National Taiwan University

Financial Technology

Project 2: Stock Price Prediction

Lecturer:
Che Lin (林澤)

Author: Tadeo Hepperle (胡彬彬), Student ID: A11922105

October 25, 2022

Contents

1 Introduction

In this project historical data on the Apple Inc. stock is used to predict the stock's price on the next day. This is a regression task for which we are going to compare multiple methods:

- A simple model always using the current day's Close price to predict the next day's Close price.
- A moving average model, that predicts the next day's Close price by a learned linear combination of the past 30 Close prices.
- An RNN Cell with one layer
- An LSTM Cell with one layer
- A GRU Cell with one layer

1.1 Description of the data

The dataset contains various technical features concerning the Apple Inc. stock price in the time interval between 2011/01/03 and 201/12/31. There is a total of 754 data points. For each data point the following 6 features are available:

- Open - price of the stock when markets opened
- High - highest price of the stock on that day
- Low - lowest price of the stock on that day
- Close - price of the stock when markets closed
- Volume - the total volume of the stock traded
- Adjusted Close - stock's value after accounting for any corporate actions

In addition to these inherent features, 5 other features were computed:

- MA10 - 10 day moving average
- MA30 - 30 day moving average
- K - from the KD-indicator line chart
- D - from the KD-indicator line chart
- Weekday - one hot encoded into 7 different binary features

The data points are continous without gaps except for weekends and other days where the stock market is closed. For method selection and model evaluation the data was split into train, validation and test sets:

- Train set - stock price from 2011/01/01 - 2012/12/31
- Validation set - stock price from 2013/01/01 - 2013/06/30
- Test set - stock price from 2013/07/01 - 2013/12/30

1.2 Description of methods

The methods used are briefly described in the following sections to provide a deeper understanding.

1.2.1 Last Value Model

The simplest model to predict the next day's Close price is to just use the current day's Close price as the prediction. This model does not need any training and can be understood as a baseline model other models can be compared to in terms of their train and test loss.

1.2.2 Moving Average Model

A moving average in it's more general definition is a linear combination of the values of a subsequence of a sequence. In this model we want to form a linear combination of the past 30 Close values to predict the next Close price. For this we need to learn 30 weights.

1.2.3 Recurrent Neural Network

The recurrent neural network (RNN) is specified by 2 parameters: *input_size* and *hidden_size*. In this project we always just use an RNN with 1 layer. The RNN learns over a period of $time\_steps = 30$ iterations. In each iteration, an input vector of size *input_size* and a hidden state vector of size *hidden_size* are concatinated into a combined vector which is fed through a ReLU-activated linear layer to produce the hidden state for the next iteration. In the initial iteration the hidden state is a zero-vector. After the *time_steps* iterations of this process, the hidden state is mapped by a linear layer to a single scalar value (the Close price we want to predict). Mini-batch gradient descent is used to train such an RNN.

1.2.4 LSTM

LSTM stands for long short-term memory. The LSTM tries to improve on the RNN by using a cell state in addition to the hidden state, which is also a vector of *hidden_size*. while the hidden state represents a short term memory because the gradients deteriorate over time, the cell state is connected via constant gradient flow and can store information for much longer.

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$
$$\tilde{c}_t = \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$
$$h_t = o_t \odot \sigma_h(c_t)$$

in the equation abobe x_t stands for the input feature vector at a given time t starting from $t = 1$ up to $t = 30$ in our case. h_t represents the hidden state, while c_t is the cell state. In each iteration of the forward pass the formulas above are applied, to generate a new h_t and c_t until finally h_{30} is mapped by a linear layer to a single scalar: the predicted Close value of the next day.

In theory, one could have multiple layers in each iteration, we will just use 1-layer LSTMs and GRUs in this project.

1.2.5 GRU

GRU stands for gated recurrent unit. In comparison to the LSTM it lacks the input/update gate and the cell state but has been found to produce similar results at a lower computational cost in many applications. In each forward pass the following formulas are applied to get from the input x_t and the hidden state h_t to the next hidden state:

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z)$$
$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r)$$
$$\hat{h}_t = \phi_h(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h)$$
$$h_t = z_t \odot \hat{h}_t + (1 - z_t) \odot h_{t-1}$$

2 Results

First we will provide descriptive statistics and graphics of the dataset before moving on to compare the regression methods with each other.

2.1 Descriptive Statistics

The price of the Apple Inc. stock fluctuated between \$67.9 and \$134.1 in the given time frame. A volume between \$800.000 and \$15.000.000 was traded every active day. The 3 charts in Figure 2.1 display the price development, traded volume and KD-line in the entire timeframe of almost 2 years.

In the topmost candle-chart the blue line represents the 10-Day moving average, while the orange line is the 30-Day moving average.

The bottom chart shows the K (blue line) and D (orange line) statistics, both common technical indicators.

In the candlestick chart, red sticks represent days where the stockprice decreased, green sticks indicate an increase in the price. The same color coding

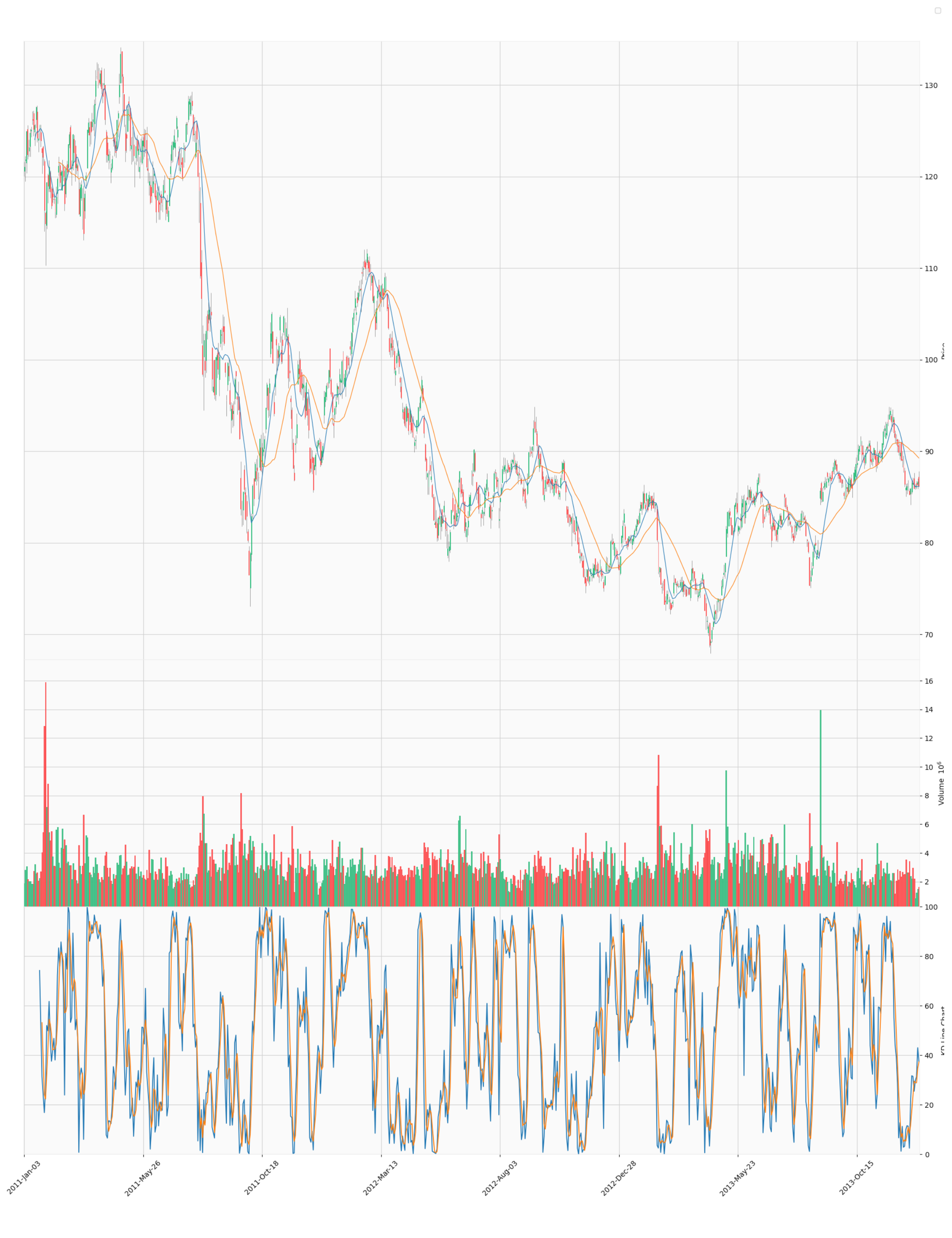


Figure 1: Sequence diagram

is also used for the bars of the volume chart. We see that the proportion of increasing days is roughly equal to decreasing days. The KD-chart oscillator curve looks mainly random, we cannot observe a strong pattern from it. Before applying any prediction model the features and labels (Close price of the next day) were min-max normalized to a range between 0 and 1. The weekdays were computed from the date and one-hot-encoded.

2.2 Last Value Model

As a baseline for all further learning we need to know how well we can predict the next Close price with a very simple model: What if we just take the last day's Close price as the prediction? Such a model yields results on the 3 data set slices like displayed in Table 1 and gives price predictions like shown in Figure 2.

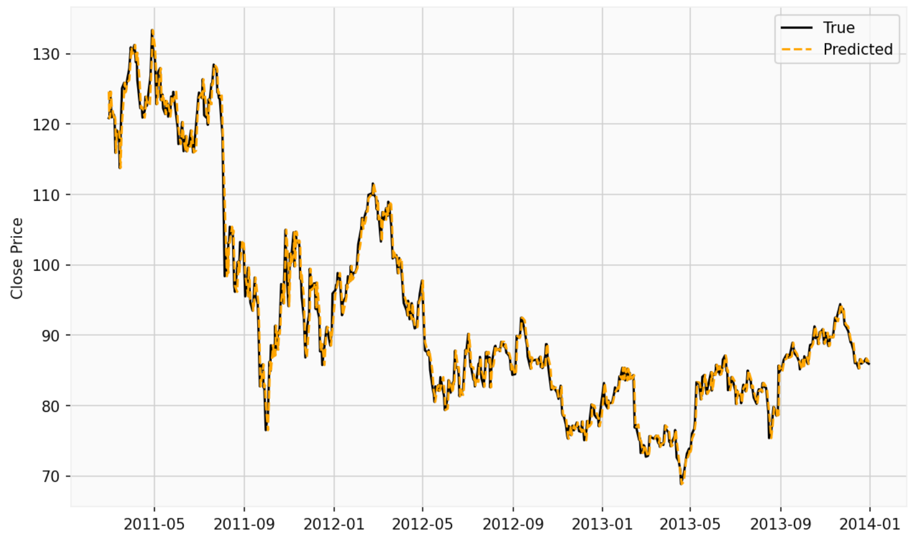


Figure 2: Predictions of the Last Value Model

Table 1: MSE of the Last Value Model on train, validation and test set

	Train set	Validation set	Test set
MSE	0.00102	0.00035	0.00028

It will be compared to other models below. The MSE is a popular loss function for regression tasks and is defined as

$$MSE = \frac{1}{n} \sum_i (y_i - \hat{y}_i)^2$$

where y_i are the true values and $\hat{y}_i$ are the predictions.

2.3 Moving Average Model

Pytorch was used to train a neural network with 1 linaer layer that models the next Close Price as a Moving Average of the last 30 days with variable weights. The weights were initialized to resemble the Last Value model at first, s.t. $w = [0, 0, \ldots, 0, 1]$ and $Card(w) = 30$. The validation set was used to determine when to stop the training. There was no visible overfitting on the validation set, that's why it was run for 800 epochs, with batchsize 32 and a learning rate of 0.001 and showed good convergence.

The final moving average coefficients resemble a lot of similarity to the Last Value Model: The vector looks a lot like this: [..., 0.256, 0.796], where all preceding values are smaller than 0.120 in absolute value. This means that still the most recent value has the greatest weight in predicting the next Close value. Loss curve and predictions on the test set can be viewed in Figure 3 and Figure 4.

Loss statistics in terms of MSE can be viewed in Table 2.

2.4 RNN, LSTM and GRU

for the RNN, LSTM and GRU via hyperparameter tuning the following paramters were deemed best:

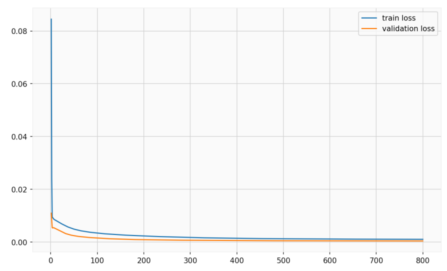
- *epochs* = 50 (RNN, GRU) | 100 (for LSTM) - after 50 epochs there was visible overfitting as the validation test error began to rise again. The LSTM took a bit longer until the overfitting zone was hit.

- $learning\_rate = 0.001$ - a larger learning rate converged faster but the MSE did not go as low and jumped around more. Smaller learning rates increased the number of EPOCHS needed without yielding better results.
- $size\_of\_hidden\_state = 64$ - seemed appropriate, but did not affect the outcome too much (32 and 128 were tried as well).
- $size\_of\_input = 17$ - all available features were used, 7 of which are the one-hot-encoded weekdays, other than that: *Close*, *Open*, *Low*, $High$, *Adj.Close*, $Volume$, $MA10$, $MA30$, K, D.
- $batch\_size = 32$ - gave good results.

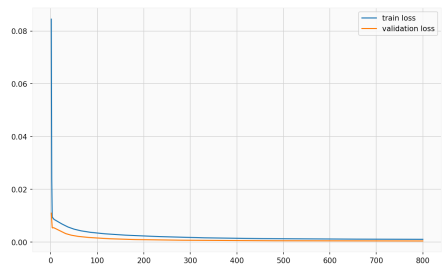
Notice, that the paramters were the same for all 3 models (RNN, LSTM, GRU), except for the $epochs = 100$ in the LSTM. The cell state size in the LSTM equals the hidden size (64). All models used a time frame of 30 days for the predictions. Figure 3 shows how the loss developed during training. Sadly because the first 2-3 Epochs had a quite dramatic loss, the following continous decline in the MSE is not visible too well. Notice that the MSE is related to the normalized stock price values, that range from 0 to 1. The real MSE of the price in US-dollars is likely around x times as high, where

$$x \approx (max(Close) - min(Close))^2 \approx (134.1 - 67.9)^2 \approx 4382$$

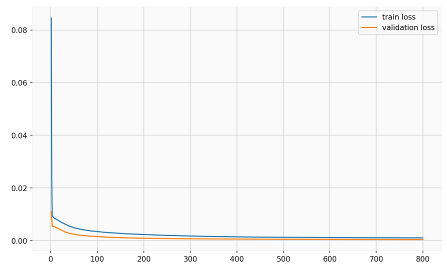
Figure 4 shows how the 4 different models perform on predicting unseen values in the test set given the last 30 observations. As we can see, all of them somewhat seem to have learned the representation of the data and make use of it. But if we look at the values in Table 2 we see that all of the models perform similarly but none of them outperforms the simple Last Value Model. In experimentations with more epochs it was possible to push the train error down to as little as 0.00007 for RNN, GRU and LSTM, however at the cost of increasing validation and test error. This is not desirable and shows overfitting. Thus given the limited data, none of the models seems to be actually a good choice for the prediction of the values in the test set. In fact the MSE of the neural network models seems



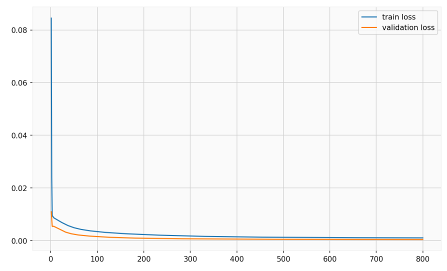
(a) Moving Average Model



(b) RNN



(c) LSTM



(d) GRU

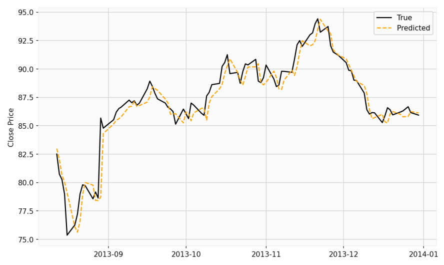
Figure 3: Train and validation set losses (MSE) of Moving Average Model, RNN, LSTM and GRU throughout training

to be almost two times as high as the MSE of the baseline Last Value Model on the test set as the column ”*Comparison*” in Table 2 shows.

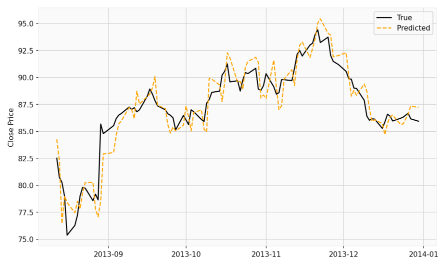
Table 2: MSE of the Last Value Model on train, valaidation and test set

MSE	Train	Validation	Test	Comparison
Last Value Model	0.00102	0.00035	0.00028	100%
Moving Average Model	0.00103	0.00042	0.00034	121%
RNN	0.00106	0.00087	0.00055	196%
LSTM	0.00112	0.00058	0.00050	179%
GRU	0.00118	0.00050	0.00050	179%

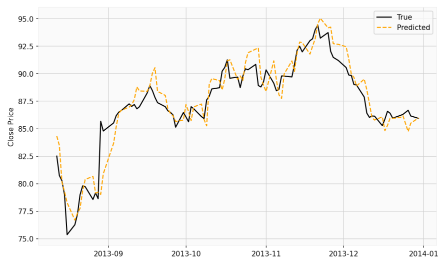
RNN, LSTM and GRU show similar performance on this task. The observed MSE rates are not different enough to provide evidence that one would be better than another. Also the lower test MSE of GRU and LSTM might be more due to randomness and is not wildly different from the pure RNN.



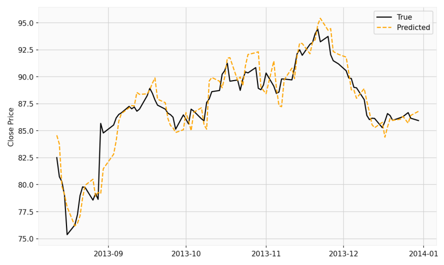
(a) Moving Average Model



(b) RNN



(c) LSTM



(d) GRU

Figure 4: predcitions on the test set (dotted line) of Moving Average Model, RNN, LSTM and GRU

3 Discussion

In economics there is the theory of efficient markets that states that all publically know factors considering a stock are already priced in, in the stock price and therefore the future value of a stock does not follow any predictable pattern. If there was such a pattern, whoever finds it, would abuse the pattern to make money and in the process destroy the pattern through the stock transactions themselves. Since we only use publically known past prices for our prediction it is not surprising that the last day's Close value seems to be the best prediction for the next day. Also having such limited data (less than 500 datapoints) in the train set does not help the model to detect such a pattern if there was one. Our models seem to overfit the data quite quickly, meaning they pick up on noise in the train set. Their predictions on the test set are, as we saw, worse than if one would just have taken the last Close value. Maybe using more datapoints could help to bring out the qualities of recurrent neural networks. One strategy would be to use not daily but secondly stock prices to predict movements even within days. In this way we could easily have more than 10.000 times as many datapoints, if we had stock data for every second instead of every day.