Counter-Strike: Global Offensive Match Predictor

CICS 397A Final Project

Matthew Hrones

December 4, 2020

**Introduction**

The goal for this project was to build a model to predict the outcome of professional matches of the First-Person Shooter video game *Counter-Strike: Global Offensive*, or CS:GO. It is a computer mouse-and-keyboard game, where players are divided into two teams of five players: “Terrorists” (T) and “Counter-Terrorists” (CT). Each match is broken up into “Rounds,” lasting around two minutes each. The Terrorist’s objective to plant a bomb at one of two bombsites and guard it until it detonates, or eliminate every Counter-Terrorist. The Counter-Terrorists, meanwhile, are trying to prevent the Terrorists from planting the bomb, or defuse it if the Ts manage to plant the bomb. Alternatively, they could eliminate every Terrorist to win the round. The match consists of a maximum of 30 rounds, and winning one will get that team a point on the board, and the first team to reach 16 rounds wins the game. Teams switch sides at half times and ties will go to overtime until one team wins, hence no ties can occur.

The goal of this model, as said before, will be to predict the outcome of matches. A good indicator for which team will win the game is simply which team has the greater difference between Kills and Deaths, or “Kill / Death Diff”. Whichever team ends up getting more kills compared to deaths is more likely to win the game.

The model uses statistics and information from the first stages of an ongoing CS:GO tournament, *Dreamhack Winter 2020*. This tournament features 16 top teams from around Europe, lasting from November 30th to December 6th. Original collection consisted only the group stage games and the players who participated. This model was re-trained over multiple days as new match concluded and their results were released. Hence, the model made predictions on Thursday’s matches using only the data from the group stage. After the matches on Thursday, and the predictions were compared to the true outcome, the model was re-trained using the group stage matches and the matches from Thursday, and then used to make predictions on Friday’s matches. After another comparison, the model was re-trained with all matches that have happened this week in order to make predictions on the Saturday’s matches, which we do not have results for as of the time I’m writing this.

**Data Overview**

The original collection of the data set was done just after the group stage of the tournament had been completed, but has been updated as the tournament has progressed over the past few days. Data was web-scraped using the Google Chrome Driver from <https://www.hltv.org/>.

The two main CSV files are “dreamhack2020\_players.CSV” and “dreamhack2020\_matches.CSV”. Both files contain the match information for previous matches in the tournament. The matches.CSV contains information such as the date of the match, the teams that played, the results, and most importantly a link to the statistics of the players from that match. The players.CSV has the stats for each player from that match, and additionally contains some all-time statistics for the player. Some of these stats include “Kill/Death Ratio,” “Kill/Round Ratio,” etc. Additionally, in this data set lies the all-important “Kill / Death Diff,” which will end up being the target for our model. The rest of the data in this CSV will later go on to be training data. The other CSV files that are included either were upcoming matches or are upcoming matches as of the Friday this is due.

**Code Overview**

The code is broken up into three main files, cs\_scraper.py, cs\_futurematches.py, and cs\_predictor.py. cs\_scraper.py and cs\_futurematches.py function similarly, but the main difference is that cs\_scraper.py is for past matches where as cs\_futurematches.py is for, well you guessed it, future matches.

These programs initially take in a set of match links, given by the user. They then use the Google Chrome webdriver to scrape information off the match page. Initially it takes the base match stats from the page. The cs\_scraper.py program uniquely will find which maps were played in that game, and collect all statistical results from that game. Both programs will then go on to find the historic statistics for that player, and save them to two CSV files: one containing match data, and one containing player data. Both programs will run for every match link that is put into the match links list. The differences are that scraper will produce a CSV will match results, and player performance for that game. The cs\_ futuregames.py file will simply get the teams that are to play, the date, and the tournament. Both files collect historical statistics for players on each team.

Then, we have cs\_predictor.py. This program firstly creates a unique identifier column to cross reference between the match and player CSV files. Next, it finds the total net KD for each match in the training data, as that is what we will use to determine who won the match. Then, the program re-encodes some of the training data, such as encoding which team won to a binary variable, and assigning each player and team a unique integer value. and then trains a linear regression model around it, targeting each player’s net KD. We find the regression accuracy and the mean squared error, the move on to predict future matches!

The program then takes in some CSV file of upcoming matches, and predicts their outcome. It performs the same re-encoding process, and prints the final predictions to a new CSV file.

**Outputs**

This algorithm was re-trained over multiple days, so these results the include the accuracy of the prediction model as more data is added. The outputs are included in the ‘Predictions’ folder. The name of the folder the output resides is “one less” than the amount of data that the model had to predict it. For example, the prediction in the Thursday folder was predicted using only group stage data, the ‘fri’ prediction was made using the data from the group stage and Thursdays matches, etc.

On the first iteration, the model had an accuracy of 0.5485, with a MSE of 6.2467. Upon adding Thursday’s match data to the training data, the model’s accuracy went to 0.5726, with a MSE of 5.7702. We have marginal improvement! However, when adding Friday’s match data, our accuracy decreased to 0.5315, with an MSE of 6.8865. Our model had curiously decreased in accuracy as more data came in.

However, the model that achieved MSE of 5.7702 was not half bad at predicting matches! It predicted one upset and two favorite wins. The one upset it predicted that did not turn out was with a team that had only played two games and gone straight to the quarterfinals, so the model did not have much information for that team. But not bad for the second go around!

**Future Explorations**

While the algorithm current can collect the data, none of these models currently use the map to train the data. I would be super interested in seeing how, given a map, the model can predict net KD. I would also like to see if my model can work with a larger scope of teams with a wider data set, to potentially predict to some criterion such as “All games of teams in the top 20.” I’d like to also look for more optimal ways to predict games, potentially finding a more deterministic statistic in the game.

**Instructions on Running Code**

The submission is distributed across four folders: The code, Input data, Predictions, and Upcoming matches. Each folder has CSV files in a format either a) ready to be used by the code or b) outputted by the code. Where you can use which CSVs will be in the instructions below. One thing, which is slightly obvious, don’t run any ‘upcoming match’ with a set of data that includes those matches. For example, don’t predict Friday’s game with the model that uses the training data that includes Friday’s matches (dreamhack2020\_players\_after\_fri).

* **cs\_scraper.py and future\_matches.py:** Two User Inputs Required
  + List of match links that can be assigned to the ‘match\_links’ variable, Examples for matches, and the required format, are in code around referenced line
    - cs\_scraper.py:123 , cs\_futurematches.py:121
  + CSV output match and player data, */INPUT DATA/*
    - cs\_scraper.py:457 , cs\_futurematches.py:317
* **cs\_predictor.py:** Three User Inputs Required
  + Training Data CSVs */INPUT DATA/*
    - cs\_predictor.py:26-27
  + Matches to be predicted */UPCOMING MATCHES/*
    - cs\_predictor.py:241-242
  + Prediction Outputs */PREDICTIONS/*
    - cs\_predictor:356-357