Game AI: Assignment 1

Ajay Kumar Malik (am21932@essex.ac.uk)

This project deals with the creation and analysis of a new bot for the game “The Resistance”. The created bot used decision tree classifier to learn and make an informed decision while choosing a team when the bot becomes a leader. As the bot tries to create the best possible team when leader, the strategy can be named **Horses for courses**.

With this strategy the bot can try and improve the win percentage over numerous consecutive games.

1. Introduction:

The objective of the project is to create an agent that can choose a team which has a higher chance of winning on a mission.

The motivation of trying to create such a bot comes from the real-world professional games wherein the opponents’ study and analyse the performance of individuals and strategize accordingly. The analysis is also required for every new player that joins the team. Similarly, for new entrant the bot will have to study the player first and then from his experience decide if the bot makes a good player to be had on the team.

2. The Resistance:

2.1 Game Description

The Resistance is a game that is played between 5 players at a given time. It was designed by Don Eskridge. The players are divided into two groups, one group of 3 representing the resistance and the other group of 2 playing as spies. The game is based on the fictional storyline of a war between the government and the resistance group. The government being aware of the case tries to infiltrate and sabotage the resistance’s efforts.

The spies are aware of each other, but the resistance is totally unaware.

2.2 Rules

The Resistance game is played out in 5 turns. Each turn involves a set number of players that can be on a particular mission. The first one has 2, second has 3, third 2, and fourth and fifth have 3 players on a mission.

Resistance group wins on 3 successful mission completions. If spies can incur 3 losses to the resistance group, then the spies win. Also, it counts as a win for the spies after 5 consecutive mission rejections.

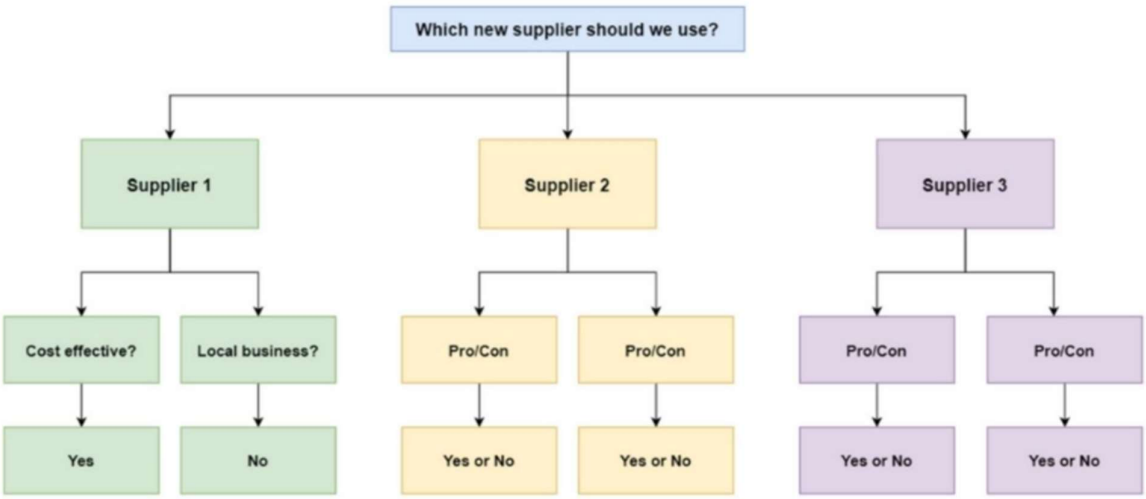
A new leader is chosen on every turn and the leader assembles a team that will make an attempt to win the mission. Other players cast a vote and decide if the selected team should go on a mission or if a new leader is to be chosen. If the majority votes are in favour of the chosen team to go on the mission, then the selected team plays. The chosen players then cast a secret vote to decide the fate of the mission. The resistance players vote for the success of the mission and the spies have an option whether to pass or fail the mission. If not sabotaged by the spies, the resistance group wins the mission else it counts as a win for the spies.

3. Background:

The agent, JrPluto in our case first observes the other players and then figures out who performs better for any given mission. After the observation a decision tree is built to help figure out which player should JrPluto select in his team when he becomes a leader.

3.1 Decision Tree

Decision trees help us to display a given dataset in the form of a tree. The dataset is usually separated into predictors and target. The predictors are the attributes that the act as an input to the decision tree and target is the output that our decision tree should provide us with.



Decision Tree Example [1]

For this assignment, the decision tree helps our bot to figure out which other player to choose from.

4. Techniques Implemented

The bot uses decision tree to analyse and to choose from the pool of the other players to choose them for a mission led by it.

4.1 Scikit-learn decision tree

The required decision tree for the bot is created using the scikit-learn python library [2]. DecisionTreeClassifier is used which after reading the data suggests the best possible player to be had on the team.

4.2 Data Collection

The bot generates the data that is required to generate the decision tree. The data is stored in a csv file and once the data generation is complete, the file is divided into two – BotsData-train.csv and BotsData-test.csv. As suggested by the names, the BotsData-train.csv file is used to train the decision tree and the BotsData-test.csv if used to test the created Decision Tree.

Each column in the file is as follows: -

- Turn – Represents the turn that is in progression, it can be an integer between 1 and 5
- IsLogicaltonSuspect – Boolean value representing if Logicalton is a suspect before we start the mission.
- IsTrickertonSuspect – Boolean value representing if Trickerton is a suspect before we start the mission.
- IsBounderSuspect – Boolean value representing if Bounder is a suspect before the start of the mission.
- IsSimpletonSuspect – Yes/No representing if Simpeton is a suspect.
- Player – The player to be selected in our team for the mission.

Turn	IsLogicaltonSuspect	IsTrickertonSuspect	IsBounderSuspect	IsSimpletonSuspect	Player
1	No	No	No	No	Trickerton
2	No	No	No	Yes	Logicalton
3	Yes	No	Yes	Yes	Trickerton
1	No	No	No	No	Trickerton
2	No	No	No	No	Logicalton
3	Yes	Yes	No	Yes	Bounder
4	Yes	Yes	Yes	Yes	Logicalton
5	Yes	Yes	Yes	Yes	Bounder
1	No	No	No	No	Trickerton
2	No	No	No	No	Logicalton
3	Yes	No	Yes	Yes	Trickerton
4	Yes	No	Yes	Yes	Trickerton
1	No	No	No	No	Trickerton
2	No	No	No	No	Logicalton
2	No	No	No	No	Logicalton
3	No	No	No	No	Bounder

Table 1 – Example Data Set

Predictors – Turn, IsLogicaltonSuspect, IsTrickertonSuspect, IsBounderSuspect and IsSimpletonSuspect.

Target – Player

4.3 Learning Process

The data in the Table 1 suggests which player should be selected for a mission. To reach this inference, the bot had first observed itself and the other bots over a span of

10000 games and then figured which player has a better chance of winning any given round.

The number of missions won by the bots is as shown: -

Player	Mission 1	Mission 2	Mission 3	Mission 4	Mission 5
Logicalton	1466	1261	1727	1896	1063
Trickerton	2358	1253	1750	1706	829
Bounder	1463	1128	1761	1825	1097
JrPluto	1544	876	1490	1363	760
Simpleton	1485	1038	1584	1574	922

Table – 2

From table 2 it can be observed that Trickerton stands out in the first mission of any game but performs rather poorly in the last mission. Trickerton is just an average player in the rest of the missions. Logicalton and Bounder are the ones that maintain their form throughout the game. Our bot, JrPluto loses its sheen in the second and the last round.

Table 2 helps our decision tree to suggest which bot has the higher chance of helping us win the round.

4.4 Team Selection

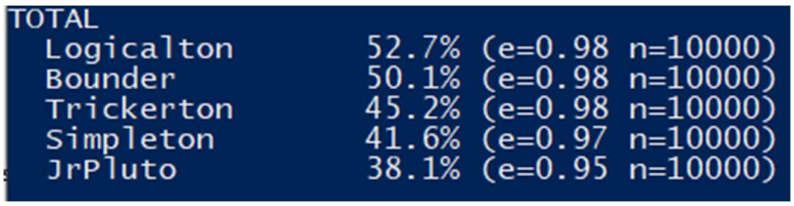
When selecting a team, the agent always selects itself and one or two other players depending on how many players can go for a mission on a particular round. If it is a 2-player mission, then the decision tree suggests the possible player to have in the team and if it is 3-player mission, then one player is chosen as per the decision trees suggestion and the other one is chosen at random.

Decision tree takes into consideration if the player that it suggests is a spy and also what mission number, we are in. In case all of the available bots are suspects, then the player is solely chosen on the basis of the mission number that will be played.

5. Experimental Study:

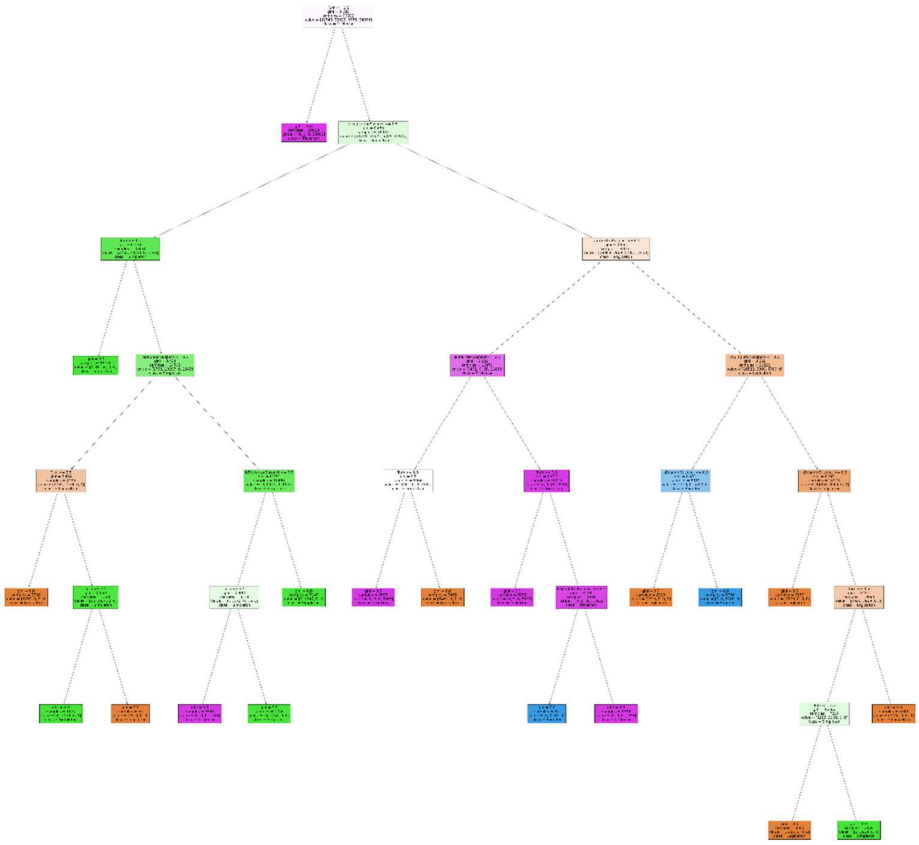
Our bot JrPluto was tested against the other bots in the intermediates.py. JrPluto acted like an observer for the first 10000 games to see who plays better in which round (as represented in the table 2).

While being an observer, JrPluto did not achieve a lot of success and was the worst ranked player as shown: -

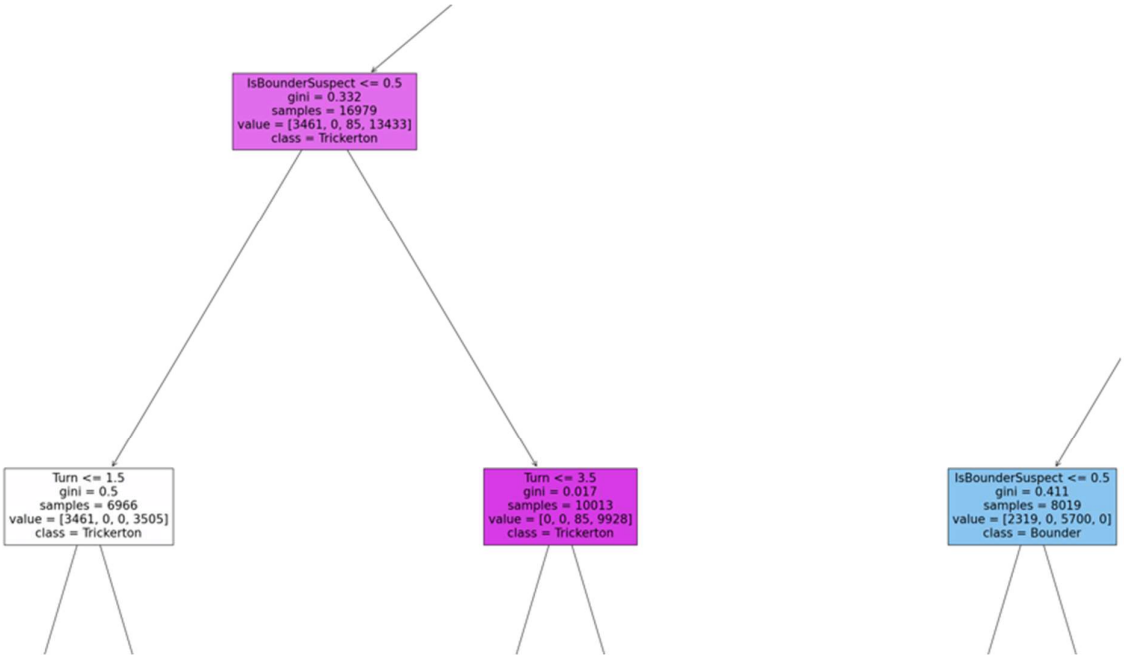


After observation, a data file was created to create a decision tree that would suggest a teammate for the round when JrPluto acts as a leader. This file was divided into two different files, one acting as a training file and other as a testing file.

The decision tree that was generated using the data is as shown: -



A zoomed in subsection of the decision tree is shown below: -



The model comes out with a 100 percent accuracy.

5.1 Performance

As the bot can only choose the players in one game, that too if he becomes the leader before the game finishes, the strategy does not show a huge amount of success in the numbers generated. It only improves by a rank and becomes a better player than Simpleton achieving 41.1 percentage of success as shown: -

```
TOTAL
  Logicalton        51.0% (e=0.98 n=10000)
  Bounder           48.6% (e=0.98 n=10000)
  Trickerton        45.0% (e=0.97 n=10000)
  JrPluto           41.1% (e=0.96 n=10000)
  Simpleton         40.2% (e=0.96 n=10000)
```

It is also observed that JrPluto's increase in winning percentage has impacted other bots' numbers. This confirms that JrPluto is now making better choices as a leader.

6. Analysis:

The results show that JrPluto has improved in its performance as a direct outcome of making informed decisions when choosing team. The game when played at intermediate level (that is when played with intermediate.py) is dominated by spies. The spies regularly

outsmart the resistance group. So just choosing a better team will not really give a significant boost to JrPluto's performance.

7. Overall conclusions:

Using decision tree classifier has helped in improving the win percentage of JrPluto, albeit not much as JrPluto at max gets to choose players only once in a game.

The drawback of the strategy that has been used is that it observes and documents the already existing players in the game. This strategy will not work if a new player enters the game. There is a scope of improvement in how the strategy has been implemented. If we can cater for the new entrants, then this strategy becomes far more useful.

Also just using one single strategy is not enough to improve the performance of an agent significantly as other bots show better intelligence with their game. If we can make JrPluto better at predicting spies we can further improve the performance.

8. References

1. Image taken from https://www.ayoa.com/templates/decision-tree-template/.
2. Scikit Learn - https://scikit-learn.org/stable/