Fußball-Bundesliga Spielprognosen

Vanessa Tsingunidis, Moritz Kniebel und Tim Fischer

07.02.2019

Agenda

4 diskrete Teilbereiche:

1. Repräsentation der Daten
2. Beschaffung der Daten
3. Vorhersage
4. Interaktion

- Focus: Designentscheidungen

Repräsentation der Daten

Ansprüche:

- mehr als nur *bei Runtime*
- keine Duplikate
- einfach unterteilbar/abfragbar
- effizient

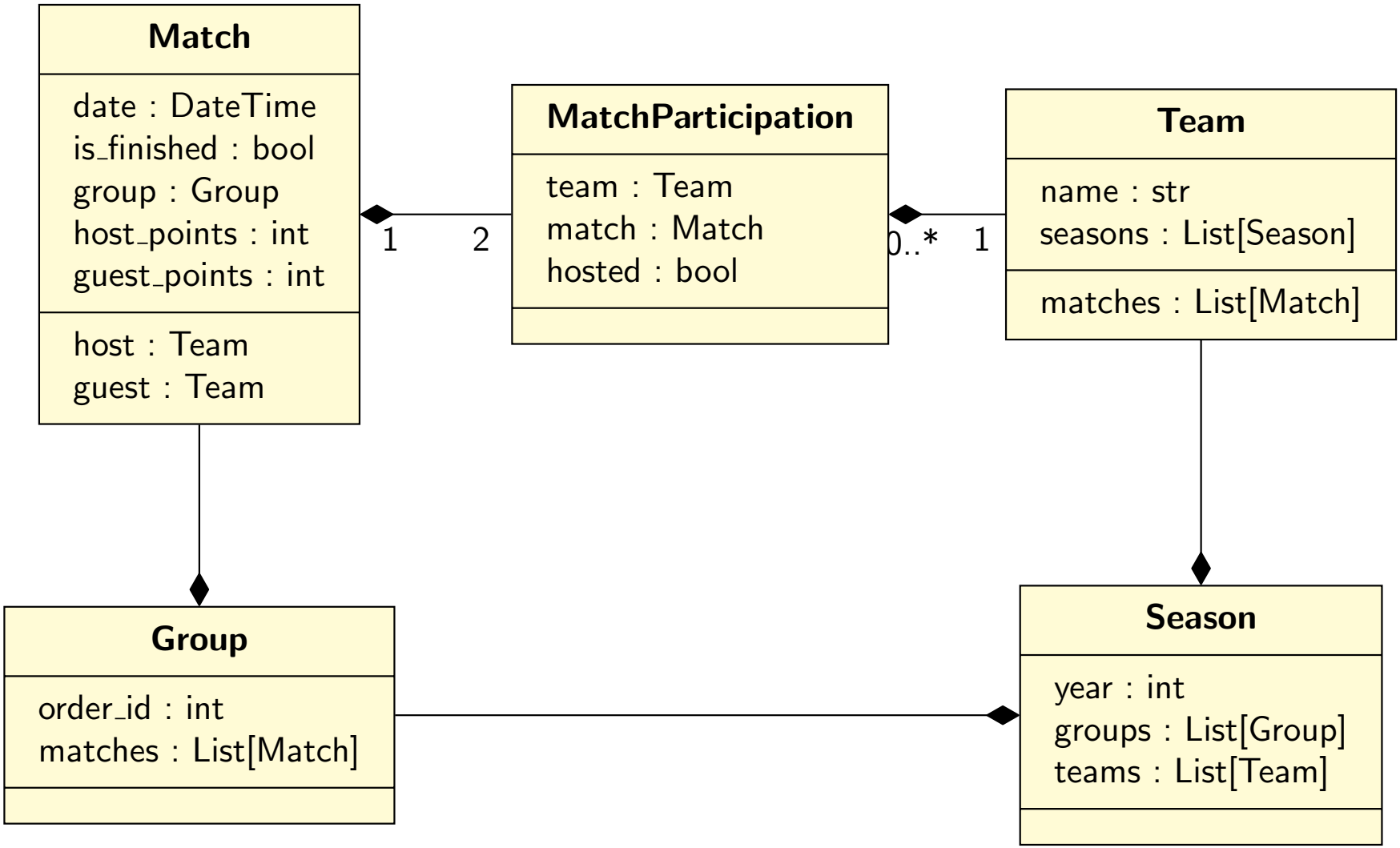
⇒ Relationale Datenbank

Designvorgabe:

- unabhängig von Webframeworks
- möglichst wenig Overhead
- einfache Darstellung in Code

⇒ ORM via SQLAlchemy (`https://www.sqlalchemy.org/`)

Repräsentation der Daten: DB-Model



Repräsentation der Daten: Datenselektion

Problem

Das Bauen/Bilden von Queries zur Auswahl von Spielen auf Basis deren Spieltag oder gar von Teams auf Basis der Spielen, ist langwierig und kompliziert...

```
Query(Team).join(
    Team.seasons,
    Team.match_participations,
    MatchParticipation.match,
    Match.group
).filter(
    Match.is_finished,
    ...
)
```

Repräsentation der Daten: Datenselektion (src/db/selectors.py)

Idee

Stellen wir eine derartige Auswahl als Zeitraum dar, welcher dann intern die Logik für das Erstellen der Queries enthält.

Implementation

```
range_selector = RangeSelector(
    start=RangePoint(year=2016, group=1),
    end=RangePoint(year=2016, group=34)
)

with DB.get_session() as session:
    print(*range_selector.build_team_query().with_session(session), sep="\n")
```

Beschaffung der Daten

3 Möglichkeiten:

1. statischer Datensatz
2. Web Scraping
3. öffentliche API

$\Rightarrow$ alle haben Vor- und Nachteile...

Beschaffung der Daten: statischer Datensatz

Pro

- kein Internetzwang
- kein zusätzlich Aufwand für die Datenaufbereitung

Kontra

- möglicher organisatorischer Aufwand
- erschwertes Verbreiten neuer Datensätze an Endnutzer

Beschaffung der Daten: Web Scraping

Pro

- wenig organisatorischer Aufwand
- immer aktuelle Daten

Kontra

- eigene Parse-Arbeit
- zusätzlich Aufwand für die Datenaufbereitung
- Willkür der Anbieter
- Internetzwang
- Legalität

Beschaffung der Daten: öffentliche API

Pro

- wenig organisatorischer Aufwand
- immer aktuelle Daten
- keine eigene Parse-Arbeit

Kontra

- *zusätzlich Aufwand für die Datenaufbereitung*
- Willkür der Anbieter
- Internetzwang

Beschaffung der Daten

3 Möglichkeiten:

1. statischer Datensatz
2. Web Scraping
3. öffentliche API (`https://www.openligadb.de`)

Beschaffung der Daten: Dataprocessing-Pipeline

Problem

Wie bekommen wir die Daten aus der API-Response in die Datenbank?

Beschaffung der Daten: Dataprocessing-Pipeline (src/acquisition/*)

Idee

Beschreiben wir für jedes Feld in den DB-Modellen, durch eine *Konkatenation* von *Transformationen*, wie man von dem Response aus zu dem jeweiligen Wert kommt.

Implementation

```
pipeline: Pipeline[Model] = Pipeline({
    MatchParticipation: {
        "team": Get("Team") | GetOrCreate(Team, match_targets=["id"]),
        "match_id": Get("MatchID"),
        "hosted": Get("hosted"),
    },
    Team: {
        "id": Get("TeamId"),
        "name": If(
            cond=lambda data: "ShortName" in data and data["ShortName"],
            then=Get("ShortName"),
            else_=Get("TeamName"),
        )
    },
    ...
})
```

Vorhersage

1. Poisson-Regression
2. Dixon-Coles (`http://web.math.ku.dk/~rolf/teaching/thesis/DixonColes.pdf`)

Poisson-Regression: Basis

Kernidee

Die Anzahl der geschossene Tore eines Teams in einem Spiel kann mittels der Angriffs- und Verteidigungsstärken der beiden Teams ermittelt werden.

Annahme

$$X_{i,j} \approx Poisson(\alpha_i \beta_j \gamma)$$
$$Y_{i,j} \approx Poisson(\alpha_j \beta_i)$$

- α_t und β_t stellen jeweils die Angriffs- und Verteidigungsstärke des Team t dar
- i und j stellen die beiden Teams dar
- γ stellt den durchschnittlichen Heimvorteil dar

Poisson-Regression: Modell

Bivariates Poisson

- Anzahl der Tore der Heim- bzw. Gastmannschaft

$$P(X_{i,j} = x, Y_{i,j} = y) = \frac{e^{-\lambda}\lambda^x}{x!}\frac{e^{-\mu}\mu^y}{y!}$$

$$\text{where } \lambda = \alpha_i\beta_j\gamma \quad \mu = \alpha_j\beta_i$$

Poisson-Regression: Modell

Likelyhood Funktion

$$L(\alpha_i, \beta_i, \gamma; i = 0, \ldots, |T|) = \prod_{m=0}^{|M|} \frac{e^{-\lambda}\lambda^{x_m}}{x_m!} \frac{e^{-\mu}\mu^{y_m}}{y_m!}$$

$$\text{where } \lambda_m = \alpha_{i_m}\beta_{j_m}\gamma \quad \mu_m = \alpha_{j_m}\beta_{i_m}$$

- i_m und j_m stellen die Teams im Spiel m dar
- x_m und y_m stellen die geschossenen Tore im Spiel m dar

Dixon-Coles: Basis

Annahme

- Poisson überschätzt in den niedrigen Torbereichen
- Nicht alle Daten sind gleich viel Wert, je jünger sie sind desto relevanter sind sie.

Dixon-Coles: Modell

τ-Funktion

- Diese Funktion stellt den Grad dar, um welchen die Wahrscheinlichkeiten für niedrige Torergebnisse (< 2) angepasst werden.

$$\tau_{\lambda,\mu}(x,y) = \begin{cases} 1-\lambda\mu\rho, & \text{if } x=y=0 \\ 1+\lambda\rho, & \text{if } x=0, y=1 \\ 1+\mu\rho, & \text{if } x=1, y=0 \\ 1-\rho, & \text{if } x=1, y=1 \\ 1, & \text{otherwise} \end{cases}$$

$$D(X_{i,j}=x, Y_{i,j}=y) = \tau_{\lambda,\mu}(x,y)\frac{e^{-\lambda}\lambda^x}{x!}\frac{e^{-\mu}\mu^y}{y!}$$

- ρ stellt hier eine Korrekturfaktor dar

Dixon-Coles: Modell

Likleyhood with Time Decay

- Nicht alle Daten haben die gleiche Relevanz.

$$\phi(t) = e^{\xi t}$$

$$L(\alpha_i, \beta_i, \rho, \gamma; i = 0, \ldots, |T|) = \prod_{m=0}^{|M|} \left\{ \tau_{\lambda_m, \mu_m}(x_m, y_m) \frac{e^{-\lambda_m} \lambda_m^{x_m}}{x_m!} \frac{e^{-\mu_m} \mu_m^{y_m}}{y_m!} \right\}^{\phi(t_0 - t_m)}$$

$$\text{where } \lambda_m = \alpha_{i_m} \beta_{j_m} \gamma \qquad \mu_m = \alpha_{j_m} \beta_{i_m}$$

- t_0 stellt den Zeitpunkt des jüngsten Spiels dar
- t_m stellt den Zeitpunkt des Spiels m dar
- ξ stellt einen Faktor zur Kontrolle des Relevanzabfalls dar ($\xi = 0.0065$)

Dixon-Coles: Modell

Maximierung der Likelyhood Funktion

Initialwerte:

- Angriff: $0.01 \rightarrow \alpha$
- Verteidigung: $-0.08 \rightarrow \beta$
- Korrektur: $0.03 \rightarrow \rho$
- Heimvorteil: $0.06 \rightarrow \gamma$

Einschränkungen:

- $\frac{1}{|T|} \sum_i \alpha_i = 1$

Vorhersage: Interpretation der Resultate

...

Interaktion

3 Möglichkeiten:

1. CLI
2. Web GUI
3. Native GUI

Interaktion: GUI Optionen

Web Interface

Pro

- flexible
- *"hostbar"*

Kontra

- extra Libraries
- HTML/CSS (JS)

Native Interface

Pro

- in Python-*"STL"*
- einfacher Systemzugriff

Kontra

- Komplexere Code
- nicht so hübsch

Interaktion

3 Möglichkeiten:

1. CLI
2. Web GUI
3. Native GUI

$\Rightarrow$ Click (`https://palletsprojects.com/p/click/`)
$\Rightarrow$ tkinter (`https://docs.python.org/3.5/library/tkinter.html`)

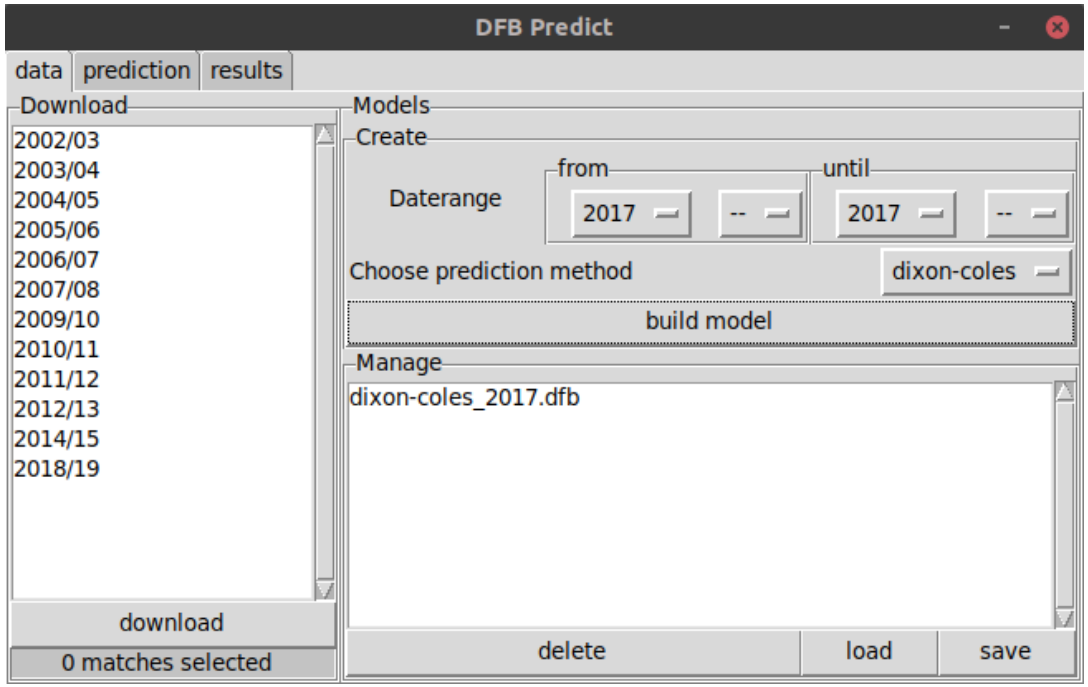
Interaktion: CLI (src/cli.py)

Layout

Interaktion: Data Tab

Funktion

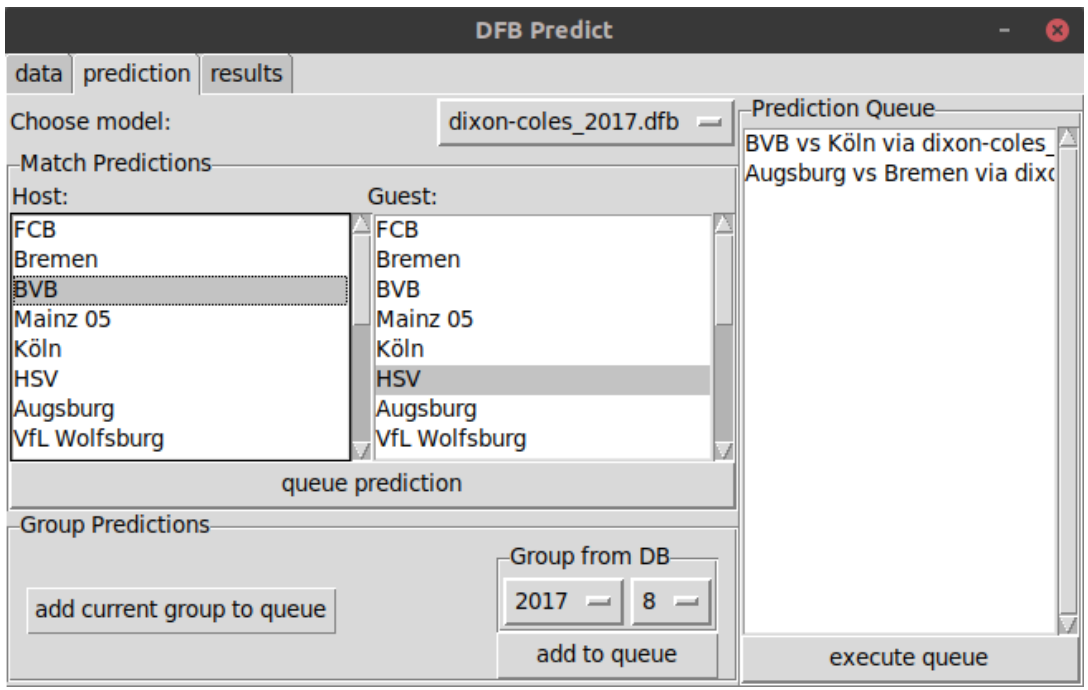
- Herrunterladen neuer Daten
- Bauen neuer Modelle
- Laden/Speichern existierender Modelle



Interaktion: Prediction Tab

Funktion

- Spezifische Spiele vorhersagen
- Aktuellen Spieltag vorhersagen
- Spieltag aus der Datenbank vorhersagen



Interaktion: Result Tab

Funktion

- Vorhersagen begutachten