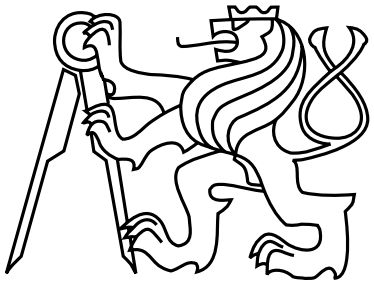
Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science and Engineering



Master's Thesis

# Roboadvisory system for automated portfolio management

*Bc. Tomáš Drtina*

Supervisor: Ing. Ondřej Vaněk, Ph.D.

Study Programme: Open Informatics, Master

Field of Study: Artificial Intelligence

January 8, 2019

# ZADÁNÍ DIPLOMOVÉ PRÁCE

## I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Drtina** Jméno: **Tomáš** Osobní číslo: **420266**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Umělá inteligence**

## II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

**Systém robotického brokera pro automatický management portfolia**

Název diplomové práce anglicky:

**Roboadvisory system for automated portfolio manamgement**

Pokyny pro vypracování:

Current trend in portfolio management is partial or full automation. This means that an intelligent algorithm advice the portfolio manager how to select instruments from the portfolio universe, it continuously monitors instruments' performance and collects risk-related information and advices on portfolio rebalancing while taking into account transaction costs and risk model chosen by the portfolio manager. This thesis should design an algorithm which should be able to rebalance a portfolio continuously and evaluate this algorithm on a set of scenarios. More specifically, the student should:
1. Understand the problem of roboadvisory, portfolio management, specifically algorithmic approaches to portfolio management
2. Design a portfolio management algorithm taking into account portfolio risk. The algorithm should be able to periodically rebalance the portfolio taking into account portfolio risk and transaction costs
3. Select an instrument universe, collect historical data and define a set of scenarios
4. Evaluate the algorithm on a constructed set of scenarios

Seznam doporučené literatury:

[1] Scalable Capital: Scalable Capital Investment Process, white paper. Online: https://uk.scalable.capital/assets/3x37a8xgm11/1ypERGd2JmUmOcsGsCe4uE/91d5745c007641f72b5b293b30c038b/WhitePaper_ScalableCapital_UK.pdf
[2] Krokhmal, P., Palmquist, J. and Uryasev, S., 2002. Portfolio optimization with conditional value-at-risk objective and constraints. Journal of risk, 4, pp.43-68.
[3] Rockafellar, R.T. and Uryasev, S., 2000. Optimization of conditional value-at-risk. Journal of risk, 2, pp.21-42.

Jméno a pracoviště vedoucí(ho) diplomové práce:

**Ing. Ondřej Vaněk, Ph.D., centrum umělé inteligence FEL**

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **20.02.2018** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce: **30.09.2019**

| Ing. Ondřej Vaněk, Ph.D.<br>podpis vedoucí(ho) práce | podpis vedoucí(ho) ústavu/katedry | prof. Ing. Pavel Ripka, CSc.<br>podpis děkana(ky) |
|---|---|---|

## III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

14/5/18

Datum převzetí zadání

Podpis studenta

# Aknowledgements

I would like to thank my supervisor Ing. Ondřej Vaněk, PhD for his guidance, advice and patience with me. I would also like to thank Mr. Radim Krejčí for his professional consultation. And last but not least I would like to thank my family for their continuous support during studies.

# Author statement for undergraduate thesis:

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, January 1, 2019 .....

# Abstract

In today's world a big trend of automation of human labor can be observed across many industries. The thesis explores an opportunity to develop a decision support tool for investment banks which manage large financial portfolios. Thus the goal is to create an algorithm that can support portfolio manager from an investment bank in his work by monitoring the stock market and providing him advices about investment opportunities and potential risks. This algorithm processes historical data for a given stock and predict the future value and whether or not is profitable to invest in such stock. Also it keeps track of the potential risk and recomputes the portfolio if the potential risk get outside the set limit. The goal was to take into account all practical constraints of portfolio management including trading fees, portfolio rebalancing and structured risk model etc. We demonstrate on a number of scenarios the added value of the algorithm and its ability to manage the portfolio with significant profits.

# Abstrakt

V dnešním světě můžeme sledovat v mnoha odvětvích velký trend automatizace lidské práce. Tato práce prozkoumává příležitosti k vývoji nástroje pro podporu rozhodování se pro investiční banky, které spravují velké množství finančních portfolií. Cilem této práce je vytvořit algoritmus, který bude pomáhat v práci správci portfolia v investiční bance tím, že za něj bude sledovat akciový trh a hledat investiční příležitosti nebo případné ohrožení investice. Tento algoritmus na základě historických dat predikuje vývoj ceny akcie a rozhoduje, zda je výhodné do dané akcie investovat. Správce zároveň má možnost si nastavit limit, jak moc je ochotný riskovat a tento algoritmu bude hlídat, zda nebyl překročen tento limit, případně přehodnotí celé portfolio. Cílem je zahrnout všechna praktická omezení včetně poplatků

za transakci, vyvažování portfolia a strukturovaný riskový profil. Demonstrujeme na několika scénářích přidanou hodnotu tohoto algoritmu a jeho schopnost spravovat portfolio s významným ziskem.

# Contents

# List of Figures

# Chapter 1

# Introduction

The current trend is portfolio management is automation using artificial intelligence. The benefit of such automation is that AI can control much larger number of stocks and can predict changes in price very fast and promptly react to them. The goal is to have AI that either automatically trades and fully manages the portfolio or is just a robo-advisor which helps the portfolio manager find investment opportunities and warn against possible losses.

The problem is that we have large number of instruments that changes its price every day and a portfolio manager who can reliably watch over limited subset of these instruments. The task is to create an algorithm that can watch over the historical data from stock market and predict which stocks will rise in its value and invest into them.

In this work we will focus on predicting the future values only from the historical data and try to find the optimal portfolio which will result in the higher investment appreciation. However, the higher the yield is usually the higher the risk is. Therefore we will introduce risk measure called Conditional value at risk (CVaR) 2.5.2 for which we can define confidence level and accepted loss which will the optimizer take in consideration when searching for the optimal portfolio.

At the end of this work we will show that with this approach we are able to achieve profits of up to tens of percent while trading only stocks of large technological companies and even higher profits when we add cryptocurrencies into our portfolio.

In the first part we will define the prediction models 2.1 and the risk model 2.5. After that we will define the optimization problem and construct a linear program describing our optimization task 4. And finally evaluate the resulting application against real-world data 6.4.

# Chapter 2

# Methodology

There are several subtasks when finding the optimal portfolio. First of them is the prediction of the price of instruments discussed in section 2.1. When knowing historical prices of given instrument we would like to predict future prices over a specified horizon. But the prediction is not enough by itself, we need to estimate the probability distribution of the future prices, in which we can use error of the model. The error of model can be estimated by using prediction model on older data and comparing the predicted value with the real values 2.2.

By using these probability distributions we can sample scenarios, how the market could be look after the specified horizon 2.4. These scenarios are key parts of the LP solver, which does the optimization portion of the problem.

## 2.1 Price prediction models

There are many ways how to predict values in timeseries from trivial models like linear regression2.1.1 to more advanced models like ARIMA 2.1.3 and triple exponential smoothing 2.1.6.

We will define timeseries $T$ which contains $n$ equally distributed points

$$T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

and function $Pred(t)$, which returns the value of the instrument after $t$ time periods after the last observation $(x_n, y_n)$.

### 2.1.1 Linear regression model

Linear regression is one of the simplest models. We are simplifying given timeseries to linear function in the form of $y = \beta x + \alpha$ which minimizes Mean squared error (MSE). The MSE is computed as average squared error of all points from timeseries:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\epsilon_i = \frac{1}{n}\sum_{i=1}^{n}[y_i - \beta x_i + \alpha]^2 \tag{2.1}$$

The optimal regression line parameters $\hat{\beta}$ and $\hat{\alpha}$ are then computed by minimizing MSE:

$$(\hat{\beta}, \hat{\alpha}) = \arg\min_{\beta,\alpha} \sum_{i=1}^{n} (\beta x_i + \alpha - y_i)^2 \tag{2.2}$$

And the result:

$$\begin{aligned} \hat{\beta} &= \frac{\bar{xy} - \bar{x}\bar{y}}{\bar{x^2} - \bar{x}^2} \\ \hat{\alpha} &= \bar{y} - \hat{\beta}\bar{x} \end{aligned} \tag{2.3}$$

where

$$\bar{xy} = \frac{1}{n} \sum_{i=1}^{N} x_i y_i \tag{2.4}$$

The resulting prediction function is then defined as $Pred(t) = \beta(n + t) + \alpha$.

### 2.1.2 Moving average model

The idea of the Moving average (MA) model is that the observation will be average of last $n$ points. MA models suffer from lagging behind. If values in the timeseries starts rising after being constant for long period the model will undervalue them and if they start falling the model will overvalue them. The more values are taken in account when computing MA model, the longer it will take for any change in trend to manifest in the model.

We will use our defined timeseries $T$ and show how to predict next value in series. The number of values used in computing MA model is $k$.

$$y_{n+1} = \frac{y_n + y_{n-1} + \cdots + y_{n-k+1}}{k} \tag{2.5}$$

### 2.1.3 ARIMA model

ARIMA stands for autoregressive integrated moving average[6]. This model combines the approach of moving average models and autoregressive models, while using integrated data, meaning we do not predict the actual value, but we predict the relative change from the last known point in time.h In this work we will be using signaflo implementation[1], which is written in Java.

### 2.1.4 Single exponential smoothing

Single exponential smoothing models have the similar idea as MA model, but instead of computing average from last $k$ points, we compute weighted moving average. The weights decrease exponentially as the we are taking older data. The basic formula for single exponential smoothing is

$$S_n = \alpha y_n + (1 - \alpha) S_{n-1} \quad 0 < \alpha \leq 1 \quad n \geq 2 \tag{2.6}$$

[1] https://github.com/signaflo/java-timeseries

where $S_n$ are smoothed values of the timeseries. This model requires to be initialized, because when we want to compute the first available smoothed value $S_2$ we need to have $S_1$, but this formula does not provide a way, how to get it. Common practice is to take the first value $y_1$ or average of first $L$ values $1/L \sum_{l=1}^{L} y_l$. We also need to set the parameter $\alpha$, which controls how much older data influence current observation. The closer $\alpha$ is to 1 the more we rely on the last observation instead of the computed smoothed values. The value of parameter $\alpha$ can be set as a constant or by computing smoothed values $(S_2, \ldots, S_n)$ for many different values of $\alpha$ and selecting the one, that minimizes MSE.

The basic formula of single exponential smoothing is recurrent and when we expand it, we can see where the exponential decrease of weights for older data comes from.

$$\begin{aligned} S_n &= \alpha y_n + (1-\alpha) S_{n-1} \\ &= \alpha y_n + (1-\alpha)\alpha y_{n-1} + (1-\alpha)^2 S_{n-2} \\ &= \alpha \sum_{i=0}^{n-2} (1-\alpha)^i y_{n-i} + (1-\alpha)^{n-2} S_1 \end{aligned} \tag{2.7}$$

With increasing $i$ (going further into past) the exponent of $1-\alpha$ gets higher and since $\alpha$ is from interval $(0, 1]$ the coefficient of $y_{n-i}$ in the sum decreases exponentially and the parameter $\alpha$ indicates the speed of the decline.

If we want to predict future values, where we do not have the real observations we cannot use the formula 2.6, because we do not know the value of $y_n$. What we can do is fixate the observation($y_n$) in the equation to the last known observation($y_{last}$) and continue in the same way as previously:

$$\begin{aligned} S_{n+1} &= \alpha y_{last} + (1-\alpha) S_n \\ S_{n+2} &= \alpha y_{last} + (1-\alpha) S_{n+1} \\ S_{n+t} &= \alpha y_{last} + (1-\alpha) S_{n+t-1} \end{aligned} \tag{2.8}$$

Suppose the $x$ values in our timeseries are subsequent days and we want to predict the value for next year ($t = 365$). We can use the previous formula 365 times or by using the same logic as in 2.7 we can expand the recurrent formula:

$$\begin{aligned} S_{n+t} &= \alpha y_{last} + (1-\alpha) S_{n+i-1} \\ S_{n+t} &= \alpha y_{last} + (1-\alpha)\alpha y_{last} + (1-\alpha)^2 S_{n+t-2} \\ S_{n+t} &= \alpha y_{last} + (1-\alpha)\alpha y_{last} + (1-\alpha)^2 \alpha y_{last} + (1-\alpha)^3 S_{n+t-3} \\ S_{n+t} &= \alpha y_{last} \sum_{i=0}^{t-1} (1-\alpha)^i + (1-\alpha)^t S_n \end{aligned} \tag{2.9}$$

The prediction function in this model is then simply $Pred(t) = S_{n+t}$. As we can see we only need to remember the last observation $y_{last}$ and its smoothed value $S_n$ to make prediction for any number of days to the future.

### 2.1.5 Double exponential smoothing

Single exponential smoothing does not provide good results on data with trend. A solution to this problem is computing of an additional parameter, which controls the trend in the

data. This formula was originally formulated by Charles C. Holt [3].

$$\begin{aligned} S_n &= \alpha y_n + (1-\alpha)(S_{n-1} + b_{n-1}) \qquad 0 < \alpha \leq 1 \quad n \geq 2 \\ b_n &= \gamma(S_n - S_{n-1}) + (1-\gamma)b_{n-1} \qquad 0 < \gamma \leq 1 \quad n \geq 2 \end{aligned} \tag{2.10}$$

Similarly to single exponential smoothing we compute $S_n$ and $b_n$ for $n \geq 3$, therefore we need to initialize $S_1$ and $b_1$ by other means. How to obtain $S_1$ was already defined in previous section. The $b_2$ value can be computing by taking first or average of first $k$ points of integrated timeseries $T' = (y_2 - y_1, y_3 - y_2, \ldots, y_n - y_{n-1})$. An other way how to get $b_1$ is to compute the difference between the first and the $k$-th point divided by the distance between the points $b_1 = \frac{y_k - y_1}{k-1}$.

Computing optimal values of $\alpha$ and $\gamma$ is, however, not as straightforward as it was in single exponential smoothing. We can still set them as constants, but if we want to get the optimal values we need to find pair $(\hat{\alpha}, \hat{\gamma})$ that minimizes MSE. This method requires non-linear optimization techniques, such as the Marquardt Algorithm [8].

Predicting values can be done using following formula

$$Pred(t) = S_n + tb_n \tag{2.11}$$

and similarly to single exponential smoothing only the $S_n$ and $b_n$ values are required for prediction, so we do not need to keep the whole timeseries for predictions.

### 2.1.6 Triple exponential smoothing

Like double exponential smoothing solves the drawback of not taking trends in data into account, triple exponential smoothing handle the seasonality in the data. We define the formulas for triple exponential smoothing as Peter B. Winters [13] known as Holt-Winter algorithm below:

$$\begin{aligned} S_n &= \alpha \frac{y_n}{I_{n-L}} + (1-\alpha)(S_{n-1} + b_{n-1}) \\ b_n &= \gamma(S_n - S_{n-1}) + (1-\gamma)b_{n-1} \\ I_n &= \beta \frac{y_n}{S_n} + (1-\beta)I_{n-L} \end{aligned} \tag{2.12}$$

where $L$ is the length of one period and $I_n$ is the seasonal parameter and the prediction formula

$$Pred(t) = (S_n + tb_n)I_{n-L+t} \tag{2.13}$$

To be able to compute $S_n$ we need to have value of $I_{n-L}$, therefore it is required for our timeseries $T$ to have at least $L$ points. The value of $I_n$ depends on value of $I_{n-L}$, so the first $L$ values of $I_n$ have to be initialized by other means. However, to properly initialize $b_1$ value using proposed formula

$$b_1 = \frac{1}{L}\left(\frac{y_{L+1} - y_1}{L} + \frac{y_{L+2} - y_2}{L} + \cdots + \frac{y_{L+L} - y_L}{L}\right) \tag{2.14}$$

we will need at least 2 full seasons ($N \geq 2L$). To initialize seasonal parameters $(I_1, \ldots, I_L)$ we will first split the timeseries $T$ into set of seasons $Season_1, Season_2, \ldots, Season_S$ where

$S$ is the number of seasons contained in the timeseries $T$. For simplicity we will use in the initialization only those seasons that are complete, in other words we will ignore the last season unless $N = kL, k \in \mathbb{N}$.

$$\begin{aligned} Season_1 &= (y_1, y_2, \ldots, y_L) \\ Season_2 &= (y_{L+1}, y_{L+2}, \ldots, y_{L+L}) \\ &\vdots \\ Season_S &= (y_{(S-1)L+1}, y_{(S-1)L+2}, \ldots, y_{SL}) \end{aligned} \tag{2.15}$$

Now we can define seasonal mean $\bar{Season}_i$ as mean of all points, that are in $Season_i$ as:

$$\bar{Season}_i = \frac{1}{L} \sum_{k=1}^{L} y_{k+(i-1)L} \tag{2.16}$$

With the previous formula and our simplification we initialize the $I_i$ values using formula

$$I_i = \frac{1}{S} \sum_{k=1}^{S} \frac{y_{i+(k-1)L}}{\bar{Season}_k} \qquad i \in [1, \ldots, L] \tag{2.17}$$

Setting the parameters $\alpha$, $\beta$ and $\gamma$ is similar to the single and double exponential smoothing. We can either set them as constants or we can find such parameters, which minimizes MSE. In contrast to previous smoothing models, where we needed only the last observation, the last smoothed value and the last trend parameter, in triple exponential smoothing model we will also need the last $L$ seasonal parameters, therefore the size of the model grows linearly with the number of data point in one period.

## 2.2 Model error estimation

Price prediction models give us only a single value for given timeseries and for the prediction date. It does not give us any information about the certainty of the predicted value or about the probability of error it made. The goal of this section is to model probability distribution from predicted values of our model and to create a function that will take our predicted value and return the predicted value with noise sampled from the error probability distribution.

### 2.2.1 Mean squared error

We will assume, that the model can underestimate and overestimate with the same probability and the error distribution is Gaussian distribution. The mean of the error distribution is zero and the only task is the estimation of variance. Let us also assume, that we have timeseries $T$ as defined in section 2.1 with $n$ points, selected model needs at least $k$ points to learn successfully, we want to predict value after $t$ time periods and we want to measure the error on at least $l$ points, then $n \geq k + t + l$ must hold.

By providing our model with only limited data $(y_1, \ldots, y_i), i \in [k, n-t]$ we can predict the value for $\hat{y_{i+t}}$ for which we also have the real value. Using this strategy we can predict values for $(y_{k+t+1}, y_{k+t+2}, \ldots, y_n)$. From computed predicted values $\hat{y}$ and the real values $y$

we can easily get squared errors and the mean of squared errors will give us the variance $\sigma^2$ that can be used in our error normal distribution.

$$\sigma^2 = \frac{1}{n-t-k+1}\sum_{i=k}^{n-t}(y_{i+t} - Pred_k(t))^2 \tag{2.18}$$

where $Pred_k(t)$ is the prediction function defined earlier on limited data $(y_1, \ldots, y_k)$. Knowing the variance we now have the probability distribution for the error on the given model for predictions of $t$ time periods. Changing the mean to value of $Pred(t)$ we have the probability distribution of the predicted value.

## 2.3 Timeseries transformation

The issue with predictors defined in the section 2.1 is that these predictors can predict arbitrary timeseries, meaning they can predict value to be negative. In case of stocks, if the company goes bankrupt the shareholders are paid the last and might not get anything, but cannot lose more than the original investment, therefore the value of the stock is always non-negative [12]. Similarly for crypto-currency the value of the currency can be as low as zero, but cannot be negative. When a predictor forecasts a value to be negative (the investor loses more than the original investment) the excepted output is much worse then what can actually happen.

To avoid this problem we introduce timeseries transformation $tr : \mathbb{R}_0^+ -> \mathbb{R}$ where $\mathbb{R}_0^+ = \{x \in \mathbb{R} \,|x \geq 0\}$, that will convert input timeseries $T$, which contains only non-negative values $x \in \mathbb{R}_0^+$, to transformed timeseries $T'$ containing real numbers $\mathbb{R}$.

$$T' = \{(x_1, tr(y_1)), (x_2, tr(y_2)), \ldots, (x_n, tr(y_n))\} \tag{2.19}$$

We will also define modified predictor $Pred'(t)$, which will predict future values based on transformed timeseries $T'$. To convert the prediction from $Pred'(t)$ we will define back transformation $tr' : \mathbb{R} -> \mathbb{R}_0^+$. Using transformation $tr$, modified predictor $Pred'(t)$ and back transformation we can create prediction that is guaranteed to be non-negative.

$$\hat{Pred}(t) = tr'(Pred'(t)) \tag{2.20}$$

Additionally we will constrain transformations $tr$ and $tr'$ by requiring the following properties:

$$tr \text{ is monotonic} \tag{2.21}$$

$$y = tr'(tr(y)) \qquad \forall y \in \mathbb{R}_0^+ \tag{2.22}$$

The property 2.22 implies that applying back transformation on transformed timeseries $T'$ will result in the original timeseries $T$. Another implication is that transformation $tr$ is injective meaning $tr(y_1) = tr(y_2) => y_1 = y_2$. Also by combining property 2.22 and property 2.21 we will get that $tr$ is not only monotonic, but strictly monotonic.

### 2.3.1 Identity transformation

Since non-negative numbers are subset of real numbers, we can set the transformation $tr$ as identity $tr(y) = y$. As a result the transformed timeseries and the original timeseries are the same $T' = T$. To ensure condition 2.22 the back transformation $tr'$ must be identity as well for all non-negative values. The negative values in prediction means, that the company went bankrupt. As mentioned before, when company goes bankrupt the payout of the stock can get as low as zero, but never under zero, so we will set the back transformation for negative values equal to zero.

Following are the transformation and back transformation formulas:

$$tr(y) = y \tag{2.23}$$

$$tr'(y) = \begin{cases} y & y \geq 0 \\ 0 & otherwise \end{cases} \tag{2.24}$$

### 2.3.2 Box-Cox power transformation

In research made by Proietti and Lütkepohl (2011) [9] was shown that applying transformation on timeseries may lead to more accurate predictions. Specifically they have used Box-Cox transformation on macro-economics timeseries. Box-Cox transformation is defined as [2]:

$$tr_\lambda(y) = \begin{cases} \frac{y^\lambda - 1}{\lambda} & \lambda \neq 0 \\ \ln y & \lambda = 0 \end{cases} \tag{2.25}$$

where ln is natural logarithm. We can easily compute the inverse functions, that will give us the back transformation:

$$tr'_\lambda(y) = \begin{cases} (1 + \lambda y)^{\frac{1}{\lambda}} & \lambda \neq 0 \\ e^y & \lambda = 0 \end{cases} \tag{2.26}$$

However we cannot use any value for parameter $\lambda$, because we have defined required properties 2.21 and 2.22. For example if we set $\lambda = 2$ then the back transformation $tr'$ is not defined for negative values, however predictors can predict negative value on transformed timeseries, therefore this parameter cannot be used.

On the other hand by setting $\lambda = 0$ we will get transformations $tr(y) = \ln y$ and $tr'(y) = e^y$. These transformations do satisfy our required properties 2.21 and 2.22, but the problem is that $tr(0)$ is not defined. This will cause issues when the original timeseries contains zeros. However, if there is zero in the input data, that means that the company went already bankrupt and we can assume that it will not recover. With this assumption in mind we can always predict the value of stock to be zero, if there is zero in the input data. Also in Proietti and Lütkepohl (2011) [9] it was shown that in most cases the optimal parameter $\lambda$ equals to zero, therefore we will use natural logarithm ln and exponential function $e^y$ as the transformation and back transformation respectively.

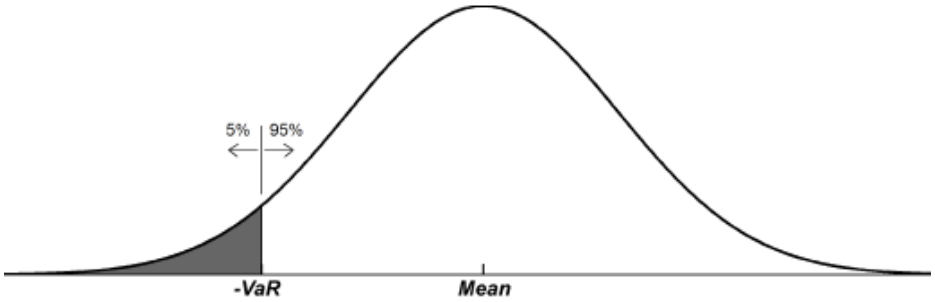



Figure 2.1: This figure show the VaR and CVaR (dark gray area) in some probability distribution chart.
Source: <http://investsolver.com/conditional-value-risk-calculator/>

## 2.4 Scenarios

Price prediction models and model error estimators provide us with the continuous distribution of the price of given instrument after $t$ time periods. If we want to invest only in one instrument this would be sufficient, but we need multiple instruments and a way how to combine them. Furthermore, we would like to replace continuous distributions with discrete scenarios so our optimization problem will be solvable using discrete optimization and continuous optimization will not be required.

### 2.4.1 Independent instruments scenarios

When generating these scenarios we are assuming that there is no correlation between instruments and that value of each instrument can be modeled independently on other instruments. For each instrument we can compute its model and error/price distribution and for each scenario just sample a value for all instruments. Creating this type of scenarios is computationally very efficient.

However, we are introducing a simplification which might lead to scenarios where similar companies or companies from the same sector will behave differently although they all rely on the same sector or some other factor. On the other hand if the companies are from the same sector and behaved similarly in the past, then the predictor should predict them behave similarly in the future as well.

## 2.5 Risk measurement

### 2.5.1 Value at risk

Value at risk (VaR) or more precisely $\alpha$-VaR or $\mathrm{VaR}_\alpha$ ($\alpha \in (0, 1)$) is a measure of risk of loss. Its value is the highest loss of given portfolio or instrument with confidence of $1 - \alpha$. As an example let $P$ be a portfolio, which at some time $t_1$ has value $v_1 = 1000$ USD and suppose we know, that $\mathrm{VaR}_{0.05}(P) = 100$ USD over the holding period T. Therefore we can say,

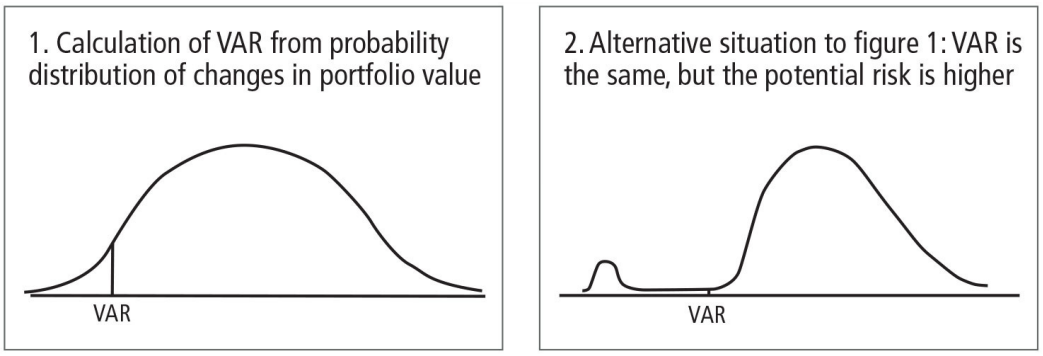



Figure 2.2: Example of two different portfolios, which have the same VaR, but potential risks are different.
Source: <https://www.risk.net/risk-magazine/technical-paper/1506669/var-versus-expected-shortfall>

that with confidence of 95% the value $v_2$ of portfolio $P$ at time $t_2 = t_1 + T$ will be greater or equal to $v_1 - \mathrm{VaR}_{0.05}(P) = 900$ USD. $\mathrm{VaR}_\alpha$ can be defined using cumulative distribution function (CDF) $F_X(x)$ [1]

$$\begin{aligned} \mathrm{VaR}_\alpha(X) &= -inf\{x \mid F_X(x) > \alpha\} \\ &= -inf\{x \mid \mathbb{P}[X \leq x] > \alpha\} \end{aligned} \tag{2.27}$$

where $X$ is random variable, which in our case is the value of single instrument or the whole portfolio. The only requirement for the portfolio or instrument is, that the CDF is known.

There are, however, also problems with VaR measure. It does not take in account anything, that has confidence lower than $\alpha$ and two portfolios which behave the same in the $1-\alpha$ confidence band, but differ otherwise, will have the same $\mathrm{VaR}_\alpha$ value even though the risk of one portfolio can be higher than the other one 2.2.

Let $P_1$ and $P_2$ be portfolios both with current value of 1000 USD. For $P_1$ we have 1% chance, that we will lose everything, 3% chance of losing 10%, 6% chance of not gaining anything and 90% chance of gaining 10%. For $P_2$ we have 4% chance, that we will lose everything, 6% chance of not gaining anything and 90% chance of gaining 13%.

$$\begin{aligned} \mathbb{P}[loss(P_1) = 1000] &= 0.01 \\ \mathbb{P}[loss(P_1) = 100] &= 0.03 \\ \mathbb{P}[loss(P_1) = 0] &= 0.06 \\ \mathbb{P}[loss(P_1) = -100] &= 0.90 \\ \mathrm{VaR}_{0.05}(P_2) &= 0 \\ \mathbb{E}(P_1) = 0.01 \cdot 0 + 0.03 \cdot 900 + 0.06 \cdot 1000 + 0.90 \cdot 1100 &= 1077 \end{aligned} \tag{2.28}$$

$$\begin{aligned} \mathbb{P}[loss(P_2) = 1000] &= 0.04 \\ \mathbb{P}[loss(P_2) = 0] &= 0.06 \\ \mathbb{P}[loss(P_2) = -130] &= 0.90 \\ \mathrm{VaR}_{0.05}(P_2) &= 0 \\ \mathbb{E}(P_2) = 0.04 \cdot 0 + 0.06 \cdot 1000 + 0.90 \cdot 1130 &= 1077 \end{aligned} \tag{2.29}$$

In terms of expected value and $\text{VaR}_{0.05}$ the portfolios behave the same, but chance of losing everything is four-times higher in $P_2$ while the maximal gain increases from 10% to only 13%.

In Artzner et al. (1999) [1] they present four desired properties for measures of risk. One of them is subadditivity. Some function $f$ has the property of subadditivity if the following holds for all $x$ and $y$: $f(x+y) \leq f(x) + f(y)$. This property is desired for the risk measure, because if it does not hold, we can split portfolio into smaller chunks and sum of risks of these chunks will be smaller than the risk of the original portfolio. Unfortunately VaR does not have this property which can be shown on their example.

Let us assume, that we have two possibilities how to invest. In the first one $P_1$ we need to invest 1000 USD and if the price of some instrument $I$ will be less than $U$ will will get 1100 USD otherwise nothing. The second one $P_2$ is similar to the first one, but we will get 1100 USD only if the price of $I$ will be more than $L$ ($L < U$) otherwise we lose our initial investment. The probabilities of losing are in both cases 4% meaning $\mathbb{P}[price(I) \geq U] = \mathbb{P}[price(I) \leq L] = 0.04$. The 5% value at risk is for both cases $-100$ since the probability of gaining is 96%.

The problem occurs, when we invest in both $P_1$ and $P_2$ at the same time. We cannot lose on both investments, because condition for losing are mutually exclusive, the chance of losing on one of the investments is now 8%. The 5% value at risk for combining $P_1$ and $P_2$ is 1000.

$$\begin{aligned} \text{VaR}_{0.05}(P_1 + P_2) &\nleq \text{VaR}_{0.05}(P_1) + \text{VaR}_{0.05}(P_2) \\ 1000 &\nleq -100 + (-100) \end{aligned} \tag{2.30}$$

The required condition for subadditivity does not hold, therefore VaR does not have the subadditivity property.

### 2.5.2 Conditional value at risk

Conditional value at risk (CVaR), also known as expected shortfall or expected tail loss, tries to solve the drawbacks in VaR risk measure. Before we formally define CVaR let us define loss function $f(P, y)$, where $P$ is selected portfolio and $y \in R^m$ random vector of prices of individual items in the portfolio (assuming there are $m$ items) with known probability density function $\mathbb{P}[y]$. CVaR is then defined as mean loss, where loss is higher or equal to $\text{VaR}_\alpha$ [10]. Note that in the cited article $\alpha$ is defined differently.

$$\text{CVaR}_\alpha(P) = \alpha^{-1} \int_{f(P,y) \geq \text{VaR}_\alpha(P)} f(P, y)\mathbb{P}[y]dy \tag{2.31}$$

If we know the value of $\text{VaR}_\alpha$(P) beforehand, we can create new function $F_\alpha(P, \beta)$, where $\beta$ is the $\text{VaR}_\alpha(P)$ computed. Additionally we can subtract $\beta$ from the loss function inside the integral and add it back outside without changing the result.

$$\begin{aligned} F_\alpha(P, \beta) &= \alpha^{-1} \int_{f(P,y) \geq \beta} f(P, y)\mathbb{P}[y]dy \\ &= \beta + \alpha^{-1} \int_{f(P,y) \geq \beta} (f(P, y) - \beta)\mathbb{P}[y]dy \end{aligned} \tag{2.32}$$

We can notice, that $f(P,y) - \beta \geq 0$ if the condition in integral is met and $f(P,y) - \beta < 0$ if not. If replace negative values of $f(P,y) - \beta$ with zero, we no longer need the condition.

$$F_\alpha(P,\beta) = \beta + \alpha^{-1} \int_{y \in \mathbb{R}^m} \max\{f(P,y) - \beta, 0\} \mathbb{P}[y] dy \tag{2.33}$$

As defined and proved in Rockafellar, Uryasev (2000) [10] in Theorem 1, $\mathrm{CVaR}_\alpha(P)$ can be computed by minimizing $F_\alpha(P,\beta)$ with respect to $\beta$.

$$\mathrm{CVaR}_\alpha(P) = \min_{\beta \in \mathbb{R}} F_\alpha(P,\beta) \tag{2.34}$$

Using this formula to compute CVaR eliminates the requirement of computing VaR which can be computationally hard, moreover the value of VaR arises as byproduct.

Unlike VaR, which ignores the values outside confidence band, CVaR takes into account all values outside this band and computes their mean value. If we look at the example from previous section 2.5.1 specifically formulas 2.28 and 2.29. The expected values and $\mathrm{VaR}_{0.05}$ had the same values for both $P_1$ and $P_2$. Since we already know the value of VaR we can use formula 2.33.

$$\begin{aligned}
\mathrm{CVaR}_{0.05}(P_1) &= 0 + 0.05^{-1}(\max\{1000 - 0, 0\} \cdot 0.01 + \max\{100 - 0, 0\} \cdot 0.03 + \\
&\qquad \max\{0 - 0, 0\} \cdot 0.06 + \max\{-100 - 0, 0\} \cdot 0.9) \\
&= 260 \\
\mathrm{CVaR}_{0.05}(P_2) &= 0 + 0.05^{-1}(\max\{1000 - 0, 0\} \cdot 0.04 + \max\{0 - 0, 0\} \cdot 0.06 + \\
&\qquad \max\{-130 - 0, 0\} \cdot 0.9) \\
&= 800
\end{aligned} \tag{2.35}$$

We can see CVaR of $P_2$ is higher, because there is higher risk of losing everything, than in $P_1$ and optimizer using CVaR as risk measure would prefer $P_1$ over $P_2$. Also CVaR has the property of subadditivity, therefore splitting portfolio into smaller portfolios cannot yield lower risks.

# Chapter 3

# Related work

## 3.1 Modern portfolio theory

The foundation of portfolio optimization field was laid in essay by Harry Markowitz in 1952 [7], where he designed a mathematical framework for selecting portfolio nowadays known as Modern portfolio theory (MPT). The basic idea of this framework is, that the investor has a believe about the future performance of the market and considers return of the portfolio as desired thing and variance of returns an undesired thing.

In his work he defines return of portfolio $R$ per invested dollar as a weighted average of returns on individual instruments $R_i$, where weights $X_i$ are the portions of portfolio invested in the instrument. The weights $X_i$ are non-negative since short selling is prohibited. The return of a instrument $R_i$ is defined as a sum of discounted returns

$$R = \sum_{i=1}^{N} X_i R_i \tag{3.1}$$

$$R_i = \sum_{t=1}^{\infty} d_{it} r_{it} \tag{3.2}$$

where N is the number of instruments in portfolio, $r_{it}$ return per dollar invested in instrument $i$ at time $t$ and $d_{it}$ discount factor for instrument $i$ at time $t$. If we consider a static model, we can replace discounted returns over time with single expected return of the instrument $R_i = r_i$. If we want then to maximize the returns we can set $X_i = 1$ for such $i$ that $R_i$ is maximal or in case of multiple $R_{i_k}, k \in \{1, 2, \ldots, K\}$ are maximal any allocation of $X_i$ such that $\sum_{k=1}^{K} X_{i_k} = 1$ yields the same maximal return.

We further expect, that investor's believes about return of instruments $R_i$ are probabilities rather than single values. If $R_i$ are random variables, then $R$ which is weighted average of $R_i$ is random variable as well. Let $\mu_i$ be the expected value of $R_i$ and $\sigma_{ij}$ be the covariance between $R_i$ and $R_j$ or variance of $R_i$ if $i = j$. We can compute the expected value of the whole portfolio $E$:

$$E = \sum_{i=1}^{N} X_i \mu_i \tag{3.3}$$

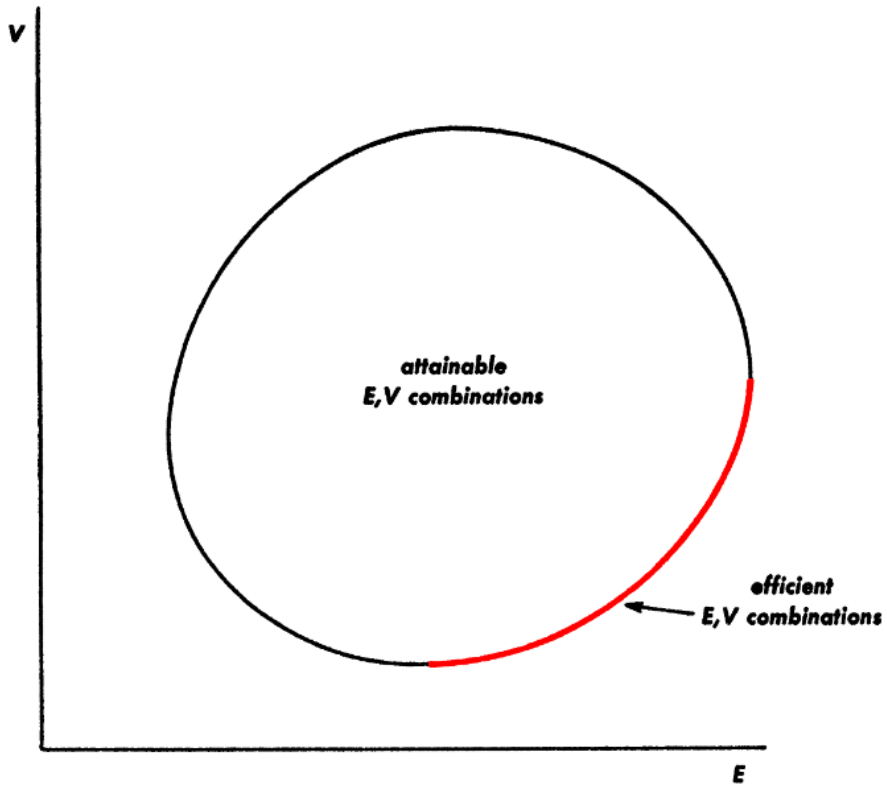



Figure 3.1: Efficient frontier (red) forming a curve along the edge of the edge os the set of attainable portfolios [7]

and its variance $V$:

$$E = \sum_{i=1}^{N} \sum_{j=1}^{N} \sigma_{ij} X_i X_j \tag{3.4}$$

$$\sigma_{ij} = \mathbb{E}[(R_i - \mathbb{E}[R_i])(R_j - \mathbb{E}[R_j])] \tag{3.5}$$

For fixed $(\mu_i, \sigma_{ij})$ we will get set of attainable pairs $(E, V)$. Since we want the the expected value $E$ as high as possible and at the same time we want variance $V$ as low as possible many of those pairs (portfolios) will be outperformed by others. If there are two portfolios $X$ and $\bar{X}$ with their expected values and variances $(E, V)$ and $(\bar{E}, \bar{V})$ respectively and $V \leq \bar{V}, E \geq \bar{E}$ holds, then we say, that $X$ outperforms $\bar{X}$. We call portfolio $X$ efficient, if there is no other portfolio $\hat{X}$ that outperforms $X$. If the set of attainable results forms continuous space, the set of efficient portfolios forms a curve along the edge os the set of attainable results as shown in figure 3.1. This curve is called *efficient frontier* and optimal portfolios in terms of high expected returns and low variance will be on the efficient frontier. By selecting portfolio from the efficient frontier we have a guarantee, that there is no other portfolio with higher expected return that does have lower or equal risk and if we want higher yield we need to accept higher risk.

The drawback of this approach is that using variance as the measure of a risk. High variance does not necessarily mean, that there is high risk of loss, rather it signals that there higher chance of price change, although it might be to our benefit. Variance is computed by deviation from mean, but does not reflect in which direction this deviation is. Another drawback is, that we need to compute covariance for every pair of instruments in the portfolio and computing $n(n+1)/2$ covariances can be computationally difficult.

## 3.2 Scalable Capital

Scalable Capital is a company providing investment and robo-advisory service, where client can specify his investment universe and tolerance for risk and the service will provide advice what trades should the client do or can itself manage the portfolio and do the trades completely without the client. Its main focus is on trading Exchange-traded funds (ETFs), which are funds tradable on exchanges similarly to stock. ETFs are investment companies that offer its clients to invest into fund that invests in stocks, bonds and other assets [11].

The robo-advisor first creates optimal portfolio for the client based on his choice of instruments in portfolio and his risk tolerance. Then monitors the portfolio whether the portfolio complies with the requests from the client. When the robo-advisor finds out, that some constraints are violated, then it starts rebalancing process.

Since investors generally do not want to limit possible income, but want to limit losses using variance as a risk measure is not preferred as it is symmetrical. To measure risk of loss other methods need to be used. Scalable Capital uses three different measures [5]: Value at risk (VaR), Expected shortfall (ES) (also known as CVaR) and Maximum Drawdown (MDD). VaR and CVaR were already defined in sections 2.5.1 and 2.5.2 respectively. MDD is expected maximal loss and can be also expressed as $100\% - \text{VaR}$. Note that we expect that MDD will never be crossed.

# Chapter 4

# Technical approach

## 4.1 Problem definition

We have defined set of $N-1$ instruments (stocks, crypto currencies, etc.) in which we are willing to invest our money and one bank account for which we define, there are no transaction fees, zero risk, but also zero interest. Other instruments have defined transaction fee as a percentage of the cost of the transaction (for simplicity we expect buy and sell has the same fee). Each instrument also has limit, how much can be invested in. This limit is defined as percentage of current value of the portfolio and helps keeping the portfolio diverse, therefore less sensitive, when one of the instruments in the portfolio reports loss. For each instrument in the portfolio we also have the data, how it performed in the history.

Our task is then to find such portfolio, that maximizes profit under specific constraints. These constraints are the risk of the portfolio in the horizon of one year cannot exceed given threshold with defined confidence. On top of that we want our algorithm to regularly reevaluate the portfolio and suggest trades to further increase the expected return and/or lower the risk under the defined threshold, when the market performs worse than expected by previous run of the algorithm.

## 4.2 Solution overview

Computing the optimal portfolio consists of several sub-tasks creating a processing chain. Our goal is to maximize outcome, therefore we need to know expected value of each instrument after one year. We also need to keep risk under given threshold with defined confidence. We will use previously defined CVaR 2.5.2 as risk measure. The formula for computing CVaR 2.33 contains integration over random vector $y$ describing possible states of market and having integral in optimization program constraints will make optimization difficult. Therefore, we need to discretize $y$, which can be done by sampling different scenarios.

### 4.2.1 Prediction of future market state

In this part we will assume, that all instruments in our portfolio are independent of each other, therefore we can model each instrument independently. Our price prediction models 2.1 can only give us one value as prediction, but does not give us estimation of the prediction error. However, we can train these models on older data and predict a value for which we already know the real value. This way we get the errors, that selected model generates on this instrument and can create probability distribution of the future price. As a predictor model we will use triple exponential smoothing or ARIMA, because stock data in general have trends and seasonality and predictions using single or double exponential smoothing or linear regression will not perform well on this type of timeseries.

### 4.2.2 Discretization

Now we have the random vector $y$ of the future prices of instruments, but we would like to generate discrete scenarios from the distributions. Since we have made an assumption, that instruments in the portfolio are independent, for each scenario we just sample a price for every instrument. The CVaR formula is then modified as follows:

$$\mathrm{CVaR}_\alpha(P) = \min_{\beta\in\mathbb{R}} F_\alpha(P,\beta) = \min_{\beta\in\mathbb{R}}[\beta + \alpha^{-1}\int_{y\in\mathbb{R}^m} \max\{f(P,y)-\beta,0\}\mathbb{P}[y]dy] \tag{4.1}$$

$$\mathrm{CVaR}_\alpha(P) = \min_{\beta\in\mathbb{R}}[\beta + \alpha^{-1}\sum_{i=1}^{J} \pi_i \max\{f(P,y_i)-\beta,0\}] \tag{4.2}$$

where J is the number of generated scenarios and $\pi_i$ is probability of given scenario.

## 4.3 Portfolio optimization program

Now that we have prepared scenarios and have define confidence $(1-\alpha)$, accepted loss and maximum value of each instrument we can assemble the optimization program. Since we cannot buy fraction of a stock linear program will not be enough for us, because some variable have to be integers. Now let us define the parts of the program. This part is based on Krokhmal et al.(2002) [4].

### 4.3.1 Objective function

Our main goal is to maximize returns, so in the objective function we are maximizing the sum of instrument volume $x_i$ multiplied by expected value of the instrument $\mathbb{E}[y_i]$.

$$\max_x \sum_{i=1}^{N} x_i\mathbb{E}[y_i] = \min_x -\sum_{i=1}^{N} x_i\mathbb{E}[y_i] \tag{4.3}$$

### 4.3.2 Risk constraints

We have defined formula for computing CVaR over given set of scenarios 4.2 as minimum of some function $\hat{F}_x$. However, we do not actually compute the minimum, since our only concern is to keep the CVaR under some threshold. The threshold is defined as a percentage of current value of the portfolio $\omega$.

$$\beta + \alpha^{-1} \sum_{j=1}^{J} \pi_i \max\{f(P, y_j) - \beta, 0\} \leq \omega \sum_{i=1}^{N} q_i x_i^0 \tag{4.4}$$

where $q_i$ is current price of instrument $i$ and $x_i^0$ is the initial volume of the instrument. But now we have function max inside our constraint. We will introduce auxiliary variables $z_1, z_2, \ldots, z_J$, where $z_j = \max\{f(P, y_j) - \beta, 0\}$ and since we want $z_i$ as small as possible, we can set $z_j \geq \max\{f(P, y_j) - \beta, 0\}$ without changing the the result. Now we have defined $z_j$ to be greater than maximum of two values, therefore we can split the condition into two.

$$z_j \geq \max\{f(P, y_j) - \beta, 0\} \tag{4.5}$$

$$\begin{aligned} z_j &\geq f(P, y_j) - \beta \\ z_j &\geq 0 \end{aligned} \tag{4.6}$$

Next we need to define the loss function $f(P, y_j)$, where $P = (x_1, x_2, \ldots, x_N)$ and we know $x_i^0$ and $q_i$.

$$f(P, y_j; x^0, q) = \sum_{i=1}^{N} x_i^0 q_i - x_i y_{ji} \tag{4.7}$$

This gives us set of constraints for CVaR:

$$\zeta + \frac{1}{\alpha} \sum_{j=1}^{J} \pi_j z_j \leq \omega \sum_{i=1}^{N} q_i x_i^0 \tag{4.8}$$

$$z_j \geq \sum_{i=1}^{N} (q_i x_i^0 - y_{ji} x_i) - \zeta, \qquad z_j \geq 0, \qquad j \in \{1, 2, \ldots, J\} \tag{4.9}$$

### 4.3.3 Maximal instrument value constraint

Next set of constraints is for limiting how much of a portfolio can be invested in particular instrument. For each instrument we have value $v_i$, defined as percentage of total value of portfolio.

$$q_i x_i \leq v_i \sum_{k=1}^{N} q_k x_k \tag{4.10}$$

### 4.3.4 Transaction cost constraint

Each instrument has property $c_i$ which is the transaction cost in percent. The cost of a transaction is then computed as $c_i q_i |x_i - x_i^0|$. Similarly to having max function in constraint,

we do not want to have function absolute value. For each instrument we will add two auxiliary variables $u_i^+ \geq 0$ and $u_i^- \geq 0$, where $|x_i - x_i^0| = u_i^+ - u_i^-$. In optimal solution only $u_i^+$ or $u_i^-$ can be non-zero, otherwise we would be buying and selling the same instrument, which only increases expenditures, but will not bring any gain. Now we need to add a constraint that the value of portfolio before optimization has to equal value of the portfolio after optimization plus transaction fees.

$$\sum_{i=1}^{N} q_i x_i^0 = \sum_{i=1}^{N} c_i q_i (u_i^+ + u_i^-) + \sum_{i=1}^{N} q_i x_i \tag{4.11}$$

$$x_i - x_i^0 = u_i^+ - u_i^-, \qquad i \in \{1, 2, \ldots, N\} \tag{4.12}$$

$$u_i^+ \geq 0, \quad u_i^- \geq 0, \qquad i \in \{1, 2, \ldots, N\} \tag{4.13}$$

### 4.3.5 Stock volume constraints

Since our model does not support short selling of stocks, we require $x_i \geq 0$ for each instrument. Moreover we cannot buy fractions of stock, therefore we need to add constraint, that if instrument $i$ is stock, the value of $x_i$ has to an integer. For bank account and cryptocurrencies we do not requires such constraints and the value of $x_i$ can be real number.

$$x_i \geq 0, \qquad i \in \{1, 2, \ldots, N\} \tag{4.14}$$

$$x_i \in \begin{cases} \mathbb{Z}, & \text{if } x_i \text{ is stock item} \\ \mathbb{R}, & \text{otherwise} \end{cases} \qquad i \in \{1, 2, \ldots, N\} \tag{4.15}$$

### 4.3.6 Final optimization program

Now we just simply add together all previously defined constraints and get the final mixed integer linear program.

$$\min_{x,\zeta} -\sum_{i=1}^{N} \mathbb{E}[y_i] x_i \tag{4.16}$$

subject to

$$\zeta + \frac{1}{\alpha} \sum_{j=1}^{J} \pi_j z_j \leq \omega \sum_{i=1}^{N} q_i x_i^0 \tag{4.17}$$

$$z_j \geq \sum_{i=1}^{N} (q_i x_i^0 - y_{ji} x_i) - \zeta, \qquad z_j \geq 0, \qquad j \in \{1, 2, \ldots, J\} \tag{4.18}$$

$$q_i x_i \leq v_i \sum_{k=1}^{N} q_k x_k, \qquad i \in \{1, 2, \ldots, N\} \tag{4.19}$$

$$\sum_{i=1}^{N} q_i x_i^0 = \sum_{i=1}^{N} c_i q_i (u_i^+ + u_i^-) + \sum_{i=1}^{N} q_i x_i \tag{4.20}$$

$$x_i - x_i^0 = u_i^+ - u_i^-, \qquad i \in \{1, 2, \ldots, N\} \tag{4.21}$$

$$x_i \geq 0, \quad u_i^+ \geq 0, \quad u_i^- \geq 0, \qquad i \in \{1, 2, \ldots, N\} \tag{4.22}$$

$$x_i \in \begin{cases} \mathbb{Z}, & \text{if } x_i \text{ is stock item} \\ \mathbb{R}, & \text{otherwise} \end{cases} \qquad i \in \{1, 2, \ldots, N\} \tag{4.23}$$

# Chapter 5

# Implementation

Part of this work is implementation of an optimizer. This section provides information about the structure of the application and technologies used to achieve the result.

## 5.1 Technology stack

### 5.1.1 Kotlin

Kotlin is JVM-base programming language that combines Object-oriented programming (OOP) and functional programming. It is very similar to Java, but unlike Java Kotlin has nullable and not-null types. Kotlin is fully compatible with Java, therefore we can use all libraries that are written in Java, one example of such library is the implementation of ARIMA predictor.

### 5.1.2 Spring Boot

Spring Boot is great tool for Dependency injection (DI) and building webservers. Since Spring Boot 2.0 and Spring 5.0 Kotlin is one of three officially supported languages for Spring together with Java and Groovy. Spring Boot and its dependency injection engine helps to divide our application into independent modules that can be swapped without changing other parts of the application.

Our application is split into following parts:

- Data provider
- Value predictor
- Scenario generator
- Optimizer

The benefit of having independent modules is when we want to replace our ARIMA predictor with triple exponential smoothing predictor, we can just swap these modules and the rest of the application is unaware of this change. This results in a framework where one

needs to provide these four modules and the whole system is ready to optimize. It is also very easy to add more complex predictors or change the optimizer from Gurobi to other one, which has free license.

## 5.2 Data providers

Our approach to optimize portfolio is based on predicting future values of stock based on historical data, therefore we will need data sources, that can provide us information about historical prices of different stocks.

### 5.2.1 AlphaVantage

AlphaVantage is a great data source which provides realtime and historical data for free. Our concern is only about historical data, since the optimizer is for long term investments. The only downside of this API is low request rate (5 request per minute, 500 per day), however, since we only need historical data which does not change very often, we can bypass this limitation by introducing local cache. AlphaVantage has data for most large companies as well as cryptocurrencies.

## 5.3 Mathematical solvers

Since we have defined the optimization problem as a linear program, we will need a LP solver. In following sections there are examples of such solvers that are either free or they provide free academic license. Since we have used Kotlin as our main programming language, we will focus on those solvers that can be used from JVM-based language.

### 5.3.1 Gurobi Optimizer

Gurobi Optimizer is a tool for solving many optimization tasks including Linear Programs (LP), Quadratic Programs (QP), Quadratically Constrained Programs (QCP), Mixed-Integer Programs (MIP), Mixed-Integer Linear Programs (MILP), Mixed-Integer Quadratic Programs (MIQP), and Mixed-Integer Quadratically Constrained Programs (MIQCP).

Gurobi Optimizer offers APIs for many languages including Java and many other (c++, python, .NET). The Java API treats all parts of problem definition (variables, constraints, ...) as Java objects which fits very well into our application and keeps the source code clean. The license, however, is not free, but there is free academic license.

### 5.3.2 IBM ILOG CPLEX Optimizer

IBM ILOG CPLEX Optimizer ("CPLEX") is another solver for LP, MIP, QP and QCP tasks. In contrast to Gurobi Optimizer, the models are not Java object, but rather plain arrays and matrices. This ways of setting linear programs is memory efficient, however the readability of the code is worsened. Another difference from Gurobi Optimizer is that CPLEX provide free community version as well as free academic and full version.

### 5.3.3 GLPK

GNU Linear Programming Kit (GLPK) is representative of free open-source optimization tool for solving LP and MIP. It is part of GNU project and is published under GNU GPL license. It is written in ANSI C as callable library. GLPK doesn't have any Java interface, but there is a standalone project GLPK for Java, which enables using GLPK from Java application using Java Native Interface (JNI).

# Chapter 6

# Evaluation

We will divide the evaluation into several steps. First, we want to find, which predictor works the best on stock prices. In the section 6.1 we will test predictors against real-world values and measure its error in prediction. Next we will prepare portfolio and using predictors based on their error from section 6.1, we will optimize the portfolio by generating scenarios as defined in section 2.4 and then we will find the optimal distribution of our funds between the items in the the portfolio, so we maximize our return while keeping the risk low. Lastly we will use our optimizer to reevaluate the portfolio every month to take in account the changes on the market and modify the portfolio to keep the expected return maximal.

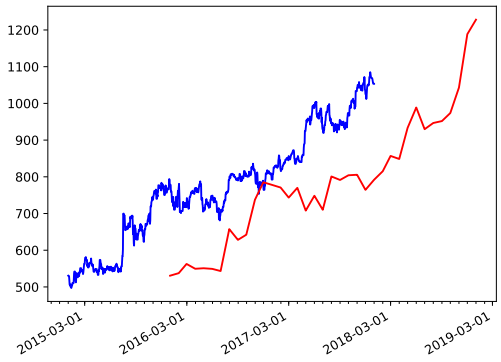
## 6.1 Predictors evaluation

The first experiment is to predict the value of the stock of Alphabet Inc. (GOOGL). We will define the time range for the experiment from 1.1.2015 to 1.1.2018. We will also define our prediction horizon as one year, therefore the predictions will be in range from 1.1.2016 to 1.1.2019. The selected predictors are ARIMA as defined in section 2.1.3 and triple exponential smoothing as defined in section 2.1.6. We will use the predictor both on the raw historical data and historical data transformed using Box-Cox transformation (see section 2.3.2).

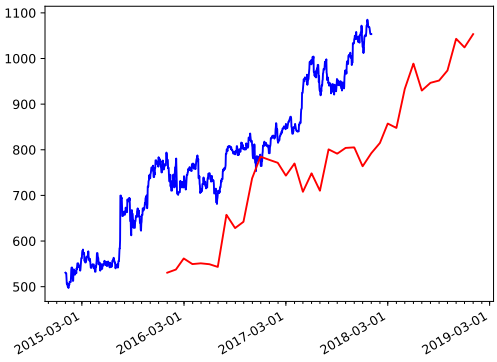
As we can see in the figure 6.1 ARIMA predictor suffers from lagging behind. On the other hand triple exponential smoothing predictor does not seem to suffer from lag as much as ARIMA predictor. In the figure 6.2 we can see the absolute error the predictors made. In addition to that we have table 6.1, where is for each predictor the mean squared error for predictions in our testing time range. The most accurate prediction for this stock was made by triple exponential smoothing predictor on raw data, however the difference between triple exponential smoothing and ARIMA predictor regardless the transformation.

To compare these results with predictions on other stocks, we will do the same experiment for Intel Corporation (INTC) and Apple Inc. (AAPL).

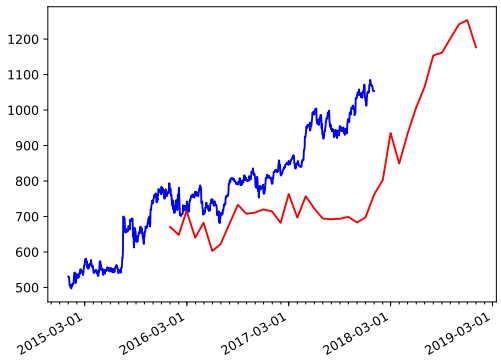
From table 6.2 we can see that for INTC stock the difference in errors between ARIMA and triple exponential smoothing predictors is several orders of magnitude in favor for ARIMA predictor. Also the combination of triple exponential smoothing and Box-Cox transformation produces large errors as can be seen on INTC and AAPL stock.



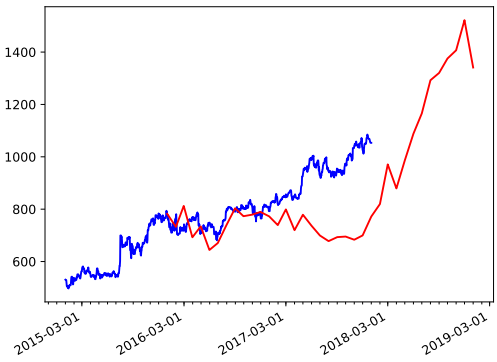

(a) ARIMA predictor




(b) ARIMA predictor with Box-Cox transformation




(c) Triple exponential smoothing predictor




(d) Triple exponential smoothing predictor with Box-Cox transformation

Figure 6.1: Predictions of stock prices of GOOGL stock with one year horizon. Blue line is the actual value, red line denotes the prediction.

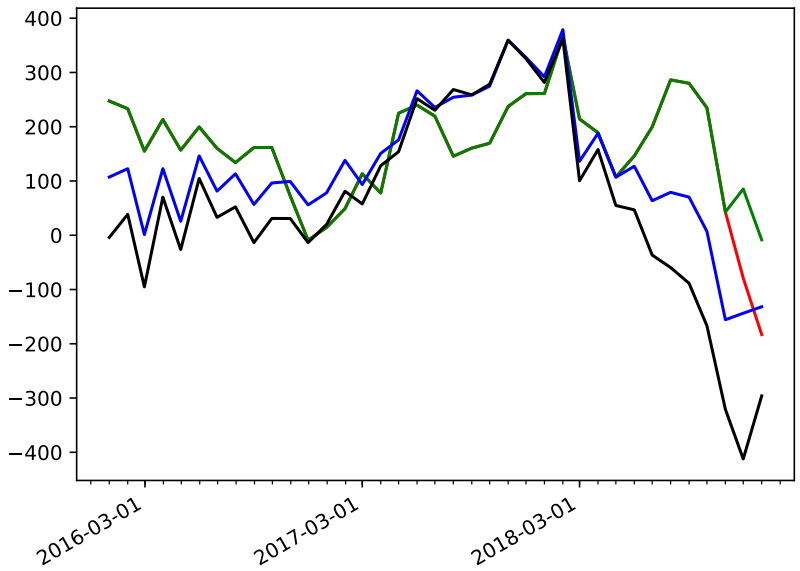



Figure 6.2: Errors in predictions of stock prices of GOOGL stock with one year horizon. ARIMA predictor is denoted as a red line, ARIMA predictor with Box-Cox transformation is denoted as a green line, triple exponential smoothing predictor is denoted as a blue line and triple exponential smoothing predictor with Box-Cox transformation is denoted as a black line.

| | Mean squared error |
|---|---|
| ARIMA | 36283.16 |
| ARIMA with BoxCox transformation | 35422.80 |
| Triple exponential smoothing | 31186.94 |
| Triple exponential smoothing with BoxCox transformation | 35225.29 |

Table 6.1: Mean squared error for all predictors for value of GOOGL stock.

| | GOOGL | INTC | AAPL |
|---|---|---|---|
| ARIMA | 36283.16 | 79.61 | 1544.64 |
| ARIMA with BoxCox transformation | 35422.80 | 90.25 | 1545.19 |
| Triple exponential smoothing | 31186.94 | 14178.65 | 1734.66 |
| Triple exponential smoothing with BoxCox transformation | 35225.29 | 1.855e7 | 1.287e11 |

Table 6.2: Mean squared error for all predictors for value of GOOGL, INTC and AAPL stock.

## 6.2 Single step optimization

In this experiment we will define a portfolio. Following are the items of which is the portfolio made:

1. Bank account
2. Apple Inc. (AAPL)
3. Microsoft Corporation (MSFT)
4. Intel Corporation (INTC)
5. Advanced Micro Devices, Inc. (AMD)
6. Facebook, Inc. (FB)
7. Alphabet Inc. (GOOGL)
8. Tesla Inc. (TSLA)
9. General Motors (GM)
10. Twitter Inc. (TWTR)

At the start of the experiment we will assume we have $100,000 USD in our bank account. We will also assume that current date is 1.1.2015 (we do not know about what happened after this date). We will set the transaction cost equal to 1% of the traded value and limit the maximal value invested in one stock to be 20% of the total value of the portfolio. Our risk limits are with probability of 80% our loss will be less than 20% of the portfolio value (with one year horizon).

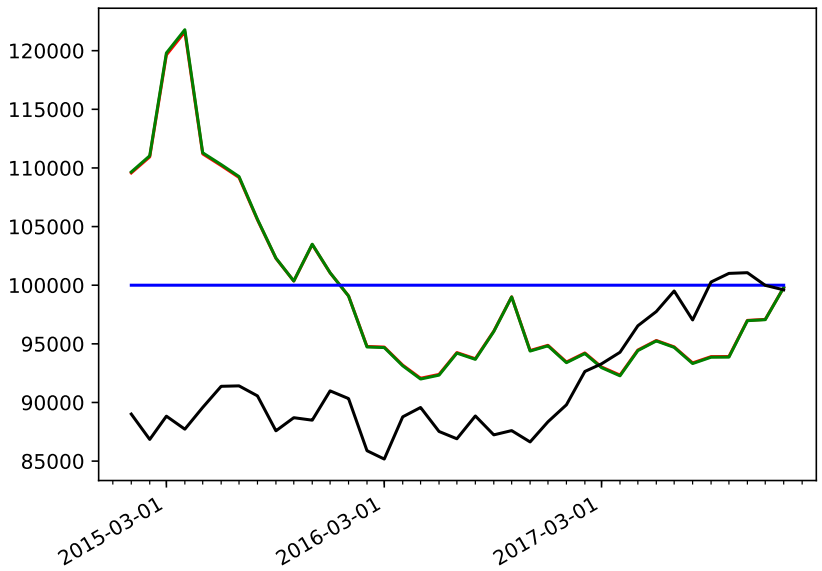



Figure 6.3: Value of portfolio, if it was optimized only at the start. ARIMA predictor is denoted as a red line, ARIMA predictor with Box-Cox transformation is denoted as a green line, triple exponential smoothing predictor is denoted as a blue line and triple exponential smoothing predictor with Box-Cox transformation is denoted as a black line.

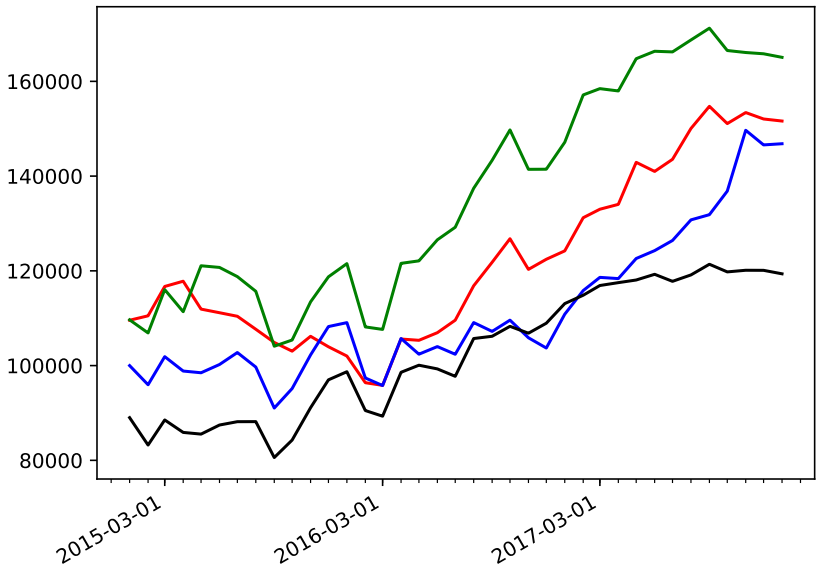



Figure 6.4: Value of portfolio, if it was optimized every month. ARIMA predictor is denoted as a red line, ARIMA predictor with Box-Cox transformation is denoted as a green line, triple exponential smoothing predictor is denoted as a blue line and triple exponential smoothing predictor with Box-Cox transformation is denoted as a black line.

In this experiment we will optimize the portfolio only at the start and watch how the value of the portfolio changes in time, if we keep the portfolio the same (the only trades can be done at the start, after that no trades are done).

As we can see in figure 6.3 the errors made by triple exponential smoothing predictor are so big, that the optimizer does not make any trade, since all of them are considered as high risk. Another point to make here is that the value of the portfolio is higher han the initial investment at the beginning, but as the time progress the value fell under the initial value. This is because we did not allowed the portfolio to reevaluate during the experiment.

## 6.3 Continuous portfolio optimization

In this experiment we will reuse the previously defined portfolio, but this time we will force the portfolio to reevaluate at the beginning of each month. In this case when one company's stock start falling, the optimizer will move assets to different stock.

From figure 6.4 we can see, that forcing portfolio to reevaluate every month has very positive effect on the value of portfolio. In this example the portfolio is optimized based on at maximum one month old data. In the previous single step optimization experiment after three years the value of the portfolio was close to the initial value. However in this

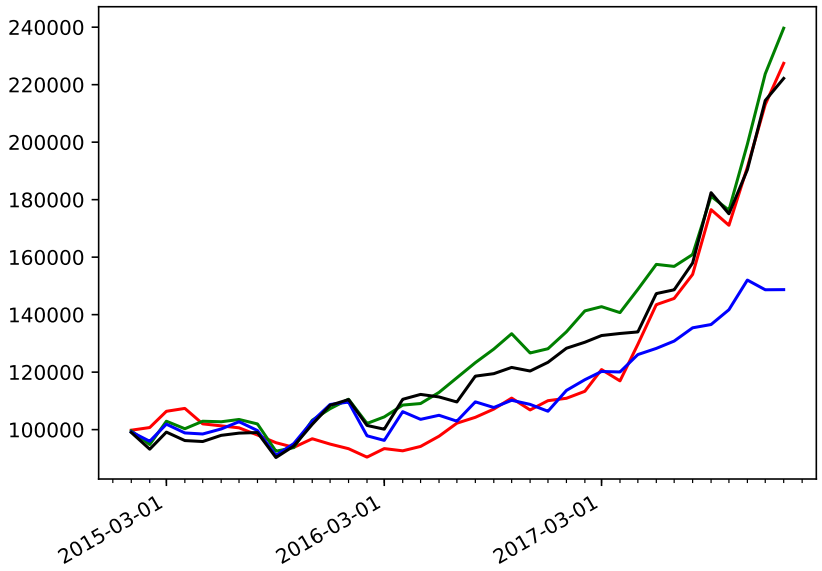



Figure 6.5: Value of portfolio containing Bitcoin, if it was optimized every month. ARIMA predictor is denoted as a red line, ARIMA predictor with Box-Cox transformation is denoted as a green line, triple exponential smoothing predictor is denoted as a blue line and triple exponential smoothing predictor with Box-Cox transformation is denoted as a black line.

### 6.3.1 Continuous portfolio optimization with cryptocurrencies

In the last years a new trend in investments appeared, investing in cryptocurrencies. Although cryptocurrencies exists for many years already, in 2017 investments in cryptocurrencies became interesting, when the value of Bitcoin (BTC) rocketed from about $500 USD for BTC to more than $19,000 USD in December 2017. To illustrate how this optimizer would behave, if the portfolio contained Bitcoin we will use the portfolio from previous section and add Bitcoin to it.

As we can see in figure 6.5 the optimizer took the advantage of Bitcoin increasing its value with high speed and in three years managed to get from $100,000 USD to about $240,000 USD

# Chapter 7

# Conclusion

In this work we have created a framework for optimizing portfolios. We have proposed several predicting algorithms, most importantly ARIMA predictor as defined in section 2.1.3 and triple exponential smoothing as defined in section 2.1.6. We have also showed how these predictors perform on real-world data and how transforming the raw data might influence the precision of prediction.

Furthermore we have proposed an algorithm that from given set of scenarios can find the optimal distribution of our assets to maximize the expected return while keeping the risk level below given threshold. We have proposed an approach how to generate scenarios from distribution of future value of the stock which we can obtain by predicting a value using one of our predictor and keeping track of the errors that the predictor made in the past. We have shown that this approach can yield returns up to 20% per year.

This framework is possible to run in production environment, however the model does not count with taxes, the only fee considered in this model is the transaction fee for every trade made. On the other hand holding stock can also yield another income in a form of dividends, which lowers the impact of taxation.

Since this work was written as a framework it is easy to add new predictors, scenario generators or even different optimizer. Currently all instruments are modeled independently on the others and this might lead to better prediction thus more accurate risk models and higher expected returns. Another way to improve this framework in the future is to create a predictor which does not suffer from lag.

# Bibliography


[1] Philippe Artzner, Freddy Delbaen, Jean-Marc Eber, and David Heath. Coherent measures of risk. *Mathematical finance*, 9(3):203–228, 1999.

[2] George EP Box and David R Cox. An analysis of transformations. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 211–252, 1964.

[3] CC Holt. Forecasting trends and seasonals by exponentially weighted averages. carnegie institute of technology. Technical report, Pittsburgh ONR memorandum, 1957.

[4] Pavlo Krokhmal, Jonas Palmquist, and Stanislav Uryasev. Portfolio optimization with conditional value-at-risk objective and constraints. *Journal of risk*, 4:43–68, 2002.

[5] Scalable Capital Limited. The scalable capital investment process. White paper, November 2016. URL <`https://uk.scalable.capital/assets/3x3i7a9xgm11/1ypERGd2JmUmOcsGsCe4uE/91d5745c007641f72fb5b293b30c039b/WhitePaper_ScalableCapital_UK.pdf`>. [Online; accessed 07-May-2018].

[6] Spyros Makridakis and Michele Hibon. Arma models and the box–jenkins methodology. *Journal of Forecasting*, 16(3):147–163, 1997.

[7] Harry Markowitz. Portfolio selection. *The journal of finance*, 7(1):77–91, 1952.

[8] National Institute of Standards and Technology. Nist/sematech e-handbook of statistical methods. URL <`https://www.itl.nist.gov/div898/handbook/`>. [Online; accessed 07-May-2018].

[9] Tommaso Proietti and Helmut Lütkepohl. Does the box–cox transformation help in forecasting macroeconomic time series? *International Journal of Forecasting*, 29(1): 88–99, 2013.

[10] R Tyrrell Rockafellar, Stanislav Uryasev, et al. Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42, 2000.

[11] U.S. Securities and Exchange Commission. Exchange-traded funds (etfs), . URL <`https://www.investor.gov/introduction-investing/basics/investment-products/stocks`>. [Online; accessed 24-December-2018].

[12] U.S. Securities and Exchange Commission. Stocks, . URL <`https://www.sec.gov/fast-answers/answersetfhtm.html`>. [Online; accessed 07-May-2018].

[13] Peter R Winters. Forecasting sales by exponentially weighted moving averages. *Management science*, 6(3):324–342, 1960.

# Appendix A

# Nomenclature

CDF cumulative distribution function

CVaR Conditional value at risk

DI Dependency injection

ES Expected shortfall

ETFs Exchange-traded funds

GLPK GNU Linear Programming Kit

JNI Java Native Interface

LP Linear Programs

MA Moving average

MDD Maximum Drawdown

MILP Mixed-Integer Linear Programs

MIP Mixed-Integer Programs

MIQCP Mixed-Integer Quadratically Constrained Programs

MIQP Mixed-Integer Quadratic Programs

MPT Modern portfolio theory

MSE Mean squared error

OOP Object-oriented programming

QCP Quadratically Constrained Programs

QP Quadratic Programs

VaR Value at risk

# Appendix B

# CD content

| Directory name | Description |
|---|---|
| figures | figures used in this thesis. |
| src | Kotlin source codes, config files and scenarios used in evaluation. |
| thesis | thesis sources in latex and pdf format |