Project: Playing Stone Age

Uku Hannes Arismaa (GitHub: kiistata)
Kevin Kekki (GitHub: stargateprovider)
Joosep Näks (GitHub: joosu77)

Project repository:
https://github.com/stargateprovider/ids-stone-age-ai

1. Business understanding

Stone Age is a board game and we think it would work well in a digital format as well, which is why we have decided to develop an AI that would play the game.

Our goals for this project are the following:

1. Port the board game Stone Age to Python.
2. Generate the dataset of random playthroughs.
3. Train a model on the generated games that plays the game better than random choice models.

We hereby leave it open and unspecified as to whether the resulting product will be:

A. An open-source solution, which anyone can use when building their digital port of the board game Stone Age in the future.
B. Only a private project, which will most likely be abandoned after its evaluation by the 'grading team'.
C. Part of a fully complete "Stone Age: The board game port for the PC, Linux and Mac" for which we will charge an appropriate sum of money.

The above will depend on how well the team will be able to complete the project and how much interest we will have in developing it further. The result will most likely benefit us by allowing us to pass the course Introduction to Data Science (IDS a.k.a. LTAT.02.002) and might benefit anyone, who wishes to implement or play the game Stone Age on a computer. There will be no assessment of costs and benefits, because there will be no monetary costs involved.

Regardless of the type of end-product, we will consider the project to be successful if it helps us gain insight on the relevant data science methods and machine learning techniques, as well as allow us to complete the IDS course. The fulfillment of these criteria will be judged by the 'grading team'.

The project should involve the use of at least one appropriate data science or machine learning method for developing the AI, which was covered during the IDS course. There could be at least 2 levels of this AI: One which would make optimal decisions whenever possible and a second one, whose decisions would be more randomised. That way, the human players could potentially choose a difficulty setting when they play the game against the computer.

Requirements for the resulting product:

- An artificial intelligence written in Python, capable of making reasonable (better than random) choices in the board game Stone Age based on the appropriate input data.
- A text-based interface used to give and receive input from the main program, which runs the AI.
- Work ideally completed before 17.12.2020 and presented on that day sometime between 14:00 and 17:00.
- No security requirements or legal obligations, since they are not applicable.

Factors that could delay the completion of the project include, but are not limited to:

- Human time management mistakes
- Inability to meet too ambitious goals
- Long term worldwide power outage, which some have predicted to happen at the end of the year

Project resources:

Human resources:

- Team members (3)
- Anna Aljanaki (IDS course instructor, GitHub: aljanaki)
- Potentially other people from the IDS course, who form the 'grading team'

Software:

- Python 3 as the main developing environment
- Python modules such as Pandas, Numpy and Sklearn (possibly more)
- Jupyter notebook for easily working with the dataset
- Git and GitHub for code collaboration purposes

Information:

- About Stone Age: https://boardgamegeek.com/boardgame/34635/stone-age
- A basic Stone Age implementation in Python, to potentially study and build upon: https://github.com/TheGameBotYT/StoneAgeAI

Our **data-mining goals** are

1. Create a dataset by letting the program play through the game multiple times making random choices.
2. Let a model learn from the random playthroughs and find a better strategy.

The **data-mining success criteria** would therefore be a model, which can make choices based on the current game state. These choices should result in a higher score in the game than random choices.

Project terminology:

Game - The board game Stone Age
Fields - fields on the game board on which players place workers aka meeples
Resources - items that workers can get from certain fields and with which players can buy cards that give points. Unlike the original game the total number resources is only bounded by the size of your RAM
Values of resources - wood is worth 3, clay 4, stone 5 and gold 6 units. This is the amount of points you gain for buying a building using that resource, you don't get any points if you have the raw resource at the end of the game
Cards aka *buildings* - on the gameboard there are 4 stacks of cards of which players can only see the top ones. They can buy those cards for resources to earn points. The game ends when one of these stacks is emptied
Production level aka *agricultural level* - for each production level the player has to feed one less worker without losing points
Worker aka *meeple* - pieces players have that they can place on different fields on the board to show they want to get a resource or buy a card

Our simplified **rules for the game** can be found in the GitHub readme file. They were too long (resulted in over 800 words) to fit here.

2. Data understanding

Data for our project is generated by us. To get the data we wrote a simplified model of the game (available in the repository) and made the computer play through the game a lot of times by making random choices.

Each row of the dataframe represents one move made by the model including the game state before the move was made and the score with which that game ended. The dataframe we are using has 80 columns:

- 11*3=33 columns for each combination of a player and a field on the gameboard, showing how many meeples that player has on that field.
- 5*3=15 columns for each combination of a player and a resource, showing how much of that resource that player owns.
- for each player a maxMeeples column, showing the total number of meeples that player owns at that moment.
- for each player a production field, showing the agriculture level of each player.
- a points field for each player, showing the amount of points that player owns at that moment. Then there are 16 columns for the top cards, showing the cost of the top card of each card stack for each resource, and 4 columns for the heights of those stacks. Then there are 2 columns showing the move that was made, the first one shows which field meeples were placed on and the second one shows how many meeples were placed on that field. Lastly there is a fitness column which shows how well the player did in that game overall, for now it is the number of points that player got divided by the average number of points in that game.

The main goal is to create a bot that can make moves with as high fitness rating as possible, so I'm looking for correlations between the fitness column and other columns. The highest correlation is with fitness is the running points of player 0 (the player who is making the move) which makes sense since with higher points you are more likely to win. Fitness also has high absolute correlation with other players' points, which was also to be expected since the fitness is dependent upon the average of all players' end scores. But either of these doesn't help with making decisions. The first useful ones are the agricultural level and food amount and other players' agricultural levels. This is logical since most random plays end with very negative scores because of starving so we know the model needs to produce food or up it's agricultural level.

We can verify that all the data we need exists, because we (will) have generated it ourselves to fit our needs for this project.

3. Project plan

1. Business understanding: Kevin 1h Uku 1h
Based on the CRISP-DM model and using our critical thinking we write a document (report) with our understanding of the project.

2. Generating data: Joosep 8h
Our produced version of the game will be used to generate the data (game states).

3. Data understanding: Joosep 2h
Once we have generated the data, we write a document with our understanding of it based on the CRISP-DM model.

4. Data preparation: Joosep 1h
Once we have evaluated the data, we will change and reformat it as we see fit. Output: New method for generating data or reformatted old data

It is difficult to predict how much preparation the data will need, but we generate it ourselves, so changes can be made easily.

4. Modeling: Joosep 20h Kevin 25h Uku 25h
Input: Data
Output: Model
Deadline: 16.12

5.Presenting: Uku 5h
Input: Model
Output: Presentation
Deadline: 17.12