* Spiel(en) mit KI und Data Science in Python –
* Dokumentation und Analyse -

|  |
| --- |
| Inhaltsverzeichnis  [Aufgabenverteilung 1](#_Toc142253238)  [Programm Aufbau 2](#_Toc142253239)  [KI-Konzept: 3](#_Toc142253240)  [Data Science 1: Statistischer Vergleich der verschiedenen KIs 4](#_Toc142253241)  [Data Science 2: Gibt es einen Vorteil/Nachteil für den/die Startspieler\*in? 7](#_Toc142253242)  [Ausblick und Fazit 11](#_Toc142253243) |

# Aufgabenverteilung

|  |  |
| --- | --- |
| *Aufgabenbereiche* | *Umsetzung* |
| Player | Lilli |
| MyBot | Jan |
| Board  -> Has\_won Logik | Lilli  Jan |
| Game | Lilli |

Die Aufgabenverteilung hat wie in der Tabelle abgebildet stattgefunden. Jedoch wurde keins der Gebiete vollständig allein umgesetzt.   
Vor Beginn der praktischen Umsetzung, haben wir gemeinsam ein Konzept entworfen und auch vor Erstellung der jeweiligen Klassen haben wir zusammen entschieden was umgesetzt werden muss und auf welche Weise. Während der generellen Bearbeitung haben wir uns gegenseitig unterstützt, Probleme gemeinsam gelöst und den Code des jeweils anderen überarbeitet und überprüft.

# Programm Aufbau

Unser Programm setzt sich zusammen aus den vier Klassen Player, Bot, Board, und Game.

Ein Objekt der **Klasse Player** verwaltet lediglich seine Information, wie den Namen des Spielers und seine Nummer. Außerdem kann er mit einem ihm übergebenen Spielfeld interagieren und darauf einen Zug machen, indem er seine Signatur (Spielernummer) an ausgewählter Position setzt.

Die **Klasse Bot**, erbt die Fähigkeiten vom Spieler und erhält noch die Möglichkeit dazu in 3 verschiedenen Komplexitätsgraden auf das Spiel zu reagieren.

Die **Klasse Board** verwaltet das Spielfeld, bietet die Möglichkeit dieses auf der Konsole zu visualisieren und kontrolliert, wenn es in Benutzung ist, seinen Zustand. Es kann erkennen welcher Spieler gewonnen hat oder ob es zu einem Unentschieden gekommen ist. Es wird vertikal, horizontal und diagonal geprüft.   
Zudem ist das Board dynamisch anpassbar und lässt sich in Verschiedenen Größen erzeugen und kann auf verschiedene Gewinnbedingungen initialisiert werden.   
Zusätzlich bietet das Board die Möglichkeit die Züge der Spieler auf ihre Gültigkeit zu kontrollieren.

Die letzte Klasse ist die **Klasse Game**. Sie fügt alle Module zusammen und ermöglicht durch das Nutzen der ihr zu Verfügung gestellten Klassen und deren Methoden einen Spielablauf.   
Dafür hat sie einmal eine Methode „start ()“, die vor Beginn des eigentlichen Spiels die notwendigen Informationen vom Benutzer erfragt. Hier werden die beiden Spieler initialisiert und falls sich unter diesen ein Bot befindet, sein Schwierigkeitsgrad vom Benutzer gewählt.  
Zusätzlich erstellt die Funktion ein Board-Objekt aufgrund der vom Benutzer gewählten Gewinnbedingung, Spalten- und Zeilenanzahl.  
Ist das erfolgreich geschehen, startet die Methode, die zweite Methode der Klasse Game, die „gameloop ()“.

Im Falle das zwei Spieler gegeneinander antreten oder ein Spieler gegen einen Bot, wird die Gameloop so lange laufen, bis eine der beiden Parteien gewonnen hat oder es zu einem Unentschieden gekommen ist.   
Dazu ruft sie einen der beiden Spieler auf und fordert diesen auf sein Zug zu machen. Wer beginnt, wird hierbei zufällig entschieden. Nach dem erfolgreichen Setzen ist der nächste Spieler dran. Ist die Runde beendet, wird dem Nutzer die Wahl gegeben erneut zu spielen oder das Spiel zu beenden.

Lässt der Benutzer zwei Bots gegeneinander spielen, kommt noch die Option hinzu, zu wählen wie häufig man die beiden gegeneinander antreten lassen möchte. Der Spielablauf bleibt sonst unverändert. Die Bots wählen ihre Züge aufgrund dem zu Beginn gewählten Schwierigkeitsgrad.

# KI-Konzept:

Es wurden 3 Versionen des Bots erstellt, die aufgrund von unterschiedlicher Priorisierung mit den gleichen Methoden unterschiedliche Spielmuster verfolgen. Dadurch ist zum einen ein defensiv ausgerichteter Bot entstanden, der erst schaut, ob er sich verteidigen muss. Wenn das an keiner Position notwendig ist, greift er an und setzt zu seinen Gunsten.  
Zum anderen ist xein offensiv ausgerichteter Bot entstanden, der zuerst schaut, ob es ihm möglich ist eine Reihe zu vervollständigen und wenn das an keiner Stelle sinnvoll ist, prüft, ob er verteidigen kann. Ist auch das an keiner Stelle notwendig wird er an eine bereits existierende Reihe anbauen.

Die Bots basieren auf 3 Methoden.  
 Die erste Methode ist für das Verteidigen verantwortlich (check\_in\_danger). Diese schaut, ob der Gegner horizontal, vertikal oder diagonal eine Reihe hat die mindestens zwei Elemente lang ist. Ist das gegeben, so berechnet sie für den spezifischen Fall eine Position, die den Gegner blockiert. Damit verhindert der Bot frühzeitig das E rreichen der Gewinnbedingung durch seinen Gegner.

Die zweite Methode ist das Angreifen (check\_finish). Diese ähnelt dem Verteidigen sehr, da sie die gleichen Methoden nutzt. Jedoch wird in der Methode statt wie beim Verteidigen die Nummer des Gegners hier die eigene Nummer übergeben. Die Methode schaut nun, ob der Bot selbst horizontal, vertikal oder diagonal eine Reihe hat, die mindestens zwei Elemente lang ist. Trifft dieser Fall ein so berechnet sie ein Feld, welches die Reihe verlängert und den Bot somit näher an den Sieg heranführt.

Die letzte Methode ist zum Start eines Angriffs des Bots gedacht (start\_attack). Sie schaut nur nach einem eigenen Feld und prüft dann, ob die Felder daneben frei sind. So macht der Bot in dieses seinen Zug und kann damit in der nächsten Runde einen Angriff starten.

Der defensive Bot schaut zuerst, ob er verteidigen kann (check\_in\_danger) und greift danach erst an (check\_finish). Der offensive Bot schaut zuerst ob das mittlere Feld frei ist, danach ob er angreifen kann (check\_finish), dann ob er verteidigen muss (check\_in\_danger) und wenn er dies auch nicht muss, so startet er einen Angriff (start\_attack).

Die Bots sind nach Schwierigkeit sortiert. Der Level 1 Bot setzt in ein zufällig gewähltes Feld. Er ist dadurch einfach zu schlagen. Der Level 2 Bot kann verteidigen und auch bei einer sich ergebenen Möglichkeit einen Angriff zu Ende bringen. Jedoch wird er selbst nie einen Angriff starten. Der Bot ist dadurch schwieriger zuschlagen, auch wenn er selbst selten gefährlich wird. Der Level 3 Bot wiederum ist der stärkste aller drei Bots, da er nicht nur passiv agiert, sondern auch aktiv angreifen kann. Er macht dies durch seine Befähigung zum Starten eines Angriffs er ist nun nicht mehr auf den Zufall angewiesen, um anzugreifen. Dies unterscheidet ihn vom Level 2 Bot.

# Data Science 1: Statistischer Vergleich der verschiedenen KIs

Die Statistiken zeigen das Gewinnverhalten der einzelnen Bots bei einem 5, 5, 4-Spiel mit 2000 Durchläufen, wobei der Anfangsspieler nach 1000 Durchläufen wechselt.

Der Level 1 Bot zeigt im Spiel gegen sich selbst ein ausgeglichenes Gewinnverhältnis mit einer geringen Anzahl an Unentschieden. Dieses ist durch die reine Zufälligkeit des Bots erwartbar gewesen. Gegen den Level 2 Bot ändert sich das Verhältnis deutlich es steigen zum einen die Anzahl der Unentschieden von 218 auf 282, zum anderen steigt die Anzahl der Niederlagen im Verhältnis zu den gespielten Spielen auf knapp 75%. Dies zeigt, dass der Level 2 Bot gegen den Level 1 Bot diesem deutlich überlegen ist, da er höhere Chancen hat zu gewinnen. Ähnlich sieht es im Spiel gegen den Level 3 Bot aus. Jedoch gibt es mit 29 Siegen und 22 Unentschieden hier kaum noch Möglichkeiten vom Level 3 bot nicht geschlagen zu werden. Dadurch wird der Level 3 Bot in seiner Stärke bestätigt, da er noch mal eine stärkere Siegesquote hat.

Der Level 2 Bot spielt gegen sich knapp doppelt so häufig unentschieden (1032), als er gewinnt (489) bzw. verliert (479). Hier zeigt sich erstmalig die Auslegung des Bots darauf das er hauptsächlich verteidigen soll. Dadurch wird hier häufig unentschieden gespielt, da sich beide Bots in ihren Spielzügen gut verteidigen. Gegen den Level 3 verliert er zwar in der Menge mehr, schafft aber auch hier noch eine große Anzahl an Unentschieden (627). Dies zeigt, dass der Bot auch gegen stärkere Gegner in der Lage ist sich zu verteidigen und ihnen hin und wieder auch gefährlich werden kann.

Der Level 3 Bot ist im Spiel gegen die anderen Bots der stärkste Bot. Gegen sich selbst ist er sehr ausgeglichen. Jedoch spielt er kaum unentschieden (80). Dies unterstreicht die Fähigkeit des Bots einen Sieg herbeizuführen. Deshalb bleiben die Unentschieden hier so gering.

Vergrößert man das Feld auf 20x20, lässt aber alle anderen Faktoren gleich so sieht man mehrere Veränderungen. Im Allgemeinen kann festgestellt werden, dass kaum noch unentschieden gespielt wird, da das Feld dafür zu groß wird.

Der Level 1 Bot verändert sich im Spiel gegen sich selbst kaum und bleibt bei seiner - durch den Zufallsbedingten - Ausgeglichenheit. Jedoch verliert er gegen den Level 2 Bot sowie den Level 3 Bot in fast bzw. allen Durchläufen. Das unterstreicht den Nachteil des Level 1 Bots.

Der Level 2 Bot ist im Spiel gegen sich selbst sehr ausgeglichen spielt aber nicht mehr Unentschieden. Das lässt sich dadurch erklären, dass dies das gleiche Ergebnis wie bei einem kleineren Feld ist die Chance für ein Unentschieden aber deutlich geringer ist. Gegen den Level 3 Bot ist es ihm nun kaum noch möglich einen Sieg dessen zu verhindern, da er mehr Fläche hat, der die er verteidigen muss. Deshalb verschiebt sich das Gewinnverhältnis hier nochmal stärker Richtung Bot 3.

Der Level 3 Bot ist im Spiel gegen die anderen Bots weiterhin der stärkste Bot. Gegen sich selbst wäre er sehr ausgeglichen. Jedoch wird hier nur ein Durchlauf mit 1000 Versuchen getestet, bei dem kein Wechsel stattfand. Dies zeigt erneut die Fähigkeit des Bots einen Sieg herbeizuführen. Deshalb gewinnt der beginnende Bot hier so extrem.

Zusammenfassend kann man sagen, dass ein größeres Feld die Unentschieden minimiert und die stärkeren Bots noch stärker macht, da der Faktor Zufall selten Spiel entscheidend ist. Hier zeigt sich die Stärke der stärkeren Bots, da diese gegen die schwächeren Bots nun deutlich häufiger gewinnen.

# Data Science 2: Gibt es einen Vorteil/Nachteil für den/die Startspieler\*in?

5,5,4-Spiel mit 1000mal Starten.

20,20,4-Spiel mit 1000mal Starten.

5,5,4-Spiel mit 1000mal Starten.

20,20,4-Spiel mit 1000mal Starten.

Man sieht an den vorangegangenen Statistiken, dass das Anfangen bei dem kleineren Feld zu einem Vorteil führt. Vorallem bei den schwächeren Bots, da diese dadurch einen Zug mehr haben und so eine höhere Wahrscheinlichkeit haben zufällig einen guten Zug zu machen. Jedoch kann man beobachten, dass dies für den Level 3 Bot im Spiel gegen sich selbst nicht gilt. Dieser gewinnt deutlich öfter, wenn er nicht startet. Auf dem großen Feld egalisiert sich das Anfangen, da hier die Wahrscheinlichkeit, um aus Zufall zu gewinnen schon so klein ist, dass der Faktor anfangen kaum ins Gewicht fällt. Jedoch sieht man, dass der Level 3 Bot im Spiel gegen sich selbst nun gewinnt, wenn er anfängt.

# Ausblick und Fazit

Fazit

Das Projekt ist vollständig vollendet. Es kann noch weitergehend ergänzt werden, diese Möglichkeiten werden aber im Ausblick behandelt. Das Projekt umfasst jetzt ein funktionierendes m,n,k-game, welches vollständig variabel ist. Dazu gibt es 3 verschieden schwere Bot-Level.

Ausblick