Group 148 Progress Report:
CS2 Match Predictor

Jayesh Anil, Jack Fountain, Asadullah Muhammad
{anilj,fountj1,muhama23}@mcmaster.ca

1 Introduction

The CS2 professional scene involves a plethora of teams competing in a variety of leagues and tournaments. As a result, there is a lot of interest in predicting the outcome of matches, and in predicting the performance of teams in different leagues and tournaments. The goal of this project is to build a model that can predict the outcome of a particular match, given the teams, the map, and when the match was played. The model can then be generalized to predict the winner of a tournament, since a tournament is simply a collection of matches.

2 Related Work

The 5 related work we could find are as follows:

- Predicting Counter-Strike Matches (Lund University)[2]:
  - This thesis uses HLTV data to engineer team-vs-team and team-on-map win rates as predictive features, mirroring our exact preprocessing approach.
  - They use multiple machine learning approaches, including Logistic Regression and Random Forests.
  - They achieved an accuracy of roughly 60% on test data.
- Predicting the outcome of CS:GO games using machine learning (Chalmers University)[1]
  - This thesis leverages historical FACEIT match results, extracting win rates per map and opponent, nearly identical to our dataset preprocessing again.
  - They utilized nerual network and regression models.
  - Their network can predict the games with an accuracy of 65.11% compared to the benchmark prediction that relies only on win rate that gets a prediction of 58,97%.
- Predicting Counter-Strike Game Outcomes with Machine Learning (Czech Technical University)[5]
  - They built large match dataset from HLTV, extracted win rates for team-vs-team and team-on-map.
  - They compared six ML algorithms, emphasizing Elo rating and Random Forests.
  - Accuracy of 64.0% has been achieved with their best performing model based on the Elo rating of players, followed by 63.0% accuracy with the random forest model and 59.8% accuracy with the convolutional neural network, all beating the baseline model with 55.0% accuracy.
- Software for predicting the outcomes of CS2 matches (GitHub, 2024)[4]
  - They automated collection and cleaning of match, map, and team statistics from HLTV.
  - They compared several ML models..
  - The github repo does not explicitly state the testing accuracy for the model.
- Bandit Modeling of Map Selection in Counter-Strike (Petri et al., Mihamerstan)[3]
  - They scraped HLTV data, calculated contextual win rates (team and map), and applied Laplace smoothing for rare cases.
  - Multi-armed bandit algorithms for optimal map selection and prediction, using win rates as algorithm context.
  - The research states that model can improve teams' predicted map win probability by up to 11 percentage points and overall match win probability by 19.8% for balanced teams.

3 Dataset

This dataset comes from HLTV.org website and includes data for every team that held a top 36

global ranking in CS2 at any point during the period from November 4, 2024, to November 3, 2025. We have a total of 12658 data rows after balancing the data.

For each team, there is a CSV file with all the matches they played in that year (including not just games against other top 36 teams, but also matches against teams that never reached that rank). The structure of each csv file for each team used in our analysis is the same and has been described in detail in Table 1.

Every row represents a single match, with columns for:

Column Name	Description
`team_name`	Name of the team (e.g., 3dmax)
`team_id`	HLTV's unique team identifier.
`map_name`	CS2 map played (e.g., Mirage, Dust2, Nuke).
`map_id`	HLTV's Unique map identifier.
`match_date`	Date of the match played (YYYY-MM-DD).
`opponent_name`	Name of the opposing team.
`opponent_id`	Opponent's HLTV ID.
`score_us`	Rounds won by the team in that game.
`score_them`	Rounds won by the opponent team in that game.
`result`	Outcome for the team ('W' for win, 'L' for loss).
`event_name`	Name of the event or tournament (e.g., ESL Pro League).
`event_id`	Unique event identifier.

Table 1: Description of each column in our scraped data from HLTV.org

3.1 Data Preprocessing

To make the analysis more meaningful, we preprocess each team's data to calculate two main statistics for win prediction:

- Win percentage vs. each opponent: For every opponent a team has faced, we calculate the proportion of matches won. This measures how "favorable" a matchup has been for the team against specific rivals, regardless of the map or event.

- Win percentage on each map: For each map, we calculate the proportion of wins out of all matches played by the team on that map (regardless of opponent or event). This helps identify map strengths and weaknesses for the team.

To calculate these stats, we group the raw match data by opponent or by map and find the win ratio in each group. When we convert all this data into features and columns to use as model input, we make sure that there's absolutely no information leakage. Essentially, for each row (representing one match), we only use data from matches that occurred before this particular one. Even though we have the entire year's results, we don't use any future information to influence the statistics for any match. So, if a team played more games later in the year, those don't affect the calculation for earlier matches.

Lastly, because we don't use the team name as an input (all inputs are stats), if one team just happened to win most of the time in the real world, the model would learn to always pick that outcome and could get artificially high accuracy just by guessing the majority. That's why we balance the dataset (by removing 1770 data samples from a total of 14428 data samples) so wins and losses are 50/50 in our training data. This way, the model's accuracy actually tells us how much it's learning from the patterns in the features, not just exploiting which class is bigger. We can see the balancing results in Figures 1 and 2.

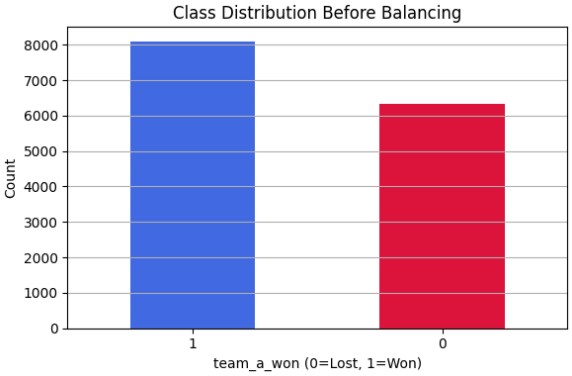

Figure 1: Class Distribution before balancing

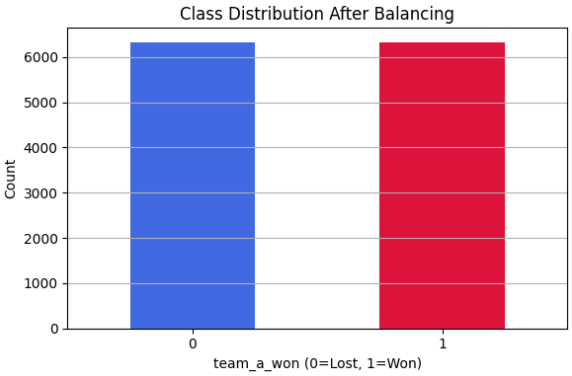

Figure 2: Class Distribution after balancing

4 Features

For each match in our dataset, we engineered a set of features that capture both global and head-to-head team performance, as well as map-specific stats relevant to the match date:

4.1 Map-Encoding

We use one-hot encoding to represent each map in CS2 (columns like `map_id_31`, `map_id_32`, ..., `map_id_48`), so only the column representing the specific map for the match will be 1 others all will be 0.

4.2 Head to Head performance for each team

We have the feature columns that represent the historical statistics for team A playing against team B in Table 2.

4.3 Map-Specific Performance for Both Teams

We have the features that include the performance data of each team on the specific map for all the matches played by this team till the date when the current match was played, explained in detail in Table 3

4.4 Global Ranking

Below are the features having the global ranking for each team when the match was played:

- `team_a_global_ranking`/ `team_b_global_ranking`: HLTV global rankings points for each team at the match date.

4.5 Label

- `team_a_won`: 1 if Team A won that particular match, 0 otherwise.

Feature	Description
`team_a_wins_vs_b`	Number of times Team A has won against Team B till the date the match was played.
`team_a_losses_vs_b`	Number of times Team A has lost to Team B up to that match date.
`team_a_total_vs_b`	Total number of previous matches between Team A and Team B up to the date when the match was played.
`team_a_winrate_vs_b`	Win rate of Team A winning when playing against Team B, calculated as $\frac{team\_a\_wins\_vs\_b}{team\_a\_total\_vs\_b}$.

Table 2: Feature Descriptions for head-to-head performance for each team

Feature	Description
`team_a_map_wins`/ `team_b_map_wins`	Number of matches won by team A/B on this specific map.
`team_a_map_losses`/ `team_b_map_losses`	Number of matches lost by team A/B on this specific map.
`team_a_map_total`/ `team_b_map_total`	Total number of matches played by team A/B on this specific map.
`team_a_map_winrate`/ `team_b_map_winrate`	Win rate for team A/B on this specific map.

Table 3: Feature Descriptions for map-specific performance for both teams

5 Implementation

Regarding our model implementation so far, we settled on a neural network for its flexibility in creating decision boundaries for binary classification tasks. The natural choice for the loss function would be binary cross-entropy loss, given our classification task. The selected optimizer was Adam, given its ability to handle noise and quick convergence speed, allowing us to iterate through a variety of hyper-parameter choices quickly. For the model learning process, mini-batch training was used to train robust models quickly and efficiently.

6 Results and Evaluation

For evaluation purposes we split our data into test and train data. (Specifically, 80% of the total data is training data, and 20% is testing data with stratification enabled to maintain balance for the class labels in both the set of data)
We are evaluating our model using the following metrics, with results:

- Accuracy: 58.49%
- Precision: 58.75%
- Recall: 57.03%
- F1-Score: 57.88%
- Confusion Matrix:

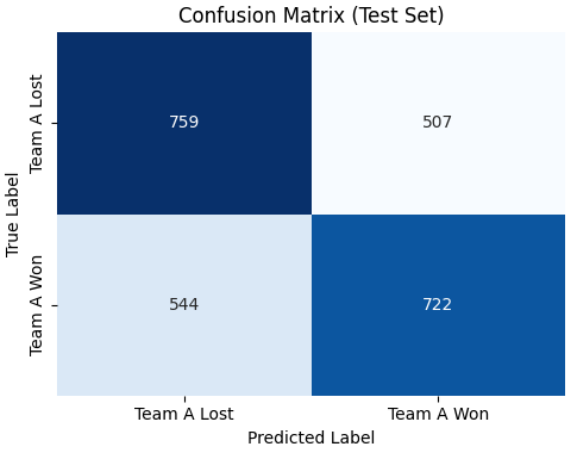

Figure 3: Confusion matrix figure

Our baseline is 50% based on the pre-processing that we did on our data.

7 Feedback and Plans

In our latest feedback, the TA pointed out that we don't need to get too deep into really granular input data for the model, like adding individual player and game stats for every team, since that could get really complicated. We finalized that we can bring in some specific player-level features if we think it would help our predictions, but for now, unless we see a huge benefit and it's not highly complicated to implement, we'll probably stick to the features we have currently.
TA also suggested that we try using other models, like decision trees and especially XGBoost. TA mentioned that the XGBoost model will improve our accuracy significantly. For our next steps, we're planning to implement XGBoost and compare its performance to our current (simpler) neural network model. If XGBoost (or even a regular decision tree) gives us better accuracy, we plan to change our model.

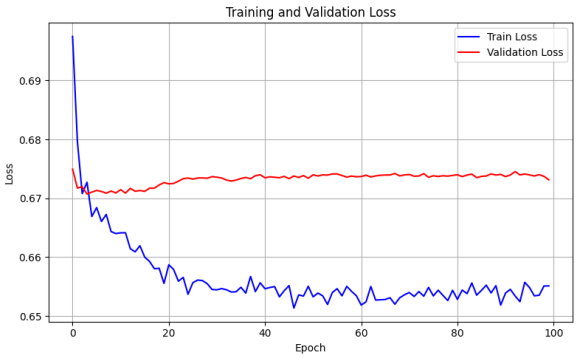

Figure 4: Training and Validation Loss Curves

Figure 5: Training and Validation Accuracy Curves

8 Team Contributions

- Jack Fountain (fountj1): Data collection, Data Pre-processing
- Asadullah Muhammad (muhama23): Data Pre-processing, Model Implementation
- Jayesh Anil (anilj): Model Implementation, Writing Report

References

[1] Arvid Björklund, William Johansson Visuri, Fredrik Lindevall, and Philip Svensson. 2018. Predicting the outcome of cs:go games using machine learning. Bachelor's Thesis.

[2] Broms, Erik and Nordansjö, William. 2024. Predicting counter-strike matches using machine learning models. Student Paper.

[3] Guido Petri, Michael H. Stanley, Alec B. Hon, Alexander Dong, Peter Xenopoulos, and Cláudio Silva. 2021. Bandit modeling of map selection in counter-strike: Global offensive.

[4] point516. 2023. cs_bot: Counter-strike prediction bot. *GitHub repository*. Accessed: 2025-11-11.

[5] Ondrej Svec. 2022. Predicting counter-strike game outcomes with machine learning.