Using Data Analytics in CS:GO

-Chris Turner

Premise

Esports are a field that has seen rising growth in popularity from both viewers and participants in recent years. Due to the nature of Esports revolving around video games that are in ongoing development, there is a wide breadth of data available which can be used by competitors and developers alike to improve their play or make balance changes to their game. Therefore, having an understanding of data analytics and using big data tools and algorithms can be important for all.

CS:GO

Counter-Strike: Global Offensive, or CS:GO as it is referred to, is one of the most popular online multiplayer games in the world. It is a team based first-person shooter that focuses on tactical positioning, team communication, reaction time, and split second decision making. Both teams seek to eliminate the opposition, but the Terrorist side additionally seeks to make their way to specific control points on the map to arm and defend a bomb. The Counter Terrorist side attempts to defend these control points and disarm the bombs before they detonate if they have been placed. The game is played for 30 rounds until one team has won 16 rounds, and after the 15th round both teams swap sides.

Dataset

I will be using a dataset provided by kaggle user “skihikingkevin” containing data of over 1400 matchmaking games and 8000 ESEA games. For the purpose of this project, I will be using data from the ESEA games as those are from a competitive league. These games take place on 7 of the playable maps in CS:GO, and have data recorded from each round of every game.

Dataset breakdown

Our dataset is comprised of 4 separate tables, containing relevant statistics from each round of each game.

- “esea_meta_demos_part1” - contains information on the map, the winning team and side, and economy stats of each round recorded.
- “esea_master_kills_demos_part1” - contains information on the kills performed by each team and side during each round as well as the status of the bomb.
- “esea_master_grenades_demos_part1” - contains information on the position, type, and damage inflicted of each grenade thrown in a round, as well as the position of the thrower and victim.
- “esea_master_dmg_demos_part1” - contains information on each damaging event happening in a round, including the attacker and victim’s team, side, position, and weapon.

Dataset breakdown

Due to our data being spread among multiple tables, we will need to selectively filter and join our data together before we can use our analytic tools and algorithms. Each game is signified by a ‘file’ value, each ‘round’ is indicated by an integer, and every event be it a kill, damage event, or grenade explosion happens on a unique ‘tick’ value. With these three values, we are able to join our data together across our tables.

NaiveBayes Classifier

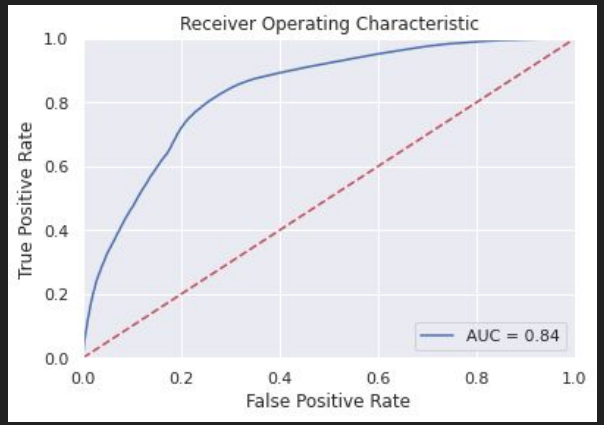
- NaiveBayes is a machine learning technique which creates a model based on the probability of an outcome based on multiple features.
- In our case, we can use NaiveBayes to create a model which determines the probability of one side winning a round over the other given a few features of the game.

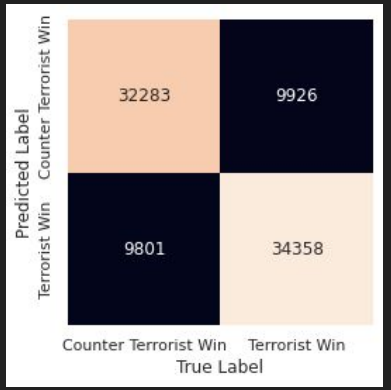
Determining Features

- Looking at our dataset and what we know about CS:GO, we can quickly determine that we can use the winning side (either Terrorist or Counter-Terrorist) as the outcome for our model. We then need to determine which other features are worth including.
- I decided to first use Round and Map as features, then later added the economy stats of both sides from the beginning of each round and whether or not the bomb had been placed during that round. Adding these features improved the performance of my model.

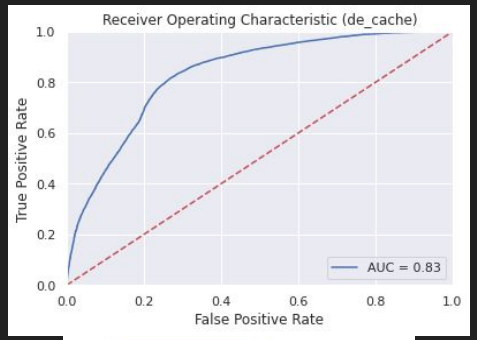
Overall Results

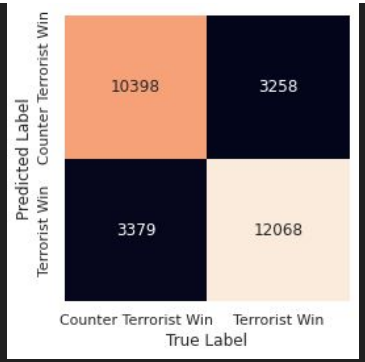
- The model's results showed an 0.84 AUC with an accuracy of 0.77.
- However this is taking all games played into account. If we were to run the model taking only the games played on particular maps into account, we can see if our model holds up equally for the various maps that CS:GO is played on.

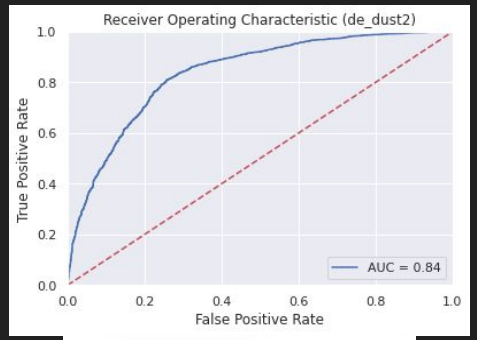


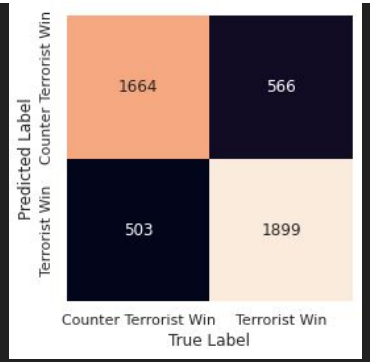


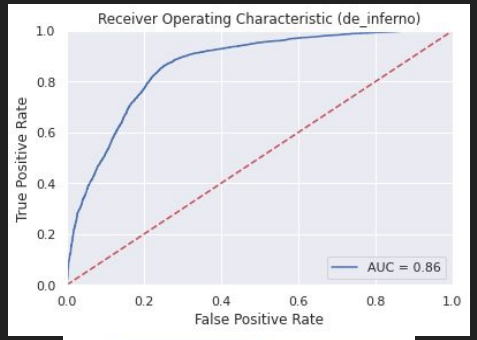
NaiveBayes - Maps (1/2)

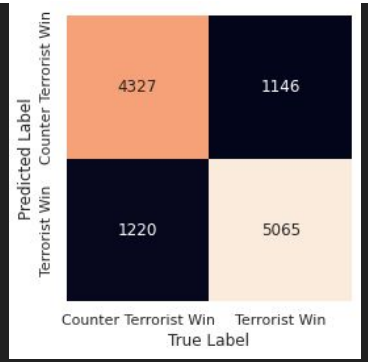


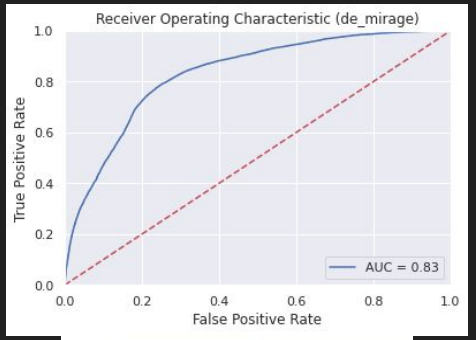




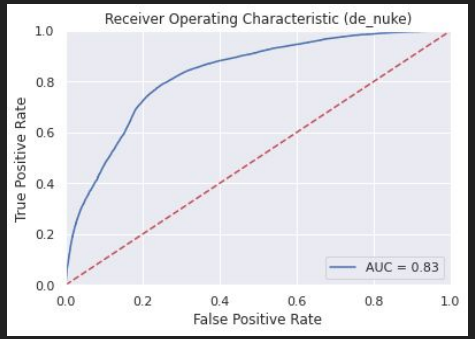


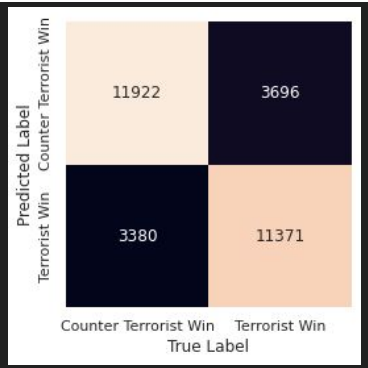


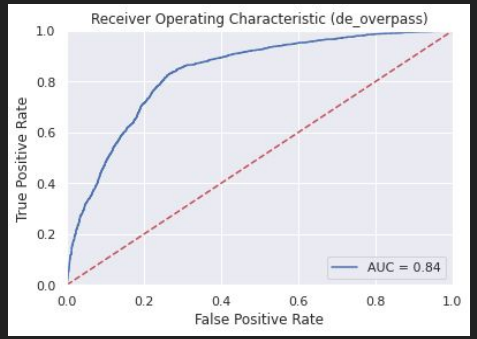


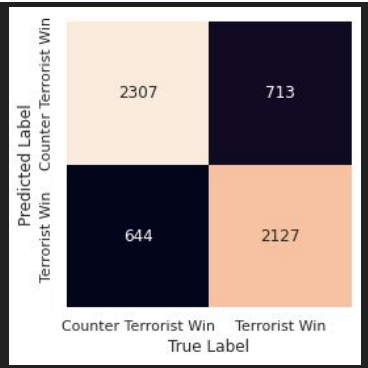


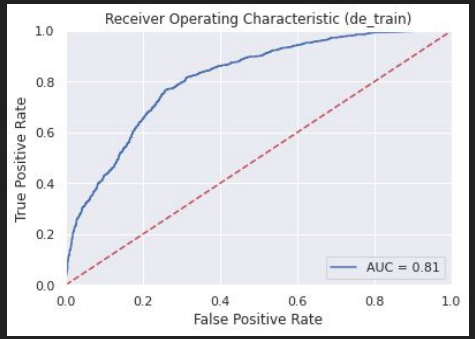
NaiveBayes - Maps (2/2)

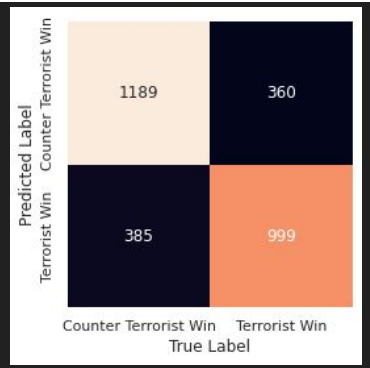






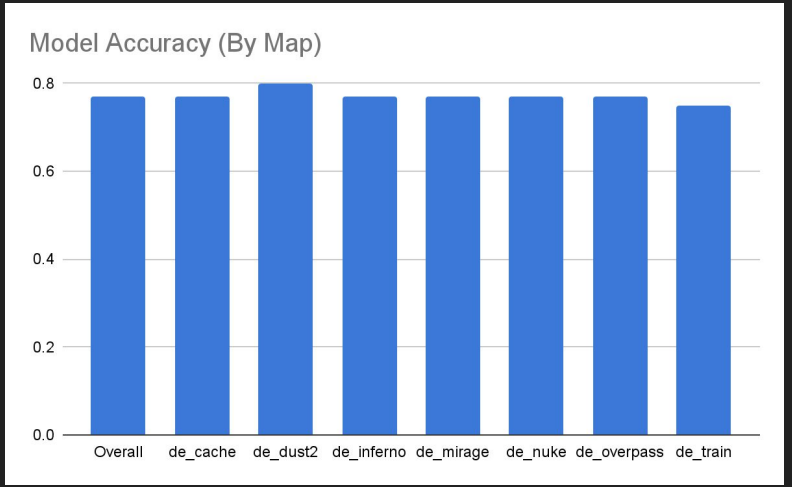






NaiveBayes - Results

So as we can see, our model stays pretty consistent across the 7 maps that were played on in terms of AUC and accuracy. Our largest variance between maps was only 0.05 in both AUC and accuracy.

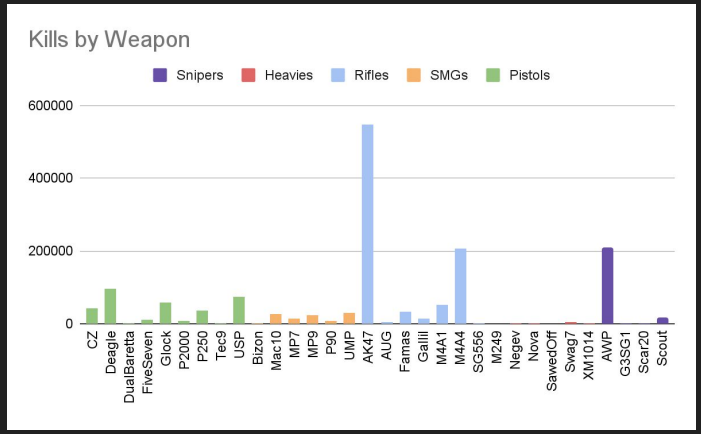


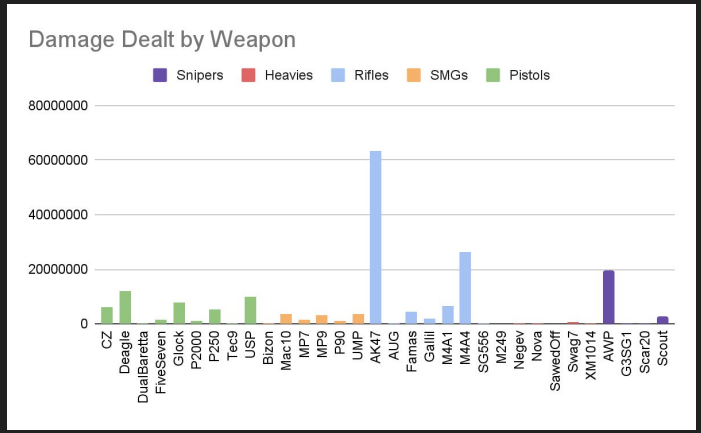
Analyzing Data

When it comes to developing a competitive game, looking at player data can allow you understand what is over or underperforming and help you decide what things need to be improved or toned down. If we want to see how the weapons in CS:GO are performing, we can use the kills secured and damage done by those weapons to determine how those weapons perform and how popular they are with players.

Analyzing Data

Looking at the data, we see that there are clear over and underperformers between weapon types and overall. In CS:GO players purchase weapons at the start of each round using money earned from their performance in the prior round. The data shows that the rifle category has two outliers that are more popular and vastly outperform the others in their weapon type as well as all other types. Additionally, the Heavy type is underperforming compared to all other types. If we were to make balancing changes with this data, we could be led to reduce the power of our largest performer, while increasing the power of Heavy weapon type as a whole.



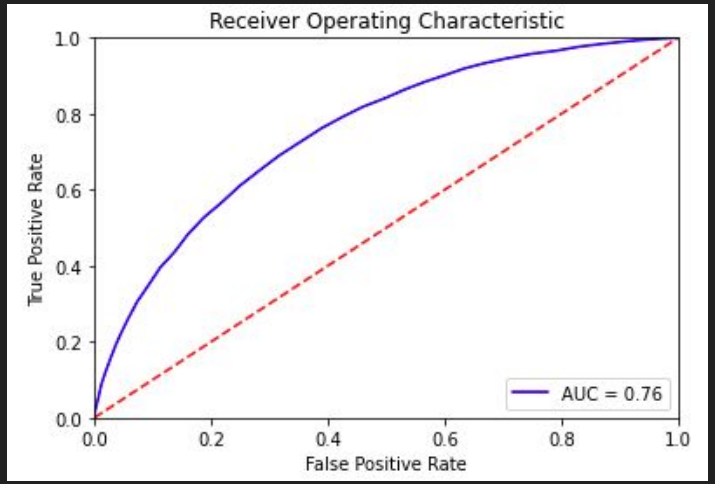


kNN Classification

- kNN or k–nearest neighbors is an algorithm that can be used to classify an object based on its nearest neighbors.
- In our dataset, we have access to all the kills the players secured in each round of each game. The positional data of the player who made those kills can also be found.
- With this I intend to use kNN to create a model that can predict whether a kill at a location is more likely to belong to one side or the other.
- And once I have this model, I will use it to create a “control map” which will indicate which areas of each map a given side has more control over.

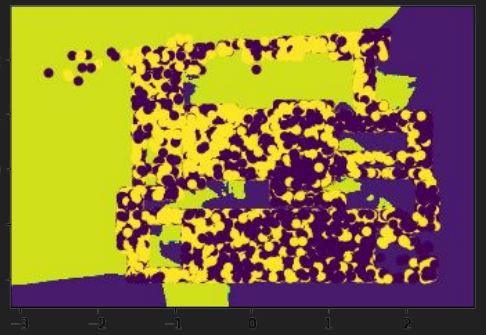
The Model

- When running the kNN model, we see an average AUC of 0.76 and accuracy of 0.7 across the various maps.
- This is fairly accurate, and will serve the purpose of our goal of making “control maps” fine.



Control Map

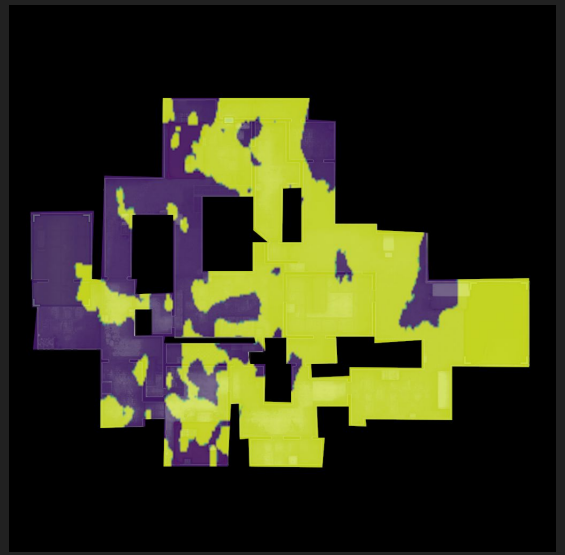
- First we generate our decision boundary using our model. We then scatter plot our kills from our model on that boundary.
- You can see that this creates a noticeable outline roughly matching our map.

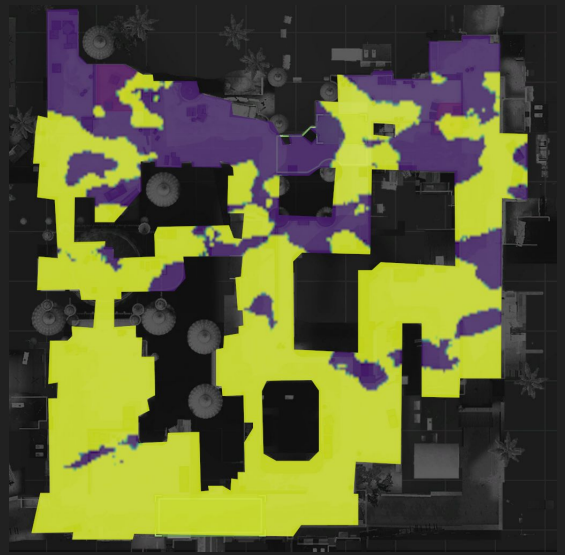


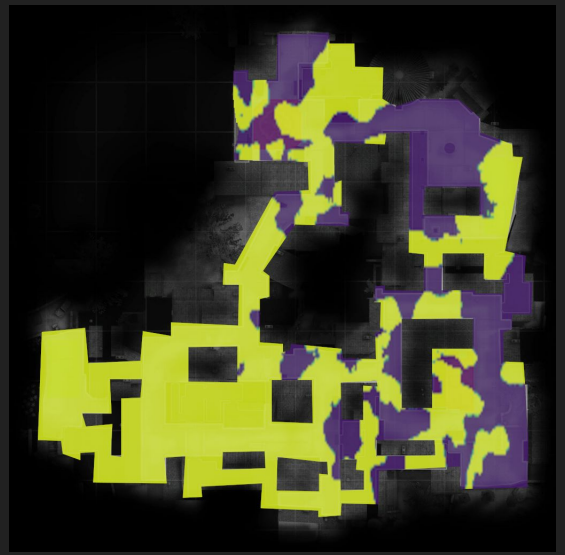
Control Map

- Now that we have the two boundaries, we can use the one with the scatter plot to find the correct scaling factor to align it with the image of our map.
- We use that scaling factor on our regular boundary, and overlay it on top of our map image. We then crop out the areas outside of the playable area of the map and we are left with this control map image as the result.

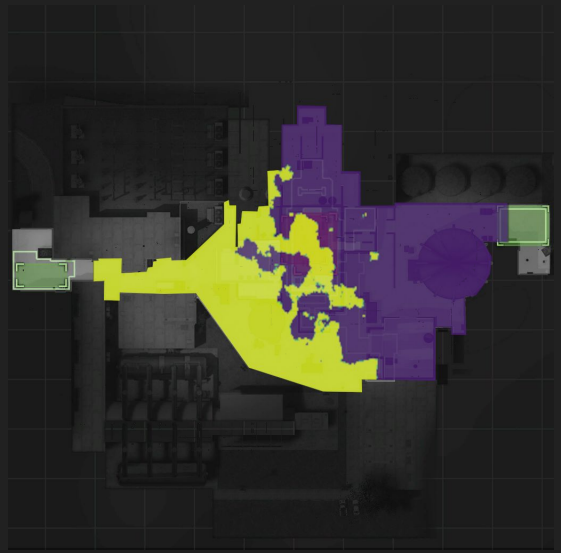
We repeat this for each of our playable maps to get unique “control maps” for all of them. The purple represents areas the Counter-Terrorist team has secured more kills and therefore has more control, and the yellow-green areas represent the Terrorist team’s control.

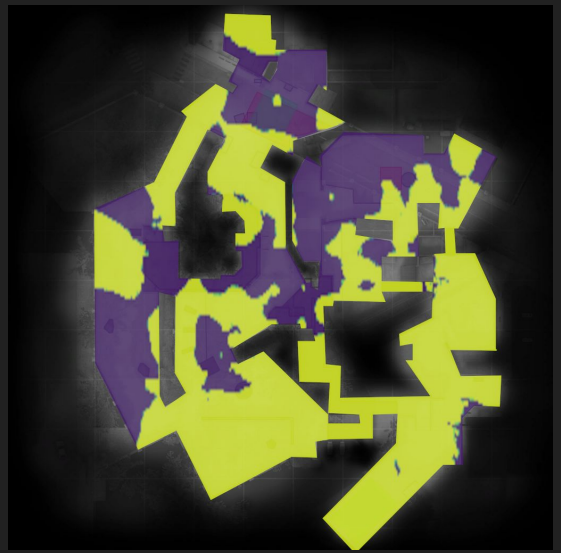






Understanding the areas of the map that one side has better control of can help players and teams improve their own play or develop new strategies, or can help the developers of the game better understand how their maps are being played and whether or not one side has an advantage.





Conclusion

We can clearly see the benefits in utilizing data analytic tools and algorithms in the space of Esports and video games. Our use of NaiveBayes demonstrates the ability to predict a winner of a round based on various criteria, and can be used by competitors who wish to improve their chances of winning. Analyzing weapon usage can help developers make decisions on improving or reducing the power of aspects in their games, while also gaining insight into how the player base is interacting with those aspects. And using kNN in a slightly unconventional way can create a tool to better understand the balance of the maps between the two sides.

References

- CS:GO Kaggle Dataset - https://www.kaggle.com/datasets/skihikingkevin/csgo-matchmaking-damage/data
- Github Containing Code - https://github.com/cwdturner/CSGO_Data_Analytics