Duale Hochschule Baden-Württemberg Mannheim

Data Exploration

Vorhersage von Aktienindizes

mithilfe latenter Weltereignis-Repräsentationen

Wirtschaftsinformatik

Data Science

Autoren:
Jan Henrik Bertrand (8556462)
David Hoffmann (2571020)
Marc Grün (9603221)
Felix Noll (9467152)

Studiengang:
WWI21DSB

GitHub Repository

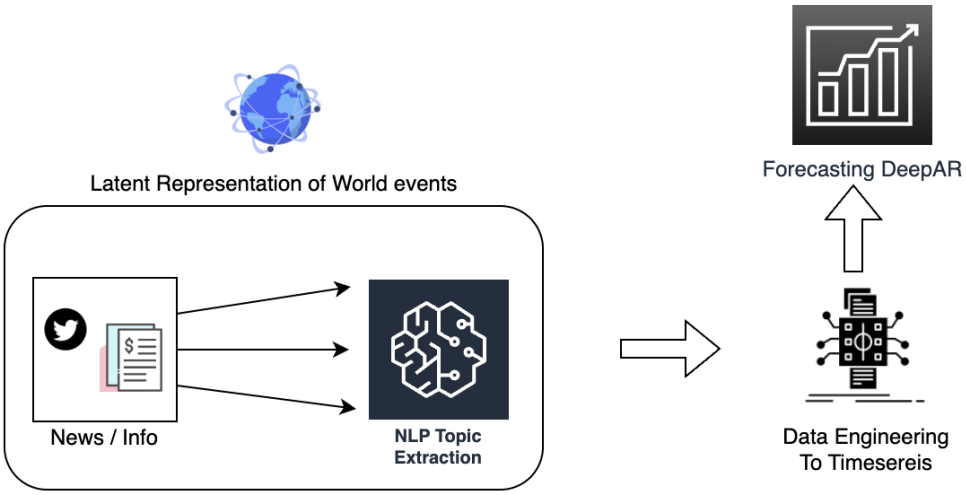
Inhaltsverzeichnis

Abkürzungsverzeichnis

AWS	Amazon Web Services
RTS	Related Time Series
FED	Federal Reserve
LDA	Latent Dirichent Allocation
MAPE	Mean Absolute Percentage Error
MASE	Mean Absolute Scaled Error
WQL	Weight Quantile Loss
MQ-CNN	Multi-Horizon Quantile Convolutional Neural Network
RMSE	Root Mean Squared Error
ETF	Exchange Traded Fund

1. Einleitung

Die initiale Projektidee war es, eine niedrigdimensionale Repräsentation des Weltgeschehens als Datengrundlage zu nutzen, um die Entwicklung der Finanzmärkte vorherzusagen. Es existiert bereits Literatur, in der ähnliche Ansätze zu Aktienvorhersagen verfolgt werden [6] [5] [3] [11]. Unsere Ausarbeitung unterscheidet sich in Methodik und Datengrundlage.



Als Datengrundlage haben wir Zeitungsartikel und Schlagzeilen verwendet, aus denen mithilfe von Natural Language Processing, Themen extrahiert werden sollten. Nach Datum aggregiert, konnten die Artikel zu Weltgeschehnissen anschließend als Zeitfolge modelliert werden. Der Hintergedanke ist, dass man später anhand von den Schlagzeilen und Themen die Bewegung des Aktienmarkts voraussagen kann und so entweder Aktien kauft, bevor diese steigen, oder eine Aktie shorted, bevor sie fällt.
Während der Implementierung, wurde relativ schnell klar, dass das NLP processing mehrere Herausforderungen mit sich bringt. Ein Beispiel hierfür ist der zusätzliche

Übersetzungsaufwand, welcher dadurch entsteht, dass Zeitungen in ihrer Landessprache geschrieben sind. Dies stellt aufgrund der nötigen Menge an Schlagzeilen, um für einen Zeitraum von mehreren Jahrzehnten eine zuverlässige Repräsentation von Ereignissen zu erhalten, eine besondere Herausforderung hinsichtlich des Ressourcenaufwands dar. Sich auf die englischsprachigen Länder zu konzentrieren, war auch keine Option, da diese das Weltgeschehen nicht adäquat widerspiegeln.
Auf der Suche nach einer effektiven Lösung sind wir auf das GDELT Projekt gestoßen, welches News weltweit abgreift, ins Englische übersetzt, das Thema extrahiert, Akteure erkennt und eine Sentiment-Angabe enthält. Da dies all unsere Anforderungen erfüllt, haben wir GDELT im Folgenden genutzt. Dies brachte den weiteren Vorteil, dass GDELT die verschiedenen Nachrichten aggregiert. Wenn eine Nachricht also von mehreren Seiten gemeldet wird, wird diese nicht als zweite Zeile angesetzt, sondern in dem Attribut „NumMentions“ hochgezählt. Auf diese Art und Weise war es einfach, die relevantesten Themen herauszufiltern.

2. Ablauf

In diesen Kapitel wird der Ablauf des Projekt beschrieben, von den initialen Gedanken, bis zum Endergebnis.

2.1. Vorgehen

Werkzeuge der Datensammlung

Um die Daten aus dem GDELT Projekt zu extrahieren, wurde einerseits ein einfacher Download, ein Webcrawler, aber auch Google BigQuery genutzt. Dies war nötig, da alle Daten vor dem 01.01.2014 nur als zusammengefasste Masterfile als Download verfügbar waren. Zwischen dem 01.01.2014 und 31.12.2015 waren die Daten in einzelnen ZIP-Files zum Download auf der GDELT-Website vorhanden. Die Events ab dem 01.01.2016 bis heute sind in der Google Cloud Platform erhältlich und können mit dem Google Big Query Tool durch SQL-Queries abgefragt werden. Da die einzelnen Events in CAMEO-Code kodiert waren, wurden diese mit Hilfe eines selbst geschriebenen Jupyter Notebooks entschlüsselt (cameo_translation.ipynb in Github). Nachdem die GDELT Daten mit den diversen Methoden heruntergeladen wurden, wurden diese transformiert und angepasst. Nach Transformation der Daten wurde festgestellt, dass die Menge an Datenpunkten pro Jahr zunahm. Daher wurden die Anzahl an Events pro Jahr auf 5000 vereinheitlicht, um die Repräsentativität des Datensatzes zu wahren. Dabei wurden die 5000 häufigsten NumMentions pro Jahr verwendet. Diese Transformationsschritte wurden hauptsächlich mit Alteryx durchgeführt. Alteryx ist ein Tool, um Datensätze zu modifizieren und transformieren. Jedoch kamen auch Pythonmodule in der Vorverarbeitung, im Modelling und in der Evaluation zum Einsatz, welche im Folgenden aufgelistet sind:

- pandas
- numpy
- logging
- argparse
- html
- string
- datetime
- os
- xml.etree
- ElementTree
- zipfile
- requests
- json
- gluonts
- sqlalchemy
- src

Zusätzlich zu den GDELT Daten wurden noch weitere Finanzdaten benötigt, um eine fundierte Basis für das Modell zu schaffen. Dies sollte es dem Modell ermöglichen, makroökonimische Zusammenhänge, wie beispielsweise Konjunkturzyklen, zu extrahieren. Dazu wurden Indizes und ETF[1]s herausgesucht, welche in Kombination, wichtige Bereiche der Wirtschaft abdecken. Dadurch sollten ökonomische Zusammenhänge erlernt werden. Es wurde sich entschieden, folgende sieben ETFs und Indices zu verwenden: NASDAQ 100, MSCI World, MSCI Emerging Market, DAX, ETF IT, ETF Öl und ETF Agrikultur. Zusätzlich dazu wurden noch weitere wirtschaftliche Kennzahlen gesammelt, welche Auswirkungen auf den Aktienmarkt haben könnten. Diese waren: Inflation pro Land (weltweit), Leitzins der FED[2], Geldmenge, BIP (weltweit) und Arbeitslosenquote (G7+ weitere).
Da wir erst später auf das GDELT-Projekt gestoßen sind, haben wir zunächst einen Datensatz genutzt, welcher CNBC Nachrichten beinhaltet [4]. Dieser wurde dann für den NLP Prozess vorbereitet. Dies geschah, indem als erstes die HTML-Nummer-Codes mit Unicode Buchstaben ersetzt, jegliche Satzzeichen entfernt und alles in Kleinbuchstaben umgewandelt wurde. Im Anschluss daran, wurden die Wörter Tokenized und Füllwörter entfernt. Am Ende wurde jedes Wort auf den Wortstamm reduziert. Jede Schlagzeile wurde dann mittels Latent Dirichlet Allocation einem Themencluster hinzugefügt. Im Folgenden wurden diese Themen dann zu einer Menge an Time Series in wöchentlicher Frequenz aggregiert.

Für eine größere und repräsentativere Datengrundlage sollten zusätzlich, zu dem CNBC Datensatz, noch weitere News von der ganzen Welt verwendet werden. Dies wäre sehr aufwändig gewesen, und wurde nicht weiter verfolgt, vor allem da der GDELT Datensatz entdeckt wurde, welcher diese News bereits beinhaltet.

[1] Exchange Traded Fund
[2] Federal Reserve

Modelling

Nachdem die Daten gesammelt und bereinigt wurden, begann der Modelling Prozess. Dem Prinzip von Occam's Razor folgend, haben wir mit der Implementierung einfacher Methoden angefangen, um uns dann sukzessive zu komplexeren Methoden vorzuarbeiten. Dieser Fortschritt (Ergebnisse und Konfiguration) wurde in einer relationalen Datenbank gespeichert.
Bevor wir auf das GDELT Projekt gestoßen sind, haben wir den CNBC news Datensatz verwendet, um die Relevanz von Ereignissen mit globaler Auswirkung über die Zeit zu verfolgen. Dazu mussten die enthaltenen Schlagzeilen jedoch zuerst Themen zugeordnet werden. Zu dieser Clusteringaufgabe wurden drei Machine Learning Algorithmen in Betracht gezogen: LDA[1], Top2Vec und BERTopic. LDA ist eine Methode, die verwendet wird, um Themen aus Texten zu extrahieren. Sie analysiert die Verteilung von Wörtern in den Dokumenten, um die Wahrscheinlichkeit zu berechnen, dass ein bestimmtes Thema in einem Dokument vorkommt. Diese Technik findet häufig Anwendung im Bereich der Textklassifikation [7]. Des Weiteren wurde Top2Vec als komplexere Alternative erwogen. Hierbei handelt es sich um eine Methode des unüberwachten Lernens, welche zunächst eine niedrig dimensionale Vektorrepräsentation der Dokumente des Korpus lernt, um dann die Texte dann in Cluster zu unterteilen. Der Vorteil von Top2Vec ist zum einen die Einbeziehung von Kontext, sowie die dynamische Erkennung von Themengebieten [2]. Die dritte Methode, die wir in Betracht gezogen haben, ist BERTopic. Dabei handelt es sich um ein vortrainiertes Modell, welches die Daten zunächst mit Hilfe von BERT codiert, und anschließend in Gruppen (Cluster) unterteilt [8].

[1] Latent Dirichent Allocation

Name	Erklärung
Index	Zahl in aufsteigender Reihenfolge
published_at	Datum der Veröffentlichung DD.MM.YYYY
topic_class	Tupel mit den Thema und dem Score. Tupel ist ein Vektor in einem latenten Vektorraum, welcher die Vorgänge und Ereignisse der Welt darstellt

Tabelle 2.1.: Outputformat

Für das Forcasting haben wir zunächst DeepAR gewählt. Hauptgrund dafür war, dass DeepAR sowohl gut mit vielen verschiedenen Timeseries umgehen kann, was bei uns von essentieller Bedeutung ist, als auch gute Resultate auf nicht vollständigen Daten erreicht, was aufgrund der dünnbesiedelten GDELT- und Finanzdaten von Vorteil ist [10]. Zur Modellierung mit DeepAR haben wir die Implementierung des Open-Source Projekts „GluonTS“ [1] verwendet. Die ersten Experimente mit DeepAR haben die, aus dem CNBC News Datensatz extrahierten, Themen-Zeitreihen zur Modellierung verwendet. Aufgrund der steigenden Komplexität durch das Hinzufügen des GDELT Datensatzes (über 400 RTS[1]), war es nicht mehr möglich, das Modell lokal zu trainieren. Im Folgenden haben wir uns dazu entschieden, die Modellierung in der AWS[2] Cloud fortzusetzen. Dort wurden Hyperparameter, wie die prediction_length, num_layers, hidden_size, context_lenght, learing rate und weight_decay optimiert. Diese wurden jedoch nicht mit automatischer Hyperparameter Optimization gelernt, sondern manuell angepasst, da die derzeitige Trainigszeit bereits zwischen 2 und 5 Stunden pro Durchgang lag. Es wurde darauf geachtet die Hyperparamether isoliert anzupassen, um die Veränderung des Modells klar auf bestimmte Änderungen zurückführen zu können.

Um die Performance der Modelle vergleichbar zu machen, wurden für jedes Modell folgende sieben Metriken berechnet und gespeichert: MASE, MAPE, RMSE, wQL_10, wQL_50, wQL_90, avg_wQL (von jeder Iteration sind diese Werte in Anhang.csv zu finden). Nach einigen Versuchen stellte sich heraus, dass eine context_length von 200, sowie ein kleines Netzwerk mit zwei Schichten und 40-60 nodes optimal zu sein scheint.

[1] Related Time Series
[2] Amazon Web Services

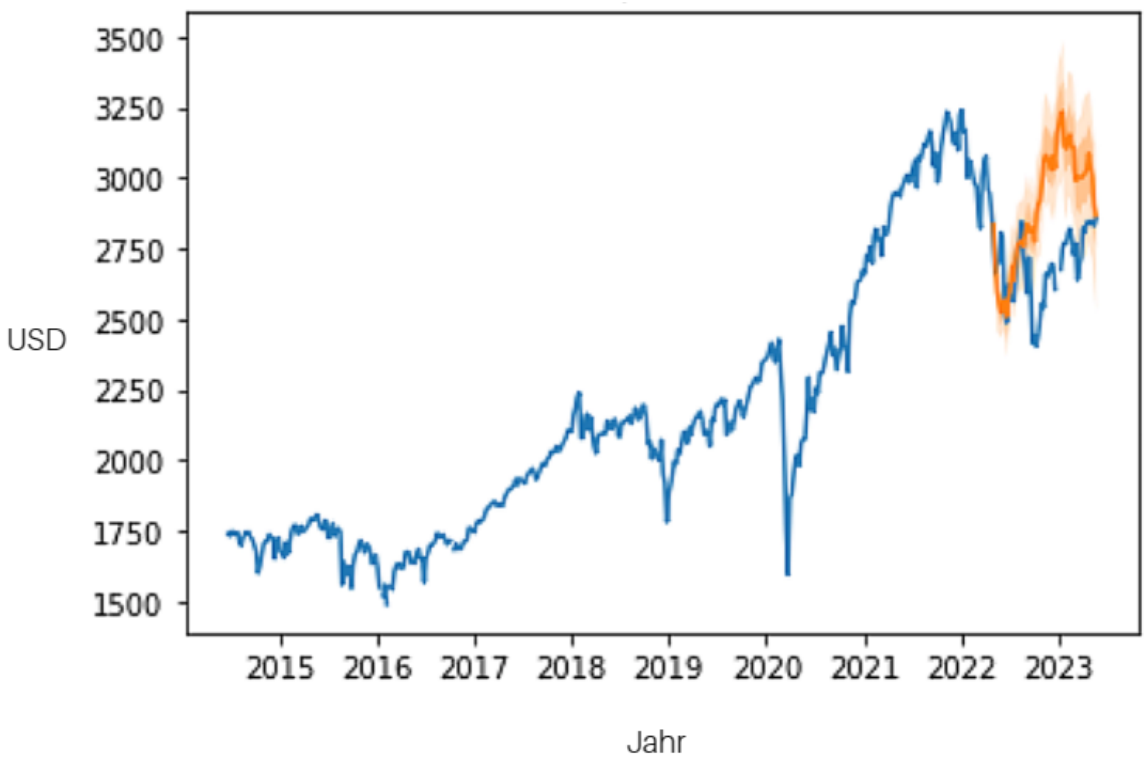


Abbildung 2.1.: DeepState Forecast

Dort war der MAPE bei circa 0.04398. Dies war unser Richtwert, um zu testen, ob andere Methoden besser sind.
Als nächstes wollten wir DeepAR mit DeepState vergleichen, um einen Referenzwert von einem anderen bekannten und häufig genutzten Modell zu bekommen. DeepState nutzt mittels Deep Learning erstellte State Models zur Approximation. State Models spiegeln bestimmende Zusammenhänge in einem System (in unserem Fall den Aktienmarkt) wieder [9]. Wir haben dabei die Hypothese verfolgt, dass dieses Modell besonders gut darin ist, Marktmechanismen aus den verfügbaren Daten zu erlernen. Die ersten Ergebnisse waren leider nicht vielversprechend.

Das Modell aus Abbildung 2.1 besitzt einen MAPE[1] von 0.111 und average WQL[2] von 0.082. Gut zu Erkennen ist, dass die Vorhersage Anfangs sehr akkurat ist, sich dann jedoch entgegengesetzt des tatsächlichen Kurses des S&P 500 entwickelt. Daher versuchten wir auch hier iterativ das Modell zu optimieren. Leider mussten wir nach mehreren Stunden Training feststellen, dass das Modell für 2022 immer noch einen positiven Trend vorhersagte. Daher passten wir die Lernrate massiv an. Dies half um

[1] Mean Absolute Percentage Error
[2] Weight Quantile Loss

zu einem Minimum auf dem Trainingsdataset zu erreichen. Trotzdem mussten wir feststellen, dass das Modell an Overfitting leidet und somit ungeeignet ist. DeepState hat einen MAPE von 0.104 und average WQL von 0.086 im Vergleich zu DeepAR mit einem MAPE von 0.061 und average WQL von 0.053.

Als zweiten Referenzpunkt, haben wir MQ-CNN[1] als Machine Learning Modell zur Vorhersage des Aktienmarkts verwendet. Zu Beginn nahm der training loss kontinuierlich ab. Jedoch stagnierte dieser irgendwann, obwohl für 2022 auch ein positiver Trend vorhergesagt wurde. Da der MAPE bei 0.279 lag, beschlossen wir uns weiter auf DeepAR zu fokussieren.

Im Folgenden haben wir uns darauf fokussiert, DeepAR weiter zu optimieren, was zu den folgenden optimalen Hyperparametern geführt hat.

Hyperparameter	**Wert**
prediction_length	50
epochs	50
num_layers	3
hidden_size	20
context_lenght	100
learing_rate	0.001
weight_decay	1e-15

Tabelle 2.2.: Optimale Hyperparameter (1 Testfenster)

Das resultierende Modell war in der Lage den Aktienmarkt der letzten beiden Jahre mit einem MAPE von 0.057 und average WQL von 0.048 vorherzusagen.

Um dieses Resultat zu bestätigen, haben wir einen Rolling Backtest über acht Jahre erstellt. Die Ergebnisse dieses Tests waren signifikant schlechter als jene mit einem einzigen Testfenster, was möglicherweise auf ein voriges Overfitting des Modells auf

[1] Multi-Horizon Quantile Convolutional Neural Network

die vorliegende Marktsituation zurückzuführen ist. Zur Verhinderung weiteren Overfittings, wurde die Modellkomplexität und die Anzahl der Lernepochen reduziert.

Hyperparameter	**Wert**
prediction_length	52
epochs	100
num_layers	3
hidden_size	20
context_length	200
learing_rate	0.001
weight_decay	1e-15

Tabelle 2.3.: Optimale Hyperparameter (6 Testfenster)

2.2. Ergebnis

Zunächst haben wir uns nur die Ergebnisse des Modells im Jahr 2022/2023 mit dem Kursverlauf des S&P 500 vergleichen. Hier war ein DeepAR Modell aus Abbildung 2.2 mit einem Testfenster am besten.

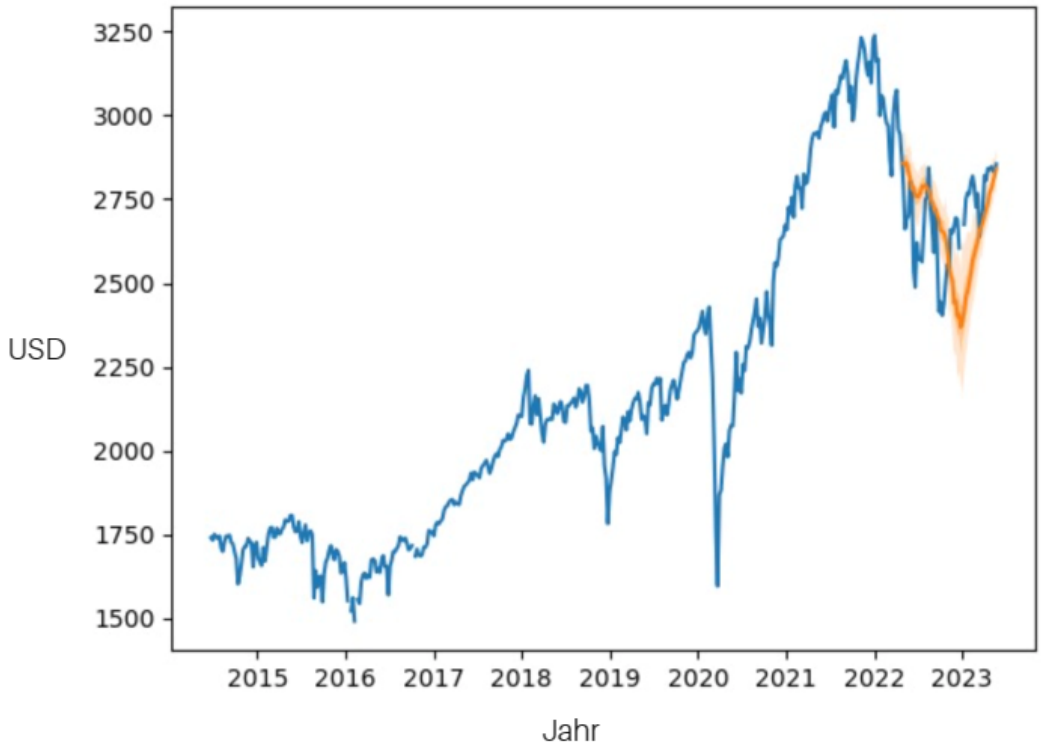


Abbildung 2.2.: DeepAR Modell mit Hyperparametern aus Tabelle 2.2. Average WQL: 0.028, MASE[1]: 8.81, MAPE: 0.041 und RMSE[2]: 196.7

Als finales Modell haben wir uns dann jedoch für ein DeepAR Modell mit angepassten Hyperparametern (2.3) und 6 Testfenstern entschieden, da dies zwar in den Metriken leicht schlechter ist, als das Modell mit einem Testfenster, jedoch generell in verschiedenen Marktsituationen besser vorhersagt und auch in unserem Anwendungsbeispiel (siehe 2.4) deutlich besser abschneidet.

Dieses Modell besitzt einen MASE von 14.38, MAPE von 0.066, RMSE von 339.19 und average WQL von 0.04.

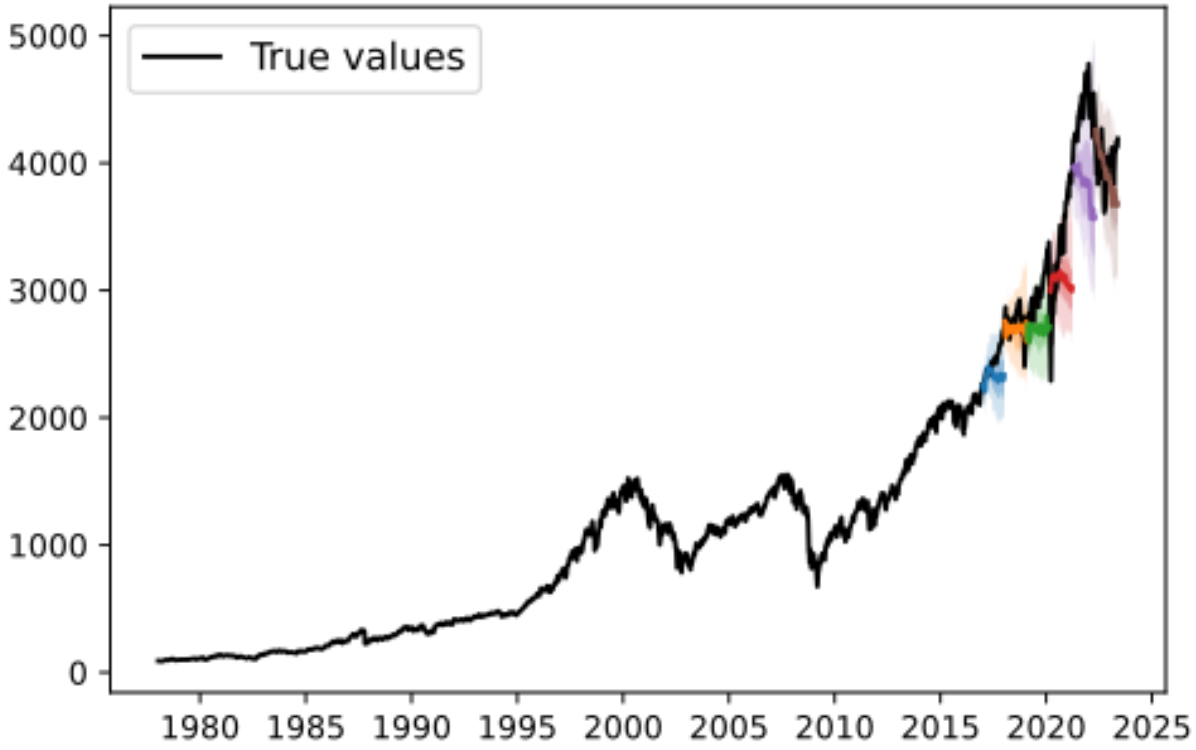


Abbildung 2.3.: Finale Modell Performance verglichen mit dem S&P 500 Index. Auf der x-Achse sind die Jahre und auf der y-Achse der Kurs des S&P 500 in US-Dollar.

Wenn man jedoch anhand des Modells versucht zu investieren, schneidet man verglichen zur Performance des S&P 500 selbst eher schlecht ab. Um zu validieren wie geeignet unser Modell zum investieren ist, haben wir uns folgendes Szenario ausgedacht: Man nehme 10.000€. Diese werden am Ende 2016 in den S&P 500 investiert. Jeden Monatsanfang, wird geschaut, wie der Aktienmarkt sich verhalten soll. Entweder kauft man mit dem gesamten verfügbaren Budget Short-Positionen (man wettet auf fallende Kurse), oder man investiert regulär, falls das Modell Kursgewinne vorhersagt. Zum Vergleich wird ein einmaliges anfängliches Investment von 10.000€ vorgenommen und am Ende der betrachteten 6 Jahre wieder verkauft. Verglichen wird also eine Anlagestrategie, die dem Modell vertraut mit einer, die anfangs einmalig investiert.

Zu beachten ist hier, dass für diesen Test die Modelle einmal trainiert wurden und jährlich eine Vorhersage treffen, während das Modell in Abbildung 2.1 auf allen verfügbaren Daten neu trainiert wurde.

Verfolgt man die Methode, welche dem Modell vertraut, hätte man am Ende einen Depotwert von 12.355€. Investiert man sein Geld und lässt es liegen, wäre der Depotwert bei 18.443€. Das heißt, auch wenn man Geld dazu gewinnt, gibt es bessere und einfachere Alternativen sein Geld zu investieren.

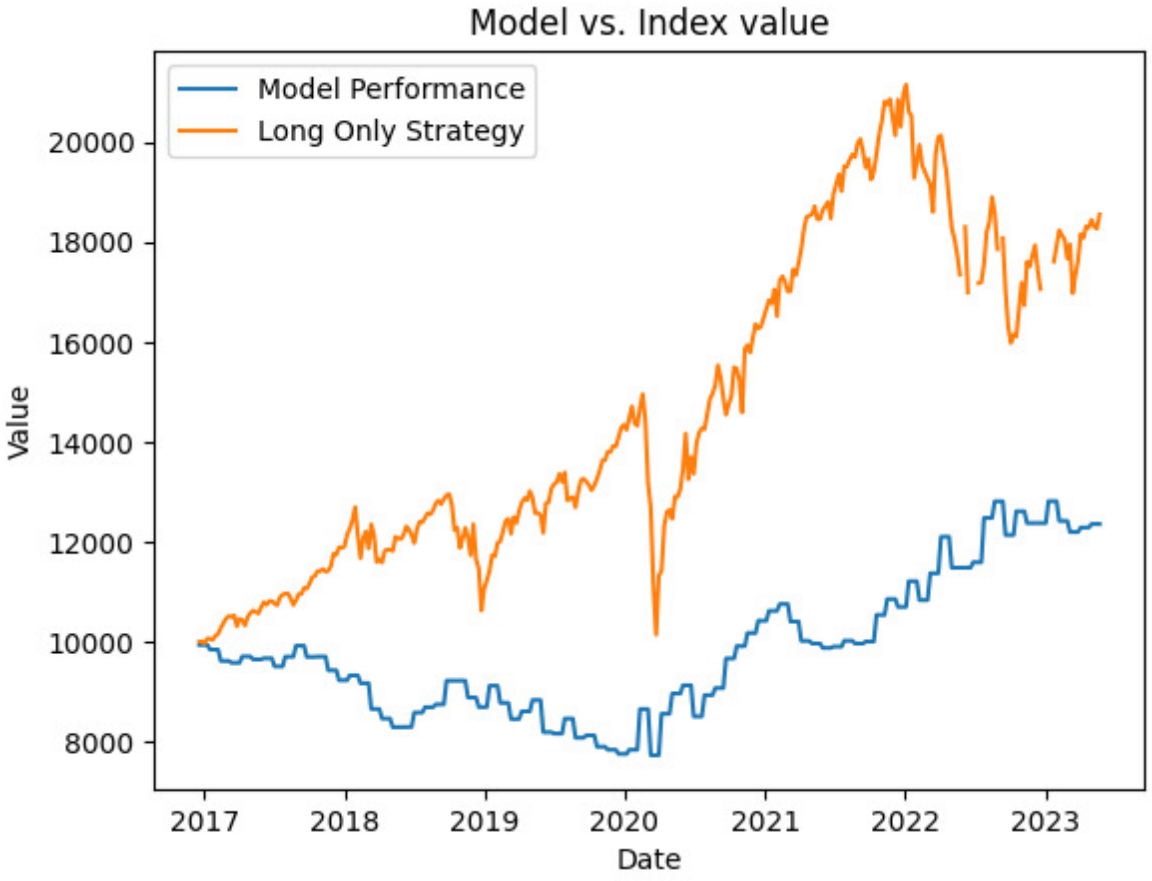


Abbildung 2.4.: Vergleich der Entwicklung eines 10.000€ Portfolios zwischen Modell-Anweisungen und stetiger Re-Investition.

Bei dem Vergleich zwischen einem Modell, welches auf GDELT-Daten trainiert wurde und einem, welches lediglich auf Finanzdaten trainiert wurde, wird deutlich, dass Ersteres bei oben beschriebenem Test unsignifikant besser abschneidet.

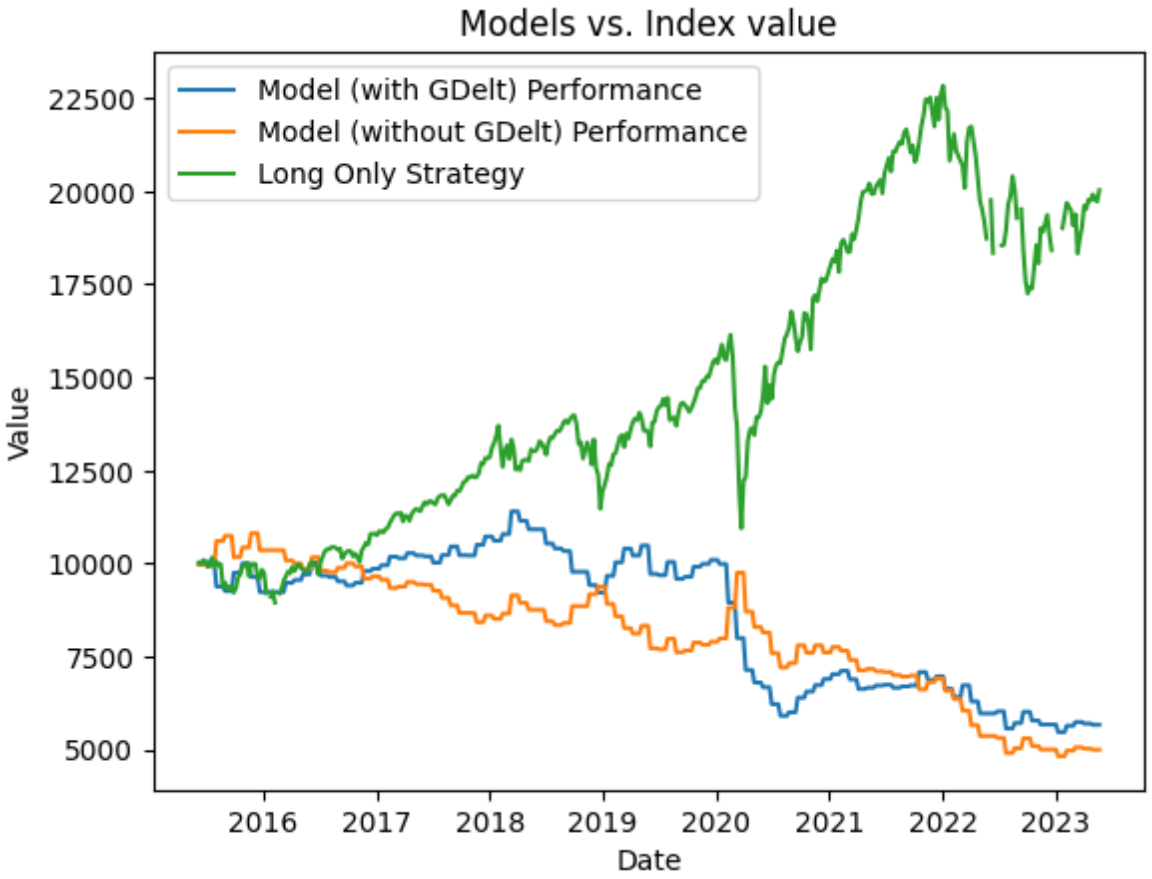


Abbildung 2.5.: Vergleich eines Modells, das auf GDELT Daten trainiert wurde vs. eines, das auf Finanzkennzahlen trainiert wurde.

Ein möglicher Grund für das Ergebnis könnte der Index sein, den wir Vorhersagen wollen. Der S&P 500 ist ein breit aufgestellter Index, welcher die Top 500 Unternehmen in den USA widerspiegelt. Dies ist einerseits ein Vorteil, da man das Investionsrisiko auf 500 Unternehmen verteilt, macht es dadurch aber schwer vorherzusagen. Negative News, wie beispielsweise der Ukraine-krieg, könnten einerseits schlecht sein für die Öl & Gas Branche, jedoch gleichzeitig gut für die Waffenindustie. Da beide Sektoren im S&P 500 vertreten sind, ist eine gute Vorhersage äußerst schwer.

Zudem besteht immer die Gefahr, dass entweder das Modell nicht komplex genug ist, oder die Daten zu noisy, was in Kombination zu solch einem Ergebnis führen könnte.

Ein weiterer Faktor ist, dass es objektiv gesehen äußerst schwer ist, solch einen Index zu schlagen. Es gibt zwar aktiv gemanagte Fonds die tatsächlich in manchen Jahren besser performen, jedoch ist das auf lange Sicht sehr selten.

3. Lessions Learned

In kleinen Schritten arbeiten

Wir haben bereits in der eigentlich noch explorativen Phase, in der wir noch verschiedene Datensätze in Betracht gezogen hatten, damit begonnen die Notebooks zunehmend in Skripte und Python Pakete mit ihren Modulen auszulagern. Jedoch mussten die Vorverarbeitungs-Skripte durch sich ändernde Datenformate der Datensätze häufig und aufwendig angepasst werden. Es hat sich somit herausgestellt, dass über den gesamten Zeitraum der explorativen Phase möglichst in Notebooks gearbeitet werden sollte, da sich die Verarbeitungsschritte hier sequentiell, schnell und einfach nachvollziehen lassen. Eine Auslagerung von Funktionalitäten in Module und schließlich die Einbettung der Prozesse in Skripte, welche sich via Kommandozeile mit allen wichtigen Argumenten aufrufen lassen, macht erst Sinn, wenn diese verstärkt von anderen Projektteilnehmern genutzt, oder gar in eine vollwertige ML-Pipeline integriert werden sollen.

Daten visualisieren

Auf Grund diverser Fehler bei der Datensammlung waren die GDELT Daten nicht nicht gleichmäßig verteilt (für einige Jahre gab es gar keine Daten und die relevanten News waren meist in den ersten fünf Tagen eines Monats). Dies wurde erst bemerkt, als die Daten bereits transformiert waren und zum ersten mal visualisiert wurden. Eine frühstmögliche Visualisierung hätte uns hier unnötige Zeitaufwendungen im Modelling erspart. Aufgrund der genannten Probleme mussten massive Anpassungen in der Datensammlung- & Transformation vorgenommen werden. Für den Prozess des Anpassens wurden die Daten in engen Abständen immer wieder visualisiert, um sicherzugehen, dass sich das Ergebnis wie erwartet verbessert.

Genügend Zeit in Datengrundlage investieren

Es wurde festgestellt, dass die Investition einer verhältnismäßig größeren Menge an Zeit in eine umfassende Datengrundlage sich als lohnenswert erweist. Zwar wollten wir am Anfang eher leicht zugängliche und einfach verständliche Datensätze benutzen, um schneller mit dem Modellierungsprozess zu starten, jedoch waren unsere Ergebnisse nicht zufriedenstellend. Die Daten hatten Lücken und Heterogenitäten, die nicht zu vernachlässigen waren. Anschließend entschieden wir uns für eine intensive Auseinandersetzung mit dem GDELT-Datensatz, welche zwar eine Menge Zeit in Anspruch nahm, jedoch in einer repräsentativen und hochqualitativen Datengrundlage resultierte. Dadurch wurden die Ergebnisse verbessert.

Gute Dokumentation

Die initialen Experimente mit DeepAR wurden zwar in der Datenbank festgehalten, die Einblicke, weshalb bestimmte Konfigurationen gut oder schlecht funktionieren allerdings nicht. Als der Fokus dann, nach der DeepState und MQ-CNN Modellierung, wieder zu DeepAR zurückgekehrt ist, wurde weiterer Fortschritt zunächst durch mangelnde Dokumentation der bisherigen Erkenntnisse aufgehalten. Diese mussten dann, durch Nachvollziehen der gespeicherten Experiment-Resultate und Konfigurationen erneut erzielt werden. Durch eine von Anfang an effektive Dokumentation, hätte man einerseits Zeit und andererseits Kosten für redundante Experimente, sparen können.

A. Anhang: Anmerkungen zum Quellcode

Die Struktur des Repository ist im enthaltenen Readme dokumentiert. Wie die Abhängigkeiten zu installieren sind, ist dort ebenfalls gezeigt. Dabei sollte eine aktuelle Version von pip verwendet werden.
Alle Skripte und Notebooks können im abgegebenen Repository jederzeit unabhängig ausgeführt werden. Der Prozess hätte chronologisch jedoch folgende Reihenfolge:

1. Data Collection: gdelt_web_crawl.ipynb
2. Data Collection: cameo_translation.ipynb
3. Data Engineering: preprocessing.py (Hier sollten die default arguments ausreichen)
4. Modelling: nlp_exploration_notebook.ipynb - Das ist das topic modelling
5. Data Engineering: timeseries_engineering.py (Hier sind die default arguments ebenfalls ausreichend)
6. Data Engineering: financial_ts_preprocessing.ipynb
7. Data Engineering: gdelt_preprocessing.ipynb
8. Modelling: modelling_environment.ipynb
9. Data Engineering: track_record.ipynb & track_record_comparison.ipynb

Literatur

[1] Alexander Alexandrov u. a. „Gluonts: Probabilistic time series models in python“. In: *arXiv preprint arXiv:1906.05264* (2019).

[2] Dimo Angelov. *TOP2VEC: DISTRIBUTED REPRESENTATIONS OF TOPICS*. 2019. URL: https://arxiv.org/pdf/2008.09470.pdf (besucht am 05. 07. 2023).

[3] Harimoto, Keiko Bao, Ruihan Ren, Xuancheng Sun, Xu Chen, Deli Zou, Yanyan. „Incorporating fine-grained events in stock movement prediction“. In: *arXiv preprint arXiv:1910.05078* (2019).

[4] *CNBC news dataset*. 2021. URL: https://data.world/crawlfeeds/cnbc-news-dataset (besucht am 09. 07. 2023).

[5] Liu, Ting Duan, Junwen Ding, Xiao Zhang, Yue. „Deep learning for event-driven stock prediction“. In: *Twenty-fourth international joint conference on artificial intelligence*. 2015.

[6] Liu, Ting Duan, Junwen Ding, Xiao Zhang, Yue. „Using structured events to predict stock price movement: An empirical investigation“. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, S. 1415–1425.

[7] Priya Dwivedi. *Extracting the main topics from your dataset using LDA in minutes*. 2018. URL: https://towardsdatascience.com/nlp-extracting-the-main-topics-from-your-dataset-using-lda-in-minutes-21486f5aa925 (besucht am 05. 07. 2023).

[8] MaartenGR. *BERTopic*$_{Wikipedia}$.

[9] Syama Sundar Rangapuram u. a. „Deep state space models for time series forecasting“. In: *Advances in neural information processing systems* 31 (2018).

[10] David Salinas. *DeepAR: Probabilistic forecasting with autoregressive recurrent networks*. 2020.

[11] Uehara, Kuniaki Yoshihara, Akira Seki, Kazuhiro. „Leveraging temporal properties of news events for stock market prediction.“ In: *Artif. Intell. Res.* 5.1 (2016), S. 103–110.