Deriving strategies for an E-Sports game Valorant using statistics/machine learning

Young You, Hao Le

Jun 2, 2022

* Summary of questions and results

We researched the following questions for the ESports game Valorant, which is an online 5 vs 5 First Person Shooter developed by Riot Games.

Q1. What are the most effective pre-strategies?

Pre-strategies are decisions made before the start of a game. These decisions include game maps, agent compositions, and starting side (attack or defense). We researched how much these decisions can influence win percentage.

The results showed that the most effective pre-strategies are agent picks and side picks based on maps. Based on the map, the starting side, choosing to start as Attack or Defense can affect the outcome of the game. Additionally, the Agent composition for a team can also affect the outcome of a game based on the map.

Q2. How is the win probability of the game changed in real-time based on post-strategies?

Post-strategies are decisions made after the start of a game. These decisions include item purchases, economy management, and game style. We researched how much these decisions can influence win percentage.

The results show that there are multiple post-strategies that heavily influence win rate. Some of these post-strategies include playing for FirstKills, avoiding FirstDeaths, planting the spike as often as possible, defusing the spike as often as possible, and choice in economy purchases. The results demonstrated that the stats that favor victory conditions such as eliminations, spike explosions, or spike defuses have the most influence on win rate.

Q3. Can machine learning predict a winner based on the result of the questions above?

We researched whether or not machine learning can predict the winning team based only on the information we used in the questions above. Also, we researched which one shows the best accuracy and which one shows the worst among the several ML models, and experimented if the performance of the lowest-performing model can be improved by tuning hyperparameters.

The result showed that it is possible to predict the winning team without score information based on the map and agents selected before the play as well as the tactics used during the game. In addition, the LogisticRegression model showed the highest performance as the ML model to find such information, and RandomizedSearchCV showed the quick and better performance when searching for HyperParameter.

* Motivation

The reason that we care about our research questions is that we want to find answers that we can apply to our own gameplay. If we can analyze pro teams and players to find out what they're doing to win more often, we can mimic them to lead our own teams to victory.

Furthermore, as fans of esports, we are motivated by being able to root out what problems cause our favorite teams to lose or what strategies they use to constantly win.

* Dataset

1. Original Data Set

We decided to use the statistical data from vlr.gg, which provides the official game history of Valorant. With the help of the CSE163 course staff, we were able to obtain a SQL file that accumulated material from the site.

The original dataset was provided as an SQLite3 database file. The project doesn't use it directly every time due to performance issues because lots of data need to be parsed in advance. For more information about the original SQL data set, please refer to this site.
https://www.kaggle.com/datasets/visualize25/valorant-pro-matches-full-data

2. Processed Data Set

data_depot.py in the project reproduces these CSV files from the SQL dataset and saves them into */data_cache* folder. The project reads these files when the project starts.

File	Data origin
matches.csv	Mostly derived from `matches` tables in the SQL database
games.csv	Mostly derived from `games` tables in the SQL database
scoreboard.csv	Mostly derived from `game_scoreboard` tables in the SQL database
rounds.csv	Built from the `RoundHistory` serialized string field in the `game_roundss` tables in the SQL database
teams.csv	Reproduced from `matches`, `games`, `game_rounds` tables in the SQL database
players.csv	Reproduced from `game_scoreboard` tables in the SQL database

Please refer to README.md in the project for more information on this.

* Method

1. Background

Valorant is a 5vs5 FPS Game, and the objective of the game is to win 13 rounds. At the start of each game, the two teams are split into two sides where one is called Attackers and the other is called Defenders. These two sides have different objectives. The Attackers need to plant the spike on the bombsite. Once planted, Defenders have to defuse it before 45 seconds otherwise the Attackers will win the round. If Defenders defuse it, they win the round. Both teams also can win by eliminating all other parties. After 12 rounds, two teams swap sides from Attackers to Defenders and Defenders to Attackers.

In between rounds, players can purchase weapons and abilities that will assist them in playing out a round. Depending on the amount of money they use to purchase these weapons and abilities, it will be recorded as an Eco, Semi-Eco, Semi-Buy, and Full-Buy which is respectively from least to greatest amount of money spent.

Every character in Valorant is unique and is called an Agent. These Agents are classified into four classes based on their abilities and role: Duelist, Sentinel, Controller, and Initiator. Both teams can play the same agents but they cannot have two of the same agents on the same side.

2. Method of Q1

1) `data_depot.py:` Decide which parameters can be changed before the game starts, and collect them. (i.e., map, agents)
2) Get map and agent information used by each team from the games.csv. Also, collect the first role in the game of each team.
3) `q_helper_functions.py`: Create additional necessary fields such as agent type. Also create useful information fields as well, such as which team won according to the score of the round.
4) Infer the index of the teams in each table using the pre-made teams.csv. Join games.csv, rounds.csv, and scoreboard.csv with each other, removing any unestimable teams in the process.
5) `q1_pre_strategies.py:` From rounds.csv, Collect which team won each game and how much the team purchased items.
6) Merge games.csv and rounds.csv to create maps, game roles, and agent information and statistics corresponding to such wins above.
7) Find the player's record in scoreboard.csv. Collect agents and game detailed stats of each player.
8) Merge those records and games.csv to create statistics on the trade-off between maps, roles, and opposing agents.
9) Combine all the information gathered above and write result statistics for the questions of the project and plot charts.

3. Method of Q2

1) `data_depot.py, q_helper_functions.py:` Decide which parameters can keep changing after the game starts, and collect them. (i.e., item purchase, game strategy), and repeat steps 2-4 in the above problem.
2) `q2_post_strategies.py:` Merge games.csv, scoreboard.csv, and rounds.csv to create information about agents, item purchase records, and detailed combat scores according to each game index.
3) Create a table based on players' agents and agent types and research stats based on them. For example, keep track of which agent type a player has played, and recalculate stats based on that agent type and each agent union.
4) Normalize the numbers so that an increase in the number of game rounds is not interpreted more advantageously due to accumulated numbers.
5) Combine all the information gathered above and write result statistics for the questions of the project and plot charts.

4. Method of Q3

1) `q3_machine_learning.py:` Decide which parameters we used q1 and q2, and filtered it into `q3.1_ml_data.csv`. These are field information that is used for ML

Fields	First Test	Second Test	Explanation
TotalRounds	V	V	The total number of rounds played in the game.
Num_Eco, SemiEco, SemiBuy, FullBuy	V	V	Item purchase amount information (4 sections) in all the rounds.
AttackFirstHalf	V	V	First role of the map (Attack 1 / Defense 0)
Kills	V		The total number of Kills in all the rounds.
Deaths	V		The total number of Deaths in all the rounds.
FirstKills	V	V	The total number of First Kills in all the rounds.
FirstDeaths	V	V	The total number of First Deaths in all the rounds.
OnevOne, Two, …, OnevFive	V		The total number of rounds that the one left alone against enemies. ** Only applicable for elimination victory type.*
Plants, Defuses	V	V	The total number of Plants attempted and Defuse in all the rounds.
Controllers, Duelists, Initiators, Sentinels	V	V	The total number of agent types that the team used
Map_XXXXX	V	V	Map name fields with one-hit encode

2) Research several ML models suitable for decision-making about win/lose based on such information. These seven ML models were used: `DecisionTreeClassifier`, `RandomForestClassifier`, `SVC`, `KNeighborsClassifier`, `LogisticRegression`, `GradientBoostingClassifier`, `AdaBoostClassifier`.
3) Shuffle the dataset and separate it into between a training set and a test set with to 7:3 ratio.
4) Train the model with the train set and provide a test set to measure the accuracy of predicting win/loss with the above information.
5) Repeat steps 3 and 4 above for all 7 ML test models. Use only default parameters for each model.
6) Repeat steps 3, 4, and 5 to make 10 trials and add up the results.
7) Run all the above tests one more time with the second data group that has less impact on the win rate. (i.e. Exclude `Kills`, `Deaths`, and `OnevOne`, …, `OnevFive` fields) Then compare the two results.
8) Adjust hyper parameters for the lowest accuracy model using `GridSearchCV` and `RandomizedSearchCV`. Check if the accuracy changes. These parameters were used for the CV test.

```
params = {
    'criterion': ['entropy', 'gini'],
    'max_depth': [3, 5, 10, 20, 30],
    'max_leaf_nodes': [3, 5, 10, 20, 30],
    'min_samples_leaf': [2, 5, 10, 20],
    'min_samples_split': [2, 5, 10, 20],
}
```

`GridSearchCV` : cv=3, n_jobs=-1

```
params = {
    'criterion': ['entropy', 'gini'],
    'max_depth': range(3, 31, 3),
    'max_leaf_nodes': range(3, 31, 3),
    'min_samples_leaf': range(2, 21, 2),
    'min_samples_split': range(2, 21, 2),
}
```

`RandomizedSearchCV` : n_iter=100, cv=3, n_jobs=-1

9) Combine all the information gathered above and write result statistics for the questions of the project and plot charts.

* Result

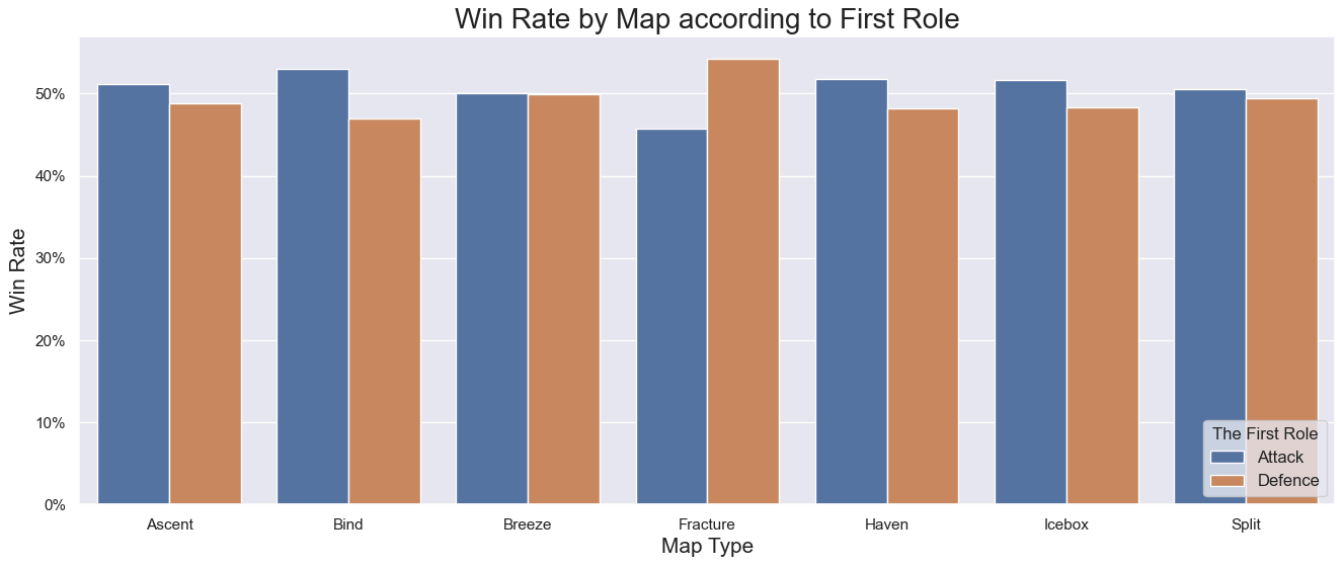
Q1. What are the most effective pre-strategies?

1) Game Maps

In relation to game maps, we researched whether there is a way to increase the win rate before the game starts by asking the following questions

a) Is this map advantageous for the first attack or for defense?

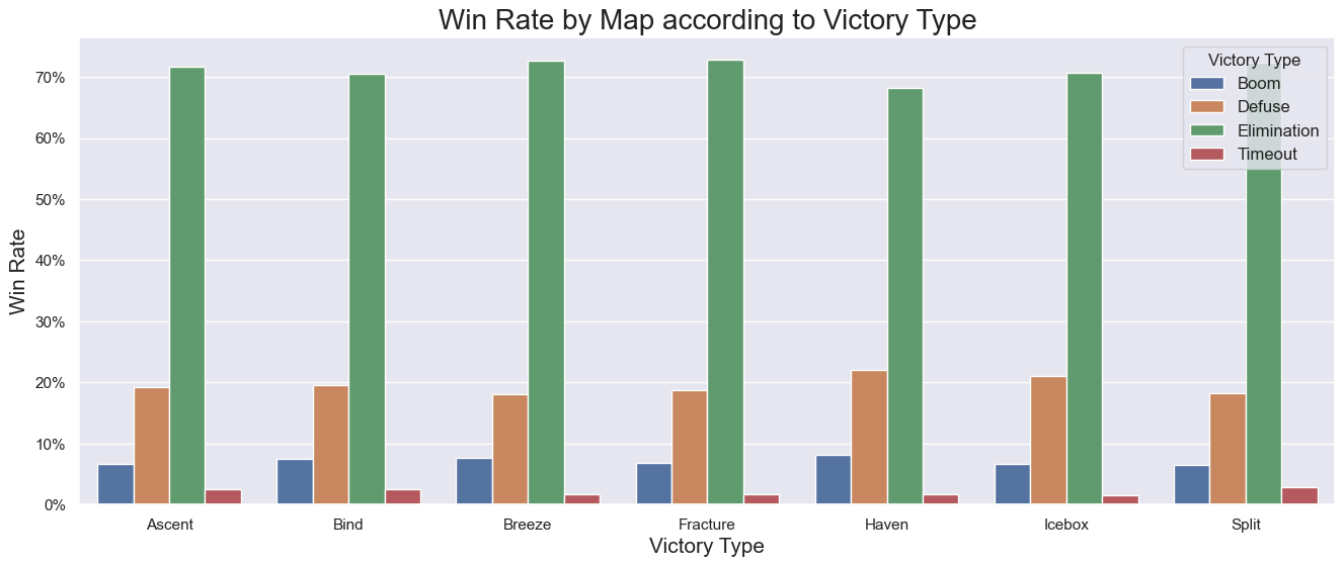
In this game, players need to win a total of 13 rounds to win the game, and the role changes after the end of the 12th round. It means the first role is guaranteed to play 12 rounds, so it affected the win rate according to the role advantage on each map.



The result shows that Ascent, Bind, Haven, and Icebox are more advantageous for Attack, Fracture is more advantageous for Defense, and there is no big difference on the Breeze and Split map.

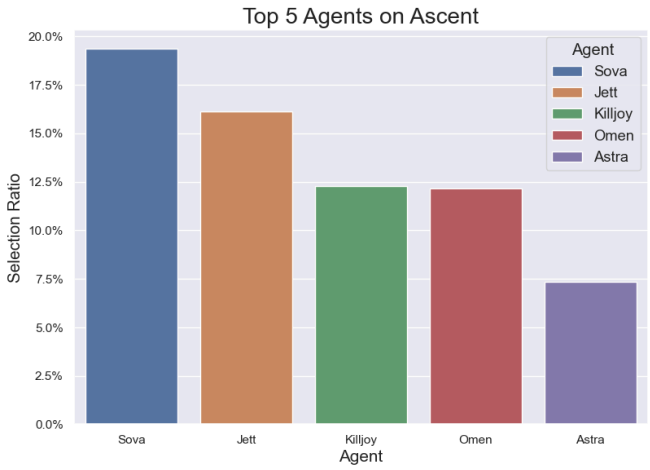
b) Among four victory conditions (Elim/Boom/Defuse/Time) winning strategy is advantageous on this map?

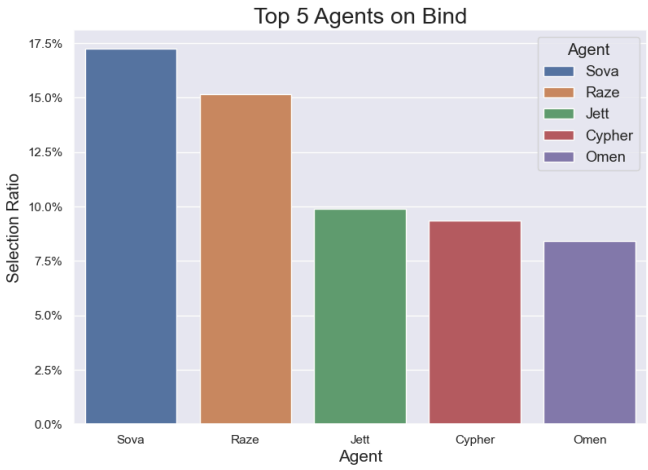
There was no significant difference in victory type for each map. Most maps showed Elim the highest, but Haven Map showed higher Boom/Defuse than other maps, so the strategy using Spike might be more efficient than other maps.

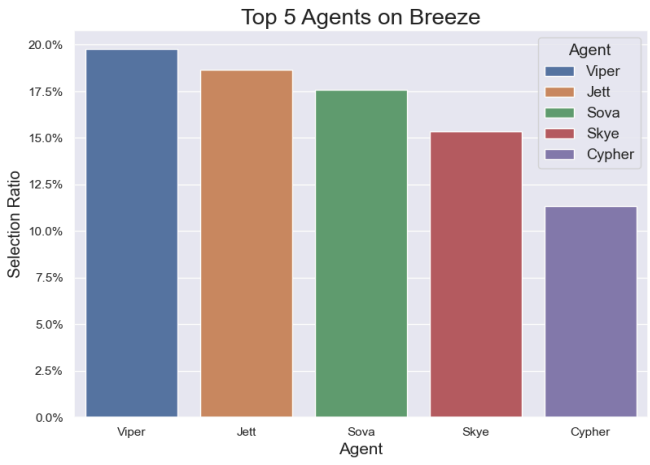


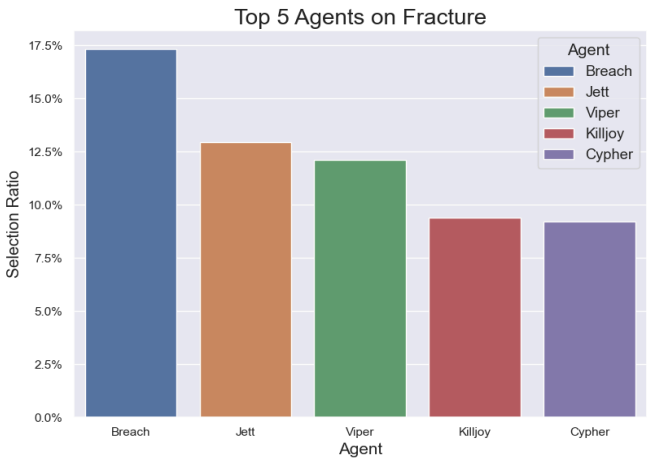
c) Which agent has the most advantage on this map?

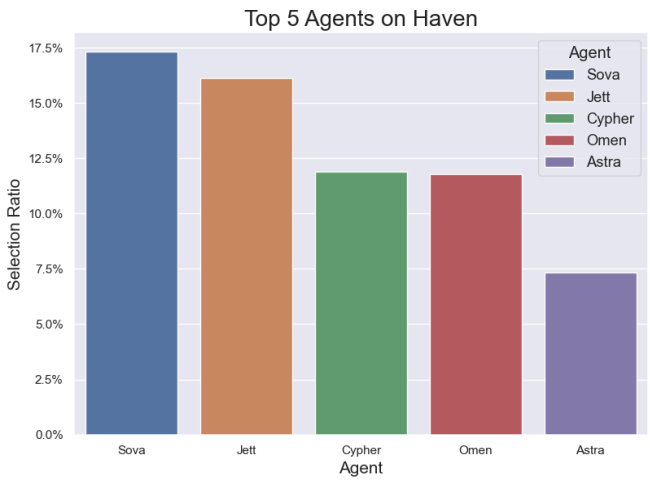
We looked at which characters were included the most when winning on a specific map. Below are the top 5 characters for each map. Sova and Jett showed the highest selection ratio on various maps when the team won.

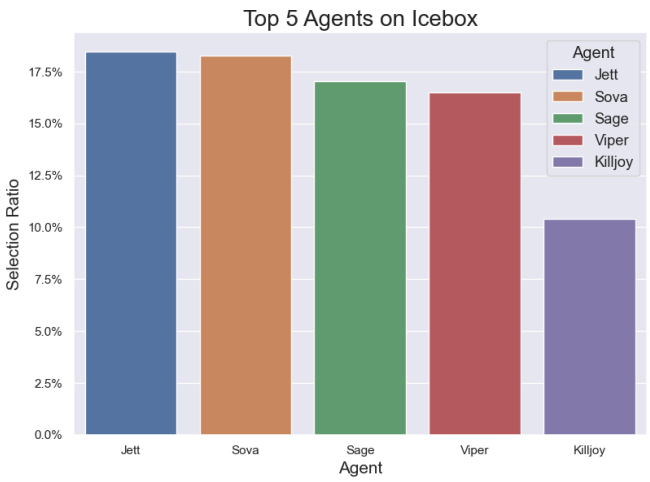


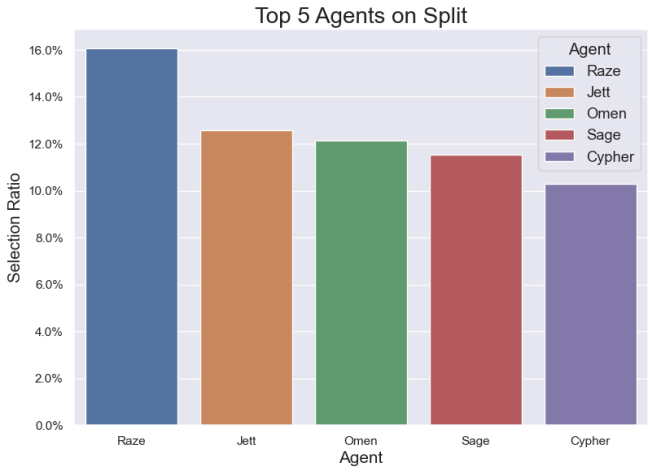








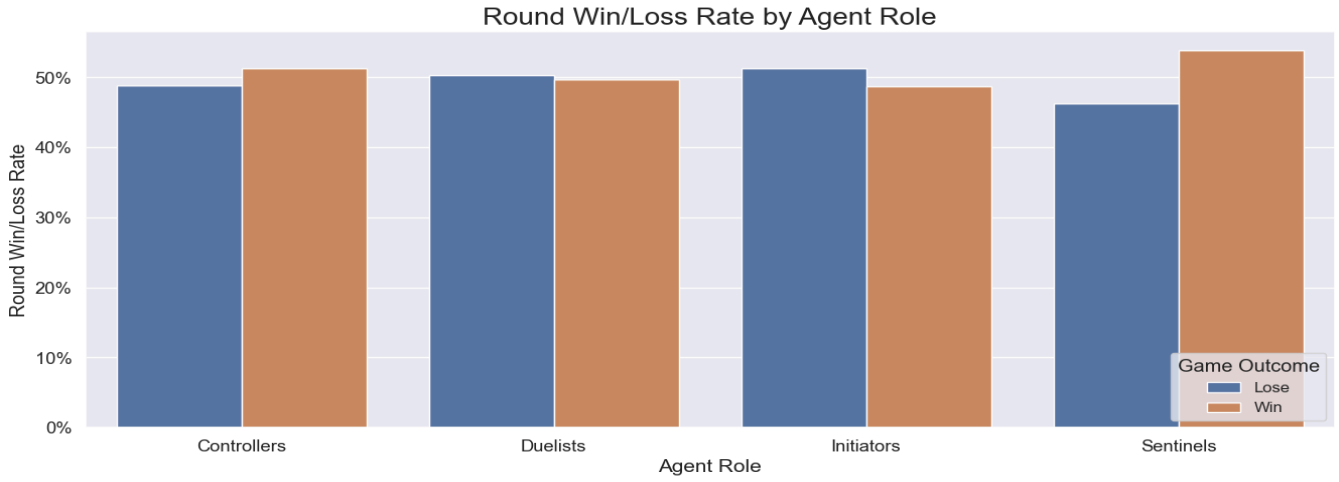




2) Agents

We researched how the choice of the agents affects the game outcome by asking the following questions.

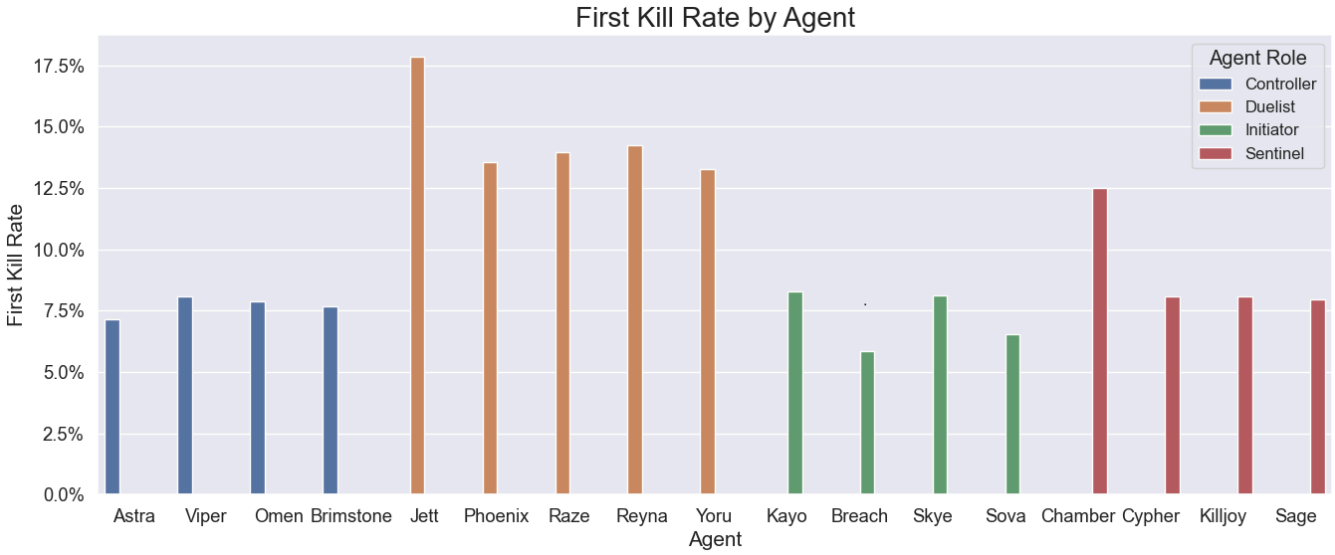
a) Win rate change rate when the same class of agent enters (Duelists / Controllers / Initiators / Sentinels)



The agent role that has the greatest win rate change when 2 or more agents in the same agent role exists on the same team are Sentinels. Teams have a higher win rate when their team composition comprises at least two or more Sentinels.

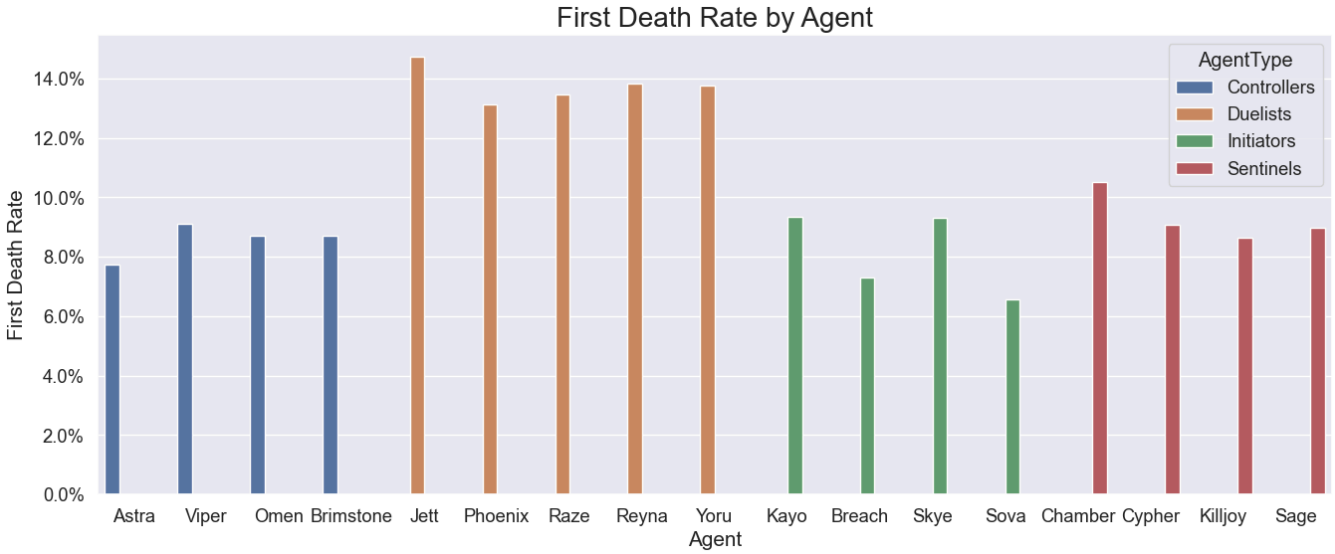
b) Which agent is more likely to kill the enemy first?

The agents that are the most likely to kill the enemy first are those from the Duelist class. Although Chamber is a Sentinel, he has a first kill rate comparable to Duelists.



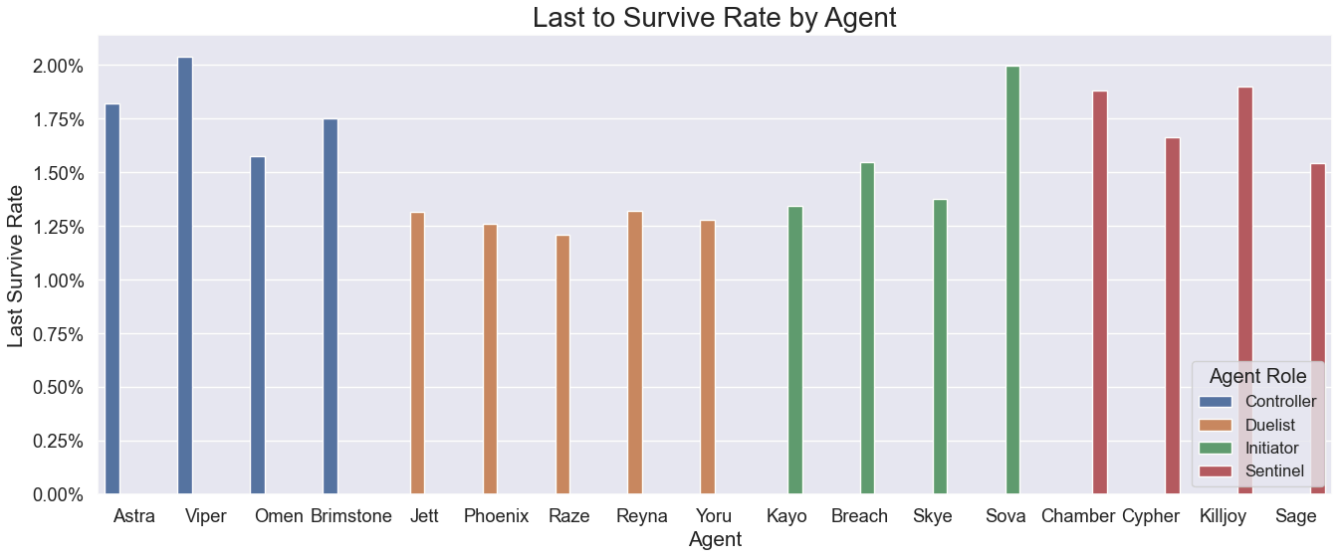
c) Which agent is more likely to be killed by the enemy first?

The agents that are more likely to be killed by the enemy first are those that are Duelists. We believe that this is influenced by the fact that Duelists are also the most likely to get the First Kill.



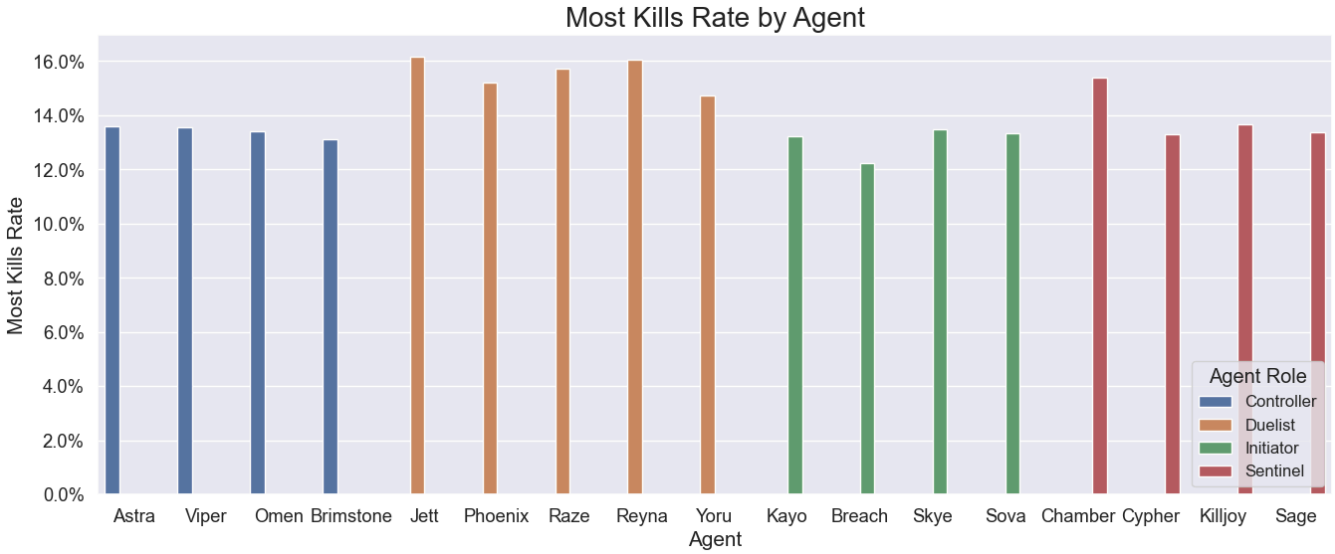
d) Which agent is more likely to be the last to survive on the team?

The agents that are the most likely to be the last to survive on a team are Viper, Astra, Sova, and KillJoy. Duelists and Initiators are the least likely to be the last to survive, however, Sova is an outlier and is the second most likely agent to be the last to survive. Controllers and Sentinels are the most likely Agent Roles to be the last to survive.



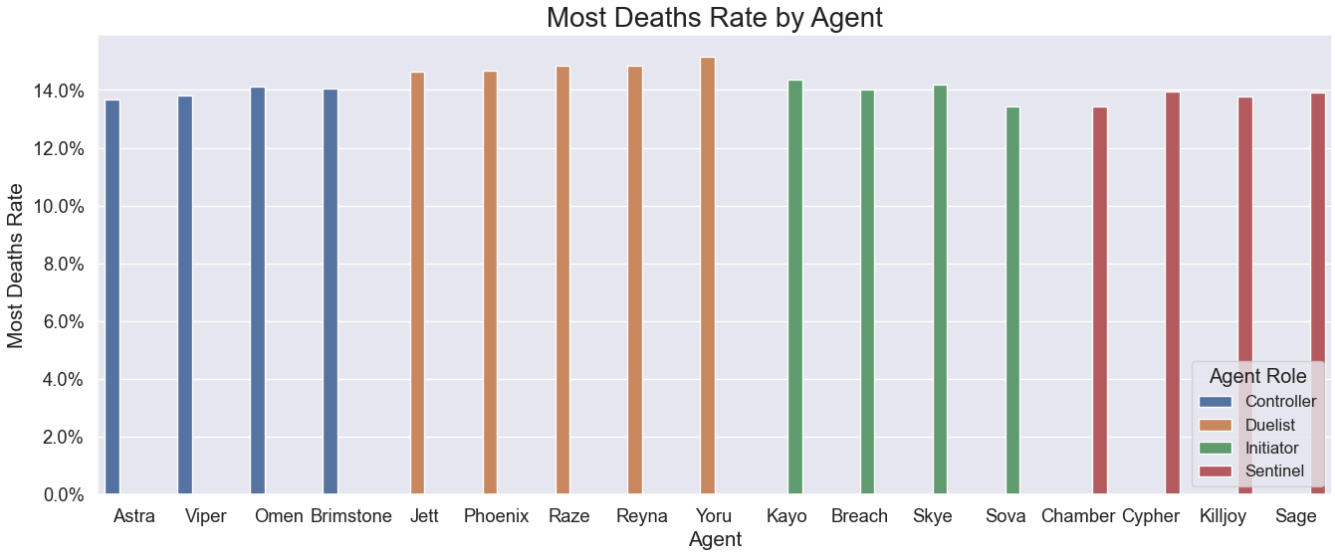
e) Which agent is more likely to kill the most enemies?

The agents that are the most likely to have the most kills at the end of a game are Duelists. Chamber is an outlier and has a most kills rate comparable to Duelists.



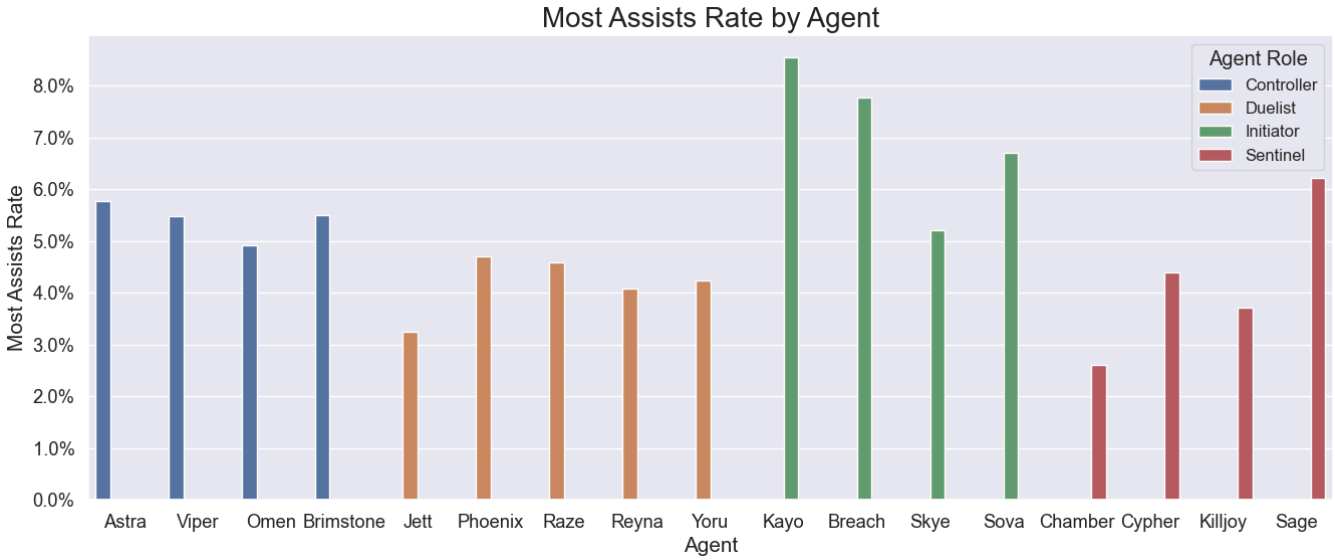
f) Which agent is most likely to be killed by the enemies?

The agents that earn the most deaths are Duelists. We believe that this is influenced by the fact that Duelists are the most likely to earn the most kills.



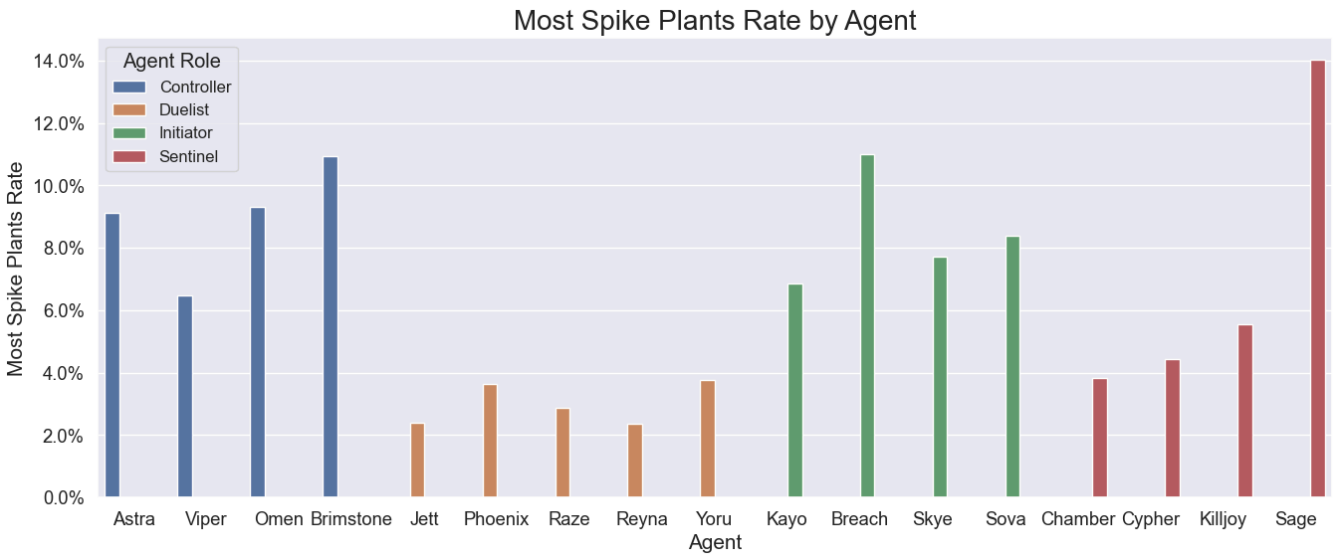
g) Which agent is most likely to assist to kill enemies?

The agents that are the most likely to assist an ally in killing an enemy are Kayo, Breach, Sova, and Sage. Sage is the only Agent out of these four that is not an Initiator. We see that Initiators are the Agent Role that earns the most assists.



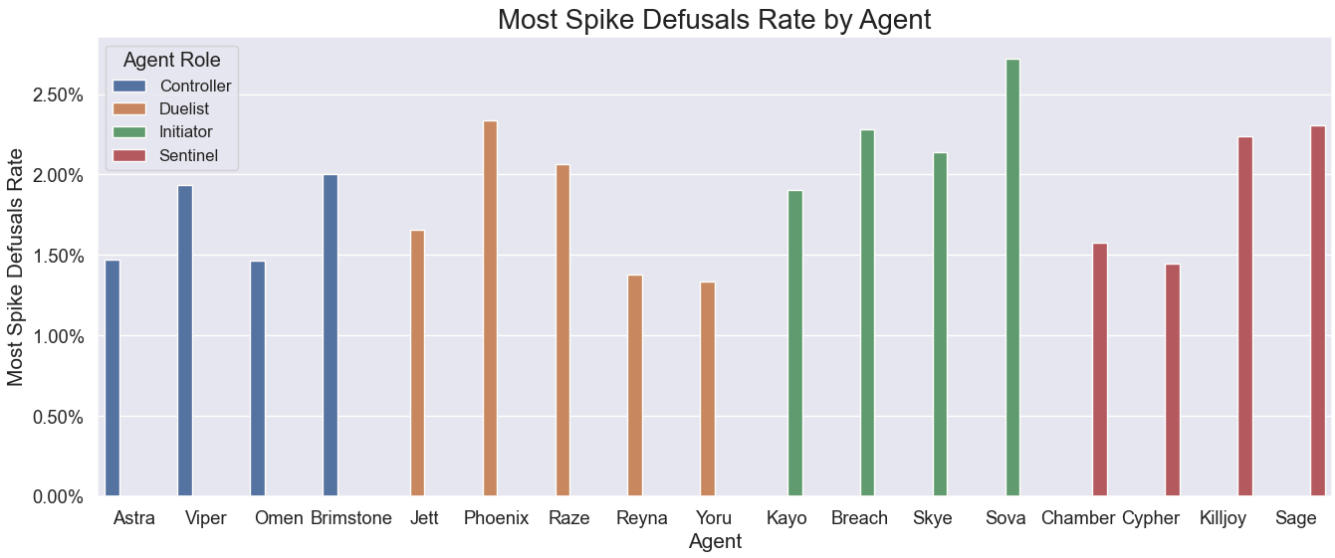
h) Which agent is most likely to plant a bomb?

The agent that is the most likely to plant a bomb is Sage. Duelists have the lowest rate in planting the spike most often.



i) Which agent is most likely to defuse a bomb?

The agent that is the most likely to defuse a bomb is Sova. Overall, Initiator agents are the most likely to defuse a spike.



Q1 Results:

Based on our result from question 1, we can conclude that there are certain maps where choosing to start either Attack or Defense side first is more advantageous, such as choosing to defend first on Fracture instead of attacking first.

Furthemore, there are certain agents that give you a better chance of winning on certain maps such as choosing to play Raze on Split. It was interesting to see that Raze was not a top 5 agent for the map, Breeze, despite being an extremely popular pick. However, the implication behind this is reasonable because Breeze is a very open map with only a few choke points which are areas that Raze excels in abusing.

Moreover, we saw that the agents that earned the most kills were Duelists. They were also the agent role that earned the most deaths, First Kills, and First Deaths. Chamber is the only non-Duelist agent that has kill stats similar to a duelist. Our hypothesis for this surprising result is because Chamber is a unique Sentinel agent whose ability enables him to obstruct enemies using guns instead of obstructions such as walls and turrets.

As we continued to analyze, we saw that the agents that earned the most assists were Initiators but a surprising result was that Sage had a very high most assist percentage despite not being an Initiator. Our hypothesis for this surprising result is because of Sage's heal ability and resurrection ability that nets her free assists when a healed or resurrected ally earns kills.

An interesting result we saw was that Duelists had the lowest spike plant rate. Our argument for why this happens is because Duelists also average the most deaths, kills, and First Deaths every game/round. Based on our analysis, the teams use agents that fall into the Duelist role to focus on eliminations instead of objectives such as planting the spike. Additionally, it's reasonable that Duelists do not get as many chances to plant the spike if they are the first to die for their team.

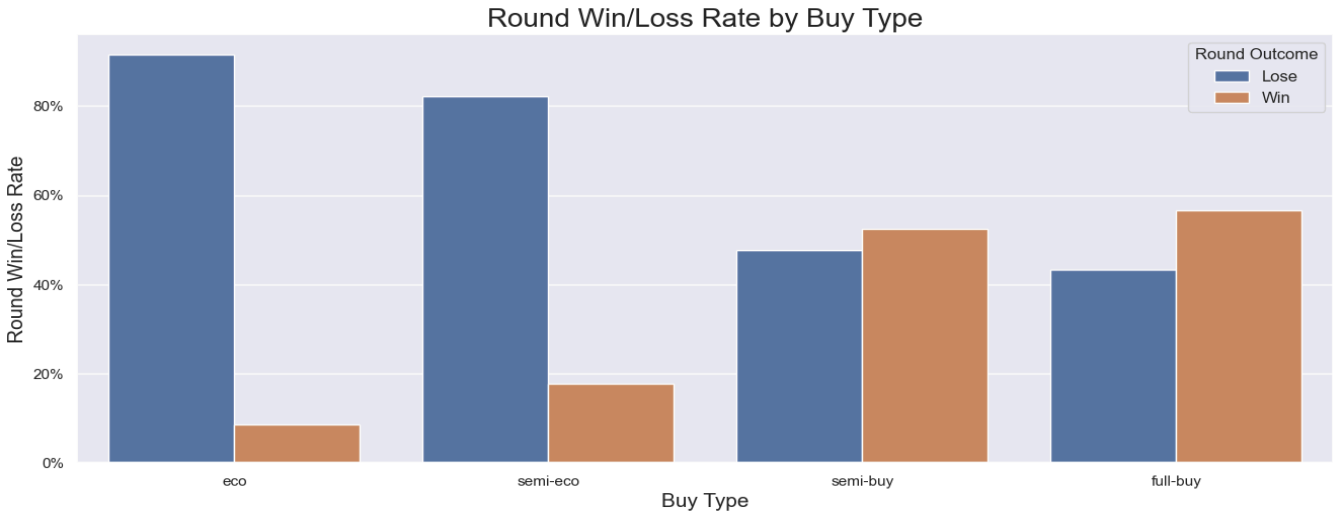
For spike defusals, we saw that Initiators are the most likely the defuse the spike with Sova being the agent earning the most spike defusals. An interesting result we saw was that the spike defusal rates was much more widely spread across the board compared to spike plants. We believe the reason for this is because planting a spike is not as influential as a win condition as compared to defusing a spike which we will explain in Q3. It's reasonable to assume that teams have less influence over choosing who defuses a spike as compared to when choosing who to plant a spike.

Q2. How is the win probability of the game changed in real-time based on post-strategies?

1) Item Purchases

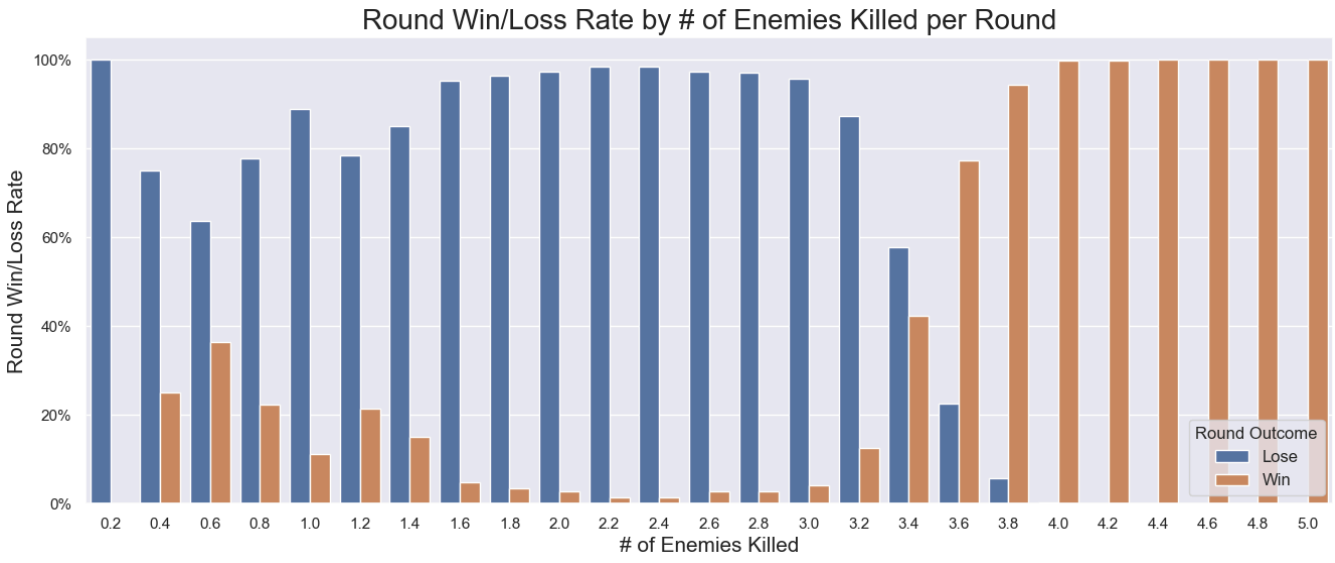
a) How different are win rates according to the amount of item purchase?

The results show that spending more money results in a better chance of winning a round. The chances of winning an eco round or semi-eco round are very slim. A semi-buy or full-buy will give you much better odds of winning a round. Thus, the more credits spent by a team, the more likely they are to win a round. Conversely, the less a team spends on items and abilities, the more likely they are to lose a round. These results show that it is advantageous to spend semi-buy or more at once, rather than frequently paying semi-eco.



2) Battles

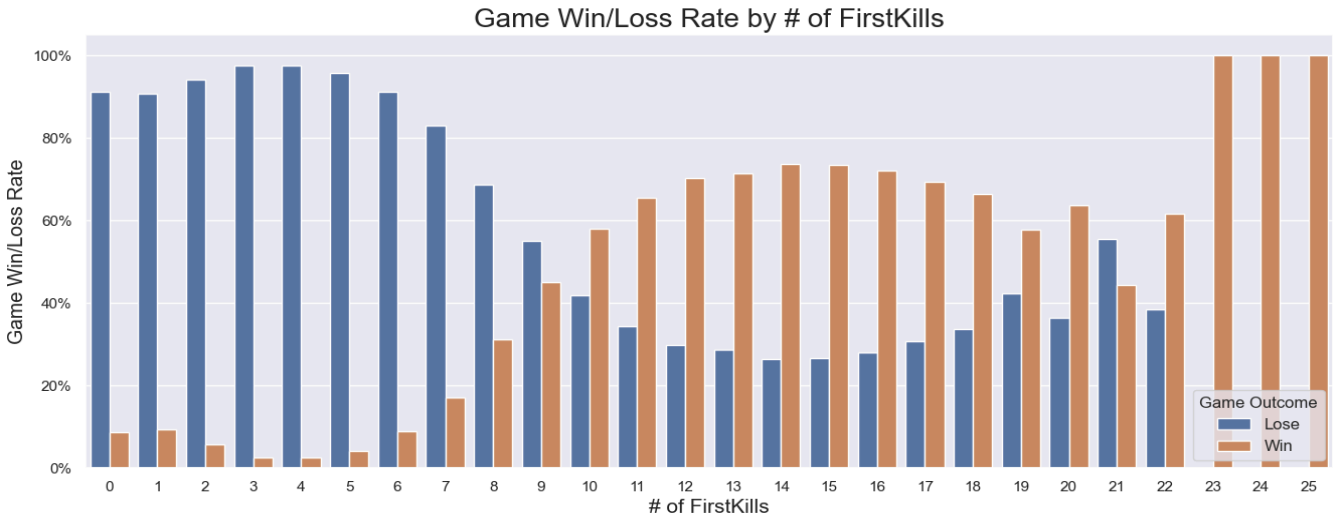
a) How does the win rate change according to the total number of the enemy kills?



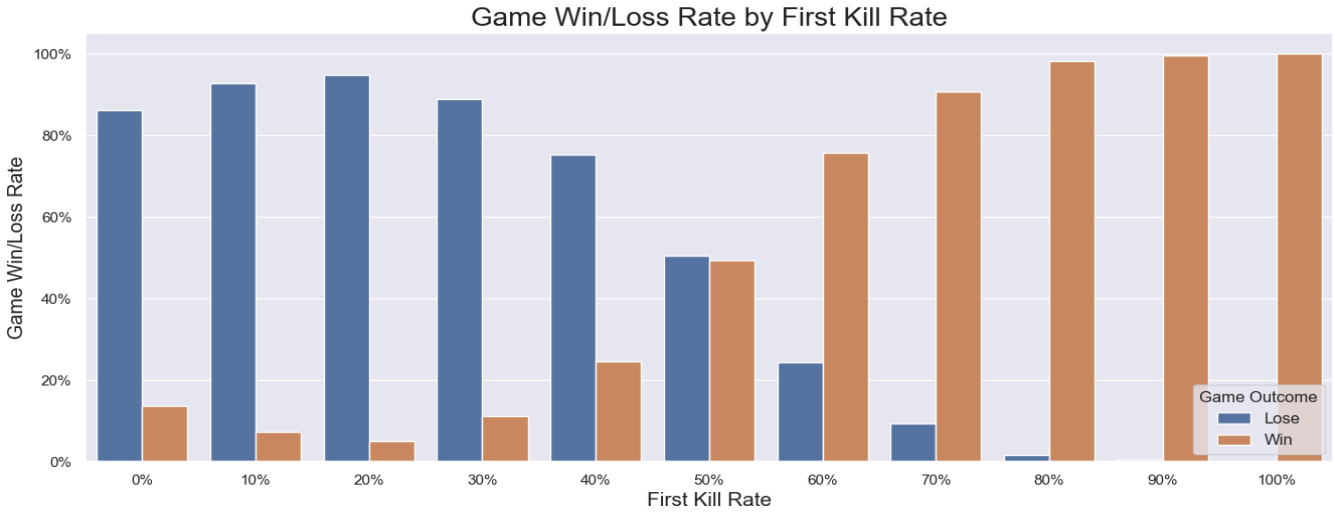
We saw that your win rate increases dramatically when a team averages above 3.2 enemies kills per round. As predicted, killing 5 enemies in a round will always result in a round win by elimination. If you average 3.2 kills per round, your chances of winning the round are very slim. If a team averages 0.0 to 0.2 kills a round, they have no chance of winning a round.

b) How does the win rate change according to the total number of the first kills in the game?

Based on the trend of the graph, we saw that the more first kills / first bloods that a team earns throughout a game, the more likely they are to win. Conversely, the less first kills they earn over a course of a game the less likely they are to win. Something interesting that we saw was that even if you had very little to no first kills, it is still possible to win a game.



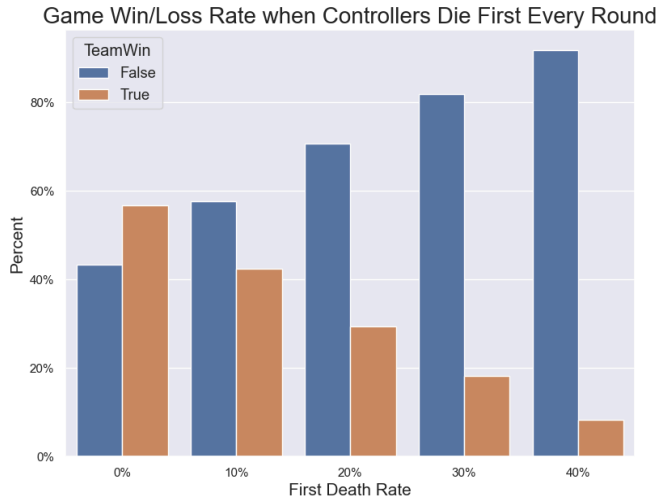
c) How does the win rate change according to the percent of the first kills in the round?

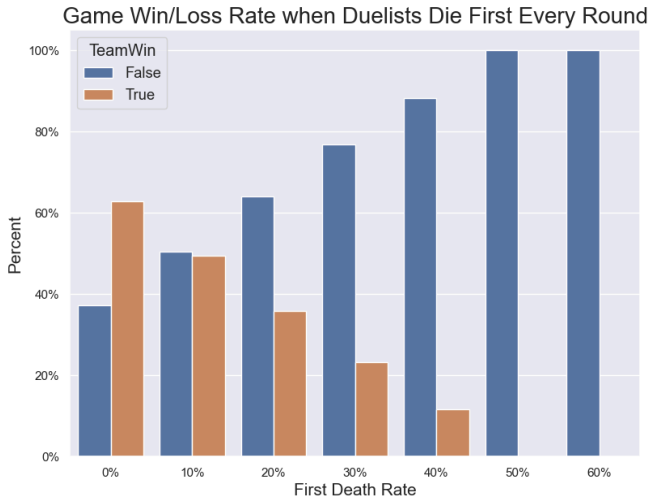


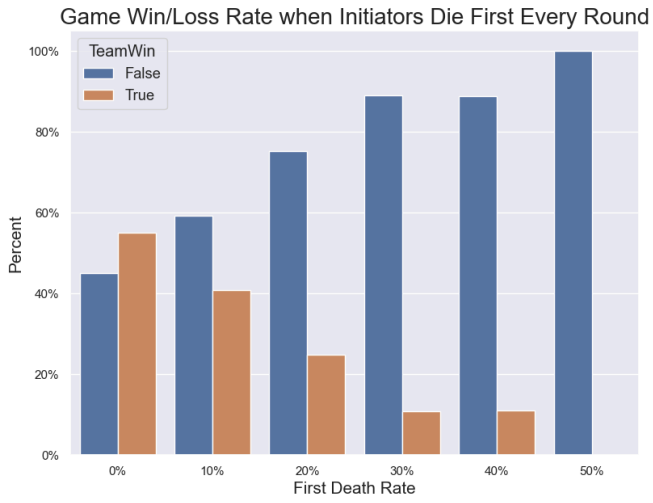
Your game win rate increases as your First Kill Rate increases. On the other hand, your game win rate decreases as your First Kill Rate decreases. It was interesting to see that if you average about 50% of the First Kills throughout a game, your chance of winning/losing that game is nearly 50%. Furthermore, if you average above 70% of the First Kills throughout every round in a game, a team's win rate will approach 100%.

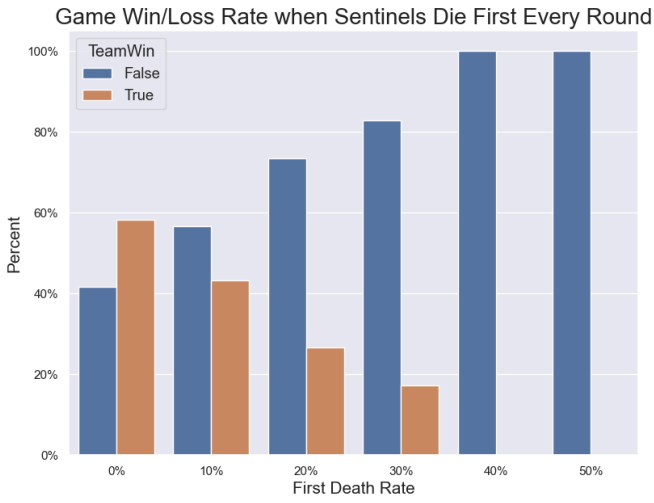
d) How does the win rate change when the team loses each type of agent first?

For every agent role, we saw that as First Death Rate increases, the win rate decreases while the loss rate increases. Therefore, no matter what role is played, if a certain agent is the first to die every round it is a massive disadvantage for a team.





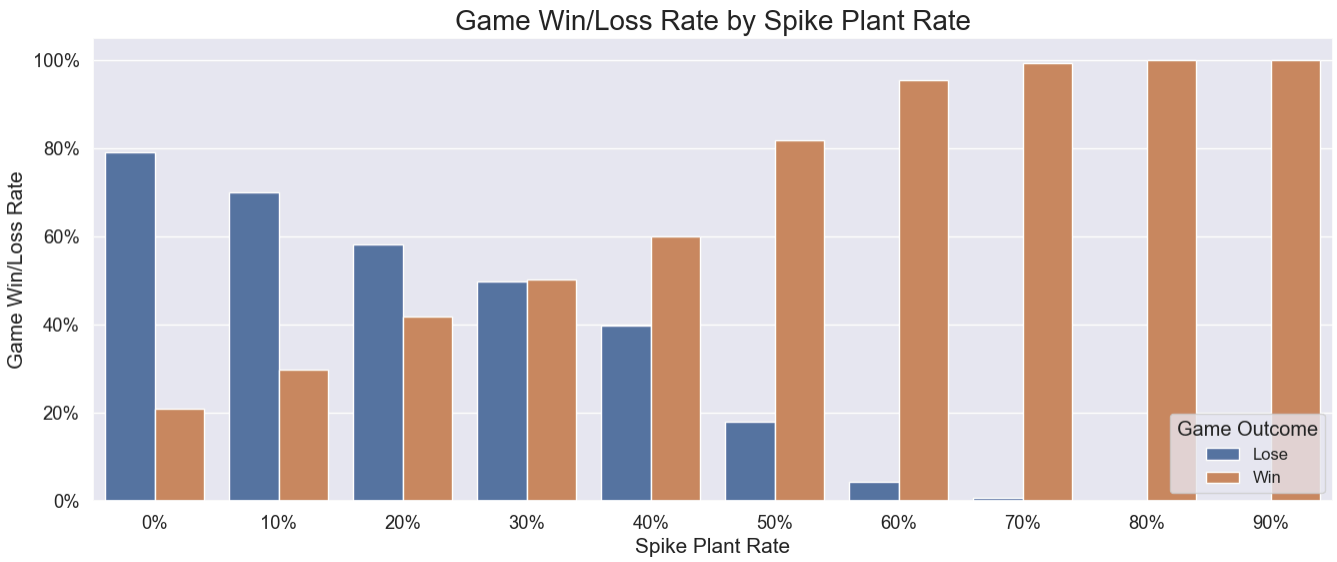




3) Victory Strategies

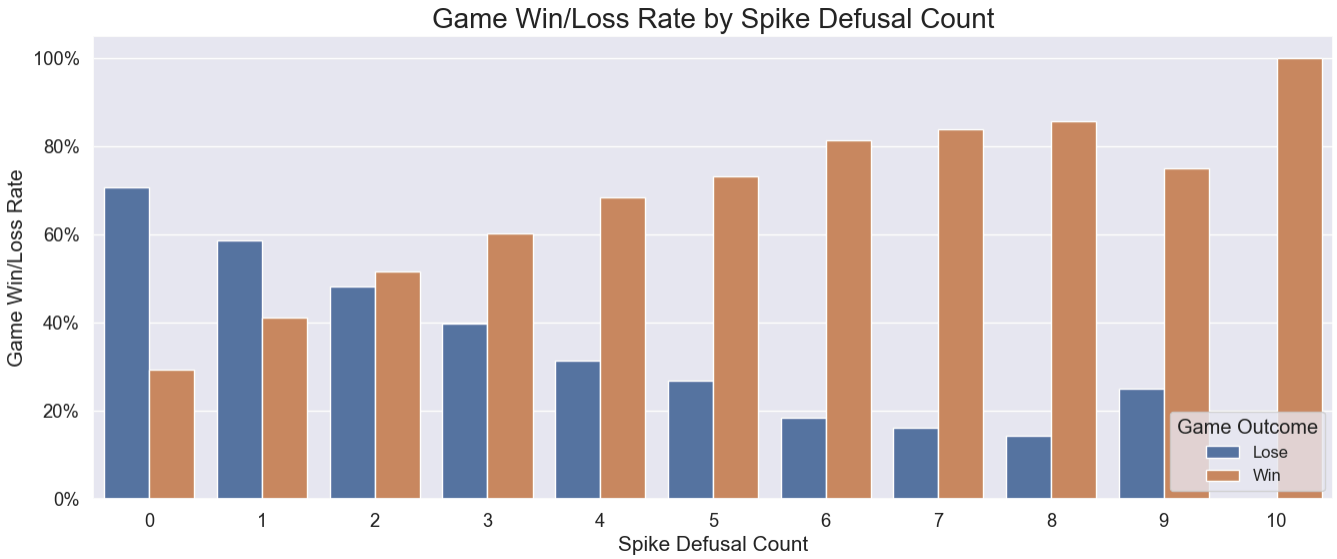
a) How does the win rate change according to the percent of the attempt to plant a bomb in the round?

As the spike plant rate increases, a team's win rate will also increase. If you are able to average above 60% spike plants on the Attacking half, your chances of winning the game overall is almost 100%.



b) How does the win rate change according to the total number of defusing bombs?

As spike defusal count increases, so does the game win rate. This is reasonable because for every spike defused in a round win earned towards winning the game overall. Something interesting we saw was that even if you defuse 0 spikes, it is still possible to win the game which is most likely through other victory conditions such as elimination so killing the enemy before they get a chance to plant the spike.



Q2 Result:

Based on our analysis of post-strategies (decisions that influence win rate after a game starts), we can conclude that win rate increases as a team averages more enemy kills per round and more total First Kills per game. An interesting result we saw was that no matter what Agent Role a person plays, if you are the First Death for your team every round, your win rate will greatly diminish. We believe that this had correlation with how averaging more First Deaths results in a lower win rate and less First Kills also results in a lower win rate.

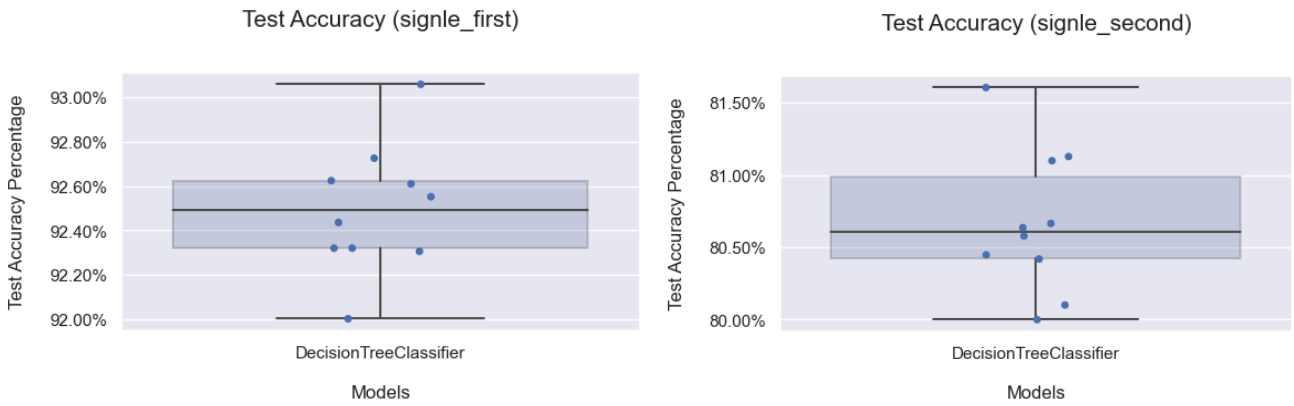
We also saw that economy management is important because the buy type that a team does for a round heavily influences their round win rate. The more money that a team spends on weapons and abilities results in a better chance of winning a round. The less money that a team spends will result in a greater chance of losing a round. For example, if a team decides to do an eco buy and spend little to no money, their chances of winning the round are slim to none.

Furthermore, we can conclude that the more often that you plant a spike on the Attackers Side, your win rate will greatly increase. On the flip side of that, we also saw that the more often you defuse a spike, your win rate will also greatly increase. An interesting result we saw was that teams have about a 30% chance of winning the game even if they get 0 spike defuses on Defense. Based on our analysis of different victory conditions, we believe that this is influenced by the result that most rounds will end by elimination instead of spike explosions or spike defusals. Therefore, it's possible to win games by simply killing your enemies before they get the chance to plant the spike and force you to defuse.

Q3. Can machine learning predict a winner based on the result of the questions above?

1) Winner prediction

a) Can machine learning can predict the winning team based only on the information we used in the questions above?

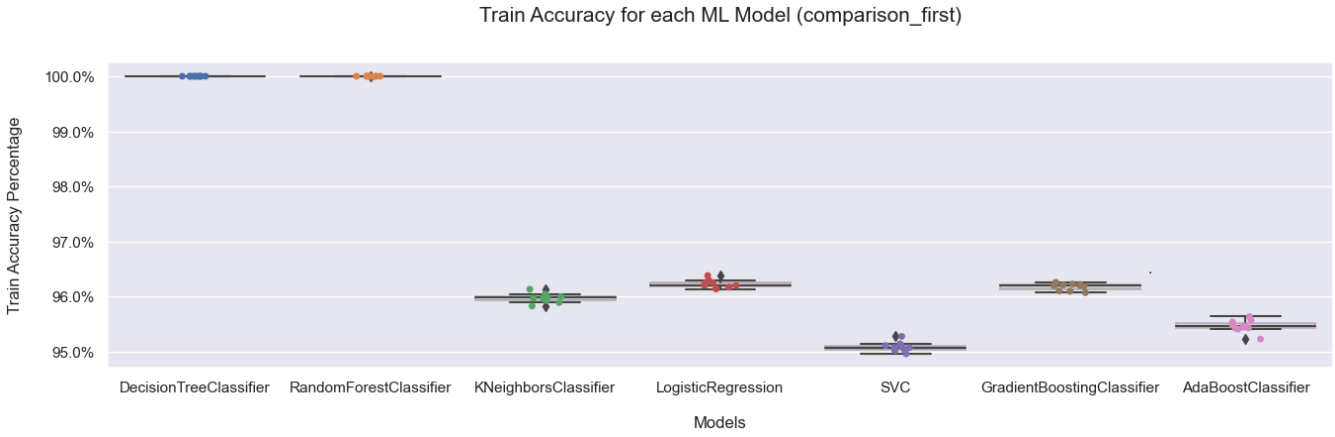


With the first data group (full data - see Method #4), the winning team was predicted with an average accuracy of 92.5%, and with the first data group (less data - see Method #4), the winning team was predicted with 80.60% accuracy. The basic DecisionTreeClassifier model that we learned in the course was used.

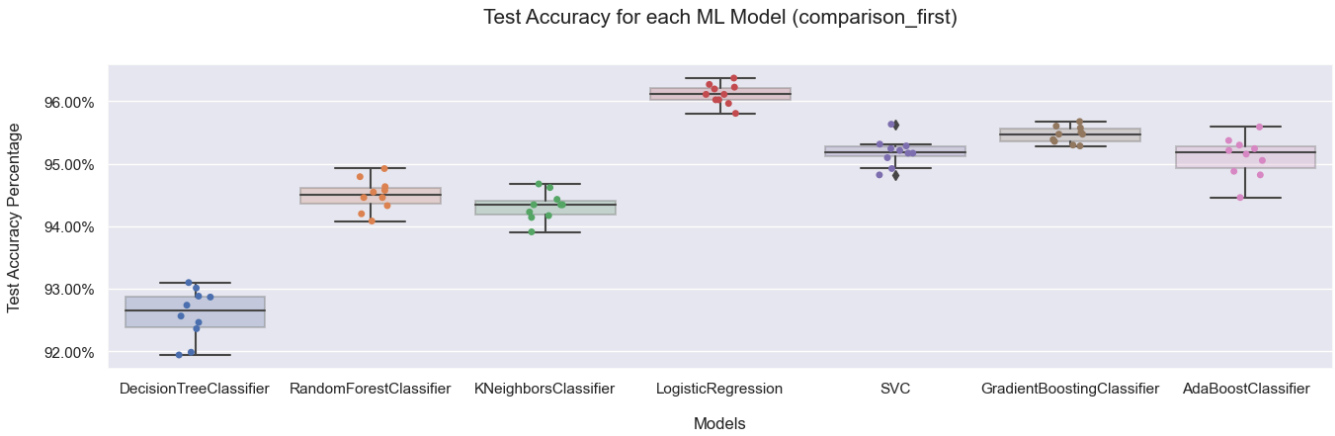
In view of these results, we have come to the conclusion that only the information used in Q1 and Q2 above is sufficient to predict victory or defeat.

2) ML Models

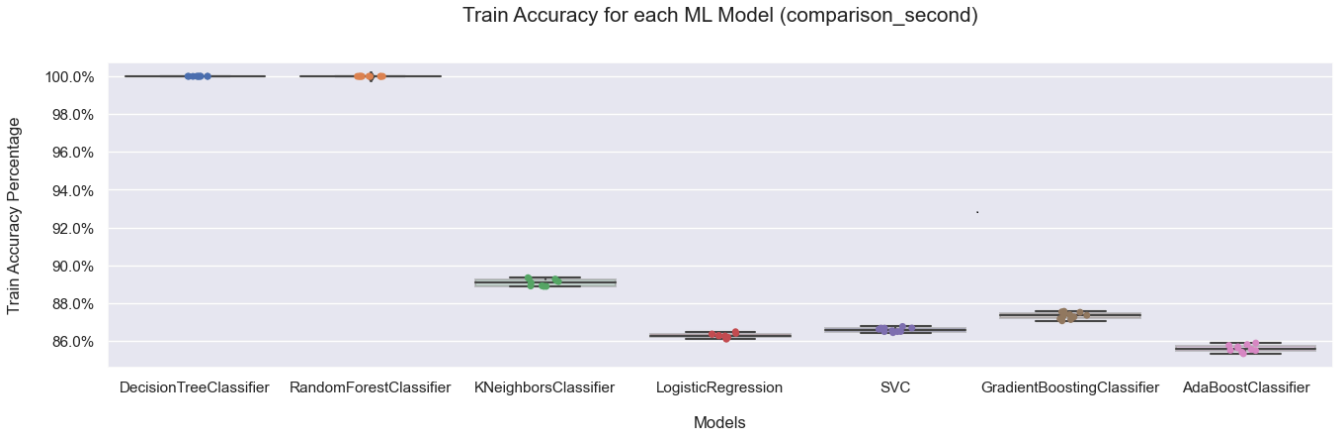
a) Which one does show the best and worst accuracy among the several ML models?



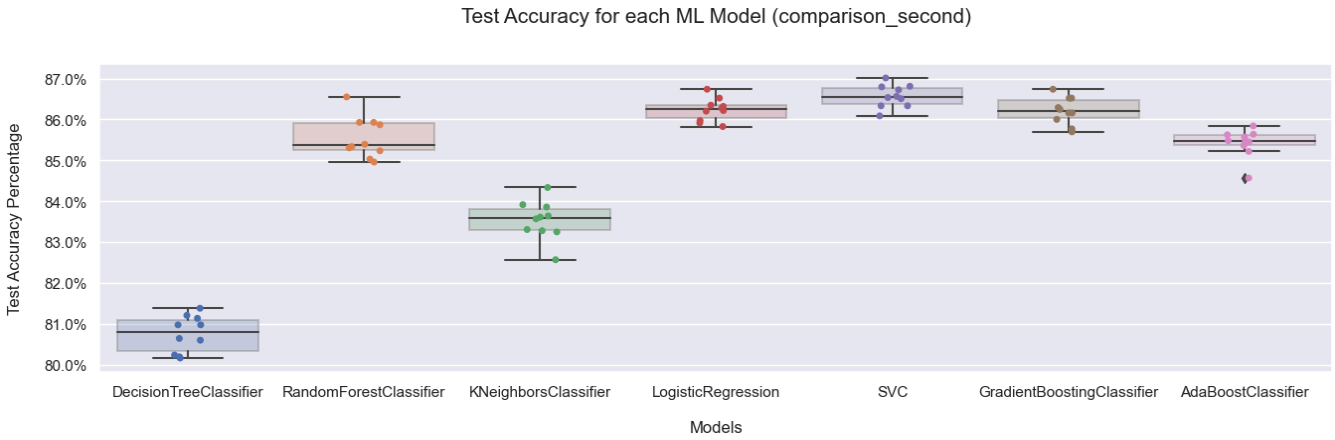
As a result of testing 7 models only with default hyper parameters and measuring accuracy with training data, DecisionTreeClassifier and RandomForestClassifier showed 100% overfitting accuracy, and others showed about 96% accuracy.



When the accuracy was measured with test data, the overfitted DecisionTreeClassifier and RandomForestClassifier had the lowest performance at 92.75%, and LogisticRegression showed more than 96% performance.



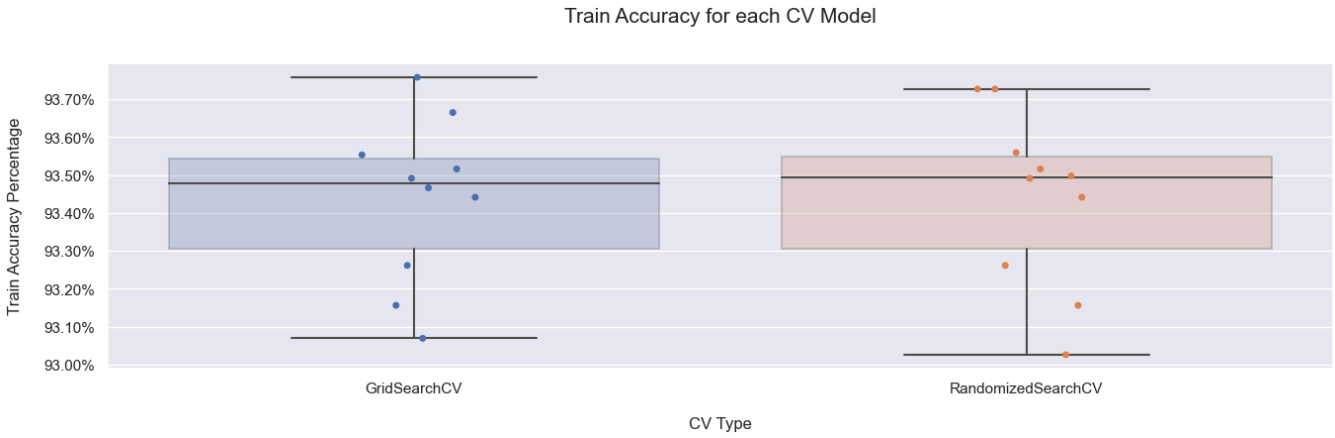
With the second data group, DecisionTreeClassifier and RandomForestClassifier showed 100% overfitted accuracy with training data again, but others showed even less accuracy, about 88%.



When the accuracy was measured with the test data, DecisionTreeClassifier showed the lowest performance at 80.75%, and this time, SVC showed the highest performance at 86.5%.

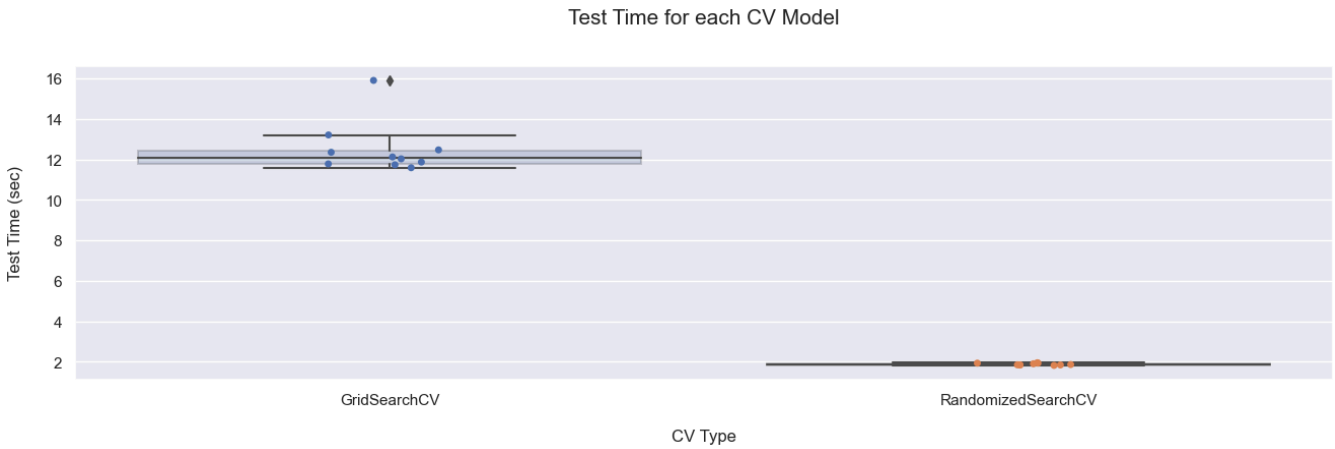
3) Hyperparameters

a) Can the performance of the lowest-performing model be improved by tuning hyperparameters? How much performance can be improved with GridSearchCV and RandomizedSearchCV?



GridSearchCV and RandomizedSearchCV were used to find appropriate hyperparameters of DecisionTreeClassifier, which showed the lowest performance.

As a result of 10 trials with the first data set, the accuracy of DecisionTreeClassifier increased from 92.75% to 93.5%. There was little difference between GridSearchCV and RandomizedSearchCV, but RandomizedSearchCV found slightly better parameters.



There was a big difference in execution time. Whereas GridSearchCV took 12 seconds on average, RandomizedSearchCV found appropriate parameters in 2 seconds.

Q3 Result:

The result showed that it is possible to predict the winning team without any information related to the round score with a 96% probability, and even without detailed battle-related information, ML models predicted the winner with an 86.5% probability. These results make it reasonable to expect that the information analyzed in our last questions 1 and 2 actually has a big impact on wins and losses.

Given that our data is for finding the binary probability of win/lose based on a variety of statuses, it is a reasonable result that LogisticRegression and SVC showed the best performance in the model performance comparison because they are special models for binary classification. Thus, we can expect that using such models to classify binary results (e.g., true/false) will yield better performance than the DecisionTreeClassifier used by default in the course.

In addition, it was also seen that the adjustment of hyperparameters had a considerable effect on the model performance. In the process of finding hyper parameters, RandomizedSearchCV quickly found good parameters in a wide range much faster than GridSearchCV. Considering that GridSearchCV tests the number of all cases of the CV params, it seems like a good way to start by looking for approximate hyper parameters in a wide range with RandomizedSearchCV and then fine-tuning with GridSearchCV.

Although not mentioned in the experiment above, given that LogisticRegression is overwhelmingly faster than SVC, the result shows that the combination of LogisticRegression and RandomizedSearchCV is the best combination for this experiment.

* Impact and Limitations

Potential Implications:

The potential implications of our results is that decisions such as which side a team decides to start on, agent composition, and map choice has influence and can affect the outcome of a game. Additionally, decisions after a game starts such as victory conditions and elimination vs objective playstyle can also influence the outcome of a game.

Benefits/Harm:

The people that are most likely to benefit from our analysis are people that are below the professional level for Valorant but are forming strategies for amateur/semi-professional teams. We believe that they can use this data to forge their own playstyle or mimic what the pros are doing.

However, we believe that the ones that could be “harmed” by our analysis are the pros that want to hide their strategies. At the moment, Valorant has no replay system so our statistical data analysis like what we have here is one of the only substitutes that Valorant Esports teams can use. By sharing this analysis, teams that want to hide their strategies to prevent others from countering their playstyle or victory conditions, will use this against them.

Moreover, games require a variety of strategies, and people want to enjoy a variety of play. However, approaching the game with such data analysis, the pattern with the best win rate might be uniform, and in that case, the true fun of Esports can be lost.

* Challenge Goals

1. Messy Data

1. Although we didn't need to scrape the web data as we initially intended, we had to reprocess it because the data was not clean. The database was given as an SQL database type, and some data was in the serialized 2-dimensional strings, not in fields. (It was similar to JSON/JS/Pickle but it was not). To account for this, we had to write a function that completely parsed and split the strings.
2. The information used for each table is different (ex. Team Abbr vs Team Name), so the data had to be inferred and organized.
3. For these tasks, we had to create a data_depot.py data manager class, and it took much more time than expected which halted our progress.

2. Multiple Datasets

1. We had to join(merge) many datasets before we could use any of it to calculate data.
2. Multiple Datasets: We will draw answers and plot results by synthesizing the information divided into multiple datasets. For example, the overall game results and game details have been divided into separate data. This process will involve extracting the desired fields or re-processing them to make a proper series of data in data fields.

3. Machine Learning

1. We compared many ML models to find the best performance model for our Valorant data.
2. GridSearchCV and RandomizedSearchCV were used to find the hyperparameter, and the performance was compared.
3. Originally we planned to only track the data for Sentinels, the esports team, not the Agent Role. However, as per our mentor’s suggestion, we decided to track general data instead because we would have a larger breadth of data to work with which would give more depth and complexity to our work. As a result of this change, we were able to track more data and make more accurate comparisons.

* Work Plan Evaluation

Most of our work plan was accurate except for plotting the graphs. It did not take us very long to set up an environment where our team could both work on the code; however, deriving the data and making it correspond to our research questions took a very long time as we predicted.

An estimate that was far from reality was that plotting our graphs would only take about 5 hours. It took us much longer than expected because we misunderstood some of the columns in the CSV's that we processed from analyzing the data we retrieved from vlr.gg. After making our corrections, plotting the graphs went smoothly but our initial estimate was far from reality due to our error in understanding.

* Testing

To test our code we created a separate data set called 'tester.py' which is stored in '/data_test'. This data set is similar to the real game records that were used to analyze and plot results. However, it was modified specifically for testing purposes. If you're interested in rebuilding the test set, please refer to the doc-comment in the 'test.py' folder.

Upon rebuilding the test set, we validated the test results by comparing and using the spreadsheet test which could be found here: https://docs.google.com/spreadsheets/d/136dxtK0nFMaxDbPqj14jztwnFK-LmqMQ5OWpbbE8H30

After using 'tester.py' to build our test data set, we wrote code that allowed us to run the entire project with the test data in substitution of our real-world data. To execute the project with the test data set, allocate the `data_depot.py` class with a `test_mode=True` parameter.

```
15  def main():
16      """
17      Main method of the final project
18      It dispatches all questions and saves its results.
19      """
20      # Initialize the data depot class
21      print("- Initializing data depot")
22      data_depot = DataDepot(test_mode=False, rebuild=False)
```

In test mode, we made it so that the research results won't be saved to the disk in non-test mode, and the 'Rebuild SQL to CSV' option will also be ignored. This decision was made to convenience testing purposes.

We created a smaller data set that you could use to test our work. Furthermore, we gave clients an option where they could even validate the results of the smaller data set. Moreover, we gave clients the power to test our entire project using this test data set including all of its methods for analysis and plotting. Therefore, not only can clients execute our project using the real world data, they can also validate and analyze the results from the test data.

Thus, our work is credible because we created a smaller data set which reflects the real world data set and give you the option to not only test the smaller data set but also the real world data set. By having access to both of these data sets, us programmers and the client can compare the results from executing the program on these two data sets and can compare them to find any errors or irregularities.

* Collaboration

Work Reference:
We referred to lots of external sources during our work. Among them, we got a lot of inspiration from the materials below.

* Hyperparameter Tuning of Decision Tree Classifier Using GridSearchCV
https://ai.plainenglish.io/hyperparameter-tuning-of-decision-tree-classifier-using-gridsearchcv-2a6ebcaffeda

* Python > sklearn - Classification
https://hyemin-kim.github.io/2020/07/26/S-Python-sklearn2/

* Data-driven Advice for Applying Machine Learning to Bioinformatics Problems
https://arxiv.org/abs/1708.05070

Game records - https://www.vlr.gg/
This is the source of our original dataset which was scraped. We used this website to help us interpret the results of the analysis of our dataset and validate the actual data to find any anomalies or errors.