A Work Project, presented as part of the requirements for the Award of a Master's degree in Finance from the dNova School of Business and Economics

Using image recognition for trading

Berkay Günes
(40804)

Work project carried out under the supervision of:

Patricia Xufre

04-01-2021

Abstract

This research aims to gain a deeper understanding of how image recognition can be applied in trading.

In recent years, artificial intelligence has influenced various industries, including the financial sector. It can be observed that there are new ways of predicting stock trends. This paper aims to address the question to what extend convolutional neural networks can be used in trading.

On the empirical side several convolutional neural networks have been analyzed and were compared with a zero-predictive model. This study's results do not show reliable results that convolutional neural networks should be used for this sort of task.

Keywords:
Convolutional Neural Networks, Image Recognition, Finance, Trading

This work used infrastructure and resources funded by Fundação para a Ciência e a Tecnologia (UID/ECO/00124/2013, UID/ECO/00124/2019 and Social Sciences DataLab, Project 22209), POR Lisboa (LISBOA-01-0145-FEDER-007722 and Social Sciences DataLab, Project 22209) and POR Norte (Social Sciences DataLab, Project 22209).

Table of Contents

1 Introduction

The first chapter describes the importance of computer vision. Furthermore, the use of neural networks in finance will be discussed, and lastly, the study's research question is clarified.

1.1 Background

Computer vision systems have long been able to rival the human eye and continue to improve. Autonomous driving, virtual reality, and augmented reality are just some of the application areas for computer vision. With increasing computational capabilities, the deployment of artificial intelligence got more popular. For computer vision, a convolutional neural network (CNN), called LeNet-5, proposed by Yann LeCun, was the breakthrough in 1998. This model was steadily expanded and won the ImageNet Object detection challenge for the first time in 2012 (Schmelzer, 2019). In the challenge, algorithms compete to recognize and describe pictures, whose accuracy increased from 71.3% to over 97.3 % since the contest's launch in 2010 (Bernard, 2018). Since then, all winning entries are based on CNNs and deep learning models are off to the races.

1.2 Problem Statement

The technical revolution through convolutional neural networks is sweeping through many industries and opening up new possibilities. Can these astonishing results also be transferred to trading? In the past years, artificial intelligence already changed the financial sector massively. Machine learning models took over the assessment and processing of credit inquiries. Companies evaluate massive databases with the help of deep learning. This helps prevent fraud and handle resource-intensive, repetitive processes and customer services automatically without any quality loss (Columbus, 2020). Besides these successes in the financial sector, big hedge funds, such as Renaissance Technologies or Two Sigma, work under high pressure to generate high returns using computers, math and big data. They are competing with firms such as AQR Capital Management and BlackRock (Vardi, 2017).

As shown, CNNs are good at detecting objects in images. The author expects that this can be useful to detect trends in stock charts, as up until today, many traders decide by using perception and experience. Furthermore, it might spot relationships that human cannot find easily. Accurately predicting the trend of stocks can help to avoid risk and secure higher returns.

This study aims to examine the effects of CNNs on technical analysis through literature and quantitative research. The following question is based on the recent successes of neural networks: *"To what extend can convolutional neural networks be used for image recognition in trading?"*

2 Literature / Background

The literature review presents the theory starting from the financial point, followed by defining CNNs, how they are built and their usage in the financial environment.

2.1 Trading in Finance

Within securities analysis, a distinction is made between the two approaches. One of them is the fundamental analysis, which is based on the thesis that a company has a fair value to which the stock market price also adjusts in the long term. If the calculated intrinsic value corresponds to the value at which the company is traded on the market at that time, it is said to be a fairly valued company. However, if the intrinsic value of the company is higher than the market value, the fundamental analysts see this as a buy signal because, in his opinion, the company is undervalued. Complementary to this is the technical analysis, which is to study and predict price movements in the financial market based on an asset's chart history. It is based on the assumption that there are recurring, observable events with similar, probable future developments. Assuming that the random walk theory, which suggests that changes in stock prices are independent of each other, is not accurate, value can be generated by technical analysis (Cochrane & Maskowitz, 2017). The fundamental of technical analysis is that all decision-relevant information about the past and future is already in the price. Technical analysts do not attempt to measure the intrinsic value but instead try to identify trends in the

early stages of their development for trading in the direction of those trends. This presupposes that capital markets are not efficient. Otherwise, it would not be possible to forecast and arbitrate the trend itself by analysing it. Therefore, information diffusion in the market must exist (Murphy, 1999).

2.1.1 Technical Indicators

Thus, depending on which discipline a chart analyst follows, specific geometric patterns or purely statistical quantitative indicators can be used (Pring, 2014). In the following, some popular technical indicators will be described.

Moving averages

One of them is moving averages, which is a method for smoothing time series. The smoothing is done by removing higher frequency components. As a result, a new data point set consists of the mean values of equally sized subsets of the original data point set. Using the arithmetic mean values of the specified window results in the simple moving average (SMA) (Pring, 2014).

$$SMA_t = \frac{1}{n} \sum_{k=t-n+1}^{t} price_k \tag{1}$$

While all prices are given the same weight when calculating the SMA, the idea behind the exponential moving average (EMA) is to provide recent prices higher weights. For this purpose, a decay factor λ_{EMA} is introduced, which results from the time window n. Using this constant, the current EMA is calculated from the previous EMA and the current closing price (Pring, 2014).

$$EMA_t = \lambda_{EMA} * price_t + (1 - \lambda_{EMA}) EMA_{t-1} \; with \; \lambda_{EMA} = \frac{2}{(n+1)} \tag{2}$$

Moving Average Convergence Divergence

The moving average convergence divergence (MACD) is calculated from the difference between two exponential moving averages. It is a trend-following momentum indicator. Usually, the MACD is calculated by subtracting the 26-day EMA from the 12-day. The analysis

is mostly used in connection with a signal line, which is usually a nine-day EMA of the MACD (Pring, 2014).

$$MACD_t = EMA_{12} - EMA_{26} \; and \; Signal_t = EMA_9 \; of \; MACD \tag{3}$$

A positive MACD indicates an upward, a negative MACD a downward trend. The distance of the MACD from its zero lines indicates the strength of the trend. If the distance between the signal line and the MACD increases, the trend increases, and vice versa (Pring, 2014).

2.1.2 Behavioral Finance

Another essential factor for technical analysis is psychological elements that affect market participants behavior. The psychological factors, in particular, can cause market participants to behave similarly in comparable market cycles so that similar price patterns arise over time. This imposes that history can repeat itself in financial markets and be put to profitable use (Murphy, 1999). In other words, it can be seen as a self-fulfilling prophecy. If more and more investors follow the chart analysis, more and more investors act on the chart analysis signals.

Another argument that supports capital markets are not efficient is the prospect theory. With their prospect theory, Kahneman and Tversky already showed in the late 1970s that a loss hurts investors twice as much as a profit of the same amount would trigger positive emotions. This different perception of pain means that losses are realized too late but gains too early (Kahneman & Tversky, 1979).

2.2 Neural networks

The concept of neural networks is usually traced back to the functionality of the human brain. However, these models are not designed as realistic models of biological neurons but can help make the concept more visible. Instead, they are a mathematical framework for learning representations from data (Chollet, 2018). The neural networks core idea is that many simple computing units can work together to imitate intelligent behavior. A single one of these units is called a neuron. A simple neuron has several inputs x, which are multiplied by weights w, and

a bias b is added, which is considered a constant correction value (Goodfellow, Bengio, & Courville, 2016). The activation of a neuron can be calculated as follows:

$$y = b + \sum_{i=1}^{n} x_i w_i = x^T w + b \tag{4}$$

The activation represents the stimulation of the neuron. Only when it exceeds a threshold, the neuron becomes activated. Although a single neuron can map simple relationships, more complex learning tasks cannot be solved. For this purpose, neurons are arranged in fully connected layers, creating artificial neural networks.

2.2.1 Activation function

It is a node added to the output of neurons, also known as the transfer function, to determine the outcome. Furthermore, without activation functions, there would be many different linear combinations of the inputs. Adding an activation function to the neurons leaves linearity behind and complex models can be created with non-linear relationships.

ReLu and leaky ReLu

In modern neural networks, it is recommended to use the rectified linear unit (ReLu) defined by the activation function $ReLu(x) = \max(0, x)$. If this function is applied to the output of a neuron, it yields it as non-linear. However, rectified linear units are very close to being linear, helping to preserve linear model properties. The problem is that any negative values instantly go to zero, reducing the model's ability to fit or train the data correctly**Invalid source specified.**. Leaky ReLu seeks to fix the ReLu problem. Instead of the function being zero when $x < 0$, a leaky ReLu will have a small negative slope, as α in the following formula:

$$leaky\ ReLu(x) = \begin{cases} x\ if\ x \geq 0 \\ \alpha x\ if\ x < 0 \end{cases} \tag{5}$$

Logistic Sigmoid and Hyperbolic Tangent (linear?)

Preceding the rectified linear units, most neural networks used either the logistic activation function $\sigma(x) = \frac{1}{1+e^{-x}}$ or the hyperbolic tangent activation function $tanh(x) = \frac{2}{1+e^{-2x}} - 1$.

These activation functions are closely related as the logistic sigmoid function is a rescaled version of the hyperbolic tangent functions because $\tanh(x) = 2\sigma(2x) - 1$.

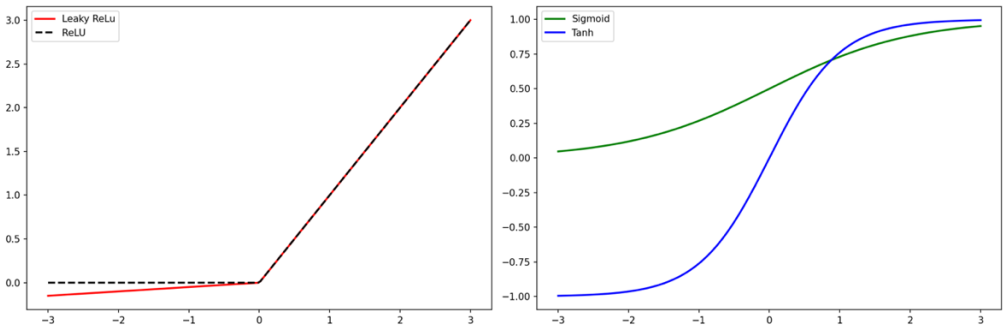

Figure 1: Visualizing different activation functions.

2.2.2 Convolutional Neural Network

Convolutional neural networks are particular neural networks for time-series, speech and image data. The two most essential components of the CNN are the convolutional layer, the namesake for these networks, and the pooling layer. Figure 2 illustrates the structure of the LeNet-5. In contrast to traditional neural networks in which each neuron is wholly connected to all neurons of the layer in front and behind it, neurons in the convolutional layer are only connected to their receptive field's pixels. This is also called sparse interactions. In other words, traditional neural networks learn global patterns, while CNNs learn local patterns (Chollet, 2018).

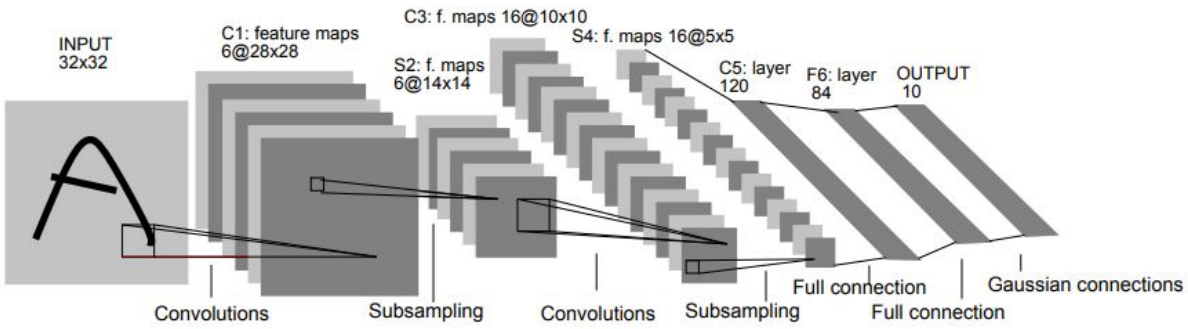

Figure 2: Convolutional neural network with the structure of LeNet-5, which is also used in this study from the original paper (Lecun, Bottou, Bengio, & Haffner, 1998).

Kernel

Sparse interactions are mastered by making the kernel smaller than the input. It ensures that the same patterns are recognized throughout the image and that fewer parameters have to be optimized, which speeds up processing. It works as a filter with a constant step size that scans

over the input pixel matrix to extract features, convolving the image. The kernel moves from left to right and jumps to the next lower line after each pass until the entire image is traversed. The output is referred to as the feature map. Feature map values for two-dimensional images, are calculated according to the following formula (6). The input is denoted by I, the kernel by K and the indexes of the output's rows and columns by m and n (Goodfellow, Bengio, & Courville, 2016).

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \quad (6)$$

Instead of a single kernel, several kernels can be applied in parallel, each of which generates a feature map. These result in a feature map matrix.

Pooling Layer

A classical layer of a convolutional neural network consists of three parts. In the first part, several convolutions are performed to produce the feature map. Afterward, the feature map is run through a non-linear activation function, such as ReLu. In the last part, pooling layers are used to modify the output further. The pooling layer aims to pass on only the most relevant signals to the next layer by downsampling the incoming layer's size. This also reduces the number of parameters, the computation time and storage space.

However, there are several pooling functions. One of them is the max pooling operation, which simply reports the maximum output within a rectangular neighborhood and discards the others. Another one is the average pooling operation, which reports the average of the rectangular neighborhood (Goodfellow, Bengio, & Courville, 2016).

Densely Connected Networks

One or more fully connected layers follow the last pooling layer. These layers have a typical neural network structure in which all neurons are connected to all inputs and all outputs. This must first be flattened into a vector to feed the convolutional and pooling layers matrix output into the dense layers. The filters output signals are independent of an object's position, so there

are no longer any position features but location-independent information. The problem of overfitting often arises here. One possibility of regularization is dropout layers, which ensure that a randomly determined percentage of neurons is switched off and is not considered in the next calculation step. This prevents the neurons from adapting too much to the training data and thus no longer generalizes. However, batch normalization can optimize the convolutional layers, making dropout unnecessary at this layer as it reduces internal covariate shift (Li, Chen, Hu, & Yang, 2018).

2.2.3 Convolutional neural networks in Finance

From conventional methods to deep learning models, there are various approaches to forecasting financial time series. In the past artificial neural networks, support vector machines, decision trees and ensembles of classifiers were among the most preferred machine learning algorithms for price predictions in trading. With the introduction of deep learning, models such as CNNs started becoming more notable. However, since this is a new research field, the number of publications in this area is limited. Sim et al. (2019) used a convolutional neural network in their paper to predict the S&P 500. They build four models, including different feature sets, which learn the moving patterns of various indicators for 30 minutes and forecast the increase or decrease in the stock price after one minute (Sim, Kim, & Ahn, 2019). Another research very close to this is from Arratia and Sepulveda (2019). They use a 1D-array and convert the financial data into images through recurrence plots and feed them into a convolutional neural network. Arratia and Sepulveda use 12 months of data to predict the increase or decrease in the next month. With 10-fold cross-validation, they achieve an accuracy score of 61.1% with the 1D-array and 63.22% with the images. They also have shown that better results can be obtained by preprocessing the input to images (Arratia & Sepulveda, 2019). Other publications similar to this one, but with different problem statements exist as well. One of them is from Chen et al. (2019), where they encode time series as different types of images and predict several different candlestick patterns. Their model archived an accuracy of 90.7%,

proving that convolutional neural networks work very well for this task (Chen, Tsai, & Wang, 2019). Besides, there is some research using hybrid models. Kim and Kim (2019) use a hybrid model build of a CNN and LSTM to predict the stock price after 5 minutes using 30 minutes of data. Their research outperformed the single models and showed that prediction error could be drastically reduced (Kim & Kim, 2019). Also, Hao and Gao (2020) used a hybrid model consisting of CNN and LSTM. They predicted whether the stock would go up after one week or one month. For the former, they archived an accuracy of about 66.59%. They have even achieved 74.55% accuracy for the latter, although it is further in the future.

3 Methodology

In this chapter, research methods are presented and explained, including data processing and data analysis methods.

3.1 Data collection

For the stock market prediction, the data for four stocks have been retrieved from Eikon Reuters. Access to the platform is made available by the university. All stocks are from different industries and in the DAX, a stock market index consisting of Germany's 30 major companies. Bayer was chosen as a pharmaceutical company, Daimler for the automotive industry, Deutsche Bank for the financial sector and SAP as a software company. All data ranges from November 2000 to November 2020 and is sampled daily. The Open, High, Low, Close (OHLC) price and the volume have been retrieved for the stocks. After acquiring data, the following step was to create technical indicators. These include different moving averages, bollinger bands and the MACD, which will be used as explanatory variables for the CNN. Additional features that have been created are the upper and lower shadow. The former is calculated by subtracting the close from the day's high price, and the latter subtracts the open from the lower price.

Furthermore, economic data has been added as features. Here, the gold price, as well as the foreign exchange of EUR/USD, have been added, which are as well daily sampled. The purpose

of adding economic data is to see whether it is useful to include additional information in the technical analysis.

3.2 Triple-Barrier Method

After creating the features, a labeling method is required. A simple approach for a machine learning model is trying to predict an asset's price at some fixed time in the future. It can also be approached by predicting discretized returns. As previously shown, this labeling method is generally used in academia and can be described as follows,

$$y_t = \begin{cases} -1 \; if \; r_{t,t+h} < threshold \\ 0 \; if \; r_{t,t+h} = threshold \\ 1 \; if \; r_{t,t+h} > threshold \end{cases} \quad with \; r_{t,t+h} = \frac{P_{t,t+h}}{P_t} - 1 \tag{7}$$

where y denotes the label, r the return, p the price, t the time and h the time horizon in the future. However, this method has two addressable problems. Firstly, the same threshold is applied regardless of the observed volatility. This can lead to the problem that thresholds are too far apart or too close together. Secondly, it is path-independent, meaning that only the return at the end matters. This is problematic as with this approach reality is not reflected. In the real-world, profit and loss thresholds will be set before the investment. If an intermediate return hits the profit thresholds, it will be realized as a profit, even if it would be a loss at the end of the fixed-time horizon, and vice versa. The triple-barrier method, named after the strategy's two horizontal and vertical barriers, will be used to label the data to address the previously mentioned problems. The vertical barrier is pre-defined as the expiration limit of the position. The two horizontal barriers are defined by the profit and loss limits, resulting from the multiple of the dynamic volatility. The dynamic volatility is calculated as follows,

$$\sigma_t^2 = \lambda \sigma_{t-1}^2 + (1-\lambda) r_{t-1}^2 \; with \; \lambda = \frac{2}{(span+1)} \tag{8}$$

where r_t is the return and the smooting parameter λ controls for the decay. If the upper barrier is first to hit, the value is set to 1 and if the lower barrier is reached first, the value is set to -1.

If the purchase times out before either limit is broken and the vertical barrier is touched, the value is set to the return (López de Prado, 2018).

Figure 3: Labeling based on the tripe-barrier method

For this analysis, the time horizon taken into consideration is ten days. The upper and lower multiple is set to two. Furthermore, the approach of the labeling has been adjusted to make it a binary classification task. Therefore, if the vertical barrier is hit, the data was labeled according to the return as the time was up. If the return was slightly positive, the data was labeled as a 1 for buy and 0 if the return was negative. This was done because the classes would be highly unbalanced with a multi-classification approach due to data limitations.

3.3 Feature selection

Feature selection is a machine learning approach in which only a subset of the available features is used for the model. It is necessary because it is sometimes technically impossible to include all features as differentiation problems arise when a large number of features but only a small number of data points are available. Besides that, the model not only works faster with a reduced number of features but usually is also much more efficient.

Firstly, it was checked whether some variables show multicollinearity. If linear interdependencies correlate, there is the problem of multicollinearity. The analysis of the correlation coefficients, for example, serves to reveal multicollinearity. A very high positive or negative correlation indicates a strong relationship by the regression and thus multicollinearity.

Furthermore, this also shows that one variable can be expressed as a linear combination of other columns. This does affect not only linear models but also non-linear models (Bruce & Bruce, 2017). This is because the ambiguity does not lie in the interpretation but in the data and the model that can learn from it. The formula for calculating the correlation coefficient can thus be summarized as follows and is done on the raw dataset:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{\bar{x}})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{9}$$

After removing the multicollinear variables, the next process was to find the most useful features from the original features. The set S of k-combinations is calculated by the following formula, where n is the number of elements.

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!} \tag{10}$$

Doing this iteratively until all combinations have been tested and calculating the associated performance score, the best feature combination can be found. This provides the best performing set, which does not necessarily have to include all features. Due to computational resource limits, not all combinations could be calculated. Therefore, the set of combinations was limited for k from 4 to 6. This number was chosen because of the hierarchical cluster analysis, which is an algorithm that groups similar objects into one. The features within each cluster are broadly identical to each other. Furthermore, with this approach, combinations will be tested with only technical indicators and others containing the economic data as well.

3.4 Fractional Calculus

After having the final features, the next step is to verify that the time series is stationary. In finance, it is common to find non-stationary time series. However, it is a necessary property for inferential analysis. If data is not stationary, all assumptions made based on that the data is independent and identically distributed (IID) would be wrong. For time series modeling weak

stationary is satisfactory. A stochastic process $(x_t)_{t\epsilon\mathbb{T}}$ is stationary if the following conditions hold:

1. *the expected value* E *is constant, i.e. for all* $t \in \mathbb{T}$ *applies* $\mathrm{E}(x_t) = \mu$
2. *the variance* Var *is finite, i.e. for all* $t \in \mathbb{T}$ *applies* $Var(x_t) < \infty$
3. *the autocovariance does not depend on the shift* s*, i.e. for all* s, t_1, t_2 $\mathbb{T}$ *applies* $cov(x_{t_1}, x_{t_2}) = cov(x_{s+t_1}, x_{s+t_2})$

Most finance papers attempt to recover stationary time series by computing its return or by integer differencing which comes at the expense of unnecessarily removing too much memory from the original time series. One way to counteract this problem is to apply fractional differentiation. This method aims to find the minimum amount of differentiation *d* to make the time series stationary while preserving as much memory as possible. So, a fractionally differenced time series $\tilde{X}_t$ retains with the real non-integer differencing d memory of the original series but also is stationary.

$$\tilde{X}_t = \sum_{k=0}^{\infty} w_k X_{t-k} \tag{11}$$

$$X = \{X_t,\ X_{t-1}, X_{t-2}, \dots,\ X_{t-k}\} \tag{12}$$

with weights ω

$$\omega = \left\{1, -d, \frac{d(d-1)}{2!}, -\frac{d(d-1)(d-2)}{3!}, \dots, (-1)^k \prod_{i=0}^{k-1} \frac{d-i}{k!}, \dots\right\} \tag{13}$$

$$\omega_k = -\omega_{k-1} \frac{d-k+1}{k} \; for\; k \geq 1 \; and \; \omega_0 = 1 \tag{14}$$

However, as the weights converge asymptotically to zero with this method, the fixed-width window fractional differentiation with control for weight loss will be used instead of the expanding window fractional differentiation. This approach drops weights and the according to datapoint, if it falls below a given threshold τ.

$$\widetilde{\omega}_k = \begin{cases} \omega_k \; if \; \omega_k \geq \tau \\ 0 \; if \; \omega_k < \tau \end{cases} \tag{15}$$

The fixed window approach's with control for weight loss benefits are (i) it overcomes the negative drift caused by the expanding window, (ii) it creates a stationary time series, but it is not gaussian and (iii) it also remains excess skewness and kurtosis (López de Prado, 2018).

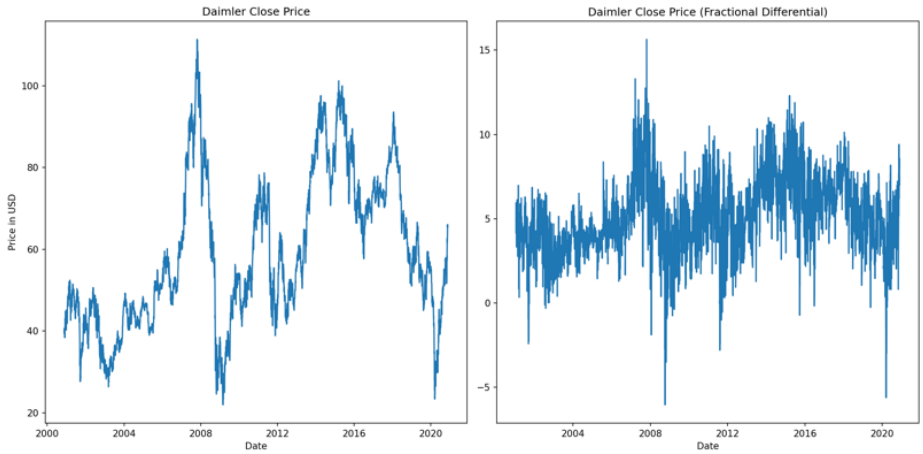

Figure 4: The left plot shows the close price of Daimler. The right plot shows the same plot fractional differentiated. Here, we also see that the time series starts later in time as we dropped weights below the threshold 0.001.

With the help of the ADF-test, variables can be tested for stationarity and the differentiation factor d can be found**,** which leads to a stationary time series with maximum memory preservation (López de Prado, 2018). The ADF Test expands the DF test equation to include high order regressive process in the model.

$$y_t = c + \beta_t + \phi_1 \Delta Y_{t-1} + \phi_1 \Delta Y_{t-2} + \cdots + \phi_p \Delta Y_{t-p} + \varepsilon_t \tag{16}$$

The null hypothesis of the ADF-test states that the time series is stationary. In contrast, the alternative hypothesis states that you reject the null hypothesis and infer that the time series is not stationary. Figure 5 below shows that it is possible to achieve stationarity without giving up all memory.

In this research, the ADF-test is used to check whether the time series is stationary or not. If the null hypothesis is not rejected, the time series will be fractionally differentiated. Therefore, the coefficient d starting at 0, is increased by 0.05 and the ADF-test is applied on the differentiated time series. If the null hypothesis still is not rejected, this process is iteratively applied until,

with 95 % confidence, the null hypothesis can be rejected. The critical value for the ADF-test at 5% is about -2.86. Afterward, the input data for the model has to be created.

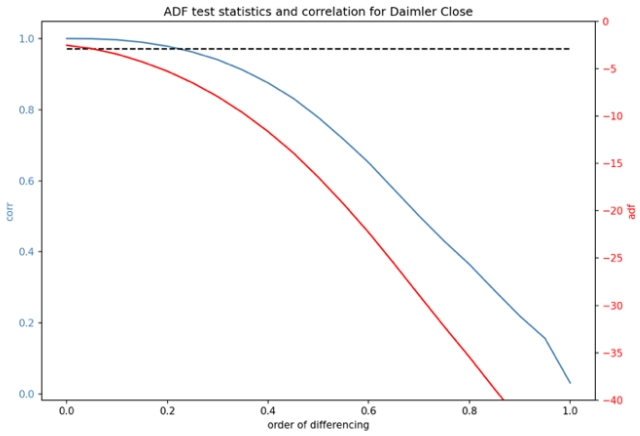

Figure 5: The horizontal dashed line in black marks the 5% interval for the ADF-test(right y-axis). The x-axis shows the differentiation factor. The left axis shows the correlation between the original and differentiated time series.

3.5 Gramian Angular Fields

It is essential to find an appropriate representation for the input data (Chollet, 2018). As the convolutional neural networks sees an input as array of pixels, the time series can be transformed. One of these methodologies is called Gramian Angular Field (GAF). It is a 2D representation of a time series, proposed by Wang and Oates (2015). GAF represent the time series data in a polar coordinate system and use different operations to convert these angles into symmetry matrices. It was proven that this approach performs better in time series problems than a 1D representation of the data.

The first step of creating a GAF matrix is to normalize the time series *X* into the range of [-1; 1] or [0; 1]. The following equation from Wang and Oates (2015) shows how to normalize data:

$$\tilde{x}_{i\,[-1;1]} = \frac{(x_i - X_{max}) + (x_i - X_{min})}{X_{max} - X_{min}} \quad or \quad \tilde{x}_{i\,[0;1]} = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{17}$$

where $\tilde{x}$ denotes the normalized data. The scaled time series is then transformed into polar coordinated. The value is interpreted as an angular cosine and the time stamp t_i as a radius *r*, where *N* is a constant factor to regularize the polar coordinate system's length.

$$\begin{cases} \phi = arcos(\tilde{x}_i), -1 \leq \tilde{x}_i \leq 1, \tilde{x}_i \in \tilde{X} \\ r = \frac{t_i}{N}, t_i \in \mathbb{N} \end{cases} \quad (18)$$

Afterward, two possible GAFs can be created, the Gramian Angular Summation Field (GASF) and the Gramian Angular Difference Field, where *I* is the unit vector (Wang & Oates, 2015).

$$GASF = \left[\cos\left(\emptyset_i + \emptyset_j\right)\right] = \tilde{X}' \times \tilde{X} - \sqrt{I - \tilde{X}^2}' \times \sqrt{I - \tilde{X}^2} \quad (19)$$

$$GADF = \left[\sin\left(\emptyset_i + \emptyset_j\right)\right] = \sqrt{I - \tilde{X}^2}' \times \tilde{X} - \tilde{X}' \times \sqrt{I - \tilde{X}^2} \quad (20)$$

In this research, both methods, summation as well as difference, have been tested. To create the images, 30 consecutive trading days have been included, which corresponds to 1 ½ month. This number was chosen because this period should be sufficient to form chart patterns. Each GAF results in a 30 x 30 array (width and height of each image). In this study, there are approximately about 4775 observations for each stock.

3.6 Evaluation

When developing neural networks, it is crucial to check their validity. A particular method for evaluating the performance of the model is cross-validation. The most commonly used version of cross-validation is where the model data is divided into two mutually exclusive sets, a larger one, the training set, and a smaller one, the test set. The larger amount of data is used to build a model, while the smaller amount of data is used to validate the model by applying the model to the smaller amount of data and comparing the results with the actual value. This process is repeated with different subsets until each object of the dataset has been used once for testing (Müller & Guido, 2017).

However, for cross-validating time-series, a different approach has to be introduced. A naïve approach would lead to the problem that the past is predicted with future data. Thus, it would allow the model to learn from the future so that it would overfit easily ((Brownlee, 2020). One approach is the walk-forward validation. The idea behind the approach is to use continuous cross-validation and avoid the linking between observations. The data for this has been divided

into ten walks. The model is trained for 800 days and tested over the following 400 days in each walk.

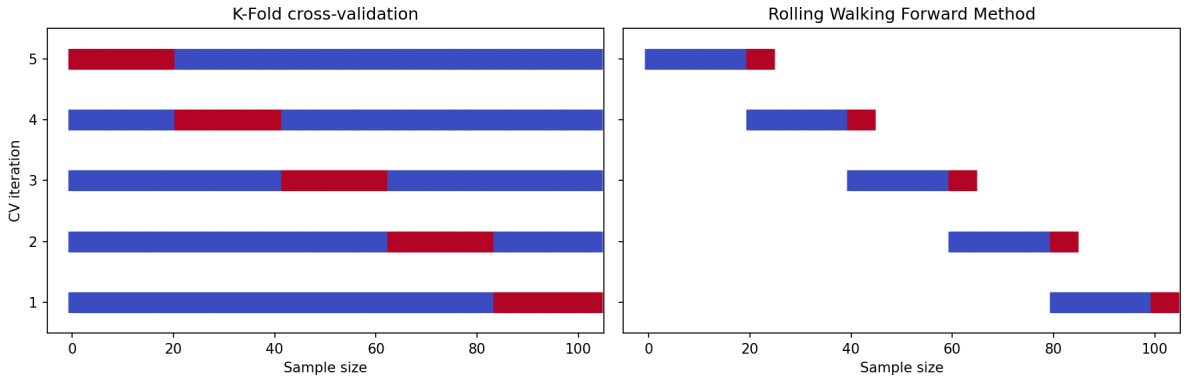

Figure 6: Visualizing the k-fold cross-validation and walking-forward method.

This stock trend prediction has a binary target variable and quantitative measures can be derived from the output. The output of neural networks can be divided into four categories:

- True Positive (TP): The rising price was determined correctly.
- False Positive (FP): A rising price was predicted, but it fell.
- True Negative (TN): The falling price was determined correctly.
- False Negative (FN): The falling price was predicted, but it rose.

As the labels are fairly balanced, accuracy will be used as the evaluation metric. Accuracy represents the fraction of samples the model gets right and is calculated as follows:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{21}$$

This simple evaluation method will be used because both classes can be equally weighted since profits and losses can be made in both cases. Especially in the case of high accuracy, the results should be checked. It happens that a model also always predicts the same class. Therefore, we will use a zero predictive power model as the baseline classifier. A zero predictive model only predicts the more frequently represented class in the dataset.

3.7 Hyperparameters

During the training process, weights within the model are automatically changed so that it is increasingly able to establish the desired relationship between features and prediction. What

remains unchanged during the training process is the basic architecture of the model. The parts, which remain the same throughout the training process, are called hyperparameters. Since the hyperparameters significantly influence model performance, their optimization is an essential step in model development and can substantially improve prediction accuracy. There are different approaches for optimizing hyperparameters, but only two of the most used techniques will be covered in this work.

One of them is the grid search, which means trying all possible combinations of the hyperparameters and the set of parameters with the best performance wins (Müller & Guido, 2017). The drawback of grid search that the run time increases exponentially with an increase in hyperparameters. If there are m hyperparameters, each taking the most n values, the growth is given by $O(n^m)$.

Another approach, the one used in this work, is the random search. It is a technique where a random combination of hyperparameters is chosen and helps find the near-optimal set. As in grid search, the best set in all iterations will be the output (Bergstra & Bengio, 2012). However, this approach's benefit is that with about 60 iterations, 95 % of the time, the best 5 % sets of parameters can be found, regardless of the grid size. If n is the number of iterations, it can be proven by the following equation, where p denotes the probability (Zheng, 2015).

$$1 - (1 - p)^n = confidence \rightarrow 1 - (1 - 5\%)^n = 95\%, n \approx 59 \quad (22)$$

3.8 CNN specifications

The constructed CNN consists of two convolutional layers, followed each by an activation function and a pooling layer, two fully connected layers and an output layer. Rather the convolutional layer is followed by a batch normalization layer or the fully connected layer by a dropout layer, which will depend on the hyperparameter tuning. Also, the dropout, as well as neuron values, are tuned. For the kernel, different sizes can be adapted: 3x3, 5x5 and 7x7. Decreasing the kernel size results in catching more details of the image. The proposed model will have a kernel size of 3x3 due to the small image size (30x30). The structure is very similar

to the LeNet structure because adding more layers would increase the model's complexity. Due to the low number of training data, a more complex structure could cause the model to overfit and reduce the accuracy on the test data.

Compiling the model, binary cross-entropy is used as the cost function (Brownlee, 2019). Cross-entropy is a cost function often used for classification tasks. The cost function indicates the deviation between the network's output and the desired result. The aim is to minimize this function,

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log\big(p(y_i)\big) + (1 - y_i)\log\big(1 - p(y_i)\big) \tag{23}$$

where y denotes the label and $p(y)$ the predicted probability of being that class for all point N. As an optimizer adaptive moment (Adam) is used (Kingma & Ba, 2015). It is a method to calculate the learning rates for each parameter adaptively. The method stores the mean of the previous gradients v_t and the average of the previously squared gradients s_t. These are estimates of the first and second momentum. At t_0 the vectors v_0 and s_0 are initialized as 0 and w_0 is the initial parameter vector. The momentum is the sum of the two previous gradients, which are stored in a vector. This vector speeds up the gradient process when the gradient changes little, and vice versa. The variables β_1, β_2 indicate by how much the parameters should decay.

$$v_t = \beta_1 v_{t-1} + (1 - \beta_1) g_t \tag{24}$$

$$s_t = \beta_2 s_{t-1} + (1 - \beta_2) g_t^2 \tag{25}$$

To overcome that v_t and s_t tend towards zero, the first and second momentum will be estimated as follows:

$$\hat{v}_t = \frac{v_t}{1 - \beta_1^t} \tag{26}$$

$$\hat{s}_t = \frac{s_t}{1 - \beta_2^t} \tag{27}$$

The weight update w_t is performed as follows:

$$\Delta w_t = \frac{-\eta \hat{v}_t}{\left(\sqrt{\hat{s}_t} + \epsilon\right)} \tag{28}$$

$$w_{t+1} = w_t + \Delta w_t \tag{29}$$

To output the results the softmax function is used, which is defined as follows:

$$softmax(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{30}$$

As with sigmoid, the output lies in the interval of [0, 1]. The sum of all probabilities is always 1. The output layer with the softmax activation function contains as many neurons as classes are to be determined – in this case two. The output value indicates with which probability it lies in the specified class. Training the model, the number of epochs run, and the batch size will be tuned as well. Increasing the number of epochs will boost the model until a turning point after which the performance gets worse again. When all batches have passed through the neural network once, an epoch is complete.

4 Results

In this chapter, the results of the data analysis are presented. All analyses in this research were performed using TensorFlow in *Python 3.8.5*.

4.1 Data Preprocessing

After downloading the data, the features have been created. A list of all 22 features can be found in table 1.

Table 1: List of features for all stocks

All features							
1	Close	7	SMA 50-days	13	EMA 12-days	19	Close Gold
2	Open	8	EMA 10-days	14	EMA 26-days	20	High Foreign Exchange (EUR/USD)
3	Low	9	EMA 50-days	15	MACD	21	Low Foreign Exchange (EUR/USD)
4	High	10	SMA 20-days	16	Signal line	22	Open Foreign Exchange (EUR/USD)
5	Volume	11	Upper band	17	f01 (upper shadow)		
6	SMA 10-days	12	Lower band	18	f02 (lower shadow)		

The next step was calculating the correlation coefficients for the exploratory variables. To visualize the correlation matrices, a heatmap has been used (see appendix, attachment 1). As we can see, there exists multicollinearity between features, which belong to the same group. The hierarchical clustering of features has been visualized as a dendrogram to show the features, which belong to the same group (see appendix, attachment 3). To overcome multicollinearity, features with a correlation coefficient of higher than 98% have been dropped. This reduced the number of features for Daimler from 22 to 9 and for the other stocks from 22 to 8. Afterward, the ADF-test was applied to find the minimum differentiation factor for a fractional differentiated time series that is dropping weights below 0.001 (see table 2).

Table 2: Shows the differentiation factor after removing multicollinear features. "x" denotes that the feature was not in the feature list of the associated stock.

	Close	Close_gold	High_fx	MACD	SMA_50	Signal_Line	Volume	f01	f02
Daimler	0,05	0,30	0,10	0,00	0,75	0,00	0,00	0,00	0,00
Bayer	0,20	0,30	0,10	0,00	x	0,00	0,00	0,00	0,00
SAP	0,20	0,30	0,10	0,00	x	0,00	0,00	0,00	0,00
Deutsche Bank	0,20	0,30	0,10	0,00	x	0,00	0,00	0,00	0,00

Having all features stationary, the data is normalized in the interval [-1;1] before creating the input images (see appendix, attachment 4). To find the best set of combinations for the features left, the score for all possible combinations with 4, 5, and 6 features were calculated. These numbers have been chosen due to the clusters in the previous mentioned dendrogram. In the following table 3 the score including all features from table 1 as well as the best scores achieved with a reduced number of features is shown. All models performed better with a reduced number of features

Table 3: Comparing model performance for all features and the best set of features in accuracy.

Stock	Method	Features	Accuracy (%)
Bayer	Summation	All features included	0.502
Bayer	Difference	All features included	0.509
Bayer	Difference	Close, Volume, MACD, Signal Line, f01, f02	0.526
Daimler	Summation	All features included	0.495
Daimler	Difference	All features included	0.494
Daimler	Summation	Volume, Signal Line, f01, Close Gold,	0.529

Deutsche Bank	Summation	All features included	0.492
Deutsche Bank	Difference	All features included	0.504
Deutsche Bank	Summation	Close, MACD, Close Gold, High FX	0.529
SAP	Summation	All features included	0.491
SAP	Difference	All features included	0.507
SAP	Summation	Close, MACD, f01, Close Gold, High FX	0.522

4.1 Final CNN models

After having the best set of features shown above, these features have been used to find the optimal hyperparameters within the grid (see table 4) with the randomized search. The results obtained from the randomized search are shown in table 5.

Table 4: Grid for the randomized search

Hyperparameters of the grid			
Pooling:	max, average	Dropout	0, 0.1, 0.2, 0.3, 0.4, 0.5
Activation func.:	relu, sigmoid, tanh	Batch size	1, 16, 32, 64
Learning rate:	0.001, 0.0001	Neurons	32, 64, 128, 256
Batch norm.	True, False	Epochs	10, 20, 30, 50

Table 5: Best set of parameters through random search

Stock	Pooling	Activation	Learning r.	Batch norm.	Dropout	Batch	Neurons	Epochs
Bayer	average	sigmoid	0.0001	False	0.3	32	32	50
Daimler	max	tanh	0.0001	False	0	16	256	10
Deutsche Bank	max	relu	0.001	False	0.5	32	128	10
SAP	average	sigmoid	0.001	False	0.1	16	64	50

With the hyperparameters from table 5, the accuracies shown in table 6 have been achieved. The final score is compared to the zero predictive power models. The results show that the parametric optimization had little impact on improving model performance. Moreover, in three out of four cases, the proposed model exceeds the zero predictive models.

Table 6: Final model performance compared to a zero predictive model

Stock	Zero predictive power	Accuracy
Bayer	0.554	*0.569*
Daimler	0.528	*0.541*
Deutsche Bank	0.501	*0.543*
SAP	*0.546*	0.544

5 Discussion

In this chapter, the previous work is summarized. It then describes the implications, limitations and conclusions of this study. Suggestions for future research are also presented.

5.1 Conclusion

Forecasting the direction of financial time series needs an outright different framework compared to regular machine learning classification tasks such as loan prediction or classifying images of dogs. To incorporate the existing heteroskedasticity in time series, the data was labelled using this triple-barrier method. It was predicted whether the share price of a company would fall or rise within ten days. Moreover, to make the data stationary, the concept of fractional differentiation was applied. This helps that all assumptions made based on IID hold. Despite the fact that all the upper preprocessing steps were incorporated, the research question cannot be answered unambiguously. Upon closer inspection of the results, three out of four models performed better than the baseline classifier, but with only a small difference in two cases. To have more reliable results the same methodology has been applied to stocks of different industries. This was based on the assumption that stock prices from different sectors have performed differently over time and different results could be obtained, which was not the case. Further the hyperparameter tuning had little impact on the results. This possibly happened because the basic structure was based on successful literature. In real-life trading the risk of losing money with this model would be too high. Furthermore, this model gives no insight about how the bets should be sized, which could lead to the problem that sometimes we win a little and lose a lot, and vice versa.

5.2 Implications

Hopefully, this research showed that literature in this area is very misleading. To the best of my knowledge, almost all literature approaches use a fixed time horizon for labelling the data. Furthermore, they do not attempt to make the time series stationary or violate the model by predicting the past with future data. This study tried to overcome all these fallacies and build as accurate a model as possible to the real world. Furthermore, the LSTM model, which has proven to perform extraordinarily, has not been used as it learns to shift the time series and, therefore, usually highly overfits. An example, which was found online, can be seen in the

appendix, attachment 5 figure 1. In conclusion, in live trading, these models would perform poorly as accuracy was only slightly above 50%. It should also be mentioned that there is a high magnitude of uncertainty in stock markets, which makes it difficult to forecast stock trends.

All in all, in the author's opinion, technical indicators are no longer good predictors, as they are twined together and have become unprofitable to use. The findings indicate that the DAX in which the stocks are located is efficient, at least efficient enough that technical analysis does not work on the stocks. They might have been useful in the past, but the trading environment has changed drastically. Also, stock markets are affected by many factors such as economic environment, political policy and natural factors, which have not been considered in this analysis. The best feature sets include in three out of four cases economic data, showing that adding additional information is relevant in predicting stock trends. So, using the foreign exchange and gold price might have been a step in the right direction, nevertheless, are in a strong interrelation with stock prices and not certainly independent. According to the author's opinion, the focus should not be forecasting or classifying stock prices and instead finding out what influences them.

5.3 Future work

This was one of the first research in image recognition for trading and according to this, it only gives a broad idea of the topic. Further research could relate to this topic using intraday data. Learning pattern from daily data is abstract and having intraday data, the model can learn more details about the market. Additional, if more data is available, the data can be classified in multiclass: rise, fall, hold. Also, the threshold can be adjusted to see the performance on different levels. Furthermore, as we have seen, economic data has high feature importance. Therefore, it might be interesting to include other economic data such as interest rates or even use alternative data, such as satellite images of trucks. Other future research could be using the same approach to markets, which are not highly quant traded or to cryptocurrencies.

6 Appendix

Attachment 1

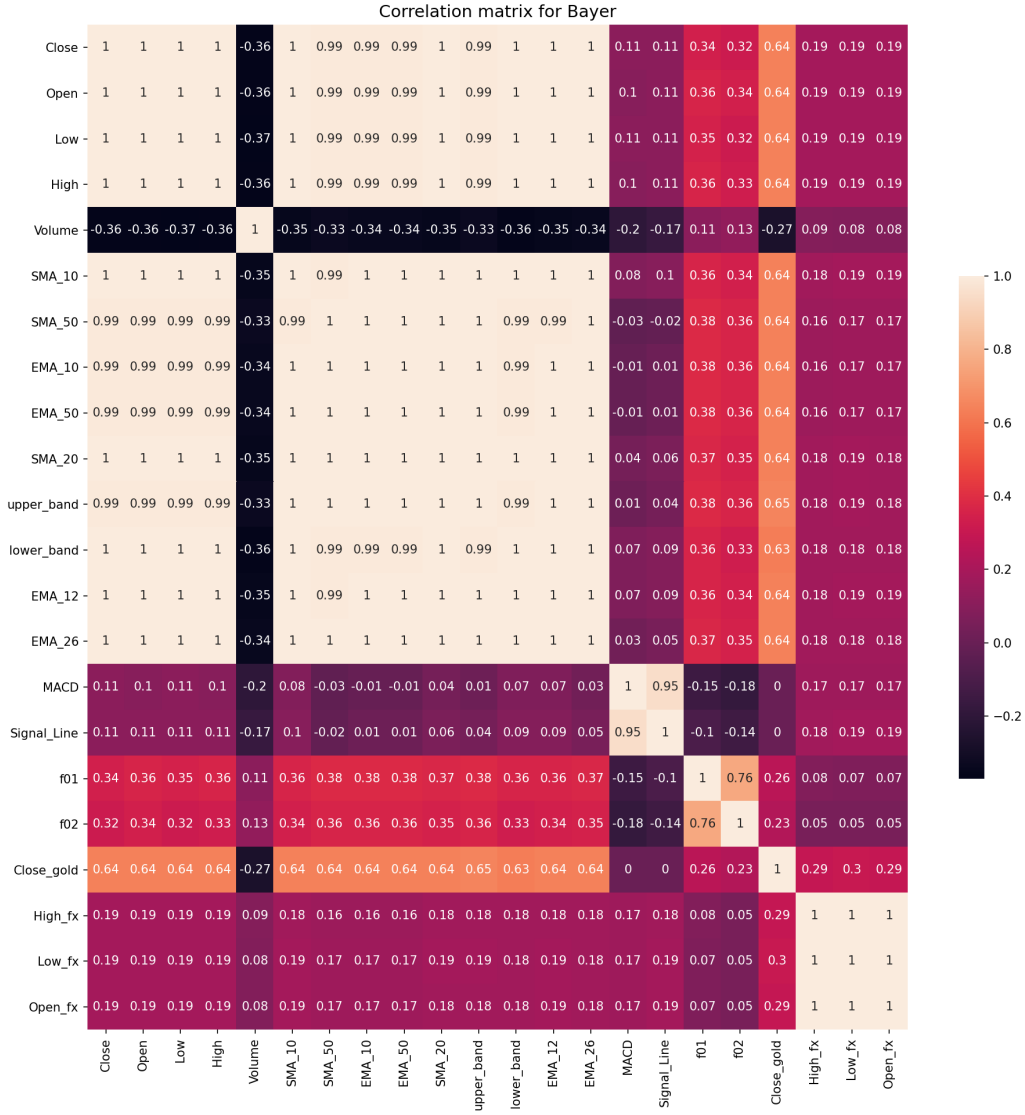

Figure 1: Corelation matrix including all features for Bayer.

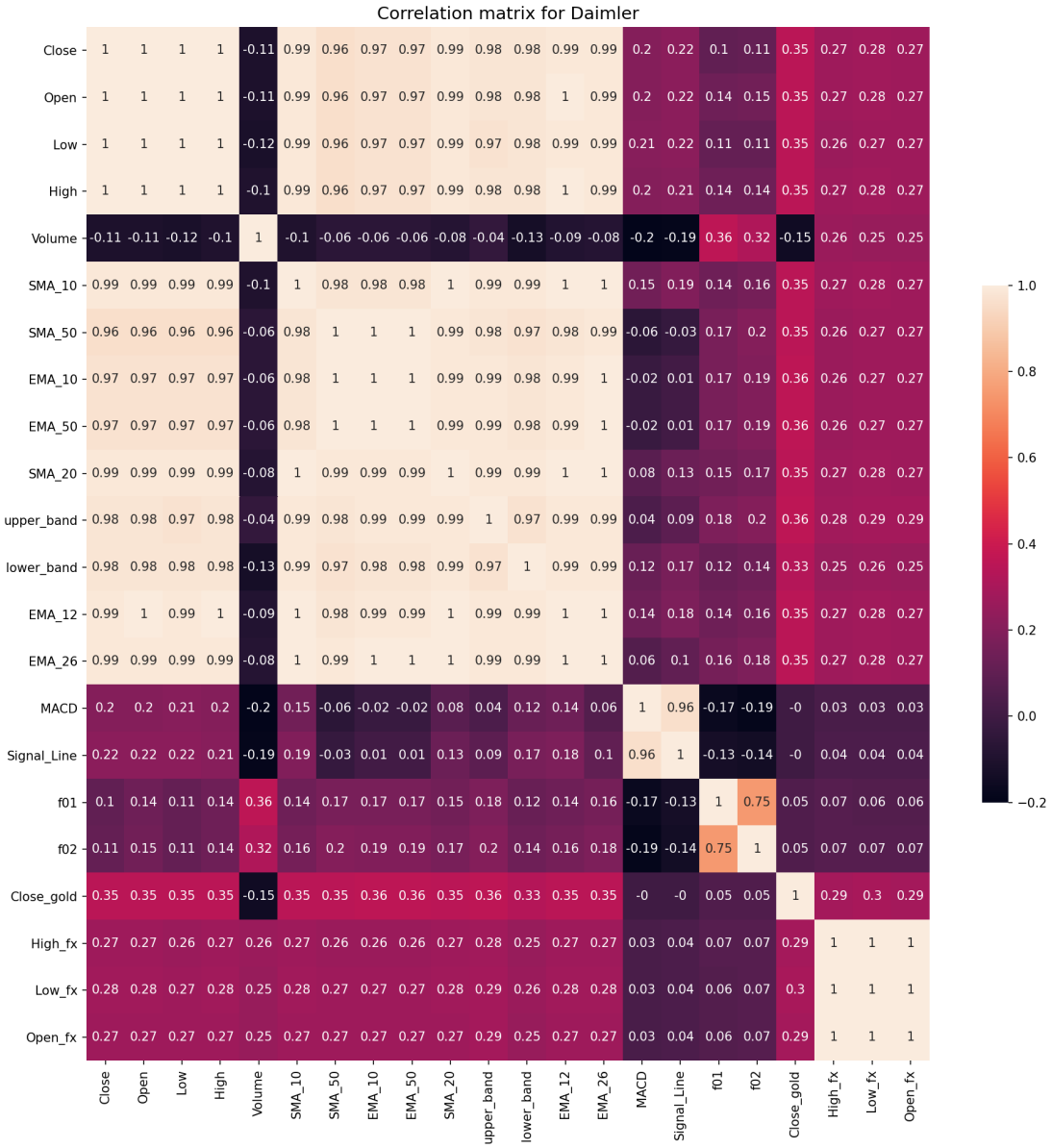

Figure 2: Correlation matrix including all features for Daimler.

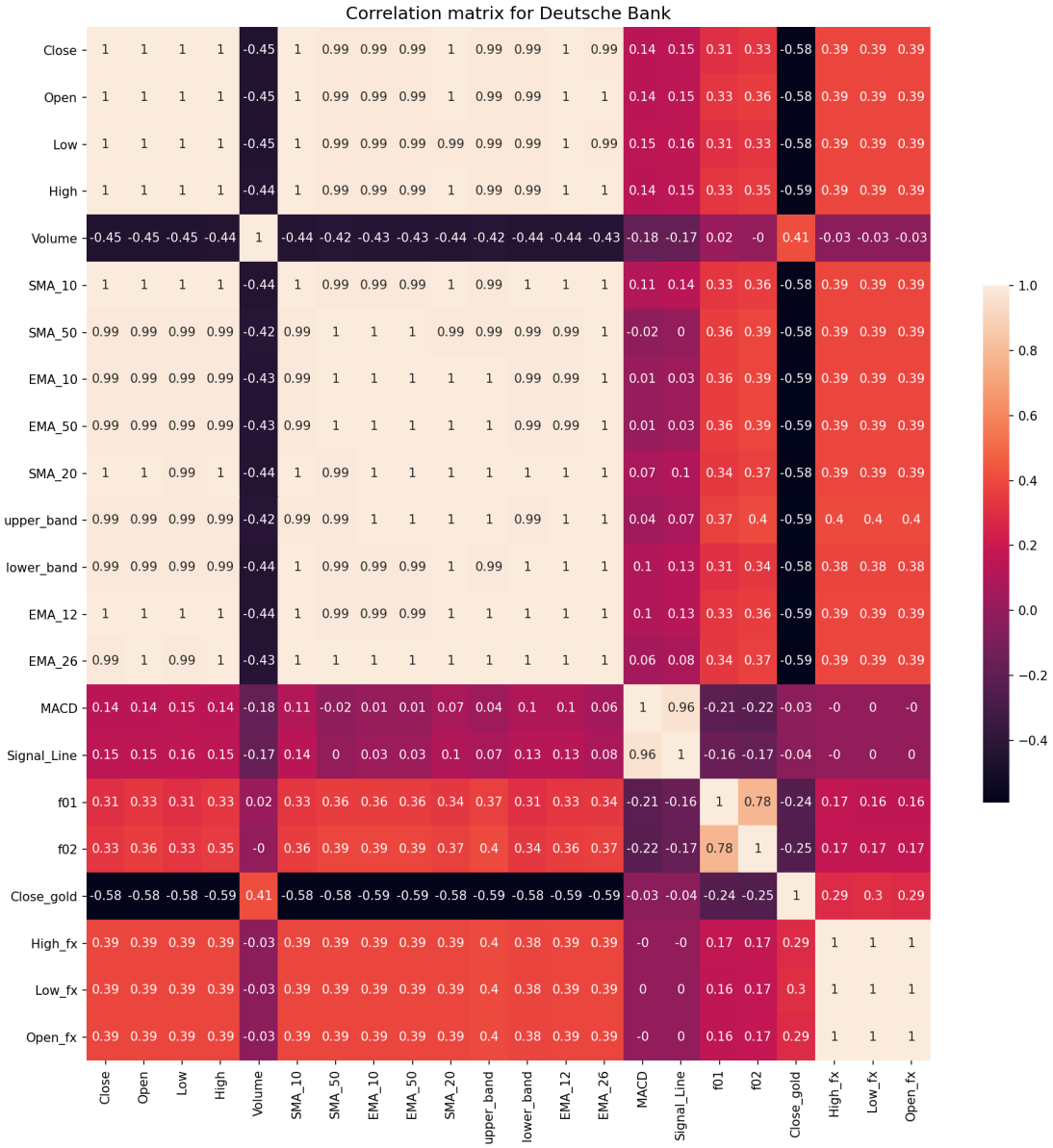

Figure 3: Correlation matrix including all features for Deutsche Bank.

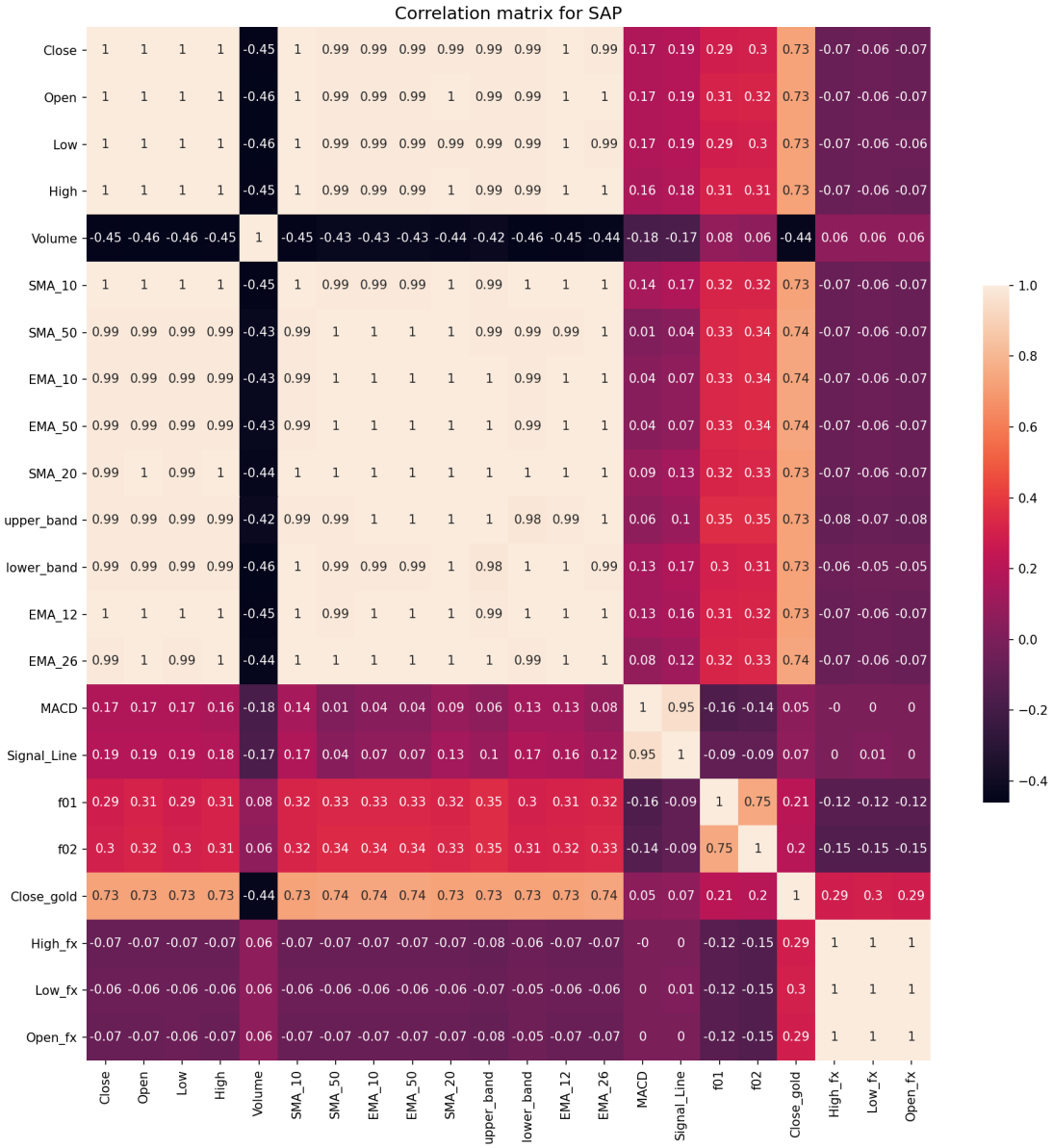

Figure 4: Correlation matrix including all features for SAP.

Attachment 2

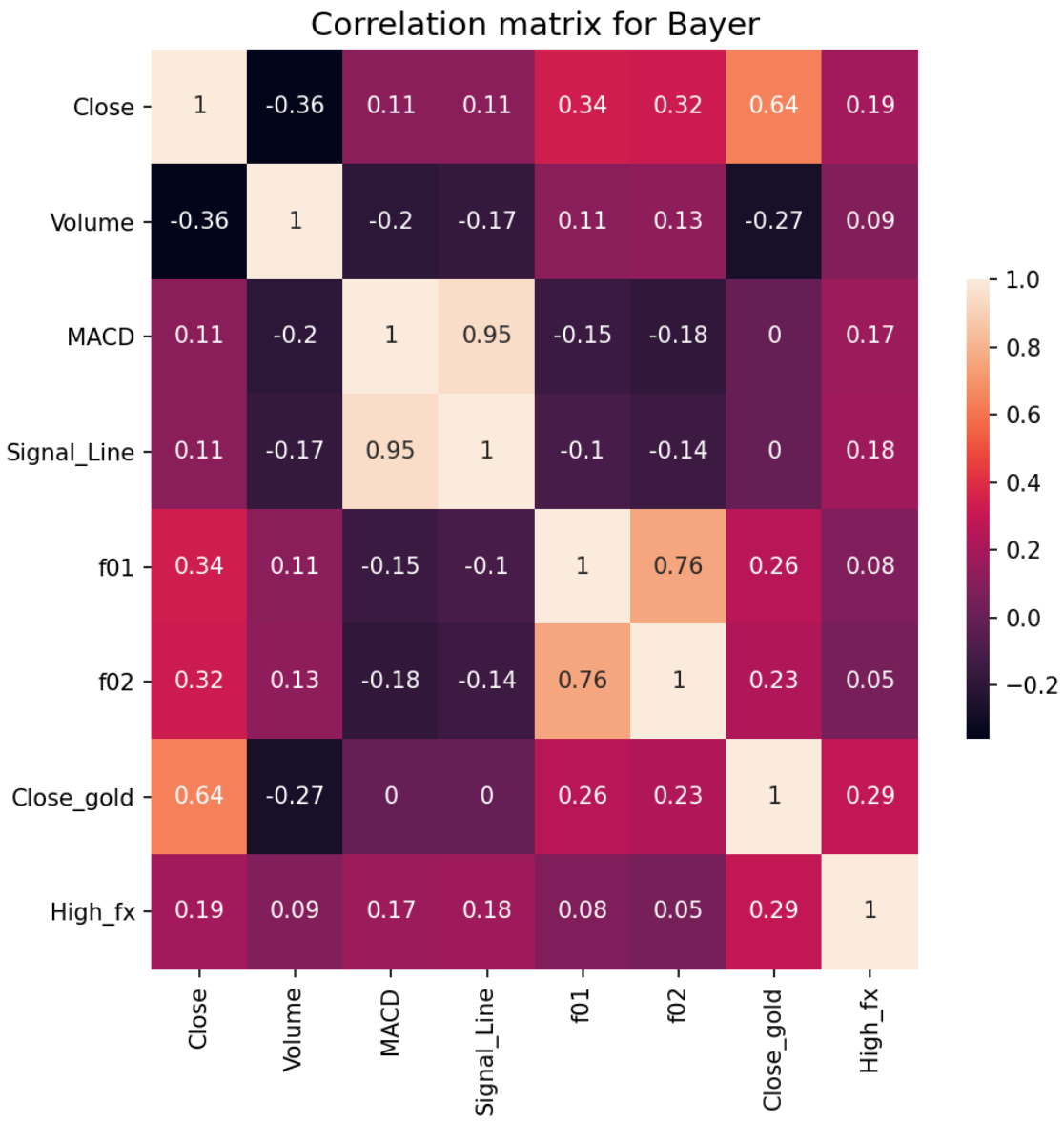

Figure 1: Correlation matrix after dropping multicollinear variables for Bayer.

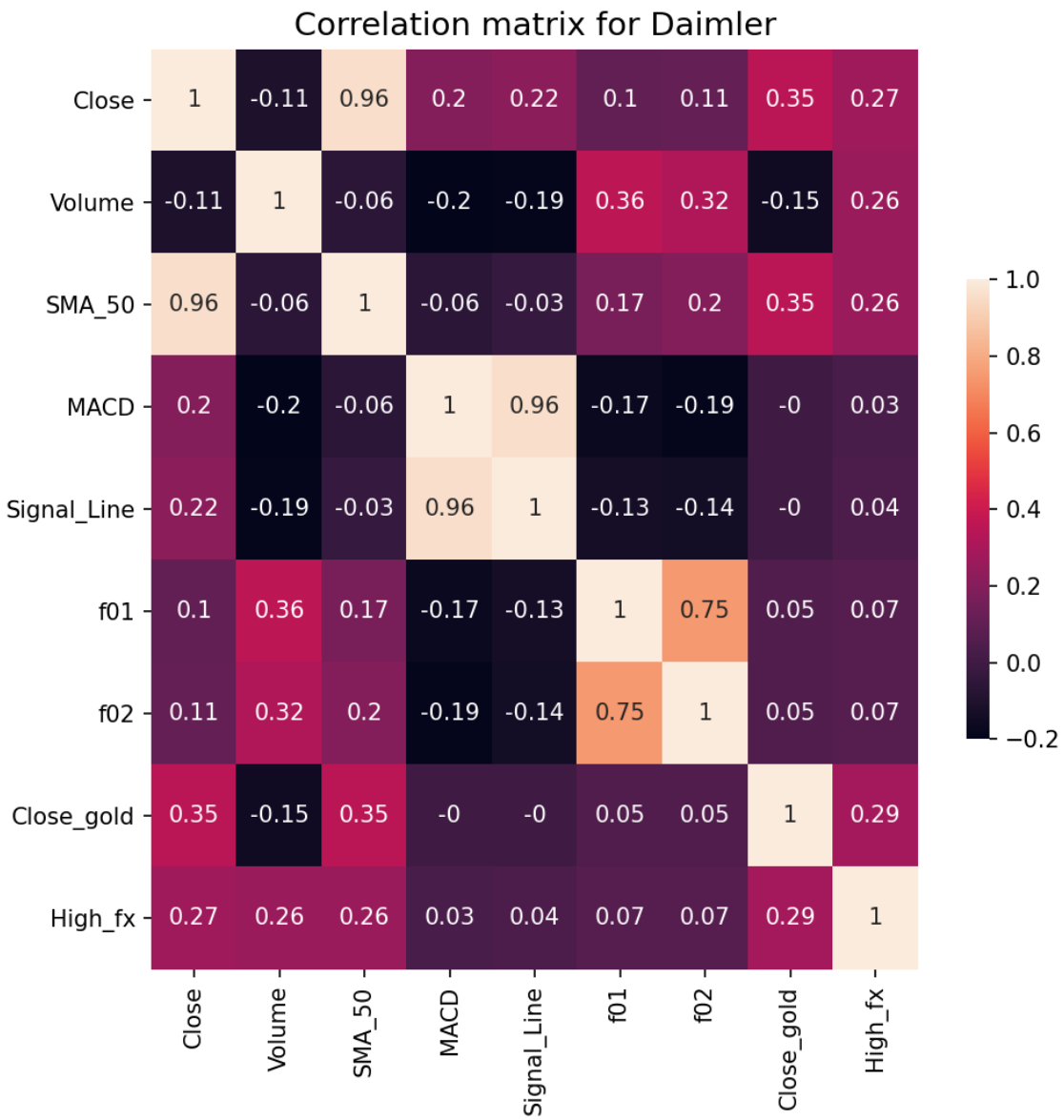

Figure 2: Correlation matrix after dropping multicollinear variables for Daimler.

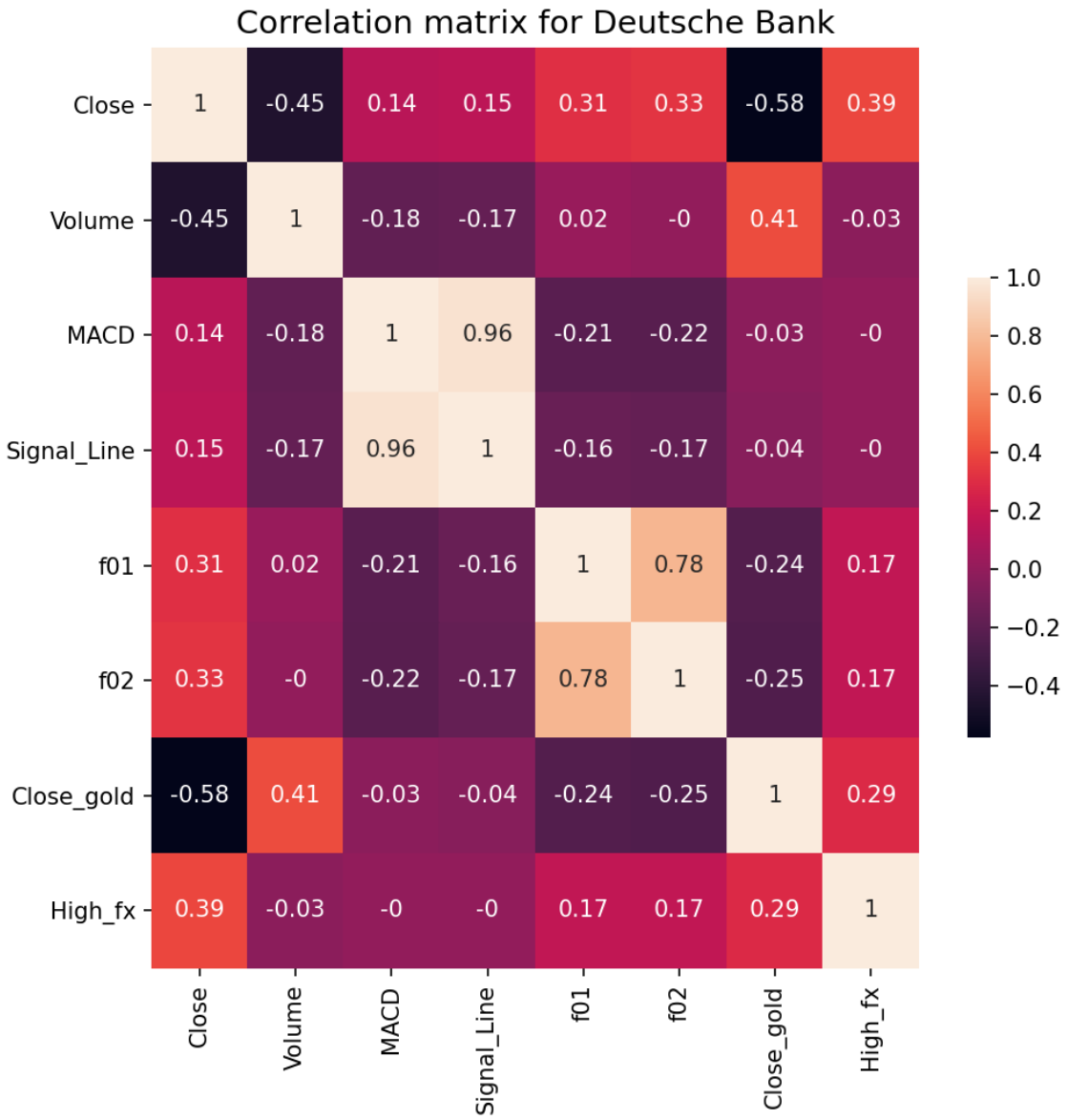

Figure 3: Correlation matrix after dropping multicollinear variables for Deutsche Bank.

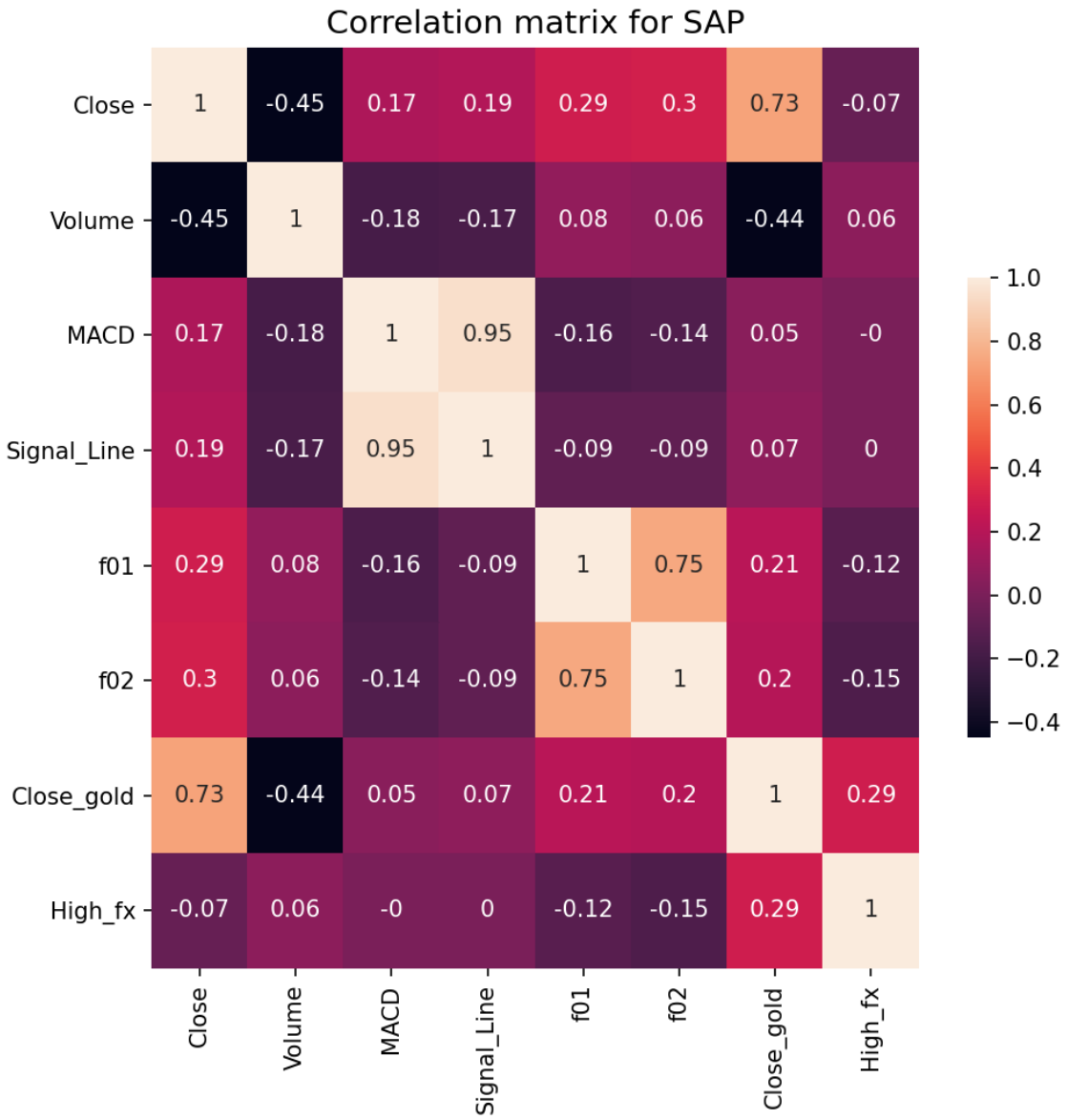

Figure 4: Correlation matrix after dropping multicollinear variables for SAP.

Attachment 3 (insert caption)

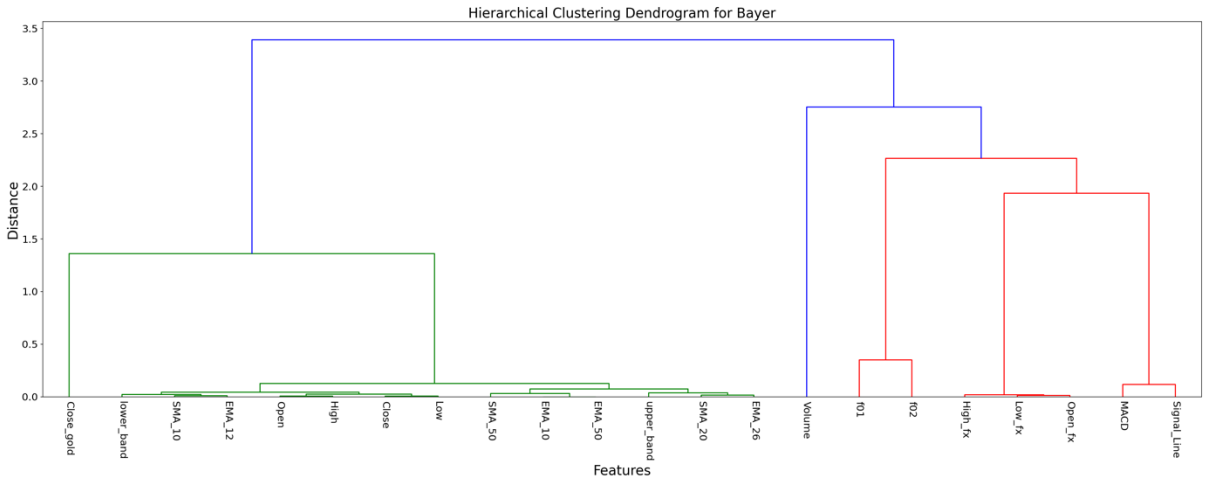

Figure 1: The hierarchical cluster for Bayer's features, visualized as a dendrogram.

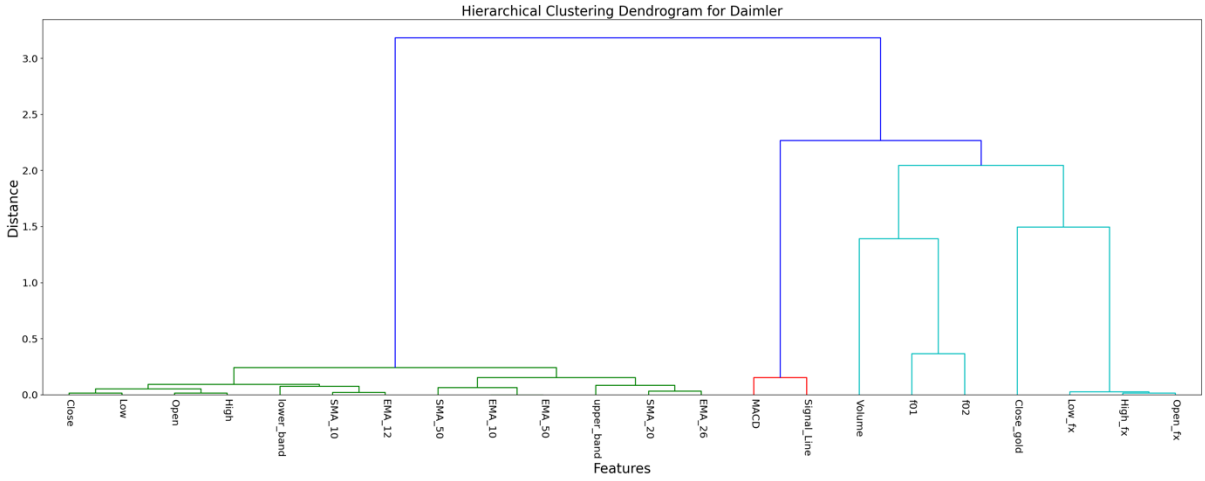

Figure 2: The hierarchical cluster for Daimler's features, visualized as a dendrogram.

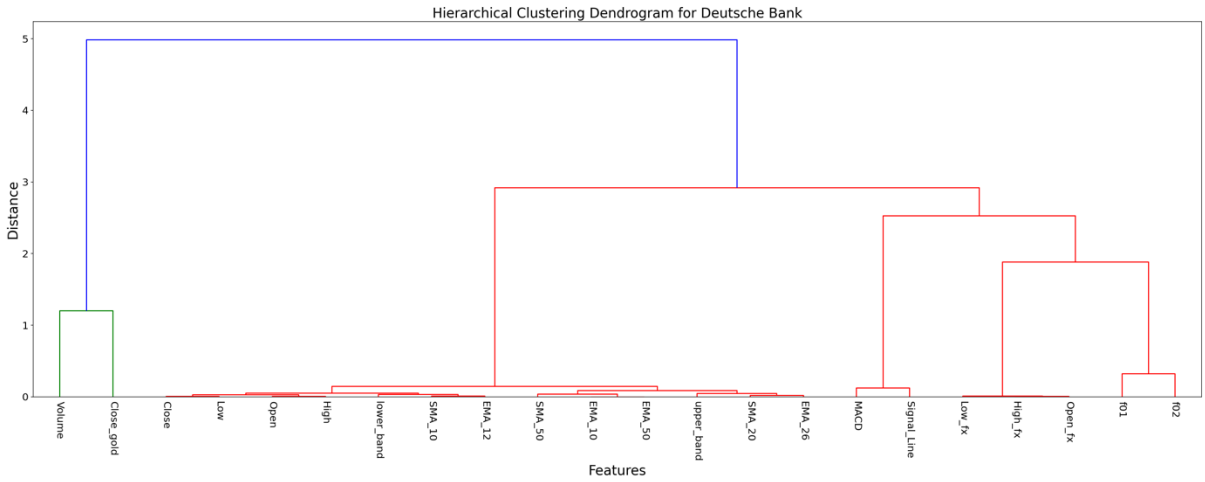

Figure 3: The hierarchical cluster for Deutsche Bank's features, visualized as a dendrogram.

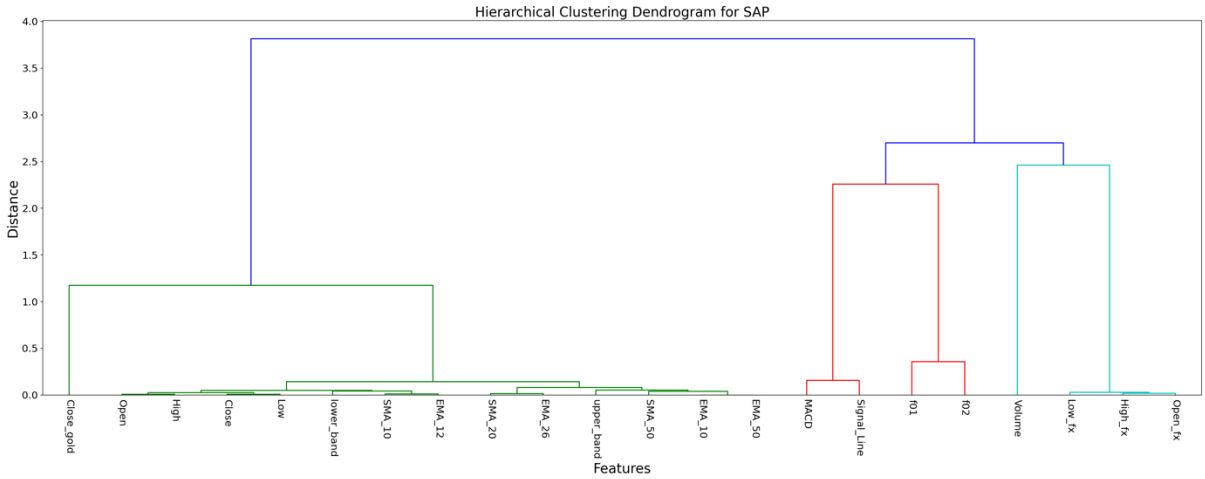

Figure 4: The hierarchical cluster for SAP's features, visualized as a dendrogram.

Attachment 4

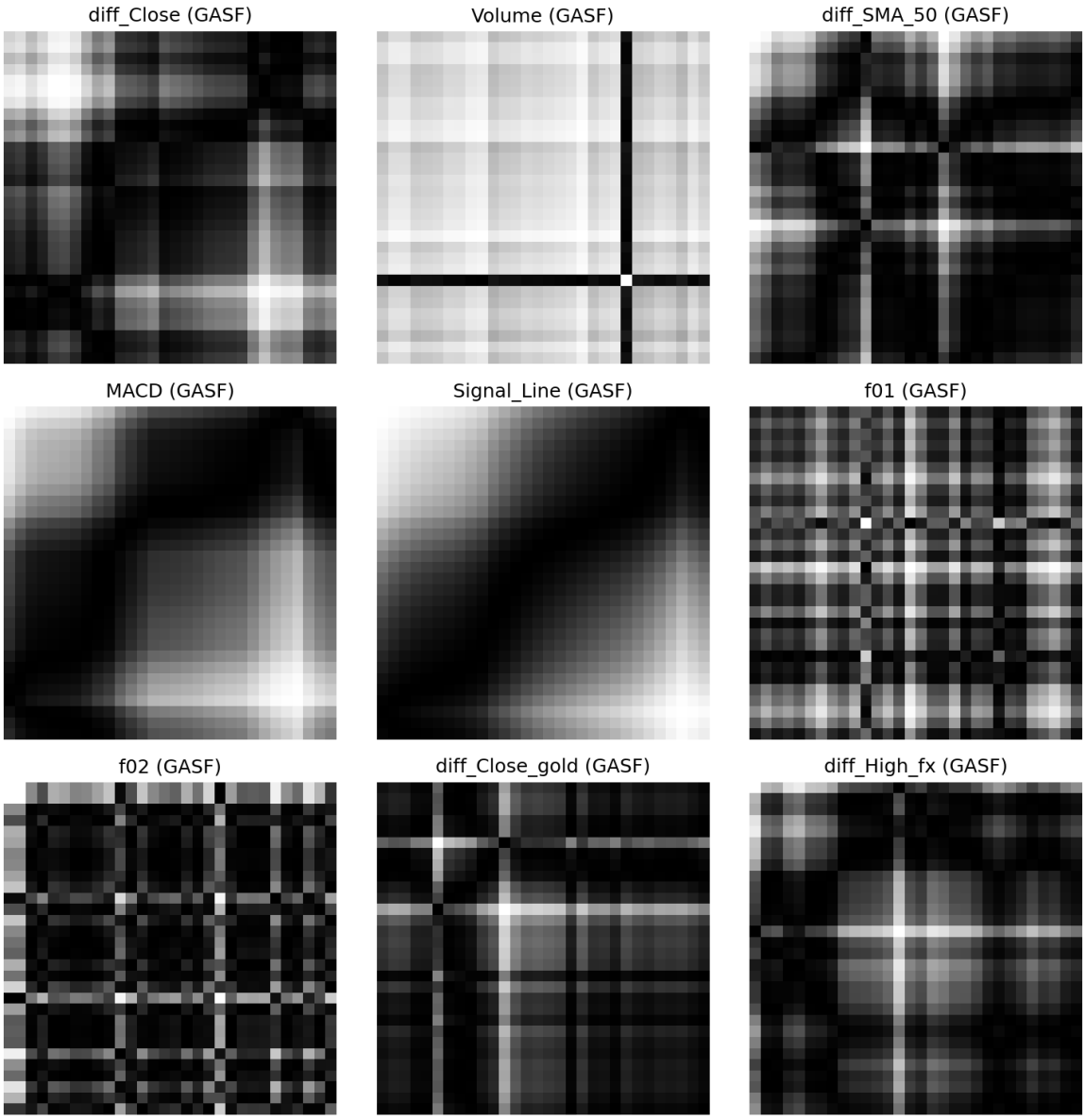

Figure 1: Plots the GASFs for Daimler. The "diff_" before some feature names shows that this feature has been fractional differentiated.

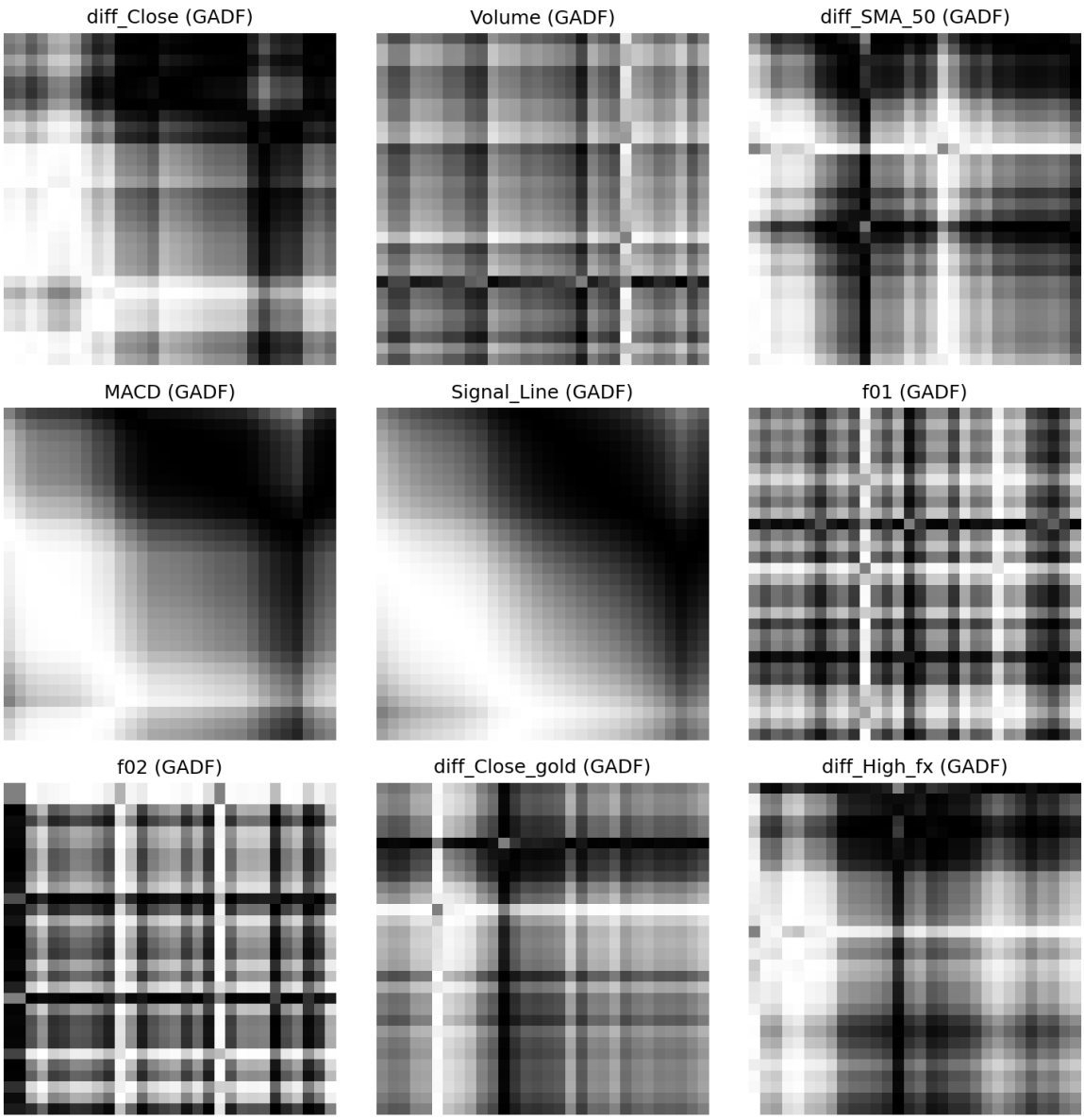

Figure 2: Plots the GADFs for Daimler. The "diff_" before some feature names shows that this feature has been fractional differentiated.

Attachment 5

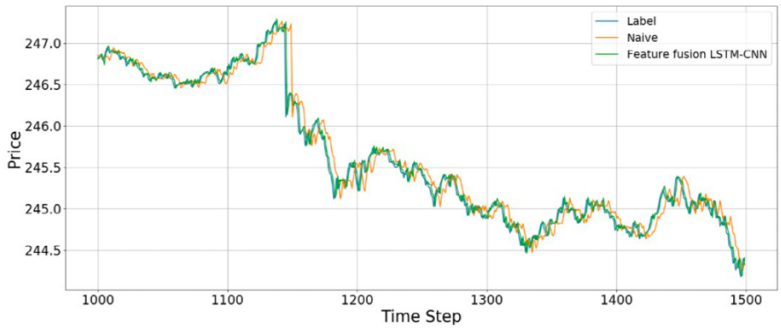

Fig 17. An example of predicting stock prices using the feature fusion LSTM-CNN model with a test dataset of between 1,000 and 1,500 data points. The input data are candlebar chart and stock time series.

Figure 1: This plot shows the prediction of a LSTM model. It shows how the model learns to shift the time series and make good predictions. The image is from the paper "Forecasting stock prices with a feature fusion LSTM-CNN model using different representations of the same data" from Kim and Kim (2019).