Spielregeln

Alle Spieler starten mit **25** Geld. Initial müssen alle Spieler ein Modell auswählen und ihre Daten beschaffen.
Es werden insgesamt 5 Runden gespielt. Pro Runde wird reihum gewürfelt.

Ein Spielzug besteht aus folgenden Schritten:

1. Geld auf Basis von Kosten und Gewinn berechnen
2. würfeln (je nach Ereignis evtl. mehrfach)
3. abschließend auf Basis vom Ereignis neu Geld/Kosten/Gewinn berechnen

Gewinn
Der Gewinn berechnet sich aus der Accuracy des gewählten Modells minus 0,3 mal 15, dies anschließend aufgerundet. Beispiele:

$(0{,}85 - 0{,}3) * 15 = \mathbf{9}$
$(0{,}6 - 0{,}3) * 15 = \mathbf{5}$

initiale Kosten

Deep Learning: **4** Geld pro Runde	*KNN*: **4** Geld pro Runde
Decision Tree: **2** Geld pro Runde	*Regelsystem*: **2** Geld pro Runde

weitere Kosten

- Daten beschaffen:
  - initial: **10** Geld
  - später: **5** Geld
- Passenden ML Ansatz designen: **5** Geld
  - Extrakosten Deep Learning: **5** Geld
    (jede Iteration braucht länger und bessere Hardware)
- Modell nachtrainieren: **2** Geld

Ereignisse

Die Ereignisse treten in Abhängigkeit von der gewürfelten Zahl ein.

⚀ Die Last verändert sich (nochmal würfeln)

⚀⚁⚂ Last steigt: Ressourcen kosten jetzt das Doppelte (Maximum: 10)

⚃⚄⚅ Last sinkt: Ressourcen kosten jetzt die Hälfte (Minimum: 1)

⚀⚁ Übles Ereignis (nochmal würfeln)

Buzzer abfeuern nicht vergessen!

⚀⚁ Kunden fordern Erklärung

→ Notabschaltung und Fallback auf Regelsystem bei geringer Erklärbarkeit

⚂⚃ Investigativer Journalismus: Bias / Altersdiskriminierung

→ Notabschaltung und Fallback auf (Not-)Regelsystem, wenn Alter für Vorhersage zwingend notwendig ist

⚄ Adversarial Attack: Angreifer versuchen, die Vorhersage zu manipulieren

→ Bei simplen Decision Boundaries funktioniert dies leicht,
aber du bemerkst den Schaden erst spät, da kein Drift vorliegt: **du gehst pleite**

⚅ Hackerangriff

→ Wenn du Kundendaten in Produktion hast: **du bist aus dem Geschäft**

⚂⚃ Die Welt ändert sich, Drift setzt ein

- Du musst neue Daten beschaffen: **5** Geld
- Beim Regelsystem fallen keine weiteren Kosten an, ansonsten nochmal würfeln

⚀⚁ Das Modell performt auf den neuen Daten immer noch super

⚂⚃ Es muss mit neuen Daten trainiert werden, das langt aber: **2** Geld

→ bei instabilem Modell doppelte Kosten

⚄ Dramatischer Absturz der Performance

→ Notabschaltung und zurück zu Regelsystem

⚅ Neue Modell-Architektur erreicht ursprüngliche Performance
mit vertretbarem Aufwand

→ passenden ML-Ansatz nochmal designen: **5** Geld
(hier auch bei Deep Learning)

⚅ Es läuft alles rund