Stock market prediction

Eine Stock market prediction soll Aktienkurse voraussagen und so dem Anwender / Benutzer einen maximalen Gewinn an den Aktienmärkten ermöglichen. Die Analysemöglichkeiten sind dabei vielschichtig, lässt sich aber in zwei Analysebereiche zusammenfassen.Zum Ersten ist dort die technisch/mathematische Analyse, diese trifft Preisrichtungsvorhersagen bzw. Wertevorhersagen anhand vorhandener Aktienverläufe. Das Zweite ist eine analytische Methode, welche sich vor allem mit komplexeren Daten beschäftigt. Datenquellen sind hier oft Quartalsberichte und öffentlich zugängliche Informationen. Diese Daten müssen jedoch, um ein System zu trainieren, noch seperat aufgearbeitet werden.

Projektablauf

geeignetes Neuronales Netz finden → **Auswahl eines Algorithmen** → **Auswahl von Vergleichsalgorithmen**

- geeignetes Neuronales Netz finden: Convolutional Neural Network CNN oder Recurrent Neural Network (RNN)
- Auswahl eines Algorithmen: LSTM, Long short-term memory
- Auswahl von Vergleichsalgorithmen: SVR / SVM & Reinforcement mit QLearning

Phase 1:

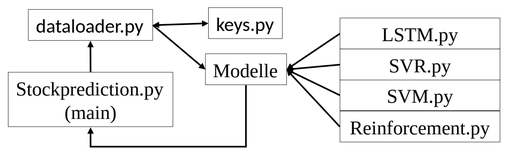
Abgriff von Livedaten / Daten über Schnittstellen

Alpha Vantage API
time-series intraday, Modellbildung und Training der Modelle

Phase 2:

Datenaufbereitung → **Programm** → **Training**

- Datenaufbereitung: Trennung von Trainings- und Testdaten
- Programm:



- Training: Modellentwicklung pro Aktie

Phase 3:

Vorhersage

Erzeugen eines Dataplotes mit der Vorhersage von x-Werten in die Zukunft

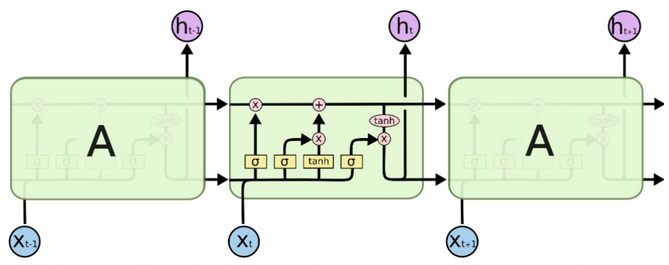
Automatisierung

Kauf und Verkauf
Realisiert in der reinforcement.py über states & actions

Phase 4:

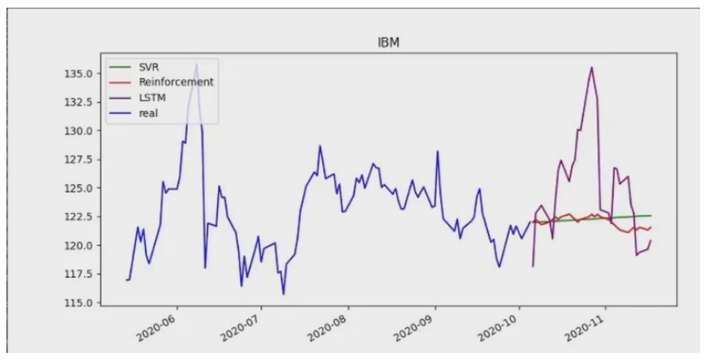
Training

Warum wir Long short-term memory (LSTM) benutzen ?



Als Fazit kann man sagen, dass wir uns speziell für LSTM entschieden haben, da wir seine Stärken vor allem in der Vorhersage von Zeitreihen sehen. Als Alternative sehen wir das Hidden Markov Modell (HMM), einen Vorgängeralgorithmus. Nach eingehender Recherche sehen wir hier die Nachteile zum LSTM vor allem in der Implementierung und teilweise Recht komplexen Problemstellungen in der Programmierung. Bei der Implementierung müsste sehr genau auf den Informationsgehalt der vorhandenen Daten geachtet werden. Ein Problem könnte hier sein, dass ein falsches oder sich änderndes Modell nicht sofort entdeckt wird. Der LSTM als Nachfolger der HMM bietet durch das Umgehen des im Backpropagtion erzeugten Rundungsfehlers, roundoff errors, bereits einige gute Problemlösungen.

Vorhersage



Es werden x-Tage eingestellt, welche die Algorithmen Werte in der Zukunft errechnen

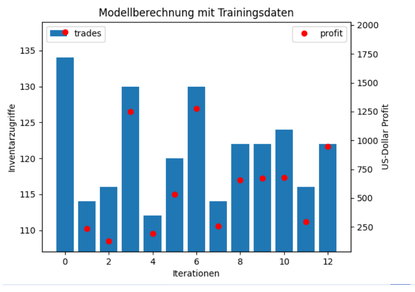
- Lila: LSTM
- Rot: Reinforcement
- Grün: SVR
- Blau: reale Daten bis zum heutigen Tag

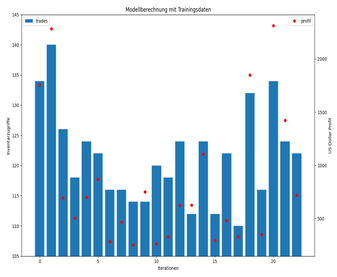
SVR: Es werden bis zu 100 Werte vorausgesagt. Je weiter sich die Voraussage vom letzten tatsächlichen Wert entfernt desto unsicherer wird die Berechnungsgrundlage.
Immer mehr errechnete Werte werden verwendet, um einen weiteren Wert zu errechnen, während der Anteil der realen Werte immer weiter abnimmt.

Automatisierung

Ausführung:

- Modell der reinforcement.py lernt mit 75 % des verfügbaren Strings, n[0.75:]
- Bei jedem neuen Wert n[:n+1] wird eine Action erzeugt, die dem Programm sagt, ob er kaufen, verkaufen oder die Aktie halten soll
- Beim Kauf: Aktienwert wird in ein Inventarplatz des Agenten gesteckt
- Beim Verkauf: Verkaufspreis - letzter Wert der Aktie des Inventars
  - Erzeugen von reward für Trainingsoptimierung





Modellbildung:

- Modell A : 10 Modelle
- Modell B: 22 Modelle
- Speicherung jedes 10te Iteration
  - A(0&10) || B(0,10,20)
- Eingangsstrings sind gleich
- Inventarzugriffe sind Käufe und Verkäufe!
- Startgeld bei beiden 10000 US-Dollar

Autonomes Kaufen / Verkaufen:

- Test läuft bis zum Ende des in der Api aufgerufenen Strings [0.75:.....]
- Selbstständiger Kauf und Verkauf durch die in Modell 10 bestimmten Werte

```
DATA
Am 40.Tag besitzt: 0 kauft zu: $775.97
Am 41.Tag besitzt: 1 verkauft zu: $784.80 -> Profit: $8.83
Am 44.Tag besitzt: 0 kauft zu: $780.23
Am 45.Tag besitzt: 1 verkauft zu: $785.79 -> Profit: $5.56
Am 47.Tag besitzt: 0 kauft zu: $775.88
Am 48.Tag besitzt: 1 verkauft zu: $764.33 -> Verlust: -$11.55
Am 58.Tag besitzt: 0 kauft zu: $815.65
Am 59.Tag besitzt: 1 verkauft zu: $809.84 -> Verlust: -$5.81
Am 62.Tag besitzt: 0 kauft zu: $812.20
Am 63.Tag besitzt: 1 verkauft zu: $809.68 -> Verlust: -$2.52
Am 64.Tag besitzt: 0 kauft zu: $807.80
Am 65.Tag besitzt: 1 verkauft zu: $809.93 -> Profit: $2.13
Am 66.Tag besitzt: 0 kauft zu: $804.57
Am 67.Tag besitzt: 1 kauft zu: $802.88
Am 68.Tag besitzt: 2 verkauft zu: $792.45 -> Verlust: -$12.12
Am 69.Tag besitzt: 1 verkauft zu: $808.01 -> Profit: $5.13
Am 83.Tag besitzt: 0 kauft zu: $849.53
Am 84.Tag besitzt: 1 verkauft zu: $858.45 -> Profit: $8.92
Am 85.Tag besitzt: 0 kauft zu: $856.98
Am 86.Tag besitzt: 1 verkauft zu: $845.03 -> Verlust: -$11.95
Am 88.Tag besitzt: 0 kauft zu: $820.19
Am 89.Tag besitzt: 1 verkauft zu: $815.24 -> Verlust: -$4.95
Am 90.Tag besitzt: 0 kauft zu: $818.26
Am 93.Tag besitzt: 1 verkauft zu: $829.23 -> Profit: $10.97
--------------------------------
GOOGL Total Profit: -$7.36
--------------------------------
Total profit is: -$7.36
```


Autonome Systeme Sommersemester 2020
Projektgruppe: ML-Based-StockMarket-Prediction
Projektmitglieder: Jan Dehlen, Mahdi el-Mesoudy und Alexander Klee