UCLouvain

epl

École polytechnique de Louvain

Generating Data for Financial Portfolio Optimization

Author: **Harold Corlùy**
Supervisors: **Siegfried Nijssen, Gianmarco Aversano**
Readers: **Siegfried Nijssen, Gianmarco Aversano, Marco Saerens**
Academic year 2021–2022
Master [120] in Data Science: Engineering

Abstract

This Thesis investigates on the benefits and inconveniences of using data generative models for financial portfolio optimization (FPO). In a first step, we explain the state of the art generative approaches. Then, we focus on a novel approach, until now unemployed to FPO, which is to use generative models that combine time series forecasting models with normalizing flows. We focus on two specific financial strategies, a strategy that maximizes the Sharpe Ratio, and a strategy that minimizes the volatility of a selected portfolio and make comparisons between the multiple approaches and the different models that we choose.
Finally we draw conclusions and try to give an unbiased opinion on the results gathered throughout this thesis.

Acknowledgment

I want to thank all the people that helped me during my studies and this last year.

I would like to give a huge thanks to my two supervisors. Professor Siegfried Nijssen, thank you for your availability and your good advice during the meetings. Gianmarco Aversano, thank you also very much for all the help you provided during this thesis, the questions you answered very quickly (and sometimes way past work hours) and your overall enthusiasm towards this big project.

A big thank you also to Marco Saerens that accepted to be a part of the jury and to the entire team at Euranova that came with this thesis idea and that already provided some inside for the baseline.

Finally I would like to thank my parents that always supported me during my studies and my study sessions and which provided the healthy environment that helped me grow and study in the best of ways.

Contents

Chapter 1

Introduction

Most investors want one thing: to invest in the "best" assets that will give them the highest returns with little to no risks. The ability to find a good set of financial assets, better known as portfolio, that satisfies those requirements is essential for investors.

In 1952, Markovitz introduced Modern Portfolio Theory (MPT) and won a Nobel prize for his work [1]. The purpose of MPT is to maximize the returns while minimizing the volatility or risk of the portfolio.

This theory is mostly based on diversification. Because investments are often high risk - high return or low risk - low return, if an investor chooses a healthy mix between those two types of assets they can overcome the risks and attain decent returns. Correlation is also an important factor in MPT. If assets are chosen such that they are not correlated, then the consequences of one asset crashing are dampened by the fact that the other will not necessarily crash (as they are not correlated).

The second principle upon which MPT relies is the Efficient Frontier, the expression for the Efficient Frontier focuses on the covariance matrix. As explained above, MPT compares the mean return of a portfolio to the risk or variance of that same portfolio. The efficient frontier represents all the best portfolio returns in function of the risk. The expression for the efficient frontier is dependent on the covariance matrix of the daily returns.

In the financial community there exists what we call strategies, they guide the way assets are chosen in a portfolio. Strategies can vary a lot, some of them can, for example, choose to completely minimize volatility. Having the most diversified portfolio will probably not give the highest returns but it will stay consistent and vary very little. Another might be to maximize the returns or to find a good compromise between the two. A good middle ground is described by W. Sharpe as the Sharpe Ratio [2].

Regardless of the strategy, the most important aspect of FPO is to gather a

lot of data in order to be prepared for most of the scenarios that might occur in the future. The problem is that the data is very scarce, even more when you take companies into account that have just recently been created.

Historically, portfolio managers have very simply used the real historical data to estimate the covariance matrix between assets. The covariance matrix is an essential part because it reflects the relationship between the different assets in a portfolio. Nowadays, with the surge of machine learning (ML), there is a growing interest into using ML to generate more data to help FPO. This data generation process could also help to improve the protection against sudden fluctuations in the market. Historical data is only one possible realization of the past, one sample of an unknown distribution. Thus, being able to generate more (past) possible scenarios could lead to an improvement in the estimation of the covariance matrix between the financial assets. Another relevant factor is that financial managers may tend to include observations that are too far into the past so that they have enough data to perform sample covariance estimation. However, at the same time data that are too old may be detrimental for this process because the covariance structure of a chosen set of assets is not static over time: it may indeed change and be best represented by the most recent observations.

Many data generation algorithms already exist. The objective of the present Thesis is to research and analyze if they can be useful for FPO. Generative models such as Generative Adversarial Networks (GANs) [3], Variational Autoencoders (VAEs) [4], Normalizing Flows (NFs) [5] and Copulas [6] learn the joint probability distribution from available data and therefore allow for data generation by sampling from the learnt distribution. By training a generative model on past financial data, one could generate a possibly infinite number of new samples which would aid the process of sample covariance estimation. Unlike GANs and VAEs, NFs are not only able to generate new data, but also provide the means for density estimation and are thus a more interesting choice.

The machine learning community has repeatedly shown how GANs can produce realistic synthetic samples based on a real-life dataset. Image generation has never been more effective today thanks to GANs. GANs are based on a generator-discriminator architecture, they train and improve together. VAEs on the other hand have an encoder and decoder structure, where the encoder estimates the mean and standard deviation from its input. Simply put, it learns a mapping from an unknown distribution to a simple one. Samples that are very similar in the latent space will produce similar results as output, whereas for GANs this is not the case. VAEs are also a handy way to generate synthetic data but when looking at image generation, we can see that the generated images are always blurry due to a small noise added before the decoder. NFs follow a framework where it estimates

the joint probability distribution of the asset returns. NFs have proven to be more effective than GANs when dealing with tabular data, while under-performing when dealing with images. S. Bond-Taylor et al. resume really nicely the strength and weakness of these three approaches [7]. Basically, they say that GANs and VAE are very convenient in general and that these approaches each have their own strengths and that the gap between them is very narrow. GANs have a lot of different types of implementation in function of the task they have to perform. Wasserstein GAN [8] uses a Wasserstein loss and a gradient penalty to solve issues with a classical GAN approach. This approach is not flawless and it is well explained in "Flows Succeed Where GANs Fail: Lessons from Low-Dimensional Data" [9]. T. Liu and J. Regier explain in their paper that GANs are well suited for image generation, but for statistical modeling approaches it is often not the best choice. They also show how NF even outperforms WGAN in their own Wasserstein metric.

Finally, Copulas are a way to decompose the joint probability into their marginals. Copulas try to map different inputs to known distributions and then it will link these distributions from the different inputs in order to form the joint distribution. Once we have that joint distribution, we can sample data and find the covariance matrix. Copulas are not flawless and some of the problems with generating financial data is explained in a paper from S. Watts [10]. Parameterizing the Copula functions and defining the correlation function is not always optimal and can in some cases pose problems.

There are many parameters involved when seeking the best portfolio following a particular strategy. One of them is the choice for the covariance matrix. Because we don't know the true distribution of the data, we have to estimate the covariance matrix. Sample Covariance is a way to estimate the covariance matrix based on the mean and the data samples.
It is also possible to define another estimator for the covariance matrix, the shrinkage covariance. In "Honey, I Shrunk the Sample Covariance Matrix." [11] O. Ledoit and M. Wolf explain that by using a sample covariance, we are subject to a lot of estimation errors. They propose a technique that transforms the sample covariance estimation, this is done by bringing the most extreme coefficients towards more central values.

Even more improvements have been proposed since then for the shrinkage covariance estimation, an example of this is the OAS covariance estimation proposed in the paper by Yilun Chen *et al.* [12]. Thus, the choice for the estimation of the covariance is something we also have to take into account.

The data we are going to use in this Thesis is a batch of historical assets from assets taken from the Willshire 5000 list from August 2021. A more in depth explanation

about the dataset and its utilization will be given later on (section 2.1.1).

The objective of this Master Thesis is to report and analyze the performances from various data generation techniques to augment financial data for financial portfolio optimization (FPO). As mentioned above, the available data represent only one possible realization of the past and thus a more accurate covariance matrix could be estimated if the actual joint probability distribution of the time series was available or could somehow be estimated.

As mentioned, FPO relies on the covariance structure of the data. This structure is inferred using (augmented or not) past data for a certain set of financial assets. However, the covariance structure of the data that are too far away into the past may be different from the one observed in the present or that will be observed in the future. Even if this covariance structure is estimated correctly, it may not be helpful for FPO to, say, increase returns in the future. To solve this problem, the present Master Thesis proposes the combination of NF with time series forecasting, as done in [13], to generate possible future data from which to estimate the sample covariance matrix FPO depends on. To the authors' knowledge, this is the first time that such an approach is used for financial data augmentation. The present Thesis will thus compare the results obtained from FPO when using the sample covariance matrix estimated from (1) the true data, (2) data generated from NF or other generative models, (3) data generated using the approach of [13] or other time-series forecasting models for future data generation.

The comparison will then rely on multiple settings. We are going to look at the influence of the the prediction period, to find if long term prediction have an effect on the performances of the models. Another analysis is the influence of the number of assets we put into an initial portfolio. Finally the most interesting analysis is the influence of the time period for the evaluation. We want to know if the models are overall good at augmenting financial portfolios over long periods of time. This could be useful because if the model don't show good results at any given time, this means they are not reliable. The whole analysis is always done looking at the two aforementioned financial strategies and only show the potential of the generative models. They should not be considered exhaustive for all the use cases one could encounter.

Chapter 2

Background

The main objective of this Thesis is to find the best model that will generate synthetic data to augment our initial dataset consisting of financial returns for multiple assets. Good synthetic data is needed for the purpose of finding the best financial portfolio. But before we explain all the models we are considering and the results we get, we first have to give an explanation of the financial terms we are using. We also need to define what and why we are considering different covariance matrices as explained in the introduction.

2.1 Financial Background

In this section I will describe the main concepts behind investing strategies and finance theory that we will be using for this work. An explanation of the dataset used throughout the work will also be given.

2.1.1 Dataset

A financial dataset is a collection of historical data for a set of assets. For example, we may have daily returns (shown in percentage) for N assets over M days, thus a $M \times N$ matrix. The problems with financial datasets is that the data is very scarce. An ideal dataset would contain a lot of data in order to evaluate and allocate our assets in the best possible way.
Looking at financial portfolios we see that there are a lot of assets with different allocation weights. The way we choose those weights is linked to a particular financial strategy. The common theme behind the way we compute the different strategies is that if we have more data, then we have more information in order to choose how to allocate (over the assets) a total budget we want to invest. The percentage of the total budget allocated for a specific asset is usually referred to

as weight. To give an intuition of a typical dataset for a portfolio we are going to analyze this simplified table.

Index	Asset 1	Asset 2	Asset 3
1	0.051	-0.062	0.01
2	0.035	0.028	0.12
3	0.064	-0.001	-0.087
4	-0.015	-0.047	0.074

Table 2.1: Example of a dataset representing a portfolio

In this simplified example we can see a portfolio of 3 assets where every row corresponds to a return percentage between two consecutive (market) days. We can see for example that asset 2 is in a period where its stock is going down while asset 1 and 3 is more trending upwards.

An important aspect about financial datasets is that the order of the rows does not matter as this table will be used to estimate a covariance matrix and a mean of the rows. The second metric we need for the asset allocation problem in some financial strategies (Maximize Sharpe Ratio 2.1.3) is the mean return on investment for all the assets in our dataset/portfolio. This also implies that we don't need an ordering between the rows.

Finally, the purpose of data augmentation techniques that we are going to explore, is to generate synthetic data that needs to be added to the initial dataset in order to have more data and thus, we expect also better results when optimizing for a particular financial strategy.

In this thesis, the datasets represent the data of all the assets of one portfolio before a weight has been associated to each asset.

Used datasets

In this thesis we are making computations based on assets found in the Wilshire 5000 list from August 2021 [1]. We chose this list because it contains a lot of assets from diverse industries.

The historic data from those assets are gathered from the Yahoo finance site [2]. The pipeline to collect all of this has been automated and can be regularly updated in case we want the most recent data for those assets.

[1]The complete list is available at `https://github.com/derekbanas/Python4Finance/blob/main/Wilshire-5000-Stocks.csv`

[2]Website available at : `https://finance.yahoo.com/`

A lot of successful companies are very recent. The technology industry plays a big role in this phenomenon. Because of this, we don't have a lot of data to train our models. This is also a part of the problem we try to tackle in this thesis. If we can interpolate/extrapolate very precisely, this can help models improve their training performance and coincidentally future predictions can possibly be improved.
For the reason mentioned above, we are considering data spanning from 03-01-2015 until 10-11-2021.

During the test, we are trying to simulate real life scenarios in order to show how a financial portfolio could evolve using different strategies and different models. Most portfolios are not very big. For this reason, we are sampling randomly 35 to 65 assets (depending on the test we are performing) from the Wilshire 5000 index and we are putting them together. This does not mean that the final portfolio contains 35 to 65 assets, this means it will find the combination of assets that improve the most the strategy we are trying to achieve and it will also put weights to the new combination in order for some assets to have a bigger impact overall.

Summary :

- 3036 assets from the Wilshire 5000 index
- Historic data gathered from Yahoo finance
- Dataset size varying between 35 to 65 assets
- Data starting from 03-01-2015 until 10-11-2021

2.1.2 Efficient Frontier

The first aspect I am going to discuss is the Efficient Frontier or EF. EF is a means to compare different weighted portfolios (portfolios where weights have been assigned to each of its assets) according to their efficiency. Here, efficiency is defined as the returns we get with a portfolio with respect to the volatility of that portfolio. The optimal portfolio is defined as the one that will output the highest expected return given a level of risk.

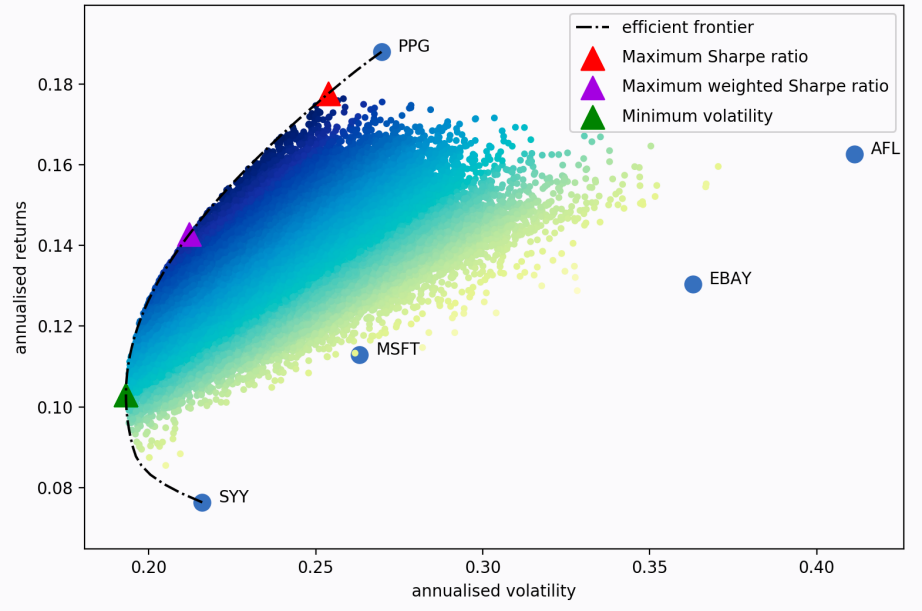

Figure 2.1: Example of how the efficient frontier looks like. (Source : [14])

Figure 2.1 is a depiction of an efficient frontier with different strategies that I will define in the following section.
Every dot on the image represents a portfolio while the hue shows how a portfolio is performing. If the color is very dark, this means the chosen portfolio is closer to the efficient frontier, which on its turn means there are very few portfolios that outperform the chosen one for a given level of volatility/risk.
In other words, the efficient frontier is represented by the upper edge on the figure 2.1 and constitutes an upper limit where the return on investment is maximal given a level of risk. Points underneath the curve show sub-optimal portfolios because we can find a portfolio with higher returns given the same risk level.

The easiest way of obtaining the efficient frontier for a dataset of assets is to take the weights that we allocate to the assets as variables and try a lot of combinations. We then only keep the combinations with the highest return on investment per risk level and plot the results where each point corresponds to a portfolio (with weights allocated to each asset of the portfolio). We then draw a line between all these "optimal" portfolios and this gives us the efficient frontier. Naturally this only works when we try a lot of different combinations.

2.1.3 MPT Strategies

Harry Markovitz wrote Modern Portfolio Theory or MPT in a dissertation named "Portfolio Selection" [1]. The purpose of the dissertation is to describe a number of strategies that can define how an investor should invest in order to optimize a specific financial metric.
In this thesis we are going to focus on two main financial strategies. The first one being the Minimum Volatility strategy and the second one being the Maximimum Sharpe ratio strategy.

In this section we will use the following notations :

- w : a vector of portfolio weights where $\sum_i w_i = 1$
- Σ : the covariance matrix of all the asset data
- μ : a vector of mean return on investment(ROI) for the assets over a year (annualised returns of assets)
- $w^T \Sigma w$: the variance of the portfolio
- $w^T \mu$: the expected return of the portfolio

Minimum Variance

As its names implies, the minimum variance strategy will minimize the volatility of the portfolio while keeping a reasonable return on investment. On the figure above 2.1, we can see that as the leftmost point.
One way of minimizing the volatility is by taking a diverse portfolio with non correlated assets. If one of them plunges this does not have a fatal consequence on the portfolio returns.
If we know from 2.1.3 the formula for the variance and we know that the formula for the volatility is

$$\sqrt{w^T \Sigma w} \tag{2.1}$$

Then the minimum variance strategy finds the weights that it is assigning to a portfolio by optimizing this equation :

$$w^* = argmin_w \quad \frac{1}{2} w^T \Sigma w \tag{2.2}$$

Maximize Sharpe ratio

The Sharpe ratio [2] is a notion W. Sharpe introduced in order to have a metric that defines a good balance between return on investment and volatility or risk taken.
The formula for the Sharpe ratio is the following :

$$SR(x) = \frac{r_x - R_f}{\sigma_x} \tag{2.3}$$

with

- x the assets(portfolio) we want to invest in
- r_x the expected/average rate of return of x
- R_f the return of a risk free rate, the rate of return with zero risk (by default it is set to 2% a year)
- σ_x the standard deviation of r_x

We can see the Sharpe ratio as a measure of the expected excess return per unit of risk.
We can also rewrite the formula using the notation mentioned above 2.1.3 as

$$SR = \frac{w^T \mu - R_f}{\sqrt{w^T \Sigma w}} \tag{2.4}$$

The goal of this strategy is then to find the portfolio(best asset allocations) that result in the highest Sharpe ratio.

2.2 Covariance Matrix

The covariance matrix is a measure of similarity between the different assets in a dataset. If we consider that assets all follow their own distribution, as complex as they may be, the covariance matrix will measure how these distributions relate to each other when they fluctuate.

It is a measure of variability between two random variables. As mentioned in the introduction, the choice for the covariance matrix can have an impact on the performances of the models we are evaluating.

In the previous section 2.1 we saw that the covariance matrix plays a huge role in the definition of the different financial strategies. We now understand why changing the choice for the computation of the covariance matrix can affect the results for the specific optimization factor.

2.2.1 Sample Covariance

The formula for the sample covariance matrix where we estimate the covariance between variable X and variable Y is the following:

$$cov_{xy} = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \hat{X})(Y_i - \hat{Y}) \tag{2.5}$$

With

- n : the number of observations of each asset
- $\hat{X}, \hat{Y}$ are the sample means associated to their random variable
- we use $n-1$ because we use the sample mean

The sample mean and sample covariance matrix are unbiased estimators. The disadvantages of using this covariance matrix is that it is subject to outliers which can result into significant changes when analyzing the outcomes of the financial strategies.
For this reason we are going to look at other methods for evaluating the covariance matrix such as the shrinkage covariance.

2.2.2 Shrinkage Covariance

In the paper "Honey, I Shrunk the Sample Covariance Matrix" [11] Ledoit and Wolf explain how it is unthinkable to use the sample covariance matrix for portfolio optimization. There are too much estimation errors that can lead to incosistent mean-variance results. Ledoit and Wolf propose that the sample covariance matrix goes through a series of transformations in what they call *shrinkage*.
In the sample covariance matrix, when an estimated coefficient is extremely high and the estimation isn't correct, it goes to pair with a high estimation error. In other words the estimation of the coefficient isn't reliable, this can cause major discrepancies between the true covariance matrix and the estimated one which can lead to faulty results when using this estimated matrix for other purposes. The same can occur for extremely low coefficients resulting into large negative errors. What shrinkage is doing is that it pulls down these high coefficients and compensates for the negative error. That way, incorrect estimation won't have a big impact on the final result. In the paper they call this "the shrinkage of extremes towards the center". Shrinkage is not something new and the main innovation the paper brings, is how the implementation shrinks some coefficients and decides which ones to shrink. More information about this last part can be found in the appendix of their paper [11].

In 2009, Yilun Chen *et al.* proposed a new type of shrinkage covariance matrix that would improve the estimation for samples that are Gaussian. Their approach is to condition the covariance estimation from the Ledoit-Wolf method. The second part of the process is to further reduce the estimation error by introducing an iterative shrinkage estimator. The name of that new matrix is called OAS [12].
In the figure below 2.2, we analyze the evolution of the Squared error between the real covariance matrix and the estimated one. The shrinkage we see depends on the method we are choosing. Their purpose here, is to show that OAS doesn't need a lot of samples to find its optimal shrinkage coefficient, which in its turn means the estimation error cannot be improved in a considerable manner.

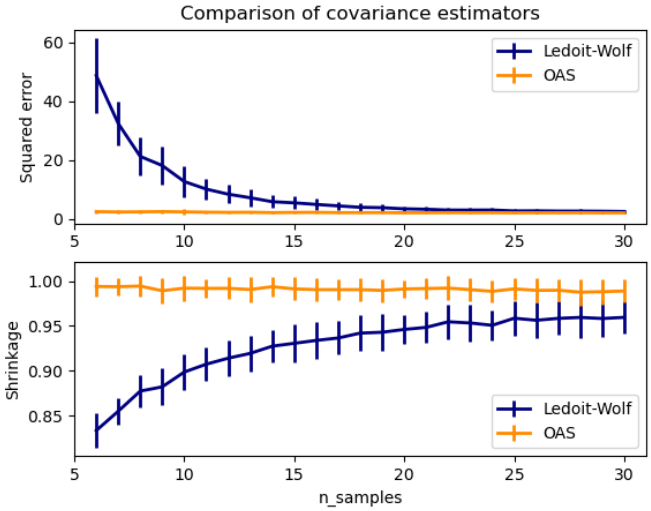

Figure 2.2: Comparison between OAS and Ledoit-Wolf for Gaussian data (Source : [15])

In Figure 2.2, the differences between Ledoit-Wolf and the OAS approach are shown using Gaussian distributed data. This clearly shows an improvement in convergence for the estimation with few samples.
The main bottleneck for our use case is that the data has to be Gaussian. It could therefore show worse results overall compared to the Ledoit-Wolf approach.
For the implementation of those covariance matrices we are using the covariance package from sklearn [3] .

[3]The documentation for this package can be found at : `https://scikit-learn.org/stable/modules/covariance.html`

Chapter 3

Problem Statement

The main problem we are trying to solve in this thesis is to achieve improvements in FPO by generating new data and by augmenting the initial dataset.

The problem can be solved in multiple ways. The existing approaches (section 4.1) rely on the estimation of the past joint probability distribution in order to find dependencies between assets found in a dataset. In this thesis we propose a new approach that has, to our knowledge, never been applied in this field. We will not estimate the past joint distribution, because we want to estimate the future joint probability distribution. By using a time series forecasting model, first described in the paper by the Zalando Research team [13], that combines normalizing flows with forecasting, we are able to generate new synthetic data that could be used when optimizing financial strategies.

The idea behind this approach can be justified by the fact that the dependencies between the assets may vary considerably through time. By using time series forecasting models, we are able to ignore old dependencies and focus on more recent (or forecast) ones. If we look at the Max Sharpe strategy (section 2.1.3), we see two important components when trying the find the best portfolio optimizing the Sharpe ratio. The first one being the mean ROI and the second being the volatility of a given portfolio (covariance matrix). Our intuition is that the normalizing flow part of the model will improve the overall estimation of dependencies between the assets in our tabular dataset (portfolio) and the forecasting part of the model will improve the estimation of future data points.

These models were created in order to improve time series predictions or forecasts. Our approach doesn't focus on predicting the future, instead we use time series forecasting models in order to generate possible future data. From the generated data, we compute the covariance matrix and mean return of the assets which we will use to improve the performance of a financial optimization problem.

This approach is novel in that it attempts to account for future correlations instead of relying on past ones.

Naturally we want to make a clear comparison between different approaches for the augmentation of financial datasets and therefore we have to define clear model selection criteria and a comparison framework.
The different criteria associated to the model comparisons are:

- **Multivariate**: This is important as it reflects if an asset in a portfolio is influenced by the other assets for the data generation process.
- **Training efficiency**: A criterion that cannot be neglected is the training of the models. If the training is computationally too expensive, this could be a huge drawback for its selection.
- **Selective training context**: The training context is the part of the training dataset a model chooses to generate new data based on past dependencies. If the model can choose the interesting training samples that yield the most information qua dependency, it could be beneficial for the estimation of future dependencies. We distinguish models that use the entirety of the training data as context for their joint estimation, from models that only use some parts of the training data. These last models are selective.
- **Forecasting quality**: This criterion is essential when, for instance, we want to optimize according to the Max Sharpe strategy, because we need the mean ROI of the assets in a portfolio. With forecasting, this means that the models will try to estimate the future evolution of the stocks instead of relying on old evolution that could also be outdated.
- **Density reproduction**: Some of the evaluated models are specialized in the generation of samples that will look as realistic as possible. These models will not, by design, try to be close to the distribution of the input data, but instead they will generate the most probable outcome every time. This could have an impact on the estimation of the covariance matrix and thus on the results attached to a particular financial strategy.

We did not consider the quality of the data because it is not essential in our context. For images, the quality is a major criterion but for financial data, the most important aspect of the augmented datasets is the ability to find the right dependencies between the different assets from the same dataset.

Chapter 4

Models and Contributions

In this section we are going to analyze which model to consider and why. For our data generation process we distinguish two different types of models. The first class is the generative models. The second group of models is the time series forecasting models class. This chapter is therefore dedicated to giving a broad overview of the models used in this master thesis in order to understand how they can and will be used to generate data to enrich a financial database of assets returns.

4.1 Generative Models

There are a lot of use cases where we can use generative models to augment the data we have at our disposal. This is often to have a broader view on what we can or could find in our dataset. If we can estimate how the data we have behaves, in other words if we can estimate the distribution of our dataset, then it is easy to sample new examples that were unseen. With the distribution that is estimated we can also easily compute the covariance matrix. In our case, this estimation of the covariance matrix is essential and it was made apparent when we talked about the financial strategies 2.1.3.
We will see that some models even allow us to learn a mapping from a known distribution(and chosen by ourselves) to an unknown distribution (the true distribution of the data). By doing so, it will learn how likely certain samples of data are. An example is as likely as the mapping of the amount of times it can be found in the initial dataset and compared to the occurrences of other examples in the dataset.

4.1.1 Generative Adversarial Network (GAN)

Introduced by Good Fellow *et al.*. in 2014 [3], GANs are now widespread for generating realistic data. By combining a "Discriminator" with a "Generator" the

model is trained to create synthetic data that resembles real data.

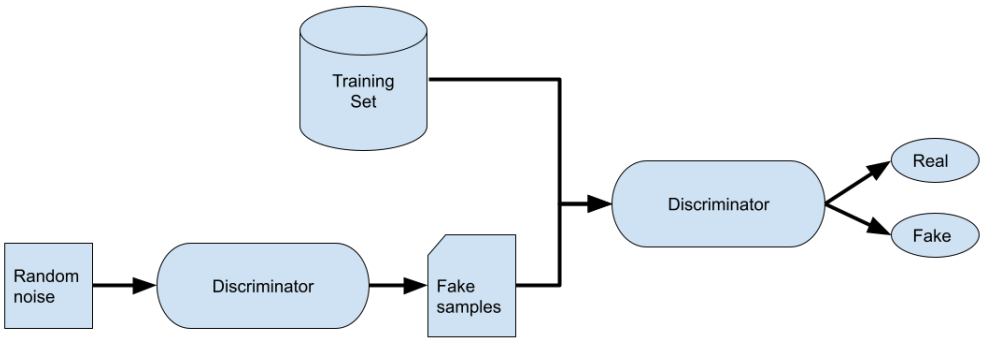

Figure 4.1: Global Architecture of a GANs

In Figure 4.1 we can see how the architecture for GANs is organized. The role of the “Generator” is to generate new unseen data based on noise given as input. The role of the “Discriminator” on the other hand is to classify real data from synthetic data. The idea behind this network is that there is a sort of competition between the Generator and the Discriminator. The Discriminator must be very good at distinguishing between real and fake samples from its input. The Generator must be good at misleading the Discriminator. The Generator thus has to learn how to generate convincing samples that are in fact fake.
A common example is the generation of images. If we have a dataset of human faces, at the end of a perfect training, the Generator will perform quite well at generating human faces that have never existed in the first place, while the Discriminator will be able to separate human faces from everything else.
The purpose of the network is to minimize the classification error loss in order for the performances to be maximal.
In the previous paragraph, I spoke about "perfect" training, this is because the training stage of a GAN is really hard. It is difficult because the Generator and Discriminator have to follow what we call a Nash Equilibrium [16] in order to stay optimal. This means that both models, the Generator and the Discriminator, must learn at the same rate else it can occur that one of them becomes “too” powerful compared to the other. In that case, the training will stagnate.
If the training for the Discriminator is too perfect, it can also face vanishing gradient problems. The network will then learn nearly nothing and we will have no good results for the Generator.

The main problems with GANs are thus the convergence and stability issues. A lot of work has already been done in that space in order to counter some of the problems but it would take a completely new master thesis project to go through all of them and to make an in depth analysis.

Conditional Tabular GAN (CTGAN)

In this thesis, we are using modified version of GANs called CTGAN. This model is a variation of the classical generative adversarial networks. It is designed to model the "distribution"/dependencies of tabular data and then to sample rows from that distribution. The idea is thus to use this model to sample unseen data from the "distribution" of the historical data with returns of the assets.
The advantage of such a model is that it can take in a dataset and generate multivariate data based on the different assets. The whole process behind this idea and the theory can be found in the paper "Modeling Tabular Data using Conditional GAN" from Lei Xu *et al.* [17]
We use this model because it gives overall good results for generating samples based on tabular data and this also gives us a comparison model based on a GAN approach that could fit our requirements.

4.1.2 Variational Autoencoder (VAE)

The second type of models we consider are the Variational Autoencoders.[4] This family of models is based on the popular Autoencoders. The purpose of Autoencoders is to map a given input to a fixed vector in a latent space, this is the encoder part. The decoder part then tries to reconstruct the input based on the fixed vector.

Autoencoders are useful but cannot generate new data, this is why VAEs are used instead. The idea behind VAEs is that we can map some data to a given distribution. If we take that data as input, we can find a mapping between the input and the distribution by searching for the appropriate mean and standard deviation associated to that distribution with the encoder. Later, we can reconstruct the input in the decoder. VAEs are designed in a way that if we add a small noise when using the decoder we get a completely new output that reflects well enough the input but that is different.

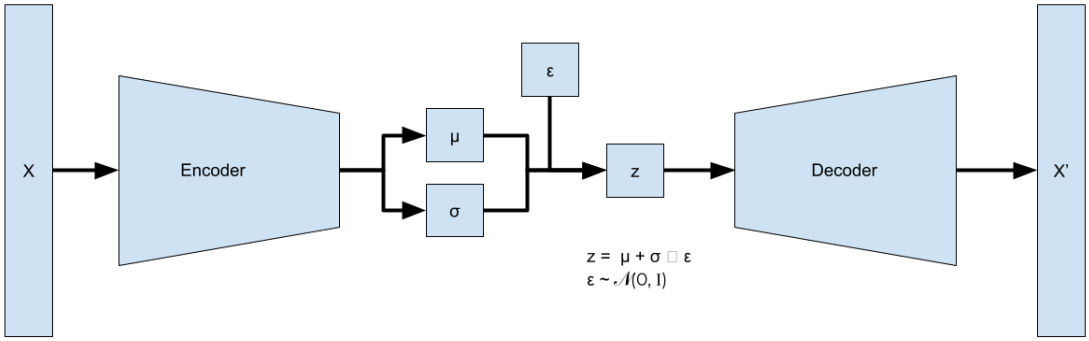

Figure 4.2: Global Architecture of a VAE

In Figure 4.2 we see the global architecture of VAEs. The idea is that once we have decided on a distribution that could be fitted on the input data, the next step is to find the mean and the standard deviation as shown in figure 4.2. This is done by the encoder, that is just a bunch of layers that compresses the input. With the parameters and the distribution, we are now able to reconstruct the input data. This reconstruction phase of the input is outputted by the decoder by sampling from the latent distribution. Because the decoder has learned to map values from the distribution to an output thanks to the z (reparametrization of the distribution), we can now simply sample multiple times from that latent distribution and pass it through the decoder in order to generate completely new synthetic data. We can thus in principle generate an unlimited amount of synthetic data.
The purpose of a VAE network is to minimize the difference between the reconstruction X' and its input X and to ensure that the distribution returned by the encoder is close to the given distribution.
The disadvantages of VAEs are similar to the disadvantages of GANs, they are difficult at converging. They take a lot of time to train.
In addition, another disadvantage of VAEs is that you need to make an assumption on the distribution.
Finally, if we want to generate synthetic data, a small noise ϵ has to be added, and this can sometimes result in weird outputs. With images, the output is often blurry.

An advantage over GANs is thus that the training is easier but the results are often not as good when GANs are well-trained.

TVAE

In the same paper as the one for CTGAN we can also find TVAE [17]. This type of model is simply an adaptation of the Variation Autoencoder approach to tabular data. The idea behind the creation of TVAE models was to challenge the performances of CTGANs. We are therefore using both models in this thesis in order to assess if, for our type of dataset, one could help us augment the historical data in a way that could ameliorate performances when applying financial strategies. Similar to the CTGAN model, TVAE can also take tabular data that is intrinsically multivariate and generate synthetic data based on the joint distribution of the data.

4.1.3 Normalizing Flow (NF)

Normalizing flows are a series of invertible transformations that map a simple distribution to a difficult one. Generating data is therefore as simple as sampling from the complex distribution through the simple one.
In the case of financial data we can see the complex distribution as the joint multivariate fitted with the data.

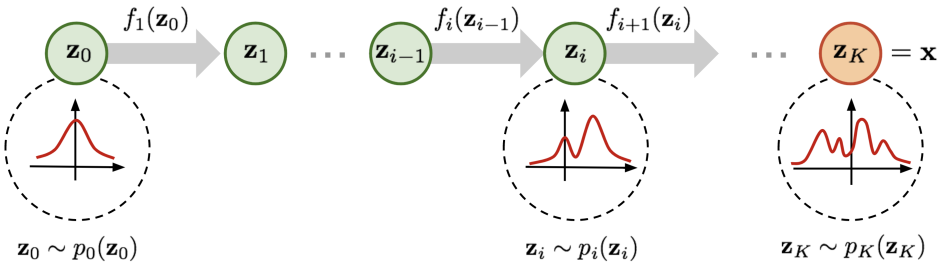

Figure 4.3: Normalizing Flows as a series of invertible transformations. (Source : Lilian Weng [18])

The advantage with normalizing flows is that you get an estimation of the distribution to which the input data is mapped. You can also check the bijection to the simple distribution. This is something VAEs lack, you can get a mapping between the output and the latent distribution but you don't have a mapping between the (sample) distribution and the (complex) output distribution. In other words, because we have a simple input distribution it is easier to sample from a complex output distribution. Another advantage is that the training is very easy and stable compared to GANs. The convergence issue is also not a problem compared to VAEs and GANs.
Normalizing flow models give quick results without a lot of parameter tweaking.

The main disadvantage of NFs is that the quality of a well trained and adjusted GAN or VAE is often better than that of a NF when looking at the generated samples. (Comparison available on Table 2 from a study on Deep Generative Models [7]). Naturally it depends on the kind of data, NFs are very promising when reproducing the input density.

Theoretical Background

Because Normalizing Flows play a big part in this work, some notions need to be understood and thoroughly explained. This part of the work is therefore dedicated to giving more insights in the theory behind NFs.
Normalizing flow is a generative approach that samples data from a joint distribution. If we have a simple density function $p_Z(z)$, for example a Gaussian function, and we want to transform this function to represent a more complex density function $p_X(x)$, we could do so by applying a set of differentiable transformations f. If $x = f(z)$ a transformation from the simple distribution, then the change of variable formula is written as :

$$p_X(x) = p_Z(z) \left| det \left(\frac{\delta x}{\delta z} \right) \right|^{-1} \tag{4.1}$$

One could easily sample from this distribution if the base density function is simple. The conditions to make this work are the following:

1. The function f has to be easily invertible
2. The determinant of the Jacobian should be easy to compute

Taking the log-likelihood from this function gives us :

$$log\ p_X(x) = log\ p_Z(z) - log\ \left| det \left(\frac{\delta x}{\delta z} \right) \right| \tag{4.2}$$

If we have multiple transformations we have :

$$x = z_k = f_k \circ f_{k-1} \circ ... \circ f_1(z_0) \tag{4.3}$$

Now for the combination between normalizing flow and autoregressive modeling [1]. If we see the joint probability density function as

$$p(x_1, x_2, ..., x_n) = \prod_i^n p(x_i | x_{<i}) \tag{4.4}$$

[1]In this case, autoregessive does not mean in function of time but it means in function of previous iterations

We can build the model as

$$x_i = \mu_i + z_i \cdot exp(\alpha_i) \quad \text{where} \quad \mu_i = f_{\mu_i}(x_{<i}); \quad \alpha_i = f_{\alpha_i}(x_{<i}); \quad z_i \sim N(0,1) \tag{4.5}$$

Where μ_i is the mean and α_i is the log standard deviation based on previous iterations (<i). f_{μ_i} and f_{α_i} are unconstrained scalar functions computing the mean and log standard deviation of the i^{th} conditional given all previous variables [2]. The z_i is a random number following a simple distribution that can be used to generate new data.
From this function we can also compute the random numbers z_i given a datapoint x_i

$$z_i = (x_i - \mu_i) \cdot exp(-\alpha_i) \quad \text{where} \quad \mu_i = f_{\mu_i}(x_{<i}); \quad \alpha_i = f_{\alpha_i}(x_{<i}) \tag{4.6}$$

For the determinant we know that the Jacobian is a triangular matrix by design and we can therefore write

$$\left| det \left(\frac{\delta x}{\delta z} \right) \right| = exp(\sum_i \alpha_i) \quad \text{where} \quad \alpha_i = f_{\alpha_i}(x_{<i}) \tag{4.7}$$

Combining Equation 4.7 and equation 4.6 into equation 4.1 we can get the density $p_X(x)$. Sampling from that complex distribution is then as simple as to sample from the Gaussian distribution with z_i.
We can also check whether the training was correct by looking at the bijection and checking that z_i correctly follows the simple distribution we defined.

MAF

Different strategies already exist using this combination of techniques. Real NVP [20] is an example of that. Real NVP is restricting the generation of the first $x_{<d}$ elements (d<n). Masked Autoregressive Flow (MAF) [19] is a generalization of Real NVP and doesn't constrain the first elements.
These models are combinations of an autoregressive model that uses normalizing flow to sample from the estimated joint probability.

[2] Described in section 3.1 from the paper developping MAF [19]

4.1.4 Copula

A copula can be seen as a function that describes the dependence between multiple random variables. We can write a multivariate distribution function as a set of marginal distribution functions and a copula. The copula links the marginal distributions to a multivariate distribution

The copula of the random variables $(X_1, ..., X_d)$ can be defined as the joint cumulative distribution function of the vector with d standard uniform random variables

$$U = (U_1, ..., U_d) = (F_1(X_1), ..., F_d(X_d)) \tag{4.8}$$

with $F_i(x) = \Pr[X_i < x]$ the marginal probability distribution of X_i

The formulation of the copula is then

$$C(u_1, ..., u_d) = \Pr[U_1 < u_1, ..., U_d < u_d] \tag{4.9}$$

Copulas are often used in the financial sector because of their ability to estimate the correlation between multiple assets. It is very powerful when we try to minimize the risk choosing a portfolio of assets because if we know the relations between the different assets, we can then easier diversify our portfolio with a bigger confidence. Sadly, Copula is not flawless and an example of that is depicted in the paper "The Gaussian Copula and the Financial Crisis: A Recipe for Disaster or Cooking the Books?" [10]

Gaussian Copula

This is a variant of the Copula where we assume that the variables follow a multivariate normal distribution.
We chose it because it is the most standard form of a Copula and because, as mentioned previously, copula is often used with financial datasets and has shown it can give good results when appropriately used.

Copula GAN

Copula Gan is the combination of Gaussian Copula and GANs. It combines the flexibility of the Gaussian Copula model together with the performances of CTGAN 4.1.1.
The methodology is that it first transforms each non categorical variables from the input using the Gaussian Copula approach to learn the distribution of the data. The second step is to fit the CTGAN model onto the transformed table.
We also use this model in our comparisons in order to assess whether the combination

of two well performing models in the field of generation of synthetic data could lead to even better results.

4.1.5 Summary

Criteria Comparison for data generation models					
Criteria	Multivariate	Training efficiency	Training context	Forecasting quality	Density reproduction
CTGAN	Yes	- -	+	-	-
TVAE	Yes	-	- -	-	++
MAF	Yes	+	- -	-	++
Gaussian Copula	Yes	++	- -	-	++
Copula GAN	Yes	- -	+	-	-

Table 4.1: Overview of the different criteria for generative models modelling the past. The columns represent the different criteria detailed in section 3.

Looking at table 4.1, we see that the main distinction between these models is the training. The easiest model to use is the Gaussian Copula generative model. On the other side of the spectrum the most difficult ones to train are the GAN models that really need to converge in order to have good augmented datasets.
All the models are utilizing and generating multivariate data and rely on most of the training data to generate synthetic samples (except for GAN based models). GANs can use most of the training data, but because it tries to generate the most realistic samples it can also decide to discard some samples when training because they aren't representative enough and thus only rely on the last few samples.
The purpose of GAN based models is to generate the most representative samples. For this reason, the distribution is not reproduced perfectly and this can have a negative effect on the estimation of the dependencies between the assets. All the other models are trained to generate data looking like the distribution of the trained dataset.
Also, none of the aforementioned models are trying to forecast when generating new data. The quality of the forecasting reflects the past evolution of the stocks and it can therefore be outdated similarly to the training context.

4.2 Time Series Forecasting models

Time series forecasting is a well known problem in research. As its name implies, it is trying to predict the future or extrapolate on the input data that are time series. A lot of strategies have already been developed in order to predict “future” data based on available observations. LSTMs and Transformers are the state of the art neural networks when it comes to predicting sequential data.
In this thesis we present the novel idea of combining existing generative models (GMs) with time series forecasting models (TSFMs) for financial portfolio optimization because we want to combine the ability of GMs to have good covariance matrix estimation with the ability of TSFMs to have good general predictions.

In the paper "Multivariate Probabilistic Time Series Forecasting via Conditioned Normalizing Flows" [13] we are presented with a new approach to the problem. In this work, starting from a dataset consisting of a multivariate time series, Normalizing Flow (section 4.1.3) is used to model the joint probability distribution of new future points (conditioned on past data). By sampling from this distribution, predictions of future values are possible.

We are, with time series forecasting models, trying to estimate the future joint distribution of a portfolio.
Our intuition is that these models can outperform the state of the art models that are used for financial portfolio optimization. The reason for this, is that instead of just relying on the underlying dependencies between the multiple assets we have in a portfolio, we will also include a temporal notion that will affect the estimation of the mean return on investment (ROI). The mean ROI is a really important metric because it is a core component of the Sharpe ratio (section 2.1.3). When we are maximizing the Sharpe ratio, we want to maximize the mean ROI while minimizing the volatility of the portfolio. If we are able to estimate well enough the future joint distribution of the data, the mean ROI estimation will be better and thus also the performance when maximizing the sharpe ratio.

4.2.1 Transformer based models

Transformers are now the go to neural network architecture in 2022. The model was introduced in 2017 in the paper "Attention is all you need" [21]. They are widely used for sequence to sequence tasks. Instead of using recurrence as for RNNs, transformers use an attention mechanism. This change is especially useful and noticeable when we want to use parallelization during training.

Architecture

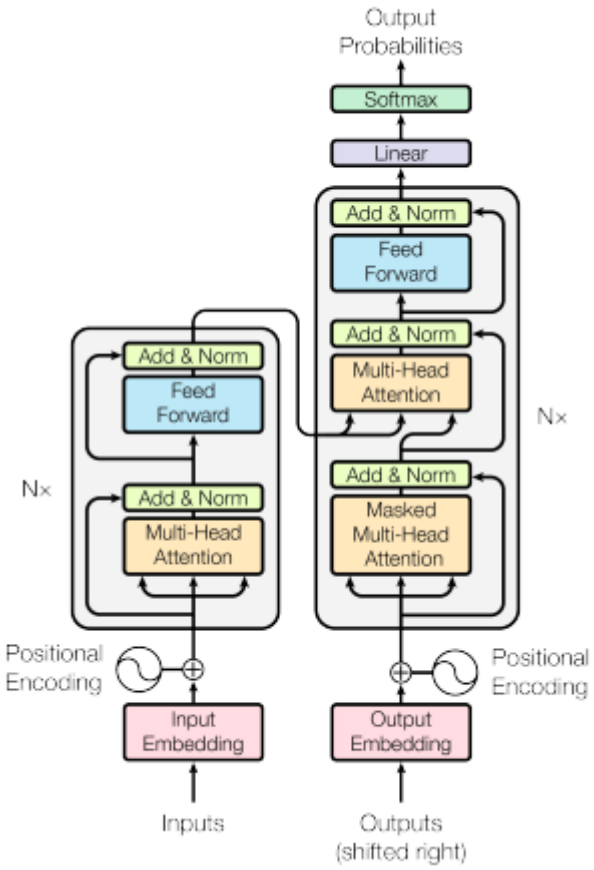

Figure 4.4: Transformer model architecture (Source : [21])

From the architecture 4.4 of the model we can deduce

- Encoder Decoder structure
- Attention modules
- Feed-Fodward and Normalizing layers
- Embeddings and Softmax
- Positional Encoding

Encoder Decoder Strucure

- **Encoder** is a set of layers that are stacked together. The encoder is bidirectional, this means we can go from input to embedding and vice versa, and creates a general embedding of the input sequence.
- **Decoder** is also a set of stacked layers but that is unidirectional because of the masked attention module. The utilization of a mask is used to prevent positions to access information at future positions that it tries to predict.

Attention modules

Attention is really important in our context because it will generate data based on the most important part of the training data. In other words it will put more attention on parts of the training data where the dependencies are the most interesting for generating future samples. We distinguish two parts of a typical attention mechanism.
The two modules are :

- Scaled-Dot Product module
- Multi-Head Attention module

More information about these modules has been given in the initial paper by Ian J. Goodfellow et al. [21], but their use in our context is to put more attention on the interesting parts of the training data.

Positional Encoding

This little addition is useful to remember the order in the sequence we try to feed to the network. This is because in contrary to RNNs, transformers see the input as a set of vectors with no sequential order. The relative position is added to the embedded representation of each sequence.

Temporal Fusion Transformer (TFT)

TFT is an attention-based Deep Neural Network. [22] It is a powerful type of transformer with multiple advantages described in an article by Nikos Kafritsas [23]. Here is a list of advantages over other time series forecasting models listed in the article:

- Diversified Features : Different types of temporal and time-invariant data are possible.

- Heterogeneous time series: Multiple time series are permitted during training which is very interesting for our use case with multiple assets.
- Multi-horizon forecasting : We can get prediction intervals which can be very useful when generating data and thus sampling from the "prediction" distribution.
- Interpretability : Because the model uses transformers and thus an attention mechanism, we could analyze feature importances
- Performance : It outperforms similar time series forecasting models.

Because of all these advantages we took the model into account in our final comparisons. A more in depth explanation about the implementation itself can be found in the initial article by the Google and Oxford team [22].

Transformer MAF

We chose the subject from the thesis based on a particular paper in the introduction [13]. The paper introduced a novel approach of multivariate probabilistic time-series forecasting. We can also find a section about "Temporal Conditioned Normalizing Flows" in the paper where a combination of transformers with normalizing flows is described. Although we are not using conditioning (based on time independent data) in our case, we can still use this technique for our own use.
The Normalizing Flows are used to model the conditional joint distribution at time t of all the time series by conditioning on an embedding from an attention module of the time series up to time $t-1$.

4.2.2 LSTM based models

Long Short-Term Memory Networks (LSTMs) [24] can be described as a type of Recurrent Neural Network (RNN) that tackles some of the problems of classical RNNs. One of the big problems with recurrent neural networks is the fact that it has a hard time handling long term dependencies that we can find in large sequences. This problem can lead to vanishing/exploding gradients [3] with the back propagation.

[3] An explanation on the vanishing/ exploding gradient problem by Roger Grosse : https://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/readings/L15%20Exploding%20and%20Vanishing%20Gradients.pdf

Long Short-Term Memory

In the paper "Speech Recognition with Deep Recurrent Neural Networks" [25] Alex Graves *et al.* describe the architecture of a LSTM memory cell. In a typical LSTM architecture, multiple LSTM cells would follow each other. The weights in a LSTM model are learned with Backpropagation through time. [4]

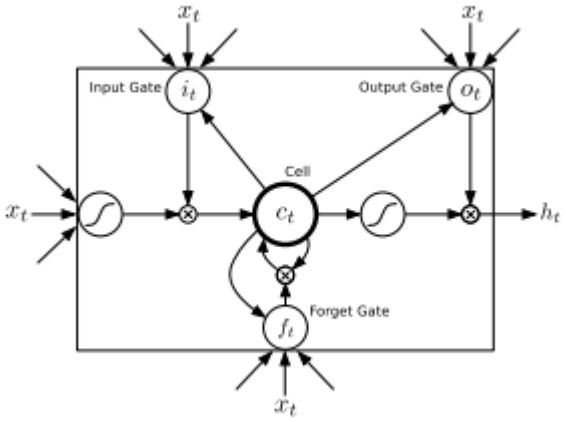

Figure 4.5: Long Short-Term Memory Cell (Source : [25])

In the structure we can find multiple components:

- x_t : the input sequence
- h_t : a hidden vector sequence
- c_t : the cell state
- **input gate** : an option to include or not x_t
- **forget gate** : an option to choose whether h_t depends or not on c_t
- **output gate** : an option to choose whether c_t depends on c_{t-1} or not

The strength of LSTM models are naturally to make sequence to sequence predictions. They are often used in NLP problems where, for example, we want to translate a sentence from a language to another.
As explained above, LSTM can also handle the problem of vanishing/exploding gradient that we encounter with long sequences.

[4] An explanation on Backpropagation through time : `https://en.wikipedia.org/wiki/Backpropagation_through_time`

LSTM MAF

LSTM-MAF is very similar to Transformer-MAF except for a few points in training. For this model, the MAF part of an LSTM-MAF is used to model the conditional joint distribution at time t of all the time series by conditioning on the hidden state h_t.

The advantage of a Transformer based model over an LSTM approach here is that everything can be learned in parallel which can have an influence on the time needed for the training when there are a lot of assets.
The advantage of LSTM based models over the Transformer approach in this context, is that for long sequences with not many assets it is quicker to train.

LSTM DeepVAR

DeepAR is another type of probabilistic time series forecasting model and DeepVAR is a multivariate adaptation of it that we use in this thesis. More information about the model can be found in the article from Salinas *et al.* [26].

The main idea behind this model is that we combine a DeepVAR approach with an integration of LSTM. [27]
It has some LSTM properties because it implements LSTM cells for the evaluation of the hidden layer of the autoregressive recurrent network. The details of the implementation of a DeepAR with LSTM cells are given in the paper mentioned above. In general, DeepAR has shown good probabilistic forecasting performances. DeepAR has also shown it can handle datasets with few historical data. A lot of well performing assets have not been created that long ago and it is therefore a plus with such datasets.
Because of all these advantages it was an easy pick for our comparisons.

4.2.3 Summary

Criteria Comparison for data generation models					
Criteria	Multivariate	Training efficiency	Training context	Forecasting quality	Density reproduction
Transformer MAF	Yes	-	++	++	+
TFT	No	-	++	++	- -
LSTM MAF	Yes	-	++	++	+
LSTM DeepVAR	Yes	-	++	++	-

Table 4.2: Overview of the different criteria for generative models modelling the future

On table 4.2 we look at the different criteria characterizing the generation of financial data with time series forecasting models. One big difference we immediately notice, is that the TFT model is the only model that doesn't rely on multivariate data but instead it will individually evaluate the different assets of the dataset and perform forecasts. All the other models are generating multivariate data that is based on the future joint probability density function of the assets. The training is not the easiest, because it takes a lot of time due to their deep architecture. On the other side, because they all rely on a transformer or LSTM based architecture, they can all focus on a specific part of the training data (selective training context) in order to forecast.
Because TFT doesn't rely on the multivariate aspect of the data and only tries to forecast based on a particular pattern it finds in the training data, we cannot say that there is a reproduction of the density/distribution of the assets. On the other hand, LSTM DeepVAR and all the MAF based models are trying to reproduce the dependencies(MAF models)/patterns(DeepVAR model) between the different assets from the past while "predicting" the future evolution of the stocks.

Chapter 5

Experimental Setup

5.1 Data Generation

At the end of the introduction we mentioned our approach innovates in a few ways. What I meant by that is that the models we are using aren't new at all, but the methodology behind our approach is.
In this section we are describing how the models shown in the comparisons are generating new synthetic data.

Again, we are distinguishing the two types of models. Both have their own way of generating data, and we have to manipulate that generation in order to be able to use the data to our advantage when optimizing according to a financial strategy.

5.1.1 Methodology: TVAE-MAF-GANs

The first category of models, is an approach where we estimate the past joint probability distribution. The generation is straight forward and is described below.

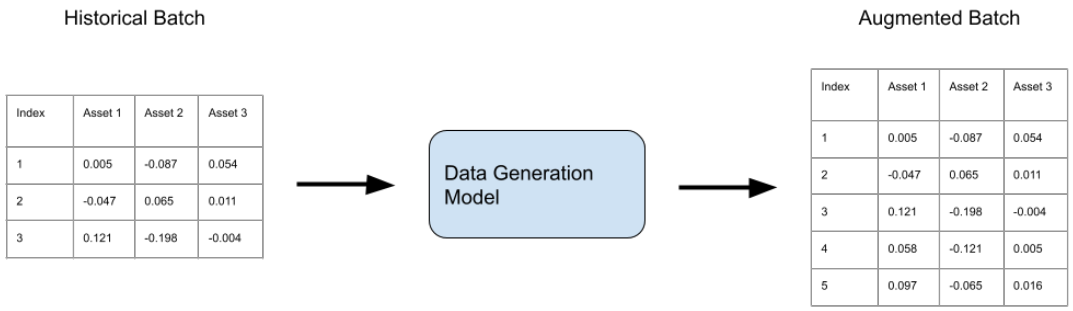

Figure 5.1: Simplified Data augmentation framework with an example dataset: row 4 and 5 were generated and augment the original dataset.

In the Figure 5.1 we can see how our models will augment the datasets. In this example, we have a dataset containing 3 assets. Every row then represents a percentage change between two consecutive days in asset price. The purpose of our Data Generation models is then to augment the dataset by sampling new rows that fit into the dataset that we have. In other words, we find new samples that are completely possible given the past distribution of the data. Each sample corresponds to a percentage variation of every asset found in a dataset/ initial portfolio. In theory, we could generate an infinite amount of new rows that would give us the dependencies of typical evolution of a portfolio.
More information about the datasets has also been discussed in the Background section 2.1.1.

5.1.2 Methodology: Time series forecasting

In this section we are going to analyze the differences between how time series forecasting models usually predict the future against how we are using them to generate new data. The methodology behind the generation of new data points will be clear after reading this section.

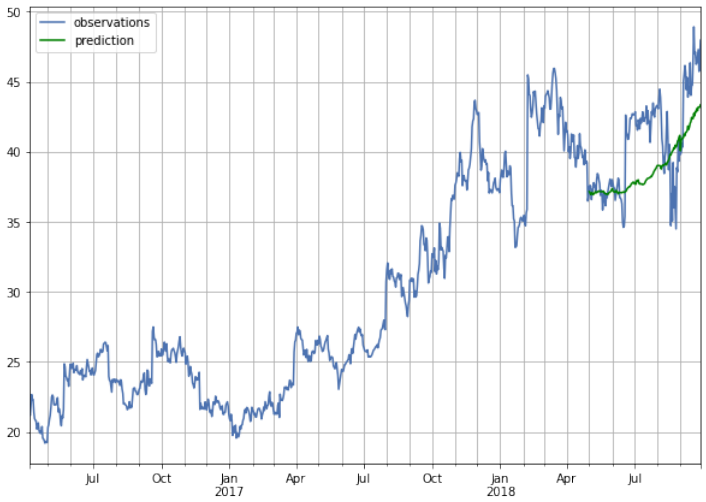

Figure 5.2: Example of a prediction for a single asset. The blue line represents the true evolution of the asset price while the green line shows the prediction from a time series forecasting model.

In Figure 5.2 we see a prediction of a time series forecasting model indicated by a green line, while the blue line is the real data associated to that particular asset. In a normal portfolio we would have a multivariate distribution that we would like to forecast but for the sake of simplicity we are showing how it forecasts for a single asset time series. The neat detail behind time series forecasting models is that we are also able to generate multiple scenarios. The prediction shown on the figure is thus the mean of all the scenarios generated.

Beneath, we show how we can represent all these scenarios and how we can use them in our case.

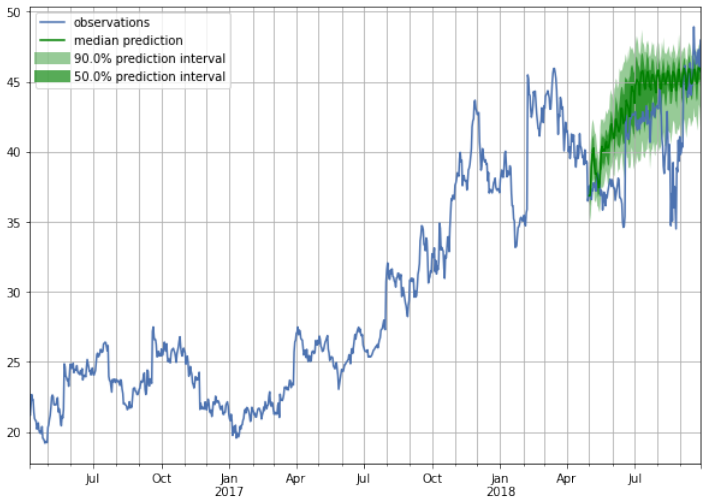

Figure 5.3: Example of data generation for a single asset: for each future time step, new data points can be generated by sampling from the future (green) distribution.

In Figure 5.3 we immediately see the difference with the previous figure. Instead of simply having one prediction, we have a a distribution of the forecasts for that asset. We get that distribution by sampling a lot of scenarios and then looking at the prediction intervals. The scenarios that are the most likely will end up in the more dark region of the forecasting while unlikely scenarios are shown in a clear green color.
We see that the estimation is absolutely not perfect because a lot of the true datapoints are not even included in the prediction interval. We always have to keep in mind that we are trying to predict stock returns. If the prediction was perfect we would already be rich.
We are using all those scenarios in our datasets and computing the percentage change between two consecutive days in order to have more data available for our use case.

The image below shows the pipeline we follow to generate new data with the novel approach.

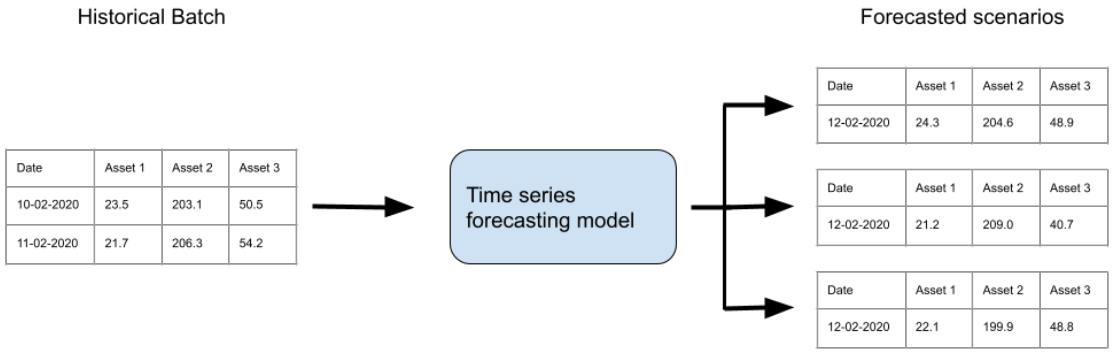

Figure 5.4: Schema with an example of a time series forecasting data generation pipeline. Here, many possible instances for one future day are generated and can be used to augment the original dataset. The procedure can go on for more future days if desired.

On the simplified example above 5.4 I present how time series forecasting models can be used to generate new data points. Instead of generating new entries in a given table, we are creating new scenarios and thus new tables.
In this example, the existing data consists of 3 assets with returns associated to two consecutive days. The task here is to predict the outcome for the next day by generating 3 independently sampled scenarios.
Looking at figure 5.1, we can see that this new data for this approach, cannot immediately be used. We first have to compute the daily returns percentage change for all the scenarios and concatenate them together. For the simplified example above, the scenarios are useless because we only have 1 new date generated per scenario but we need at least 2 new date entries in order to compute percentage changes.

5.2 Backtesting

In this section, we are talking about the procedure we follow to evaluate the performance of our models compared to existing ones.

A model is evaluated based on a few key parameters:

- The financial strategy
- The assets it works on

- The length of the evaluation period
- The evaluation period

5.2.1 Financial Strategy

Strategies have already been discussed in another section 2.1 and they are essential when talking about portfolio optimization. They define how certain assets are chosen to maximize a criteria.
In our case we distinguish two main strategies. The first one is the maximisation of the Sharpe Ratio and the second one is the minimization of the volatility/risk.

5.2.2 Asset Choice

As discussed in the dataset section 2.1.1, a dataset consists of 35 to 65 assets picked at random in the list of more or less 3000 assets from the Wilshire 5000 index. Creating a dataset in that way doesn't take into account the dependencies between the assets. It could well be that all the assets originate from the same industry type (e.g. pharmaceutical). In that case that particular dataset is very correlated and minimizing the volatility can therefore be a challenge because with such a strategy, diversification is recommended.

In order to have diverse cases in our tests we repeat the procedure of sampling the assets more than 50 times in order to create above 50 distinct datasets. The tests are then conducted on these datasets and we can analyze how the models perform with various and independent dataset choices.

5.2.3 Length of the evaluation period

An important criteria in the evaluation process is the choice for the "length of the prediction period". A model could be very good at predicting long term dependencies between assets and thus be very good at optimising a metric, but it can also be very bad at predicting the short term fluctuations. This could be a result of volatile data.
In our case we chose to go for an evaluation period spanning between 20 to 250 days, knowing that a trading year comprises about 252 "open" days. With a short training period this doesn't give us a lot of wiggle room in terms of the size of the evaluation period.

5.2.4 The evaluation period

The choice for the evaluation period can have a huge impact on the results. It can vary a lot depending on the fact we are in a period of financial crisis or not. Normally we would expect the data generation techniques to mitigate these changes in period but nevertheless, the results will still be affected by these sudden fluctuations.
In most of the analysis we will perform, we will looked at one particular period, this is to give an idea on what to expect with the techniques we are using. This doesn't mean we will always see similar results.
We will herefore conduct a research about the change in performances overtime to look how models behave with variable time spans.

5.2.5 Evaluation plan

Finally, if we want to bring everything together we have to talk about the entire pipeline as a whole.
First, our data is assembled and more than 50 distinct datasets are formed. Then, all the datasets are tested and evaluated one by one.

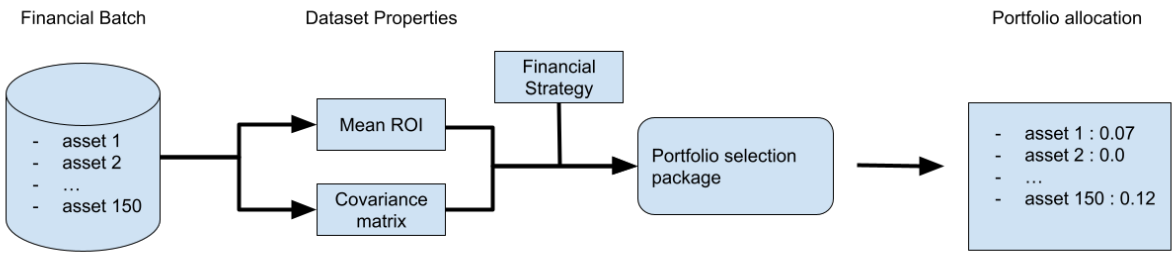

Figure 5.5: Portfolio optimization pipeline.

The procedure for is the following:

1. The dataset is passed to the models and these are fit on the data.
2. The data is augmented (new data points are generated) depending on the particular model we choose. This is our new synthetic dataset. The framework has been described on Figure 5.1.

3. Important metrics used by the strategies are computed (i.e. individual mean ROI of the synthetic data, covariance matrix).

4. The synthetic data is then used by the financial optimization package pypfopt [28] based on the chosen strategy and a dictionary with weights associated to assets is outputted as we can see on Figure 5.5.

5. We evaluate the performance of the model by comparing its performance (on held-out data points) with the performance of a "perfect model" or an oracle model. A perfect model (with the best performance) can be retrieved by using the real future data (i.e. the actual future covariance matrix and the future mean return on investments).

6. Finally we repeat this procedure for the more than 50 distinct datasets and plot the comparison based on the results gotten.

5.3 Experiments

There are two families of models we are analyzing. The first one consists of Generative models that learn the joint probability distribution from the available historical data until the period we are evaluating. These models sample from the joint probability distribution. The distribution is naturally multivariate and the relationship between different assets from the same dataset can be learned and represented. This sampling could theoretically be performed an infinite amount of times. The more data we have, the better the estimated covariance matrix is. The main idea behind this family of models is therefore to improve the estimation of the covariance matrix by augmenting the dataset with more alike datapoints.
With such an approach, we are considering that the future is an image of the past. Future evolution is less important in this case than the relationship we are assessing between the different assets. Nevertheless, this approach should be better at optimizing a portfolio for the Sharpe ratio and the volatility because both of them are dependent on the covariance matrix.

Here is the list of generative models modelling the past used :

- MAF (Normalizing Flow, section 4.1.3)
- Gaussian Copula (Copula, section 4.1.4)
- TVAE (Variational Autoencoder, section 4.1.2)
- CTGAN (Conditional GAN, section 4.1.1)

- Copula GAN (CTGAN + Copula, section 4.1.4)

The second family of models are the time series forecasting models with joint probability density function estimation that are not per se used to predict the future evolution of the stock and its return. The covariance matrix is evaluated on the basis of what the models are outputting in the future. This family of models is also the reason behind the research of this master thesis. The idea originates from a paper written by the Zalando Research team [13].
Some of the models we are testing are not multivariate (TFT) and therefore we would expect a worse estimation of the future covariance matrix.

Here is the list of time series forecasting models modelling the future we are using:

- Temporal Fusion Transformer (section 4.2.1)
- Transformer MAF (section 4.2.1)
- LSTM MAF (section 4.2.2)
- LSTM DeepVAR (section 4.2.2)

In the comparison process we want to show as much as possible the effect of using these machine learning models compared to what we could have gotten from the real data. We also compare with the best possible setting where we take the future covariance matrix and the future mean return on investment. At last, we also combine the future covariance matrix with the mean return of the past historic data. In that way, it is possible to conceive the impact of the covariance matrix on different strategies. For example, in the max sharp strategy we use the mean return and the covariance matrix, if the covariance matrix is set to the ideal future setting, then we can see the effect that the mean return has.

For the meta-parameters of the models we always try to keep the same for all of them (where we can) in order to be as neutral as possible (explanation in Appendix a). This means that it could well be that the models are not tweaked to optimality but for our purpose, it is not a problem.

The evaluation plan for the tests has been detailed above, so we can immediately talk about what tests we have performed, why we chose them and what we can get out of them.
The first step in the testing process is to gather as much information as possible.

The section about backtesting 5.2 explains how the data is gathered, but in summary we pass every dataset into all the different models. As output we get an augmented dataset or as we call, the "synthetic dataset".
Once the synthetic dataset has been created we compute the mean return and the covariance matrix on that dataset. As explained above, these two metrics are utilized during the evaluation of the strategies.
A comparison graph is then outputted and we can visualize what model performs better than the others.
The last step is the explanation of the problem. We try to identify what metric could help us understand the behavior of certain models. Why one yields better results than others.

The tests we want to perform are the following :

- Analyse the effect of the prediction length on the results
- Analyse the effect of the number of assets in a dataset
- Analyse the effect of the financial strategy on the results
- Analyse the effect of the evaluation period
- Make an overall comparison of the results overtime

Chapter 6

Results and analysis

In the previous chapter, we asked some questions that needed an answer in order to understand how different factors of a portfolio prediction task influence the results. This part of the thesis is tackling these questions and based on different graphs and comparisons, we propose a logical interpretation of those results.

6.1 Outcomes with different financial strategies

This part of the results is dedicated to a detailed explanation of the performances of our models with respect to the Max Sharpe strategy (section 2.1.3) and the Min Volatility strategy (section 2.1.3).

6.1.1 Max Sharpe ratio

The Sharpe ratio (section 2.1.3) is a metric that takes the volatility and the returns of a set of assets with their allocated weight. The optimization has as purpose to maximize the return while minimizing the volatility or risk of a given portfolio.
We compare all the data generation models in this section and show how they perform against an optimization based on only the past data, no augmentation. Because the metric depends on the volatility and the returns, we can also identify how the augmented data is used to estimate these two submetrics with respect to a given model.

Normalized Sharpe Comparison for Generative models				
Models	50d - 35a	50d - 65a	150d - 35a	150d - 65a
Oracle	1.000	1.000	1.000	1.000
Ideal Covariance	0.325	0.344	0.387	0.359
True data	0.330	0.301	0.416	0.366
MAF	0.351	<u>0.312</u>	<u>0.505</u>	0.452
Gaussian Copula	0.348	0.234	0.361	0.272
Copula GAN	0.329	0.247	0.477	0.414
TVAE	0.304	0.217	0.385	0.301
CTGAN	0.339	0.251	0.452	0.396
TFT	**0.375**	0.306	0.403	0.408
Transformer MAF	<u>0.362</u>	**0.333**	**0.543**	**0.532**
LSTM MAF	0.323	0.288	0.490	<u>0.519</u>
LSTM DeepVAR	0.361	0.327	0.491	0.432

Table 6.1: Comparison table for the Sharpe ratio normalized with the ideal setting. The best models according to a setting are marked in bold, while the second best performing are underlined. We don't mark unattainable models.The settings (column names) are a combination of the evaluation period length and the number of assets in every dataset. In the first column we have "50d - 35a", this thus stands for 50 days evaluation period, 35 assets per dataset.

The table above 6.2 shows the mean of the sharpe ratio results for every single model. This mean is normalized by the ideal setting explained below. A perfect model will be closer to 1.
The formula for the normalization is the following:

$$R = \frac{\sum_{i=0}^{n} X_i / Y_i}{n} \tag{6.1}$$

where

- **R**: the normalized result based on a financial strategy
- $\mathbf{X}_i$: the result from a particular model trained on the augmentation of dataset i
- $\mathbf{Y}_i$: the result from the ideal setting/oracle based on the true future metrics from dataset i
- **n**: the number of datasets we are evaluating

In other words we divide each result for a given dataset by the result on that same dataset but computed by the ideal setting. Then we take the mean of all the divisions.
In this table we distinguish four categories of models (separated by horizontal lines).

1. The first category consists of models that are unattainable because they take into account future data. The ideal setting/oracle has the covariance matrix of the future data and the mean return of the future data. On the other hand, the ideal covariance model only has the future covariance matrix but keeps the mean return of the training/past data.

2. The second category only contains one model and it is the Real setting/ True data. This model is composed of data that is not augmented. In other words only the past data.

3. The third category represents all the generative models (section 4.1) that will not try to forecast the past data, but it will fit and try to find the joint distribution of the past.

4. Finally, the last category contains all the time series forecasting models that will try to find the future joint distribution of the assets in our dataset.

We will make an analysis on the effect of a longer evaluation period or a bigger portfolio size in later sections (6.2 6.3), hence we are not discussing that here but instead we focus on the ranking of the models.
In nearly all the different settings we see that our Transformer MAF model is the best performing one. Compared to a not augmented approach we clearly see a significant difference. In general we can see that the time series forecasting models are achieving good results compared to the first category of models. The normalizing flow approach with MAF shows also promising normalized sharpe ratios in all the settings.
A reason for Transformer MAF to have better results than a classic MAF approach could be that it has an overall good estimation of the dependencies between the assets, and a good estimation of the returns in the future whereas the classical MAF model lacks this last part. The main difference between the two is that the classical MAF approach tries to estimate the joint distribution on the available data, while the time series forecasting MAF approach tries to estimate the joint distribution of the future of that same data.

I have to clarify that the results we are witnessing here are also a direct effect of the period we are choosing to evaluate. In a later section we will see that

there is an important effect for the choice of evaluation period (section 6.5). The results are therefore not representative of a universal truth for all the use cases, here we are considering one particular period and analyzing different sets of assets, not all the possibilities. They represent the possibilities that we could obtain by augmenting financial data.

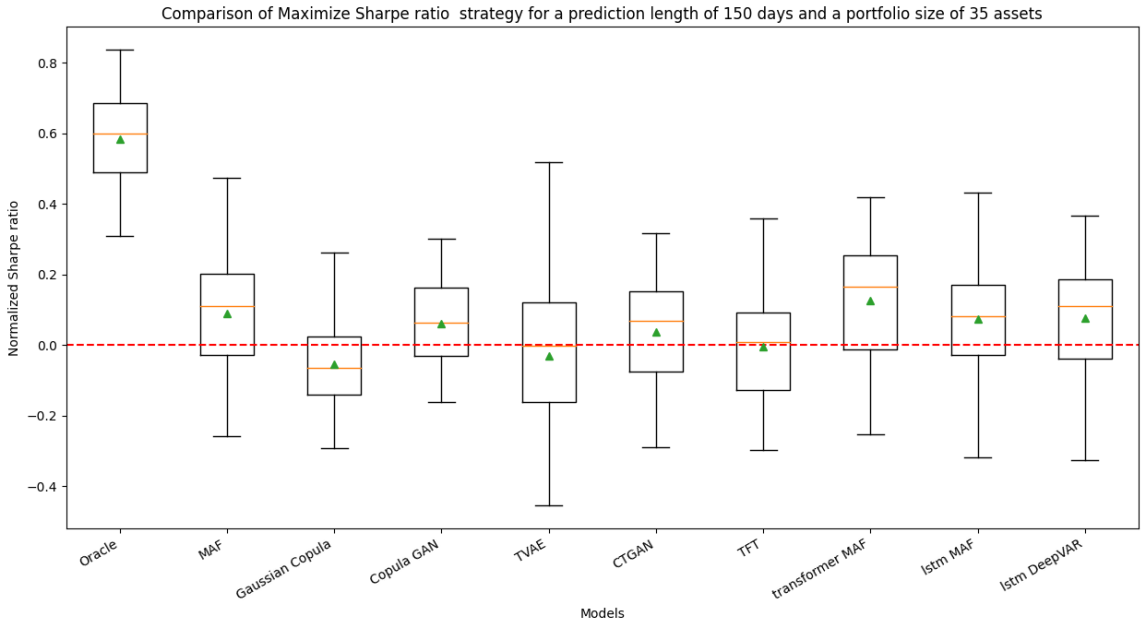

Figure 6.1: Box plot showing the Max Sharpe for 50 datasets with respect to generative models. The example is for a prediction length of 150 days and a portfolio size of 35 assets. The score is normalized by the true data and the oracle data.

In Figure 6.1 we show a boxplot of the mean returns with a MAX Sharpe strategy but this time normalized by the true data and the oracle data.
The normalization for this boxplot is straight forward:

$$result_{normalized} = \frac{result_{model} - result_{true}}{result_{oracle}} \tag{6.2}$$

We cannot simply divide by the True result because it may be negative whereas the result for the oracle is always positive in all our experiments.
The red line represents the mean of the true/past data, which is always 0 due to

the normalization. The first model is the Oracle and stands for the ideal setting. The same conclusions can be drawn for the different data generation models as the ones for the table. We clearly see that most of the time, time series forecasting models have better outcomes than the true data, but certainly not all the time. We see that the MAF approaches have overall the best normalized results.

Returns for Max Sharpe

As explained previously, the Sharpe ratio consists of the return on investment and the risk for a dataset of assets. If we have a good Sharpe ratio this could mean that the returns are very high and that the risk estimation isn't very good, or even the inverse. If the returns are higher than the Sharpe ratio, this means that the selection of asset weights has also a higher volatility compared to an approach with the same normalized return rate. We can therefore analyze what the mean returns are for all the different models after optimizing the Max Sharpe ratio.

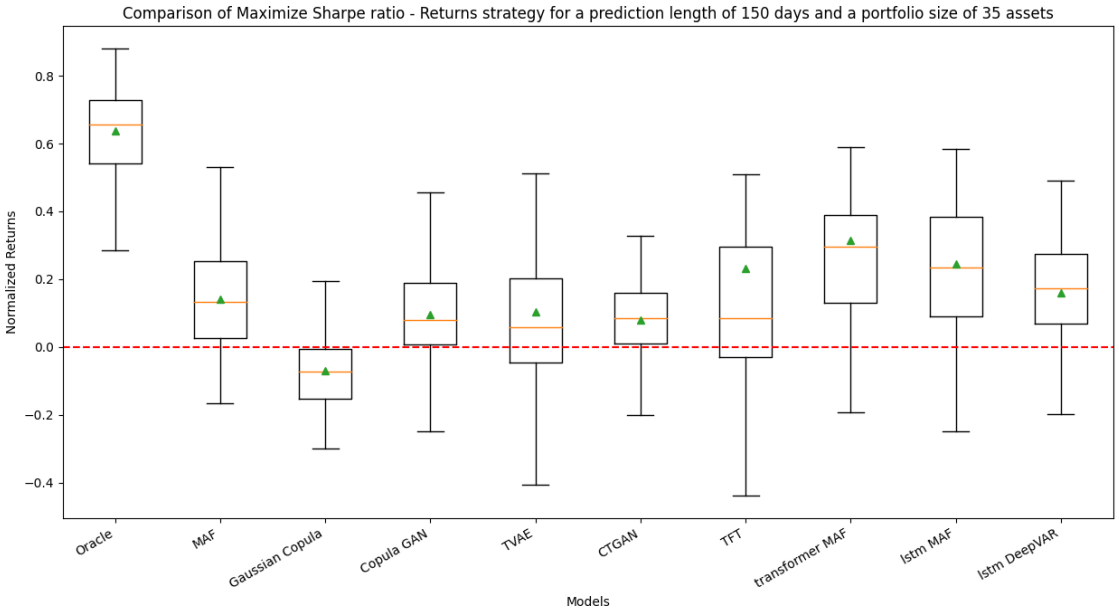

Figure 6.2: Box plot showing the Max Sharpe Returns for 50 datasets with respect to generative models. The example is for a prediction length of 150 days. The score is normalized as in Equation 6.2

In Figure 6.2 we see that the returns are in general very high compared to the

returns of the real setting. Just by looking at the Transformer MAF model, we see mean returns that are the highest and not too far from the oracle returns.

Volatility for Max Sharpe

Similar to the previous paragraph, we can conduct the same analysis for the volatility part of the Sharpe ratio.

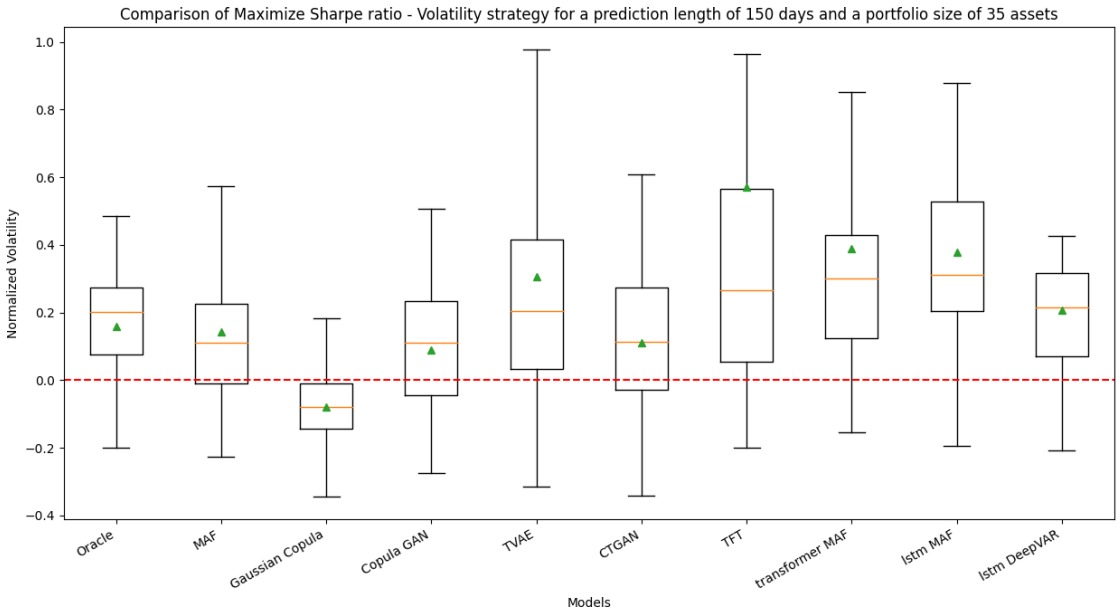

Figure 6.3: Box plot showing the Max Sharpe Volatility for 50 datasets with respect to generative models. The example is for a prediction length of 150 days. The score is normalized as in Equation 6.2.

As we saw in Figure 6.2 the returns were much higher than the returns for the real setting. The returns were also higher than their associated Sharpe ratio, for this reason the volatility of the portfolios is also high in most of the models. In Figure 6.3 we even see that the volatility for the portfolio chosen by the ideal setting is higher than the volatility of the real setting. This clearly proves that the Sharpe ratio tries to find the best compromise between volatility and returns.

6.1.2 Minimum Volatility

Similarly to the Max Sharpe strategy, we are analyzing the performances of the models but this time optimized according to the Minimum Volatility strategy (section 2.1.3). Something we have to keep in mind through this analysis is that the volatility only depends on the covariance matrix. A good model for the Min Volatility strategy thus must evaluate correctly the future dependencies between the assets.

Normalized Volatility Comparison for Generative models				
Models	50d - 35a	50d - 65a	150d - 35a	150d - 65a
Oracle	1.0	1.0	1.0	1.0
Ideal Covariance	1.0	1.0	1.0	1.0
True data	**1.288**	**1.394**	**1.225**	**1.31**
MAF	1.835	2.17	1.74	1.962
Gaussian Copula	<u>1.372</u>	<u>1.469</u>	<u>1.271</u>	<u>1.367</u>
Copula GAN	1.606	1.974	1.473	1.714
TVAE	2.164	2.405	1.835	2.455
CTGAN	1.667	2.023	1.533	1.78
TFT	1.437	1.622	1.689	2.017
Transformer MAF	1.433	1.686	1.344	1.534
LSTM MAF	1.398	1.73	1.377	1.502
LSTM DeepVAR	1.57	1.884	1.715	1.914

Table 6.2: Comparison table for the volatility normalized with the ideal setting. The best models according to a setting are marked in bold, while the second best performing are underlined. We don't mark unattainable models.The settings (column names) are a combination of the evaluation period length and the number of assets in every dataset. In the first column we have "50d - 35a", this thus stands for 50 days evaluation period, 35 assets per dataset

The descriptions of the layout of the table has already been done in the previous section 6.1.1.
Here we see that none of the models outperform the real setting/true data consisting of the covariance matrix from the past data. We also see that the classical MAF approach has a lot of trouble with the Min Volatility strategy, while the Transformer MAF approach handles it better but still isn't better than the true data.
It is difficult to explain why we see such behaviors. The sole conclusion we can take is that the data augmentation models aren't well suited to find the covariance matrix of the future joint distribution.

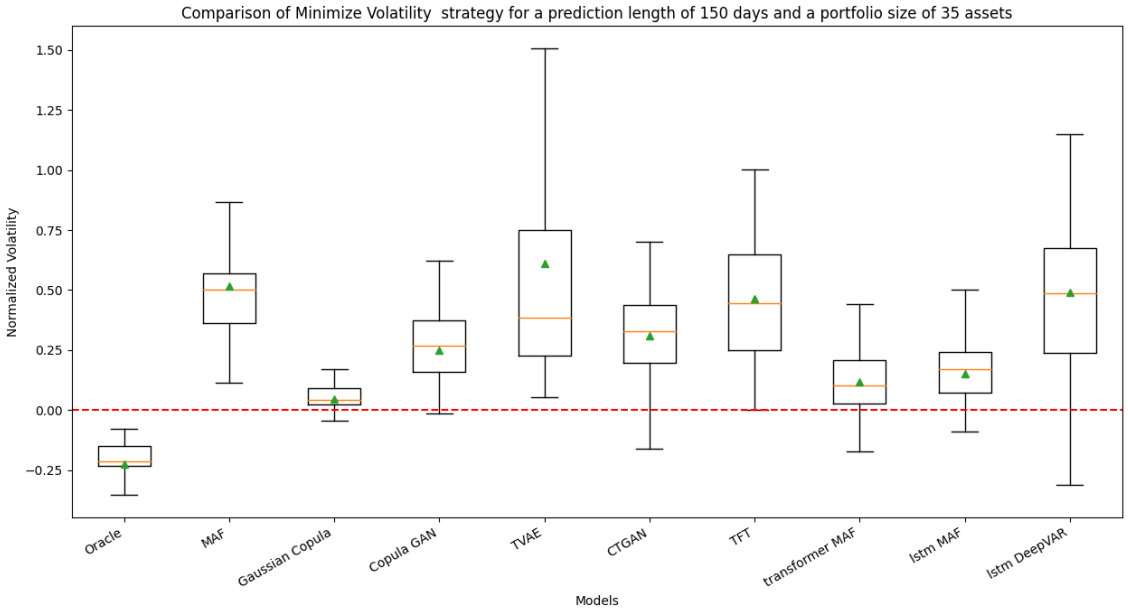

Figure 6.4: Box plot showing the Volatility for 50 datasets with respect to generative models. The example is for a prediction length of 150 days. The score is normalized as in Equation 6.2.

On the Figure above 6.4 we see a boxplot representing the comparison between the different models in evaluating the volatility. Points above the red line have a worse volatility rate than the mean volatility for the real setting/true data.
We understand that there is no feasible model (that we are evaluating), that has a better mean volatility than the true data. Only the ideal setting overcomes it. Visually it is also clear that the Transformer MAF and LSTM MAF are well performing while the Gaussian Copula is the best data generation model for the evaluation of the volatility.

6.2 Effect of the prediction length on the outcome

The purpose of this analysis is to assess whether there is a relationship between our common results for the different metrics, and the length of the prediction period. This is all according to a particular financial strategy and according to specific models.
The idea behind this study is that we want to know whether the long term future is easier to estimate than the short-term future. We expect long term dependencies to fade the farther we are from the past, correlation between the assets could be very different through time. On the other hand, we could expect short-term predictions to be worse because of the volatility that could vary a lot in the short term. To make sure that the comparison makes sense, we take the same datasets, with the same assets and training period, and we are just evaluating the differences between both outcomes.

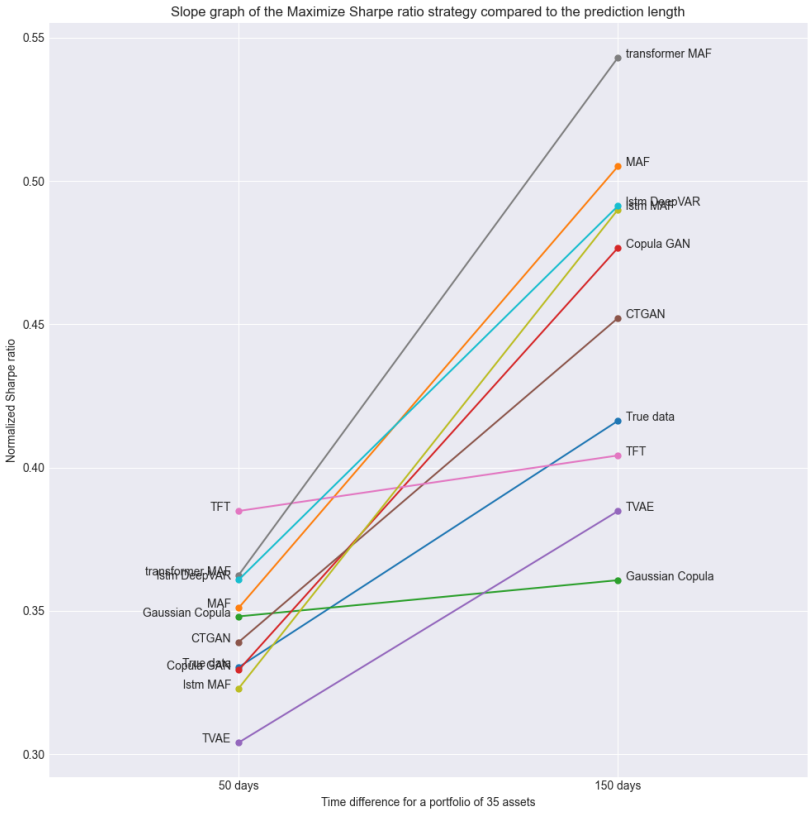

Figure 6.5: Influence of the prediction length with respect to the normalized Sharpe ratio. The slope represents the difference between the mean of the normalized scores of the 2 different settings (50-150 days). Every portfolio contains 35 assets.

In figure 6.5, I present a slope graph that shows the difference between results obtained on a short-term evaluation period (50 days), and on a long-term evaluation period (150 days). These results are for portfolios containing 35 assets, in the appendix you will find the same slope but on portfolios of 65 assets b.2. The conclusion from datasets with more assets are the same in this case.
Both scenarios are normalized by the result from the ideal setting. The ideal setting consists of the portfolio allocation package that gives weights associated to every asset based on the covariance matrix of the evaluation period and based on the mean return on investment (ROI) from the portfolio during the evaluation period.

From the graph we can deduce that every single model performs better when it is tasked to augment the data for a portfolio in the long-term. We can also see from the graph that time series forecasting models are in general better performing than the simple fitted generative models for the MAX Sharp strategy 2.1.3. This is even more true in the long term for that same strategy. An explanation for this could be that in the long term, the main difference between the normal generative models and our novel approach is that we have a better estimation of the future relationships between the assets and a better estimation of the mean ROI.

In the appendix b.1, I have also included the same graph but this time evaluated on the MIN volatility strategy 2.1.3. The observations are nearly the same. We still have a better normalized score when evaluating in the long term (except for a small proportion of the models). The explanation behind this could be similar to the explanation for the MAX Sharpe strategy, the short term could be too volatile and this can lead to wrong estimations of the dependencies between the assets (covariance matrix).

6.3 Effect of the portfolio size on the outcome

The second study is to analyze the effect of the portfolio size on the different models. As we know, a good portfolio of assets needs to be diversified in order to not end up without any money after a financial collapse. We now try to know if having a lot of assets in our dataset, has an effect on the results we get when evaluating a model according to a financial strategy.
Because the training of the model takes a lot of time, we are only evaluating two different portfolio sizes. The first type of dataset will contain approximately 35 assets, while the second type of dataset contains 65 assets.
The assets in the two families of datasets are chosen at random. All the models are evaluated on the same dataset. On the other hand dataset 1 from the family with 35 assets doesn't contain all the assets from dataset 1 from the family with 65 assets. In section 2.1.1, I explain how the assets are chosen in order to form datasets.

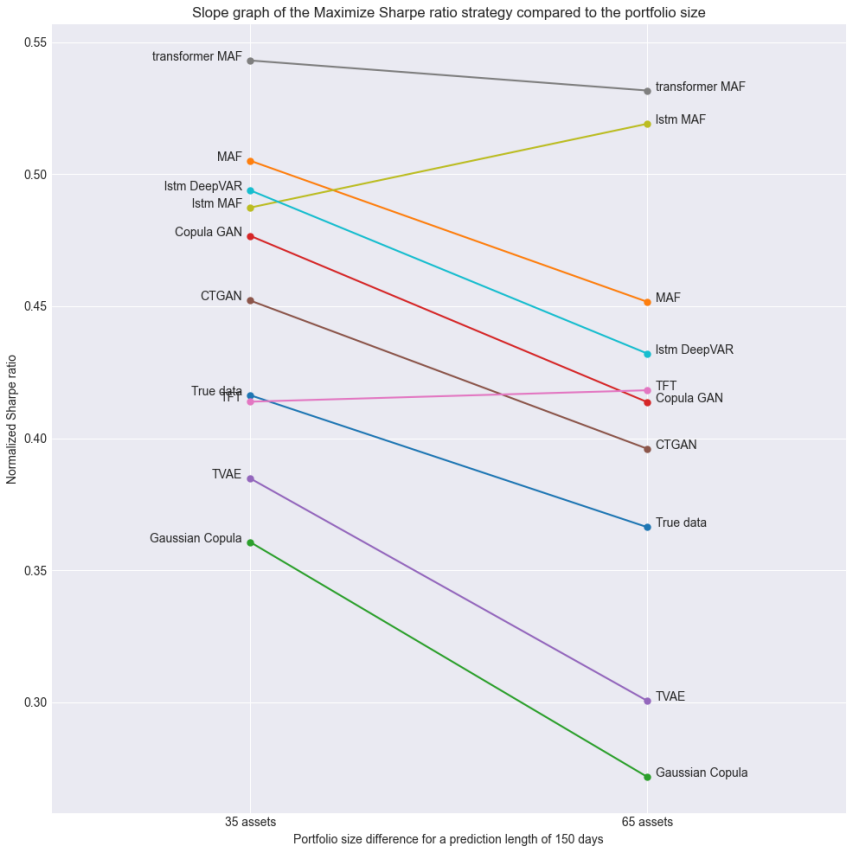

Figure 6.6: Influence of the portfolio size with respect to the normalized Sharpe ratio for estimation length of 150 days.

The results in Figure 6.6 show that if we have 65 assets in our portfolio, the results associated to a model will tend to degrade. We also have to keep in mind that all the results are normalized by the oracle (results are unattainable).
We have also evaluated the effect of the size of the portfolio when optimizing according to the Min Volatility strategy (section 2.1.3). The results (Appendix b.4) are opposite to the Sharpe results because it is a minimization in this case. If we have a larger financial portfolio, we will tend to have a higher volatility.
The same conclusions can be drawn for an evaluation period of 50 days as we can see in the Appendix b.5.

We cannot know for sure why this is, without analyzing it way more in depth. The intuition we have behind these results is that it is possible that it is more difficult for models to learn the dependencies between the assets if there are a lot of them. The problem behind this explanation is that the model with the true data, that is not augmented, also degrades.
Another explanation could be that it is linked to the randomness behind the choice of assets. As explained in 2.1.1. The assets in a dataset are drawn at random in a list with the Wilshire 5000 assets from September 2021. Because the assets are completely different between the family of datasets with 35 assets and the family of datasets with 65 assets, it could well be that in general the assets have less connections between past and future and thus the results are in general worse. The fact that we normalize by the oracle results and not the real results could also support this explanation.

6.4 Influence of the size of the past

This section is dedicated to the analysis of the length of the training set on the performances of the models. The intuition we have is that a longer training length means we have more information and thus the relationships between the assets are better evaluated. On the other hand, if the training set is too long, some relationships could be outdated, for example if there has been a reorganization of a business. In that case, parts of the training don't make sense anymore.

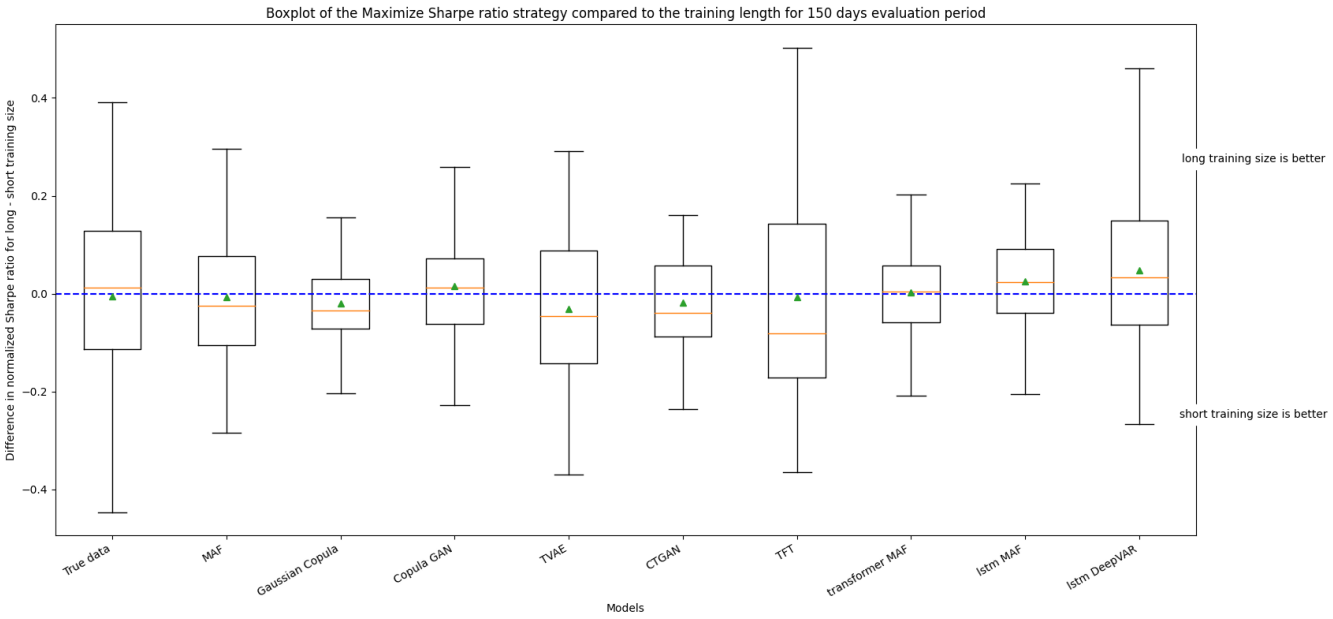

Figure 6.7: Boxplot showing the difference between the results of models trained on a training period of 2015-2020 data and 2017-2020 data. The results are the Sharpe ratio normalized by the Oracle. This boxplot is computed on 60 distinct datasets (with the same datasets when comparing both settings).

In Figure 6.7 we see the difference between the two different sizes of training data. Looking at the boxplot it is apparent that there is nearly no difference in normalized Sharpe ratio between the two. It becomes apparent when looking at the blue line that represents the threshold where there is no difference.

Figure b.7 from the Appendix shows the classical type slope that I have presented earlier. The problem with such a graph is that we don't see the standard deviation and it thus isn't really representative of the data. In addition in such a graph we only take the difference of the overall mean of the 2 settings instead of

looking at the difference for each of the 60 datasets and then looking at the average and standard deviation of all those differences. We only present those graphs when the difference is apparent, which is not the case here.
The conclusion is that we cannot infer anything for the max Sharpe strategy when changing the training size. Naturally, we would assume that at a given moment there would indeed be a clear difference (when the training length is really small), but we didn't explore it further.

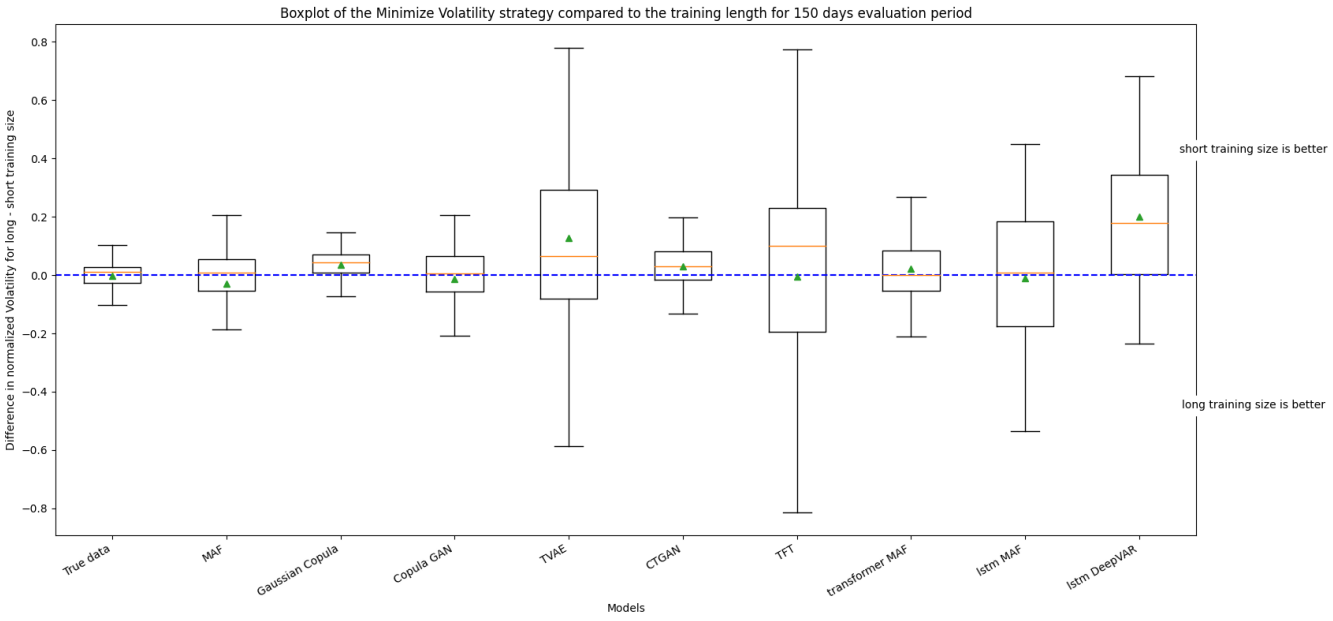

Figure 6.8: Boxplot showing the difference between the results of models trained on a training period of 2015-2020 data and 2017-2020 data. The results are the volatility normalized by the Oracle. This boxplot is computed on 60 distinct datasets (with the same datasets when comparing both settings).

Now if we look at the same graph 6.8 but based on the min volatility instead of the max sharp, we can still draw no clear conclusion for this change of setting. The range of the differences is clearly overlapping equally the 0 mark, this goes to show that in these two settings the results are very similar for the Min Volatility strategy.

6.5 Influence of the evaluation period (crash/normal)

In the previous analysis we undertook (section 6.4), we have already studied the impact of a longer evaluation period, but we have not checked the importance of the period itself. We also want to know if the period we choose for the evaluation has an impact on the performances of the models.
In order to have a clear distinction between two periods we trained on two distinct time sections:

- First, we trained the models based on training data from 2015 until 2017 and then evaluated over 2018. This is what we will call the normal period, because there wasn't any major world crash of the economy.
- The second category is trained based on training data going from 2017 until 2019 and then evaluated over 2020. Obviously in 2020 a pandemic occurred which consequently led to a major crash of the economy. This will be our Covid period in the graphs.

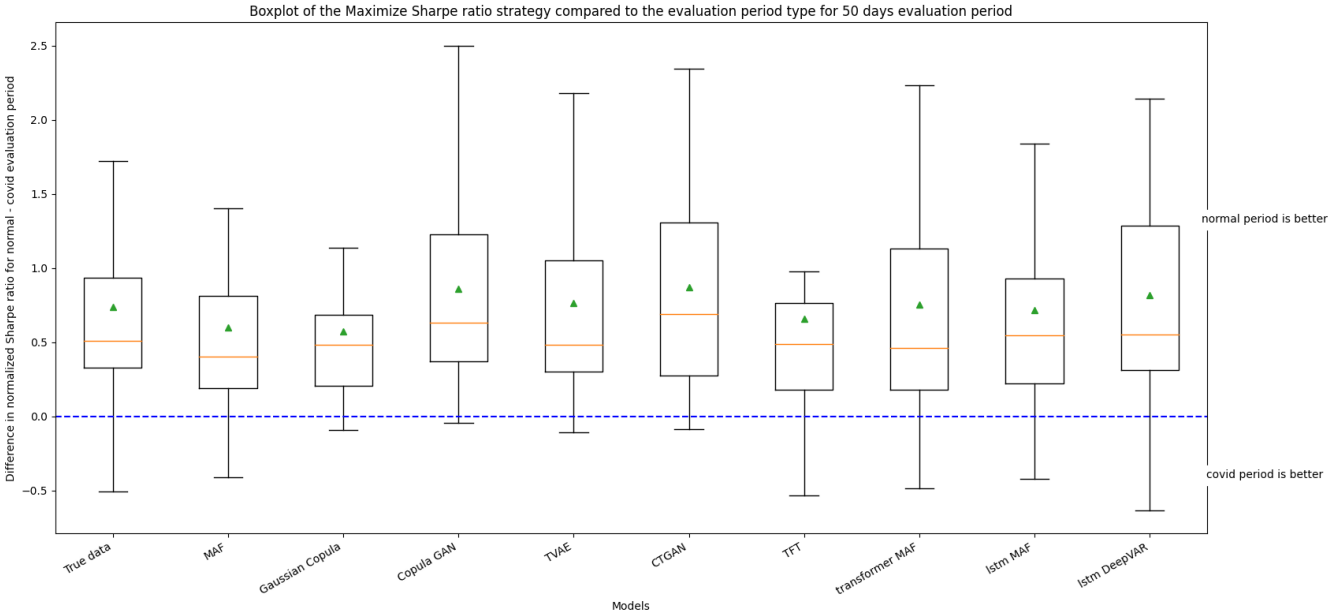

Figure 6.9: Boxplot showing the difference between the results of models trained on a training period of 2015-17(normal) data and 2017-2019(covid) data. The results are the Sharpe Ratio normalized by the Oracle. This boxplot is computed on 60 distinct datasets (with the same datasets when comparing both settings).

The first Figure 6.9 reports results when maximizing the Sharpe ratio. As we can see, these results show that the data augmentation models led to better performances during normal periods. This may be explained by the fact that, in the Covid case, the future period saw an abrupt change in the covariance structure and thus the past covariance structure (generative model) isn't valid anymore and the future trend is too difficult to be predicted (time series forecasting models). This clearly shows that there is a huge effect of the period we select for the evaluation. In Figure b.9 in the Appendix we can see more or less the same image, but this time with a slope representation. The conclusions are very clear and don't really need much explanation in such a graph. We can make the same deduction for the slope graph as the boxplot because we normalize by the Oracle data, that isn't considerably affected by the period.

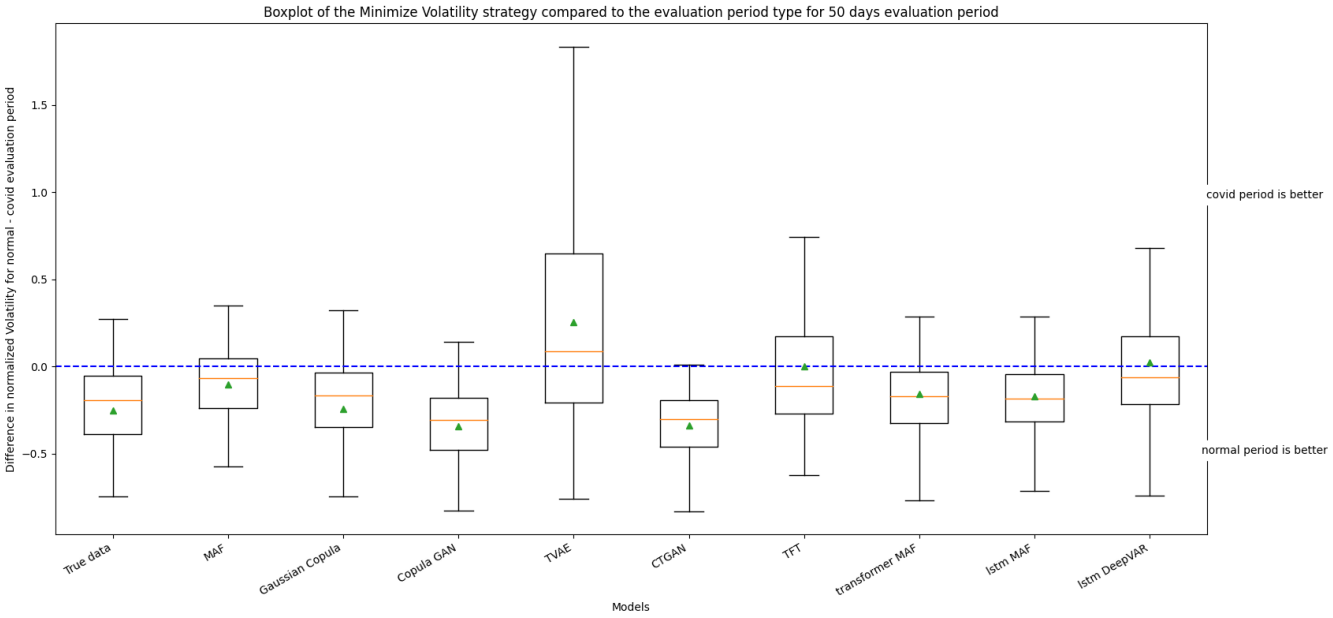

Figure 6.10: Boxplot showing the difference between the results of models trained on a training period of 2015-17(normal) data and 2017-2019(covid) data. The results are the volatility normalized by the Oracle. This boxplot is computed on 60 distinct datasets (with the same datasets when comparing both settings).

Figure 6.10 shows the same effects from the period as the previous image. When we want to compute the minimum volatility strategy, it is affected by the training period.

6.6 Comparison between different choices for the Covariance matrix

In the introduction we talked about small tweaks that we could apply to our pipeline in order to improve the overall results of our models. Here we show how changing the covariance matrix has an impact on those results. The only change we made to the code is that in the first case we evaluate the empirical covariance matrix on the synthetic data. The second case is a Ledoit-Wolf covariance matrix [11]. The third and last case is the OAS covariance matrix [12]. An explanation on those three covariance matrices has been given in the Background 2.2.

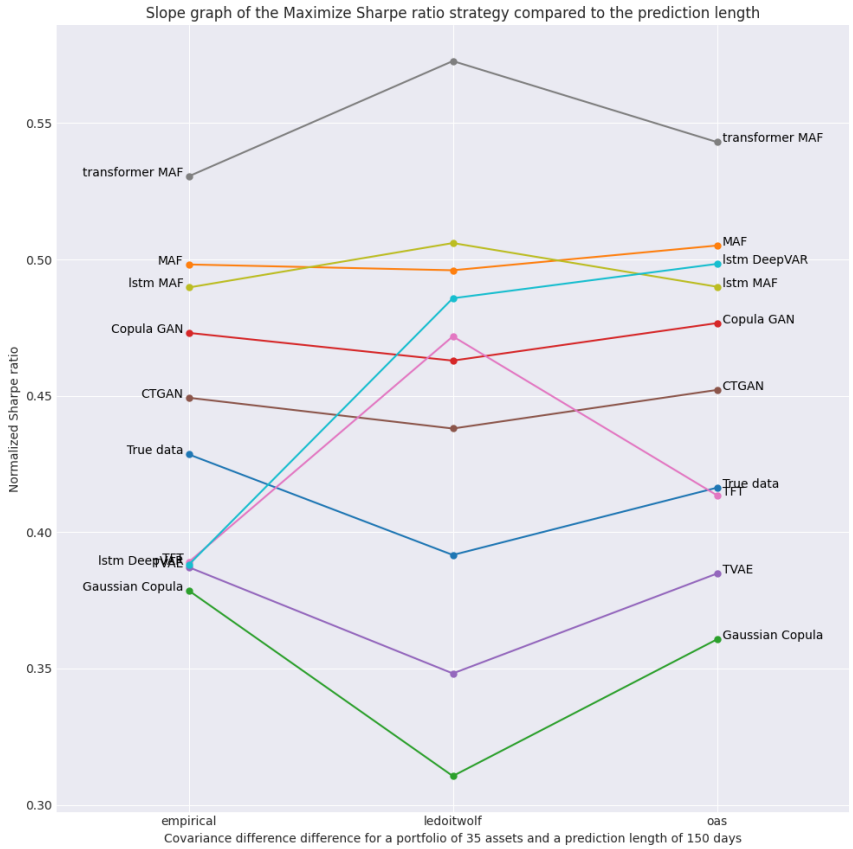

Figure 6.11: Differences between the covariance matrices with respect to the normalized Sharpe ratio. Difference was taken for datasets containing 35 assets and with a prediction length of 150 days.

In this first graph 6.11 we look at the influence of the covariance matrix on the

normalized Sharpe ratio results for the different models. Contrarily at what was explained in the paper from Ledoit-Wolf [11] we do not see a huge improvement between the different types of matrices. This could maybe be a direct result of the fact that there are very few outliers in our datasets. For the Ledoit-Wolf covariance matrix, the results are the most fluctuating. They are better for the time series forecasting models but worse for the others. The problem with the Ledoit-Wolf covariance matrix is that there is a big convergence issue. A lot of datasets results had to be dropped because it couldn't converge.
In the Appendix b.11 we show an alternative view of this comparison in a boxplot format, where the standard deviation of the result is also visible. Again, even on that graph, no clear difference is noticeable.

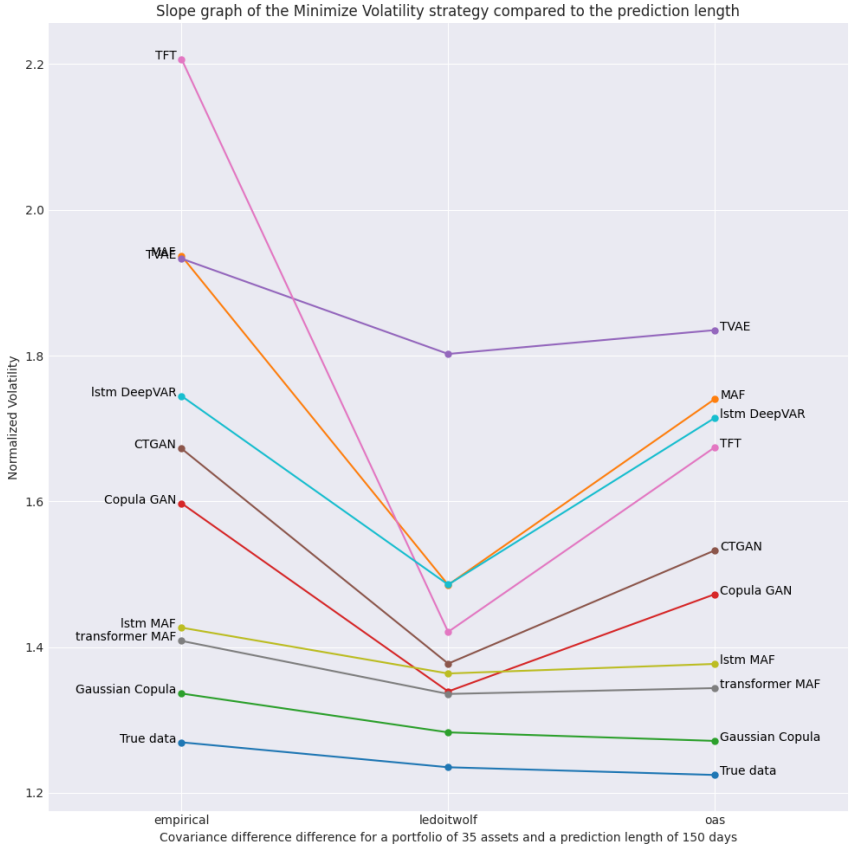

Figure 6.12: Differences between the covariance matrices with respect to the normalized Volatility. Difference was taken for datasets containing 35 assets and with a prediction length of 150 days.

In the second graph 6.12 the visualization of the difference in results for the

minimum volatility strategy are shown depending on the covariance matrix. Again, here we see clearly that Ledoit-Wolf has in general the best results and OAS is also very close to that.
If we look at the second version of the same graph in the appendix b.12 we notice the same performance difference, but we also see that it isn't as pronounced as in the slope graph because we have the standard deviation that is also perceptible.
One of the reasons Ledoit-Wolf is better performing than OAS can be explained by the fact that OAS is meant for Gaussian distributed data [15].
In our thesis we are relying solely on the OAS covariance matrix because in a lot of cases we can't use the Ledoit-Wolf estimation due to the fact that it is not converging. This never occurs for OAS on our datasets.

In conclusion, there is no significant difference between OAS and the sample covariance matrix in our case as we can see from the graphs. We also can't use Ledoit-Wolf because of its reliability problems due to the convergences issues. This shows that the impact of the choice of covariance matrix is minimal compared to other tweakable paramaters that we can find in the other sections of this chapter. We are not saying Ledoit and Wolf talk nonsense in their paper [11], what we are saying is that the difference is very negligible for our specific data.

6.7 Comparison of the performances overtime

In the previous sections, we described how the generative models performed in different use cases. We saw how the length of the evaluation period plays a big role when looking at the results (section 6.2). We also looked at the influence of the portfolio size (section 6.3) and the training length (section 6.4). Lastly and more importantly, we looked at the importance of the choice of data span for the evaluation period (section 6.5). We noticed that an evaluation period in a crisis for example, has an important degrading effect on the overall results of the models. We therefore dedicate this section to giving a big picture of how the models perform in the short term, in the long term and over time.

6.7.1 Short-term: 20 days evaluation period

This is a little summary of what will be seen in the graphs below:

- The results show the evolution of performances of generative models overtime.
- Everything is normalized by the results of the true data. The red line represents the threshold for the normalized score. When the plot lines are above the red line for the maximization, this means our models are performing better than the true data. For a minimization it is the contrary.
- The x-axis shows the evolution of the performances where the ticks represent the beginning of the evaluation period: if we have 12/'16 on the x-axis and the evaluation period is 250 days, this means that the evaluation goes from 12 december 2016 until 12 december 2017 (!!! because 250 days represent approximately 1 market year).
- Every point represents the mean of the performances for 10 datasets containing each 35 assets for a given time span.

In Figure 6.13, for the Max Sharp strategy, we see no clear improvement through time of using augmented datasets compared to using the True data. We clearly see big fluctuations in performances, but there is no clear distinction between the models in this setting.

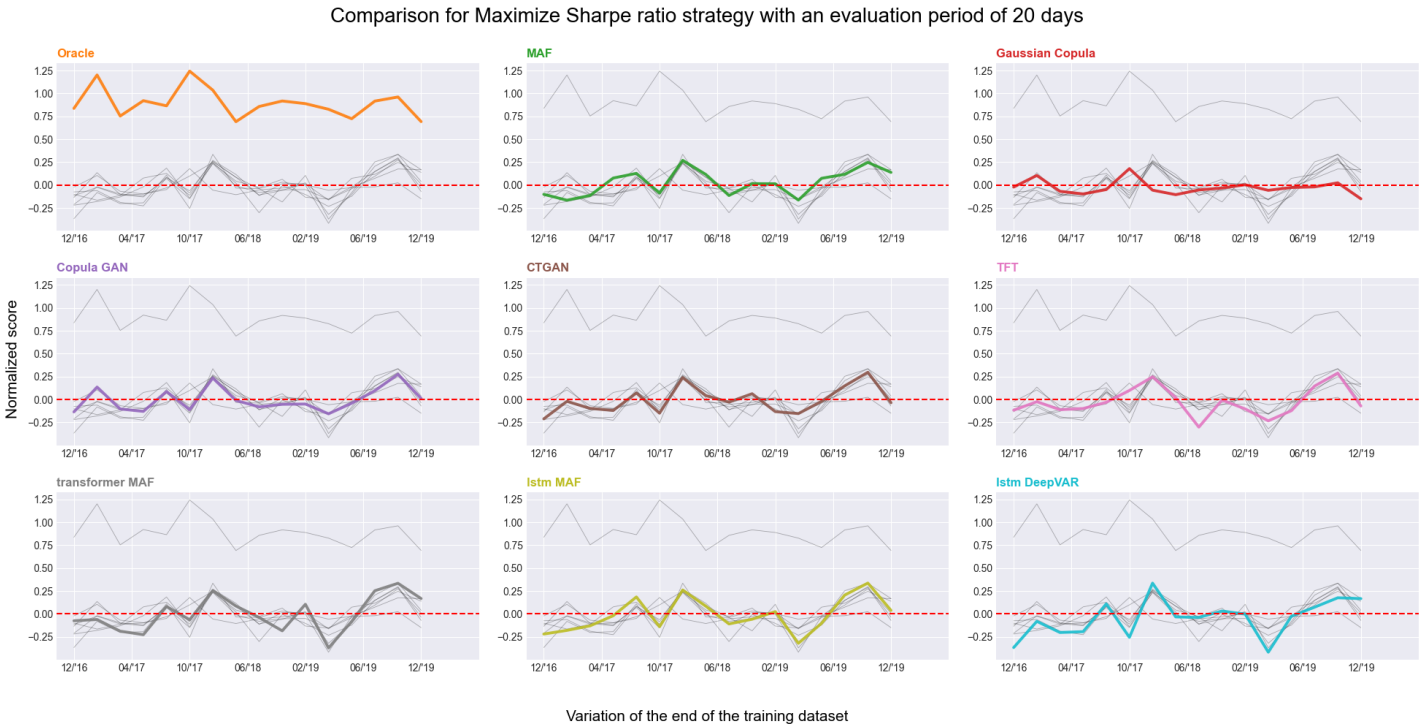

Figure 6.13: Performance difference through time for a Max Sharp optimization and a 20 days evaluation period. Higher is better.

On the contrary, we can see improvements when looking at the second graph 6.14. In this figure, we clearly see that the GAN approaches are outperforming the true data.

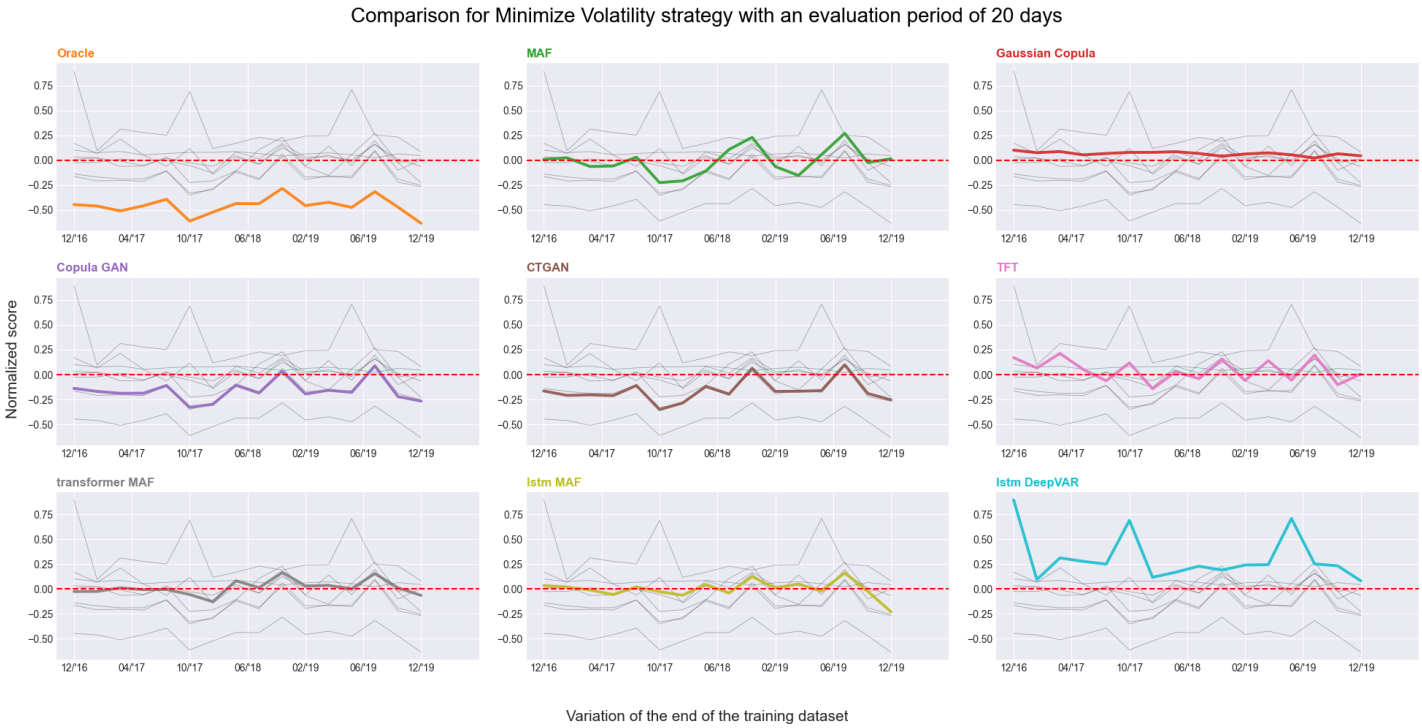

Figure 6.14: Performance difference through time for a Min Volatility optimization and a 20 days evaluation period. Lower is better.

The explanation we have behind these results is that on the short term, the data is very volatile, the evaluation is therefore very difficult, but the GAN approaches are mitigating this problem by mostly fitting the models on the last part of the training data. This way, old dependencies are discarded and it can improve the covariance estimation for the short term. In the appendix b.1, we clearly see the two GAN models distinguished from the others with mostly negative results. This means the volatility we try to minimize is better than the volatility we could get with the true data.
The reason we don't see such improvements for the Max Sharpe strategy is that the returns are playing an important role in the performances, and because the data is too volatile, it is too difficult to estimate.

6.7.2 Medium-term: 150 days evaluation period

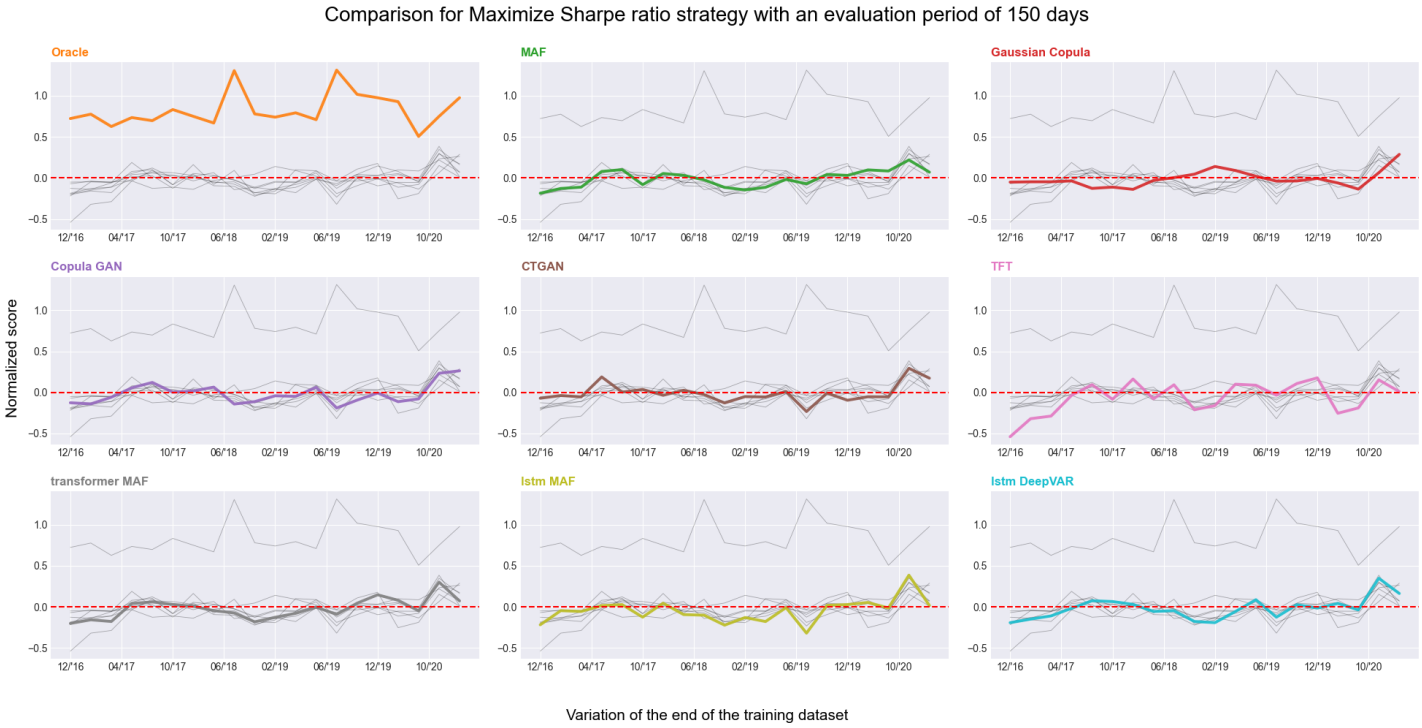

Figure 6.15: Performance difference through time for a Max Sharp optimization and a 150 days evaluation period. Higher is better.

For an evaluation period of 150 days we see absolutely no overall improvement in performances. Looking at the graph for the Max Sharpe strategy 6.15, we see the results for the augmented datasets steadily stagnating around the red threshold line.

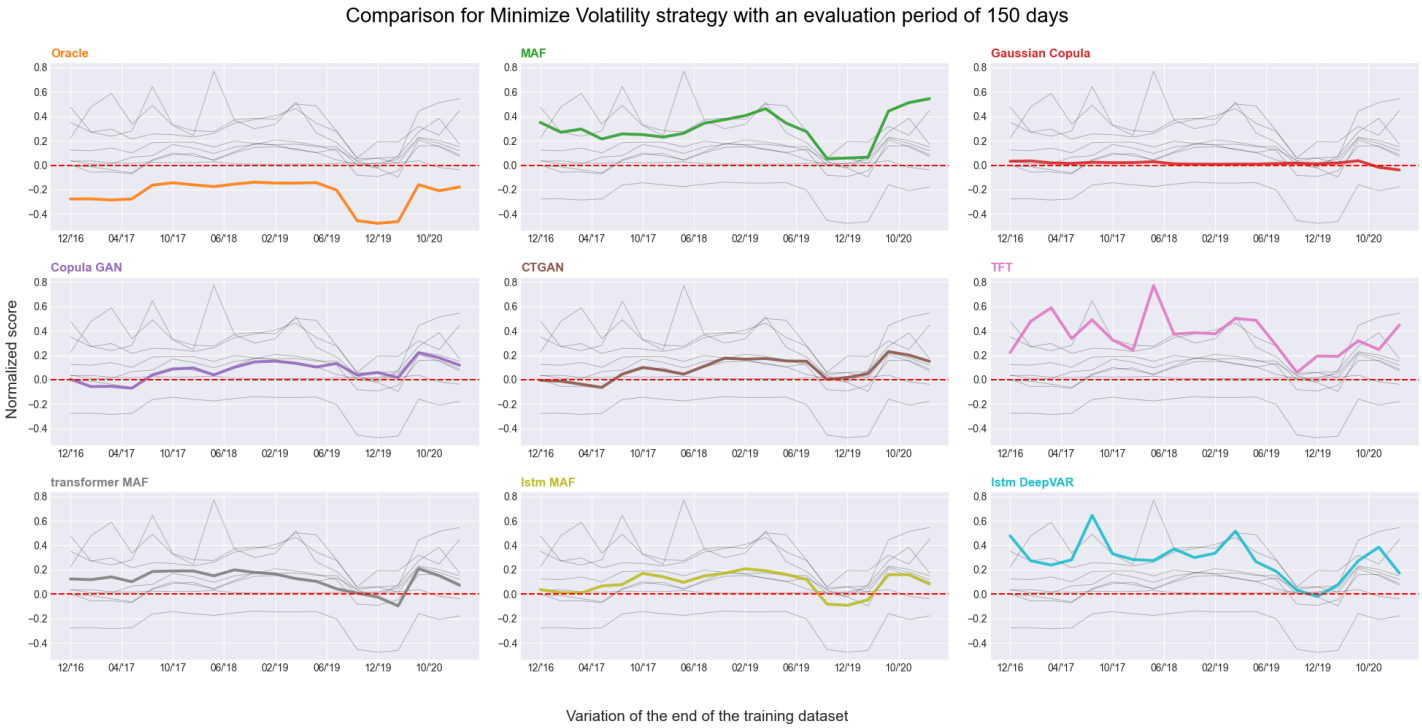

Figure 6.16: Performance difference through time for a Min Volatility optimization and a 150 days evaluation period. Lower is better.

For the Min Volatility in figure 6.16, we also notice that all the models are overall worse than the true data. In this last graph we can also see that the Gaussian Copula is the best performing of all generative models (with the same results as the true data).

6.7.3 Long-term: 250 days evaluation period

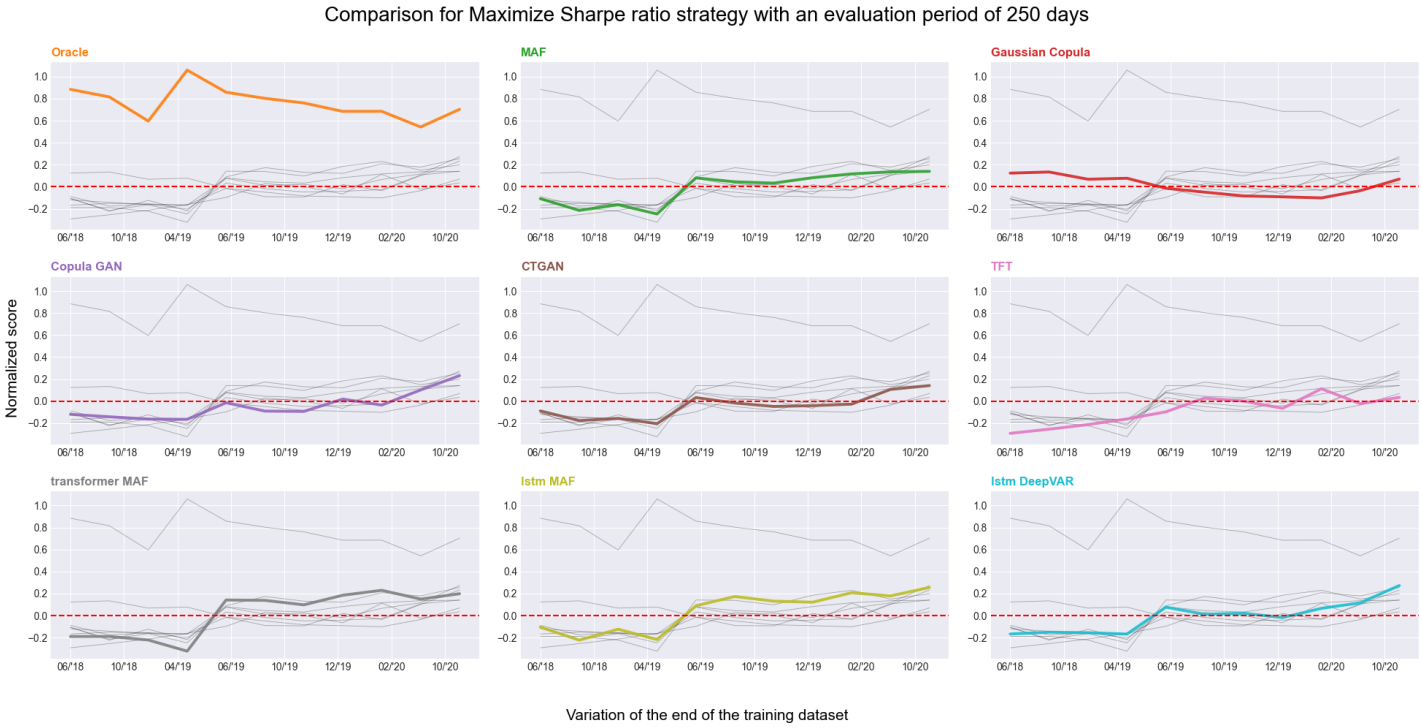

Figure 6.17: Performance difference through time for a Max Sharp optimization and a 250 days evaluation period. Higher is better.

The last analysis we are executing, is the study for the evolution of performances for long term evaluation periods.
Looking at the Figure 6.17, we can immediately see improvements with augmented datasets, and more so for time series forecasting models. The first four datapoints for all the models are not better than the true data, and this can be caused by the fact that the models didn't have enough data for the training phase. 250 days accounts for 1 whole market year, but for a reminder, we only have training data spanning from 2015. We then see a flip in results in june 2019 and from then on models are better performing.
The MAF approach is better than the true data, but slightly worse than the Transformer and LSTM MAF that is a combination of normalizing flows with time series forecasting. This can be explained by the fact that the time series forecasting approach is better at estimating the mean ROI and the future dependencies, while the MAF is needed for the covariance (dependencies) evaluation. In the appendix

b.2 we can find all the results in a table format. On that table, we clearly see higher scores for the time series forecasting models after some training time. From this table, we can thus deduce that the LSTM and Transformer MAF approach have the most potential to have high results for long term MAX Sharpe optimization.

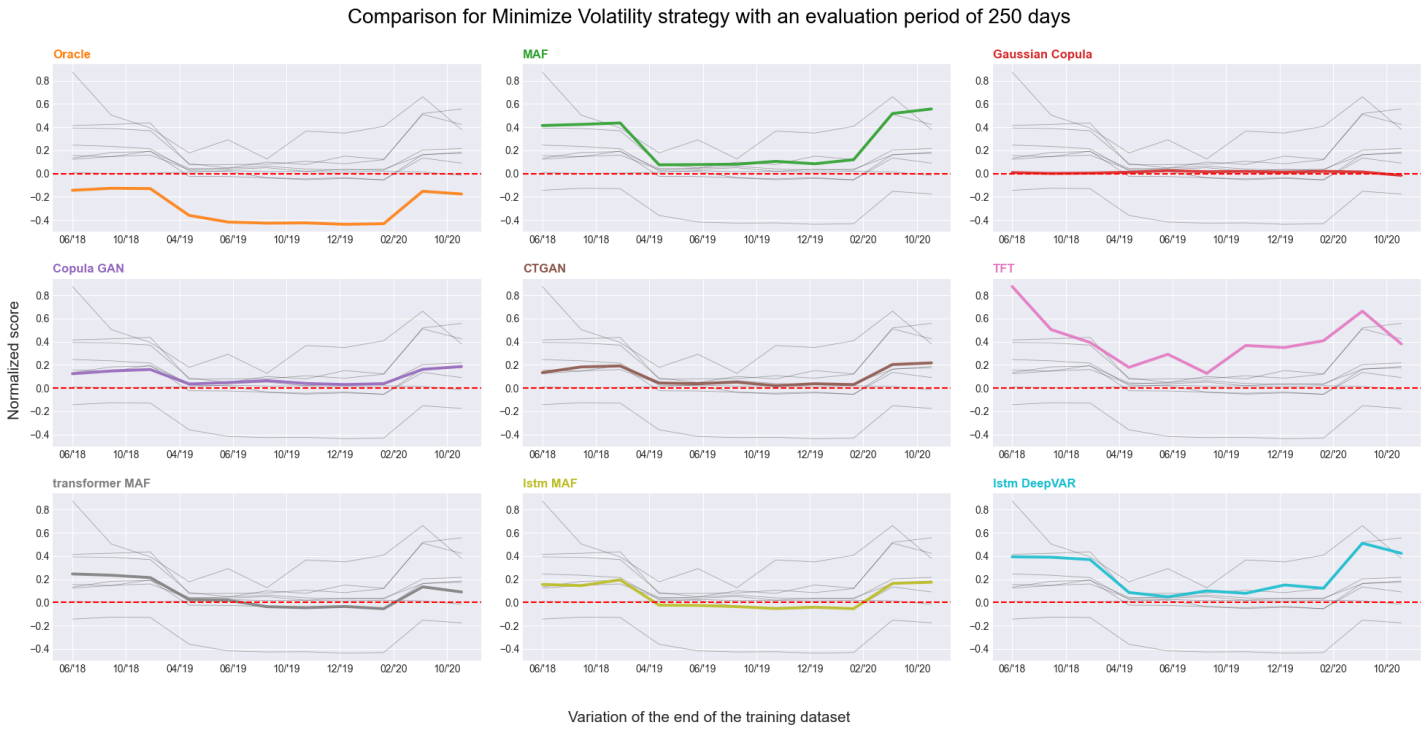

Figure 6.18: Performance difference through time for a Min Volatility optimization and a 250 days evaluation period. Lower is better.

Again, for the Min Volatility strategy in Figure 6.18, we cannot rely on any of the models since the performance are nearly always worse than the true data, except for a brief period of time for LSTM and Transformer MAF.

Chapter 7

Conclusion

This work has described that a lot of different techniques exist in order to generate synthetic tabular data that could be used for financial portfolio optimization (FPO). It has shown that time series forecasting models with normalizing flows are really promising and shows good results for some metrics and some settings.

At the beginning of the work we have introduced the different generative models we are using for the data augmentation of tabular financial datasets. We have developed why the use of such models are justified in this context.

We then talked about the way those generative models are able to augment such datasets and defined the approaches we would take in order to compare these models. We always kept a clear distinction between the models generating data based on the past joint probability distribution versus the novel approach, which generates data based on the future joint probability distribution.

7.1 Results

The next step was then to generate as much data as possible in order the visualize and analyze the behavior of the different models in various settings and in numerous different studies. We always based the studies on two different financial strategies. The first one being a strategy that wants to maximize the Sharpe ratio (section 2.1.3). The second strategy on the other hand, tries to minimize the Volatility for given assets (section 2.1.3).
The key takeaways from the studies are the following:

- **Outcome with different financial strategies (6.1):** In this study we looked at the difference in performance between the different models. The purpose was to show the potential of the models and how different components behave under certain strategies (mean Return and Volatility for the Max Sharpe strategy). We concluded that time series forecasting models

could be used to generate synthetic data improving some performances in a particular context. We also noticed that no models were able to improve the performances of the Min Volatility strategy.

- **Effect of the prediction length on the outcome (6.2):** In the particular context of the section we noticed that the prediction length had a big influence on the performances of the strategies. For both the Max Sharpe as the Min Volatility strategy, the results were better for medium-term predictions(150days) compared to short-term predictions(50 days)

- **Effect of the portfolio size on the outcome (6.3):** The portfolio size in turn, also has an influence on the performances. Larger portfolio sizes have a negative effect on the outcomes.

- **Influence of the size of the past (6.4):** In this section, we analyzed the influence of the size of the training data on the results. We saw that there was no clear distinction between the two different sizes we took into account. Our intuition is that the difference wasn't significant enough to cause changes in outcomes. A future analysis could look into shorter training sizes, but in the meantime the results don't seem too sensitive to this metric.

- **Influence of the evaluation period (6.5):** An important study we conducted was the influence of the evaluation period. In this analysis we noticed there was a huge gap in performances between models that generated data for a period during a crash (covid) and models generating data during a "normal" period. The crash led to worse performances.

- **Comparison between different choices for the Covariance matrix (6.6):** As explained in the introduction, the covariance matrix should play a role in the outcomes. We looked at the difference between Ledoit-Wolf, OAS and a sample covariance matrix (section 2.2). As we discussed, the only noticeable difference were models for the Min Volatility strategy and the Ledoit-Wolf covariance matrix that produced the best results. The problem with such a matrix is that it doesn't always converge and thus it cannot reliably be used.

- **Comparison of the performances overtime (6.7):** The last and most interesting exploration was the evolution of the results through time. We looked at the performances of models in short-term (50days), medium-term (150 days) and long-term (250days). In the short-term, GAN models over performed against the other choices for the Min Volatility strategy but no model was significantly better for the Max Sharpe strategy. In the medium-term, we didn't see any model that stood out. Finally in the long-term,

the MAF based model were clearly the best for the Max Sharpe strategy, this was after a small period of time were the models needed more data for their training. For the Min Volatility strategy, we couldn't draw any direct conclusions.

7.2 Improvements and Future works

Finally the purpose of this thesis was above all to compare the different techniques and to explain the strengths and weaknesses of all of them while analyzing their performance.

And as mentioned previously, the results we are exposing do not show all the different use cases we would encounter in all the portfolios. We are considering a particular period and a small subset of all the possible combinations of assets. This work shows the possibilities we can have with financial data augmentation, a deeper analysis would be required in order to look at way more cases.

As future work we propose to increase the number of analysis we have performed. Another possible work could be the study of the sensitivity of the results to the parameters of the model. In the Appendix a, we talk about the hyper-parameters that were used during the generation of the data. We fixed those parameters holistically. This means that tuning could have contributed to better studies, but we let this analysis to future research.

Something we also have not been able to show is the link between some properties linked to augmented datasets (the distribution of the augmented dataset, the difference between future and generated dataset, etc.) and the performances. We could for example make an in-depth analysis of the difference between the ideal covariance matrix and the matrix we obtain after data augmentation and look at its effect on the performance.

The present Thesis analyzed the performance of generative models for FPO but, according to the authors' opinion, in a limited pool of different scenarios. The present Thesis can however lay the ground for further studies, to assess the potential of generative models in the financial field.

Bibliography

[1] Harry Markowitz. "Portfolio selection". In: *The Journal of Finance.* Vol. 7, No. 1, pp. 77-91 (Mar. 1952). URL: https://www.jstor.org/stable/2975974?origin=crossref.

[2] William F. Sharpe. "The Sharpe Ratio". In: *The Journal of Portfolio Management* Vol. 21, No. 1, pp. 49-58 (Fall 1994). URL: https://jpm.pm-research.com/content/21/1/49.

[3] Ian J. Goodfellow et al. "Generative Adversarial Networks". In: *Advances in Neural Information Processing Systems, pp. 2672–2680.* June 2014. URL: https://arxiv.org/abs/1406.2661.

[4] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings.* 2014. arXiv: http://arxiv.org/abs/1312.6114v10 [stat.ML].

[5] Shakir Mohamed Danilo Jimenez Rezende. "Variational Inference with Normalizing Flows". In: *International Conference on Machine Learning.* 2015. URL: https://arxiv.org/abs/1505.05770.

[6] Sanket Kamthe, Samuel Assefa, and Marc Deisenroth. *Copula Flows for Synthetic Data Generation.* Jan. 2021. URL: https://arxiv.org/abs/2101.00598.

[7] Sam Bond-Taylor et al. *Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models.* Mar. 2021. URL: https://arxiv.org/abs/2103.04922.

[8] Arjovsky et al. *Wasserstein GAN.* 2017. URL: http://arxiv.org/abs/1701.07875.

[9] Tianci Liu and Jeffrey Regier. *An Empirical Comparison of GANs and Normalizing Flows for Density Estimation.* 2021. URL: https://arxiv.org/abs/2006.10175.

[10] Samuel Watts. *The Gaussian Copula and the Financial Crisis: A Recipe for Disaster or Cooking the Books?* June 2016. URL: http://samueldwatts.com/wp-content/uploads/2016/08/Watts-Gaussian-Copula_Financial_Crisis.pdf.

[11] Olivier Ledoit and Michael Wolf. "Honey, I Shrunk the Sample Covariance Matrix". In: *The Journal of Portfolio Management* 30.4 (2004), pp. 110–119. ISSN: 0095-4918. DOI: 10.3905/jpm.2004.110. eprint: https://jpm.pm-research.com/content/30/4/110.full.pdf. URL: https://jpm.pm-research.com/content/30/4/110.

[12] Yilun Chen et al. *Shrinkage Algorithms for MMSE Covariance Estimation*. 2009. URL: https://arxiv.org/abs/0907.4698.

[13] Kashif Rasul et al. *Multivariate Probabilistic Time Series Forecasting via Conditioned Normalizing Flows*. 2020. URL: https://arxiv.org/abs/2002.06103.

[14] *Image of the Efficient Frontier*. 2018. URL: https://pyportfolioopt.readthedocs.io/en/latest/UserGuide.html#id2.

[15] *Ledoit-Wolf vs OAS estimation*. URL: https://scikit-learn.org/stable/auto_examples/covariance/plot_lw_vs_oas.html#sphx-glr-auto-examples-covariance-plot-lw-vs-oas-py.

[16] *Nash equilibrium*. DOI: 10.1093/oi/authority.20110803100223327. URL: https://www.oxfordreference.com/view/10.1093/oi/authority.20110803100223327.

[17] Lei Xu et al. *Modeling Tabular Data using Conditional GAN*. 2019. URL: https://arxiv.org/abs/1907.00503.

[18] Lilian Weng. *Flow-Based Deep Generative Models*. 2017. URL: https://lilianweng.github.io/posts/2018-10-13-flow-models/.

[19] George Papamakarios, Theo Pavlakou, and Iain Murray. *Masked Autoregressive Flow for Density Estimation*. 2017. URL: http://arxiv.org/abs/1705.07057.

[20] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. *Density estimation using Real NVP*. 2016. URL: http://arxiv.org/abs/1605.08803.

[21] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[22] Bryan Lim et al. *Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting*. 2019. URL: https://arxiv.org/abs/1912.09363.

[23] Nikos Kafritsas. "Temporal Fusion Transformer: Time Series Forecasting with Interpretability". In: (2021). URL: https://towardsdatascience.com/temporal-fusion-transformer-googles-model-for-interpretable-time-series-forecasting-5aa17beb621.

[24] Ralf C. Staudemeyer and Eric Rothstein Morris. "Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks". In: (2019). URL: https://arxiv.org/abs/1909.09586.

[25] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2013, pp. 6645–6649. DOI: 10.1109/ICASSP.2013.6638947. URL: https://arxiv.org/abs/1303.5778.

[26] David Salinas et al. "DeepAR: Probabilistic forecasting with autoregressive recurrent networks". In: *International Journal of Forecasting* 36.3 (2020), pp. 1181–1191. DOI: 10.1016/j.ijforecast.2019. URL: https://arxiv.org/pdf/1704.04110.

[27] D. Salinas et al. "High-Dimensional Multivariate Forecasting with Low-Rank Gaussian Copula Processes". In: *International Conference on Neural Information Processing Systems*. NEURIPS'19 (2019).

[28] *Financial Optimization package for python*. URL: https://pyportfolioopt.readthedocs.io/en/latest/.

Appendix a

Code - Model metaparameters and implementation

In this section we will summarize all the important parameters that we use during the training process of the models.
The code for the master thesis is available at `https://github.com/harcorluy/Master_Thesis` [1]. The code is devided in three parts:

- Baseline : This part is dedicated to the MAF/ CMAF model and is hugely inspired by the code made available by the Euranova team
- NewT : This stand for "new technique" and as it name implies, it contains all the useful code necessary to run time series forecasting models
- General : This last part is dedicated to the generation and comparison of the two different approaches.(this also includes other baseline approaches such as copulas, tvae, GANs,...) [2]

I MAF

The implementation of this Normalizing flow technique originates from a project that had already been done by the EuraNova team. It is based on the "MaskedAutoregressiveFlow" function available by the tensorflow package. [3]

[1] Due to privacy reasons I cannot publish the code publicly, should you want access to it, you can send me a mail at : `harold.corluy@student.uclouvain.be`

[2] It contains also code to download the different asset data. Some of the code needed for the download and treatment of the data is taken from a YouTube course from Derek Banas available on his Github: `https://github.com/derekbanas/Python4Finance/blob/main/Python%20for%20Finance%201.ipynb`

[3] The documentation is available at: `https://www.tensorflow.org/probability/api_docs/python/tfp/bijectors/MaskedAutoregressiveFlow`

- Epochs : 100
- Scenarios : 32
- Periods : 500

With the scenarios the number of distinct runs we execute for each dataset. The periods are the output size of each scenario. For example, if we have a dataset with 50 assets, we would get an output matrix of size (32,50,500).

II Gaussian Copula, Copula GAN, TVAE, CT-GAN

The implementation for these models was very simple because packages already exist with everything simplified for the user. The package is called sdv and stands for synthetic data vault [4]. It is very useful for use cases where we want synthetic datasets that nearly perfectly replicate the true data.
The only important change we made for our use case is the following:

- Periods : 20 * size of the evaluation periods

III TFT, LSTM MAF, Transformer MAF, LSTM DeepVAR

These are all the models that belong to the second family of models i.e. the probabilistic time series forecasting models. The implementation is based on two important packages. The first one is gluonts [5], a package that helps with probabilistic time series modeling. The second package was created by the zalande research team, and it is called pytorchts [6]. This package is heavily based on the gluonts package but simplifies the creation of some models we are using. All the metaparameters are the default ones except for a few :

- Periods : size of the evaluation period
- Scenarios : 200
- Epochs : 50

[4]The documenation is available at : `https://sdv.dev/SDV/user_guides/single_table/index.html`

[5]The documenation is available at: `https://ts.gluon.ai/`

[6]The documenation is available at: `https://github.com/zalandoresearch/pytorch-ts`

- Batch size : 32
- Context length : 4 * size of evaluation period
- With the context length the size of the sequence we pass to the model as context for what it has to output.

Appendix b

Results

I Effect of the prediction length on the outcome

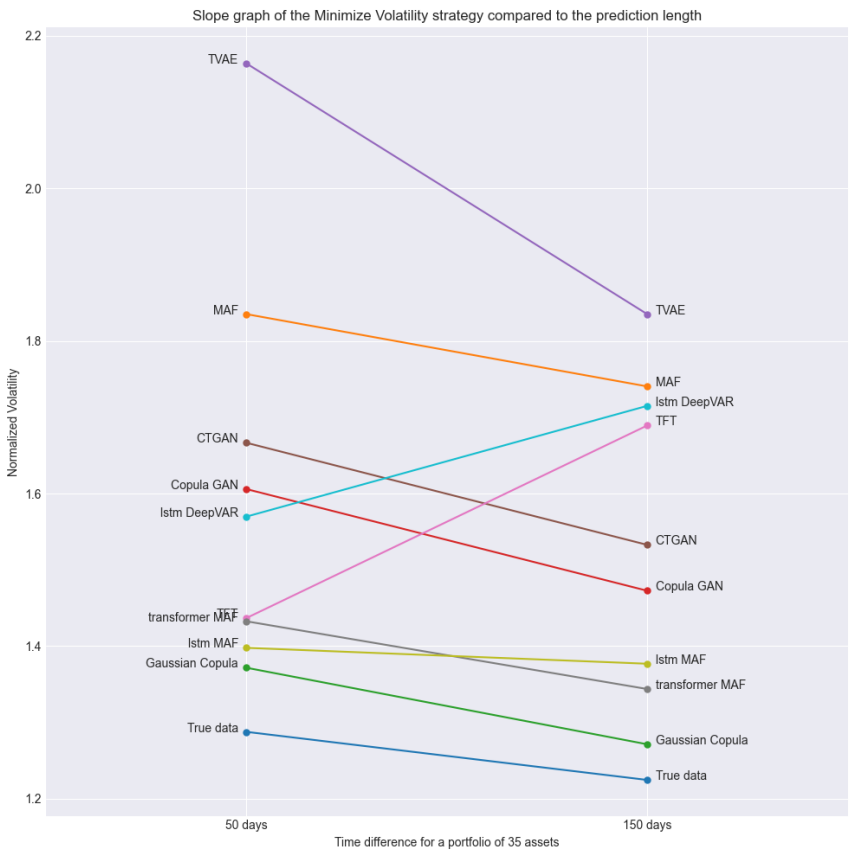

Figure b.1: Influence of the prediction length with respect to the normalized volatility

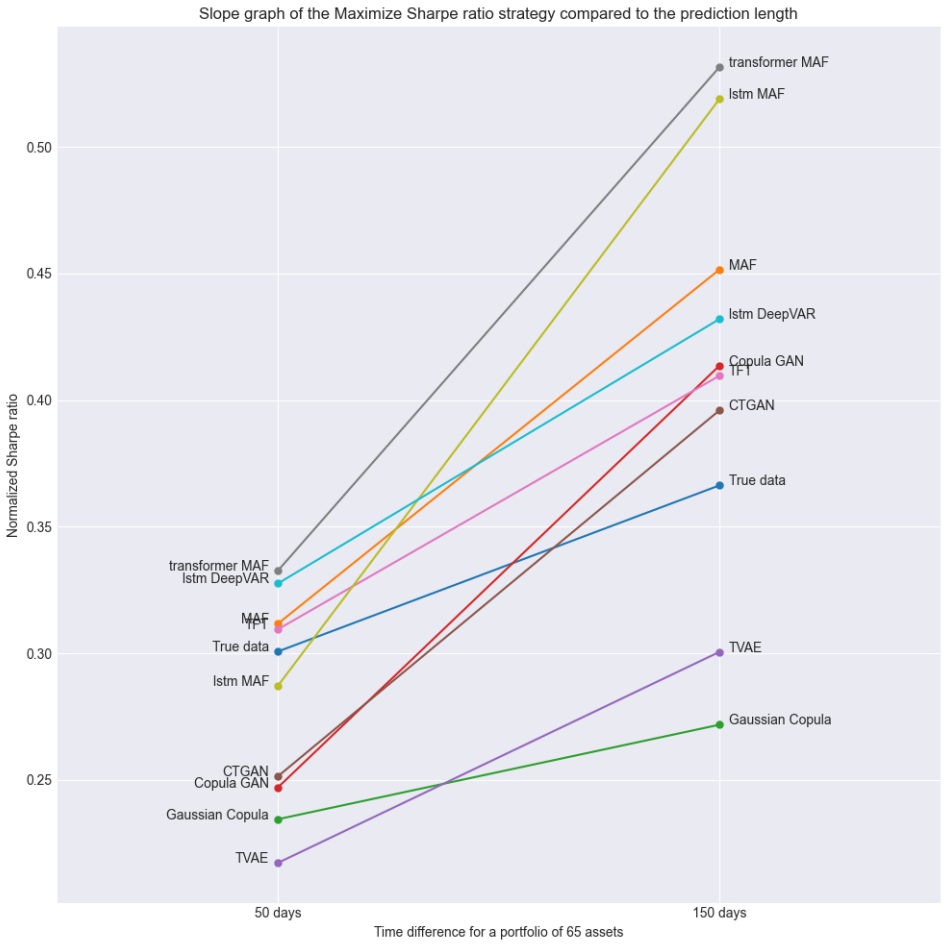

Figure b.2: Influence of the prediction length with respect to the normalized Sharpe ratio

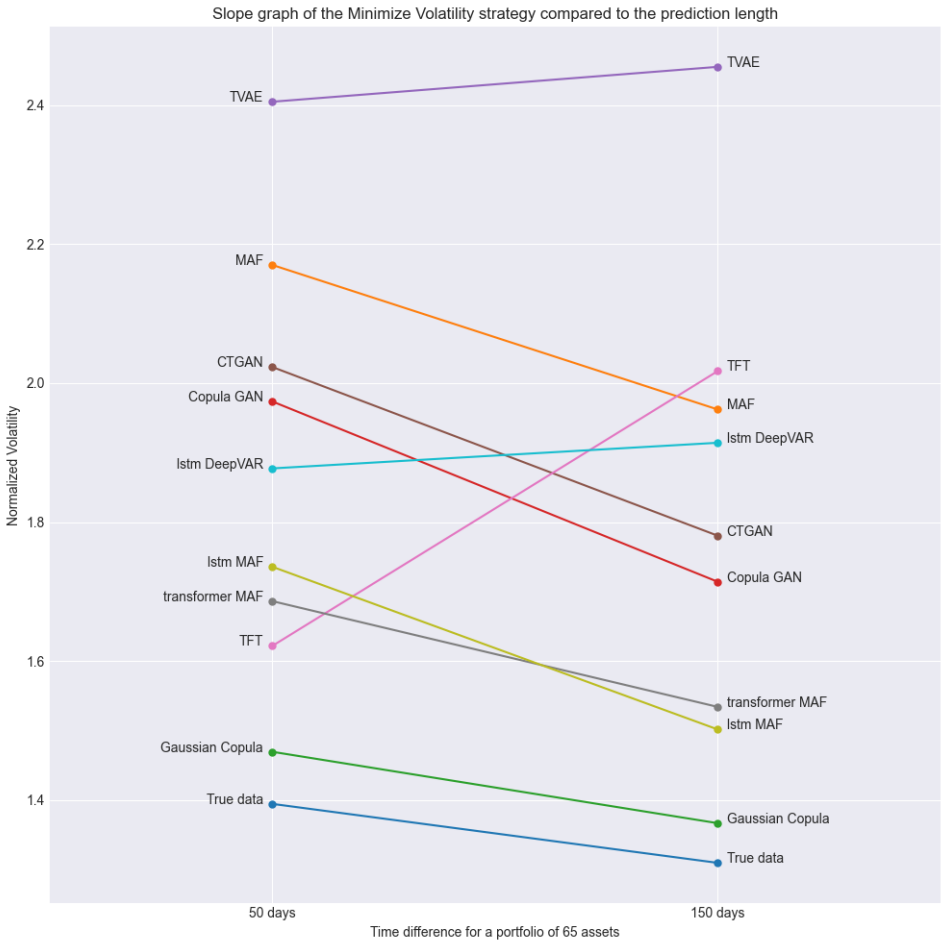

Figure b.3: Influence of the prediction length with respect to the normalized Sharpe ratio

II Effect of the portfolio size on the outcome

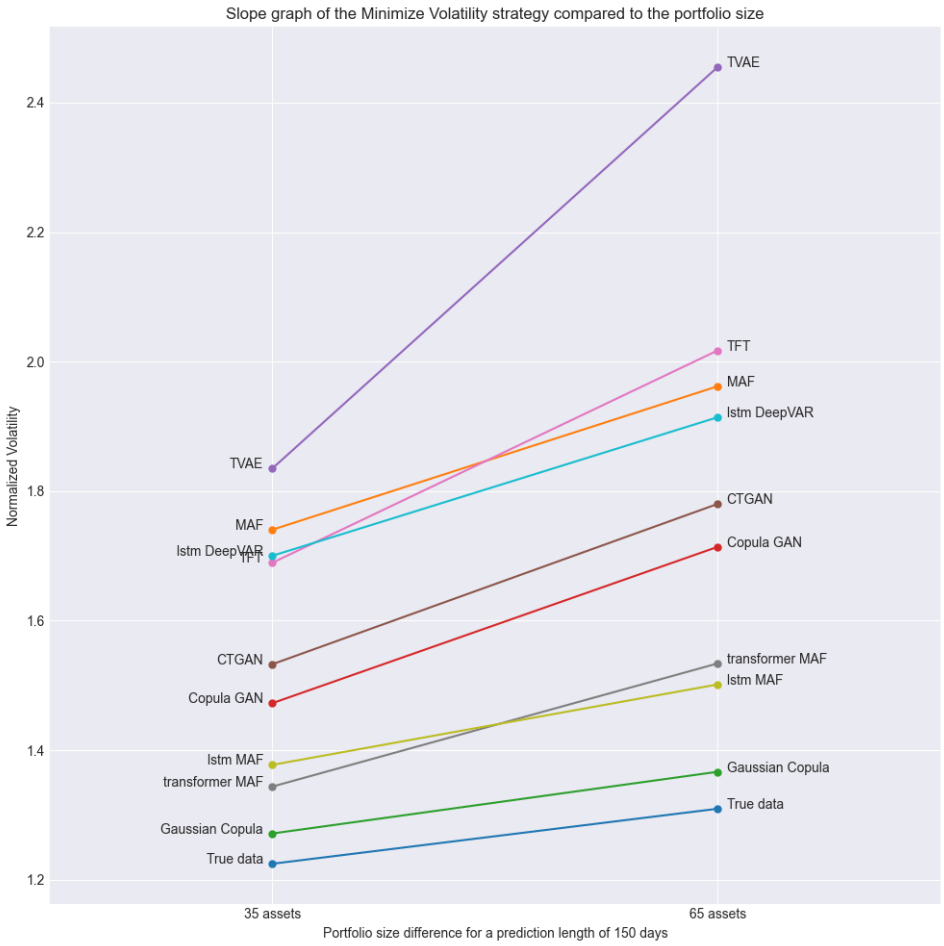

Figure b.4: Influence of the portfolio size with respect to the normalized volatility

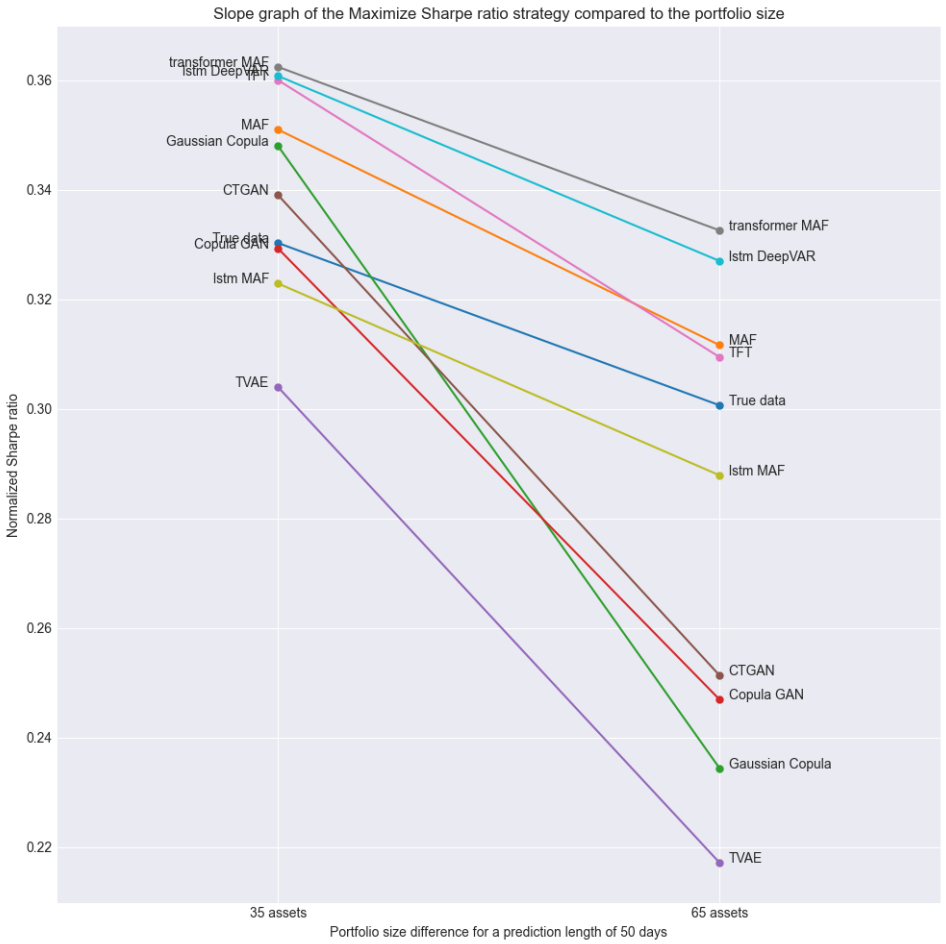

Figure b.5: Influence of the portfolio size with respect to the normalized volatility

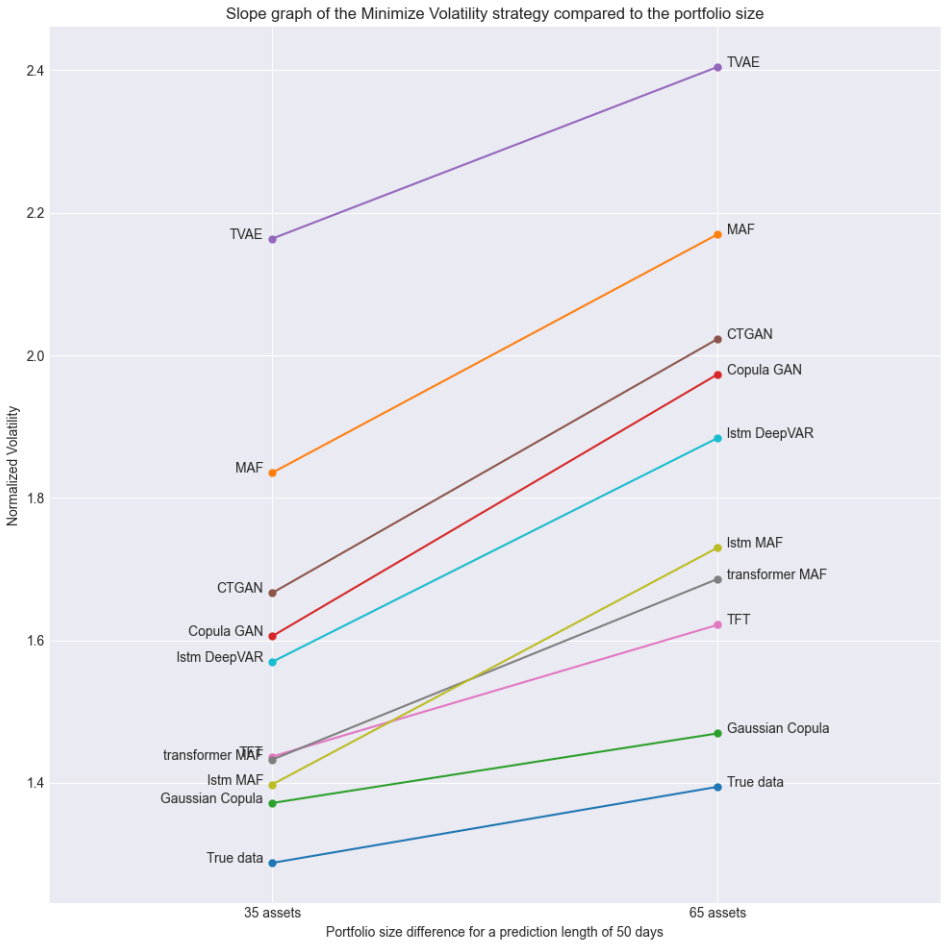

Figure b.6: Influence of the portfolio size with respect to the normalized volatility

III Influence of the size of the past

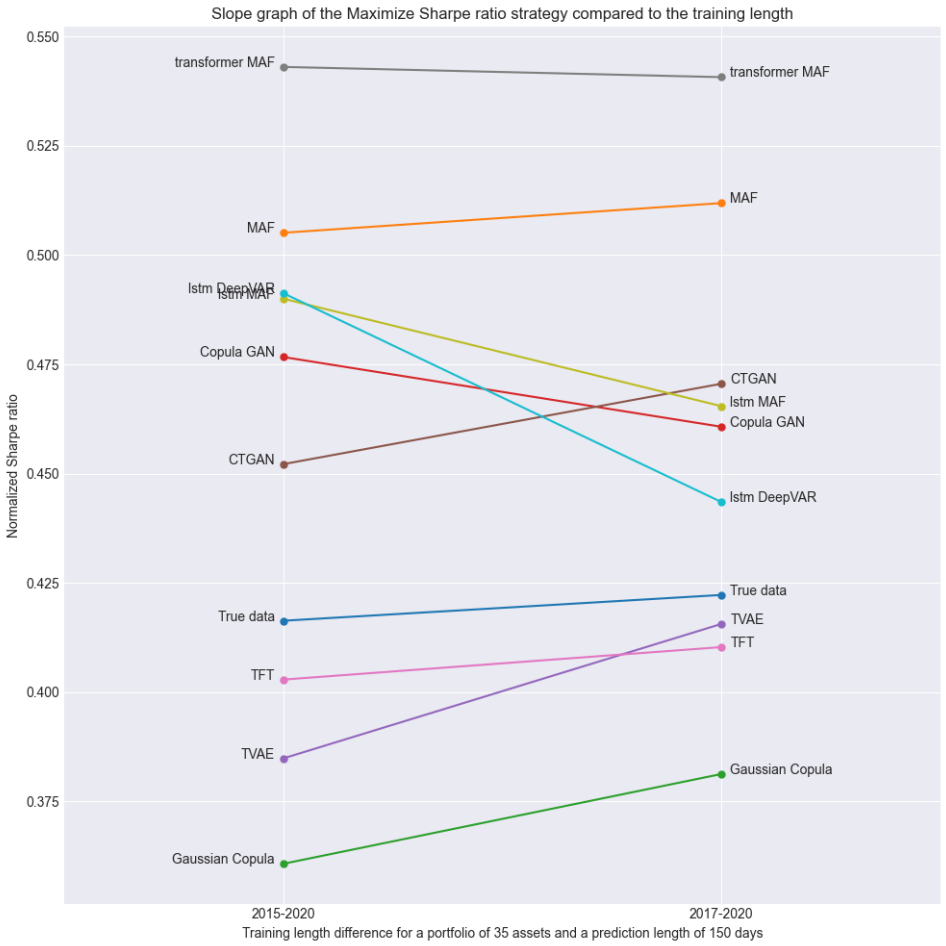

Figure b.7: Influence of the training length with respect to the normalized Sharpe ratio

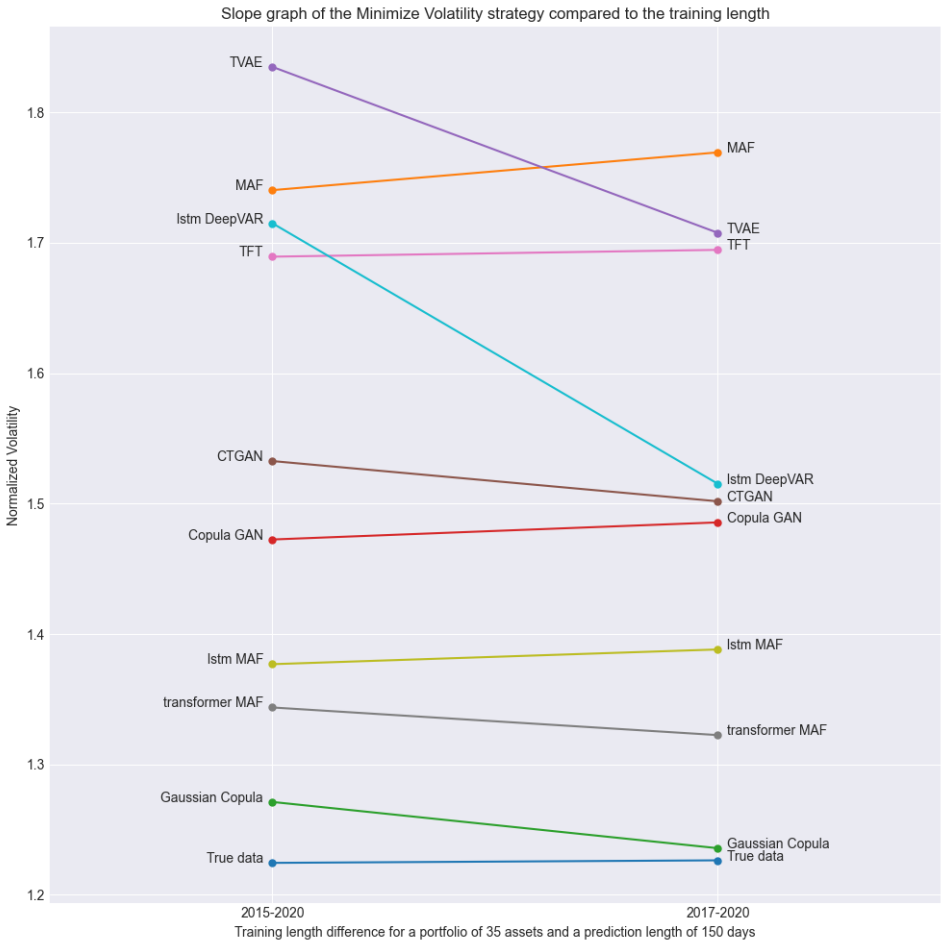

Figure b.8: Influence of the training length with respect to the normalized Volatility

IV Influence of the evaluation period (crash/normal)

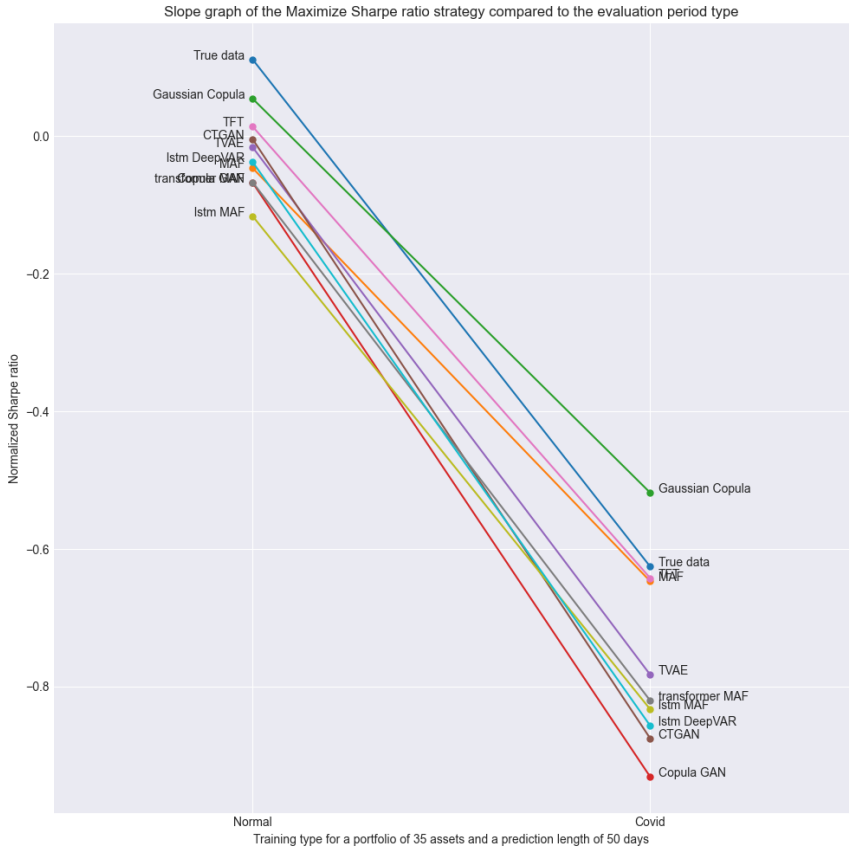

Figure b.9: Influence of the evaluation period(crash/normal) with respect to the normalized Sharpe ratio

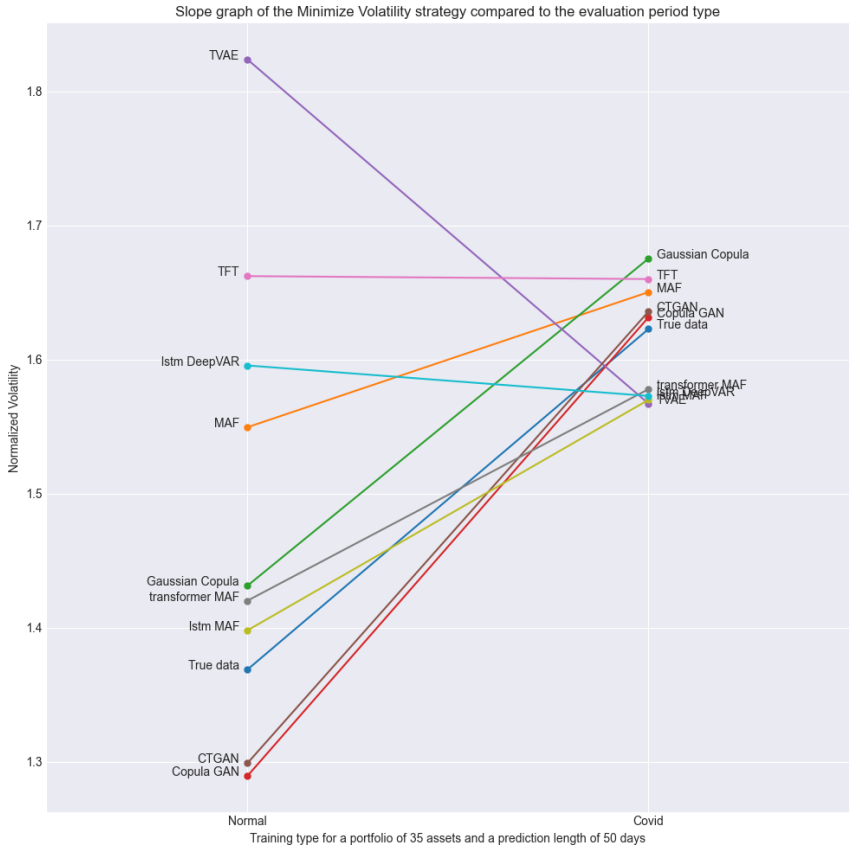

Figure b.10: Influence of the evaluation period(crash/normal) with respect to the normalized Volatility

V Comparison between different choices for the Covariance matrix

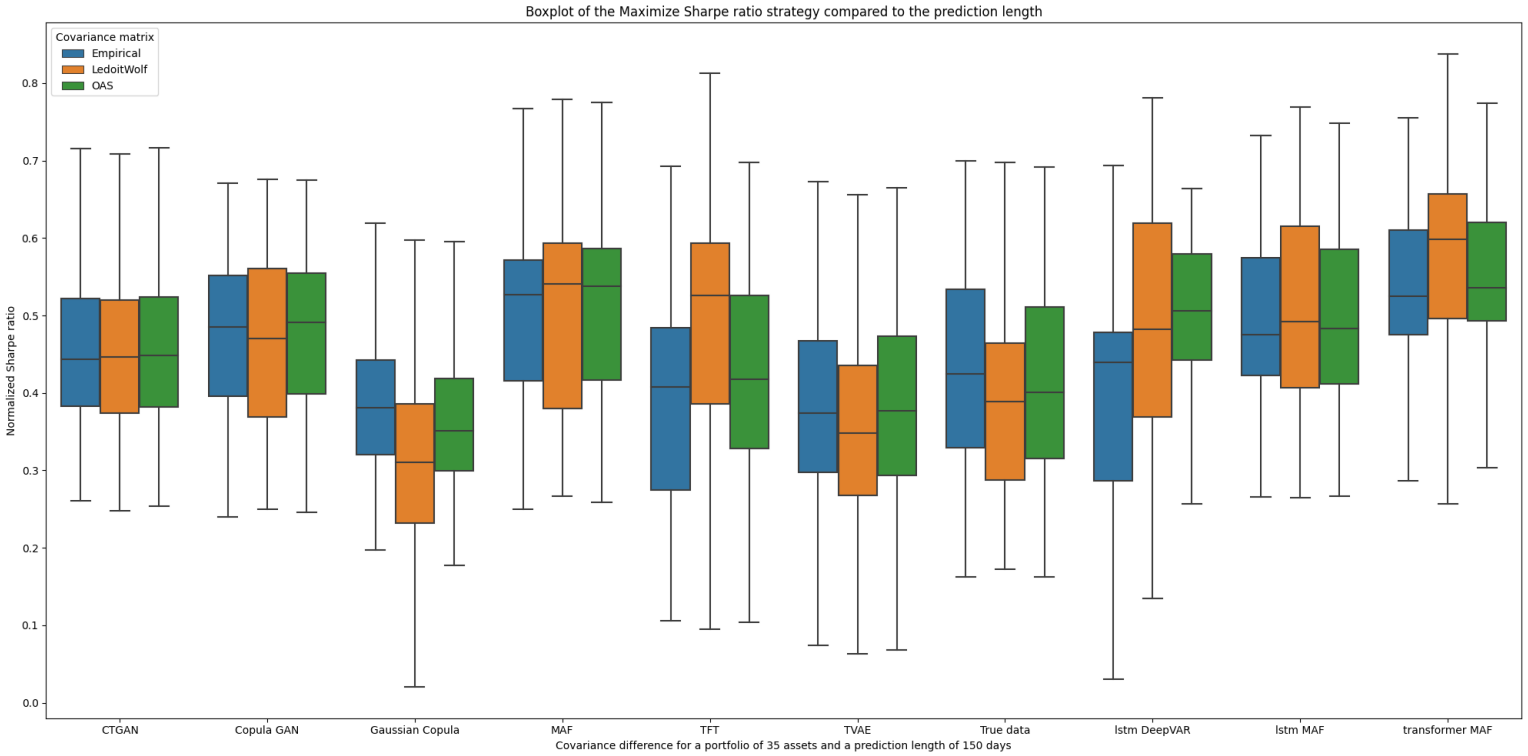

Figure b.11: Influence of the evaluation period(crash/normal) with respect to the normalized Sharpe ratio

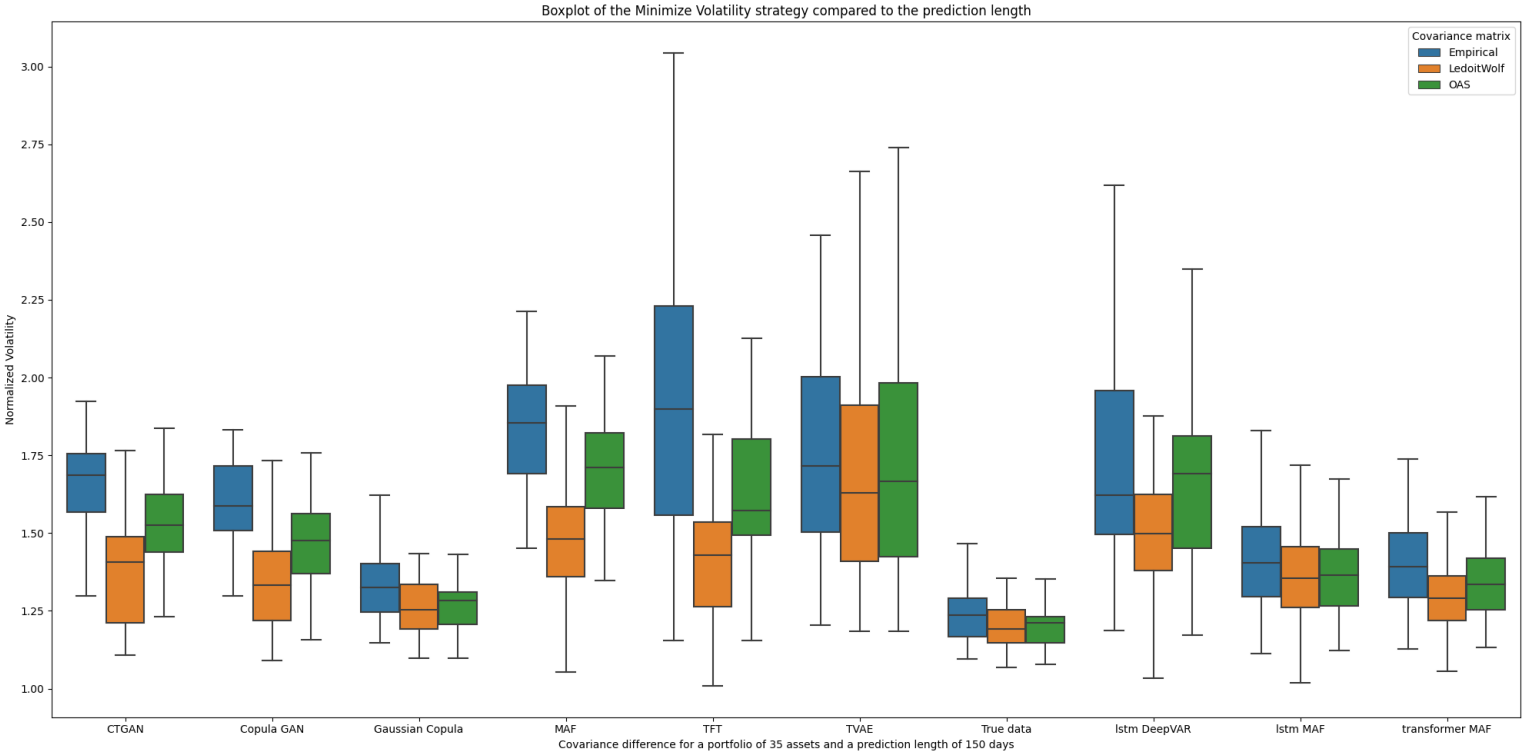

Figure b.12: Influence of the evaluation period(crash/normal) with respect to the normalized Volatility

VI Comparison of the performances overtime

* Short-term : 20 days evaluation period

Min Volatility Strategy

The following graphs are a representation of the distribution of the 10 datasets for each time span as explained in section 6.7. With the following boxplot we can understand the mean and standard deviation of the results for a given model and a given setting.

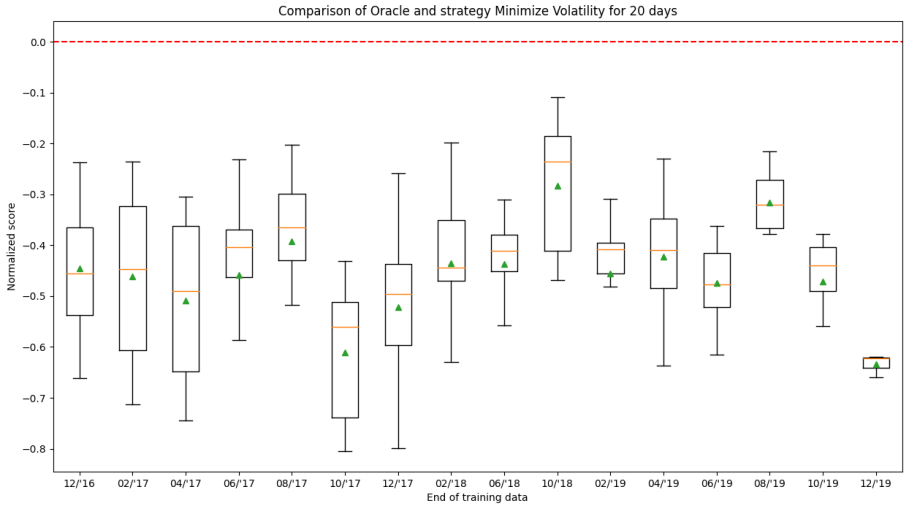

Figure b.13: Comparison of the performances of the Oracle overtime with respect to the normalized Volatility

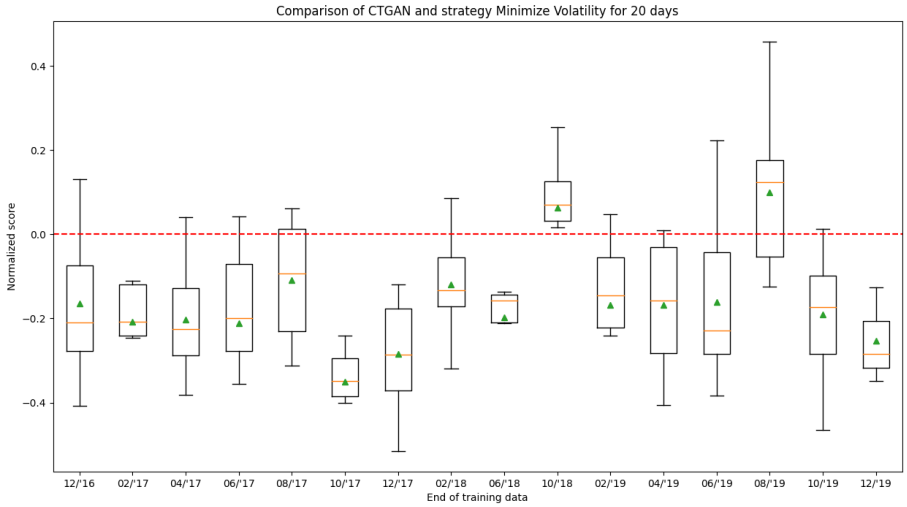

Figure b.14: Comparison of the performances of the CTGAN model overtime with respect to the normalized Volatility

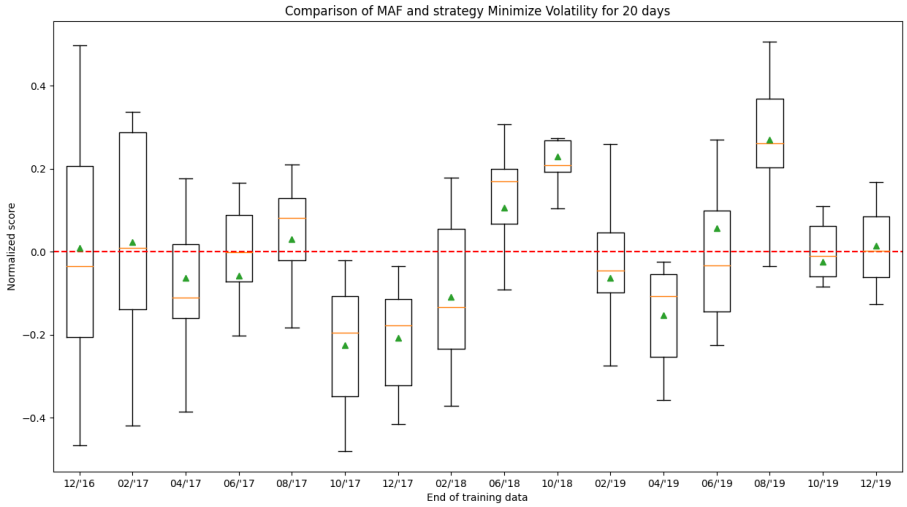

Figure b.15: Comparison of the performances of the MAF model overtime with respect to the normalized Volatility

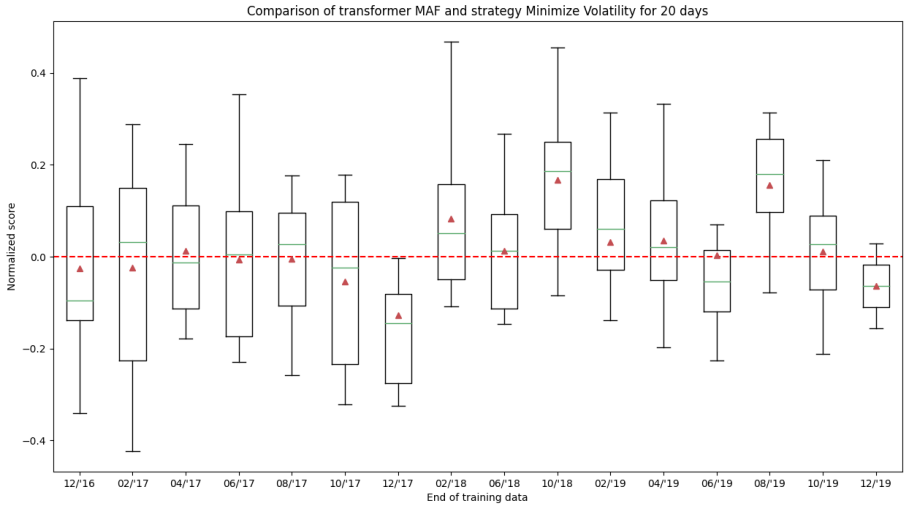

Figure b.16: Comparison of the performances of the transformer-MAF overtime with respect to the normalized Volatility

Normalized Volatility Comparison for Generative models										
Models	Oracle	MAF	Gaussian Copula	Copula GAN	TVAE	CTGAN	Transformer MAF	LSTM MAF	TFT	LSTM Deep-VAR
12/'16 - 01/'17	-0.446	0.0082	0.1006	-0.138	0.504	**-0.164**	-0.024	0.0353	0.1699	0.8943
02/'17 - 03/'17	-0.460	0.0239	0.0754	-0.168	0.337	**-0.208**	-0.023	0.0208	0.0661	0.0948
04/'17 - 05/'17	-0.509	-0.063	0.0868	-0.187	0.479	**-0.202**	0.0124	-0.009	0.2125	0.3126
06/'17 - 07/'17	-0.458	-0.058	0.0536	-0.187	0.725	**-0.211**	-0.006	-0.055	0.0524	0.2776
08/'17 - 09/'17	-0.392	0.0302	0.0671	**-0.109**	0.520	**-0.109**	-0.004	0.0212	-0.062	0.2506
10/'17 - 11/'17	-0.611	-0.225	0.0784	-0.328	0.602	**-0.350**	-0.054	-0.026	0.1180	0.6894
12/'17 - 01/'18	-0.522	-0.207	0.0786	**-0.298**	0.482	-0.284	-0.127	-0.061	-0.142	0.1176
02/'18 - 03/'18	-0.435	-0.108	0.0855	-0.104	0.337	**-0.119**	0.0828	0.0468	0.0294	0.1688
06/'18 - 07/'18	-0.436	0.1067	0.0673	-0.183	0.677	**-0.196**	0.0133	-0.039	-0.040	0.2292
10/'18 - 11/'18	-0.283	0.2299	0.0401	**0.0327**	0.452	0.0638	0.1675	0.1247	0.1516	0.1904
02/'19 - 03/'19	-0.456	-0.063	0.0622	**-0.194**	0.602	-0.168	0.0311	0.0096	-0.058	0.2410
04/'19 - 05/'19	-0.422	-0.153	0.0735	-0.156	0.375	**-0.167**	0.0352	0.0502	0.1401	0.2446
06/'19 - 07/'19	-0.474	0.0559	0.0540	**-0.178**	0.704	-0.160	0.0027	-0.020	-0.055	0.7079
08/'19 - 09/'19	-0.316	0.2694	0.0218	**0.0874**	0.250	0.0989	0.1556	0.1619	0.1966	0.2522
10/'19 - 11/'19	-0.471	-0.024	0.0643	**-0.221**	0.785	-0.190	0.0119	-0.024	-0.101	0.2329
12/'19 - 01/'20	-0.634	0.0143	0.0440	**-0.265**	0.771	-0.253	-0.063	-0.228	0.0051	0.0814

Table b.1: Comparison table for the volatility normalized with the true data for and evaluation period of 20 days

† Long-term : 250 days evaluation period

Max Sharpe Strategy

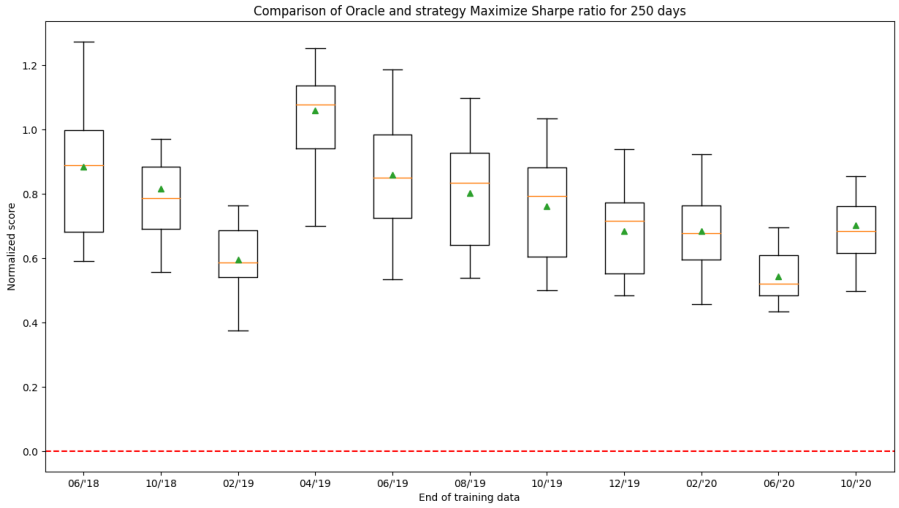

Figure b.17: Comparison of the performances of the Oracle overtime with respect to the normalized Sharpe ratio

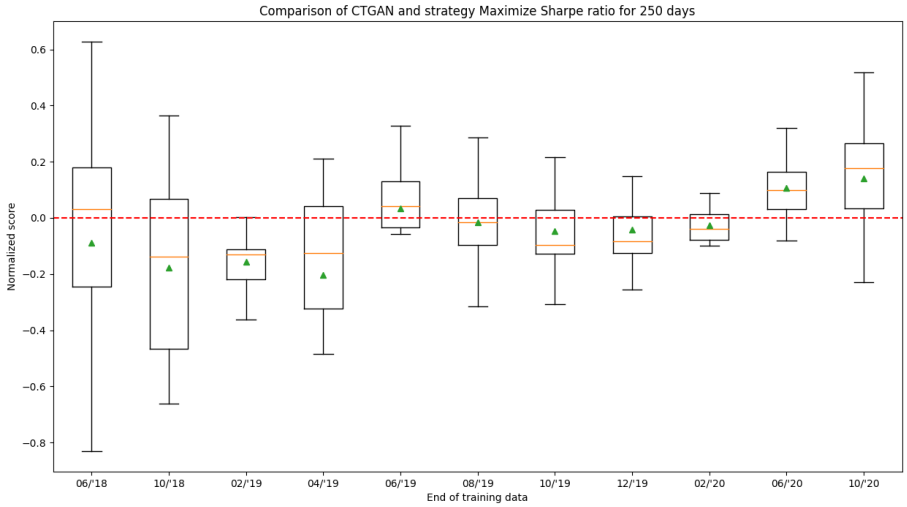

Figure b.18: Comparison of the performances of the CTGAN model overtime with respect to the normalized Sharpe ratio

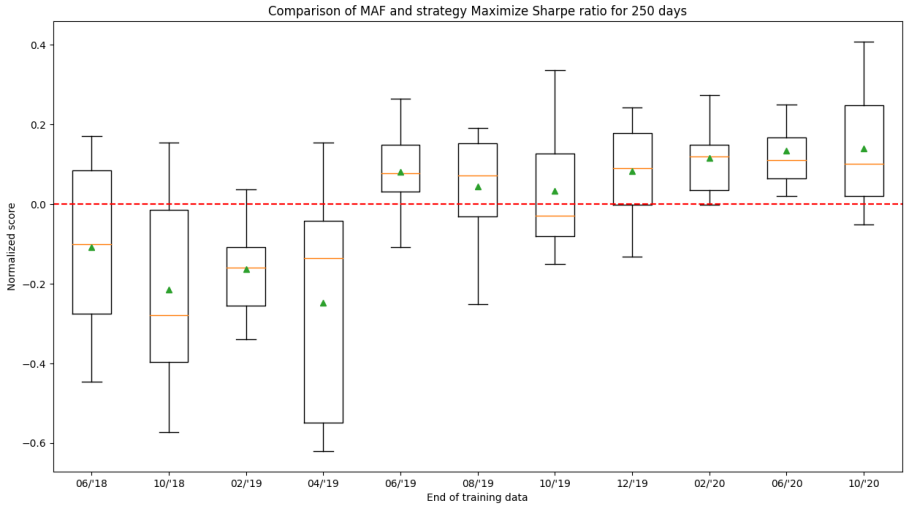

Figure b.19: Comparison of the performances of the MAF model overtime with respect to the normalized Sharpe ratio

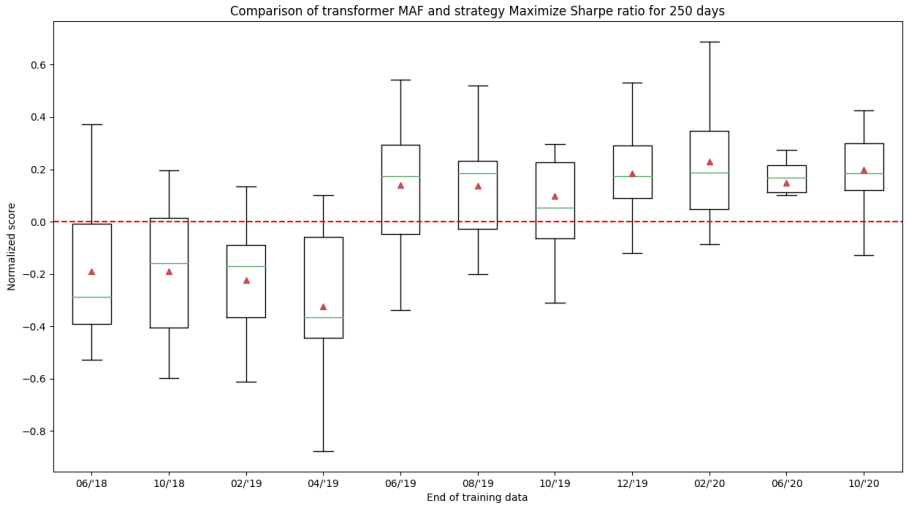

Figure b.20: Comparison of the performances of the transformer-MAF model overtime with respect to the normalized Sharpe ratio

Normalized Sharpe ratio Comparison for Generative models										
Models	Oracle	MAF	Gaussian Copula	Copula GAN	TVAE	CTGAN	Transformer MAF	LSTM MAF	TFT	LSTM Deep-VAR
12/'16 - 01/'17	0.8839	-0.108	**0.1235**	-0.119	-0.006	-0.088	-0.190	-0.104	-0.292	-0.167
02/'17 - 03/'17	0.8155	-0.214	0.1338	-0.141	**0.2472**	-0.176	-0.189	-0.223	-0.254	-0.150
04/'17 - 05/'17	0.5960	-0.163	**0.0679**	-0.162	0.0040	-0.156	-0.221	-0.123	-0.213	-0.155
06/'17 - 07/'17	1.0589	-0.247	**0.0771**	-0.165	-0.173	-0.204	-0.323	-0.218	-0.161	-0.167
08/'17 - 09/'17	0.8580	0.0803	-0.013	-0.014	-0.087	0.0326	**0.1413**	0.0893	-0.097	0.0765
10/'17 - 11/'17	0.8025	0.0444	-0.049	-0.090	-0.115	-0.016	0.1380	**0.1733**	0.0289	0.0118
12/'17 - 01/'18	0.7603	0.0328	-0.083	-0.093	-0.113	-0.048	0.0975	**0.1308**	-6.437	0.0215
02/'18 - 03/'18	0.6843	0.0821	-0.091	0.0187	-0.135	-0.040	**0.1848**	0.1216	-0.065	-0.018
06/'18 - 07/'18	0.6849	0.1162	-0.101	-0.035	-0.101	-0.026	**0.2295**	0.2077	0.1124	0.0659
10/'18 - 11/'18	0.5424	0.1351	-0.035	0.1014	-0.014	0.1068	0.1497	**0.1785**	-0.023	0.1160
02/'19 - 03/'19	0.7028	0.1395	0.0693	0.2316	0.0565	0.1413	0.1985	**0.2550**	0.0348	0.2726

Table b.2: Comparison table for the Sharpe ratio normalized with the true data for an evaluation period of 250 days

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**