CS430 Project: An Analysis of the Final Sixteen 2021 CS:GO Major Teams

Ralf Leggett
u1801916

***Abstract*—CS:GO is a competitive team-based video game that offers a rich source of noiseless data for analysis. In recent years, teams competing at the highest level have increasingly used data analytics to improve their performance. This project investigates how well readily-available statistics on past games translate into the ability to predict the outcomes of rounds within games and the games themselves. To do this, a scraper was implemented to produce a dataset on sixteen teams. Initial results indicate that predictions with reasonable accuracy are possible, but more data is required to generate models with good accuracies.**

1. Introduction

CS:GO is a team-based competitive video game based on a counter-terrorism scenario. Matches between teams may be a best of 1, 3 or 5 games, with each game taking place on a different map. Games consist of rounds played between the counter-terrorists (CTs) and terrorists (Ts) with the winner being the first team to 16 rounds. The two teams take turns playing as the CTs or Ts, swapping after 15 rounds have been played.

Over the 20 years that the franchise has existed, teams have increasing employed statistics and analytics, transforming it from being a game solely about who is better at clicking heads to one in which highly-complex strategies and in-depth analysis are necessary to compete at the top level. Indeed many teams employ people specifically to analyse their own squad as well as their opponents' to identify weaknesses and come up with new strategies or set-pieces.

CS:GO, and video games in general, are particularly appealing for data analysis due to their closed nature and the absolute control possible over them. There are almost no external factors to consider and gathering data is quick, cheap and noiseless. It is, however, very difficult to capture the complexity and nuances of the game in statistics. The main aim of this project is to examine how well statistics on teams and players translate into the ability to predict the outcome of games. Data from the final sixteen teams of the 2021 CS:GO Major, the biggest tournament in the last couple of years, will be used. The data on the Major will form the test set, with data on previous events forming the training set. This report will assume the reader has no knowledge of the game; relevant information will be introduced where necessary.

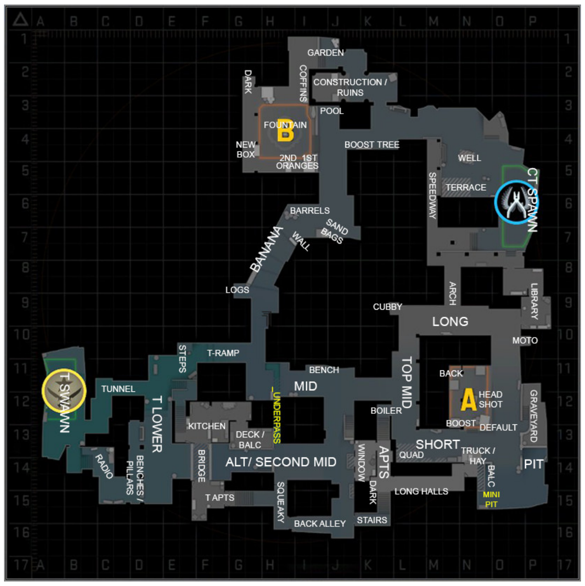


Figure 1: Inferno, a map in CS:GO, including callouts for different areas. "T-spawn" and "CT-spawn" are where the Ts and CTs start each round. "Top mid" and "banana" are the two most contested areas, being the main entrances for the Ts into the A and B bombsites respectively.

1.1. A Brief Introduction to CS:GO

CS:GO is played between two 5-man teams who play as either the CTs or Ts and swap after 15 rounds. Each round, the Ts' objective is to plant the bomb on one of two bombsites (A or B) before the round timer expires and defend the bomb until it explodes. The CTs' objective is to defuse the planted bomb before it explodes. Both sides can also win the round by eliminating the other side before either side completes their objective. Players cannot heal during a round; if they die, they will be respawned in the following round. If a game goes to 15-15, overtime is played, which is akin to extra time in football. Teams play 3-round halves, swapping sides (CT/T) at the half. The first to win 4 rounds wins the game. If overtime goes to 3-3, another overtime is played ad infinitum until one team wins 4 rounds in an overtime.

Winning a round grants players a large amount of money. Losing a round grants them much less, although losing consecutive rounds grants larger amounts to try to get the losing team back in the game. The Ts also gain a modest sum for planting the bomb, even if they do not win the round. Money is used to buy equipment including armour, weapons and utility (such as flashbangs). Any players that survive a round carry their remaining equipment into the next round. Economy management is crucial to the game, as even the money for winning a round is not enough to buy all the equipment required. Teams often concede rounds in unfavourable scenarios to save their equipment and have a better shot in the following round.

One of the most appealing aspects of the game as a viewer is the way teams work with the limited information available. For the CTs, 5 men must be spread across the map to defend two bombsites. Playing without information will always result in a unfavourable fight against 5 Ts taking a bombsite so the CTs must find out which bombsite the Ts will take and move manpower accordingly. The Ts aim to deny map control, and hence information, and push the CTs back into the bombsites without losing their own players.

2. The Dataset

Although CS:GO datasets were found on Kaggle, they lacked in many ways. Datasets were were either too old, did not contain enough detailed information, or included data on many lower-tier teams that do not make heavy use of analysis and so are less interesting for data analysis. The decision was made to create a new dataset gathered from HLTV.org[1], an independent news and statistics-gathering company for CS:GO. This was done from scratch with a scraper that limits requests to a maximum of 2 per second coded in Python, with the help of the requests[2] and beautiful soup[3] libraries.

2.1. Scraping HLTV

HLTV creates unique ids for every data object and conveniently places them in the url. This allowed creating a scraper with relative ease. The Python script written to scrape HLTV generates dictionaries, which can be thought of as tables, as shown in Figure 2. There is some redundancy in the tables for ease of data analysis.

First, the ids and names of the final sixteen teams were scraped. Following this, the ids and names of the players that played in the Major for each of these teams were scraped. This allowed creating the “team” and “player” tables.

It was then possible to scrape every game played between each of the teams while ensuring the rosters that played for each team were the same as those that played in the Major, using one of the functions provided by HLTV. Analysis of the data gathered at this point revealed that one of the teams - Virtus.Pro - had made a roster change specifically for the Major. This meant that there would be no previous games that could be used as the training set for this team.

The decision was made to scrape every game played between each of the teams while ensuring that at least 4 out of the 5 players on each team were the same as those that played in the Major. This decision balances the size of the

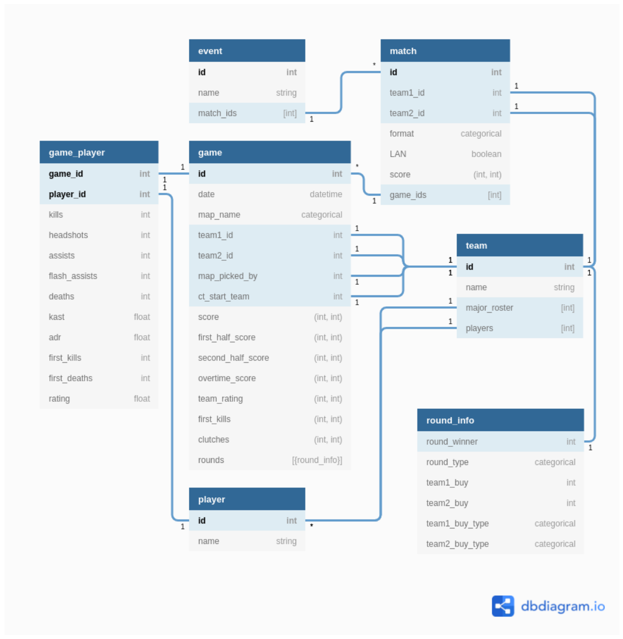


Figure 2: Diagram showing relations between each of the scraped tables.

dataset available with ensuring the data is still relevant, as the core of the team is still the same. This resulted in the “game” table. Note that the “round” attribute of the “game” table is a list of dictionaries containing information on each round, shown in the diagram as “round_info”. This step also required scraping information on the new 5th players to update the “player” table.

The matches that each game belonged to were scraped to create the “match” table, as well as the events the matches belonged to to make the “event” table. Statistics on each player for each game were also gathered in the “game_player” table. The total dataset in json format is around 8.5mb.

2.2. Data Cleaning

There are a couple of notable data cleaning tasks that were carried out on the dataset. Games that did not follow the usual MR16 format (first to 16 rounds), such as tie-breakers and some charity games, were removed from the dataset. The inclusion of the “events” table allows removing matches from the dataset that belong to more casual or informal tournaments, such as CS Summit. This was not done initially due to the fairly limited amount of data on certain matchups. Also of note is that one game in the dataset does not have economy statistics as the feature had not been introduced to HLTV at the time it was played. This game was left in as the data is still useful for other tasks, but is removed when required by the task.

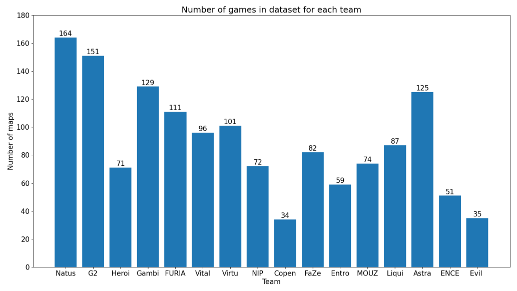
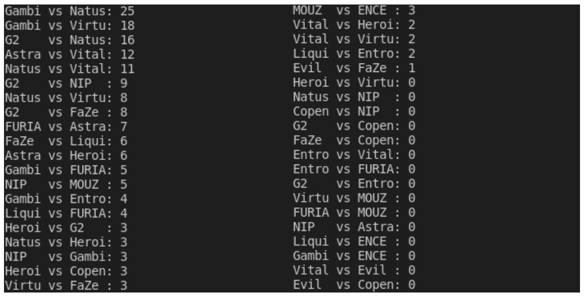


Figure 3: Frequency of teams in dataset. Team names shortened to first 5 letters.



```
Gambi vs Natus: 25                    MOUZ  vs ENCE : 3
Gambi vs Virtu: 18                    Vital vs Heroi: 2
G2    vs Natus: 16                    Vital vs Virtu: 2
Astra vs Vital: 12                    Liqui vs Entro: 2
Natus vs Vital: 11                    Evil  vs FaZe : 1
G2    vs NIP  : 9                     Heroi vs Virtu: 0
Natus vs Virtu: 8                     Natus vs NIP  : 0
G2    vs FaZe : 8                     Copen vs NIP  : 0
FURIA vs Astra: 7                     G2    vs Copen: 0
FaZe  vs Liqui: 6                     FaZe  vs Copen: 0
Astra vs Heroi: 6                     Entro vs Vital: 0
Gambi vs FURIA: 5                     Entro vs FURIA: 0
NIP   vs MOUZ : 5                     G2    vs Entro: 0
Gambi vs Entro: 4                     Virtu vs MOUZ : 0
Liqui vs FURIA: 4                     FURIA vs MOUZ : 0
Heroi vs G2   : 3                     NIP   vs Astra: 0
Natus vs Heroi: 3                     Liqui vs ENCE : 0
NIP   vs Gambi: 3                     Gambi vs ENCE : 0
Heroi vs Copen: 3                     Vital vs Evil : 0
Virtu vs FaZe : 3                     Evil  vs Copen: 0
```

Figure 4: A list showing the number of games in the training set for each of the matchups that occurred in the Major (the test set). Team names shortened to first 5 letters.

3. Analysis

3.1. Dataset Composition

Analysis began by investigating some summary statistics. Figure 3 shows how many games involve each of the teams. Immediately it is obvious that the dataset is very unbalanced, with Copenhagen Flames, Evil Geniuses and Ence being particularly under-represented. This is perhaps unsurprising, as these teams are generally considered tier-2 teams, while the rest are tier-1. This is loosely comparable to Championship vs Premiership teams in English football; teams in the two leagues will rarely face each other.

Investigating this further results in the table shown in Figure 4 that displays the frequency with which the matchups that occurred in the Major appear in the rest of the dataset. A large number of them do not occur at all in the dataset, and since there is no way to gather data on these specific matchups, the analysis for them will have to be based on games against other teams.

Finally, the distribution of games in the dataset by date was plotted, to understand the recency of the data. Since the major was played early in 2021Q4, there are fewer games in this quarter, but the distribution quickly tails off after 2020Q4 with very few games older than this, most likely due to roster changes in most, but not all teams.

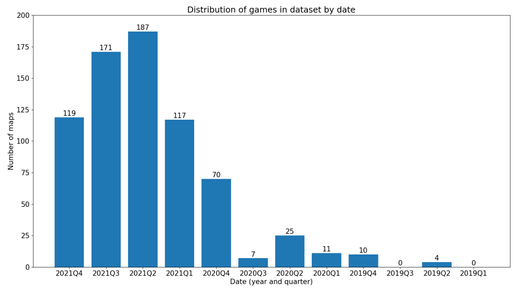


Figure 5: A barchart showing the distribution of games in the dataset by date.

3.2. Hypotheses

The following list explains the main hypotheses to be investigated with the dataset:

- Maps are generally slightly CT-biased.
- HLTV ratings primarily depend on kills, with kills in even or unfavourable scenarios being weighted more heavily; ADR (average damage per round) and KAST (percentage of rounds the player got a kill or assist, survived or won by timeout i.e. was not a liability to their team) are also important.
- Rounds can be predicted with reasonable accuracy based primarily on the value of the equipment each team has.
- Games in the 2021 CS:GO Major can be predicted with reasonable accuracy based on historic data.

3.3. Map Biases

Most maps are slightly CT-biased, i.e. the CTs generally win more rounds than the Ts. The degree of the bias varies map to map, and changes over time as teams come up with new strategies or maps are altered by the developers. To investigate this, the percentage of rounds won by the CTs and the Ts for each map was found across all rounds in the dataset. This should help average out the fact that some teams are stronger on certain maps, and that some prefer one side (CT/T) over the other. For comparison, HLTV was used to find the same statistics across all top 20 teams over the last 12 months. These parameters were chosen to match the calibre of teams and the distribution of maps by date (see Figure 5) in the dataset. The percentages are reasonably similar with surprising results for overpass, and particularly train, which are both famously CT-sided. Ancient was also much more even, albeit still CT-biased.

3.4. Predicting Ratings

HLTV's rating 2.0 is the secret formula widely used by analysts to rate a player's performance in a game. Due to the complexity of CS:GO, it is likely that the formula is calculated using lots of data, including data gathered on

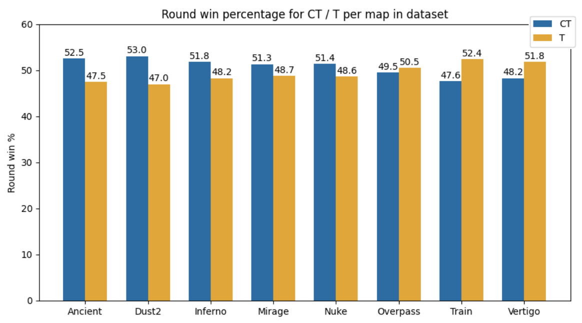


Figure 6: A barchart showing the percentage of CT and T rounds won for each map across the dataset.

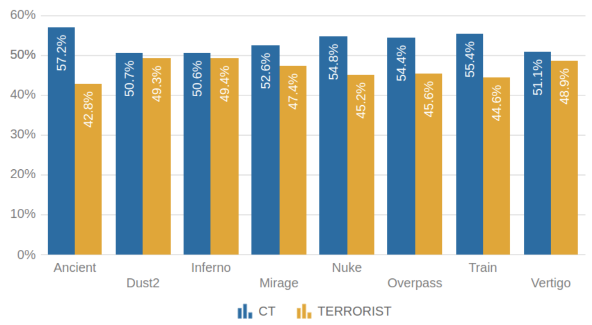


Figure 7: A barchart from HLTV showing the percentage of CT and T rounds won for each map for matches involving the top 20 teams in the last 12 months.

players within rounds. Despite this, it would be interesting to see how well players' ratings can be predicted using the much higher-level summary statistics gathered in the dataset under the "game_player" table (see Figure 2). The table was converted to .arff format to allow investigation with Weka, with the data on the major forming the test set.

The hypothesis is that kills primarily determine a player's rating. Although kills do not necessarily lead to round wins, they are probably the most impactful action a player can take in general. Not all kills are equal however, as getting the kill in a 1v1 scenario to win the round is a lot more important to winning the game than getting the final kill in a 5v1 scenario which your team is practically guaranteed to win already.

Although it was not possible to gather data on the number of players on each side at the time of each kill, the dataset does contain information on first kills, which greatly increase the odds of winning a round and are generally the most impactful kills in the game. The "first_kills" attribute is, therefore, expected to be important as well, despite the overlap with the "kills" attribute.

ADR (average damage per round) is also expected to contribute to rating predictions as doing damage to a player makes it easier to kill them. However, damage only truly matters if it is converted to a kill, as measured by the number

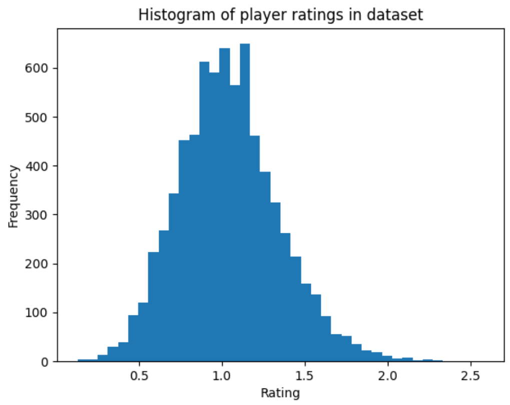


Figure 8: A histogram showing the distribution of player ratings in the dataset.

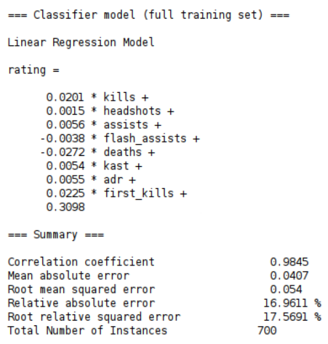
of assists which might also be useful. KAST, loosely the percentage of rounds the player was not a liability to their team, is also expected to be important.

To investigate these hypotheses, the distribution of ratings was investigated first with numpy and matplotlib, to contextualise error values for models. The mean was found to be 1.045, and the standard deviation 0.3035. The distribution (see Figure 8) is roughly normal, with a longer tail on the right side (max value 2.58) than the left (min value 0.13).

Next, Weka's attribute evaluator was used to investigate which of the attributes are most important in predicting ratings. As expected, kills, ADR, KAST and first kills all feature, but surprisingly, deaths was also important and assists did not appear.

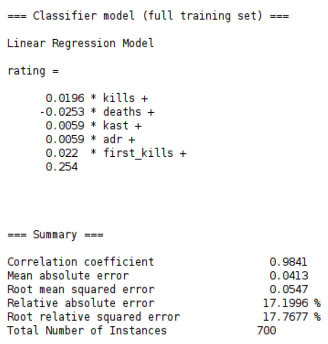
Various Weka methods and parameter were tried to find a good model with both the full set of attributes and the set of attributes selected by Weka's evaluator. First, the zeroR rule was used to generate a baseline which predicts a constant value and has a root mean squared error (RMSE) of 0.3076 and mean absolute error (MAE) of 0.2399 (instructive due to the presence of outliers).

The best-performing simple model with both the full set of attributes and selected set was linear regression. The best-performing model overall was M5P, which creates a decision tree to choose one of sixteen linear regression models to use for predicting the rating. These results are shown in Figure 9. Interestingly, the model with the full set of attributes does not use first deaths at all, and weights flash assists slightly negatively (which does not make sense). Headshots are practically irrelevant, but we see that assists is weighted about as heavily as KAST and ADR, so the hypothesis above does hold. Adding assists to the set of selected features improves the linear regression model to a MAE of 0.041 and RMSE of 0.0541 (not shown in the figure). Overall, these models perform surprisingly well given the limited information used.



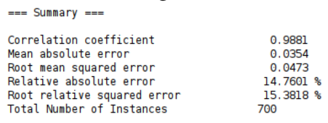
```
=== Classifier model (full training set) ===

Linear Regression Model

rating =

      0.0201 * kills +
      0.0015 * headshots +
      0.0056 * assists +
     -0.0038 * flash_assists +
     -0.0272 * deaths +
      0.0054 * kast +
      0.0055 * adr +
      0.0225 * first_kills +
      0.3098

=== Summary ===

Correlation coefficient                  0.9845
Mean absolute error                      0.0407
Root mean squared error                  0.054
Relative absolute error                 16.9611 %
Root relative squared error             17.5691 %
Total Number of Instances              700
```

(a) Linear regression (full)



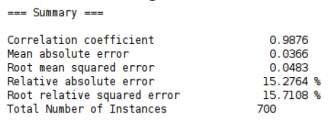
```
=== Classifier model (full training set) ===

Linear Regression Model

rating =

      0.0196 * kills +
     -0.0253 * deaths +
      0.0059 * kast +
      0.0059 * adr +
      0.022  * first_kills +
      0.254

=== Summary ===

Correlation coefficient                  0.9841
Mean absolute error                      0.0413
Root mean squared error                  0.0547
Relative absolute error                 17.1996 %
Root relative squared error             17.7677 %
Total Number of Instances              700
```

(b) Linear regression (select)



```
=== Summary ===

Correlation coefficient                  0.9881
Mean absolute error                      0.0354
Root mean squared error                  0.0473
Relative absolute error                 14.7601 %
Root relative squared error             15.3818 %
Total Number of Instances              700
```

(c) M5P (full)



```
=== Summary ===

Correlation coefficient                  0.9876
Mean absolute error                      0.0366
Root mean squared error                  0.0483
Relative absolute error                 15.2764 %
Root relative squared error             15.7108 %
Total Number of Instances              700
```

(d) M5P (select)

Figure 9: Best results from using various Weka regression models to predict players' HLTV rating with either the full attribute set (full) or the smaller set of selected attributes (select).

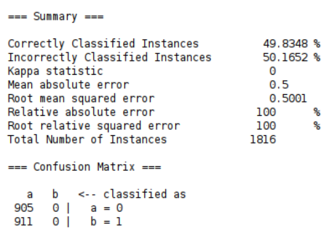
3.5. Predicting Round Wins

The greater the equipment advantage of one team over the other, the better their chances are of winning the round. Due to the map biases discussed above, and the fact that some teams are better at certain maps or better at one side (CT/T), including teams and maps as categorical features may improve predictions.

To test these hypotheses, the games played at the major were used as the test set, and the rest of the dataset as the training set. A small bit of data cleaning was performed to remove games without economy data (see subsection 2.2), and the data was converted to .arff format. The features available were the map, the CT side team name, the T side team name, the value of the CTs' equipment and the value of the Ts' equipment. Classifying the round winner as either the CTs or the Ts was the task.

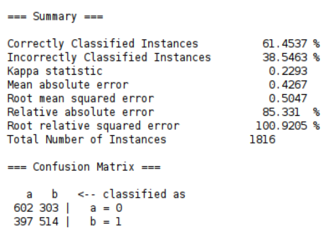
To begin, several methods available on Weka were used. The majority classifier "trained" on the training set actually does worse than 50% on the test set, but the class distribution is roughly even; for this reason, the analysis will examine accuracy only. The parameters for each of the other methods shown in Figure 10 were played around with, and the best results are shown. Nearest neighbour performs worse than the other methods, which all have accuracies between 65% and 66%. The pruned J48 decision tree does best, with an accuracy of almost 66%. Removing the map, CT team name and T name features does not degrade results much, with the pruned J48 tree achieving the best accuracy at 65.804%.

Given the map biases and differences in ability of the teams in the dataset, it is interesting to see if a neural network is able to perform better than the methods provided by Weka. The categorical features were one-hot encoded



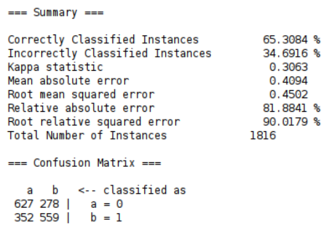
```
=== Summary ===

Correctly Classified Instances          49.8348 %
Incorrectly Classified Instances        50.1652 %
Kappa statistic                          0
Mean absolute error                      0.5
Root mean squared error                  0.5001
Relative absolute error                100      %
Root relative squared error            100      %
Total Number of Instances             1816

=== Confusion Matrix ===

   a   b   <-- classified as
 905   0 |   a = 0
 911   0 |   b = 1
```

(a) Majority classifier



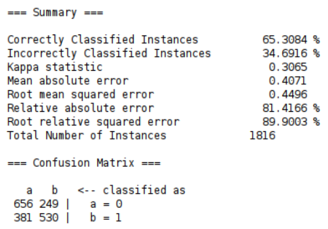
```
=== Summary ===

Correctly Classified Instances          61.4537 %
Incorrectly Classified Instances        38.5463 %
Kappa statistic                          0.2293
Mean absolute error                      0.4267
Root mean squared error                  0.5047
Relative absolute error                 85.331  %
Root relative squared error            100.9205 %
Total Number of Instances             1816

=== Confusion Matrix ===

   a   b   <-- classified as
 602 303 |   a = 0
 397 514 |   b = 1
```

(b) 5-nearest neighbour



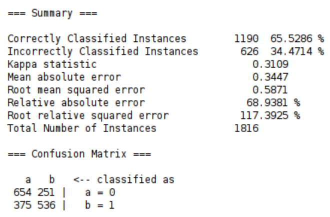
```
=== Summary ===

Correctly Classified Instances          65.3084 %
Incorrectly Classified Instances        34.6916 %
Kappa statistic                          0.3063
Mean absolute error                      0.4094
Root mean squared error                  0.4502
Relative absolute error                 81.8841 %
Root relative squared error             90.0179 %
Total Number of Instances             1816

=== Confusion Matrix ===

   a   b   <-- classified as
 627 278 |   a = 0
 352 559 |   b = 1
```

(c) Logistic regression



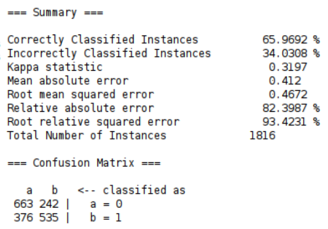
```
=== Summary ===

Correctly Classified Instances          65.3084 %
Incorrectly Classified Instances        34.6916 %
Kappa statistic                          0.3065
Mean absolute error                      0.4071
Root mean squared error                  0.4496
Relative absolute error                 81.4166 %
Root relative squared error             89.9003 %
Total Number of Instances             1816

=== Confusion Matrix ===

   a   b   <-- classified as
 656 249 |   a = 0
 381 530 |   b = 1
```

(d) Naive Bayes



```
=== Summary ===

Correctly Classified Instances        1190   65.5286 %
Incorrectly Classified Instances       626   34.4714 %
Kappa statistic                          0.3109
Mean absolute error                      0.3447
Root mean squared error                  0.5871
Relative absolute error                 68.9381 %
Root relative squared error            117.3925 %
Total Number of Instances             1816

=== Confusion Matrix ===

   a   b   <-- classified as
 654 251 |   a = 0
 375 536 |   b = 1
```

(e) SVM



```
=== Summary ===

Correctly Classified Instances          65.9692 %
Incorrectly Classified Instances        34.0308 %
Kappa statistic                          0.3197
Mean absolute error                      0.412
Root mean squared error                  0.4672
Relative absolute error                 82.3987 %
Root relative squared error             93.4231 %
Total Number of Instances             1816

=== Confusion Matrix ===

   a   b   <-- classified as
 663 242 |   a = 0
 376 535 |   b = 1
```

(f) J48 decision tree

Figure 10: Best results from using various Weka classifiers to predict the CT or T side as the round winner.

and a network built with Keras. Different architectures were experimented with, including batch normalisation layers, L1-regularisation for feature selection, depth and neurons per layer. Despite this, the best model found only barely outperformed the decision tree with an accuracy of 66.24%.

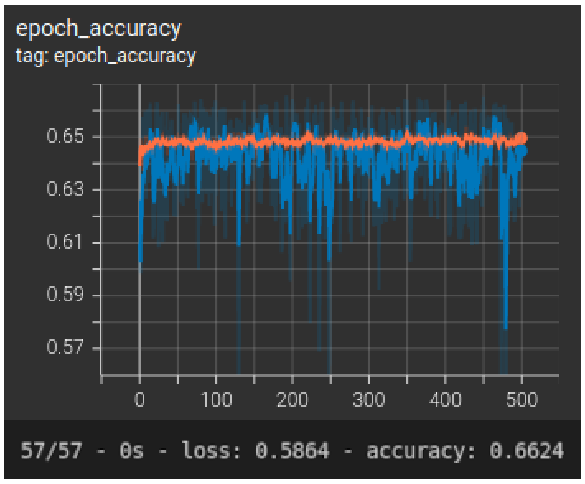
This result is somewhat surprising and shows the model is underfitting considerably. To remedy this, more features are required. This will be discussed more in subsection 4.2.

3.6. Predicting the 2021 CS:GO Major

Perhaps the most interesting task with this dataset is to attempt to predict the winner of a game based on past data. To do this, the games were ordered by date and, starting with the oldest game, a row was added to the data containing features calculated off of all available past data. In this way, future data was not used to predict game outcomes. The games played at the Major were used as the test set.

The following set of features was tracked for each team:

- Average team rating: indicator for how good a team is on average
- Proportion of matches won: another indicator for how good the team is
- Average round difference: a large positive round difference indicates the team often wins convincingly



```
57/57 - 0s - loss: 0.5864 - accuracy: 0.6624
```

Figure 11: A graph of the training and validation accuracy over 500 epochs and the test accuracy for the best model for predicting the CT or T side as the round winner.

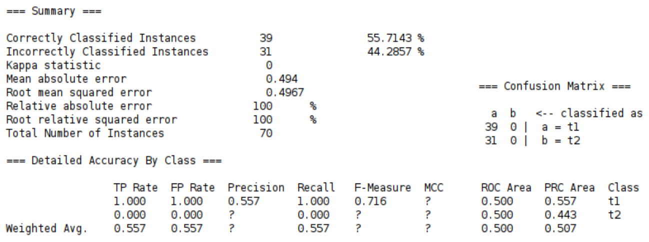
- Average opponent rating: a low average rating indicates the team tends to play easier opponents (which could skew the rest of the statistics)
- Average first kill success: the average proportion of first kills in a game that go the team's way
- Average first kill difference: a high value shows the team is good at putting themselves in advantaged situations early in rounds

These statistics were also tracked for each map for each team, as some teams are better at some maps than others. Each value was initialised to zero, and updated with the statistics for a game once the row for that game had been added to the data. Each row consists of the map name, who picked the map (team1, team2, decider), who started as the CTs, the set of features for each team and the set of features for each team for the map in question.

The data was investigated using Weka. There is a very slight class imbalance in both the train and test sets, with team1 winning 360 games and losing 291 in the training set. The zeroR classifier gets an accuracy of 55.71% on the test set as a result. Various classifiers, parameters and subsets of features were tried, with the highest accuracy shared between several combinations being 64.29%. The zeroR result and best combinations are shown in Figure 12.

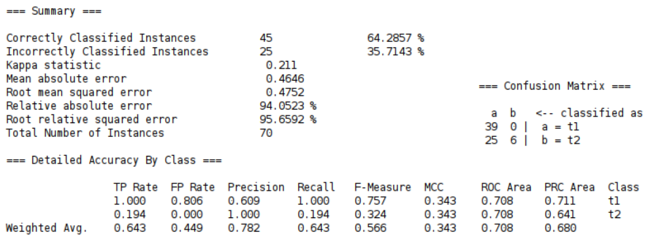
AdaBoost and SVM both achieved this accuracy with the full set of features. Interestingly, logistic regression was only able to achieve this once the map-specific features were removed. The logistic regression and SVM models had a roughly even spread between false positives and false negatives, but AdaBoost had no false negatives and a lot of false positives. Despite this, AdaBoost achieves the highest AUC-ROC value and so is probably the best model out of the three. Due to the very marginal performance increases from using a neural network in the previous task, this was not attempted here.

To understand why logistic regression was better with the map-specific features removed, the number of games played on each map for each team was plotted, as shown in Figure 13. Each graph has the same y-axis scale for easier comparison. The graph has some interesting characteristics. Several teams have maps with no games in the dataset; these maps are the team's permaban. When choosing the maps played in a match, each team first gets to ban one of the maps and some always ban the same map (their permaban). Also note that Train was replaced with Ancient in the map pool, so several teams that permabanned Train switched to permabanning Ancient and both Train and Ancient both tend to have fewer games. It is also clear that several teams have very little data to train on, and even less when this data is map-specific, so it is not surprising that map-specific



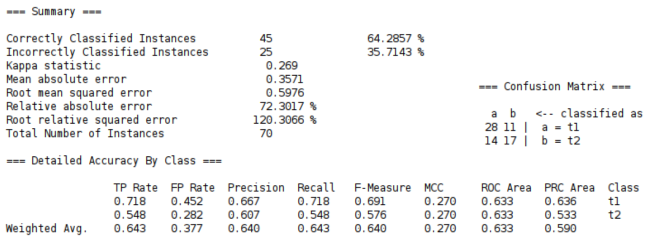
```
=== Summary ===

Correctly Classified Instances          39               55.7143 %
Incorrectly Classified Instances        31               44.2857 %
Kappa statistic                          0
Mean absolute error                      0.494
Root mean squared error                  0.4967                       === Confusion Matrix ===
Relative absolute error                100      %
Root relative squared error            100      %                       a  b   <-- classified as
Total Number of Instances               70                             39  0 |  a = t1
                                                                       31  0 |  b = t2
=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    1.000    0.557      1.000    0.716      ?        0.500     0.557     t1
                 0.000    0.000    ?          0.000    ?          ?        0.500     0.443     t2
Weighted Avg.    0.557    0.557    ?          0.557    ?          ?        0.500     0.507
```

(a) zeroR



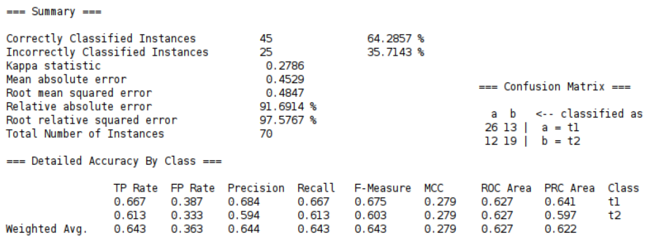
```
=== Summary ===

Correctly Classified Instances          45               64.2857 %
Incorrectly Classified Instances        25               35.7143 %
Kappa statistic                          0.211
Mean absolute error                      0.4646
Root mean squared error                  0.4752                       === Confusion Matrix ===
Relative absolute error                 94.0523 %
Root relative squared error             95.6592 %                       a  b   <-- classified as
Total Number of Instances               70                             39  0 |  a = t1
                                                                       25  6 |  b = t2
=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.806    0.609      1.000    0.757      0.343    0.708     0.711     t1
                 0.194    0.000    1.000      0.194    0.324      0.343    0.708     0.641     t2
Weighted Avg.    0.643    0.449    0.782      0.643    0.566      0.343    0.708     0.680
```

(b) AdaBoost



```
=== Summary ===

Correctly Classified Instances          45               64.2857 %
Incorrectly Classified Instances        25               35.7143 %
Kappa statistic                          0.269
Mean absolute error                      0.3571
Root mean squared error                  0.5976                       === Confusion Matrix ===
Relative absolute error                 72.3017 %
Root relative squared error            120.3066 %                       a  b   <-- classified as
Total Number of Instances               70                             28 11 |  a = t1
                                                                       14 17 |  b = t2
=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.718    0.452    0.667      0.718    0.691      0.270    0.633     0.636     t1
                 0.548    0.282    0.607      0.548    0.576      0.270    0.633     0.533     t2
Weighted Avg.    0.643    0.377    0.640      0.643    0.640      0.270    0.633     0.590
```

(c) SVM



```
=== Summary ===

Correctly Classified Instances          45               64.2857 %
Incorrectly Classified Instances        25               35.7143 %
Kappa statistic                          0.2786
Mean absolute error                      0.4529
Root mean squared error                  0.4847                       === Confusion Matrix ===
Relative absolute error                 91.6914 %
Root relative squared error             97.5767 %                       a  b   <-- classified as
Total Number of Instances               70                             26 13 |  a = t1
                                                                       12 19 |  b = t2
=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.667    0.387    0.684      0.667    0.675      0.279    0.627     0.641     t1
                 0.613    0.333    0.594      0.613    0.603      0.279    0.627     0.597     t2
Weighted Avg.    0.643    0.363    0.644      0.643    0.643      0.279    0.627     0.622
```

(d) Logistic regression

Figure 12: ZeroR and best results from using various Weka classifiers to predict the winner of games.

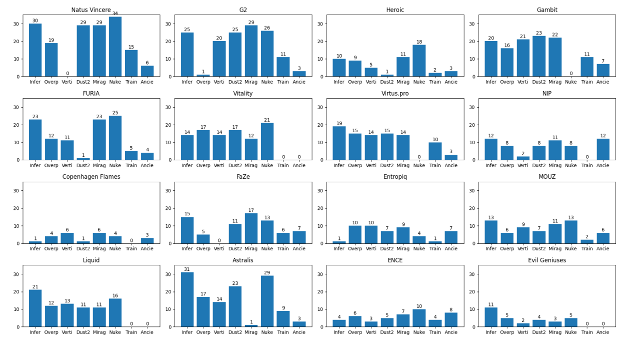


Figure 13: A graph showing the number of games played on each map for each team.

features are not very helpful.

Another point to note is that there is a lot of variation not accounted for in the data. Most of the training data comes from tournaments played online, but the Major was played on LAN in a stadium with a crowd. There are some technical differences between LAN and online, but the crowd is a big psychological factor and some teams clearly struggled. This is perhaps the reason that Gambit did worse than expected after performing very well in online events throughout the year. Copenhagen Flames, a tier-2 team, also did surprisingly well during the Major, perhaps due to the limited data available for analysis by their opponents. It is also worth noting that many teams save strategies or bring out surprising map choices in the Major to throw off their opponents, which past data cannot reflect.

Despite this, the classifiers found show there is some merit to using historical data to predict who will win a game. There are several improvements that could be investigated for this task that will be discussed in 4.2.

4. Conclusions

This project gathered a dataset on CS:GO teams, analysed it to understand its composition, and primarily investigated how well the summary statistics available translate into the ability to predict round and game outcomes. Although it showed there is some merit to doing this, there is clearly room for improvement.

4.1. Limitations

The main limitations to this project were the size of the dataset and the granularity of the data. The size was impacted mainly by the fact that only games played between the final 16 teams were included. This made it much easier to manage the data, but limited the available data considerably, especially since there were several teams in different regions or tiers that do not regularly play one another. The granularity was affected by what is available on HLTV. Economy data is given at a round-by-round level, but the rest of the data are summary statistics for whole games at best.

4.2. Recommendations for Future Analysis

The recommendations primarily address these limitations, and introduce some new tasks that might be worth analysing.

Round predictions were mainly hindered by a lack of round-by-round data. Information on how players are performing as the game progresses could give insight into the following rounds, as players might be having an on- or off-game. A game in which a team's entry player is consistently winning the first kill is one in which the team would be expected to win the majority of rounds, for example. The task was potentially also held back by a lack of data, as the team and map features were not useful, but it is a known fact that some teams perform better on certain maps (Natus Vincere had a 20 game win-streak on Nuke coming into the finals of the Major, for example).

Higher granularity data is attainable by parsing demo files, which store a game's state at every tick (time-step). There are tools for this, such as demoparser[4], but scraping both summary statistics and demos from HLTV as well as parsing the demo files was far out of the scope of this project. If this data were available, it might be interesting to investigate the prediction of round outcomes live as the round progresses.

Game predictions were mainly held back by a lack of data. As shown in Figure 13, some teams had very little data, and even less when the data was split across maps. Scraping data on every game involving at least one of the teams (instead of both) in the final 16 for the Major would massively increase the size of the dataset and greatly improve the features picked out for this task. Although far out of the scope of the project, using a neural network to learn features, instead of designing them manually, might also help.

Finally, an additional analytical task that would be interesting to investigate is the prediction of map vetoes. Being able to do this successfully would greatly assist teams by allowing them to focus on strategising for fewer maps for each match they play. Vetoes should be predictable with high accuracy, as teams tend to ban maps they are bad at and pick maps that either they are good at or their opponents are bad at.

Project code: ***https://github.com/ralfleggett/CS430_cswk***

References

[1] HLTV. "Hltv." (2021), [Online]. Available: http://hltv.org (visited on 12/29/2021).

[2] K. Reitz. "Requests: Http for humans™." (2021), [Online]. Available: https://docs.python-requests.org/en/latest/ (visited on 12/29/2021).

[3] L. Richardson. "Beautiful soup documentation." (2020), [Online]. Available: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (visited on 12/29/2021).

[4] k. Ryan Moe. “Demoparser.” (2018), [Online]. Available: https : / / github . com / ibm - dev - incubator / demoparser (visited on 12/29/2021).