**Leitfaden für nachvollziehbare Schritte**

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

**1.1 Inhaltlich**

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

**1.2 Begründung desThemas**

**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.

Allgemeines Tutorial zur Klassifikation tabellarischer Daten mit Tensorflow:

https://www.tensorflow.org/tutorials/structured\_data/preprocessing\_layers?hl=en

**2.2 Fragestellung**

Wie genau können Fußballergebnisse (Multiclass Classification) und Toranzahl (Regression) anhand von Statistiken vergangener Spiele (wie z.B. Ballbesitz, Anzahl Schüsse, Ecken, etc.) vorhergesagt werden?

**2.3 Stand der Forschung**

**2.4 Wissenslücke**

**2.5 Methode**

Datenquelle: https://www.kaggle.com/josephvm/european-club-football-dataset?select=matches.csv

Die CSV-Datei *matches.csv* enthält 24240 Zeilen (Fußballspiele von 6 europäischen Topligen) und 210 Spalten mit Informationen zu Spielergebnis, Statistiken (wie z.B. Ballbesitz) und Aufstellung. Für die Problemstellung sind allerdings nur die folgenden 19 Spalten relevant:

| Spaltenname | Erläuterung |
| --- | --- |
| home | Name der Heimmannschaft |
| away | Name der Auswärtsmannschaft |
| date | Datum (ohne Jahresangabe) |
| year | Startjahr der Saison |
| League | Bezeichnung der Liga |
| game\_status | Spielstatus |
| home\_score / away\_score | Anzahl Tore von Heim-/Auswärtsmannschaft |
| home\_possessionPct | Anteil Ballbesitz von Heimmannschaft |
| home\_shotsSummary / away\_shotsSummary | Anzahl Torschüsse von Heim-/Auswärtsmannschaft |
| home\_foulsCommitted / away\_foulsCommitted | Anzahl Fouls von Heim-/Auswärtsmannschaft |
| home\_offsides / away\_offsides | Anzahl Abseits von Heim-/Auswärtsmannschaft |
| home\_wonCorners / away\_wonCorners | Anzahl Ecken für Heim-/Auswärtsmannschaft |
| home\_saves / away\_saves | Anzahl Paraden von Heim-/Auswärtsmannschaft |

**Tabelle 1: Spalten der Rohdaten**

Nach Bereinigung und Vorverarbeitung der Rohdaten bleiben 20950 Zeilen übrig.

Die Spalten *home\_score* / *away\_score* bilden die Zielvariablen der Regressionsprobleme. Zudem werden aus den beiden Spalten die Zielvariable für das Klassifikationsproblem abgeleitet. Diese wird *winner* benannt und hat die Ausprägungen 0 (Sieg der Heimmannschaft), 1 (Sieg der Auswärtsmannschaft) und 2 (Unentschieden) mit folgenden Anzahlen:

0: 9488

1: 6163

2: 5165

Ein primitives Modell, welches immer einen Heimsieg vorhersagen würde, würde also in 45,58% der Fälle richtig liegen. Folglich sollte das Klassifikation-Modell eine *Accuracy* von über 0,46 aufweisen.

Die Daten werden zunächst so aufbereitet, dass man für jedes Spiel die Mittelwerte (von *home\_score* / *away\_score* bis *home\_saves* / *away\_saves*) der vorangegangenen *n* (z.B. 5) Spieltage von beiden Mannschaften ableitet. Danach wird zu jedem Mittelwert die Differenz zwischen Heim- und Auswärtsmannschaft gebildet (also z.B. *shots\_diff* = *mean\_shots\_home* - *mean\_shots\_away*). Diese Differenzen bzw. Deltas bilden die Features, so dass schließlich folgende Targets und numerischen Features vorliegen:

| Spaltenname | Erläuterung |
| --- | --- |
| home\_score / away\_score | Targets für Regression |
| winner | Target für Klassifikation |
| points\_diff | Delta der durchschn. erzielten Punkte (aus den letzten *n* Spielen) |
| possession\_diff | Delta des durchschn. erzielten Ballbesitzes |
| goals\_diff / goals\_opp\_diff | Delta der durchschn. geschossenen / kassierten Tore |
| shots\_diff / shots\_opp\_diff | Delta der durchschn. abgegebenen / kassierten Torschüsse |
| fouls\_diff / fouls\_opp\_diff | Delta der durchschn. ausgeteilten / kassierten Fouls |
| offsides\_diff / offsides\_opp\_diff | Delta der durchschn. aufgestellten / kassierten Abseitsstellungen |
| corners\_diff / corners\_opp\_diff | Delta der durchschn. geschossenen / kassierten Ecken |
| saves\_diff / saves\_opp\_diff | Delta der durchschn. gemachten / kassierten Paraden |

**Tabelle 2: Targets und Features**

Die Datenaufbereitung ist durch verschiedene Funktionen und die Modellierung in drei Klassen implementiert. Für die Modellierung werden die aufbereiteten Daten in Trainings- (75%), Validierungs- (15%) und Testdatensatz (10%) geteilt und anschließend (von Pandas-Dataframes) in Tensorflow-Datasets umgewandelt. Die neuralen Netze sind für die Klassifikation und Regression sehr ähnlich aufgebaut: Nach dem Input-Layer folgt ein Normalization-Layer, wo alle Features normalisiert werden. Danach folgen zwei Dense-Layer mit 128 Knoten und Relu-Aktivierungsfunktion, wobei jeweils nur die Hälfte der Knoten (pro Epoche) trainiert wird (Dropout 0.5), um Overfitting zu vermeiden. Die beiden Modelle unterscheiden sich nur beim Output-Layer und bei Loss-Funktion und Metrik:

* Für die Klassifikation enthält der Output-Layer drei Knoten (d.h. einen pro Klasse) mit Softmax-Aktivierungsfunktion. Als Loss-Funktion wird *Categorical Crossentropy* und als Metrik *Accuracy* verwendet.
* Für die Regression enthält der Output-Layer einen Knoten mit linearer Aktivierungsfunktion. Als Loss-Funktion wird *Mean Squared Error (MSE)* und als Metrik *Mean Absolut Error (MAE)* verwendet.

Es wird jeweils mit 100 Epochen trainiert, wobei ein *Early-Stopping*-Mechanismus implementiert ist, der dafür sorgt, dass die Trainingsphase vorzeitig beendet wird, wenn sich der Loss-Wert beim Validierungsdatensatz seit 50 Epochen nicht mehr verbessert hat. Die trainierten Modelle werden anhand der zurückgehaltenen Testdaten evaluiert.

**2.6 Ergebnisse**

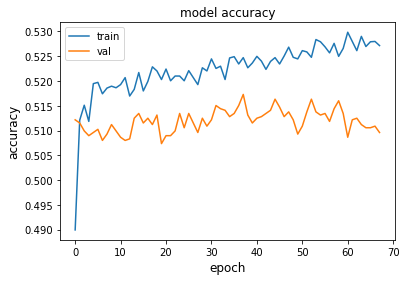
Es wurden verschiedene Einstellungen bzw. Hyperparameter durchgespielt:

* 1 bis 3 Dense-Layer
* 32/64/128/256 Knoten
* 0.2 bis 0.5 Dropout

Zudem wurde mit unterschiedlichem *n* (4/5/6/8/10) für die Anzahl an zu berücksichtigen Spiele bei der Bildung der Mittelwerte (während der Datenaufbereitung) modelliert. Dabei konnten für n=8 und den o.g. Einstellungen (2 Dense-Layer mit 128 Knoten und Dropout 0.5) die besten Ergebnisse erzielt werden. Letztere werden nun hier ausgegeben:

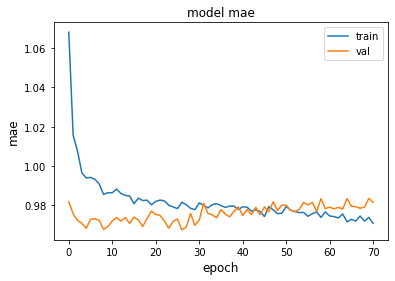
Multiclass Classification

* Training & Validation Score (s. Abb. 1)
* Test Score: 0.5317 (Accuracy)
* Early Stopping bei Epoche 68

Abbildung 1: Validation Curve für Multiclass Classification

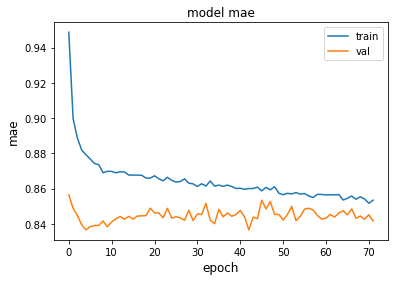
Regression (für Tore der Heimmannschaft)

* Training & Validation Score (s. Abb. 2)
* Test Score: 1.6186 (MSE) bzw. 1.0023 (MAE)
* Early Stopping bei Epoche 71

Abbildung 2: Validation Curve für Regression (Tore der Heimmannschaft)

Regression (für Tore der Auwärtsmannschaft)

* Training & Validation Score (s. Abb. 3)
* Test Score: 1.2742 (MSE) bzw. 0.8710 (MAE)
* Early Stopping bei Epoche 72

Abbildung 3: Validation Curve für Regression (Tore der Auwärtsmannschaft)

Fazit: Bei allen drei Modellen liegen jeweils Training, Validation und Test Score nicht weit auseinander. Die Modelle sind also recht robust und weißen (dank *Dropout* und *Early Stopping*) wenig bis kein Overfitting auf. Der Klassifizierer kann mit den Modellen der Wettanbieter, die eine Accuracy von ca. 0,53 haben (s.o.), mithalten.

**2.7 Ausblick**

Die Modelle können durch *Hyperparameter Tuning* (z.B. mit *Keras Tuner* oder scikit-learn’s *GridSearchCV*) optimiert werden. 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).