\Theta(x_t|z_t) - logP_\Theta(x_{t+1}|\hat{z}_{t+1})] + KL(Q_\phi||P(Z))$$

where $\hat{Q}_\psi(\hat{Z}|\mu_t, u_t) = \mathcal{N}(A_t\mu_t + B_t u_t + o_t, C_t)$, $Q_\phi(Z|x_{t+1})) = \mathcal{N}(\mu_t, \Sigma_t)$, and $P_\Theta = \mathcal{N}(0, I)$ are all Gaussian, so $KL(\hat{Q}_\psi(\hat{Z}|\mu_t, u_t)||Q_\phi(Z|x_{t+1}))$ could be represented as:

$$KL[\mathcal{N}(\mu_0, \Sigma_0)||\mathcal{N}(\mu_1, \Sigma_1)] =$$

$$\frac{1}{2}\Big(\sum_i \frac{\sigma_{0,i}^2 + \sigma_{0,i}^2 v_i r_i}{\sigma_{1,i}^2} + \sum_i \sigma_{0,i}^2 r_i^2 +$$

$$+ \sum_i \frac{v_i^2}{\sigma_{1,i}^2} + \sum_i \frac{(\mu_1 - \mu_0)^2}{\sigma_{1,i}^2} - k +$$

$$+2\Big(\sum_i (log\sigma_{1,i}^2 - log\sigma_{0,i}^2) - log(1 + \sum_i v_i r_i)\Big)\Big)$$

where r_i and v_i are taken from $A_t = I + v_t r_t^T$ and μ_0, μ_1 are vector of expected values of left side and right side distributions respectively, also $\sigma_{0,i}^2$ and $\sigma_{1,i}^2$ are i-th diagonal elements of variance matrix of left side and right side distributions respectively.
And

$$KL(Q_\phi(x_t)||P(z_t)) = \log\frac{\sigma_{Q_\phi(x_t)}}{\sigma_{P(z_t)}} +$$

$$+\frac{\sigma_{P(z_t)}^2 + (\mu_{P(z_t)} - \mu_{Q_\phi(x_t)})^2}{2\sigma_{Q_\phi(x_t)}^2} - \frac{1}{2}$$

Also $H(Q_\phi(\hat{z}_{t+1}|x_{t+1})$ is an entropy of the Encoder. It could be written as:

$$H(Q_\phi(\hat{z}_{t+1}|x_{t+1})) = \frac{1}{2}log((2\pi e)_z^n|\Sigma_t|)$$

4.2.4 Network structure

The final structure of NNs used for modelling is:

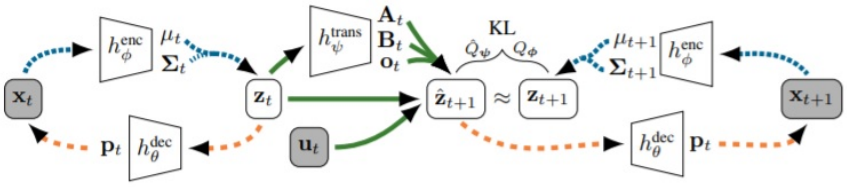
Encoder (Q_ϕ): 3 Fully connected layers alternate with 3 Leaky ReLU layers, plus one Fully connected layer for mu_z representation and one Fully connected layer for $logvar_z$ representation.

Decoder (P_Θ): One Fully connected layer with Leaky ReLU layer, plus one Fully connected layer for A_t representation, one Fully connected layer for B_t and one Fully connected layer for o_t representation.

Transition network (Q_ψ): 3 Fully connected layers alternate with 3 Leaky ReLU layers, plus one Fully connected layer for mu_x representation and one Fully connected layer for $logvar_x$ representation.

fig:nodes

Figure 4.1: Network structure



4.3 Findings

As in previous chapters we train and forecast returns based on Market caps (the same dataset as before) and similarly to Chapter 3 the changes in stock exchanges daily volumes in each Equity is used as the control. The datased was reshaped to 999 observations of matrices 45x30 as current step and 999 observations of matrices 45x30 as next step. The difference between current step and next step is 5 daily observations move forward. So Encoder and Generator based on the matrix 45x30 of 30 equities with 45 daily observations at time $(t : t + 45)$ and same dataset of volume changes for the same period of time produce embedding for Decoder and after that Decoder produces the matrix 45x30 of 30 equities with 45 daily observations for the period $(t + 5 : t + 50)$ and only last 5 time steps are taken to add to the forecast. For generation of predictions the test set not included to the VAE train set was used.

It has been tries several different numbers of epochs to train VAE with intention to find the best performance. Although the Total loss was decreasing all the time, after 300 epochs VAE started to show the signs of overfitting. Thus in this Chapter I present the results of 300 epochs VAE training.

First of all the autocorrelation of forecasted returns is as not prominent as in previous chapters. The same is true for Leverage effect (Figures 4.2, 4.4). However the Volatility clustering of forecasted returns much more prominent here then in previous models results, although not prominent enough compared to real returns (Figure 4.3).

The distributions of real returns and randomly chosen single forecasts for AAPL, MSFT and JPM are quite close, but WMT and JNJ ones look significantly different. Notice that real returns distribution of WMT and JNJ are more picked then Normal distribution and those of AAPL, MSFT and JPM are less picked.

Figure 4.6 shows that the Wasserstein distances for 100 paths are quite similar among themselves and doesn't show the big gap between smallest and largest one. All statistics show quite stable forecasts simulations.

In the Figures 4. and 4.8 we see that best and average Wasserstein distances based models present quite accurate results. There are large overlaps in distribution charts and close performance in paths charts. However the worst models

do not perform well. The additional WMT chart however shows that not all forecasts perform that well. Although WMT forecasts didn't perform well in all previous models predictions (the charts are available in the supplementary materials - Jupyter Notebooks)

Figure 4.2: Autocorrelation of real data vs model forecast

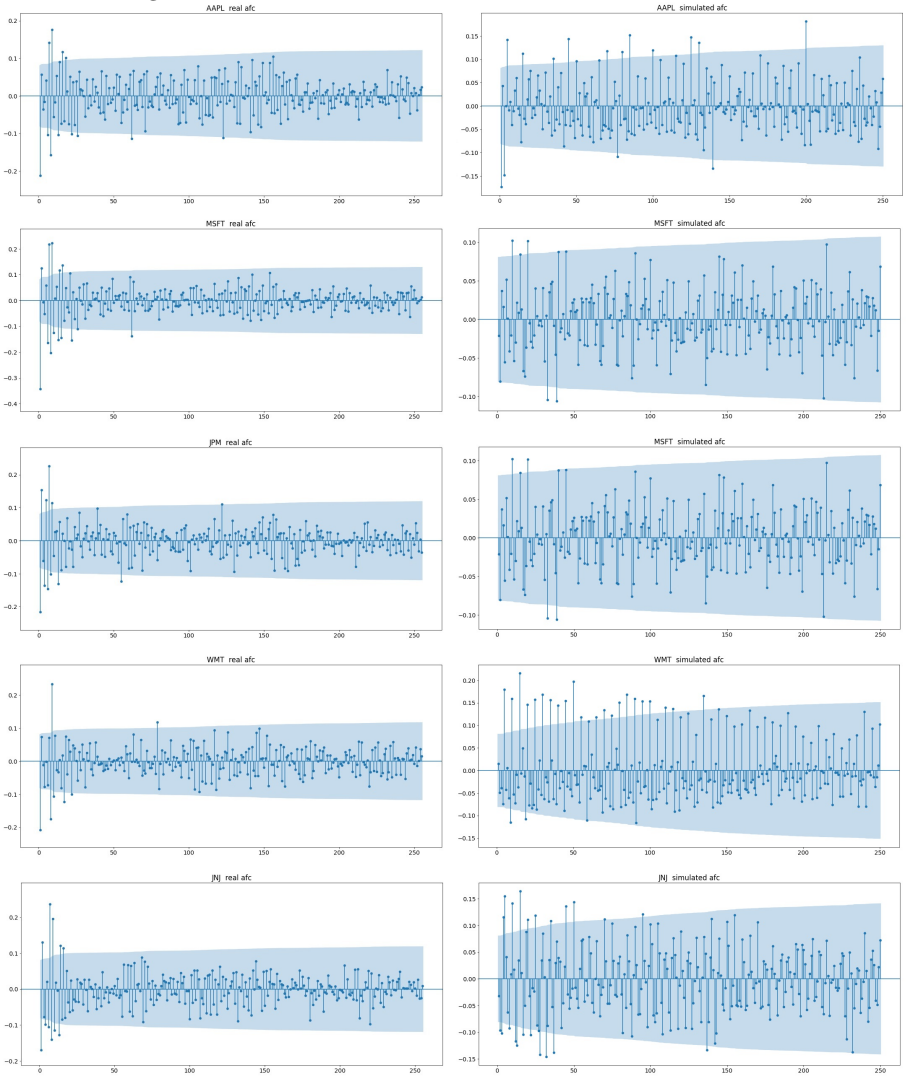


Figure 4.3: Volatility clustering of real data vs model forecast

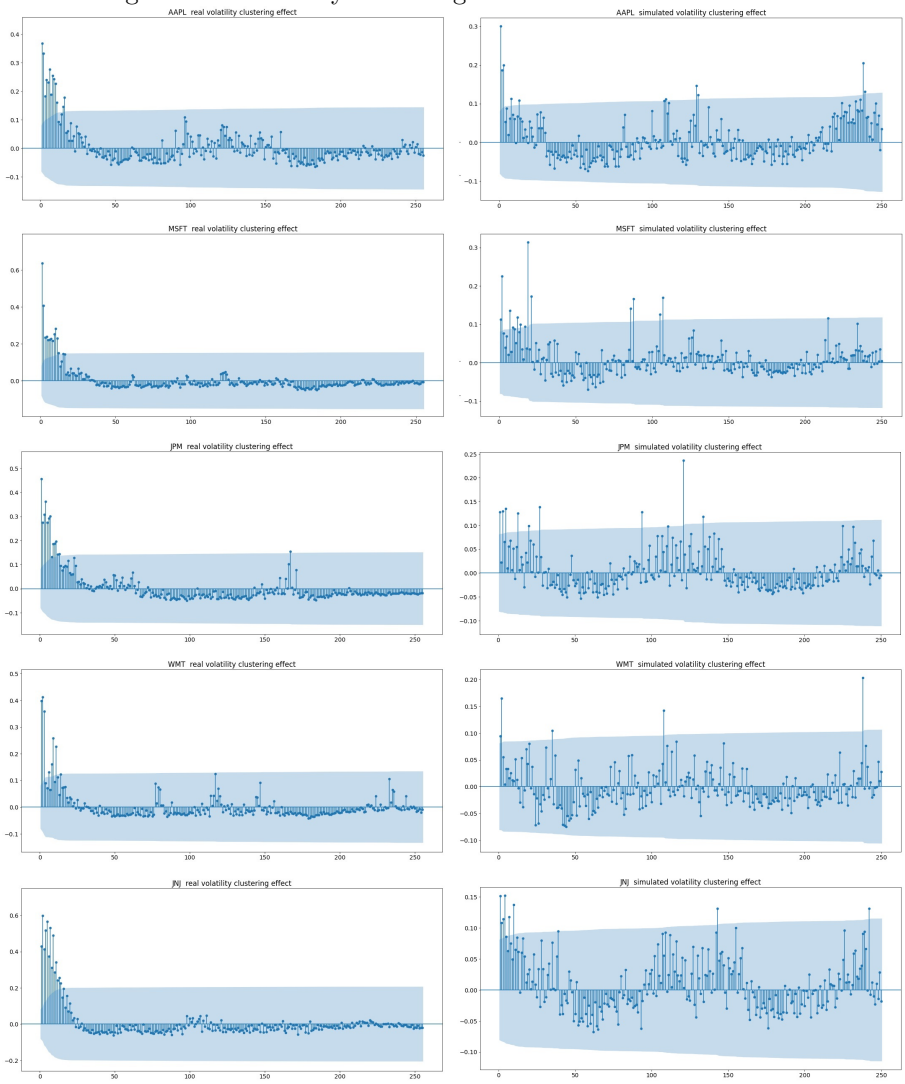


Figure 4.4: Leverage of real data vs model forecast

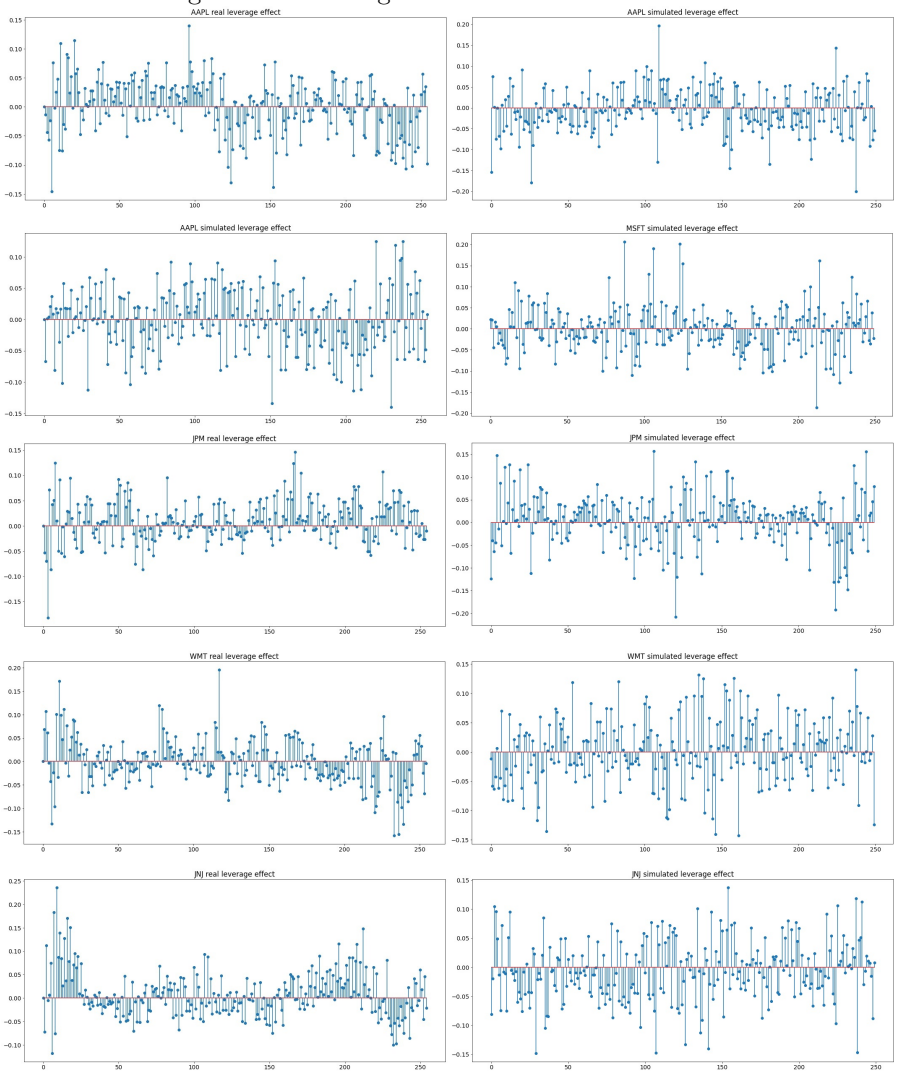


Figure 4.5: Distributions of forecasts vs real data, and theoretical distributions

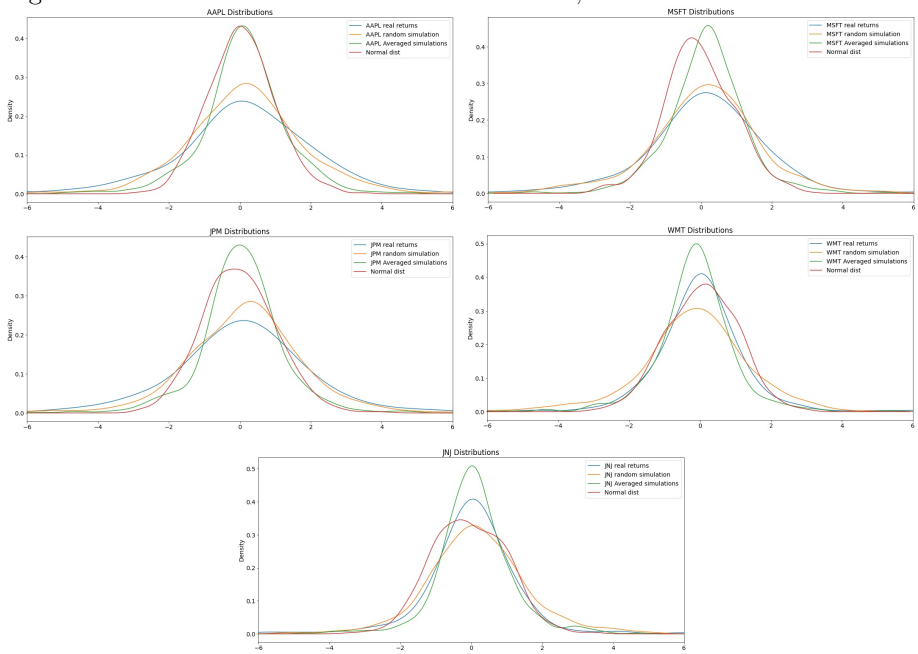


Figure 4.6: Wasserstein distances

	AAPL	MSFT	JPM	WMT	JNJ
mean	0.003630	0.002121	0.003637	0.003053	0.002461
std	0.000391	0.000257	0.000397	0.000360	0.000297
min	0.002556	0.001507	0.002604	0.002257	0.001776
25%	0.003326	0.001981	0.003378	0.002789	0.002306
50%	0.003638	0.002090	0.003610	0.003051	0.002436
75%	0.003854	0.002274	0.003903	0.003254	0.002634
max	0.004643	0.002867	0.004788	0.004167	0.003331

Figure 4.7: AAPL real Market caps VS forecasts

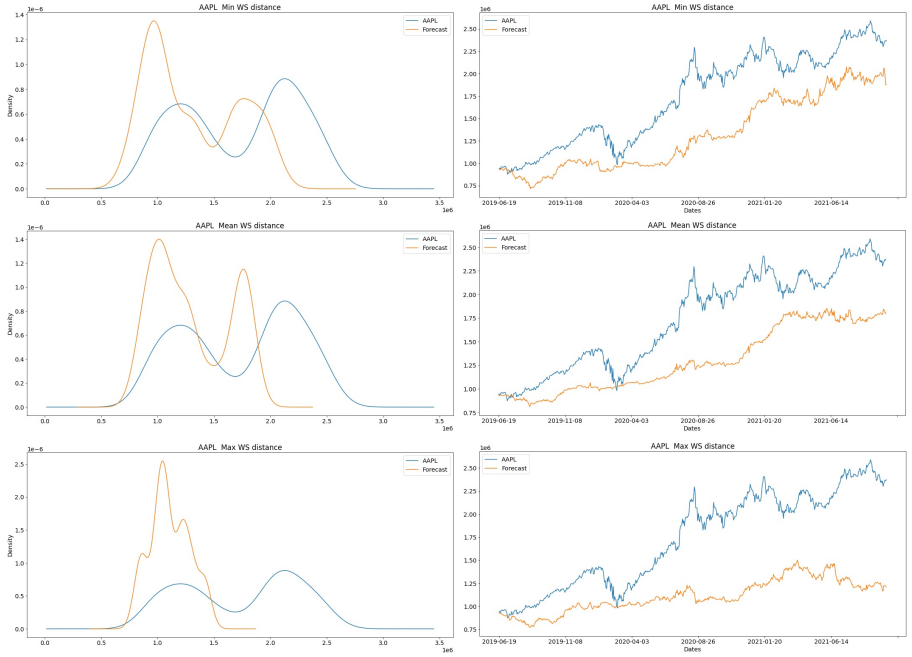


Figure 4.8: MSFT real Market caps VS forecasts

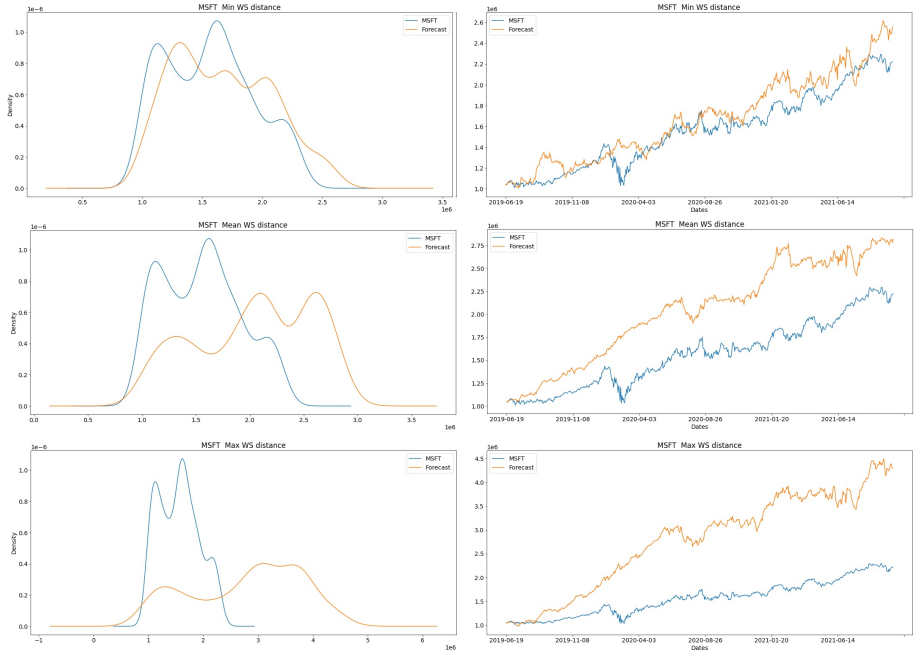
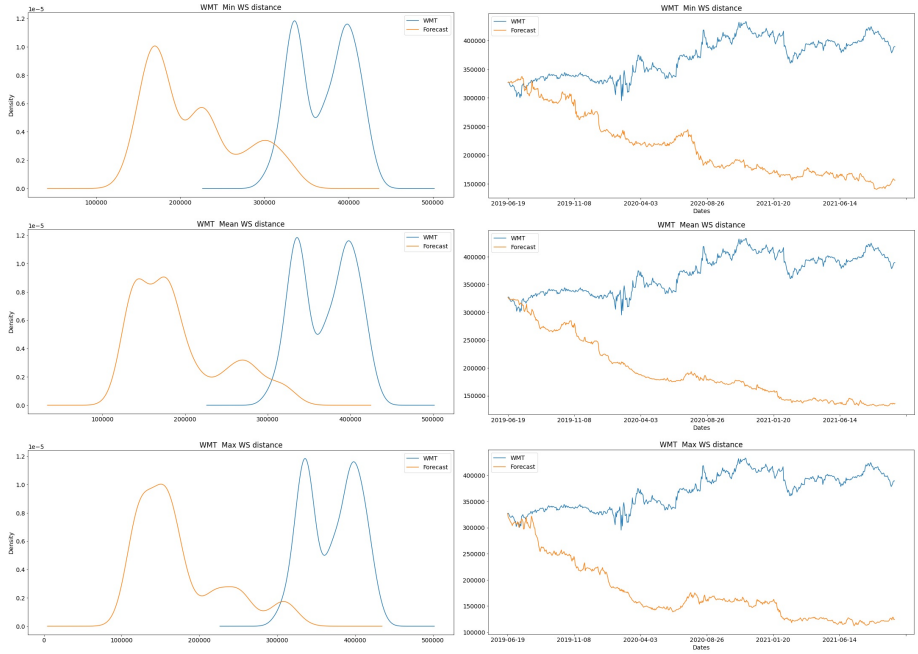


Figure 4.9: WMT real Market caps VS forecasts



Chapter 5

Conclusions and Future Work

5.1 Conclusions

In Figure 5.1 we can see the combined statistics for Wasserstein differences between the real returns test sets and the model forecasts. Remind that this statistics are received based on 100 different paths generations by each model for each Equity presented in the table.

The most important observation here that the worst case results (max distances) of VAE are significantly better then both pure AR models and AR with exogenous variable. The minimal distances paths generated by VAE however are the same or worse. And averaged paths demonstrate mixed results. But the standard deviations of distances also significantly lower for VAE generated paths then for the rest models. This fact together with much better worst case results allows to state that VAE generates much more robust forecasts then AR models.

The results above together with demonstration of VAE capability to catch up the important Volatility clustering effect (Figure 4.3) at least partially where all AR models demonstrated lack of such feature (Figures 3.3 and 2.6) gives the right to treat VAE as the best model among all models covered by this research.

5.2 Future Work

However there are the significant space for VAE performance improvement.

First of all the most warring finding is that there is no efficient measure of Leverage effect so it is not clear how to capture it in real returns and so capture it by any models. Unfortunately all tries to find any other measures except direct correlation measures were failed.

Second is the necessity to achieve more pronounced Volatility clustering effect.

Third is that these days there already are lot of different Neuro Nets structures built for the same purpose. There are well known TCN models based on

convolution layers architecture, there are models based on LSTM layers. Also there are different model optimisation techniques as Generative Adversarial Networks. Also AR models presented not only by AR-GARCH or AR-EGARCH. It is really not possible to observe all existing models in one work but it worth to compare as many favorites as possible.

Finally, the Time series prediction models itself have not value themselves. Their real benefits can be valued only as a parts of Financial assets valuation models and Portfolio management frameworks. However it is the most important and value creative part of asset valuations and PM as much of the rest parts are strait forward.

.

Figure 5.1: Wasserstein distances

Wasserstein distances statistics

Pure AR models

	AAPL	MSFT	JPM	WMT	JNJ
mean	0.004198	0.004003	0.004658	0.002193	0.002255
std	0.002375	0.001540	0.002667	0.000849	0.000674
min	0.001256	0.001461	0.001677	0.001080	0.000972
max	0.018667	0.007953	0.020005	0.006292	0.003962

AR models with exogenous variables

	AAPL	MSFT	JPM	WMT	JNJ
mean	0.003531	0.004001	0.004777	0.002254	0.002158
std	0.001652	0.001662	0.002579	0.000819	0.000756
min	0.001081	0.001725	0.001562	0.001169	0.000700
max	0.008982	0.011197	0.017208	0.005443	0.004238

Variational Auto Encoder

	AAPL	MSFT	JPM	WMT	JNJ
mean	0.003630	0.002121	0.003637	0.003053	0.002461
std	0.000391	0.000257	0.000397	0.000360	0.000297
min	0.002556	0.001507	0.002604	0.002257	0.001776
max	0.004643	0.002867	0.004788	0.004167	0.003331

Bibliography

[1] Bollerslev T., "Generalized Autoregressive Conditional Heteroskedasticity", Journal of Econometrics 31 North-Holland 21, pages 301-327, 1986.

[2] Kevin Sheppard, arch Documentation, `https://readthedocs.org/projects/arch/downloads/pdf/latest/`, 2021.

[3] D. Kingma and M. Welling. Auto-encoding variational Bayes. `https://arxiv.org/abs/1312.6114`, 2014.

[4] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. `https://arxiv.org/abs/1506.07365`, 2015.

[5] Ershad Banijamali, Rui Shu, Mohammad Ghavamzadeh, Hung Bui, Ali Ghodsi. Robust Locally-Linear Controllable Embedding `https://arxiv.org/abs/1710.05373`, 2018.