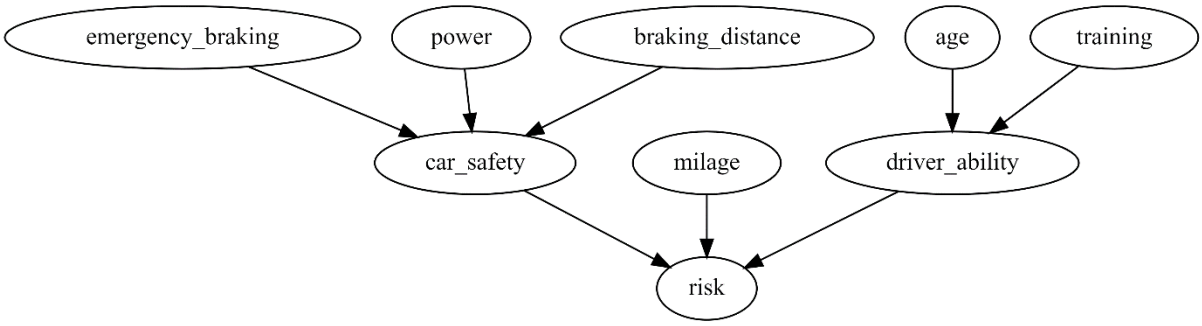
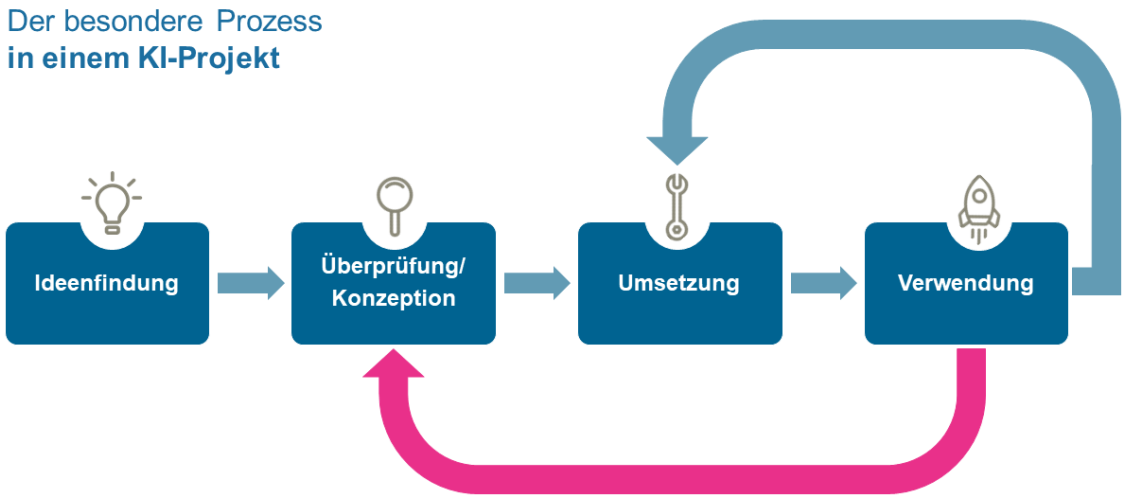
Story

Die Firma Offen Versichern GmbH bietet Kfz-Versicherungen nach geschätztem Risiko der zu Versichernden an. Für die Schätzung werden die folgenden kausalen Zusammenhänge angenommen:



- Fahrer
  - ***training***: 0/1, hat der Fahrer ein Fahrertraining absolviert
  - ***age***: Alter des Fahrers in Jahren
- Fahrzeug
  - ***emergency_braking***: 0/1, hat das Fahrzeug ein Notbremssystem
  - ***braking_distance***: Bremsweg aus Tempo 100
  - ***power***: Leistung in KW
- ***milage***: Jährliche Fahrleistung in Meilen

Für diese Zusammenhänge können wir Daten beschaffen und auf deren Basis unterschiedliche Systeme erstellen. Diese Systeme werden gemäß dem folgenden Prozess in Produktion gebracht



Du musst in der Phase „Konzeption“ ein passendes System auswählen, das sich in der Phase „Verwendung“ bewähren muss. Unterschiedliche Systeme haben unterschiedliche anfängliche und laufende Kosten und bringen unterschiedlich hohen Geschäftswert, der sich in Nutzen ausdrückt.

Glossar

Eigenschaften von Ansätzen

- Feature: Eingaben in das Vorhersage-System, also Alter, Leistung, etc.
- Erklärbarkeit: Wie gut kann einem (nicht technischen) Kunden erklärt werden, wie das System zu einem Schluss gekommen ist
- Genauigkeit: Anhand einer Test-Menge, welchen Anteil von korrekten Vorhersagen hat das Modell gemacht?
- Stabilität: Wenn neue Daten erhoben werden, verändert sich dann die Vorhersage des Modells auch für Bereiche, die von den neuen Daten gar nicht betroffen sind?
- Ressourcenbedarf: Wie aufwändig und damit teuer ist der Betrieb des Modells
- Innere Komplexität: Wie leicht ist die Funktionsweise des Modells für technische Menschen nachvollziehbar.

Ereignisse

- Drift: Die Welt ändert sich und dein Modell passt möglicherweise nicht mehr.
- Adversarial Attack: Nutzer haben die interne Funktionsweise deines Systems durchschaut und machen sich dieses Wissen zu Nutze, indem sie ihre Angaben so anpassen, dass sie einen Vorteil erhalten.

Ziel des Spiels

Durch Auswahl des Ansatzes und auch während das Spiels versucht jeder Spieler über eine feste Anzahl von Runden möglichst viele Credits zu verdienen. Dabei ergeben sich unterschiedliche Ereignisse, die den Verlauf beeinflussen und den Spielern ein Gefühl für ein Machine Learning Projekt in Produktion geben. Der Spieler mit den meisten Credits am Ende der letzten Runde gewinnt.

Startphase

1. Jeder Spieler wählt einen (ML-)Ansatz aus, mit dem das Spiel begonnen wird.
    a. **Welche Ansätze in Frage kommen wird im Exkurs ML-Ansätze eingeführt**
2. Unterschiedliche Ansätze haben unterschiedliche initiale und laufende Kosten
3. Dieser Ansatz kann (oder muss!) innerhalb des Spiels durch entsprechende Kosten bei jeder Runde geändert werden.
4. Nun wird reihum gewürfelt und eine Simulation des ML Systems in Produktion simuliert
5. Der Spieler mit dem an kürzesten zurückliegenden Komplettausfall fängt an

Spielverlauf

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

Spezialregel Modellwechsel

Wenn man feststellt, dass man mit seinem Modell nicht mehr zufrieden ist, kann man es zwischen den Schritten 1 und 2 seines eigenen Spielzugs wechseln. Dies zählt nicht als eigener Zug.
Dabei müssen keine neuen Daten beschafft werden, aber die ML-Design-Kosten fallen an.

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

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

→ wenn Alter nicht zwingend notwendig ist: dieses Feature fällt weg, damit hat das Modell eine geringere Genauigkeit, der Gewinn reduziert sich um 2

⚄ 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 musst einen anderen Ansatz wählen (s. Spezialregel)

⚃ Hackerangriff

→ Wenn du Kundendaten in Produktion hast: zahle 10 Geld, damit deine Marketingabteilung das ausbügelt, danach musst du einen anderen Ansatz wählen (s. Spezialregel)

⚂ Gutes Ereignis (nochmal würfeln)

⚀⚀⚀ Es hat sich ein Investor gefunden, der dir deine Firma abkaufen will. Würfel noch einmal, diese Zahl mal 10 ist sein Angebot. Du musst entscheiden, ob du es annimmst und damit das Spiel beendest oder weiterspielen willst

⚂⚂⚃ Dein Modell hat wegen [hier guten Grund ausdenken] an Relevanz gewonnen, dein Gewinn verdoppelt sich

⚅⚅ 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

⚁⚃ 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