CAS Artificial Intelligence/Künstliche Intelligenz

Machine Learning Innovationsprojekt – Erfahrungsbericht

Ziel der Arbeit

Dieses Innovationprojekt steht unter dem Titel «*Anwendung und Vergleich von Machine Learning und Deep Learning Algorithmen zur Vorhersage von COVID-19 Kennzahlen».* Das Ziel der Arbeit ist es also, Algorithmen aus dem Bereich des Machine Learning (ML) und Deep Learning (DL) zu entwickeln, um damit eine Vorhersage über die aktuellen und zukünftigen COVID-19 Kennzahlen zu treffen. Wir beschränken uns im Rahmen dieses Projekts auf die Vorhersage der COVID-19 Fallzahlen.

Die Daten stammen vom Portal opendata.swiss, das vom Bundesamt für Statistik betrieben wird [1].

Auswahl der Modelle

In der Anmeldung für das Projekt haben wir folgende Modelle angegeben, die wir untersuchen und miteinander vergleichen möchten. Hier soll kurz auf die einzelnen Modelle und ihre Verwendungen eingegangen werden. Die Informationen zu dem Algorithmen stammen aus dem Buch von Sebastian Raschka und Vahid Mirjalili [2].

Machine Learning

- Regression (linear, logistisch, polynomial): Algorithmen, um lineare Zusammenhänge zwischen erklärenden Variablen und Zielvariablen zu modellieren

- Support Vector Machine (SVM): leistungsfähiger und häufig eingesetzter Algorithmus für die Klassifizierung, der zum Ziel hat, um die Klassengrenzen herum einen möglichst breiten Rand (Margin) freizuhalten

- Decision Tree: hierbei handelt es sich um ein Entscheidungsbaummodell (Klassifikator), das anhand einer Reihe von Fragen die Klassenbezeichnungen erlernt

Deep Learning

- Multilayer Perceptron (MLP): darunter wird ein mehrschichtiges Feedforward-Netzwerk bezeichnet, kann sowohl für Regression als auch für Klassifizierung eingesetzt werden

- Long short-term memory (LSTM): Rekurrentes Neuronales Netzwerk, das durch den Einsatz von Speicherzellen das Problem des verschwindenden Gradienten löst

- Convolutional Neural Network (CNN): Neuronales Netzwerk (ursprünglich für die Bildklassifikation), das auf dem Extrahieren von relevanten Merkmalen aus den Daten basiert

Nach der Diskussion mit den Dozierenden sind wir zum Schluss gekommen, nur eine Auswahl der Modelle zu implementieren. Dies ausfolgenden Gründen:

1. Nicht alle oben beschriebenen Modelle sind für Regressions-Probleme (insbesondere mit Times Series Daten) geeignet. Sie kommen bei der Klassifizierung zum Einsatz.
2. Der Aufwand für die Implementierung und Evaluation aller Modelle ist sehr hoch. Mit der Beschränkung auf einige wenige Modelle können wir uns auf diese ausgewählten Modelle konzentrieren und diese genauer untersuchen.

Wir haben uns für folgende Algorithmen entschieden:

1. Polynomiale Regression
2. Convolutional Neural Network (CNN)
3. Long short-term memory (LSTM)

Arbeitsaufteilung

Wir teilten die Arbeiten im Team wie folgt auf:

Marco kümmerte sich um die Daten Aufbereitung. Dies beinhaltet den Download der CSV Dateien von [1], das Zusammenführen der einzelnen CSV Dateien zu einer einzelnen Datei (all_data.csv), die Auswahl der Features und die Erstellung einer Beschreibung der einzelnen Features. Die explorative Datenanalyse und Datenbereinigung (Preprocessing) wurde von Jonas durchgeführt. Dabei wurden Duplikate und Null-Werte entfernt, fehlende Werte interpoliert, die Daten visualisiert, etc.

Anschliessend teilten wir auch die Entwicklung der Machine Learning / Deep Learning Modelle sowie deren Training und Evaluation auf: Marco implementierte die Polynomiale Regression und Jonas die Neuronalen Netze (CNN und LSTM), trainierte die Modelle mit den Trainings-Daten und evaluierte diese dann anhand der Test-Daten.

Methodik

Als Versionsverwaltungssystem entschieden wir uns für Git, da uns dies aus unserem Arbeitsumfeld bestens bekannt war. Den Source Code verwalteten wir dabei auf GitHub. Durch das gemeinsame Repository waren Änderungen transparent sichtbar. Wir führten gegenseitig Reviews durch und konnten dadurch die Qualität des Codes stetig verbessern. Lediglich Änderungen in Jupyter Notebooks sind in Git nicht sehr gut nachvollziehbar, da nicht nur der Python Code eingecheckt wird, sondern auch die zuletzt ausgeführten Daten und Resultate, was zu vielen Differenzen führt.

Als Entwicklungsumgebung versuchten wir zuerst mit Google Colab zu arbeiten. Dabei stiessen wir jedoch auf Probleme (z.B. beim Trainieren der Neuronalen Netzwerke war die Performance nicht sehr berauschend). Weitere Tests mit Anaconda und der Entwicklung im Browser brachten auch keine weiteren Verbesserungen. Schlussendlich entschieden wir uns als Entwicklungsumgebung für Microsoft Visual Studio Code, da diese Entwicklungsumgebung viele nützliche Features wie Autovervollständigung und Refactoring integriert und Microsoft direkt Plugins für Python und Jupyter anbietet. Als Laufzeitumgebung wurde Anaconda verwendet als Remote Server in Visual Studio Code. Visual Studio Code ist ein sehr flexibler und umfangreicher Code Editor.

Herausforderungen und persönliche Erfahrungen

Die erste grosse Herausforderung war das Besorgen der Daten. Daten aus unserem Geschäftsumfeld waren schwierig aufzutreiben aufgrund von Datenschutz und administrativen Hürden. Wir wollten ursprünglich Kundendaten von CSS-Kunden klassifizieren. Also entschieden wird uns für öffentlich zugängliche Daten (vom Bund). Da erschienen uns die COVID-19 Daten aufgrund ihrer Aktualität sehr geeignet.

Die Datenaufbereitung gestaltete sich aufwändiger als geplant. Rund 75% der Zeit wurde für die Datenaufbereitung aufgewendet. Uns wurde klar, wie wichtig die Qualität der Daten für eine gute Performance der Modelle ist.

Die Optimierung der Hyperparameter der Neuronalen Netze stellte eine weitere Herausforderung dar. Wir mussten schnell feststellen, dass das Tuning mittels GridSearch bei mehreren möglichen Werten pro Hyperparameter schon ins Unermessliche läuft (Rechner mit Nvidia GeForce RTX 3070 Grafikkarte hatte nach einem Tag noch kein Resultat ausgegeben). Somit mussten wir uns auf einige wenige mögliche Werte beschränken.

Schliesslich war das Splitten der Daten in Trainings- und Testdaten und insbesondere in Subsets für die Cross-Validation eine grosse Challenge (da es sich um Time Series Daten handelt). Hier half uns vor allem der Medium Beitrag von Keita Miyaki [3] weiter, um das Splitten mittels TimeSeriesSplit von sklearn zu verstehen.

Resultate

Folgende Resultate können wir festhalten:

Algorithmus	Performance (MSE)	Laufzeit für Training	Laufzeit für Prediction
Polynomiale Regression	0.0266	0.0030s	0.0116s
CNN	0.0427	15.9702s	0.0747s
LSTM	0.0205	57.6358s	0.1137s

Fazit

Unsere Messungen zeigen auf, dass das LSTM bezüglich der Performance das beste Modell für unser Anwendungszweck ist. Dies ist jedoch mit einer hohen Laufzeit fürs Trainieren und die Predictions verbunden. Der Aufwand zur Implementierung und die Komplexität der Modelle unterscheiden sich nicht nennenswert. Hingegen die Optimierung der Hyperparameter ist bei den Neuronalen Netzwerken wesentlich aufwändiger im Vergleich zur Regression. Durch die Veröffentlichung eines Modells zur Vorhersage der COVID-19 Zahlen könnte man die Transparenz gegenüber der Bevölkerung erhöhen. Der Aufbau einer Community, welche das Modell kontinuierlich verbessert, wäre anzustreben.

Literaturverzeichnis

[1] «Opendata Swiss». https://opendata.swiss/de/dataset/covid-19-schweiz (zugegriffen Jan. 03, 2022).

[2] S. Raschka und M. Vahid, *Machine Learning mit Python und Keras, TensorFlow 2 und Scikit-learn*, 3. aktualisierte und Erweiterte Auflage. mitp, 2021.

[3] K. Miyaki, «Time Series Split with Scikit-learn», Aug. 16, 2019. https://medium.com/keita-starts-data-science/time-series-split-with-scikit-learn-74f5be38489e (zugegriffen Jan. 03, 2022).