CSGO Pro Player Performance Analysis

STAT1000J Project presentation

Hao Liang

UM-SJTU JI
Shanghai Jiao Tong University

April 14, 2023

Overview

1. Crawl data from HLTV
2. Exploration
3. Further Exploration
4. Hypothesis Test
5. Build a predictive model

HLTV

HLTV is a website dedicated to keeping track of professional players and publishing news.

Column	Description
Player	The name of the player
Nationality	The nationality of the player
Teams	The team(s) the player has played for
Maps	The number of maps the player has played
Rounds	The number of rounds the player has played
K-D Diff	The difference between the player's kills and deaths
K/D	The player's kill/death ratio
Rating 2.0	The player's rating based on their performance

Table: Description of the data.

Data Set

I use `Beautiful Soup` module in `Python` to crawl data from HLTV. The first several lines of data set I obtain are as follow:

	Player	Nationality	Teams	Maps	Rounds	K-D Diff	K/D	Rating 2.0
0	ZywOo	France	Vitality	1118	29482	+6876	1.38	1.27
1	s1mple	Ukraine	Natus Vincere	1666	44101	+9478	1.34	1.24
2	sh1ro	Russia	Cloud9	979	26066	+6329	1.46	1.23
3	deko	Russia	1WIN	545	14803	+3090	1.36	1.20
4	kaze	Malaysia	Rare Atom	948	24748	+4485	1.31	1.18
5	smooya	United Kingdom	BIG	924	24496	+4043	1.27	1.18

Figure: DataFrame overall_df

Which countries do the professional players come from?

Code

```
# The top 10 countries with the most CS:GO professional players.
country = overall_df.groupby('Nationality')['Player']\
                    .count()\
                    .rename("Count")\
                    .sort_values(ascending=False)
country.head(10)
```

Which countries do the professional players come from?

```
# The top 10 countries with the most CS:GO professional players.
Nationality
Denmark          71
United States    68
Russia           60
Brazil           60
Poland           49
Sweden           46
Australia        32
France           27
Ukraine          24
Bulgaria         24
Name: Count, dtype: int64
```

How pro players from different countries perform?

Code

```
# The top 10 countries with the highest average rating
overall_df\
        .groupby('Nationality')['Rating 2.0'].mean()\
        .sort_values(ascending=False)\
        .head(10)
```

How pro players from different countries perform?

```
# The top 10 countries with the highest average rating
Nationality
Malaysia                  1.180000
Hong Kong                 1.120000
Indonesia                 1.115000
Bosnia and Herzegovina    1.100000
Singapore                 1.083333
New Zealand               1.072857
Uruguay                   1.070000
Israel                    1.066667
Guatemala                 1.060000
Korea                     1.056667
Name: Rating 2.0, dtype: float64
```

How pro players from different countries perform?

Thinking

It seems that professionals from Asia perform well as their average ratings are far ahead of other country. However, teams from Asia performs poor in Pro Leagues. This is really a tough problem.

Can we explore further?

How about the standard deviation?

How pro players from different countries perform?

Code

```
# The top 10 countries with the highest standard deviation on rating
overall_df\
    .groupby('Nationality')['Rating 2.0']\
    .apply(np.std).sort_values(ascending=False)\
    .head(10)
```

How pro players from different countries perform?

```
# The top 10 countries with the highest standard deviation on rating
Nationality
Ukraine           0.092142
United Kingdom    0.083600
Turkey            0.079282
France            0.075628
Russia            0.074768
Germany           0.071455
Denmark           0.067550
New Zealand       0.066486
Bulgaria          0.064806
Mongolia          0.064374
Name: Rating 2.0, dtype: float64
```

How pro players from different countries perform?

Conclusion

It is worth noting that many countries with high standards are located in Europe and the CIS region, which are also home to top teams.
From my perspective, there are several reasons:

- Many top players with super high ratings come from certain countries, which raises the overall std.
- The professional leagues in Europe and CIS are highly competitive, and players from these regions may have lower ratings compared to their counterparts from Asia and America, despite being at the same skill level.

Rating difference between CT side and T side

Basic game rules

- In CS:GO, the T(Terrorist) side is the attacking team and the CT(Counter-Terrorist) side is the defending team.
- In a match, there are 15 rounds in each half, and after the first half, the teams switch sides.
- It is often more difficult for the T side to win rounds than the CT side because the CT side has the advantage of holding boom sites (A and B), which can create a difference in the statistics between the two sides.

Rating difference between CT side and T side

I first load the data from both sides and the overall data.

```
# Load data
CT_df = get_dataset('CS_GO Player CT statistics database - HLTV.org.html')
T_df = get_dataset('CS_GO Player T statistics database - HLTV.org.html')
overall_df = get_dataset('CS_GO Player statistics database - HLTV.org.html')
```

Rating difference between CT side and T side

After cleaning the data, I merge them together

```
df = CT_df.merge(right=T_df, left_on='Player', right_on='Player')
df = df.merge(right=overall_df, left_on='Player', right_on='Player')
df.sort_values(by='overall_Rating 2.0', ascending=False)
```

Rating difference between CT side and T side

`df` is shown as follow:

	Player	Nationality	Teams	Maps	CT_rounds	CT_K-D Diff	CT_K/D	CT_Rating 2.0	T_rounds	T_K-D Diff	T_K/D	T_Rating 2.0	overall_rounds	overall_K-D Diff	overall_K/D	overall_Rating 2.0
0	ZywOo	France	Vitality	1118	14817	+5078	1.61	1.39	14665	+1798	1.19	1.16	29482	+6876	1.38	1.27
2	s1mple	Ukraine	Natus Vincere	1666	22105	+6523	1.49	1.33	21996	+2955	1.20	1.17	44101	+9478	1.34	1.24
1	sh1ro	Russia	Cloud9	979	12877	+4352	1.69	1.33	13189	+1977	1.26	1.13	26066	+6329	1.46	1.23
3	deko	Russia	1WIN	545	7489	+2407	1.60	1.32	7314	+683	1.15	1.10	14803	+3090	1.36	1.20
7	saffee	Brazil	FURIA	471	6257	+1694	1.48	1.29	6235	+430	1.11	1.07	12492	+2124	1.28	1.18
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
738	djL	Sweden	Chaos	688	9396	-158	0.97	0.93	8784	-1577	0.75	0.79	18180	-1735	0.86	0.86
736	gob b	Germany	BIG	978	13043	-216	0.97	0.94	12756	-2612	0.72	0.76	25799	-2828	0.84	0.85
740	B1ad3	Ukraine	FlipSid3	899	11375	-719	0.91	0.92	11857	-2689	0.70	0.74	23232	-3408	0.80	0.83
742	PASHANOJ	Russia	Unique	620	8039	-707	0.87	0.88	8188	-1664	0.73	0.79	16227	-2371	0.80	0.83
741	HUNDEN	Denmark	Tricked	1578	20843	-1357	0.90	0.90	20310	-4881	0.67	0.73	41153	-6238	0.78	0.81

Figure: Caption

Visualization

In order to have a clearer understanding of the differences between the two sides, I use histograms to visualize my data.

The first three plots are `Overall_Rating 2.0`, `CT_Rating 2.0` and `T_Rating 2.0`

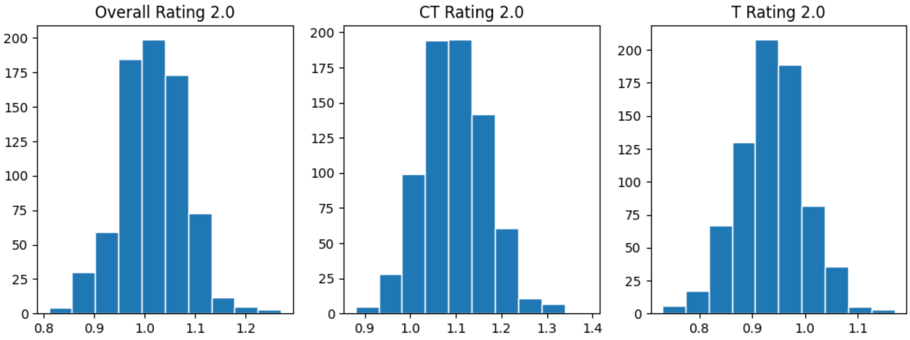
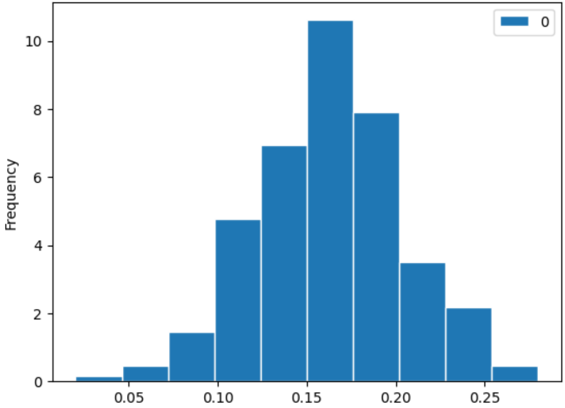


Figure: Caption

Visualization

Let's see the distribution of `CT_Rating 2.0 - T_Rating 2.0`

```
pd.DataFrame(df['CT_Rating 2.0'] - df['T_Rating 2.0']).plot(kind='
    hist', ec='w', density=True)
```
1



Hypothesis Test

Observation

Based on the figure presented in the previous slide, it appears that the average value of `CT_Rating 2.0` is 0.16 higher than that of `T_Rating 2.0`.

Hypothesis Test

Observation

Based on the figure presented in the previous slide, it appears that the average value of `CT_Rating 2.0` is 0.16 higher than that of `T_Rating 2.0`.

Null Hypothesis

The average `T_Rating 2.0` is no more than 0.16 below the average `CT_Rating 2.0`.

Alternative Hypothesis

The average `T_Rating 2.0` is significantly lower that the average `T_Rating 2.0` by more than 0.16.

Significance level: $\alpha = 0.05$

Simulation

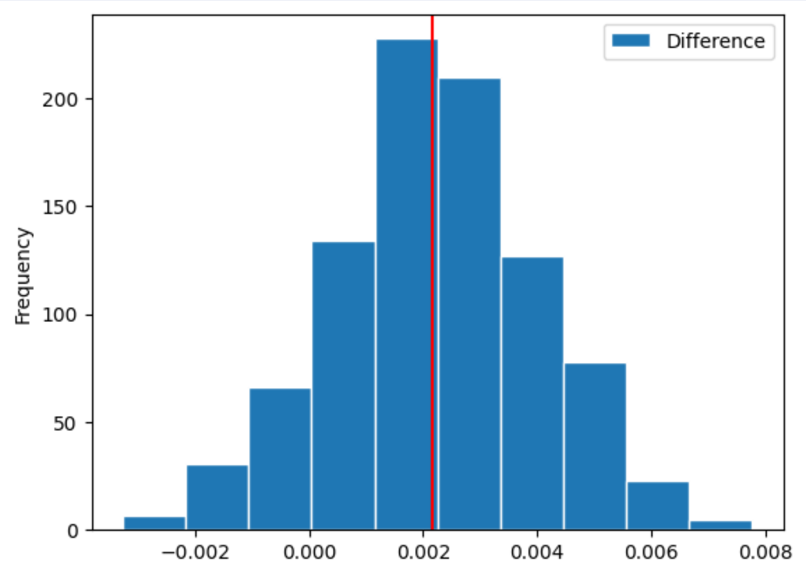
- The simulation runs 100 times.
- It calculates the mean difference between the average `T_Rating 2.0` and the average `CT_Rating 2.0` and then subtracting 0.16 from the mean difference.
- Each simulation samples 300 professionals from the merged DataFrame `df`.
- The simulation is run 1000 times for more accurate results.

Simulation

```python
n_reputation = 1000
rating_differences = []
statistic = (df['CT_Rating 2.0'] - df['T_Rating 2.0']).mean() - 0.16
# Iterate through n_reputation samples
for i in range(n_reputation):
    # Sample 300 rows from df
    sample = sample_300(df)
    difference = (sample['CT_Rating 2.0'] - sample['T_Rating 2.0']).mean() - 0.16
    rating_differences.append(difference)

pd.DataFrame().assign(Difference = rating_differences).plot(kind='hist', density=True, ec='w');
plt.axvline(statistic, color='red');
```

Simulation


Difference
Frequency
0
50
100
150
200
−0.002
0.000
0.002
0.004
0.006
0.008

Frame Title

Now, calculate the p-value as the proportion of simulation results that are at least as extreme as the observed test statistic (i.e., have a greater value).

- If p-value is greater than the significance level, I accept the null hypothesis.
- If the p-value is smaller than the significance level, I can reject the null hypothesis and conclude that the average `T_Rating 2.0` is significantly lower that the average `CT_Rating 2.0` by more than 0.16.

Code:

```
# Calculate the p-value based on the simulation results
p_value = sum(np.array(rating_differences) >= statistic) / n_reputation
# Determine if the null hypothesis is accepted or rejected based on the calculated p-value
if p_value > 0.05:
    print("p-value is", p_value, "and is greater than the significance level.")
    print("Therefore, we accept the null hypothesis.")
else:
    print("p-value is", p_value, "and is less than or equal to the significance level.")
    print("Therefore, we reject the null hypothesis and conclude that the average T_Rating 2.0 is significantly lower than the average CT_Rating 2.0 by more than 0.16.")
```

Conclusion of the test

The output of the code on previous slide:

Output

p-value is 0.489 and is greater than the significance level.
Therefore, we accept the null hypothesis.

Conclusion

So that we can conclude that teams should pay more attention to their offensive tactics in order to get more points on T side, which will definitely increase their winning rate.

Build a predictive model

Predict `Rating`

`Rating` is a comprehensive indicator that showcases a player's performance in the game. The higher the `rating`, the better the player's performance. There are many factors that affect the `rating`, such as Average Damage per Round (ADR), Kill/Death Ratio, Assists, etc. How are the weights of these factors allocated? Can we predict `rating` based on data? To answer these two questions, I build a predictive model.

Data set

I import data from `csgo_player_stats.csv`, which contains comprehensive and diverse data. Next slide shows an overview of each column in this CSV file:

Overview of `csgo_player_stats.csv`

Column Name	Description
Name	The name of the player.
Total Kills	The total number of kills made by the player in the game.
Headshot Percentage	The percentage of kills made by the player that were headshots.
Total Deaths	The total number of deaths the player has had in the game.
Kill/Death Ratio	The ratio of kills to deaths for the player.
Damage Per Round	The average amount of damage dealt by the player per round.
Grenade Damage Per Round	The average amount of damage dealt by the player with grenades per round.
Maps Played	The total number of maps the player has played.
Rounds Played	The total number of rounds the player has played.
Kills Per Round	The average number of kills made by the player per round.
Assists Per Round	The average number of assists made by the player per round.
Deaths Per Round	The average number of deaths the player has had per round.
Saved By Teamates Per Round	The average number of times the player has been saved by a teammate per round.
Saved Teamates Per Round	The average number of times the player has saved a teammate per round.
KAST	The percentage of rounds in which the player either had a kill, assist, survived or was traded.
Impact	The average impact made by the player per round, which is a measure of how much the player's actions contribute to the team's success.
Rating 2.0	The player's rating, which is a measure of their overall performance in the game. This rating takes into account a variety of factors, including K/D ratio, Damage Per Round and KAST.

Split the data

I load the data from `csgo_player_stats.csv` and store the training data in `tr` and testing data in `te` by using the `train_test_split` function from `sklearn.mode_selection`.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# Read in csv file
pred_data = pd.read_csv('csgo_player_stats.csv')
pred_data['KAST'] = pred_data['KAST'] / 100

 # Split data into training and testing sets
tr, te = train_test_split(pred_data, test_size=0.1, random_state
    =64)
```

Build a Linear Regression Model

Which factors to choose?

According to my experience, `Kill/Death Ratio`, `Damage Per Round` (ADR) and `KAST` are the three most significant factors that can influence 'Rating 2.0'.

How to do a regression

I use `LinearRegression` from `sklearn.linear_model` module to perform the linear regression.

get_pre_columns function

`get_pre_columns` is a function that gets the pre-existing columns from the DataFrame and creates a Pipeline object with a `LinearRegression` model to predict the `Rating 2.0` column. The function then fits the model, creates the prediction and returns the model and the prediction.

Build the model

```python
def get_pre_columns(df):
    model = Pipeline([
        ("SelectColumns", ColumnTransformer([("keep", "passthrough", ['Kill/Death Ratio', 'Damage Per Round', 'KAST'])])),
        ("LinearModel", LinearRegression())
    ])

    model.fit(df, df['Rating 2.0'])

    prediction = model.predict(df)

    return model, prediction
```

Regression Result

I use Root Mean Square Error (RMSE) to evaluate the regression model.

Training error:

```
model, Y_hat_tr = get_pre_columns(tr)
Y_tr = tr['Rating 2.0']
print("Training Error (RMSE):", rmse(Y_tr, Y_hat_tr))

Output: Training Error (RMSE): 0.0282546454228108
```

Testing error:

```
Y_hat_te = model.predict(te)
Y_te = te['Rating 2.0']
print("Testing Error (RMSE):", rmse(Y_te, Y_hat_te))

Output: Testing Error (RMSE): 0.03360466120451094
```

Regression Result

Let's see the coefficients of the factors and the intercept of the model:

```
model['LinearModel'].coef_, model['LinearModel'].intercept_

Output: (array([0.57537783, 0.00421085, 0.19137497]),
    -0.012197289752643892)
```

Regression Result

Conclusion

As shown in the blow cells, the RMSE of both train data and the test data are very low, which means we actually perform a good regression model on these factors.
The regression model is as follow:

$$\text{Rating 2.0} = 0.575 \times \text{K/D Ratio} + 0.004 \times \text{ADR} + 0.191 \times \text{KAST} - 0.0122$$

The End