The Resistance Agent Project

CITS3001 Algorithms, Agents and Artificial Intelligence Project

Shane Monck, 22501734

The Resistance and My Agent

The Resistance is a social deduction game with secret identities. Players are either members of the Resistance attempting to overthrow the government, or Spies trying to thwart the Resistance. The Resistance wins the game if three Missions are completed successfully; the Spies win if three Missions fail. The Resistance members have no information about anyone at the beginning of the game, while the Spies know who is or isn't their fellow Spies. My goal for this project was to build a Resistant agent that uses rational choices and uses all the information gathered from the different stages of the game to predict who the Spies could potentially be leading to more victories for the Resistance over the Spies, reflecting similarities to how a real-life game would be played where players would go from what they've seen occur and then predict who they believe is the most likely be a Spy. The Agent developed was made with logical game playing policies and uses a Gaussian Naïve Bayes Classifier for its prediction. In games of 10 members and running 10,000 different games my agent was able to predict whether other players were Spies or Resistance members with an average of approx. 88%, winning approx. 91% of these games.

Game Playing Policies the Resistance and Testing Spies Followed

The Resistance

Team Building Phase (proposing mission teams and voting):

If Team Leader:

- Always picks itself to be on the team.
- If first mission picks randomly players to join mission.
- If the Naïve Bayes Classifier has predicted all the potential spies, select those that are predicted not to be Spies on the team first.
- Picks members that are the most trusting these are chosen by random, as long as they are not predicted to be a Spy or if they are not a Spy that has outed themselves. They are first picked from the pool that has completed successful missions, if team is not full, they are then picked from a pool of players that have not failed any missions, if team is still not full members are picked by random ignoring prior information except if they are an outed Spy or not.
- Always votes yes for mission that it proposed

Not Team Leader:

- Votes yes for first mission if on the mission otherwise no
- Votes no if mission contains an outed Spy or a predicted Spy
- If no predicted or outed Spies information:
    - Votes yes if mission contains all players that have completed successful missions
    - Votes no if mission contains players that have failed missions
    - Votes yes if mission has some successful players and zero failed mission members
    - If mission contains zero successful and zero unsuccessful:
        - Votes yes if on mission
        - Votes no if not on mission

The Spies

Team Building Phase (proposing mission teams and voting):

If Team Leader:

- If not mission 4 put self on team with other resistance members only
- If mission 4 and only have 1 win so far (meaning must win mission 4 and 5) and betrayals required is 2 add itself and another spy on mission with the rest Resistance. Otherwise add just itself to mission with other Resistance members.
- Always votes yes if proposed mission

Not Team Leader:
- If mission 1 vote yes
- If not mission 1 and not mission 4:
    - Votes yes if mission contains 1 or more Spies
    - Votes no if mission contains all Spies
- If mission 4 and requires a win and two betrayals:
    - Votes yes if mission has 2 or more Spies
    - Otherwise votes no
- If mission 4 and does not require a win vote yes to any team

Mission (choosing to betrayal or not to betrayal)
- If mission 1, do not betrayal
- If mission 5, do betrayal
- If not mission 4:
    - If only spy on the team, do betrayal
    - If two spies on team 75% chance of betrayal
    - More than two spies 50% chance of betrayal
- If mission 4 and need to win with 2 betrayals
    - If only two Spies on mission, do betrayal
    - If more than two Spies 50% of betrayal
    - Else if only spy on team don't bother betraying

To summarise, the policies are based off how a real-life game would be played by people that do not know any impressive tactics just the basics. The Resistance vote and propose missions from what they have previously seen other players do and the mission outcomes, and the Spies do their best to get themselves on missions and betray whenever it is possible while doing their best to not out themselves or any other Spies. With these policies it can be easy for the Resistance to pick up patterns in the voting or mission outcomes which can be used to distinguish between the Resistance and the Spies this is when the Naïve Bayes Classifier works its best in predicting the Spies.

Naïve Bayes Classifier

Naïve Bayes is a classification algorithm that is based on Bayes theorem. Bayes theorem is a mathematical formula for determining conditional probability, for example Bayes theorem says that if event B has occurred, then we can find the probability of event A given B.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Where:
P(A|B) – The probability of event A given B (posterior)
P(B|A) – The probability of event B given A (likelihood)
P(A) – The probability of event A (prior)
P(B) – The Probability of event B (evidence)

A Naïve Bayes approach uses the assumption that each feature in the input vector is conditionally independent of other features.
Mathematically: $P(A|B,C) = P(A|C)$ which means that A is conditionally independent of B given C.

Ultimately the probability formula using conditional independence is

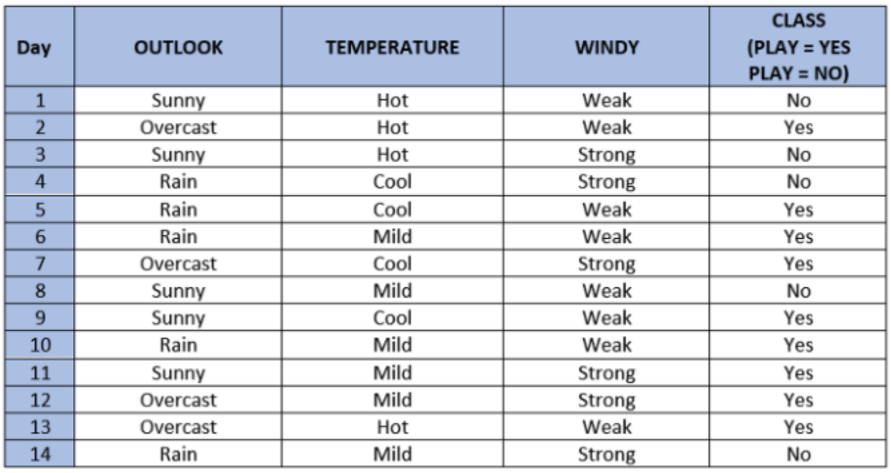
$$P(Y|X_1, X_2, \dots, X_d) = \frac{P(X_1|Y)P(X_2|Y)\dots P(X_d|Y)P(Y)}{P(X_1, X_2, \dots, X_d)}$$

So if we have two classes we can get the probability of the class Y for class 1 P(Y=1|X) and class 0 P(Y=0|X) given a query point (X).

Example: (All Images and material in Example are courtesy of: Vaibhav Jayaswal https://towardsdatascience.com/understanding-na%C3%AFve-bayes-algorithm-f9816f6f74c0)

Predicting whether conditions are appropriate for playing tennis or not

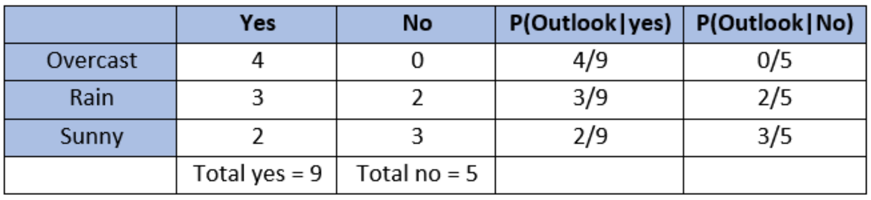
X = [outlook, temperature, windy], Y = [Yes, No]



Day	OUTLOOK	TEMPERATURE	WINDY	CLASS (PLAY = YES PLAY = NO)
1	Sunny	Hot	Weak	No
2	Overcast	Hot	Weak	Yes
3	Sunny	Hot	Strong	No
4	Rain	Cool	Strong	No
5	Rain	Cool	Weak	Yes
6	Rain	Mild	Weak	Yes
7	Overcast	Cool	Strong	Yes
8	Sunny	Mild	Weak	No
9	Sunny	Cool	Weak	Yes
10	Rain	Mild	Weak	Yes
11	Sunny	Mild	Strong	Yes
12	Overcast	Mild	Strong	Yes
13	Overcast	Hot	Weak	Yes
14	Rain	Mild	Strong	No

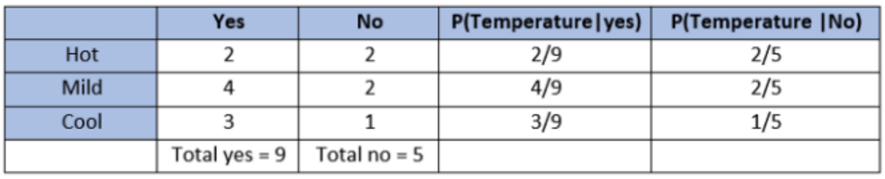
Calculating the likelihood tables:

Table 1: Outlook P(Outlook|class) - where outlook can be overcast, rain and sunny and class can be yes and no



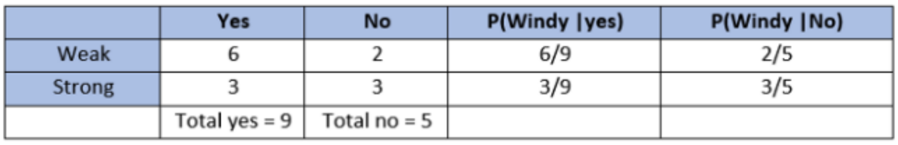
	Yes	No	P(Outlook|yes)	P(Outlook|No)
Overcast	4	0	4/9	0/5
Rain	3	2	3/9	2/5
Sunny	2	3	2/9	3/5
	Total yes = 9	Total no = 5		

Table 2: Temperature P(Temperature|class)- where temperature can be hot, mild, and cool



	Yes	No	P(Temperature|yes)	P(Temperature |No)
Hot	2	2	2/9	2/5
Mild	4	2	4/9	2/5
Cool	3	1	3/9	1/5
	Total yes = 9	Total no = 5		

Table 3: Windy P (Windy| class) where Windy can be strong and weak



	Yes	No	P(Windy |yes)	P(Windy |No)
Weak	6	2	6/9	2/5
Strong	3	3	3/9	3/5
	Total yes = 9	Total no = 5		

Priors are P(class=Yes) = 9/14 & P(class=No) = 5/14

So, given a query point X' = [sunny, cool, strong] can we play tennis?

First, we calculate P(class=Yes|X') and P(class=No|X')

$$P(\text{class = Yes}|X') = \frac{P(\text{outlook = sunny}|\text{class = yes}) * P(\text{temperature = cool}|\text{class = yes}) * P(\text{windy = strong}|\text{class = yes}) * P(\text{class = yes})}{P(X')}$$

$$P(\text{class = No}|X') = \frac{P(\text{outlook = sunny}|\text{class = No}) * P(\text{temperature = cool}|\text{class = No}) * P(\text{windy = strong}|\text{class = No}) * P(\text{class = No})}{P(X')}$$

P(X') =[P(class = yes)* P(outlook=sunny|class=yes)* P(temperature=cool|class=yes)* P(windy=strong|class=yes)] +

[P(class = no)* P(outlook=sunny|class=no)* P(temperature=cool|class=no)* P(windy=strong|class=no)]

From the likelihood table, we get the values required in the formula

$$P(X') = \left(\frac{9}{14} * \frac{2}{9} * \frac{3}{9} * \frac{3}{9}\right) + \left(\frac{5}{14} * \frac{3}{5} * \frac{1}{5} * \frac{3}{5}\right) = 0.0408$$

So then:

$$P(class = Yes|X') = \frac{\left(\frac{9}{14} * \frac{2}{9} * \frac{3}{9} * \frac{3}{9}\right)}{0.0408} = 0.387$$

$$P(class = No|X') = \frac{\left(\frac{5}{14} * \frac{3}{5} * \frac{1}{5} * \frac{3}{5}\right)}{0.0408} 0.613$$

Probability of No is higher than Yes so therefore conditions are not appropriate to play Tennis.

My Naïve Bayes Classifier

My approach to using the Naïve Bayes Classifier was to run multiple games without the Naïve Bayes Classifier to collect the training data needed to run the Agent that uses the Classifier. Ten variables are recorded throughout the game each with different weights based on how much I believe the variable relates to an action a Spy would do, as well as some variables are factored by what mission it is, as Spies are more likely to attack in latter missions.
The ten variables are: player voted for a failed mission, player went on a failed mission, player proposed a team that failed, player voted against a team that was successful on their mission, player voted against a team proposal, player voted against a team that contains members who has been on successful missions, player voted for a team that has been on failed missions, player voted for a mission they are not on, player proposed a team that has unsuccessful members, and player proposed a team that hasn't been on any missions.

The Naïve Bayes the agent uses is also Gaussian meaning we only need to calculate the mean and standard deviation of each input variable for each class value. Therefore, when making predictions the mean and standard deviation can be used to find the Gaussian Probability Density Function and in return it will provide the estimate, we need in determining whether other players are Resistance or Spies.

$$mean(x) = \frac{1}{n} * sum(x)$$

$$std(x) = \sqrt{\frac{1}{n} * sum(xi - mean(x))^2)}$$

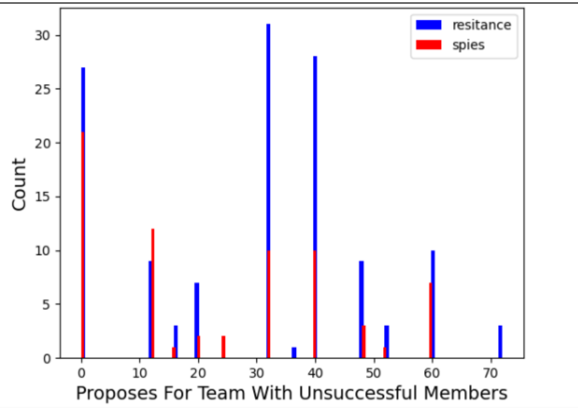
Where: n is the number of instances, xi is a specific value of the x variable for the ith instance.

$$Gaussian\ PDF = (\frac{1}{\sqrt{2 * \pi * std}} * \exp\left(-\left(\frac{x - mean^2}{2 * std^2}\right)\right))$$

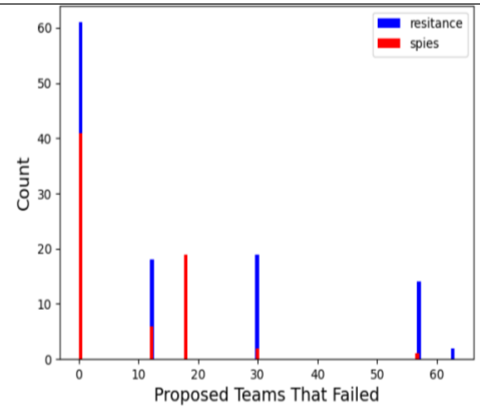
Where: mean and std are the mean and standard deviation calculated above, exp is the numerical constant e and x is the input value for the input variable.

The graphs below show the distinction between The Resistant and Spy Members, it is also the training data set used for predicting the Spies in the Naïve Bayes Classifier. The training data comes from the data collected from 40, 5 player games where the win percentage was 7.5% for the Resistance. Through trial and error of using different win percentages, size of player games and how many games, this turned out to be the best training data that gave the highest win percentage and predictions when used for the Naïve Bayes Classifier.
The mean and standard deviation for that variable in the training data set are also featured in the table below.

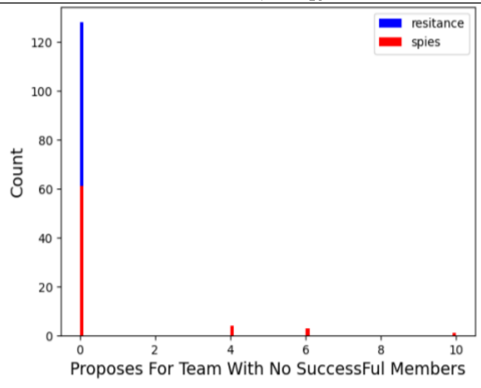
{1: Spy, 0: Resistance}



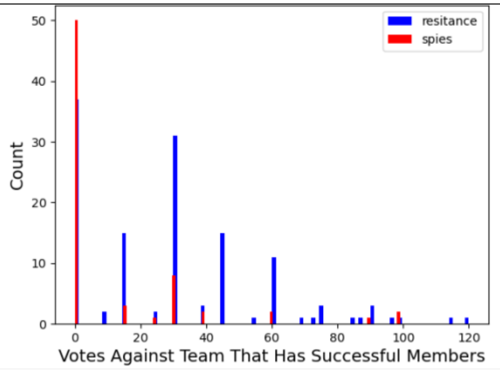
Mean, Stdev, N: {1: [22.957, 20.519, 69], 0: [29.374, 19.626, 131]}



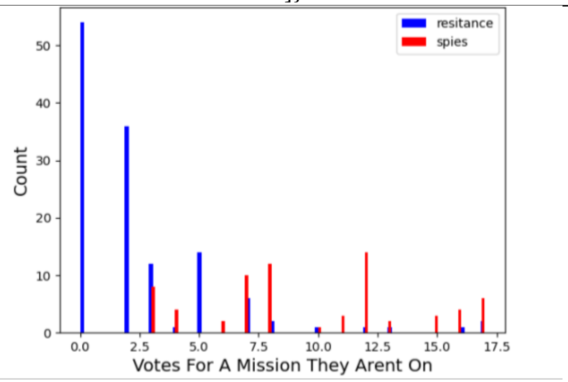
Mean, Stdev, N: {1: [7.696, 10.835, 69], 0: [15.390, 19.135, 131]}



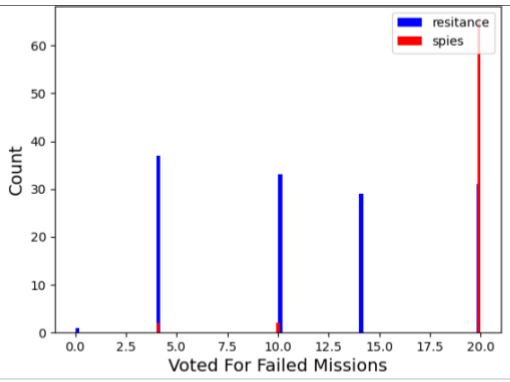
Mean, Stdev, N: {1: [0.638, 1.894, 69], 0: [0.137, 0.998, 131]}



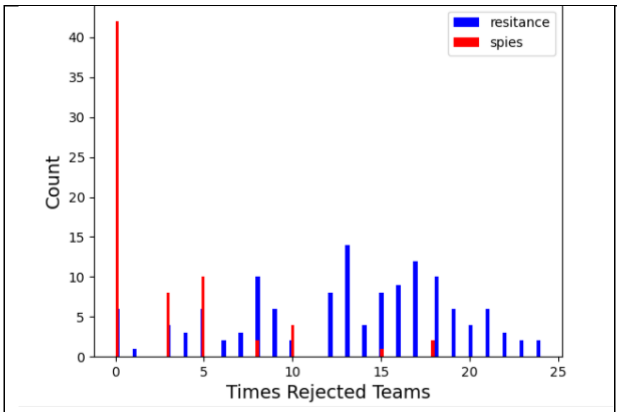
Mean, Stdev, N {1: [11.522, 23.400, 69], 0: [30.252, 28.054, 131]}



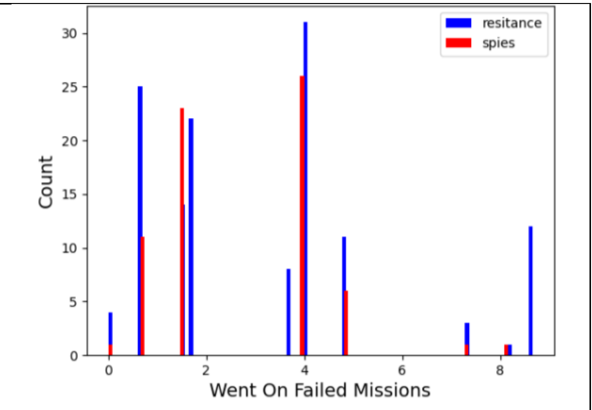
Mean, Stdev, N: {1: [9.652, 4.356, 69], 0: [2.481, 3.375, 131]}



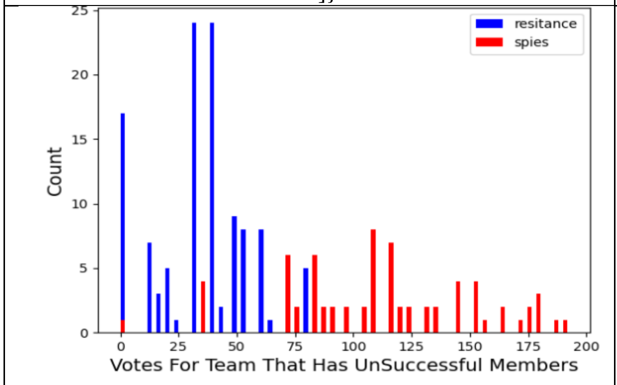
Mean, Stdev, N: {1: [19.246, 3.146, 69], 0: [11.481, 6.018, 131]}



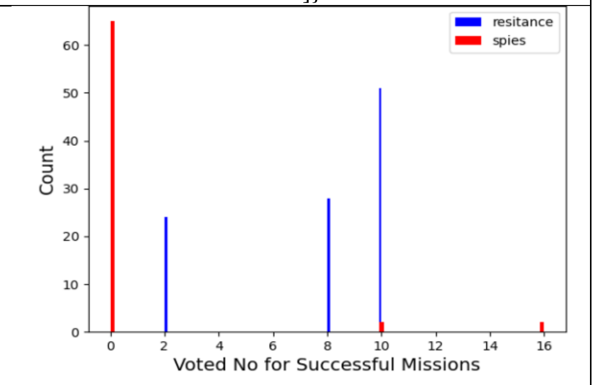
Mean, Stdev, N: {1: [2.623, 4.284, 69], 0: [13.008, 6.078, 131]}



Mean, Stdev, N:{1:[2.758, 1.750, 69], 0: [3.168, 2.448, 131]}



Mean, Stdev, N: {1: [113.101, 40.826, 69], 0: [45.465, 39.458, 131]}



Mean, Stdev, N: {1: [0.754, 3.146, 69], 0: [5.969, 4.2246, 131]}

An example prediction using the Gaussian Naïve Bayes Classifier pulled from a game scenario:
Player data array = [10, 0.5, 12, 0, 0, 0, 252, 26, 0, 4]
For the first value in the array(Voted For Failed Missions) the Gaussian probability would be

$$Gaussian\ PDF(Spy) = \left(\frac{1}{\sqrt{2*\pi}*3.146} * \exp\left(-\left(\frac{(10+1)-19.246^2}{2*3.146^2}\right)\right)\right)$$
$$= 9.74*10\text{^}6$$

$$Gaussian\ PDF(Resistance) = \left(\frac{1}{\sqrt{2*\pi}*6.018} * \exp\left(-\left(\frac{(10+1)-11.481^2}{2*6.018^2}\right)\right)\right)$$
$$= 0.351$$

The total Gaussian PDF would be the multiple of all Gaussian PDF for each column in the array (probabilities[column] *= gaussian_probability(row[i]), which is:

Calculated Gaussian PDF = {1: 3.0664445683456O3e-22, 0: 8.190061280260046e-36}

Therefore, probability of 1 is greater than probability of it being 0, so player is predicted to be a Spy. Note: Laplace Smoothing is also featured in the Naïve Bayes, this is simply adding 1 to each column so that values of 0 does not affect the calculation by multiplying 0 to the probabilities, Laplace Smoothing lead to about a 10% increase in wins for the resistance for the smaller games where the values are more frequently 0.

How Well Does It Perform?

The biggest question is: was there any improvement after the Naïve Bayes Classifier is added? And the answer is absolutely.

With 10,000 games simulated:

Number of Players	Win % Before Bayes	Win % After Bayes	Win Percent Difference	Average Prediction Accuracy
5	22.75%	59.14%	↑ 36.36%	78.23%
6	44.03%	82.57%	↑ 38.54%	84.91%
7	13.1%	40.03%	↑ 26.93%	80.62%
8	15.3%	66.30%	↑ 51%	88.83%
9	21.45%	71.45%	↑ 50%	88.42%
10	16.92%	90.75%	↑ 73.83%	88.37%

Running 10,000 games before and after the Classifier was added, saw an average win percentage gain of 46.11%. With 10 player games getting the most benefit out of the Classifier, especially since it now wins 91% of its games which is utterly impressive.
Predictions of whether players are Spies or Resistance is done accurately at an average of 84.90%, which is done surprisingly well.
The algorithm performs with a complexity of O(d*c) where d is the vectors dimension and c is the total classes, which in our situation is O(10*2) = O(20). This is also the Space complexity. Therefore, the algorithm runs efficiently and does not take much space.

Consequently, my goal for this project was reached, my Resistance agents use the in-game information collected throughout the missions and voting to accurately predict who the spies are with high accuracy leading to a high number of wins for the Resistance.

Scalability

Unfortunately, this Naïve Bayes Classifier approach isn't greatly scalable. Since the agent requires prior training data to collect the mean and standard deviation used in the Gaussian probability density calculations it will only perform well against the agent that produced this training data, essentially it will only perform well against the agents that have exact same or almost the same game playing policies, otherwise if there is a spy that performs different to what's anticipated in the training data, it can easily fool the Classifier and win more games for the Spies. It's with this reason I believe my agent will not perform greatly in the tournament. There will be moments where it will predict who the Spies are, but I picture this being a rare occasion since I predict most Spy agents made by others will be quite good at throwing off my Classifier. But nonetheless I reached my goal for the project and I'm eager to see how well it performs in the tournament, as even with just the policies implemented it should perform as a good teammate for others.
In the future I hope to scale my agent by still using the Naïve Bayes Classifier, as I was surprised on how well it performs, but also collecting more training data for the Naïve Bayes Classifier to use for different game playing agents.
Another way I imagine the agent could be scaled is being hosted on a web server and playing multiple games with the same people and recording the data over the courses of these games, again similar in the idea of my current real life playing agent, but now the idea is how over time you learn and record the strategies of the people you play with and can eventually predict whether they are the spy or not depending on frequent behaviours from them.

References

Analytics Vidhya (2019). *6 Easy Steps to Learn Naive Bayes Algorithm (with code in Python)*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/ [Accessed 3 Oct. 2021].

Brownlee, J. (2016). *Naive Bayes for Machine Learning*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/naive-bayes-for-machine-learning/ [Accessed 3 Oct. 2021].

Brownlee, J. (2019). *Naive Bayes Classifier From Scratch in Python*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/ [Accessed 11 Oct. 2021].

Hayes, A. (2020). *Bayes' Theorem*. [online] Investopedia. Available at: https://www.investopedia.com/terms/b/bayes-theorem.asp [Accessed 3 Oct. 2021].

Jayaswal, V. (2020a). *Laplace smoothing in Naïve Bayes algorithm*. [online] Towards Data Science. Available at: https://towardsdatascience.com/laplace-smoothing-in-na%C3%AFve-bayes-algorithm-9c237a8bdece [Accessed 17 Oct. 2021].

Jayaswal, V. (2020b). *Understanding Naïve Bayes Algorithm*. [online] Towards Data Science. Available at: https://towardsdatascience.com/understanding-na%C3%AFve-bayes-algorithm-f9816f6f74c0 [Accessed 10 Oct. 2021].

S, Y. (2020). *An Introduction to Naïve Bayes Classifier*. [online] Medium. Available at: https://towardsdatascience.com/introduction-to-na%C3%AFve-bayes-classifier-fa59e3e24aaf [Accessed 10 Oct. 2021].

Wikipedia Contributors (2019). *Bayes' theorem*. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Bayes%27_theorem [Accessed 2 Oct. 2021].