**Leitfaden für nachvollziehbare Schritte**

**1. Kurze Darstellung des Problembereichs / Aufriss des Themas**

**1.1 Inhaltlich**

Entwicklung und (automatisierte) Optimierung von NN-Modellen (mit Tensorflow+KerasTuner), welche die Ergebnisse von Fußballspielen vorhersagen und Aufbereitung der dafür notwendigen Daten (mit Pandas).

**1.2 Begründung des Themas**

**Darstellung der Relevanz des Themas?**

Modelle, welche Fußballspiele vorhersagen, sind zum einen für Wettbüros (zur Berechnung der Odds und damit Verlustminimierung) und zum anderen für Kunden, die gerne auf Fußballspiele wetten (zur Gewinnmaximierung) von Interesse.

**Darstellung eines persönlichen Erkenntnisinteresses.**

Auch wenn man kein Freund von Wettbüros oder Wettanbietern ist... bei Fußballbegeisterten (egal ob Liga, WM oder EM) kommt immer wieder die Frage auf: Was tippst du beim Spiel X gegen Y? Oft formieren sich Freunde oder Gleichgesinnte in Tippligen, wo es um die Ehre (und etwas Geld) geht. Da können Modelle, die einem Tipps ohne Vorwissen oder Recherche ausspucken ganz praktisch sein.

**2. Nachvollziehbare Schritte**

**2.1 Der Stand der Forschung / Auswertung der vorhandenen Literatur / Tutorials ...**

Prognosemodelle für Fußballergebnisse sind schon lange bei Wettanbietern im Einsatz, die damit eine Genauigkeit von etwa 53% (s. <https://www.kaggle.com/hugomathien/soccer>) bei Klassifikation (Sieg/Niederlage/Unentschieden) erreichen.

Literaturhinweise für Hyperparameteroptimierung:

* *Getting started with KerasTuner (Developer Guide)* https://keras.io/guides/keras\_tuner/getting\_started/
* *Visualize the hyperparameter tuning process (Developer Guide)* https://keras.io/guides/keras\_tuner/visualize\_tuning/
* *KerasTuner API*  
   https://keras.io/api/keras\_tuner/
* *Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization (Paper)* https://jmlr.org/papers/v18/16-558.html
* *Tuning the hyper-parameters of an estimator (User Guide)* <https://scikit-learn.org/stable/modules/grid_search.html>

**2.2 Fragestellung**

Können die bereits erzeugten Modelle zur Prognose der Fußballergebnisse (Multiclass Classification) und Toranzahl (Regression) durch Hyperparameteroptimierung verbessert werden?

**2.3 Stand der Forschung**

**2.4 Wissenslücke**

**2.5 Methode**

Der Python-Code ist nun in drei Dateien unterteilt:

* *data.py*: Funktionen zur Datenaufbereitung
* *model.py*: Klassen und Funktionen zur Modellierung inkl. Hyperparameteroptimierung
* *prediction.py*: Aufrufe der obigen Funktionen zum Durchführen der Multiclass Classification und der beiden Regressionsaufgaben (ohne Hyperparameteroptimierung)

Durch Ausführen von model.py (*python model.py*) wird die Hyperparameteroptimierung für alle drei Problemstellungen durchgeführt (dauert mehrere Stunden). Nach der Optimierung wird jeweils das Modell mit den besten Parametern trainiert und gegen die zurückgehaltene Testmenge evaluiert.

Als Datengrundlage dienen die bereits aufbereiteten Daten (s. NvS der letzten Projektwoche). Bei den Rohdaten handelt es sich um einen Datensatz, der auf Kaggle (<https://www.kaggle.com/josephvm/european-club-football-dataset?select=matches.csv>) verfügbar ist und am 02.11.2021 heruntergeladen wurde.

Die aufbereiteten Daten werden zuvor in Trainings-, Validierungs- und Testmenge im Verhältnis 70/15/15 unterteilt. Die Validierungs- und Testmenge ist damit gleich groß, was die Metriken MSE bzw. MAE bei der Regression zwischen den beiden Mengen vergleichbar macht. Dies war zuvor nicht der Fall, da dort ein Verhältnis von 75/15/10 gewählt wurde.

Danach werden die Merkmale standardisiert. Dieser Schritt wurde zuvor durch einen Normalization-Layer umgesetzt und nun wegen Performancegründen vorverlagert, sodass er bei der Hyperparameteroptimierung nicht jedes Mal durchgeführt wird.

Der Early-Stopping-Mechanismus ist im Vergleich zu vorher auch angepasst: Es wird nun jeweils mit maximal 200 Epochen und *patience*=20 trainiert (statt 100 und 50).

Neben *EarlyStopping* wird noch ein weiterer Callback mitgegeben, nämlich *TensorBoard*, was dafür sorgt, dass die Ergebnisse geloggt werden und anhand verschiedener Visualisierungen im Tensorboard eingesehen werden können.

Schließlich wird für die Multiclass Classification und Regression die Hyperparameteroptimierung durchgeführt. Dabei wird die Klasse *Hyperband* von *KerasTuner* mit folgenden Optionen verwendet:

* n\_hidden\_layers: 1 bis 3
* units: 32 / 64 / 128 / 256 / 512
* activation: 'relu' / 'tanh' / 'elu'
* kernel\_initializer: 'he\_uniform' / 'glorot\_uniform'
* dropout\_rate: 0.2 / 0.3 / 0.4 / 0.5
* optimizer: Adam / RMSProp
* learning\_rate: [1e-4, 1e-2]

Somit gibt es (learning\_rate mal herausgenommen) bis zu 720 (3x5x3x2x4x2) verschiedene Kombinationsmöglichkeiten.

Nach der Hyperparameteroptimierung werden die Modelle mit den besten Parametern jeweils 5x mit unterschiedlichen Datensplit (also quasi eine fünffache Kreuzvalidierung) trainiert und evaluiert. Das Gleiche wird mit den zuvor erstellten Modellen gemacht, um das Ganze vergleichen zu können.

**2.6 Ergebnisse**

**Multiclass Classification**

Zur Erinnerung: Ein primitives Modell, welches immer einen Heimsieg vorhersagen würde, hätte eine Accuracy von 0,4558.

Top 5 (mit max. Accuracy > 0,517):

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| activation | kernel\_initializer | lr | n\_layers | optimizer | rate | units0 | units1 | units2 | min\_acc | max\_acc |
| elu | glorot\_uniform | 0.0041 | 1 | Adam | 0.4 | 128 |  |  | 0.5086 | 0.5179 |
| elu | glorot\_uniform | 0.0004 | 1 | Adam | 0.5 | 64 |  |  | 0.5119 | 0.5176 |
| elu | he\_uniform | 0.0023 | 2 | RMSprop | 0.4 | 512 | 32 |  | 0.4986 | 0.5176 |
| elu | glorot\_uniform | 0.0002 | 2 | Adam | 0.4 | 32 | 64 |  | 0.5122 | 0.5173 |
| elu | glorot\_uniform | 0.0025 | 3 | RMSprop | 0.5 | 256 | 128 | 64 | 0.5010 | 0.5173 |

ELU und glorot\_uniform sowie eine Dropout-Rate von mind. 0,4 scheinen eine gute Wahl zu sein. Bei Lernrate, Optimierer und Anzahl an Layern und Neuronen ist die Lage nicht so eindeutig. Betrachtet man die minimal Accuracy so fällt auf, dass die Durchläufe mit Adam weniger streuen und eine Lernrate zwischen 0,0002 und 0,0004 wohl gut geeignet ist.

Test Accuracy – Mean (±StdDev)

* vorher: 0,5219 (±0,0094)
* nachher: 0,5205 (±0,0051)

**Regression**

Hierzu wurde aus Zeitgründen die Hyperparameteroptimierung lediglich für die Heimtore durchgeführt.

Top 5 (mit min. MAE <0,974):

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| activation | kernel\_initializer | lr | n\_layers | optimizer | rate | units | min\_mae | max\_mae |
| relu | glorot\_uniform | 0.0017 | 1 | Adam | 0.3 | 64 | 0.9698 | 0.9790 |
| relu | he\_uniform | 0.0021 | 1 | Adam | 0.3 | 32 | 0.9701 | 0.9769 |
| relu | glorot\_uniform | 0.0026 | 1 | RMSprop | 0.3 | 32 | 0.9712 | 0.9789 |
| relu | glorot\_uniform | 0.0017 | 1 | Adam | 0.3 | 64 | 0.9724 | 0.9758 |
| relu | he\_uniform | 0.0093 | 1 | RMSprop | 0.4 | 32 | 0.9739 | 0.9969 |

Hier machen Modelle mit nur einem Hidden Layer, ReLu, einer Lernrate um 0,002 und einer Dropout-Rate von 0,3 das Rennen. Zudem scheinen wieder glorot\_uniform und Adam eine gute Wahl zu sein. Für die Anzahl an Neuronen langen 64 oder sogar nur 32.

Test MAE – Mean (±StdDev) für Tore der Heimmannschaft

* vorher: 0,9822 (±0,0152)
* nachher: 0,9763 (±0,0062)

Test MAE – Mean (±StdDev) für Tore der Auswärtsmannschaft

* vorher: 0,8610 (±0,0186)
* nachher: 0,8535 (±0,0149)

**Fazit**

Durch die Hyperparameteroptimierung mit Hyperband konnte bei der Multiclass Classification keine und für die beiden Regressionsaufgaben eine geringe Verbesserung erzielt werden. Die ausbleibende Verbesserung bei der Klassifikation ist zum Einen wohl auf die „händische“ Optimierung zurückzuführen, die bei der ersten Modellierung bereits durchgeführt wurde. Zum Anderen liegt hier offenbar eine Problemstellung vor, wo Zufall eine recht große Rolle spielt und somit die Ergebnisse (zumindest mit den vorliegenden Daten) schwer zu prognostizieren sind. Es ist also nicht verwunderlich, dass die Modelle der Wettanbieter auch nur eine Accuracy von ca. 0,53 erzielen.

Neben der Regression konnte noch die Streuung bei allen drei Modellen verringert werden. Die Hyperparameteroptimierung hat somit offenbar zu robusteren Modellen geführt.

**2.7 Ausblick**

Eine Verbesserung könnte durch die Hinzunahme von weiteren Features, wie z.B. Passgenauigkeit, Marktwert oder FIFA-Ratings (s. <https://sofifa.com/teams>), erzielt werden. Dies kann allerdings recht aufwendig sein, da hierzu zusätzliche Datenquellen (ggf. durch *Web Scraping*) beschafft und (vor der Datenaufbereitung) verknüpft werden müssten.

Spannend wäre es, die Modelle für eine Vorhersage des nächsten Spieltages (z.B. der Bundesliga) zu nutzen, um sie bei echtem Einsatz evaluieren zu können. Zudem wäre es interessant, wenn man bei den Vorhersagen (zur Klassifikation) die Klassenwahrscheinlichkeiten mit ausgeben würde (analog zu scikit-learn’s *predict\_proba-*Methode).