APPLYING M.L. TO E-SPORTS GAMBLING IN CS:GO

Alec Warren, Noh Woldeyesus, Roberto Moreno, Tyler Nguyen

ABSTRACT

E-sports match results are incredibly difficult to predict, however, that doesn't stop 6.5 million people from gambling on these matches every year (Eilers & Krejcik, 2020). The goal of this project is to explore machine learning models for both a classification and a regression task. The classification task is to predict the winner of CS:GO match given each team's recent performance metrics, and the regression tasks is to predict how many more rounds the winner of a match will win vs compared to the losing team, aka the "round win difference", also using recent performance data, but also with the info of which team is presumed to be the winner. To conduct accurate research and analysis on the most recent data from CS:GO professional E-sports games, a web scraper was implemented to take data directly from the HLTV website. Multiple algorithmic and data models were implemented in order to analyze the patterns within the data including a k-neighbors classifier, a decision tree classifier, and a random forest classifier. The best model for predicting the winner of a match was the K-Nearest Neighbors Classifier, and the Support Vector Machine Classifier

INTRODUCTION

Background

Videogames as a sport, branded now as E-sports, is a relatively new concept when compared to the traditionally more physically demanding sports. Previously popular titles that had large e-sport followings like "League of Legends" and "Overwatch" featured constant changes to the game, thus rewarding the teams most capable at adaptation. As a result of this, predictions and gambling is a rather straightforward process in these aforementioned titles. Historical win-rates can be taken at their face value, and the odds usually will reflect this. In the game Counter-Strike, the subject of this particular project, the landscape rarely changes. The first Counter-Strike title was released in 2000 and the game, despite seeing many visual changes, has maintained the same core mechanics. Competitive players, for the span of 22 years have been able to practice, unlike traditional sports, for over 10 hours a day. The result of this at the highest competitive level is the pinnacle of human reaction time and precision. This tight level of competition makes CS:GO an incredibly difficult game to speculate on, as the margin of skill between teams is incredibly small.

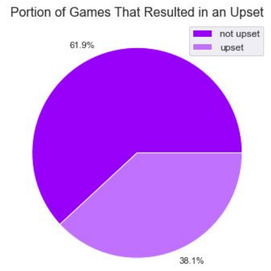
Purpose

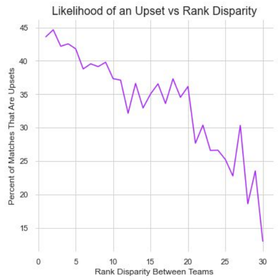
The purpose of this project is to use machine learning algorithms as a means of providing a competitive edge in the setting of CS:GO gambling. Through this data analysis, the aim is to be able to predict the outcome of future matchups between any team – hopefully uncovering hidden patterns and possible risks that might not be obvious from a traditional gambling perspective.

Goals

For the match winner predictor classification models, the expected performance of the winner predictor models was a prediction accuracy of roughly 55%, such that it is has statistically significant advantage over randomly guessing. Although this seems like an extremely low bar to clear, it is incredibly challenging to predict the winner of a CS:GO pro match because the margin of skill difference between the the professional teams is extremely narrow. And for the regression task, an average error of 3-4 rounds is the target performance for this project.

Surprising Frequency of Upsets in Pro CS:GO Matches





Methods

Web Crawling & Scraping

For the source of our data, we have chosen to leverage HLTV's website since they have an unfathomable wealth of data and statistics, dating all the way back to CS:GO's inception. The next step was to build a custom web scraper that would allow us to gather and tailor recent data to use as testing data for our prediction models. Gathering all the data and performing the initial EDA required 3 different web scraper python scripts, as well as one jupyter notebook file which aggregated all the scraped data into one comprehensive dataset (matches_dataset.csv).

EDA

The a significant portion of the necessary preparing of the data for EDA was performed during the collection of data via the web scrapers. Null values had to be handled properly, and many variables/columns were incorrectly typed or had mixed types. This was all extremely easy to do, however, thanks to the manner in which the data was scraped and stored for the dataset.

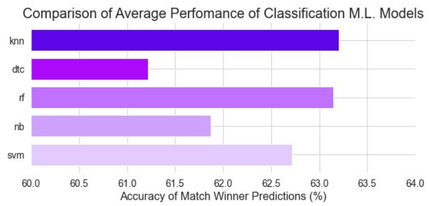
Processing Data for the Models

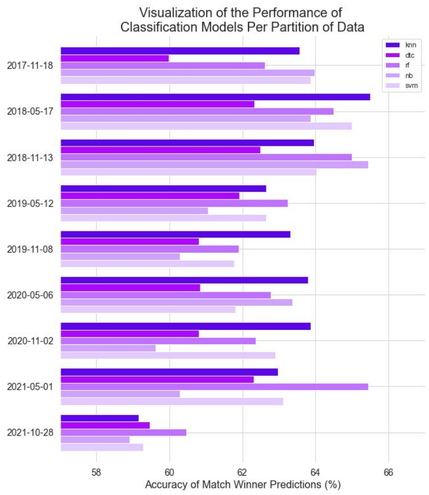
For this project, the the models have to make predictions for a match based on recent performance statistics for each of the two teams. Thus, a "performance profile" for each team and its players during a given time period is created by aggregating the stats from all games they played within this time period. And so, this method of training and testing the data involved partitioning the data temporally into distinct 6-month chunks, and then creating the aggregated "profile" for each team during this time period, to effectively measuring each teams and their players' historical performance during this time period and then using this "profile" to predict the outcome of a match that falls within this time period.

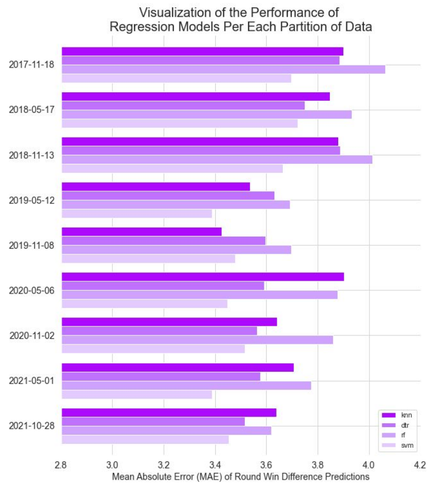
Training & Tuning The Models

Originally, three different classification models were explored: K-Nearest Neighbors, Decision Trees, and Random Forest, for both classification and regression. And then later, a Gaussian Naive Bayes Classifier and an SVM Classifier were implemented for the winner predictor task, followed by implementing an SVM Regressor for the round win difference predictor task. Using the sklearn GridSearchCV function, each model went through hyper-parameter tuning on 9 distinct 6-month-long partitions of data - each with roughly 5-7 thousand matches worth of data. GridSearchCV allowed for tuning the hyper-parameters while also performing cross-validation, ensuring the optimal settings and true performance for each model was reached, for each different partition of data.

Results







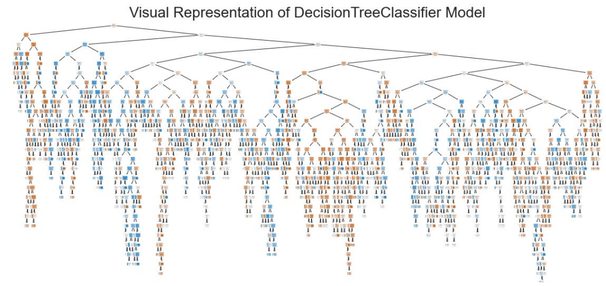
CONCLUSION

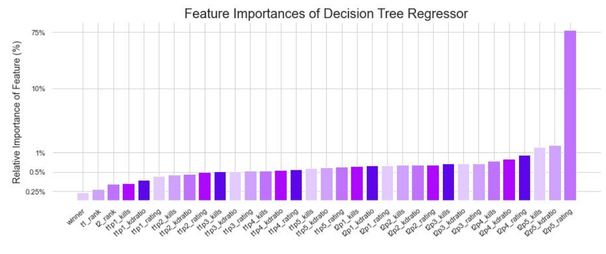
Match Winner Predictor Models (Classifiers)

All classification models exceeded the minimum performance expectation of 55% accuracy. The K-Nearest Neighbors model was the best performing classifier with 63% accuracy, followed closely by both the Random Forest model and the Support Vector Machine model with 62% accuracy. Despite these models all having extremely close averages, the KNN model is the best model because it was the most consistent, and was much faster to train and perform hyper-parameter tuning on compared to the SVM and RF models, which require several times more time to train.

Round Win Difference Predictor Models (Regressors)

Similarly, the regression models met the expected performance of 3-4 rounds of MAE (Mean Absolute Error). MAE is used as the performance metric because it is similar RMSE but less punishing for outlier results. The Best performing model was clearly the Support Vector Machine regressor with an average MAE of roughly 3.5 rounds. Despite taking the longest to tune and train and having the best performance during tuning, the Random Forest regressor was by far the worst performing on the test sets of data with an average MAE of just over 3.8 rounds.





Analyzing Decision Tree Visualizations

Major Takeaways

By looking at the visualization of the decision trees for both models, it is clear that the trees are overly complex. The feature importance graphs also illustrate how some features are much less useful than other, especially for the regression decision trees, where over 75% of the importance falls on one feature, the rating of the worst player on lower seeded team (t2p5_rating). This is by far the most surprising result from this entire experiment.

Future Work & Improvements

Despite our models' performing well, and even exceeding expectations, there are still many improvements that could be made given more time. There was unfortunately insufficient time remaining to implement these, due to needing to dedicate so much time and energy to processing and aggregating our data such that it could be used to predict both past and future matches. And thus, one major 'feature' in our dataset that was not leveraged in the models is the actual map that each game is played on. Adding the ability to leverage this categorical data would definitely be the single most impactful improvement that could be done to the models in future versions of this project. Additionally, the models currently use stats from all players as features, but this could be changed such that only the players that impact the model in a significant way (i.e. the worst players on each team) are have their stats used as features.