Thema:

Investition in Gold, Silber oder Pfizer?

Machine Learning Algorithmen als Entscheidungshilfe

Ausarbeitung

Im Rahmen des Moduls Big Data-Anwendungen und Bildanalyse

Im Fachgebiet Wirtschaftsinformatik

An der Technischen Hochschule OWL Höxter

Themensteller: Prof. Dr. Rer. Nat. Burkhard Wrenger

Betreuer: Prof. Dr. Rer. Nat. Burkhard Wrenger

Vorgelegt von:Jeffrey Böttcher
Nederlandstr.5
32825 Blomberg
01625775231
jeffrey.boettcher@stud.th-owl.de

Inhaltsverzeichnis

1. Einleitung

Mit der Zeit der Digitalisierung hat sich auch die Welt der Investitionen geändert. Vor allem in den letzten Jahren gab es enorme technologische Fortschritte. Wir leben in einer Ära, in dem Daten so wertvoll sind wie Gold. Big Data Anwendungen bieten Investoren, oder jene die investieren wollen, revolutionäre Möglichkeiten.

Diese Ausarbeitung widmet sich der Entwicklung und Implementierung einer entscheidungsfördernden Anwendung in Python, um Investitionsentscheidungen in verschiedenen Anlageklassen zu unterstützten. Primär geht es um die Algorithmen des maschinelles lernen.
Primär vorgestellte Investitionsklassen sind Gold, Silber und die Pfizer Aktie.

Da die herkömmlichen Methoden der Anlageberatung und –Analyse durch exponentiell wachsenden Datenmengen an ihre Grenzen stoßen, bietet Maschinelles Lernen eine innovative Lösung. Maschinelles lernen kann riesige Datensätze Analysieren, Muster erkennen und Vorhersagen treffen. Die Ausarbeitung präsentiert nicht nur eine theoretische Auseinandersetzung mit den Grundlagen der Investition in verschiedene Vermögenswerte, sondern setzt einen Schwerpunkt auf die Praktische Umsetzung durch die Entwicklung einer eigenen Anwendung in Python.

Durch den Einsatz von maschinellem Lernen und Data Mining wird diese Anwendung Einblicke in die Preisentwicklungen von Gold, Silber und der Pfizer-Aktie liefern. Die erzielten Ergebnisse werden verglichen und analysiert, um Muster und Trends in den Preisdaten zu identifizieren.

Obwohl die Anwendung zunächst nicht auf Echtzeitverarbeitung basiert, aufgrund experimenteller Gründe, besteht die Möglichkeit den Code leicht anzupassen, um Echtzeitverarbeitung zu ermöglichen. Dies ermöglicht eine schnellere und aktuellere Analyse der Preisentwicklungen, wodurch Anleger und Händler von aktuellen Marktbewegungen profitieren können.

Diese Arbeit hat den Zweck, Ihnen einen eingehenden Einblick in die Entstehung und den Einsatz einer solchen Entscheidungshilfe zu bieten. Dabei werden die essenziellen Algorithmen präsentiert und detailliert erläutert. Hierbei wird zunächst ein kurzes theoretisches Kapitel eingeführt, gefolgt von der Vorstellung des Konzepts, welches dann als vollständig entwickeltes Programm präsentiert wird.

1.1 Zielsetzung

Das Hauptziel meiner Ausarbeitung "Investition in Gold, Silber oder Pfizer? Machine Learning Algorithmen als Entscheidungshilfe" ist es, verschiedene Machine Learning Algorithmen vorzustellen, die als effektive Unterstützung dienen können, um die Auswahl zwischen Investitionen in Gold, Silber oder der Pfizer Aktie zu erleichtern. Dabei werden die Algorithmen und ihre Ergebnisse gegenübergestellt, um fundierte Schlussfolgerungen zu ziehen.

Der Fokus der Arbeit liegt auf der Erläuterung der Algorithmen und deren Bewertung, wobei die Aktienkurse selbst nicht im Mittelpunkt stehen. Aus diesem Grund beschränkt sich die Untersuchung auf drei bestimmte Anlageformen: Gold, Silber und Pfizer-Aktien. Dennoch lassen sich die Skripte leicht anpassen, um sie auf andere Kurse anzuwenden.

Als Ergebnis soll festgestellt werden, welche Methode die besten Prognosen im Vergleich ermöglicht.

2 Grundlagen

In diesem Abschnitt führe ich Sie in die Grundlagen der Aktienanalyse ein und erläutere die Verwendung von Algorithmen als Entscheidungshilfe. Der Einsatz von Algorithmen in der Aktienanalyse basiert auf dem Konzept des maschinellen Lernens, einer Unterdisziplin der künstlichen Intelligenz. Maschinelles Lernen ermöglicht Computern, Muster und Zusammenhänge in umfangreichen Datensätzen zu erkennen, um daraus präzise Vorhersagen und fundierte Entscheidungen abzuleiten. In der Finanzwelt haben sich verschiedene Algorithmen als äußerst wirkungsvoll erwiesen, und ich werde Ihnen einige davon näher erläutern.

Ein weiterer essenzieller Aspekt der Grundlagen ist die Vertiefung in neuronale Netze, insbesondere die Beachtung von rekurrenten neuronalen Netzen.

Abschließend im Grundlagenabschnitt werde ich die Schlüsselalgorithmen vorstellen, die in der Zeitreihenanalyse erfolgreich eingesetzt werden können.

2.1 Aktienanalyse

Die Aktienanalyse der entscheidende Punkt, um festzustellen mit welchen Chancen man zu rechnen hat und um die Risiken zu minimieren.

In der Aktienanalyse gibt zwei etablierte Methoden der Analyse: die Fundamentale Analyse und die Technische Analyse.

Da wir uns Primär auf die Algorithmen konzentrieren wollen reiße ich das Thema kurz an, damit Sie die grundlegende Idee dahinter zu verstehen.

2.1.1 Fundamentale Analyse

Anleger können mithilfe der Fundamental Analyse den sogenannten inneren, unter anderem auch als wahren oder fairen Wert bezeichnet, Wert eines Unternehmens ermitteln.[1] Die These ist das an dem Inneren Wert eines Unternehmens auch der Börsenwert anpassen wird. Sprich ein Wert welches unter dem Börsenkurs liegt spricht gegen den Kauf einer Aktie und ein Wert, der über den Börsenwert liegt, spricht für den Kauf der einer Aktie.

Die Fundamentale Analyse ist eine sehr komplexe und ausführliche Analyse, die jeweils mehrere Analysen zusammenbringt und vereint.

In der Fundamentalanalyse[2] werden üblicherweise drei Hauptkomponenten betrachtet: die Globalanalyse, die Branchenanalyse und die Unternehmensanalyse. Diese Analyseformen sind integraler Bestandteil bei der Bewertung von Aktien und bieten einen umfassenden Einblick in die wirtschaftlichen und finanziellen Aspekte eines Unternehmens.

Die Berechnung des inneren Werts einer Aktie ist für den durchschnittlichen Anleger oft eine anspruchsvolle Aufgabe. Dies liegt daran, dass ein solcher Prozess Einblicke erfordert, die aufgrund der veröffentlichten Daten, einschließlich Bilanzen, selbst bei höchster Sorgfalt und Fachkenntnissen in der Bilanzanalyse, für externe Beobachter nahezu unzugänglich sind. Angesichts der beeindruckenden Anzahl von etwa 43.000 börsennotierten Unternehmen[3] weltweit gestaltet sich die individuelle Berechnung selbst mit vollständigen Kennzahlen als nahezu unmöglich. In diesem Zusammenhang könnte die Unterstützung von Informationstechnologie (IT) eine entscheidende Rolle spielen.

Die Einbindung von Neuronalen Netzwerken und Algorithmen des maschinellen Lernens bieten einen vielversprechenden Weg, die Herausforderungen der Bewertung von Aktien zu bewältigen. Die Nutzung dieser fortschrittlichen Technologien könnte es ermöglichen, die Komplexität der Analyse zu reduzieren und gleichzeitig zu präziseren Ergebnissen zu gelangen.

In der Praxis konzentriere ich mich in dieser Arbeit besonders auf die Technische Aktienanalyse. Diese Analysemethode erfordert im Vergleich zur Fundamentalanalyse weniger zeitlichen Aufwand und kann dennoch qualitativ hochwertige Ergebnisse liefern. Die Technische Analyse nutzt Kursmuster und historische Preisbewegungen, um zukünftige Entwicklungen vorherzusagen. Diese

[1] Vgl. Prof. Dr. J. Welcker und Jörg Audörsch(1994),Technische Aktienanalyse, S. 3
[2] Link zur näheren Definition
[3] The World Bank.(2020). Listed domestic companies, total. Verfügbar unter : https://data.worldbank.org/indicator/CM.MKT.LDOM.NO (Abrufdatum: 1. Februar 2024)

Methode kann besonders effizient sein und erfreut sich großer Beliebtheit, da sie für Anleger mit unterschiedlichem Erfahrungsniveau zugänglich ist.

In diesem Kontext werde ich die Möglichkeiten des Maschinellen Lernens hervorheben und den Fokus auf die Technische Aktienanalyse legen. Diese Betrachtung soll verdeutlichen, wie moderne Technologien Investoren unterstützen können, insbesondere wenn es um die Handhabung großer Datenmengen geht und komplexe Muster in den Finanzmärkten analysiert werden müssen.

2.1.2 Technische Analyse

Die technische Analyse basiert auf der Annahme das Historische Aktien Preisbewegungen Muster und Trends bilden, die sich wiederholen werden.

In Gegensatz zu der Fundamentalanalyse sind die Daten die man für die Berechnung brauch leicht zugänglich. Eines der häufigsten Methoden der Technische Analyse ist die Chartmuster Analyse. Charts sind Kursmuster die Preisbewegungen Grafisch darstellen. Es werden mustern gesucht, die auf zukünftige Kursentwicklungen deuten können.

Beispiel einer Chartanalyse:

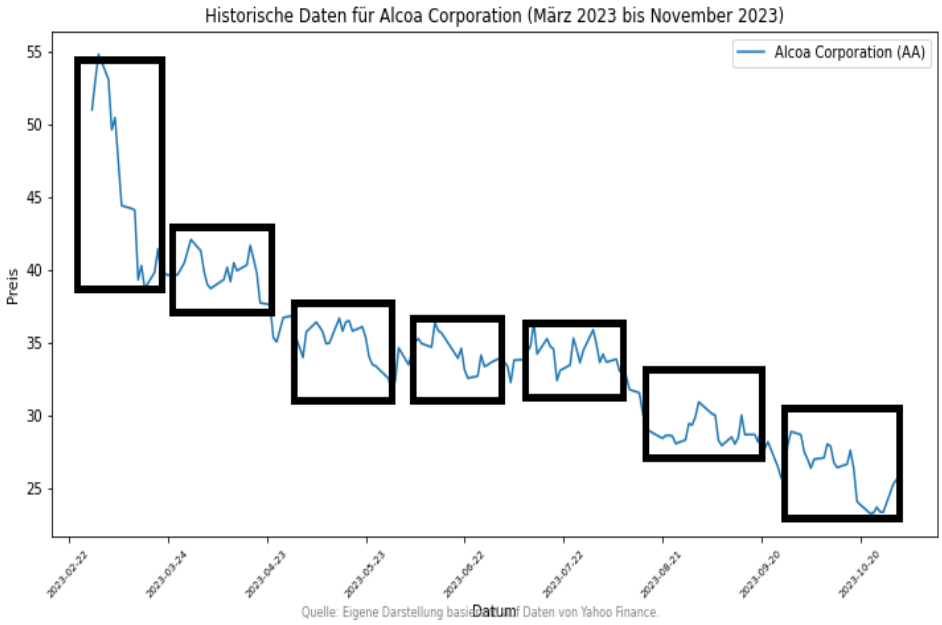


Abbildung 1: Eigene Darstellung basierend auf Daten von yFinance und der Plot Funktion von Spyder

Die schwarzen Kästen sind Charts und sollen verdeutlichen das es bei jedem Chart einem Abwärtstrend gibt. Prognosen anhand dieser Analyse festzustellen ist ebenfalls sehr aufwendig und Tagesprognosen zu schätzen oder zu erraten auch sehr risikoreich.

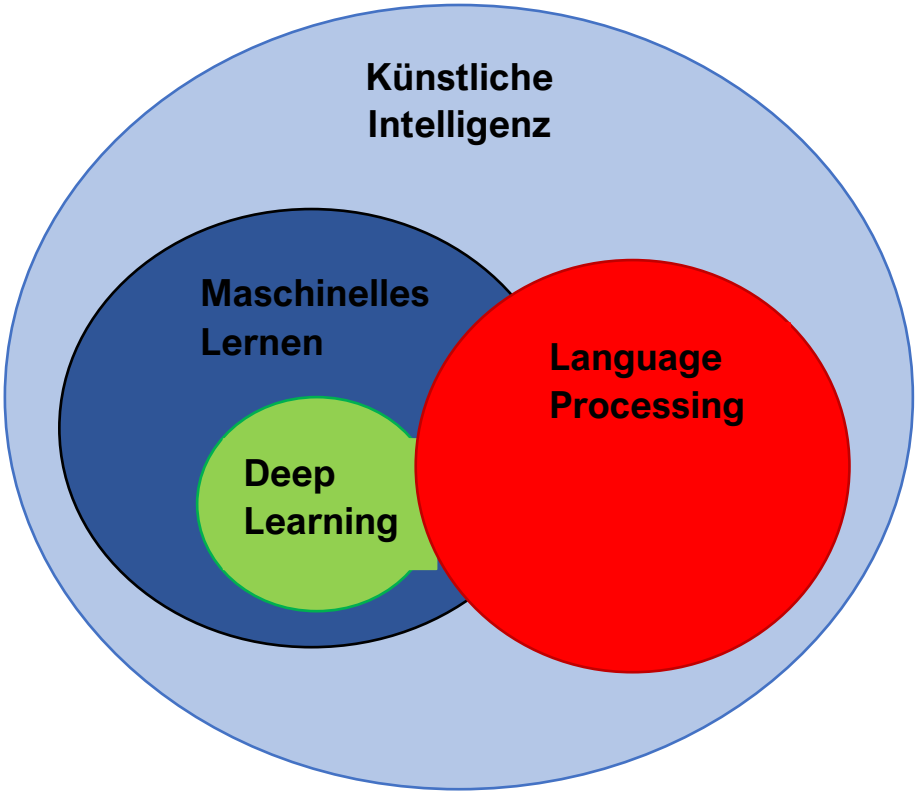
Um das an jeder Aktie handlich anzuwenden, bräuchte man viel Zeit und wäre sehr ineffizient für die heutige Zeit.

Dennoch möglich und nicht so aufwendig wie bei der Fundamentale Analyse.

Es gibt jedoch Maschinelle Algorithmen, die viel schneller und automatisierend Prognosen erstellen können. Auf diese Algorithmen möchte ich näher eingehen und ihnen näherbringen.

3. Prognostizieren von Zeitreihen mit Maschinellem Lernen: Algorithmen im Fokus

Maschinelles lernen ist ein Bereich der Künstlichen Intelligenz, welches sich mit der Entwicklung von Algorithmen beschäftigt, die durch die Verarbeitung von Daten die Fähigkeit entwickelt, aus Erfahrungen zu lernen.[4]



[4] Vgl. B. Botsch, S.1, Maschinelles Lernen – Grundlagen und Anwendungen

Abbildung 2: selbst erstellte Darstellung, Teilbereiche von KI

Maschinelle Lernmodelle können selbstständig Muster erkennen und mit diesen Wissen Entscheidungen treffen.

Es gibt unterschiedliche Arten/Modelle von maschinellem Lernen.

Überwachtes lernen ist ein Modell, welches mit bereits vorhandenen gelabelten Daten trainiert wird. Der SML-Algorithmus (überwachtes maschinelles Lernen) zielt darauf ab, zu erlernen, wie man eine spezifische Ausgabe basierend auf einem Satz von Eingabevariablen vorhersagen kann. Überwachtes maschinelles Lernen ist einer der gängigsten Modelle.

Deep Learning ist ein Modell, welches von der Funktionsweise des menschlichen Gehirns inspiriert ist.
Es werden neuronalen Netzwerke mit mehreren Schichten angewendet. Die Neuronalen Netzwerke können komplexe Muster und Hierarchien in Daten zu erfassen.
Darunter gibt es noch Unüberwachtes lernen, Halbüberwachtes Lernen, selbst überwachts Lernen und bestärkendes lernen.

Für unseren Fall benötigen wir nur das überwachte Lernen und Deep Learning, da wir mit einem Satz von Eingabe Daten und den entsprechenden Zielwerten (in diesem Fall den tatsächlichen Aktienpreisen) trainieren und wir Neuronale Netze nutzen werden.

4. Konzeption

Das Hauptziel besteht darin, mithilfe von Machine Learning Algorithmen Prognosen darüber zu erstellen, wie sich die Investitionen in Gold, Silber und Pfizer entwickeln könnte. Die Algorithmen Long Short Term Memory (LSTM), Gated Recurrent Unit (GRU), Lineare Regression, Support Vector Machines (SVM), Random Forests und Gradient Boosting Models werden dabei als Entscheidungsunterstützungssysteme verwendet. Ich werde die Algorithmen besprechen und vergleichen. Jedes der Algorithmen hat vor und Nachteile, die ich Anhand der Daten auswerten werde.
Schritt 1. Zuerst werden die Historischen Daten für Goldpreise, Silberpreise und Pfizer Schlusswerte von yFinance besorgt. Wir konzentrieren uns lediglich auf dem Preis und Zeitraum.
Schritt 2. Die Daten werden in Training- und Test-daten aufgeteilt um sie darauffolgend zu normalisieren, um verschiedene Skalen zu berücksichtigen.
Schritt 3. Die Daten werden auf Historische Daten trainiert.
Schritt 4. Evaluierung der Modelle mithilfe von Testdaten
Schritt 5. Ausgabe der Prognose

5. Algorithmen des Maschinellen Lernen

Für die folgenden Algorithmen betrachten wir den Zeitraum vom 6. Juni 2023 bis zum 1. Januar 2024. Vorhersage Tag lautet 2 Januar 2024.

Die Gold Aktie wird Pro Algorithmus einmal mit einem längeren Zeitraum getestet, um den Unterschied zu verdeutlichen. (Zeitraum: 2022.06.01-2024.01.01)

Wir nutzen die Aktiendaten für Gold (Ticker: GC=F), Silber (Ticker: SI=F) und das Pharmaunternehmen Pfizer (Ticker: PFE.DE) für diesen Vergleich. Die Daten werden von der Yfinance[5] Api zu Verfügung gestellt. Die farbige Aktienkurve zeigt die Vorhersagen für die Tage, die uns bereits bekannt sind (in der Theorie), und veranschaulicht gut, wie gut das jeweilige Modell lernt und prognostiziert. Die tatsächliche Vorhersage wird in der Konsole ausgegeben.
Im Anhang finden Sie den vollständigen Code, der Ihnen dabei hilft, ein umfassenderes Verständnis für die Algorithmen zu erlangen.

Benötigte Bibliotheken:
Pip install
yfinance
pandas_datareader
get-all-tickers
tkcalendar
Numpy
Pandas
Matplotlib
datetime
Scikit-learn

5.1 Long Short Term Memory

LSTM ist eine Form von Rekurrenten neuronalen Netzwerk.

[5] https://pypi.org/project/yfinance/

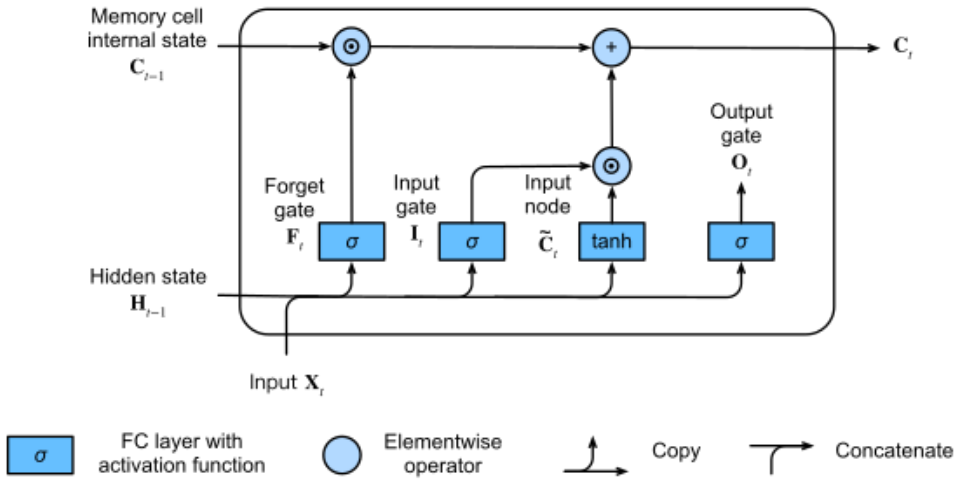


Abbildung 3: Long Short Time Memory (https://d2l.ai/chapter_recurrent-modern/lstm.html)

LSTM besteht aus der Zelle (Cell), Tore (Gates) und des Hidden State.
Die Zelle ist eine Grundlegende Einheit in den Informationen über einen längeren Zeitraum gespeichert werden kann.
Dazu hat LSTM Drei Tore, worin es Daten speichert. Input Gate, Forget Gate und Output Gate. Durch die Gates kann sich das Modell die Historischen Daten merken und dadurch zusammenhänge erschließen.

Das Forget Gate entscheidet welche Daten aus der vorherigen Zelle vergessen oder beibehalten werden sollen.
Das Input Gate entscheidet welche neuen Daten in die Zelle aufgenommen werden sollen.
Das Output Gate bestimmt, welche Daten an die nächsten Schichten des neuronalen Netzwerks weitergegeben werden.

Der sogenannte “Cell State” ist der Status der Zelle. Der Status wird durch die Tore gesteuert und ermöglicht relevante Daten über einen langen Zeitraum zu speichern. Der Cell State wird immer aktualisiert, basierend auf den Entscheidungen des Forget Gates und des Input Gates.

Last but not least, der Hidden State. Der Hidden State ist eine Art “Gedächtnis”. Jede Zelle erzeugt ein Hidden State. Der Hidden State enthält Informationen über das bisher gelernte und welche Muster in den Daten erkannt wurden.

CodeBeispiel:
“

1. *model = Sequential() #leeres Squentielles Model*
2. *model.add(LSTM(units=50, return_sequences=True, input_shape=(x_train.shape[1], 1)))*

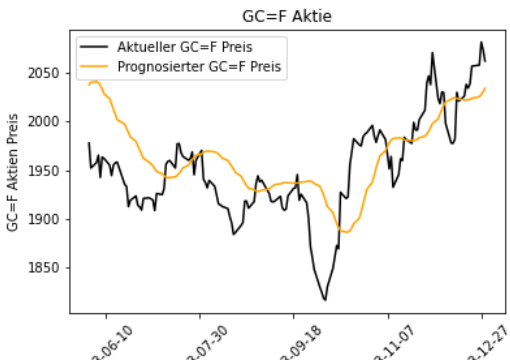
3. *model.add(Dropout(0.2))*
4. *model.add(LSTM(units=50, return_sequences=True))*
5. *model.add(Dropout(0.2))*
6. *model.add(LSTM(units=50))*
7. *model.add(Dropout(0.2))*
8. *model.add(Dense(units=1)) # Prediction of the next closing value*
9. *model.compile(optimizer='adam', loss='mean_squared_error')*
10. *model.fit(x_train, y_train, epochs=25, batch_size=32)*

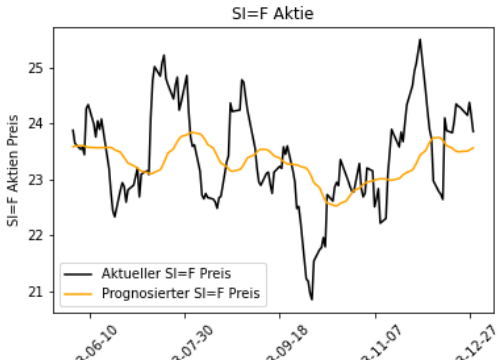
“

1. Leeres Sequentielles Modell
2. Erste Schicht hinzugefügt mit 50,, Neuronen”, gibt die ganze Sequenz zurück (Alle Hidden States)
3. Dropout Schicht wird hinzugefügt, um Over Fitting zu vermeiden.
4. Zweite Schicht wird hinzugefügt mit 50 “Neuronen”, gibt ebenfalls die gesamte Sequenz zurück
5. Zweite Dropout schicht hinzugefügt
6. Dritte Schicht die nur den letzten Hidden State zurückgibt
7. Weitere Dropout Schicht wird hinzugefügt
8. Dense Schicht (Vorhersage des nächstes Schlusswertes) mit 1 “Neuron”.
9. Kompilieren des Algorithmus. Nutzung des Adam Optimizer(Sehr effektiv bei Zeitreihenanalyse)
10. Das modell wird in 25 Trainings Epochen trainiert. Und die Batch_size beträgt 32.“Iterationen Pro Epoche = Anzahl der Trainingsdaten / Batchsize”.

Ergebnisse werden als Plot dargestellt:
Als Prognose für den nächsten Tag gilt für

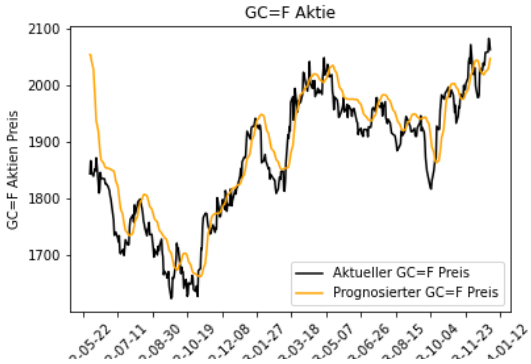
Gold = 2037.9667 Silber = 23.579666

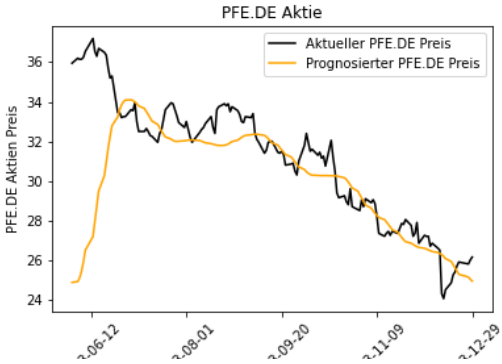




Gold von 2022-2024 = 2053.2659

Pfizer = 24.837078





Abbildungen 4-7: Eigene Darstellung von Aktien durch die Plot Funktion von Spyder

Tatsächliche Werte: Gold = 2064.40 Silber =23.73 Pfizer=29.73

Schlussfolgerung

Bei der Anwendung des LSTM-Algorithmus fällt auf, dass er mit einer größeren Datenmenge tendenziell präzisere Vorhersagen liefert. Die täglichen Vorhersagen (gelbe Kurve) sind genauer, wenn mehr Eingabedaten verwendet werden.

Insgesamt lässt sich feststellen, dass der Algorithmus auf den ersten Blick möglicherweise ungenau erscheint, jedoch liegt dies teilweise an den Eingabedaten. LSTM zeigt eine verbesserte Leistung mit einer größeren Datenmenge. Die Prognosen sind dennoch zufriedenstellend und bilden eine solide Basis für weitere Analysen und Entscheidungen.

5.2 Gated Recurrent Unit

Der Gated Recurrent Unit (GRU) Algorithmus arbeitet ebenfalls mit Toren und hat Ähnlichkeiten mit den LSTM verfahren. Der GRU-Algorithmus ist ähnlich aufgebaut und liefert teilweiße bessere Ergebnisse, weshalb es als „Fortschrittliche Variante" gesehen wird.

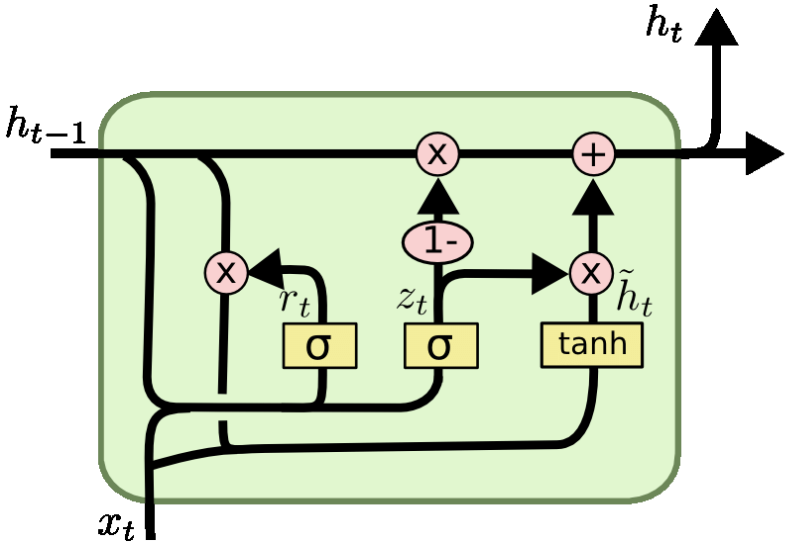


Abbildung 8: Darstellung Gated Recurrent Unit (https://www.researchgate.net/figure/Gated-Recurrent-Unit-GRU_fig4_328462205)

Gated Recurrent Unit besitzt nur zwei Tore. Das Update-Gate und das Reset-Gate.

Die Aufgabe des Update Gate besteht hauptsächlich darin, die optimale Menge an vorherigen Informationen zu bestimmen, die für zukünftige Prognosen relevant sind. Die Bedeutung dieser Funktion liegt vor allem darin, dass das Modell entscheiden kann, inwiefern vergangene Details in die Zukunft übertragen werden sollen.

Die Funktion des Reset-Gatters bestimmt, welche Informationen vernachlässigt werden sollen. Ein bedeutsamer Aspekt des Reset-Gatters kann durch einen Vergleich mit dem Forget-Gatter des Long Short-Term Memory (LSTM) verdeutlicht werden. Ähnlich wie das Forget-Gatter im LSTM dazu neigt, nicht zusammenhängende Daten zu erkennen und das Modell anweist, sie zu vernachlässigen, erfüllt auch das Reset-Gatter in der Gated Recurrent Unit (GRU) eine ähnliche Rolle. Es bestimmt, welche Informationen als irrelevant betrachtet und ignoriert werden sollen, wodurch das Modell in der Lage ist, sich auf die wesentlichen Aspekte der Vergangenheit zu konzentrieren und nicht relevante Details zu verwerfen. Dies trägt zur effizienten Verarbeitung von zeitlichen Sequenzen bei und hilft, das Modell vor Überbetonung unwichtiger Informationen zu schützen.

Code Beispiel:

“

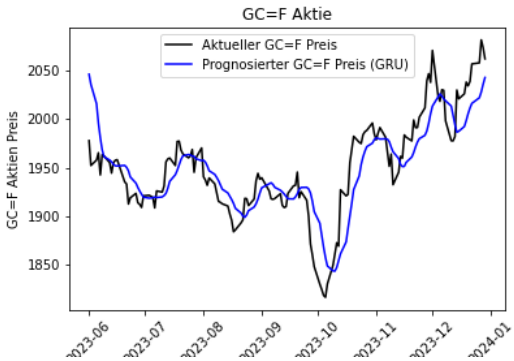
```
1. model = Sequential()
2. model.add(GRU(units=50, return_sequences=True,
   input_shape=(x_train.shape[1], 1)))
3. model.add(Dropout(0.2))
4. model.add(GRU(units=50, return_sequences=True))
5. model.add(Dropout(0.2))
6. model.add(GRU(units=50))
7. model.add(Dropout(0.2))
8. model.add(Dense(units=1))  # Prediction of the next closing value
9. model.compile(optimizer='adam', loss='mean_squared_error')
10. model.fit(x_train, y_train, epochs=25, batch_size=32)
```

”

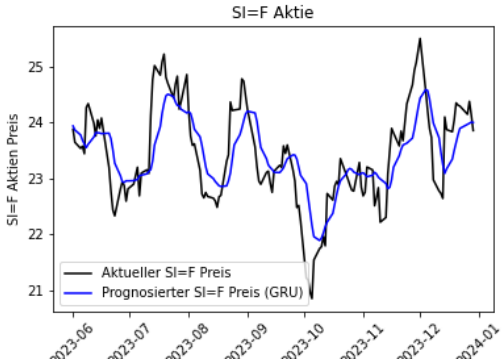
1. Intialisierung des models -> lineare Stapelung von schichten.
2. Erste Schicht wird hinzugefügt mit 50 “Neuronen”. return_sequences=True gibt an, dass die Schicht eine Sequenz von Ausgaben zurückgeben soll, die für den nächsten GRU-Layer relevant sind.
3. Dropout-Layer wird hinzugefügt. 0.2= entfernung von 20% der Neuronen, um Overfitting zu reduzieren.
4. Zweite Schicht mit 50 “Neuronen”.
5. Zweiter Dropout Layer
6. Dritte Schicht mit 50 “Neuronen”
7. Dritter Dropout Layer
8. Dense Schicht wird hinzugefügt. Jeder Ausgang der vorherigen Schicht ist mit jedem Eingang der Dense-Schicht verbunden.
9. Kompilieren des Algorithmus. Ebenfalls Nutzung des Adam Optimizer (Sehr effektiv bei Zeitreihenanalyse)
10. Training des Modells mit den Daten “x_train” und “y_train”. Rest wie bei LSTM

Ergebnisse werden als Plot dargestellt:
Als Prognose für den nächsten Tag gilt für

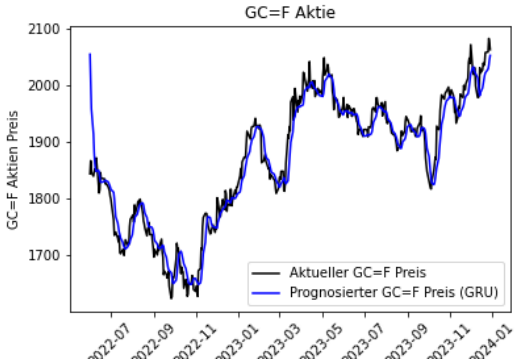
Gold = 2046.4331



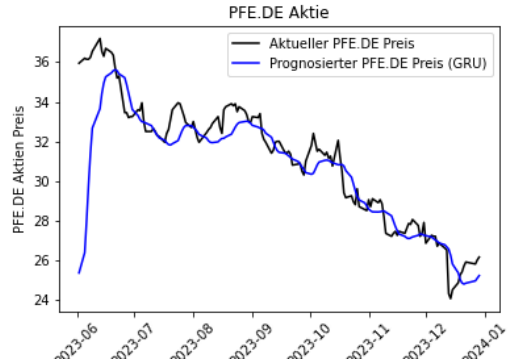
Silber = 23.94



Gold von 2022-2024 = 2054.082



Pfizer = 25.349



Abbildungen 9-12: Eigene Darstellung von Aktien durch die Plot Funktion von Spyder

Tatsächliche Werte: Gold = 2064.40 Silber =23.73 Pfizer=29.73

Schlussfolgerung

Im Vergleich zum LSTM-Verfahren sehen die Linien sauberer aus, also die vorhersagen im Allgemeinen. Zwar kommt das GRU-Modell mit weniger Daten besser aus als das LSTM-Model, jedoch schneidet es ebenfalls besser ab, wenn das Gru Model mit mehr Eingabe Daten gefüttert wird. Die vorhersagen sind auch näher an dem tatsächlichen Wert. Man kann zwar nicht pauschal sagen das dieses Verfahren besser ist, aber mit diesen Einstellungen, die vorgenommen wurden, scheint das GRU Model bessere Ergebnisse zu liefern.

5.3 Lineare Regression

Die lineare Regression ist eine sehr bekannte Mathematische Methode, die in der der Statistik angewendet wird. Im Python Code ist sie auch sehr simpel zu schreiben.

Code Ausschnitt:
“

 # Einfache lineare Regression erstellen und trainieren

1. *lin_reg = LinearRegression()*
2. *lin_reg.fit(x_train, y_train)*

...

3. *predicted_prices = lin_reg.predict(x_test)*
4. *predicted_prices = scaler.inverse_transform(predicted_prices.reshape(-1, 1))*

....

5. *prediction = lin_reg.predict(real_data)*
6. *prediction = scaler.inverse_transform(prediction.reshape(-1, 1))*

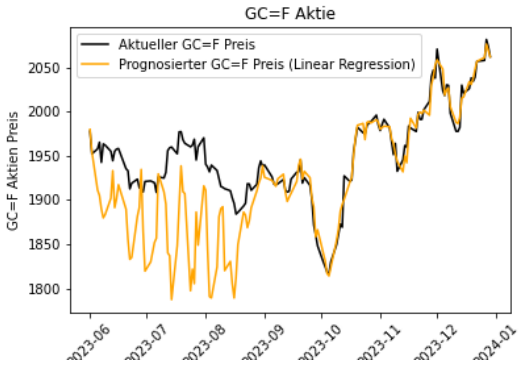
“

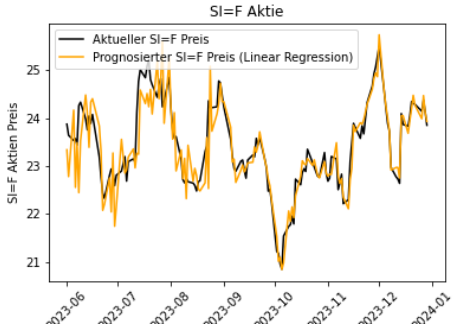
1. Objekt der Klasse “LinearRegression erstellen.
2. mit den Trainingsdaten trainieren, die von Yfinance stammen.
3. Vorhersage der Testdaten mit dem Linearen Regressionsmodell.
4. inverse Transformation wird auf die vorhergesagten Preise angewendet.
5. Nutzung der trainierten Parameter, um eine Vorhersage für die abhängige Variable zu treffen.
6. Nachdem die Vorhersage gemacht wurde, wird die inverse Transformation auf die vorhergesagten Werte angewendet.

Ergebnisse werden als Plot dargestellt:

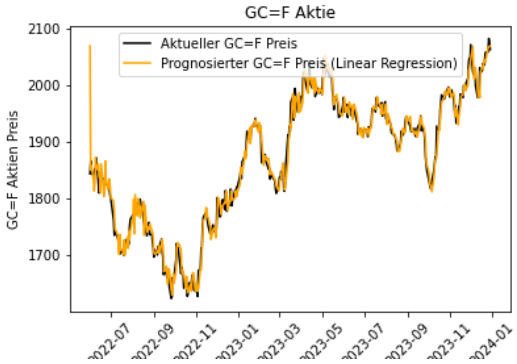
Als Prognose für den nächsten Tag gilt für

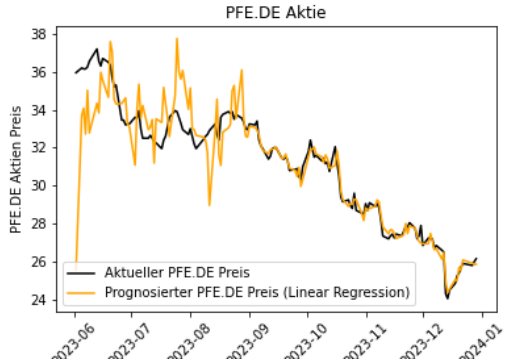
Gold = 1980.01 Silber = 23.239





Gold von 2022-2024 = 2068.761 Pfizer = 25.558





Abbildungen 13-16: Eigene Darstellung von Aktien durch die Plot Funktion von Spyder

Tatsächliche Werte: Gold = 2064.40 Silber =23.73 Pfizer=29.73

Schlussfolgerung

Am Beginn der Vorhersagen schwankt das Model sehr stark und hat sehr starke Ausreißer. Erst in der Mitte beginnt es sich einzupegeln und beginnt gute, genaue Daten zu liefern. Aber auch bei diesem Model arbeitet es mit mehr Eingabe Daten besser. Die Ergebnisse sind jedoch ebenfalls für den Beginn zufriedenstellend.

5.4 Support Vector Machines (SVM)

Der Support Vector Machines Algorithmus klassifiziert Datenobjekte. Sie werden oft in Text- oder Bildklassifizierung benutzt. Es gibt drei Arten von SVM's:

1. Lineare SVM's (nur bei Lineare Zusammenhänge)
2. Nichtlineare SVM's (komplexere Ebenen, die nicht linear sind)
3. Support Vector Regression (für Vorhersagung für Lineare und nicht lineare Beziehung der Daten.

In unserem Fall verwenden wir lineare Support Vector Regression, da der Kernel auf 'Linear' festgelegt ist.

Code Ausschnitt:

“

1. *svr = SVR(kernel='linear')*
2. *svr.fit(x_train, y_train)*

...

3. *real_data = model_inputs[-prediction_days:].reshape(1, -1)*
4. *prediction = svr.predict(real_data)*
5. *prediction = scaler.inverse_transform(prediction.reshape(-1, 1))*

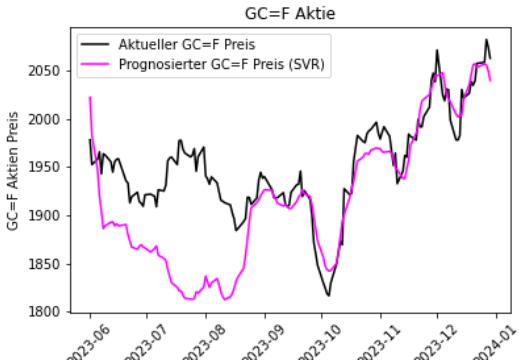
“

1. Initialisierung der klasse SVR mit der Kernel-Funktion ‘linear’.
2. Die ‘fit’ Methode um das Modell mit den Trainingsdaten zu Trainieren.
3. Die letzten 60 Tage (‘prediction_days=60’) der vorbereiteten Daten werden genommen und sie in der Variable #real_data’ transformiert.
4. Die Methode für die Vorhersage basierend auf den vorbereiteten Daten für den nächsten Tag.
5. Daten auf den ursprünglichen Datenbereich zurückwandeln.

Ergebnisse werden als Plot dargestellt:

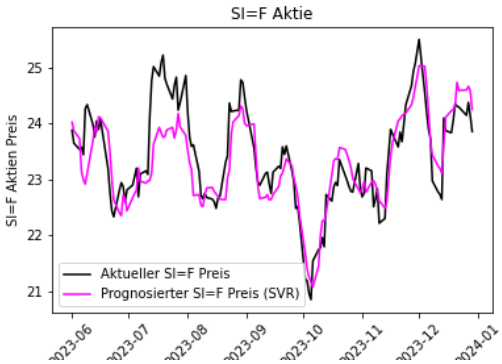
Als Prognose für den nächsten Tag gilt für

Gold= 2021.82

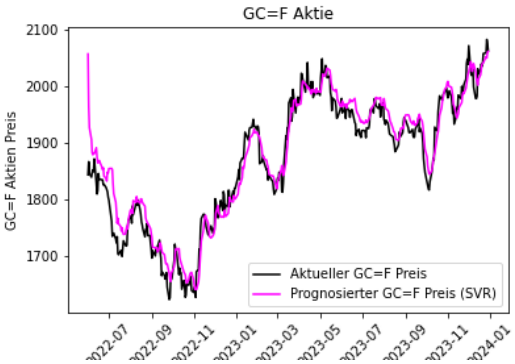


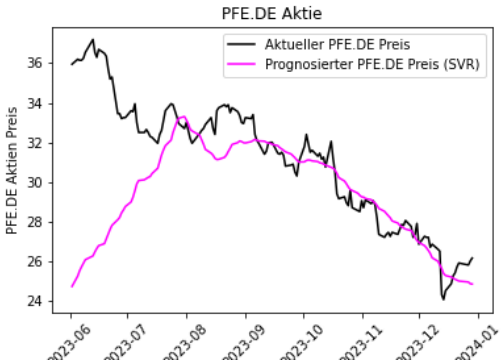
Gold von 2022-2024 = 2056.289

Silber = 24.02



Pfizer = 24.717





Abbildungen 17-20: Eigene Darstellung von Aktien durch die Plot Funktion von Spyder

Tatsächliche Werte: Gold = 2064.40 Silber =23.73 Pfizer=29.73

Schlussfolgerung

Auch bei diesem Modell sieht man anfangs ungenaue Vorhersagen. Am Ende wird es wieder genauer. Die Goldaktie mit einer höheren Laufzeit bietet wieder ein besseres Ergebnis.

5.5 Random Forests

Der Random Forests Algorithmus lässt sich ebenfalls für Klassifizierungsaufgaben und Regressionsaufgaben nutzen. Im unseren Fall brauchen für natürlich den Algorithmus für die Regressionsaufgabe.
Der Algorithmus arbeitet mit Entscheidungsbäumen.

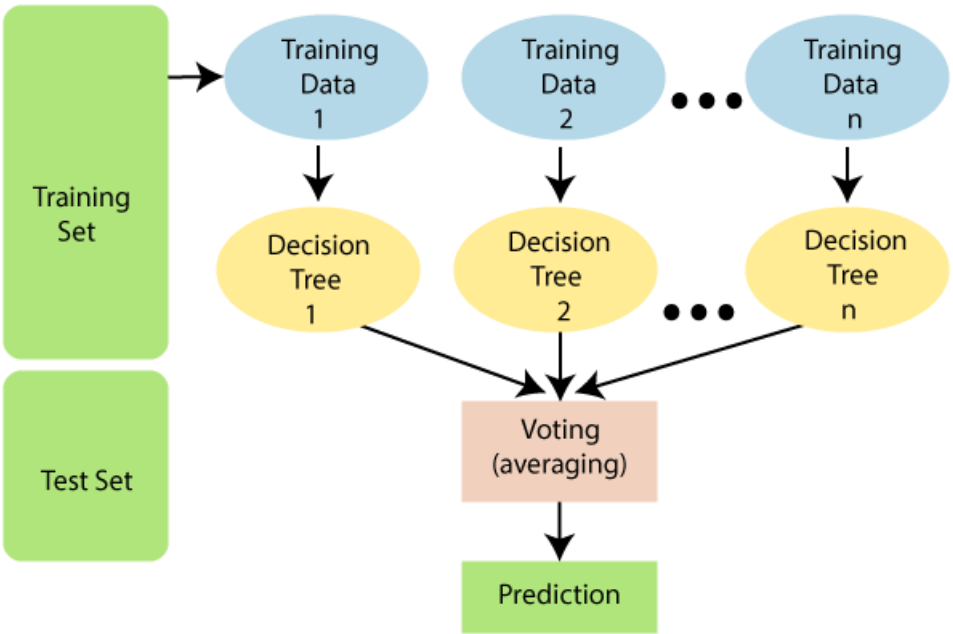


Abbildung 21: Darstellung Random Forest Algorithmus
(https://www.javatpoint.com/machine-learning-random-forest-algorithm)

Der Random-Forest-Algorithmus nutzt mehrere Entscheidungsbäume, wobei jeder innere Knoten eine Entscheidung repräsentiert und die Blätter die Endpunkte des Baums darstellen, die das Ergebnis repräsentieren. Diese Bäume werden auf zufälligen Teilen der Trainingsdaten trainiert. Das Verfahren des Bagging wird verwendet, um zufällige Untergruppen der Trainingsdaten mit Wiederholung auszuwählen und jeden Baum zu trainieren. Um eine Vorhersage zu treffen, lässt der Random-Forest-Algorithmus jeden Baum im Ensemble eine Vorhersage treffen. Bei Regressionsaufgaben wird der Durchschnitt der Vorhersagen der Bäume genommen. Das bedeutet das die Durchschnittsbildung der Vorhersagen Aller Bäume die Vorhersage entspricht.
Der Random Forest Algorithmus kann in der Praxis sehr leicht mit Python umgesetzt werden.

Code Ausschnitt:

“

1. *rf_reg = RandomForestRegressor(n_estimators=100, random_state=42)*
2. *rf_reg.fit(x_train, y_train)*

...

3. *real_data = model_inputs[-prediction_days:].reshape(1, -1)*
4. *prediction = rf_reg.predict(real_data)*
5. *prediction = scaler.inverse_transform(prediction.reshape(-1, 1))*

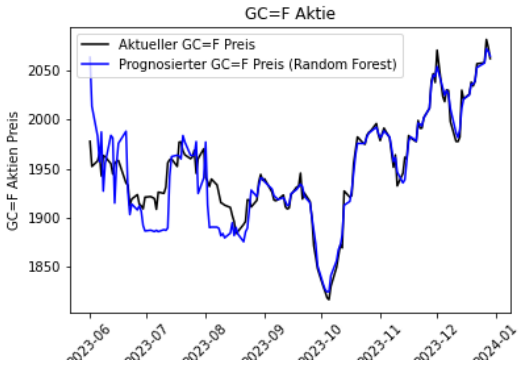
“

1. Erstellung eines Random-Forest-Regressor. Klasse für Regressionsaufgaben. ‘n_estimators=100’ gibt an wie viele Entscheidungsbäume erstellt werden sollen. ‘random_state=42’ wird benutzt, um dieselben Ergebnisse bei jeder Ausführung zu gewährleisten. Wichtig für den Vergleich für unsere verschiedene Modelle
2. Durch den Aufruf von ‘**fit()**’ werden die internen Parameter des Modells entsprechend den Trainingsdaten angepasst, sodass das Modell in der Lage ist, zukünftige Daten vorherzusagen, die ähnliche Muster wie die Trainingsdaten aufweisen.
3. durch die Methode wird sichergestellt das die richtigen Eingabedaten für die Vorhersage des nächsten Tages verwendet wird.
4. Die Methode für die Vorhersage basierend auf den vorbereiteten Daten für den nächsten Tag.
5. Daten auf den ursprünglichen Datenbereich zurückwandeln.

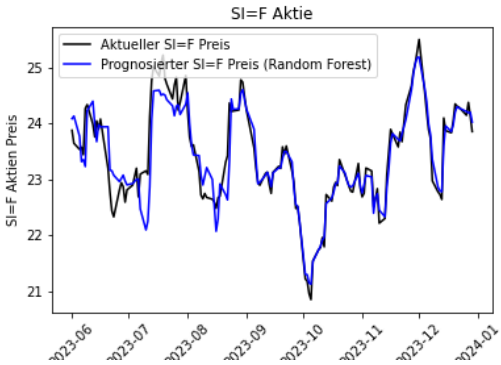
Ergebnisse werden als Plot dargestellt:

Als Prognose für den nächsten Tag gilt für

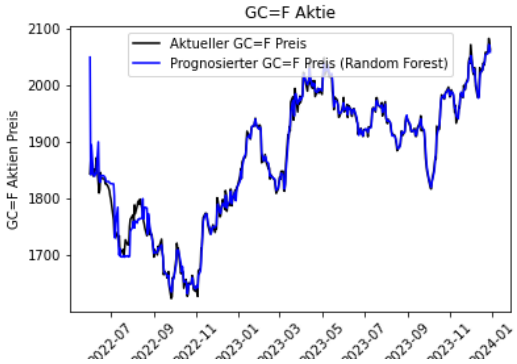
Gold = 2063.6489



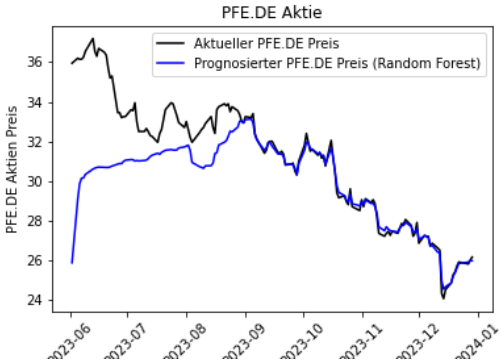
Silber = 24.08



Gold von 2022-2024 = 2048.94



Pfizer = 25.86



Abbildungen 22-25: Eigene Darstellung von Aktien durch die Plot Funktion von Spyder

Tatsächliche Werte: Gold = 2064.40 Silber =23.73 Pfizer=29.73

Schlussfolgerung

Der Random Forest Algorithmus sieht in dieser Konstellation sehr gut aus. Die Vorhersagen sind im Beginn teilweiße sehr ungenau aber zum Ende hin zeigt das Modell seine Stärke. Es ist sehr präzise und liefert sehr gute Ergebnisse. Im Gegensatz zu den bisherigen Modellen schneidet es bei dem längeren Zeitraum der Goldaktie schlechter ab. Und allgemein sind die Tagesvorhersagen genauer.

5.6 Gradient Boosting Models (GBM)

Beim Gradient Boosting-Modell werden auch Entscheidungsbäume verwendet, jedoch werden sie im Gegensatz zum Random Forest-Algorithmus einzeln und sequenziell trainiert. Jedes neue Modell zielt darauf ab, die Fehler des vorherigen Modells zu korrigieren, wodurch eine schrittweise Verbesserung der Vorhersagequalität erreicht wird.

Die Vorhersage eines Gradient Boosting-Modells erfolgt durch die Summierung der Vorhersagen aller Bäume im Ensemble. Dabei wird jedem Baum basierend auf seiner Leistung ein bestimmtes Gewicht zugewiesen. Dies ermöglicht es dem Modell, sich stärker auf die Vorhersagen der besser performenden Bäume zu verlassen und so präzisere Ergebnisse zu erzielen.

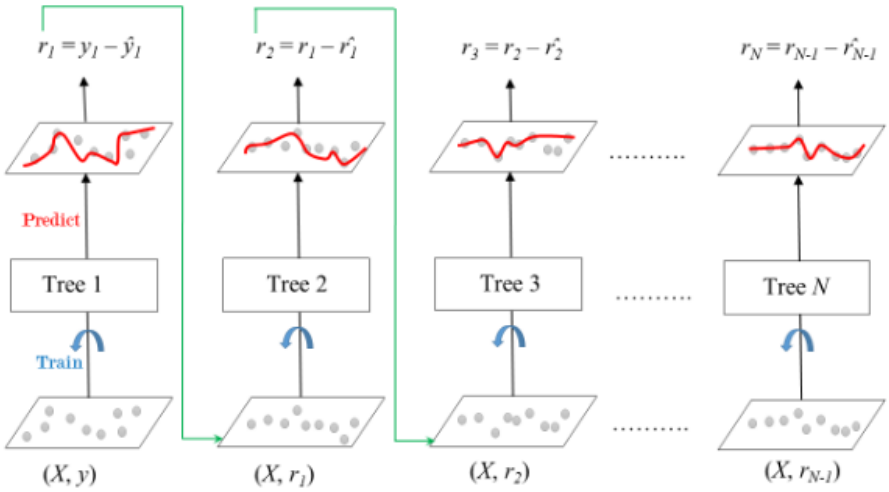


Abbildung 26: Darstellung Gradient Boosting Modell (https://www.geeksforgeeks.org/ml-gradient-boosting/)

Code Ausschnitt:

“

...

1. gbr = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, max_depth=1, random_state=0, loss='squared_error')

2. gbr.fit(x_train, y_train)

...

“

1. Objekterstellung mit 100 Bäumen. Learning_rate steuert die Beitragsrate jeden einzelnen Baumes. Max=depth regelt, ob ein Baum eher flacher oder tiefer ist. (Overfitting wird darüber reguliert). Random_state wie beim Random Forest. Die ‘loss’ funktion misst die Abweichung zwischen den tatsächlichen und den vorhergesagten Werten und dient als Maß dafür, wie gut das Modell die Daten passt.

Objekterstellung mit 100 Bäumen: Durch die Angabe von **n_estimators=100** wird ein Gradient-Boosting-Modell mit insgesamt 100 Bäumen erstellt. Jeder Baum wird während des Trainingsprozesses schrittweise hinzugefügt, um die Vorhersagegenauigkeit des Modells zu verbessern.

Learning_rate steuert die Beitragsrate jeden einzelnen Baumes: Die **learning_rate** ist ein Hyperparameter, der die Lernrate des Gradient Boosting-Algorithmus steuert. Sie bestimmt, wie stark die Vorhersagen jedes einzelnen Baumes das finale Modell

beeinflusst. Eine niedrigere Lernrate bedeutet eine langsamere Anpassung des Modells und kann dazu beitragen, Overfitting zu vermeiden, während eine höhere Lernrate eine schnellere Konvergenz ermöglicht, aber auch zu Overfitting führen kann.

Max_depth regelt, ob ein Baum eher flacher oder tiefer ist (Overfitting wird darüber reguliert): Der Parameter **max_depth** bestimmt die maximale Tiefe jedes einzelnen Entscheidungsbaums im Gradient-Boosting-Modell. Eine geringere Tiefe führt zu flacheren Bäumen, die weniger anfällig für Overfitting sind, da sie weniger spezifische Muster in den Trainingsdaten erfassen können. Eine höhere Tiefe hingegen ermöglicht es den Bäumen, komplexere Muster zu lernen, was jedoch das Risiko von Overfitting erhöht.

Random_state wie beim Random Forest: Der **random_state**-Parameter stellt sicher, dass die Ergebnisse des Gradient Boosting-Modells reproduzierbar sind. Indem ein bestimmter Wert für **random_state** festgelegt wird, wird der Zufallszahlengenerator initialisiert, was bedeutet, dass das Modell bei jeder Ausführung die gleichen Ergebnisse liefert.

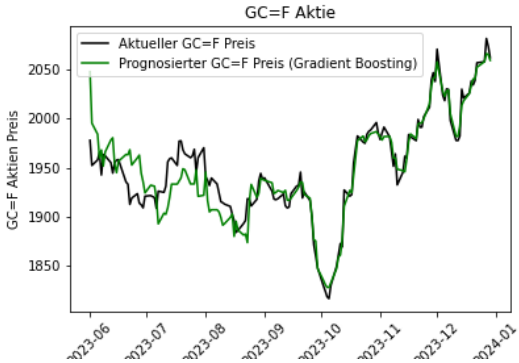
Die 'loss'-Funktion misst die Abweichung zwischen den tatsächlichen und den vorhergesagten Werten und dient als Maß dafür, wie gut das Modell die Daten passt: Die **loss**-Funktion definiert die Art der Fehlermetrik, die während des Trainings des Gradient Boosting-Modells minimiert wird. Die Wahl der Verlustfunktion hängt von der Art des Problems ab. Im vorliegenden Fall wird die quadratische Fehlerfunktion ('squared_error') verwendet, um die Abweichung zwischen den tatsächlichen und den vorhergesagten Werten zu messen.

2. Die ‘fit’ Methode um das Modell mit den Trainingsdaten zu Trainieren.

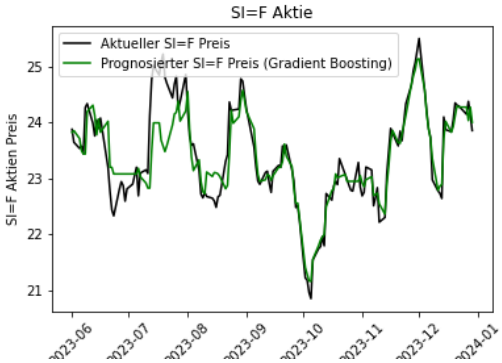
Ergebnisse werden als Plot dargestellt:

Als Prognose für den nächsten Tag gilt für

Gold = 2047.989

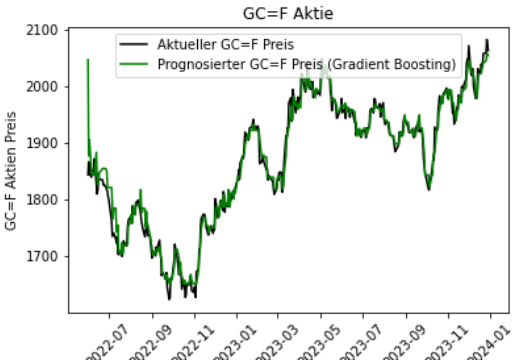


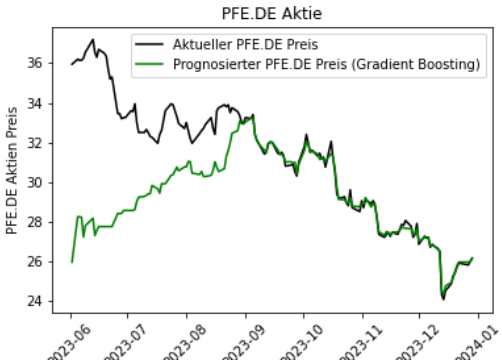
Silber = 23.82



Gold von 2022-2024 = 2045.898

Pfizer = 25.949





Abbildungen 27-30: Eigene Darstellung von Aktien durch die Plot Funktion von Spyder

Tatsächliche Werte: Gold = 2064.40 Silber =23.73 Pfizer=29.73

Schlussfolgerung

Der GradientBoostingModel Algorithmus schneidet etwas schlechter ab als die anderen. Nur bei Silber wurde sehr gut vorhergesehen. Der Algorithmus funktioniert trotzdem, jedoch müssen einige Verbesserungen vorgenommen werden.

6. Diskussion der Ergebnisse

Die Evaluierung verschiedener Algorithmen zeigt, dass jedes Modell seine eigenen Vor- und Nachteile hat. Die erzielten Ergebnisse sind vergleichbar, wobei deutlich wird, dass die Algorithmen grundlegend funktionieren, jedoch noch Anpassungen erforderlich sind, um ihre Leistungsfähigkeit zu optimieren. Sie dienen als solide Ausgangspunkte, auf denen aufgebaut werden kann. Unterschiedliche Anpassungen am Code beeinflussen die Algorithmen auf unterschiedliche Weise. Ein wesentlicher Aspekt ist die Einstellung der "prediction_days", die je nach Bedarf angepasst werden kann. Darüber hinaus kann die Anzahl der Bäume oder Neuronen (abhängig vom Algorithmus) variiert werden, um die Vorhersagegenauigkeit zu verbessern.

Es ist zu beachten, dass die Modelle zu Beginn aufgrund fehlender Daten schwach sind, aber mit zunehmender Datenmenge an Stärke gewinnen. Unser Hauptaugenmerk liegt darauf, zukünftige Vorhersagen präzise zu gestalten, was eine ausreichende Datenmenge erfordert. In dieser Hinsicht hat das Random Forest Model die besten Ergebnisse erzielt, daher könnte es sich lohnen, sich weiterhin darauf zu konzentrieren und es zu verfeinern.

Trotzdem war das Hauptziel dieser Arbeit, Ihnen die Grundlagen der maschinellen Lernalgorithmen näherzubringen, und ich hoffe, dass dies gelungen ist. Der Fokus lag darauf, die Funktionsweise dieser Algorithmen zu verdeutlichen.

7. Zusammenfassung

Die Algorithmen stellen eine solide Grundlage dar, um Aktienvorhersagen zu verbessern. Jedes Modell erfordert individuelle Anpassungen, daher gibt es keine pauschale Aussage darüber, welches besser oder schlechter ist. Es ist wichtig zu verstehen, dass jedes Modell seine Vor- und Nachteile hat. Diese Modelle eignen sich gut für den Einsatz als sogenannter "Day Trader", jedoch weniger für langfristige Investitionen.

Es ist von entscheidender Bedeutung zu erkennen, dass Aktienkurse niemals zu 100% vorhergesagt werden können. Sie sind durch eine Vielzahl von Faktoren wie globale Ereignisse, Krisen, interne Unternehmensprobleme und viele andere Faktoren unberechenbar. Die Modelle dienen lediglich als Entscheidungshilfe und sollten nicht blind übernommen werden. Diese Arbeit konzentriert sich hauptsächlich auf die Algorithmen und nicht auf den direkten Kauf von Aktien. Es ist wichtig, diese Modelle mit Bedacht und einem Verständnis für ihre Grenzen einzusetzen.

8. Anhang

LongShortTimeMemory Algorithmus

GatedRecurrentUnit Algorithmus

LineareRegression Algorithmus

GradientBoostingmodel

RandomForests

supportVectorMachines

Quellcode nicht benutzter Scripts

Diagramme und Grafiken

Plot Bilder

Quellenverzeichnis

Welcker, J. (1994): Technische Aktienanalyse. 7. Auflage, Zürich.

Andreas Sharik(2018), Fundamentalanalyse. In Gabler Wirtschaftslexikon. Abgerufen von (https://wirtschaftslexikon.gabler.de/definition/fundamentalanalyse-58126/version-344971)

The World Bank.(2020). Listed domestic companies, total. Verfügbar unter: https://data.worldbank.org/indicator/CM.MKT.LDOM.NO (Abrufdatum: 1. Februar 2024)

Francois Chollet(2018), Deep Learning mit Python und Keras, MITP-Verlags GmbH & Co. KG

Daniel Handloser(2017), Prädiktion von Aktienkursen mit Neuronalen Netzen, KIT – Die Forschungsuniversität in der Helmholtz-Gemeinschaft

Abbildung 1: Eigene Darstellung basierend auf Daten von yFinance und der Plot Funktion von Spyder (25.01.2024)

Abbildung 2: selbst erstellte Darstellung von Word, Teilbereiche von KI (01.02.2024)

Abbildung 3: Long Short Time Memory (https://d2l.ai/chapter_recurrent-modern/lstm.html)

Abbildungen 4-7: Eigene Darstellung von Aktien durch die Plot Funktion von Spyder
Abbildungen Abbildung 8: Darstellung Gated Recurrent Unit (https://www.researchgate.net/figure/Gated-Recurrent-Unit-GRU_fig4_328462205)

Abbildungen 9-12: Eigene Darstellung von Aktien durch die Plot Funktion von Spyder

Abbildungen 13-16: Eigene Darstellung von Aktien durch die Plot Funktion von Spyder

Abbildungen 17-20: Eigene Darstellung von Aktien durch die Plot Funktion von Spyder

Abbildung 21: Darstellung Random Forest Algorithmus (https://www.javatpoint.com/machine-learning-random-forest-algorithm)

Abbildungen 22-25: Eigene Darstellung von Aktien durch die Plot Funktion von Spyder

Abbildung 26: Darstellung Gradient Boosting Modell (https://www.geeksforgeeks.org/ml-gradient-boosting/)

Abbildungen 27-30: Eigene Darstellung von Aktien durch die Plot Funktion von Spyder