BACHELOR'S THESIS 2021

Predicting price trend reversals using machine learning techniques

Nicolo Ridulfo, Li Zhu

ISSN 1651-2197
LU-CS/HBG-EX: 2021-06

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

KANDIDATARBETE
Datavetenskap

LU-CS/HBG-EX: 2021-06

Predicting price trend reversals using machine learning techniques

Nicolo Ridulfo, Li Zhu

Predicting price trend reversals using machine learning techniques

Nicolo Ridulfo
ni0381ri-s@student.lu.se

Li Zhu
li0301zh-s@student.lu.se

June 21, 2021

Bachelor's thesis work carried out at
the Department of Computer Science, Lund University.

Supervisor: Marcus Klang, Marcus.Klang@cs.lth.se

Examiner: Pierre Nugues, Pierre.Nugues@cs.lth.se

Abstract

The prices of stocks on the stock market are in a constant state of oscillation. However, these oscillations will peak or bottom out periodically, which we define as a reversal event. Through the use of machine learning, we can predict when these events may occur. In addition, these events may be transformed into buy and sell signals which can be exploited for profit or incorporated into a risk management framework. Therefore, predicting reversal events might be equally or even more important than knowing the exact direction the market is going to move tomorrow. In this thesis, we examined the performance of Logistic Regression (LR), k-Nearest Neighbor (kNN), Deep Neural Network (DNN), Convolutional Neural Network (CNN) and Long Short Term Memory (LSTM) in predicting past and future reversals. The dataset used was all stocks listed in the S&P 500. Our experiments show that neural networks greatly outperformed LR and kNN, with our LSTM model "revLSTM" achieving the best performance in most cases. We also evaluated the performance impact of adding additional features such as the distance to previous reversals. By incorporating previous reversals, the Matthew Correlation Coefficient (MCC) scores were shown to more than double. Using revLSTM, a MCC of 0.17 was achieved when predicting short term future reversals and 0.54 when confirming short term past reversals. A MCC score of 0.10 was achieved when predicting long term future reversals and 0.29 confirming long term past reversals.

Keywords: machine learning, logistic regression, k-nearest neighbors, deep neural network, convolutional neural network, recurrent neural network, long short term memory, stock market, prediction, trend, reversals

Contents

Chapter 1

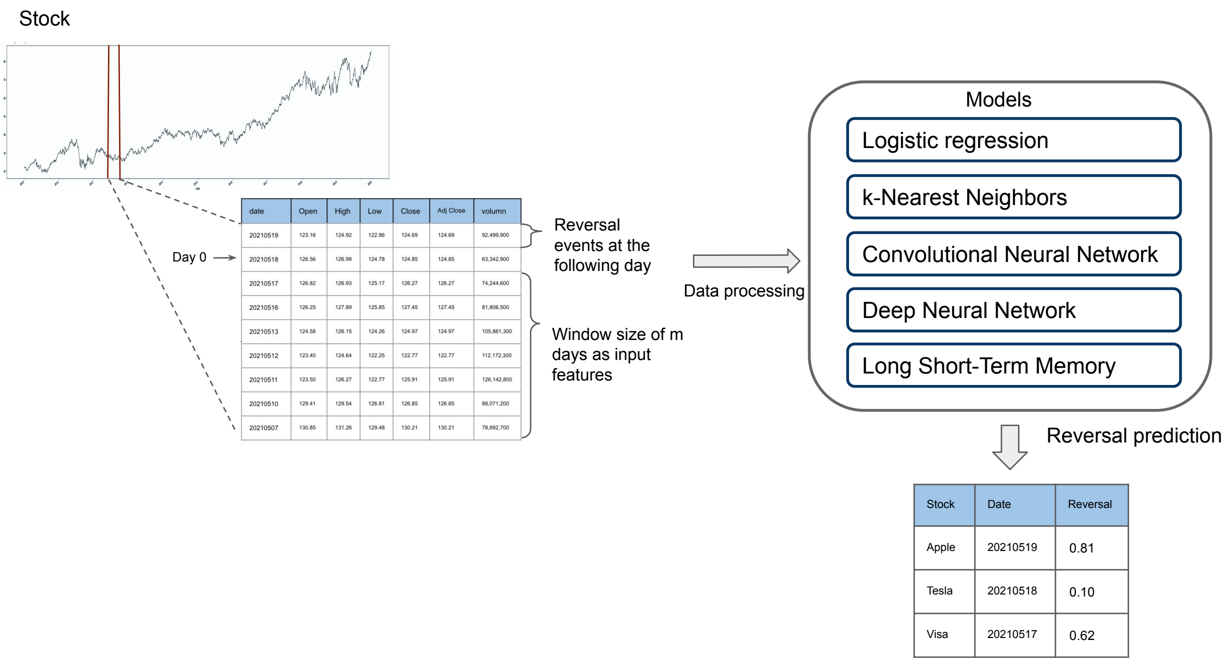
Introduction

Out of the many functions of a stock market, it could be argued that the primary ones are to enable companies to efficiently raise capital and to facilitate price discovery. A company is able to raise money by selling shares. This can in some cases be more advantageous than corporate loans or selling bonds. Price discovery entails finding a fair price for some asset. The price of an asset is whatever the participants are willing to buy and sell it for. Price discovery is a continuous process that begins when an asset is listed and does not end until the asset is unlisted. The movements of the financial systems are therefore the result of an uncountable number of humans' beliefs and actions. Despite the efficient market hypothesis stating that the current price of an asset reflects all publicly available information and thus making future predictions impossible. There are today countless studies trying to predict stock prices with good results (Selvin et al., 2017; Hiransha et al., 2018; Hoseinzade and Haratizadeh, 2018; Hoseinzade et al., 2019; Kusuma et al., 2019). However, only a small subset of these attempt to predict reversals (Christoffersen and Diebold, 2006; Bury, 2014; Jang et al., 1993) and out of these none have evaluated the performance of different machine learning models.

1.1 Problem

One could in theory exploit these stock market predictions to profit and/or to reduce risk to protect against unfavorable changes in the price of the asset. Among the ways one could exploit these predictions is timing the market. Preferably, one should buy at a low price and sell at a high one. In other words, buy in the beginning of a upward trend and sell at the top before the downward trend begins. Knowing when reversals are most likely to happen can therefore be lucrative.

One way of predicting stock market events is to look at historical stock data. As shown in Figure 1.1, a window of the data from the past m days can be fed into machine learning models after data normalization (see Section 4.1.2). The prediction could either be for a future reversal or to confirm a past reversal. The further in the future the prediction is, the more valuable it is as it allows the trader to act before the mass. In addition, the more accurate the prediction is the more capital can be committed at a constant risk.



date	Open	High	Low	Close	Adj Close	volumn
20210519	123.16	124.92	122.86	124.69	124.69	92,499,900
20210518	126.56	126.99	124.78	124.85	124.85	63,342,900
20210517	126.82	126.93	125.17	126.27	126.27	74,244,600
20210516	126.25	127.89	125.85	127.45	127.45	81,806,500
20210513	124.58	126.15	124.26	124.97	124.97	105,861,300
20210512	123.40	124.64	122.25	122.77	122.77	112,172,300
20210511	123.50	126.27	122.77	125.91	125.91	126,142,800
20210510	129.41	129.54	126.81	126.85	126.85	88,071,200
20210507	130.85	131.26	129.48	130.21	130.21	78,892,700

Stock	Date	Reversal
Apple	20210519	0.81
Tesla	20210518	0.10
Visa	20210517	0.62

Figure 1.1: Conceptual diagram for this thesis

1.2 Research question

The goal of this thesis is to investigate whether it is possible to accurately predict reversal events. Four research questions have been formulated:

- **R1**: Which methods are applicable for predicting reversals using the past m days?
- **R2**: To what accuracy can reversals p days in the future be predicted using the past m days?
- **R3**: To what accuracy can past reversals q days in the past be recognized using the past m days?
- **R4**: How will the performance of the predictions be affected by additional calculated features?

1.3 Outline

This report will have following structure:

- **Technical Background**: Here we introduce the related work, the models and the metrics used to evaluate the performance of the models.
- **Data**: This chapter presents the data: what and how the data was gathered, what features it contains and how reversals were defined and found.
- **Method**: This chapter explains how the experiments were set up, how the models were constructed and how the performance of the various reversal sizes and features were evaluated.
- **Results**: In this chapter the results are presented and interesting observations are mentioned.
- **Discussion**: In this chapter the results are discussed along with the importance of the different window sizes and reversal sizes. Finally, the strengths and weaknesses of our approach are discussed.
- **Conclusion**: Here we present out conclusion along with future work.

1.4 Contributions

This thesis contributes to:

- Knowledge in price trend reversal prediction using historical stock data using deep learning
- Insights in the relationship between window size, the size of the reversal and how far into the future the prediction is.
- Understanding regarding the impact of additional features on reversal prediction.

Chapter 2

Technical Background

2.1 Related Work

To the best of our knowledge, there are no previous studies evaluating the performance of different machine learning models on classification of trend reversals. Jang et al. (1993) created a model consisting of two shallow artificial neural networks that together predict the trend of price movement and recognized reversals based on these trend predictions in order to generate buy and sell signals. Similarly, we intend to apply artificial neural networks in this thesis in order to predict reversals. However, the neural networks used in this thesis will only be deep neural networks i.e. have more than one hidden layer (see Section 2.5) and both future and past reversals will be predicted.

There is previous work on trend prediction that is based on the properties of the distribution of returns[1](Christoffersen and Diebold, 2006) and pairwise correlation of markets (Bury, 2014). However these only predict the direction (up or down) and not reversals.

Numerous studies have examined the direction of a stock's price the following day using deep learning (Selvin et al., 2017; Hiransha et al., 2018; Hoseinzade and Haratizadeh, 2018; Hoseinzade et al., 2019; Kusuma et al., 2019). In general, deep learning outperformed other techniques in such tasks. Convolutional neural networks (CNN) achieved the best performance, outperforming recurrent neural networks (RNN), long short-term memory (LSTM) networks and multilayer perceptron (MLP) in predicting stock price movements (Selvin et al., 2017; Hiransha et al., 2018). These studies showed that deep learning is suitable for stock price prediction. Considering that, their performance on trend reversal prediction will be examined in this thesis.

[1]The distribution of changes from one day to the next

There were two main types of inputs when applying a CNN. Most studies created charts such as candlestick charts (Kusuma et al., 2019) using the historical prices and fed them as images directly to the CNN. An alternative way is to use 2D or 3D tensors. For example, Hoseinzade and Haratizadeh (2018); Hoseinzade et al. (2019) stacked the 2D data into a 3D tensor (markets × days × features). Whereas stacking the data into 3D did not enhance the prediction.

2.2 Baseline model

A baseline is a simple model to create predictions for a dataset. It is normally used to assess the performance compared to the studied models. An improved performance over the studied models is expected. A dummy classifier [2] is often used as a baseline. Dummy classifier is a classifier that uses simple rules. There are three common strategies: stratified, most frequent and uniform.

Stratified strategy generates predictions based on the classes' distribution in the training set. For example, if the proportion of non-reversal events is about 0.88% in one case in this thesis. The dummy classifier has 88% probability of predicting that a day in the test set is a non-reversal event.

The most frequent strategy, also termed Zero Rule Algorithm, assigns all days in test set as the most frequent class in the training set.

The uniform strategy generates predictions uniformly at random. Any day in the test set has equal chance of being either a reversal event or non-reversal event.

2.3 Logistic Regression

Logistic regression (LR) is a linear model used to explore relationships between independent variables and binary responses. LR has been widely used in stock market (Upadhyay et al., 2012; Attigeri et al., 2015). A logistic model can be written as Equation 2.1.

$$\log \frac{p}{1-p} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots \tag{2.1}$$

p denotes the probability when the binary response is class 1 for example, x_1 and x_2 represent independent variables while β_0, β_1 and β_2 coefficients for constant, x_1 and x_2, etc.

In the training phase, the logistic model is fitted to the input data by estimating the coefficients β_0, β_1 and β_2. The coefficients together with the input x_1 and x_2 are used to produce a prediction. The predicted probability can be converted from the predicted value using a sigmoid function (Lever et al., 2016).

[2]https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html

2.4 k-Nearest Neighbors

The k-Nearest Neighbors algorithm (kNN) has been widely used in classification and regression problems due to its simple implementation. It is able to classify and assign labels by using the majority rule. That is the class that has the k most similar samples in the training set.

The kNN relies on similarity between samples. Here, Minkowski distance, a generalization of both the Euclidean distance and the Manhattan distance, was applied. Minkowski distance measures similarity between two samples $X_1(x_{1,1}, x_{1,2}, x_{1,3}, ...x_{1,n}), X_2(x_{2,1}, x_{2,2}, x_{2,3}, ...x_{2,n})$ as the Equation 2.2. This is equivalent to Manhattan distance when p is 1 and Euclidean distance when p is 2.

$$d(X_1, X_2) = (\sum_{i=1}^{n} |x_{1,i} - x_{2,i}|^p)^{p^{-1}} \tag{2.2}$$

As shown in Figure 2.1, the kNN algorithm is based on a dataset of high-dimensional points where each point has a class. For a class-unknown sample, its class is predicted as the majority class where the k closest samples belong.

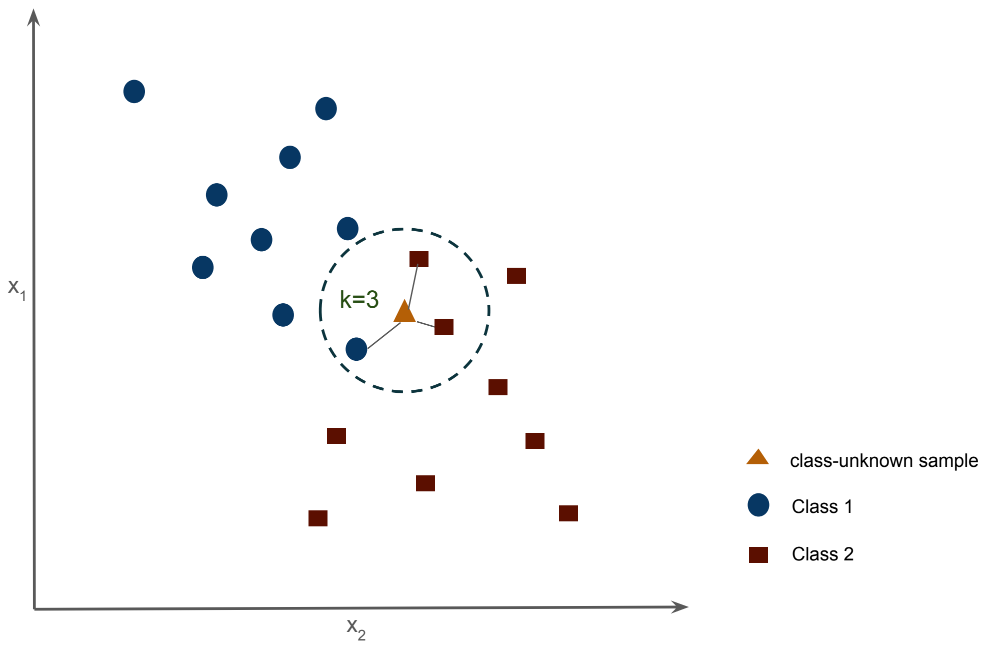


Figure 2.1: An example of applying kNN in prediction. Here, the three nearest neighbors (k=3) and Euclidean distance is used. The class-unknown sample is predicted as class 2 based on majority rule.

The performance of kNN are affected by the number of neighbors k. The choice of k depends upon the data. A large k could reduce bias caused by data heterogeneity, but could also worsen performance. The number of samples this thesis deals with is on the larger side for kNNs.

Fortunately, Facebook AI have developed a fast implementation [3] of the kNN algorithm using a new indexing data structure. This implementation was used for all kNN experiments.

2.5 Artificial Neural Network

Artificial neural networks (ANN) are a type of model that is made up by layers of neurons connected to other layers through weights. These weights enable the network to "learn" to approximate non-linear functions. As shown in Figure 2.2, an artificial neural network typically has an input layer, one or more hidden layers and an output layer. If the network has more than one hidden layer it may be called a deep neural network (DNN). Having more layers means having more weights which in turn enable the network to approximate more complex functions.

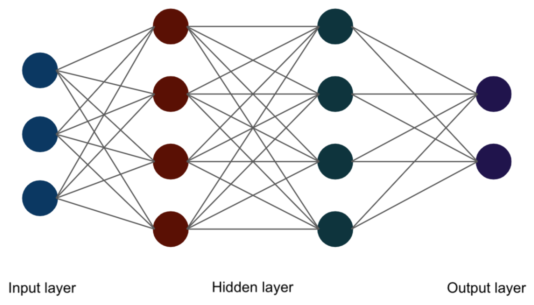


Figure 2.2: Artificial neural networks architecture

Fully connected layer

A fully connected layer, also known as a dense layer, is the primary type of hidden layer in ANNs. CNNs and LSTMs both usually include this type of layer, but they also contain other types of layers. This layer is characterized by having all the neurons of the current layer connected to all the neurons of previous layer through weights as seen in Equation 2.3.

$$A = ReLU(I \cdot W + B) \tag{2.3}$$

Where A is the activation (result) vector of the current layer, I is the input to this vector (the activation vector of the previous layer), W are the weights and B are the biases. The bias can be seen as a constant that shifts the value of the weighted inputs.

2.5.1 Forward propagation

To make an approximation or prediction, the network performs a forward propagation. This is the process of taking an input and passing it from layer to layer until it reaches the output layer. The neurons of the first layer assume the values of the input. Then the values from

[3]https://github.com/facebookresearch/faiss

these neurons are passed to the first hidden layer. Every neuron of a layer is connected to every neuron of the next layer through a weight. The value is multiplied by the weight in order to increase or decrease the value passed to the next neuron. The receiving neuron has now been passed as many values as there are neurons in the previous layer. These values are summed, and a bias is added. Finally, the activation function is applied. The purpose of the activation function is to introduce non-linearity. The activation functions used in the thesis are tanh, sigmoid and ReLU. Their plots can be seen in Figure 2.3.

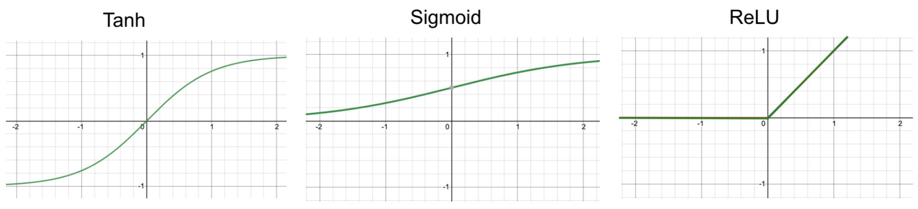


Figure 2.3: The three activation functions

Hyperbolic tangent function ($tanh(x)$)

The hyperbolic tangent of x is a function that has property of being continuous and non-linear with an output in the range [-1, 1].

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.4}$$

Sigmoid activation function ($\sigma(x)$)

The sigmoid function is also a continuous non-linear function, but whose output is in the range [0, 1]. This function used to be a popular choice as an activation function, but has in recent years been replaced by ReLU in most cases.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.5}$$

Rectified Linear Unit ($max(0, x)$)

Rectified Linear Unit (ReLU) is a function which returns zero for all non-positive x and otherwise x. This means that the output range is [0, ∞].

2.5.2 Backpropagation

Fitting this model can be done in a number of ways. One common way, and the one chosen for this thesis, is to use stochastic gradient descent with mini-batches. This is done by finding the gradient of the loss function (the error) in respect to the weights and then modifying the weights in order to minimize the loss function. The weights of the hidden layers are updated from the last layer to the first. The loss function is a function that takes the predicted value and compares it to the actual value and returns the loss. There are countless loss functions that can be used. We chose binary cross entropy as our output is binary. The loss is calculated as shown in Equation 2.6, where y_i is the actual value, $p(y_i)$ is the *i-th* output from the model and N is the number of values in the batch.

$$\text{Loss} = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i)) \tag{2.6}$$

Mini-batches refers to a technique where the network feeds forward and backpropagates multiple examples at a time in order to have a stable gradient descent.

2.6 Convolutional Neural Network

Convolutional neural networks (CNN) are also a deep learning algorithm, but have been created to take in inputs such as images or matrices and extract features. These features can then be used in order to detect and/or classify objects in images. The CNN is composed by an input layer, one or more hidden layers and an output layer. The hidden layers of a CNN typically consist of convolutional layers, pooling layers, dropout layers and fully connected layers.

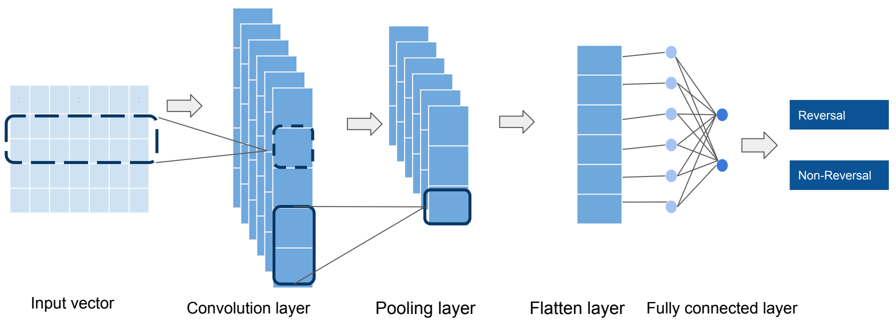


Figure 2.4: The illustration of CNN1D architecture to classify reversal/non-reversal

Input layer

Input for CNN is typically 2 or 3-dimension tensor. In this thesis a 2-dimensional tensor was used.

Convolution layers

Convolutional layers convolve the input and pass it to the next layer. This is achieved through a matrix multiply operation in order to extract high-level features by taking multiple filters.

As shown in Figure 2.5, the output is obtained by sliding the kernel over the input data and multiplying them. This shrinks the output size. In order to obtain an output with the same size as the input, padding can be used. The figure shows an example of a convolutional1D operation with only one filter, whereas multiple filters are used typically.

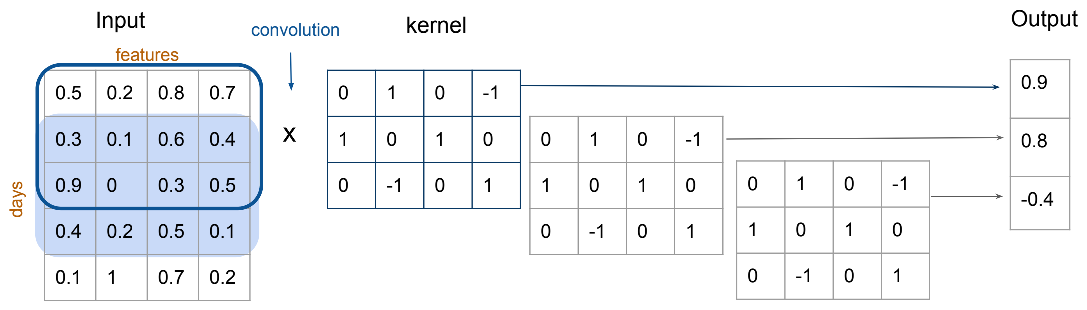


Figure 2.5: The convolutional1D operation for one filter with a 5 × 4 input data and a kernel size of 3. The stride is one and padding is zero.

Pooling layers

To reduce the size of the output of the previous layer a pooling step is performed. It can reduce a 2x2 matrix into a single value. Max and average pooling are the two most commonly used types. Max finds the largest value, while average takes the average of the 2x2 matrix. Max pooling was used in this thesis as well as in previous studies on stock price prediction (Kusuma et al., 2019).

2.7 Long Short-Term Memory (LSTM)

Recurrent neural networks (RNN) are a type of ANN where each cell has an internal state which is incorporated into the prediction. They are widely used in time series prediction as the internal state enables the network to remember information from the previous time step. Through this mechanism inputs of variable lengths can be passed to the model. Common applications for this type of neural network are time series classification and natural language processing. However, this type of network has a downside. They are more susceptible to the vanishing gradient problem. This occurs when the gradient becomes vanishingly small resulting in vanishingly small updates to the weights and therefore no further learning. An

LSTM is an improved version of an RNN as it solves this problem by having multiple components that choose what data to remember and to forget. Typically this type of model has been shown to work well for natural language processing and time series classification.

2.7.1 LSTM Cell

Each cell takes the state from the previous cell through C_{t-1} and passes it to the next cell through C_t (See Figure 2.6). But before being outputted, the state is manipulated through two operations. First it is multiplied by the output of the forget gate f_t. Due to the nature of the sigmoid function used in the forget gate, the value returned may only be between zero and one. This dictates how much of the previous signal to forget. If the cell were to decide that the previous state has to be forgotten, then f_t would have to be zero. Then the result of the input gate is added to the state. The state is now ready to be passed to the next cell. The next hidden state is calculated using the current state and the sigmoid of the concatenation of the previous hidden state and the data from the current time frame.

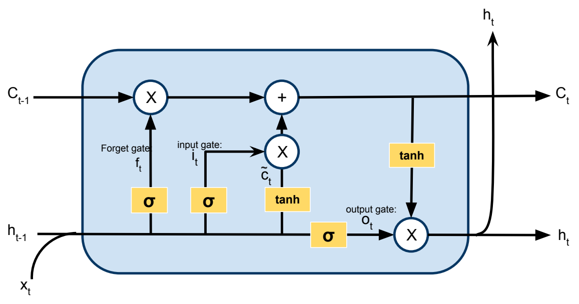


Figure 2.6: An illustration of a LSTM cell

2.8 Evaluation metrics

As shown in Figure 2.7: A True Positive (TP) is a reversal correctly predicted as reversal, a False Positive (FP) is non-reversal incorrectly identified as reversal, a True Negatives (TN) is non-reversal correctly predicted as non-reversal and a False Negatives (FN) is reversal incorrectly identified as non-reversal. Precision and recall are two common evaluation metrics for classification. By themselves, they cannot accurately portray the performance of the model. For example, if only one point in 100 is a reversal and all the points are predicted as reversals, then the recall is 1.0. While if only 99 points in 100 are reversals and all the points are predicted as reversals, the precision is 0.99. Combining recall and precision is therefore crucial.

The F1-score is a geometric mean of precision and recall (Powers, 2015). It is calculated as shown in Equation 2.7. The highest possible value is 1.0, indicating that precision and recall are perfect. While the lowest possible value is 0, when either the precision or the recall is

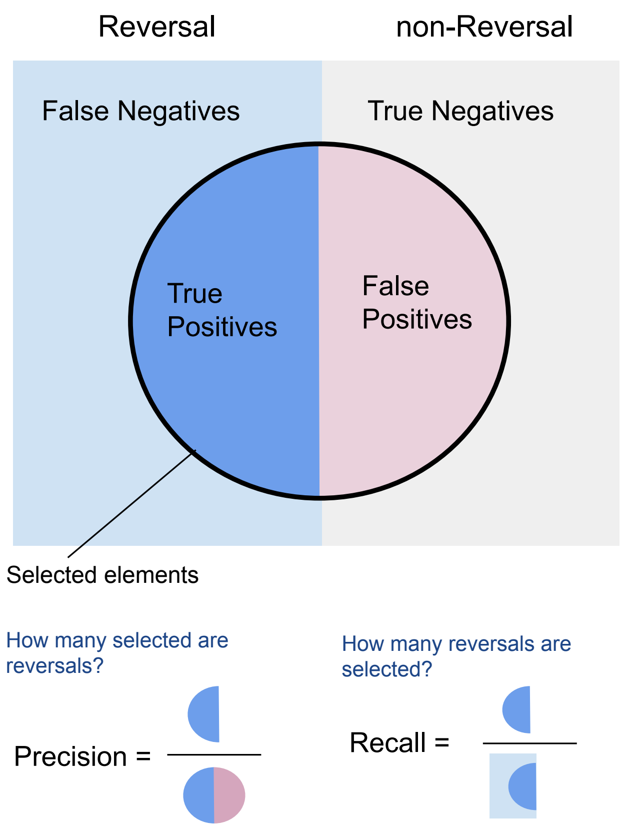


Figure 2.7: The relationship between precision, recall and confusion matrix.

zero. Therefore, the F1-score is useful either when data is imbalanced or when false positives or false negatives have different costs.

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.7}$$

In addition, another commonly used metric for imbalanced data is Matthews correlation coefficient (MCC). MCC is a reliable statistical rate and defined as Equation 2.8. It has a high score only if the prediction in all of the four confusion matrix categories obtained good results (Chicco and Jurman, 2020). Its range varies from [-1, 1], where 1 indicating a perfect prediction, 0 similar to a random prediction and - 1 means disagreement between prediction and observation (worse than random).

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{2.8}$$

Chapter 3

Data

3.1 Datasets

The dataset chosen for this thesis was based on the S&P 500 composite index. The S&P 500, is a stock market index that measures the stock performance of 500 large companies listed in the United States. It is one of the most commonly followed equity indices. Our dataset is composed of the components that make up the index. This index was chosen as it contains a good number of stocks with good data quality. If an index with a larger number of stocks would have been chosen, then the data from the smaller and lesser traded stock would have impacted the overall quality of the dataset.

3.2 Data retrieval

The resources required for this thesis are the historical price and volume data for every stock in the S&P 500 composite index. The dataset is freely available from Yahoo Finance's API. Table 3.1 shows a sample of the structure of the data. There was no need for a separate data cleaning step due to the high-level API the data was sourced from.

3.3 Features

Open, high, low, close (OHLC)

Between the times when the market opens and closes a stock's price will assume many different values. OHLC describes the opening price, highest reached price, lowest reached price and closing or final price. By summarizing the time interval, in our case a day, into these four values, a lot of the high frequency noise can be filtered out. Yahoo finance includes the adjusted close value, which is the closing price adjusted for a company's actions such as splits. This column was not used.

Volume

Further, a time interval does also have a volume. That is the number of shares that have been traded. This metric may tell an investor the amount of activity in a particular asset. For example, if the volume of a trending stock is declining it could mean that the trend's strength is declining indicating that a reversal might soon happen.

Table 3.1: Sample of historical OHLCV data

Symbol	Date	Open	High	Low	Close	Adj Close	Volume
AAPL	2010-01-04	7.622500	7.660714	7.585000	7.643214	6.547977	493729600.0
AAPL	2010-01-05	7.664286	7.699643	7.616071	7.656428	6.559296	601904800.0

Distance to price level of previous reversals (Prev)

Reversals tend to happen at the same prices. Meaning, if a reversal happened at price A, it is likely that a trend might some time in the future also reverse at the price A. The probability of reversals happening at that price increases as more reversals happen at that price. Sometimes the reversals happen at exactly the same price. However, due to the noise in the market, they are more likely to happen within a close distance. To add this feature to the dataset, the absolute percentage distance between the current close to the n.th previous reversal was calculated. This was done using the Equation 3.1.

$$prev_n = \left| \frac{\text{the n.th previous reversal close price}}{\text{the current day's close price}} - 1 \right| \tag{3.1}$$

3.4 Construction of training and test sets

We split data into training and test sets by date. Training set was restricted between the first day of 2010 and the last day of 2017 based on stocks listed on the S&P 500 index, whereas the test set was constructed from the first day of 2018 to the end of 2019. The training set was further partitioned into training and validation set for training and hyperparameter tuning using Keras build-in validation split [1]. The validation split was set to 10%, meaning that Keras automatically reserves the last 10% of the data set for validation. Hyperparameter tuning entails finding values for a model's parameters that produce the best results on the validation data. The model is then tested on the test data. The reason why the data set was partitioned by date is because the testing had to be done on out of sample data. That is, data that the model has not seen before. It would not suffice to randomly sample the data set to create training and test sets as the model might learn to "fill in the gaps".

The number of samples for every size of sliding window can be seen in Table 3.3 in the *total* column.

[1]https://keras.io/api/models/model_training_apis/

Table 3.2: The period time of our dataset, separated between the training and testing data.

Indices	Training set		Test set	
	start	end	start	end
S&P 500	2010.01.01	2017.12.31	2018.01.01	2019.12.31

3.5 Defining reversal events

Before defining trend reversals, trends should be defined. There are many ways to define a trend. Generally, one would say that rising tops and rising bottoms constitute a upwards trend, also known as a bullish trend. The opposite, lower peaks and lower bottoms, would be classified as a downwards trend, or rather a bearish trend. A trend reversal can therefore be defined as the point in which the trend transitions from bullish to bearish or vice versa. However, the fractal nature of the historical stock price results in trends containing smaller trends inside themselves as seen in Figure 3.1 and 3.2. Any of the numbers and letters in Figure 3.1 could be classified as a reversal. Yet, one would not want to use all of these reversals. Most of them might be too small for any practical use. An investor might only be interested in the one labeled *I* and *II* or perhaps the ones with a circle around the label. This would all depend on what time frame the investor is interested in.

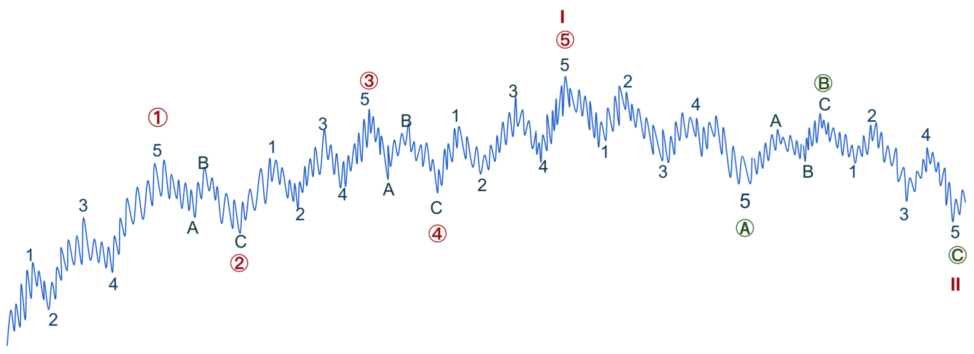


Figure 3.1: Illustration of Elliot Waves

Exactly what should be labeled as a reversal is quite difficult to know as historical price data is noisy. Labeling the data manually would both be highly time consuming and subjective. The labeling might differ substantially from person to person. A formal way to define reversals, and the one that has been chosen in this thesis, is to compare a price point with the surrounding price points. The number of data points to the left and right of the current data point define the scale of the reversals (see scipy.signal.argrelextrema [2]). Depending on the number of data points (henceforth referred to as reversal size) checked on either side, different sized reversals can be found. This thesis will examine the following values for reversal sizes 5, 10, 15, 20, 25, but will focus primarily on 5 and 10 for the prediction in the future. Intuitively, the greater the reversal size, the less reversals will be found. This makes the data quite unbalanced as seen in Table 3.3. As seen in Figure 3.2, the greater the reversal size, the

[2] https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.argrelextrema.html

bigger trends are shown by the labeling. A model trained using larger reversals might be used to trade on longer time frames, while a lower might be used for shorter ones.

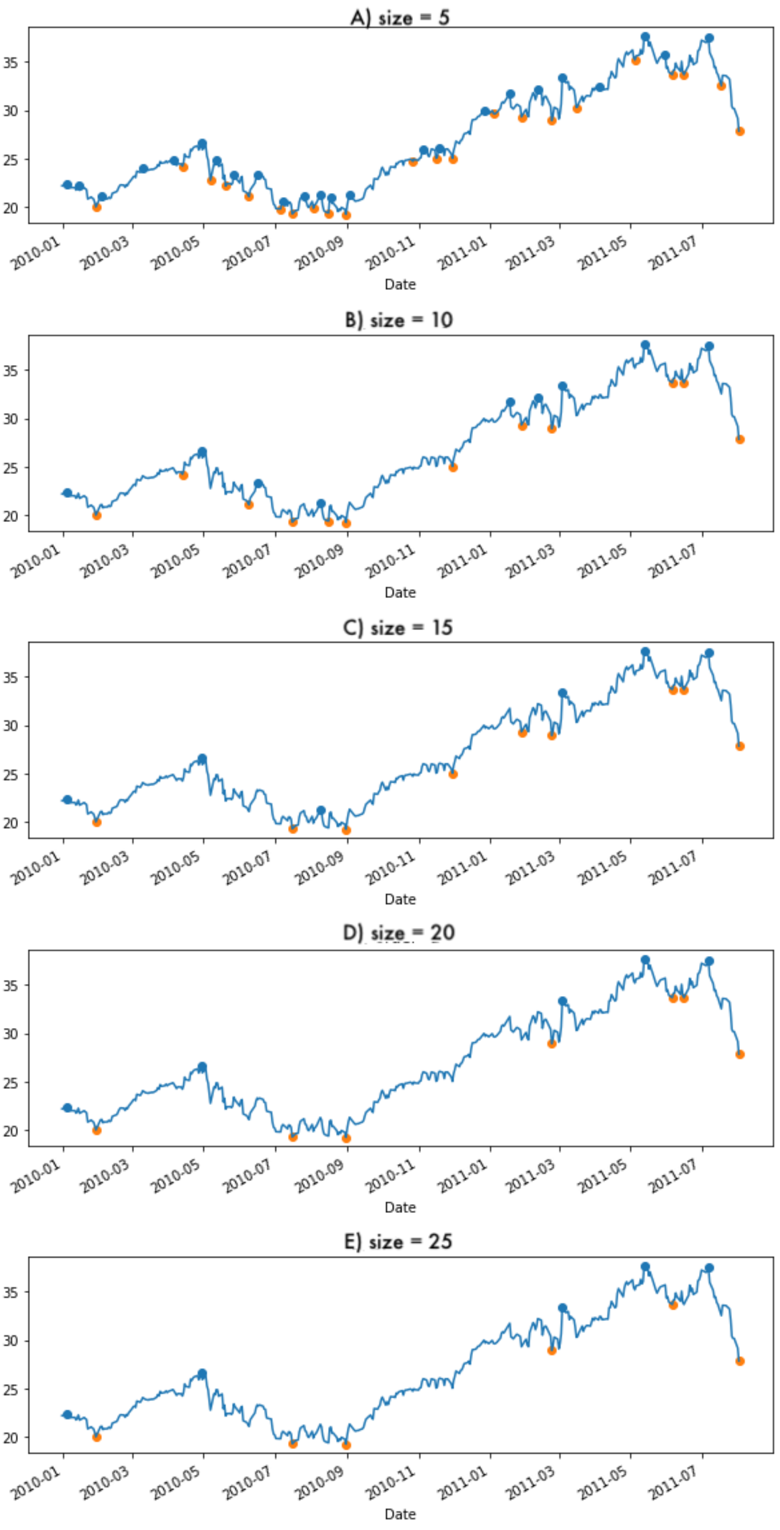


Figure 3.2: Different labeling depending on reversal size

3.6 Data characteristics

We constructed training and test data sets using the different reversal sizes. The final number of samples was less than the unprocessed number due to data prepossessing. For example, if the number of previous reversals was set to six, then all the data before the sixth reversal would have to be discarded as a minimum of six reversals would have to have happened. The ratio of reversals to non-reversals decreased as the reversal size increase, varying from 12.5% to 2.4%.

Table 3.3: Training and test sets

reversal size	Training set			Test set		
	Total	Reversal	Reversal percentage	Total	Reversal	Reversal percentage
5	929 353	116 570	12.5	247 028	29 833	12.1
10	905 340	56 447	6.2	246 815	15 385	6.2
15	888 251	36 947	4.2	246 721	9 908	4.0
20	870 343	26 688	3.1	246 450	7 560	3.1
25	849 104	20 478	2.4	246 303	6 112	2.5

3.6.1 Data imbalance

As shown in Table 3.3, our data was imbalanced since reversal events were quite fewer compared to non-reversals. Training a model directly on the data without considering this properly would result in poor prediction for the minority class. To counter this, two common approaches can be used: 1) sampling and 2) class weighting. Sampling is often referred to as under-sampling the majority class or over-sampling the minority class. One approach to doing this is to use synthetic minority oversampling technique (SMOTE) (Wang et al., 2006). Class weights incorporated different weights for the respective classes into the cost function of the applied algorithm. In this thesis we applied class weighting for LR and ANN models. The weights were calculated by counting the number of occurrences of the reversal and non-reversal events (see Equation 3.2).

$$w_i = \frac{N}{n_{classes} \times n_i} \tag{3.2}$$

where w_i is the weight for class i, N is the total number of samples, $n_{classes}$ is the total number of unique classes and n_i is the total number of occurrences of class i.

Chapter 4

Methods

4.1 Data processing

Pre-processing the data in order to make it suitable for the models is crucial. In theory, models could take raw data directly. There are models that are used for data pre-processing. However, in our case the data had to be reshaped, normalized and the additional features had to be calculated. Reshaping is required for the data to match the input requirements of the models. Normalizing and feature engineering is performed to improve performance.

4.1.1 Sliding windows

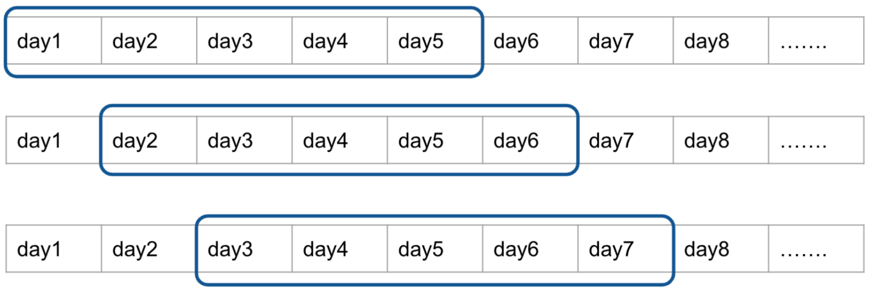


Figure 4.1: The processing of sliding windows

Sliding window is a method to feed a model a slice of sequential data. Each slice is seen as an example and used to make a prediction. We used an m-day long sliding window to slice the data. For example, we constructed features using data from a five-day window as shown in Figure 4.1. In this thesis, the window sizes of 5, 10, 20, 50 were examined.

4.1.2 Normalization

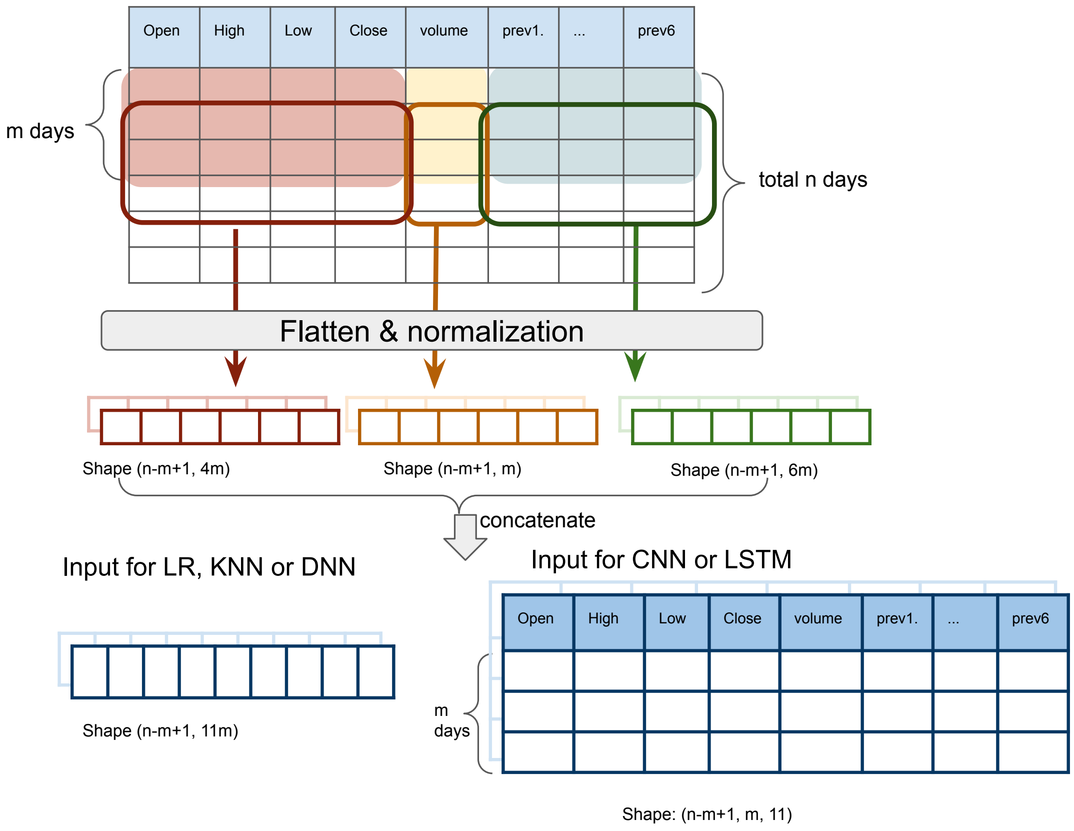


Figure 4.2: Schema of data normalization for LR, kNN, LSTM and CNN. Using a sliding window with m days, the OHLC prices for a stock were flattened as a vector, and min-max normalization was performed on this vector. Similarly, flattening and normalization was performed on the volume and the Prevs (see Section 3.3) respectively. For LR, kNN and DNN the vectors were concatenated horizontally. For the CNN and LSTM they were reshaped back into a 2D tensor.

A common best practice when training neural networks is to normalize the data so that the range is [0, 1] or [-1, 1] (Begam, 2020; Jimmy Ming-Tai Wu, 2021). This can be done using min-max normalization, which is a widely used method of scaling the data which does "not only speed up the optimal solution of gradient descent but also improve accuracy" (Jimmy Ming-Tai Wu, 2021). A min-max normalization was applied on the OHLC, volume and previous reversals data points separately as seen in Figure 4.2. This normalization re-scaled the features to the range of [0, 1] according to the Equation 4.1.

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{4.1}$$

where X is the data point, X_{norm} is the re-scaled data point and X_{max} and X_{min} are the biggest and smallest values in the window.

Lastly, the normalized features were concatenated together in different shapes depending on the model as seen in Figure 4.2.

4.1.3 BatchNormalization

BatchNormalization is used to normalize the layers' inputs by re-centering and re-scaling. It is examined in both CNN and DNN models.

4.2 Baseline model

The DummyClassifier from sklearn[1] was used as baseline. Considering imbalanced data characteristic, parameter "strategy" was set to "stratified". See the stratified strategy in section 2.2.

4.3 revLR

Our logistic regression model named revLR was implemented using sklearn[2] with the lbfgs solver and L2 regularization. Inverse of regularization strength was setting to one. The sklearn parameters were set as shown in Table 4.1.

Table 4.1: LR parameter settings

Parameter	value
penalty	'l2'
dual	False
tol	1e-4
C	1.0
fit_intercept	True
intercept_scaling	1
class_weight	'balanced'
random_state	2021
solver	'lbfgs'
max_iter	100
multi_class	'auto'
warm_start	False
l1_ratio	None

[1]https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html
[2]https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

4.4 revKNN

revKNN is our k-Nearest Neighbors based model. To select the best k, we implemented kNN varying k in the range [1, 31]. The model's performance for various values of k was evaluated on the validation dataset.

4.5 revDNN

revDNN is our DNN based model. revCNN and revLSTM are also technically deep neural networks, but they distinguish themselves from revDNN by having more just fully-connected layers. The architecture of revDNN is shown in Table 4.2 and is composed of a input layer, hidden layers and finally an output layer. Between the layer's dropouts, noise or normalization can be added. For this thesis dropouts were used to spread out the learning over the neurons. Keeping the network from depending on few neurons too much. This had a beneficial result on the performance and probably resulted in better generalization. The model was able to achieve better results by dropping some neurons between the two dense layers. The addition of Gaussian noise to the input layer was attempted to reduce overfitting (Reed and Marks, 1999).

Table 4.2: Our proposed revDNN architecture

Input-BatchNormalization
Gaussian Noise
Dense-256 ReLU
BatchNormalization
Dropout
Dense-256 ReLU
Dense-18 ReLU
Dense-1 Sigmoid

4.6 revCNN

For our revCNN model we chose to use random search (Bergstra and Bengio, 2012) which is a widely used strategy for hyperparameter optimization. We performed hyperparameter tuning using a validation data split of 20%. The parameters are showed in Table 4.3 were examined.

The architecture varies since input matrices have different sizes. The architecture for window size 5, 10 and 20 are listed in Table 4.4 and the architecture for window size 50 is the same as the others except that it only has 3 convolutional layers. It shows that our proposed CNN architecture consist of 4 convolutional layers with kernel size of 3, 1 max pooling layer, 1 dropout, 3 dense layers and batch size of 32.

Table 4.3: The tested values on each hyperparameter

Hyperparameters	Tested values
Convolutional layer	2-10
Filter	32, 48, 64, 128, 256
Kernel size	2-11
Dense	8, 32, 64, 128, 256, 512
Pooling	max, average
Batch size	32, 64, 128, 256, 512, 1024
Activation function	relu, selu, tanh, elu

Table 4.4: Our revCNN architecture

Input	Dense-8	conv1D-128,3 ReLU	conv1D-128,3 ReLU	conv1D-128,3 ReLU	conv1D-128,3 ReLU	max-pooling1D 2	Flatten	Dense-32	Dropout	Dense-1 Sigmoid

4.7 revLSTM

The LSTM model used had the structure shown in Table 4.5. Hyperparameter tuning showed that revLSTM performed best when only one LSTM layer was used. This layer was followed by a dropout layer and three dense layers.

Table 4.5: Our revLSTM architecture

Input
LSTM-256
Dropout
Dense-128 ReLU
Dense-32 ReLU
Dense-1 Sigmoid

4.8 Evaluation

The goal of this thesis is to use historical price data to make predictions on past and future predictions. As such, we first evaluated the performances of using varying window sizes, OHLCV and calculated feature (distance to price level of previous reversals) using revLR and revKNN. Then, the other models' performance was evaluated for the dataset with both OHLCV and distance to previous reversals. This was done for reversal sizes of 5 and 10 and the prediction was for the following day. The performances were evaluated using MCC and F1-scores.

Next, we examined the prediction of larger past reversal sizes (reversal size: 15, 20, 25) by revLR, revCNN, revDNN and revLSTM using MCC as performance metrics. revKNN was not examined as it performed poorly in the previous step (see Table 5.1) which was also shown in previous studies (Cosenza et al., 2020). revLR was used as the reference model in comparison with revCNN, revDNN and revLSTM.

In all the above mentioned evaluations, the MCC score and F1-score were calculated on the performance of the model on the entire training or testing dataset without any consideration to which examples came from which stocks.

Chapter 5

Results

5.1 Overview

model	reversal size of 5			reversal size of 10		
	window size	MCC	F1-score	window size	MCC	F1-score
Baseline	5	0.000	0.126	10	0.000	0.063
revKNN (k=1)	5	0.034	0.155	10	0.023	0.085
revLR	5	0.103	0.268	10	0.105	0.171
revCNN	10	0.173	0.279	10	0.153	0.184
revDNN	5	**0.174**	**0.293**	10	0.157	0.198
revLSTM	10	0.171	0.276	10	**0.163**	**0.190**

Table 5.1: Performance of future reversals prediction on reversal sizes of 5 and 10.

Table 5.1 shows the best performance of various models on reversal sizes of 5 and 10 when predicting if the reversal was going to happen the following day. Figure 5.1 shows the performance comparison of all the models. The results show the performance of the models when using their best performing window size. All the models performed using a window size of 10 days for reversal size of 10[1]. While revLR, revKNN and revDNN used a window size of 5 days and revCNN and revLSTM used a window size of 10 days when predicting reversal size of 5. As shown in the Table 5.1, the neural network approaches performed better than the reference models. It is interesting to note that the baseline model got a MCC score of 0 and a F1-score of 0.126 and 0.063 for reversal sizes of 5 and 10 respectively.

[1]reversals of size 10 = minimums or maximums compared to the previous and future 10 days

Among the three ANNs, the revLSTM performed either as well or slightly better than the revDNN and revCNN when looking at MCC. Although, the training time for the revLSTM on a CPU was much longer than revDNN and revCNN. One epoch took around 8 seconds with the revDNN, two minutes with the revCNN and over seven minutes with the revLSTM.

The actual reversals predictions plotted on a graph can be seen in Figure A.1.

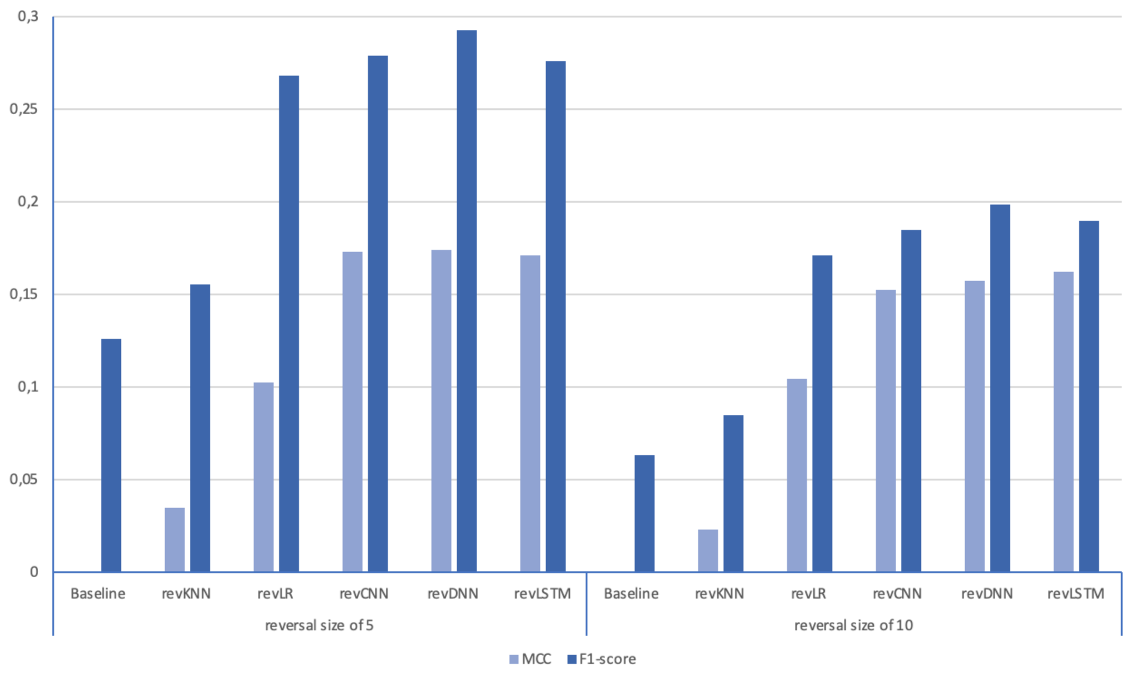


Figure 5.1: Best performances for respective models on reversal sizes of 5 and 10, respectively.

5.1.1 revLR

First, the logistic regression model was applied. The results are shown in Figure 5.2. The figure shows the comparison between using OHLCV and OHLCV+Prev. It is clear that by adding previous reversals, a substantially higher score was achieved. The best window size is 5 days for reversal of size 5, while the best window size is 10 days when the reversal size is 10.

5.1.2 revKNN

Second, the k-Nearest Neighbors algorithm was applied. The results are shown in Figure 5.2. The performance of this model was the lowest among the models. However, similarly to the other models, adding previous reversals improved its MCC scores. revKNN performed best with the same window sizes as revLR.

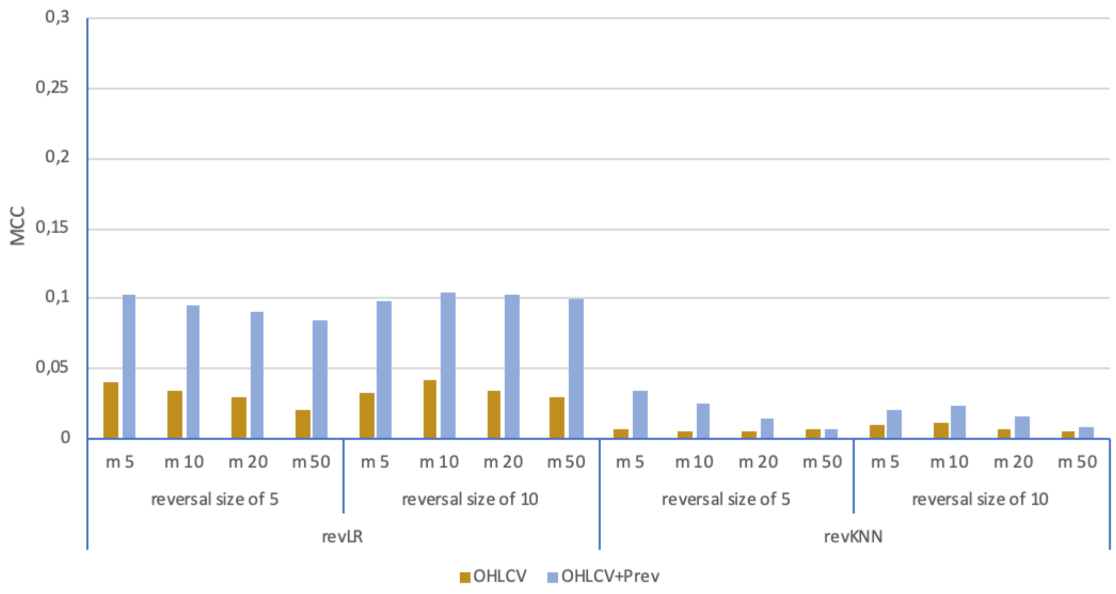


Figure 5.2: Performance of revLR and revKNN on reversal sizes of 5 and 10 with varying window sizes (m).

5.1.3 revDNN

The performance of revDNN can be seen in Figure 5.3. Generally, the best window size when reversal size was equal to 5 was 5. But when the reversal size is increased, a bigger window size performs better. One more interesting result is that the F1-Score in relation to the MCC score is higher when the reversal size is 5, and the opposite when the reversal size is 10.

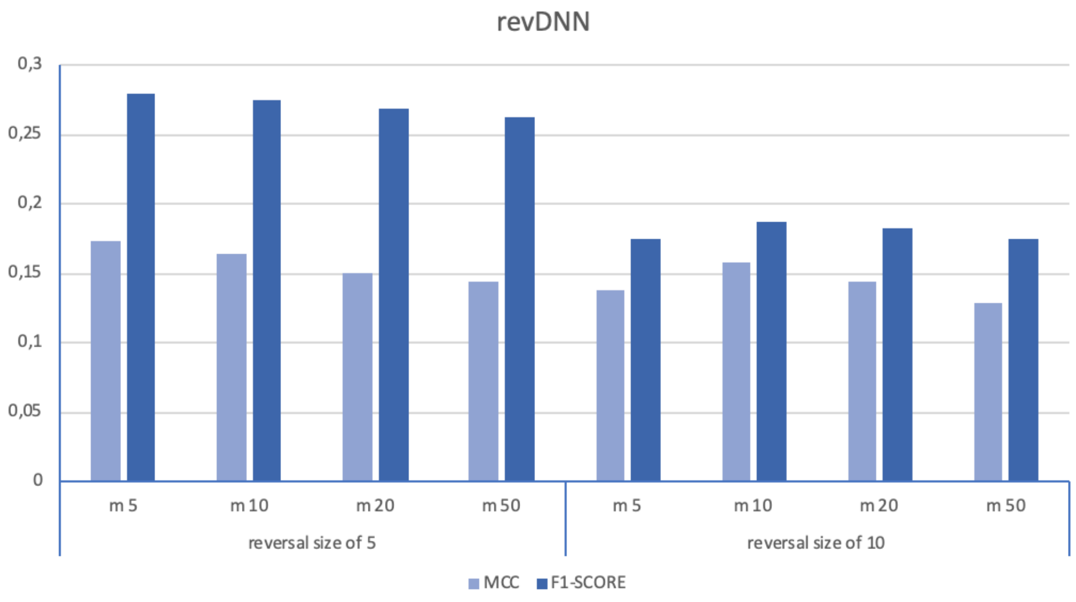


Figure 5.3: Performance of revDNN on reversal sizes of 5 and 10 with varying window sizes

5.1.4 revCNN

The performance of revCNN on reversal sizes 5 and 10 can be seen in Figure 5.4. The best window size was 10 for both reversals sizes of 5 and 10. When looking at MCC for reversal size of 10, a window size of 10 achieved the best MCC score, while a window size of 20 gave the best F1-score. This is not the same case for reversal size of 5.

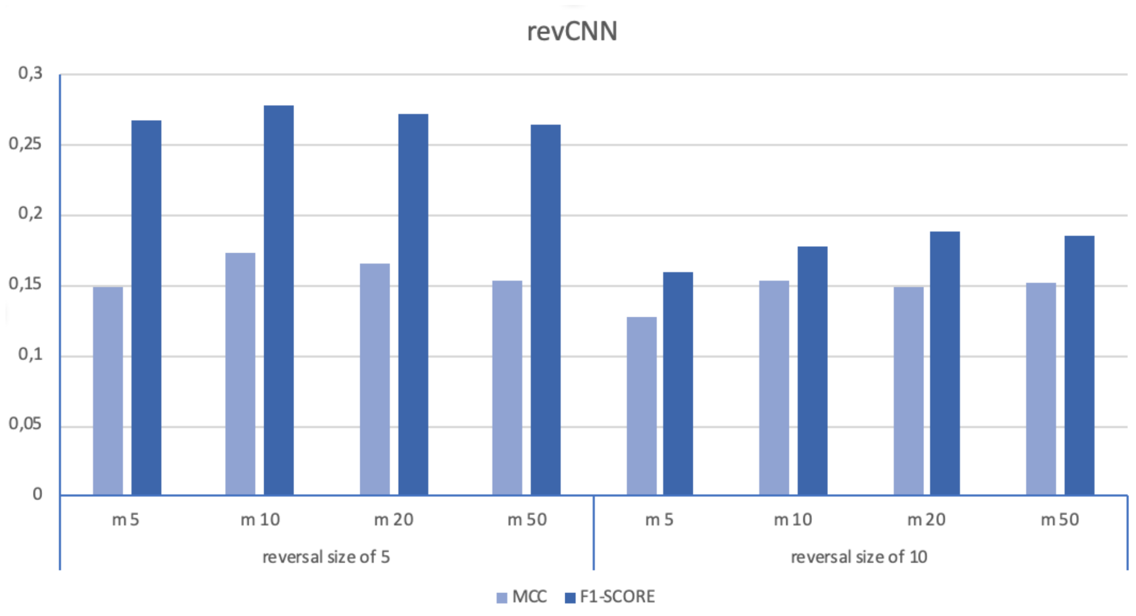


Figure 5.4: Performance of revCNN with different window sizes

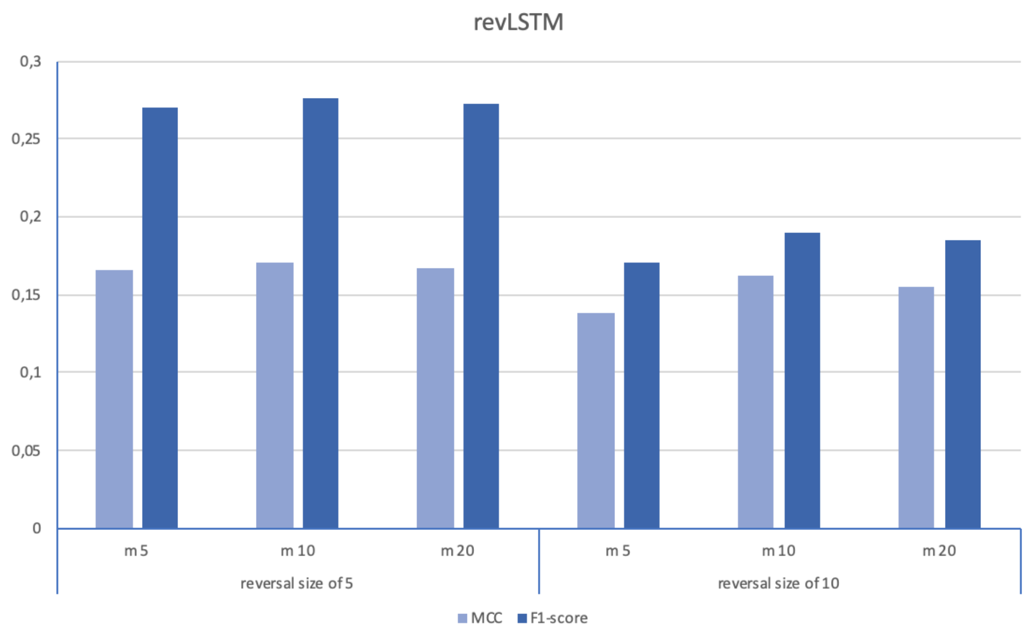


Figure 5.5: Performance of revLSTM on reversal size of 5 and 10 with various window sizes

5.1.5 revLSTM

The performance of revLSTM is shown in Figure 5.5. The best window size is 10 for both reversal sizes of 5 and 10. Both MCC-score and F1-score decreased slightly when reversal size increased.

5.2 Predicting past reversals

Prediction of past reversals using revLR, revCNN, revDNN and revLSTM were attempted and the performance was compared as shown in Figure 5.6. The revLSTM outperformed all other models.

The number after the *S* is the shift. If the shift is 0, then the model would be trying to predict that a reversal will happen today (the same data point as the last available data point). A positive shift of one or two would mean that the reversal predicted would have happened yesterday or the day before that. A negative shift means that the reversal will take place in the future. For practical applications, a shift of zero would require that the trading session has ended. Otherwise, the close price would not be the actual close. However, an approximation could be made with the current price just before the market closes.

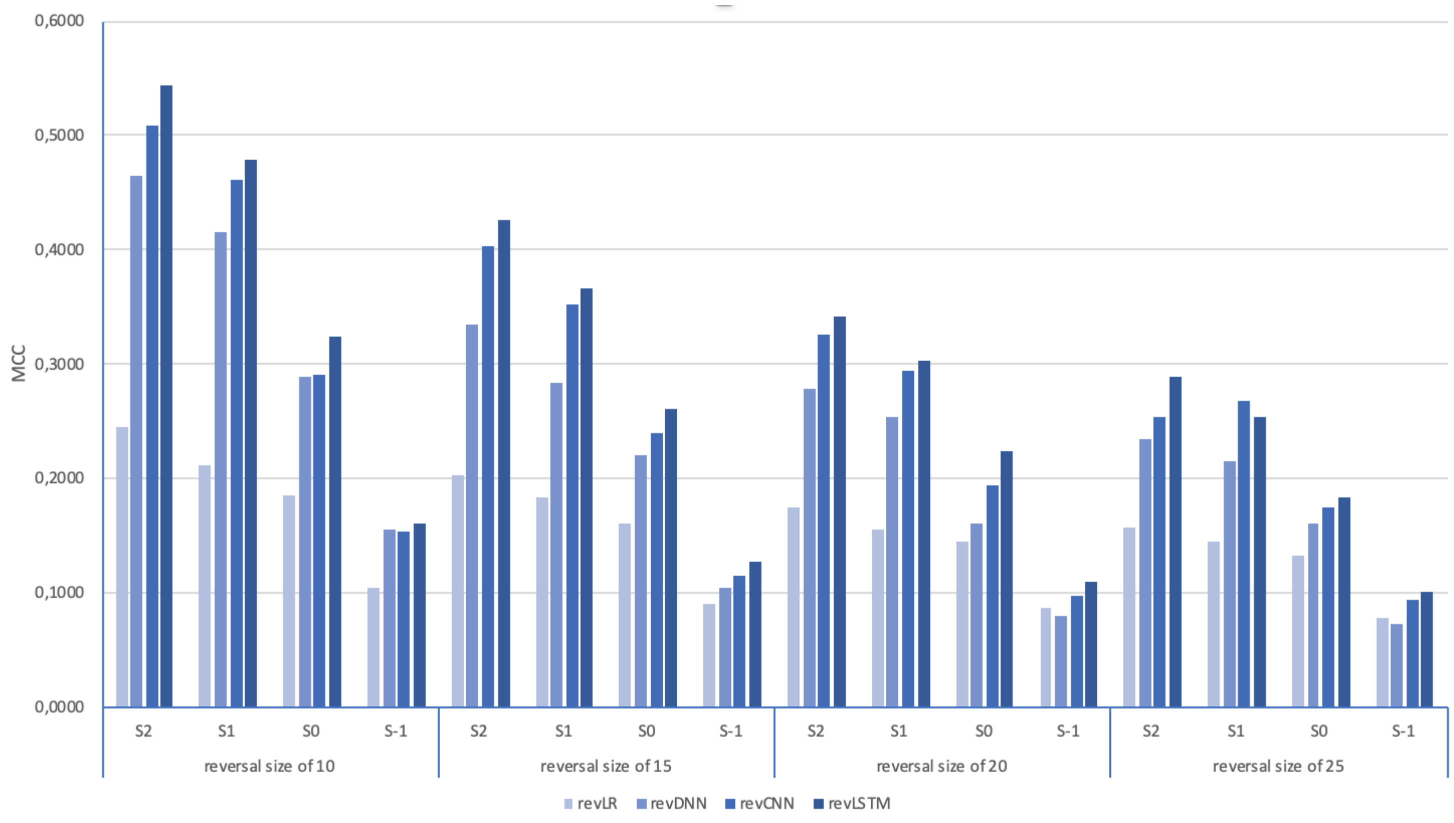


Figure 5.6: MCC comparison of all models to predict reversals on various shift

Chapter 6

Discussion

In this chapter we will discuss the definition of reversal and the performance of the various models. Their strengths and weaknesses of the models will be examined. A discussion regarding the size of the windows and trend reversals will also made. Furthermore, the model's performance at predicting past reversals have also been gathered and will be discussed here.

6.1 Definition of reversal

There are many ways reversals can be defined. For this thesis reversals were defined by looking at the greatest or smallest values in a sliding window (not to be confused by the sliding window used as the models' input). One drawback of this method is that reversals are only defined by being a local minimum or maximum in an interval of time. No consideration is given to the distance to the previous reversal. For example, if a small reversal size is chosen and the price is flat, multiple reversals might be identified even though it might only be noise. To prevent this, an alternative method of defining reversals such as the Zig-Zag Indicator [1] could be used. Similarly, this indicator also looks at local minimums and maximums, but then only recognizes a reversal if the distance to the previous one is greater than a pre-chosen percentage. The Zig-Zag method can also be adjusted in order to find different sized reversals. This can be done by changing the pre-chosen percentage. This definition was not chosen as the results found in the initial research phase were not adequate for this purpose. Many data points that we subjectively would like to have been classified as reversals were not selected. The results of the thesis might have looked completely different if this definition would have been chosen.

[1] https://www.investopedia.com/terms/z/zig_zag_indicator.asp

6.2 Models

The results showed that the neural network-based approaches performed the best for identifying reversals. This is inline with previous studies on predicting stock price movement (up or down) (Selvin et al., 2017; Hiransha et al., 2018). Among the different deep learning methods, the revLSTM achieved the best performance. This might be because they can find long-term dependencies between samples. As such, LSTMs may be suggested as a good method for classifications on time-series data in stock market.

The revDNN came in second among the models when looking at MCC and best when looking at F1-score. It was also among the fastest models to train and was very flexible in the way they might be configured. Compared to the other deep learning models, it did not require the data to be reshaped and was able to use a one-dimensional tensor as input. All of these advantages may be important when it comes to developing a model for business applications.

It should also be noted that our CNN architecture also performed well. However slightly worse compared to revLSTM and DNN. This shows the ability of revCNNs at learning to extract features in a highly noisy dataset. CNN-learned features are well applied and successful in many fields including biomedical research and stock market (Kusuma et al., 2019; Selvin et al., 2017). Our thesis reinforces the applicability of CNNs in time series classification.

Logistic regression is a less complicated approach, but well applied in the stock market prediction. Previous studies demonstrated promising performances in forecasting stock return (Upadhyay et al., 2012), as well as the direction of the next day's price (Attigeri et al., 2015). Our thesis also showed that revLR performed well in predicting reversals when looking at MCC score, but not as good as the deep learning models. And performed comparably to deep learning when looking at F1-Scores. Given that a reversal event might be more important than non-reversal event, a good F1-score from revLR is still meaningful. In addition, revLR didn't require long training times which enabled us to get an understanding of features' importance before implementing the advance models with larger hyperparameter search spaces.

revKNN did not perform well compared to the other models. Similar studies also reported bad performances by kNN in the prediction of growing stock volume (Cosenza et al., 2020). One possible explanation is that the input data is too noisy and not suitable for the model. More preprocessing of the data might be needed before the model can use it as input. This could even be done by some other model.

6.3 Importance of distance to price level of previous reversals

We employed revLR and revKNN to examine the importance of OHLCV and the distance to price level of previous reversals in predicting reversal events. Windows size of 5, 10, 20 and 50 were applied using a reversal size of 5 and 10. We found that neither revLR nor revKNN perform well using OHLCV alone. Great improvements on MCCs can be seen from

Figures 5.2 when the distance to price level of previous reversals was included in the dataset. Therefore, we included both OHLCV and distance to price level of previous reversals in all other models.

6.4 Window sizes and reversal sizes

It was expected that the smaller reversal sizes would have better results due to the greater number of reversals. However, it was surprising that a smaller window size (5 to 10 days) would often result in better performance compared to larger ones (20 to 50) as one would think that more information would be beneficial. But the window could not be too small, or the performance would start to decrease. Having a greater window either confused the model or did have high relevance for the prediction. It is interesting to note that the window size often matched the reversal size. This could be due to the likelihood of there being a reversal of size n in a m sized window when $n = m$. This symmetry could be mitigated by the use of an alternative definition for reversals such as the zig-zag indicator mentioned in section 6.1.

6.5 Predicting past reversals

As expected, predicting or rather confirming past reversals resulted in the best performance. This might be because the model is able to see the price changing direction. As opposed to only looking at past reversals and perhaps seeing the upwards or downwards momentum dying off. Classical stock trading strategies such as "moving average cross over" may also be used to predict that a reversal **has** happened. Depending on the market conditions, the prediction may be made many days after the reversal actually has happened. Our thesis focused on trying to predict that a reversal **will** happen in the future. However, by altering the prediction to trying to predict if a reversal **has** happened interesting results were gotten. Our model performed much better, for some reversal and window sizes even doubling the MCC score. Therefore, compared to moving average cross overs, our approach had a substantially smaller and constant time between the reversal happening and the model predicting such. If this alternative prediction should be used to create trading signals one would have to increase the order so that the reversals happen on larger trends. Otherwise, due to the lag, most of the potential profit might already be lost by the time the prediction is made.

6.6 Strengths and limits

Our research question was novel and may provide new value in understanding the stock market. One strength was the comprehensive assortment of machine learning models that were evaluated in this thesis. We applied different architectures of artificial neural networks and compared their results. Lastly, the significance of previous reversals was also studied and evaluated.

One weakness is that only three types of features were used as input. We didn't explore effects of adding more features such as indicators and/or concatenating multiple stocks together in order to predict multiple stocks at ones. This would be discussed in the Section 7.1. One other weakness is that the models were only trained and evaluated on the S&P 500. It would be interesting to replicate our experiments on other stocks to confirm our findings.

Chapter 7

Conclusion

In conclusion, we examined the performances of various machine learning approaches in predicting stocks' price trend reversals, something not previously studied. The results show that deep learning outperformed classical machine learning models such as LR and kNN. Among the different deep learning models, our LSTM model revLSTM achieved the best performances in classifying future and past reversal events. The accuracy to which this could be done was shown to be better than random and highly depended on the size of the reversals and whether the prediction in the future or the past. We also evaluated the performance impact of adding additional features such as the distance to previous reversals. This resulted in a significant improvement in performance which indicates that the distance to the price of previous reversals have a high significance for the prediction of past and future ones. However, there are more models to try and more feature engineering to be done.

The general trend for all models was that the further into the future the prediction was, the harder predicting became as shown by the lower MCC scores. Likewise, the larger the reversals, the worse the performance. An MCC score of 0.17 was achieved when predicting small **future** reversal and 0.54 when confirming small **past** reversals. An MCC score of 0.10 was achieved when predicting large **future** reversals and 0.29 confirming large **past** reversals.

7.1 Future work

7.1.1 Better dependent value

It is better to predict that tomorrow will be a reversal when the actual reversal will happen the day after tomorrow. Rather than predicting tomorrow while the reversal actually happens in

5 days. As it stands now, these two scenarios are equally incorrect. To improve the research question, the dependent value could have been the time to the next reversal in number of days. This could be seen as a time to failure problem. This might be an interesting topic for further theses.

7.1.2 Indicators

Besides reverting close to previous reversals, quite often reversals happen on moving averages. These tend to function as trend indicators and thus determine if the asset is still moving in an upward or downwards trend. If the price would cross a moving average, then this could indicate the end of a trend. It is therefore natural for the price to resist crossing a moving average. The typical number of previous data points used for the moving average are, ordered from strongest to weakest: 200, 100, 50, 20. Then there are exponential moving averages, these usually use 12 and 24 past data points. Besides using moving average indicators, one could also use oscillators. These are a type of indicator that are not necessarily bound to the price range of the asset. For example, the RSI indicator shows the momentum of a trend and produces values in the range of [0, 100]. Notable oscillators are RSI, MACD and Stochastic. All of these indicators could be added to the set of features in the data set.

7.1.3 Concatenating multiple stocks

Using multiple stocks as input for a model and then predicting separately whether a stock will revert could be worth looking into. The hypothesis to why predicting multiple stocks at once would be better than one is that there might be some correlation between the stocks. This approach was used successfully when trying to predict the movement of stocks (Hoseinzade and Haratizadeh, 2018). Hoseinzade and Haratizadeh (2018) attempted two different ways of concatenating stocks. The first was horizontally and the second was by stacking the two-dimensional tensors into a three-dimensional one. It was shown that the latter performed better. The stocks would however need to be shuffled in order to reduce overfitting.

References

Attigeri, G. V., M, M. P. M., Pai, R. M., and Nayak, A. (2015). Stock market prediction: A big data approach. In *TENCON 2015 - 2015 IEEE Region 10 Conference*, pages 1–5.

Begam, S. (2020). Machine learning algorithms for predicting the stock market daily returns. *International Journal Of Multidisciplinary Research In Science, Engineering And Technology*, 1.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13:281–305.

Bury, T. (2014). Predicting trend reversals using market instantaneous state. *arXiv*, abs/1310.8169.

Chicco, D. and Jurman, G. (2020). The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21.

Christoffersen, P. F. and Diebold, F. X. (2006). Financial asset returns, direction-of-change forecasting, and volatility dynamics. *Management Science*, 52(8):1273–1287.

Cosenza, D. N., Korhonen, L., Maltamo, M., Packalen, P., Strunk, J. L., Næsset, E., Gobakken, T., Soares, P., and Tomé, M. (2020). Comparison of linear regression, k-nearest neighbour and random forest methods in airborne laser-scanning-based prediction of growing stock. *Forestry: An International Journal of Forest Research*, 94(2):311–323.

Hiransha, M., Gopalakrishnan, E., Menon, V. K., and Soman, K. (2018). Nse stock market prediction using deep-learning models. *Procedia Computer Science*, 132:1351–1362. International Conference on Computational Intelligence and Data Science.

Hoseinzade, E. and Haratizadeh, S. (2018). Cnnpred: Cnn-based stock market prediction using several data sources. *arXiv*, abs/1810.08923.

Hoseinzade, E., Haratizadeh, S., and Khoeini, A. (2019). U-cnnpred: A universal cnn-based predictor for stock markets. *arXiv*, abs/1911.12540.

Jang, G.-S., Lai, F., Jiang, B.-W., Parng, T.-M., and Chien, L.-H. (1993). Intelligent stock trading system with price trend prediction and reversal recognition using dual-module neural networks. *Applied Intelligence*, 3(3):225–248.

Jimmy Ming-Tai Wu, Z. L. e. (2021). A graph-based cnn-lstm stock price prediction algorithm with leading indicators. *Multimedia Systems*.

Kusuma, R. M. I., Ho, T.-T., Kao, W.-C., Ou, Y.-Y., and Hua, K.-L. (2019). Using deep learning neural networks and candlestick chart representation to predict stock market. *arXiv*, abs/1903.12258.

Lever, J., Krzywinski, M., and Altman, N. (2016). Logistic regression. *Nature Methods*, pages 541–542.

Powers, D. (2015). What the f-measure doesn't measure: Features, flaws, fallacies and fixes. *ArXiv*, abs/1503.06410.

Reed, R. and Marks, R. J. (1999). *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. The MIT Press.

Selvin, S., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. K., and Soman, K. P. (2017). Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1643–1647.

Upadhyay, A., Bandyopadhyay, G., and Dutta, A. (2012). Forecasting stock performance in indian market using multinomial logistic regression. volume 3, pages 16–19.

Wang, J., Xu, M., Wang, H., and Zhang, J. (2006). Classification of imbalanced data by using the smote algorithm and locally linear embedding. In *2006 8th international Conference on Signal Processing*, volume 3.

Appendix A

Abbreviations

- **LR**: Logistic Regression
- **kNN**: k-Nearest Neighbors
- **ANN**: Artificial Neural Network
- **DNN**: Deep Neural Network
- **CNN**: Convolutional Neural Network
- **RNN**: Recurrent neural network
- **LSTM**:Long short-term memory
- **MCC**: Matthews correlation coefficient
- **OHLCV**: open, high, low and close prices and trading volume
- **Prev**: Distance to price level of previous reversals

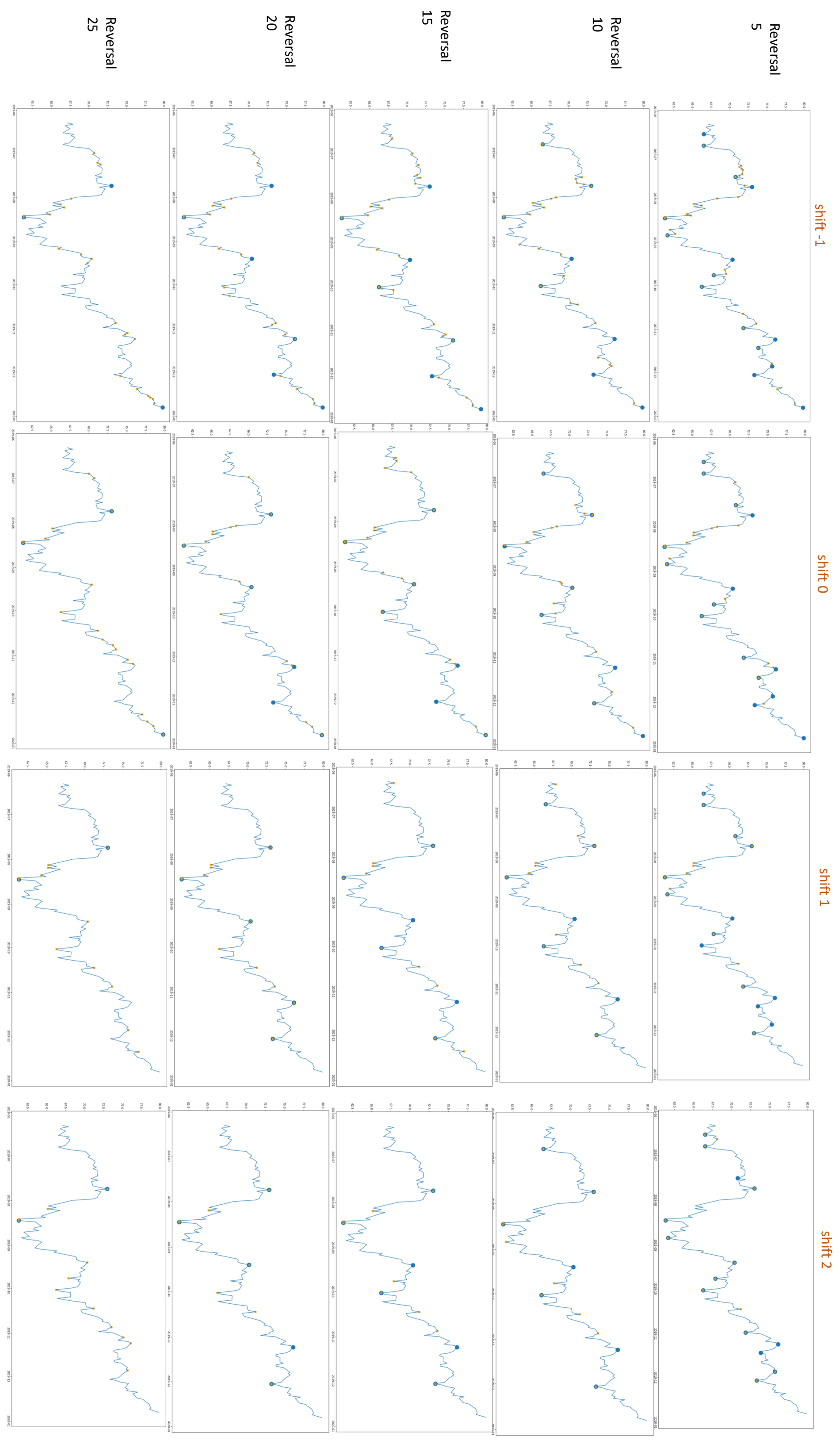


Figure A.1: Visualization of defined reversals (blue) and the predicted reversals (orange). A prediction confidence filtering was made. The model predicted values between [0, 1] for every time step. Out of the reversal event predictions (y>0.5), only the 65 percentile were plotted.

Appendix B

The profile of this thesis

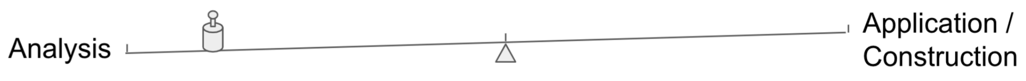


Figure B.1: Weight on this thesis project

- **Analysis**: Preliminary studies, discussion, knowledge acquisition, method choice, tool choice, problem specification, problem solving
- **Application/Construction**: design, implementation/construction, testing/evaluation
- **Other things that are added to the degree project**: time planning, meetings, report writing, prepare and hold presentation as well as opposition

Table B.1: Division of labour between authors

	Nicolo Ridulfo	Li Zhu
Analysis	50	50
Design	50	50
Implementation/Construction	50	50
Testing/Evaluation	50	50
Report och presentation	50	50