Predicting HLTV's Top 20 CS2 Players

Mark Zhdan

1 Introduction

The goal of this project is to use machine learning to predict HLTV.org's annual Top 20 CS2 player rankings based on real-world performance data. HLTV, a leading competitive Counter-Strike statistics site, publishes detailed metrics such as rating, kills per round, damage per round, and statistics versus top-tier teams. These rankings are heavily taken into consideration when ranking players, but there is still some nuance as players can win MVPs, EVPs, and trophies (1st, 2nd, 3rd place).

My motivation was to explore whether a machine learning model can replicate or approximate HLTV's rankings using publicly available statistics. I collected and engineered player performance data from HLTV and Liquipedia between 2018–2024, totaling around 100-200 samples. This prediction problem is challenging due to the small sample size, the ordinal nature of ranking, and varying data completeness.

2 Data Collection & Preprocessing

Figure 1: Overview of the HLTV stats page

The first phase of this project was collecting data. HLTV.org does not offer an open API, and its data is protected by Cloudflare's anti-bot mechanisms. In order to bypass this, I used the open-source Python library `CloudflareBypassForScraping` (`https://github.com/`

`sarperavci/CloudflareBypassForScraping`). This allowed me to route my scraping through a headless Puppeteer service to simulate human browsing.

Using this setup, I scraped all relevant player performance metrics from HLTV, applying filters such as:

- Time ranges (Jan–Dec for each year from 2018–2024)
- Opponent strength filters (vs. Top 5/10/20)
- Match types (LAN, Big Events, Majors)

The scraping script looped through player IDs and extracted individual profile and stats pages:

```
for player_id in player_ids:
    url = f"https://www.hltv.org/stats/players/{player_id}?startDate=2018-01-01&endDate=2018-
    html = cloudflare_bypass_request(url)
    soup = BeautifulSoup(html, 'html.parser')
    # extract ADR, KPR, Impact, etc.
```

In addition to HLTV data, I added MVP and EVP counts by scraping Liquipedia. Each tournament page was parsed to match player names to awards. This helped strengthen ranking targets with qualitative performance indicators that HLTV uses in their own lists since rankings are not only determined by raw stats but also team tournament performance.

After scraping, I cleaned the data by:

- Filling missing numerical values using column means
- Replacing -1.0 placeholders with `NaN`
- Converting ranks to ordinal labels (e.g., rank 1 $\rightarrow$ class 0)

This preprocessing ensured all model-ready data was numerically stable and representative of each player's true performance.

3 Methods/Case Study

I experimented with three major models: decision trees, support vector regression (SVR), and neural networks. Each model had distinct strengths and limitations.

Method 1: Regression Trees

My initial model was a decision tree regressor implemented from scratch using variance reduction to minimize MSE. The tree supported custom thresholds, depth control, and leaf node smoothing. I evaluated different depths and gain thresholds, but found the model suffered from frequent tie predictions due to small splits and limited feature interactions.

Method 2: Support Vector Regression (SVR)

Next, I transitioned to SVR, which is more suitable for small datasets with high dimensionality. I performed a grid search on parameters: `C`, `epsilon`, `kernel`, and `gamma`, using both linear and RBF kernels. Feature selection was done using `SelectKBest` to retain the 50 most informative statistics. The SVR showed promise, but plateaued early in accuracy and struggled to capture deep nonlinear patterns.

Method 3: Deep Neural Network

The best-performing model was a multi-layer perceptron implemented in PyTorch. The architecture included:

- Input layer (50 features)
- 3 hidden layers (256–128–64 neurons)
- BatchNorm and Dropout (0.3–0.4) to prevent overfitting
- Final regression layer (output = player score)

I trained the model using Adam optimizer with L2 regularization and visualized the loss curve using 5-fold cross-validation. This helped generalize across years and account for noisy or incomplete data from HLTV.

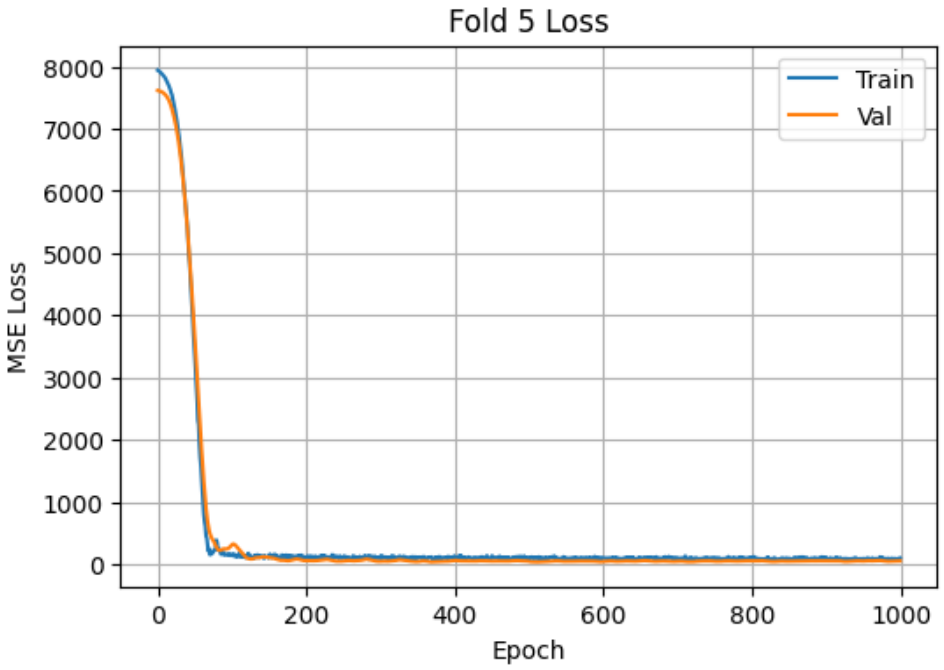

Figure 2: Training and validation loss from the neural network model.

Table 1: Model performance on test data (ranking accuracy)

Model	Test Score /100
Tree Regressor	44
SVR (RBF + GridSearch)	52
Neural Network (Dropout + L2)	**58**

4 Results and Discussion

The neural network demonstrated the highest performance and generalization. As seen in Figure 2, validation loss converged smoothly across folds, indicating strong stability without overfitting. Its flexibility allowed it to model the complex relationships between raw stats and player rankings.

To evaluate results, I compared predicted player rankings to HLTV's actual 2024 Top 20 list. I assigned up to 5 points per player: exact match (5), 1-off (4), ..., within 5 ranks (1). This approach accounts for ranking noise and subjective placement.

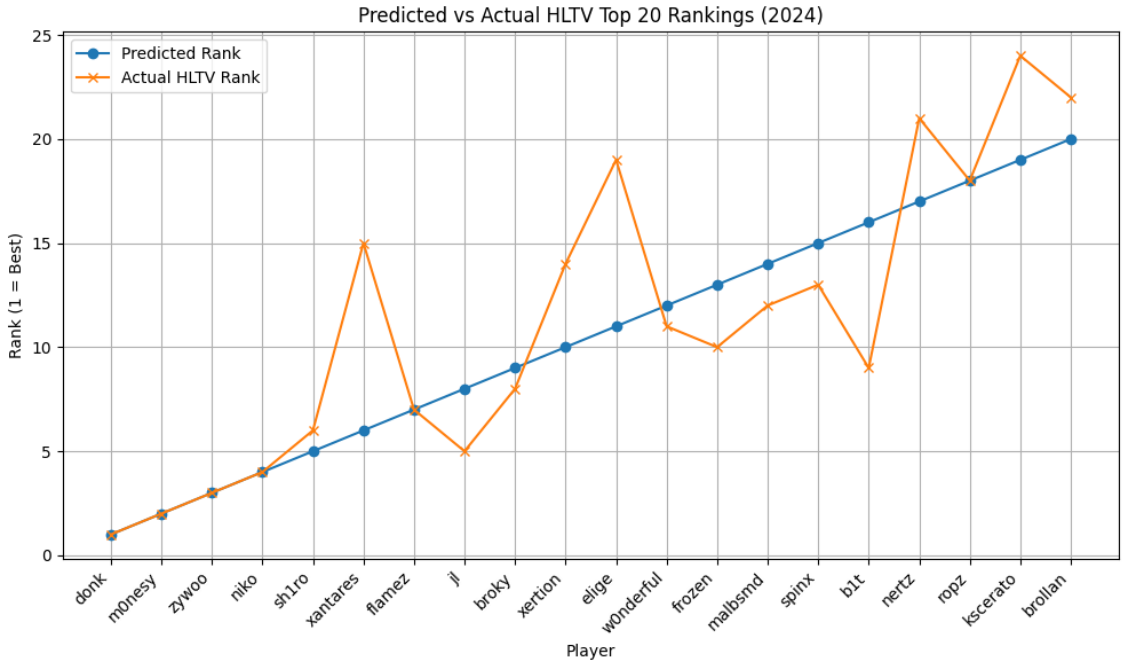

Figure 3: Neural Net predictions vs. actual HLTV Top 20 (2024).

The tree regressor struggled to differentiate elite players due to its reliance on discrete thresholds. The SVR captured some structure but flattened at higher ranks. The neural net clearly performed best, making nuanced distinctions among similarly ranked players, especially for

highly contested spots in the top 5 as it got the top 4 players correct.

Key insights:

- Cross-validation and dropout were essential for generalization
- SelectKBest improved performance by reducing noise
- Neural networks outperformed both linear and greedy models due to nonlinear depth

Feature Importance

To improve generalization and reduce noise, I applied univariate feature selection using `SelectKBest`. The top features selected consistently across folds included:

- Rating 2.0 / Rating 1.0
- KPR, ADR, DPR
- Big Events rating (b_)
- Stats vs Top 5 and Top 10 teams (e.g., `vs_top5`, `b_vs_top10`)

These features reflect HLTV's own emphasis on raw performance metrics and impact against high-tier opponents. The presence of multiple variations (e.g., big events, majors, online vs LAN) allowed the models to differentiate players beyond just averages. Models like the neural network were able to make use of these combinations more effectively than the tree or linear baselines.

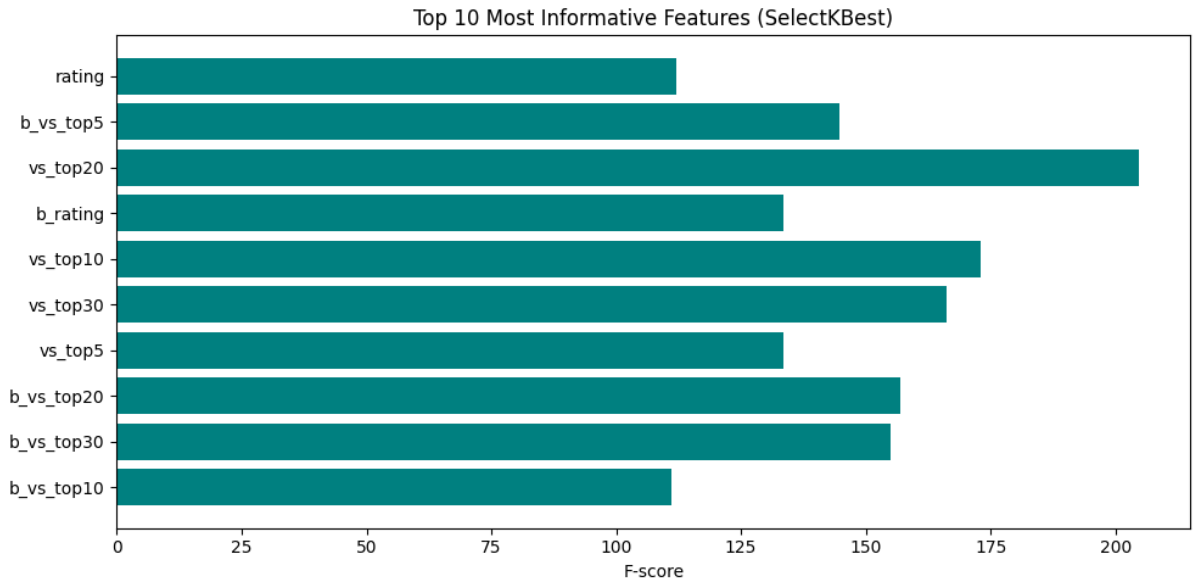

Figure 4: Most informative features selected across models.

5 Conclusion

This project applied machine learning to a real-world prediction task in Counter-Strike. I designed three models and evaluated their effectiveness on ranking players based on performance data. Through this process, I learned the value of regularization, data cleaning, and model tuning.

The neural network outperformed both SVR and decision trees, demonstrating that deep models are well suited to capturing the subtle, nonlinear nature of performance-based rankings. The results closely mirrored HLTV's 2024 Top 20 list and proved the feasibility of automating ranking tasks with a robust machine learning pipeline.

I would further improve this model by taking monthly statistic entries to take into consideration consistent form (no recency bias), add team tournament wins, and other player metrics. This wasn't feasible due to scraping time and complexity.

6 Links

Video: `https://youtu.be/6DBWzcojKW4`
GitHub Repo: `https://github.com/markzhdan/hltv-top-20-predictions`
Cleaned Datasets: `https://github.com/markzhdan/hltv-top-20-predictions/tree/main/data/clean/complex/final`
Notebooks: `https://github.com/markzhdan/hltv-top-20-predictions/tree/main/src/notebooks`