Aims and Objectives

Problem Space:

- Chess has a vast number of possible positions and variations, which makes it computationally expensive
- Minimax with alpha-beta pruning is usually used to limit the size of the game tree, but it is still quite inefficient
- Using a Naïve Bayes Classifier could help improve this.

Motivation:

Chess is still an unsolved problem, making the "best" decision could be improved through the use of machine learning models.

Scope:

Implement a minimax algorithm to evaluate chess positions to make a move then integrate a Naïve Bayes Classifier to optimise it and test this performance against Stockfish to benchmark improvements.

Deliverables

1) **Research** – Report summarising principles of MiniMax, Alpha-Beta pruning and Naïve Bayes

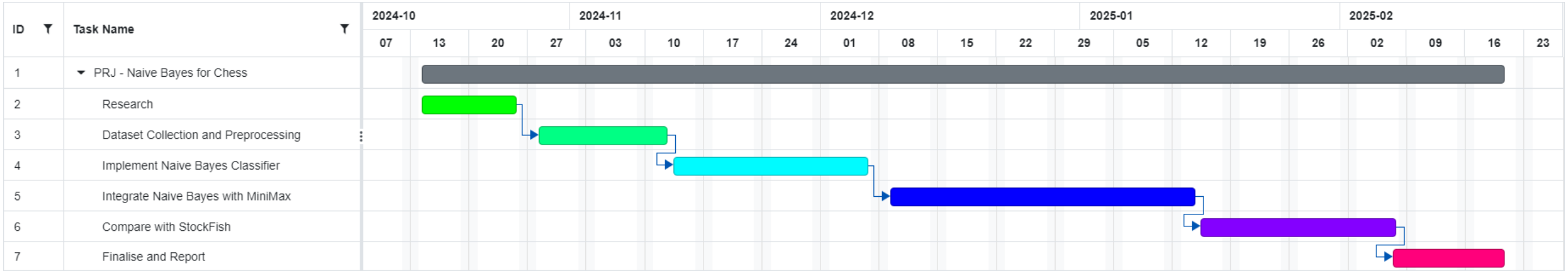
2) **Dataset** – A chess dataset that is prepared for training

3) **Naïve Bayes** – A trained Naïve Bayes classifier that can evaluate chess positions.

4) **Minimax** – Minimax algorithm that uses the classifier to evaluate positions.

5) **Stockfish** – Report summarising the performance of the new algorithm with Stockfish

6) **Final Report** – Report summarising the project and preparing the presentation



Software Platform

Language: Python

Libraries:

- scikit-learn for Naïve Bayes

- python-chess for representing chess positions

- Stockfish for benchmarking

Operating System: Windows

Bibliography:

Scikit-learn documentation: https://scikit-learn.org/1.5/modules/naive_bayes.html

AI Chess Algorithms: https://www.cs.cornell.edu/boom/2004sp/ProjectArch/Chess/algorithms.html#:~:text=The%20core%20of%20the%20chess,max%20search%20of%20the%20gamespace.&text=(or%20%22ply%22%20as%20it's,leafs%20of%20the%20search%20tree.