Sacred Heart University
Jack Welch College of Business & Technology

Professional Players' Rating Prediction Using Linear Regression
24FABUAN670BB – Data Mining – Project Report

by

Sai Keerthana Mallipeddi
Thamson Antony Arockiasamy

Instructed by

Dr. Sonal Vats

2 December 2024 AD

Contents

1. Project Overview 1
2. The What and Why 2
3. Dataset 3
4. Statistical Data Analysis 4
    - 4.1. Housekeeping 4
    - 4.2. Data Analysis 5
    - 4.3. Data Cleaning 6
    - 4.4. Exploratory Data Analysis 6
    - 4.5. Regression Analysis 10
5. Results and Conclusion 11
6. References 12

1

Project Overview

Title

Professional Players' Rating Prediction Using Linear Regression

Authors

Arockiasamy, Thamson Antony
Class *of* 2025
MS in Business Analytics
Jack Welch College of Business & Technology
arockiasamyt@mail.sacredheart.edu

Mallipeddi, Sai Keerthana
Class *of* 2025
MS in Business Analytics
Jack Welch College of Business & Technology
mallipeddis3@mail.sacredheart.edu

Motivation

In 2024, the eSports industry is evaluated to be around 4 billion USD globally and growing exponentially. A projected compound annual growth rate (CAGR) of over 21%, potentially surpassing $34 billion by 2034.

The eSports industry's growth trajectory appears unstoppable, driven by increasing investments from major corporations, recognition as a legitimate sporting category, and the continued engagement of a tech-savvy global audience.

As eSports bridges the gap between digital entertainment and competitive athletics, it represents a defining shift in how the world engages with games and sports in the digital age.

It has birthed a thriving ecosystem that includes professional teams, dedicated players, content creators, sponsors, advertisers, game developers, and fans. The rise of eSports has also spurred interest in adjacent industries, such as gaming hardware, merchandise, virtual reality, and game-based education.

In this promising business arena, it is imperative to identify and recruit professional players around the globe by companies in the sponsoring teams. To close contracts with potentially successful players ahead of the competition, an extensive model to predict and identify them is required using statistics available in the public domain.

2

The What and Why

Premise

To expedite this requirement of building an effective data driven predictive model, we are utilizing the statistics that are available about the professional level players in the popular game Counter Strike: Global Offense™.

Hypothesis

1. Player performance metrics will have a strong positive correlation with their professional ratings. The ratings are the quantitatively measurable through performance statistics.

2. Team factors such as team ranking, win rate, and synergy with teammates significantly influence a player's individual rating.

3. External factors like media presence, fan popularity or marketability have a secondary but measurable effect on professional rating.

Why do this

By examining factors influencing professional player ratings, this research benefits various stakeholders: teams, scouts, analysts, and even players themselves.

Current rating systems may underutilize advanced analytics, and this research provides insights into incorporating predictive modeling for enhanced accuracy.

With the advent of machine learning and big data, this analysis explores cutting-edge techniques to refine player evaluations.

Exploring factors beyond performance—such as media bias or popularity—can help improve the fairness of rating systems.

3

Dataset

Datatypes:
The dataset consists of below information:

Source:
The data is acquired from Kaggle [1].

Column Name	Description	Datatype	Type
Nick	Nickname/Username	Text	Qualitative
Country	Name of Country	Text	Qualitative
Stats_link	weblink of their Statistics	Text	Qualitative
Teams	Name of their Teams	Text	Qualitative
Maps_played	Number of maps the player has played	Number	Quantitative
Rounds_played	Number of rounds the player has played	Number	Quantitative
Kd_difference	Differences between Kill and Death	Number	Quantitative
Kd_ratio	Difference between Kill and Death	Float	Quantitative
Rating	Overall Rating	Number	Quantitative
Total_kill	Total number of kills by the player	Number	Quantitative
Headshot_percentage	Percentage of headshot among kills	Float	Quantitative
Total_Deaths	Total number of deaths	Number	Quantitative
Grenade_damage_per_round	Damage created by grenade in each round	Float	Quantitative
Kills_per_round	Number of kills in each round	Float	Quantitative
Assists_per_round	Number of assists in each round	Float	Quantitative
Deaths_per_round	Number of deaths in each round	Float	Quantitative
Teammate_saved_per_round	Number of times the teammates were saved in each round	Float	Quantitative
Saved_by_teammate_per_round	Number of times a teammate saved	Float	Quantitative
Saved_by_teammate_per_round	Number of times a teammate saved	Float	Quantitative
Impact	Impact created by the player	Float	Quantitative

4

Statistical Data Analysis

4.1. Housekeeping

The foremost task is to import all the necessary libraries to handle input data, create dataframes, perform numerical calculations, develop maps, do exploratory data analysis, and build, train, and test a linear regression model.

```
In [1]: import folium # To create maps
        import numpy as np # To perform calculations
        import pandas as pd # To work with dataframes
        import seaborn as sns # To create plots
        from sklearn.model_selection import train_test_split # To split the data into training and testing sets
        import matplotlib.pyplot as plt # To create plots
        from sklearn.linear_model import LinearRegression # To create a linear regression model
        from sklearn.metrics import mean_squared_error, r2_score # To evaluate the model
```

Fig 4.1.1. The output shows the names of the libraries imported and the reason for import in the comment

The input file is then imported from its location and loaded into a dataframe using the Pandas library.

```
In [2]: df = pd.read_csv(f"hltv_playerStats-complete.csv")
        df.head(5)
```

Out[2]:

	nick	country	stats_link	teams	maps_played	rounds_played	kd_difference	kd_ratio	rating	total_kills	headsho
0	ZywOo	France	https://www.hltv.org//stats/players/11893/zywoo	['Vitality', 'aAa']	970	25491	5917	1.38	1.27	21602	
1	s1mple	Ukraine	https://www.hltv.org//stats/players/7998/s1mple	['Natus Vincere']	1532	40464	8864	1.34	1.25	34647	
2	sh1ro	Russia	https://www.hltv.org//stats/players/16920/sh1ro	['Gambit Youngsters', 'Gambit']	847	22465	5361	1.45	1.23	17320	
3	deko	Russia	https://www.hltv.org//stats/players/20113/deko	['1WIN']	378	10219	2225	1.37	1.22	8219	
4	Kaze	Malaysia	https://www.hltv.org//stats/players/8950/kaze	['ViCi', 'Flash', 'MVP.karnal']	829	21617	4118	1.32	1.20	16957	

Fig 4.1.2. The input file is imported, and the first five rows are displayed

4.2. Data Analysis

To know the general information such as indexes, column names, non-null content, and datatypes.

```
In [3]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 803 entries, 0 to 802
Data columns (total 20 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   nick                        803 non-null    object 
 1   country                     803 non-null    object 
 2   stats_link                  803 non-null    object 
 3   teams                       803 non-null    object 
 4   maps_played                 803 non-null    int64  
 5   rounds_played               803 non-null    int64  
 6   kd_difference               803 non-null    int64  
 7   kd_ratio                    803 non-null    float64
 8   rating                      803 non-null    float64
 9   total_kills                 803 non-null    int64  
 10  headshot_percentage         803 non-null    float64
 11  total_deaths                803 non-null    int64  
 12  grenade_damage_per_round    803 non-null    float64
 13  kills_per_round             803 non-null    float64
 14  assists_per_round           803 non-null    float64
 15  deaths_per_round            803 non-null    float64
 16  teammate_saved_per_round    803 non-null    float64
 17  saved_by_teammate_per_round 803 non-null    float64
 18  kast                        803 non-null    float64
 19  impact                      803 non-null    float64
dtypes: float64(11), int64(5), object(4)
memory usage: 125.6+ KB
```

Fig 4.2.1. The output shows general information about the dataframe imported from the input file

The description includes each column's count, mean, standard deviation, minimum, interquartile, and maximum values.

```
In [4]: df.describe().T
```

Out[4]:

	count	mean	std	min	25%	50%	75%	max
maps_played	803.0	833.174346	402.388811	374.00	500.50	734.00	1059.00	2169.00
rounds_played	803.0	21893.596513	10607.751477	9498.00	13227.00	19174.00	27881.00	56914.00
kd_difference	803.0	585.465753	1475.806605	-6238.00	-283.00	358.00	1313.50	8864.00
kd_ratio	803.0	1.035430	0.092114	0.74	0.98	1.03	1.09	1.45
rating	803.0	1.011880	0.066560	0.77	0.97	1.01	1.05	1.27
total_kills	803.0	15142.087173	7539.729631	5530.00	9092.50	13132.00	19214.00	40884.00
headshot_percentage	803.0	45.462017	8.416641	23.60	40.50	47.30	51.45	68.40
total_deaths	803.0	14556.518057	7018.031710	5994.00	8842.50	12603.00	18226.50	38351.00
grenade_damage_per_round	803.0	4.061395	1.187467	1.40	3.20	3.90	4.80	9.10
kills_per_round	803.0	0.688904	0.044705	0.52	0.66	0.69	0.72	0.86
assists_per_round	803.0	0.131046	0.017702	0.08	0.12	0.13	0.14	0.18
deaths_per_round	803.0	0.666949	0.030042	0.53	0.65	0.67	0.69	0.75
teammate_saved_per_round	803.0	0.096015	0.011236	0.04	0.09	0.10	0.10	0.14
saved_by_teammate_per_round	803.0	0.096862	0.013084	0.06	0.09	0.10	0.11	0.16
kast	803.0	70.112827	1.790944	63.30	69.00	70.10	71.40	76.30
impact	803.0	1.054944	0.100154	0.70	0.99	1.06	1.12	1.45

Fig 4.2.2. The output shows the description of the input data

4.3. Data Cleaning

To proceed with exploratory data analysis and build the model, the numerical and categorical variables must be separated into dataframes.

In [5]:
```
df_numeric = df.select_dtypes(include=[np.number])
df_categoric = df.select_dtypes(exclude=[np.number])

df_numeric_cols = list(df_numeric.columns)
df_categoric_cols = list(df_categoric.columns)
```

Fig 4.3.1. The numeric and categorical columns are separated and stored in dataframes

4.4. Exploratory Data Analysis

4.4.1. Histogram to find the value distribution

The histogram is plotted to understand the numeric data's value occurrence distribution.

In [6]:
```
df_numeric.hist(figsize=(10, 10)) # Histograms of the numeric columns
```

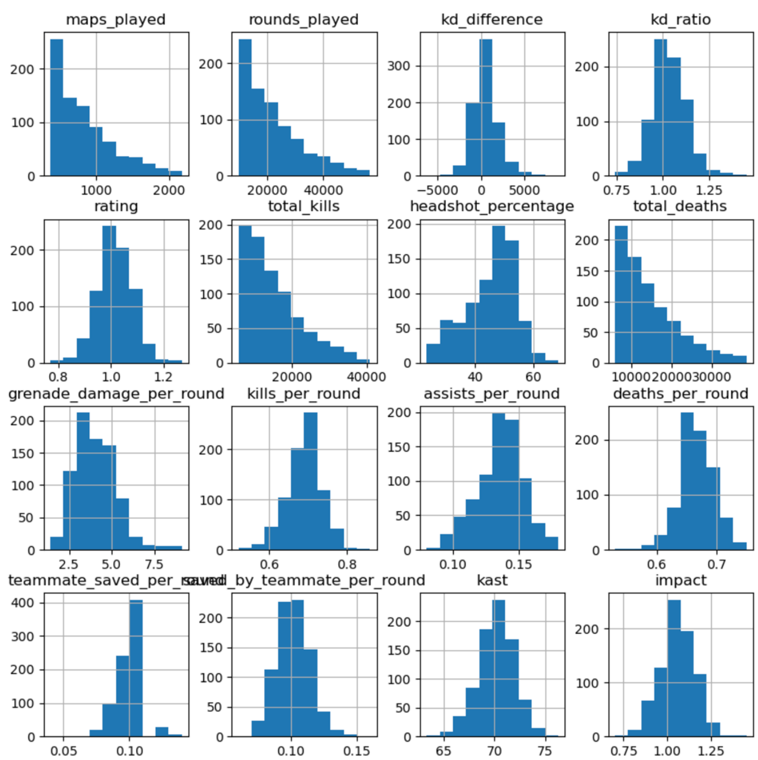

Fig 4.4.1. The output shows the value range and the number of occurrences in each numerical column

4.4.2. Analysis by Countries

To understand a broader trend globally to identify and focus on key demographics to scout players' performance, analysis by countries is required. Firstly, the number of players in each country is counted and stored in a dataframe with columns containing the country's name and count. The index is reset, and five random country's information is shown.

In [7]:
```
df_countrycounts = pd.DataFrame(df['country'].value_counts()) # Count of players from each country
df_countrycounts['countries'] = df_countrycounts.index # Create a new column with the country names
df_countrycounts = df_countrycounts.reset_index(drop=True) # Reset the index
df_countrycounts.columns = ['counts', 'country'] # Rename the columns

df_countrycounts.sample(5)  # Display a random sample of 5 rows
```

Out[7]:

	counts	country
23	8	Argentina
45	1	Taiwan
32	5	South Africa
37	4	Singapore
33	5	Latvia

Fig 4.4.2.1. The output shows the counts of five countries and their counts

The counted values are then plotted on the map.

In [8]:
```
url = 'https://raw.githubusercontent.com/python-visualization/folium/master/examples/data' # Base URL for the latest
country_shapes = f'{url}/world-countries.json' # URL for the country shapes
```

In [9]:
```
df_countrycounts.replace('United States', "United States of America", inplace = True) # Replace 'United States' with
```

In [10]:
```
m = folium.Map()    # Create a map
folium.Choropleth(
    geo_data=country_shapes,
    name='Players Counts by Country',
    data=df_countrycounts,
    columns=['country', 'counts'],
    key_on='feature.properties.name',
    nan_fill_color='grey'
).add_to(m)
m  # Display the map
```

Out[10]:

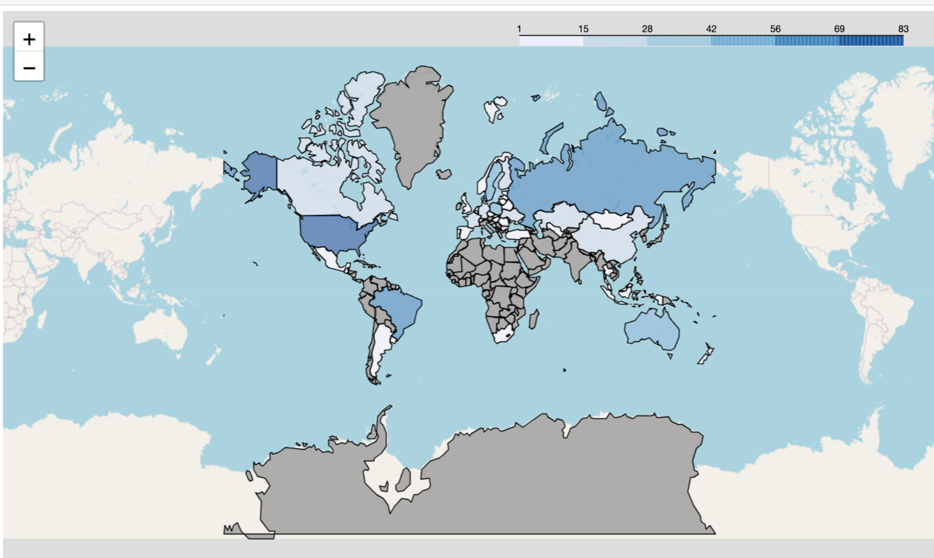

Fig 4.4.2.2. The output shows the counts of five countries and their counts

Then, the countries with less than five players are counted and stored in a list. The records of the countries on that list have been removed from the dataframe. Ten countries with the most players are stored in a list and displayed.

```
In [19]: to_remove_countries = list(df_countrycounts.loc[df_countrycounts['counts'] < 5]['country']) # List of countries with
         df_c = df[~df['country'].isin(to_remove_countries)] # Remove the countries with less than 5 players
         df_c_means = df_c.groupby('country').agg(lambda x: x.mean() if np.issubdtype(x.dtype, np.number) else x.head(1)) # C
         df_c_means = df_c_means.reset_index() # Reset the index
```

```
In [21]: to_plot_countries = list(df_countrycounts.head(11)['country']) # Plotting the top 10 countries with the most players
         df_to_plot_countries = df_c_means[df_c_means['country'].isin(to_plot_countries)] # Filter the data for the top 10 co
         df_to_plot_countries.country # Display the data
```

```
Out[21]: 1     Australia
         4        Brazil
         7         China
         9       Denmark
         11      Finland
         12       France
         13      Germany
         21       Poland
         24       Russia
         29       Sweden
         Name: country, dtype: object
```

```
In [22]: sns.set(rc={'figure.figsize':(10, 10)}) # Set the size of the plot
         sns.barplot(x='country', y='kd_difference', data=df_to_plot_countries); # Create a bar plot of the top 10 countries
```

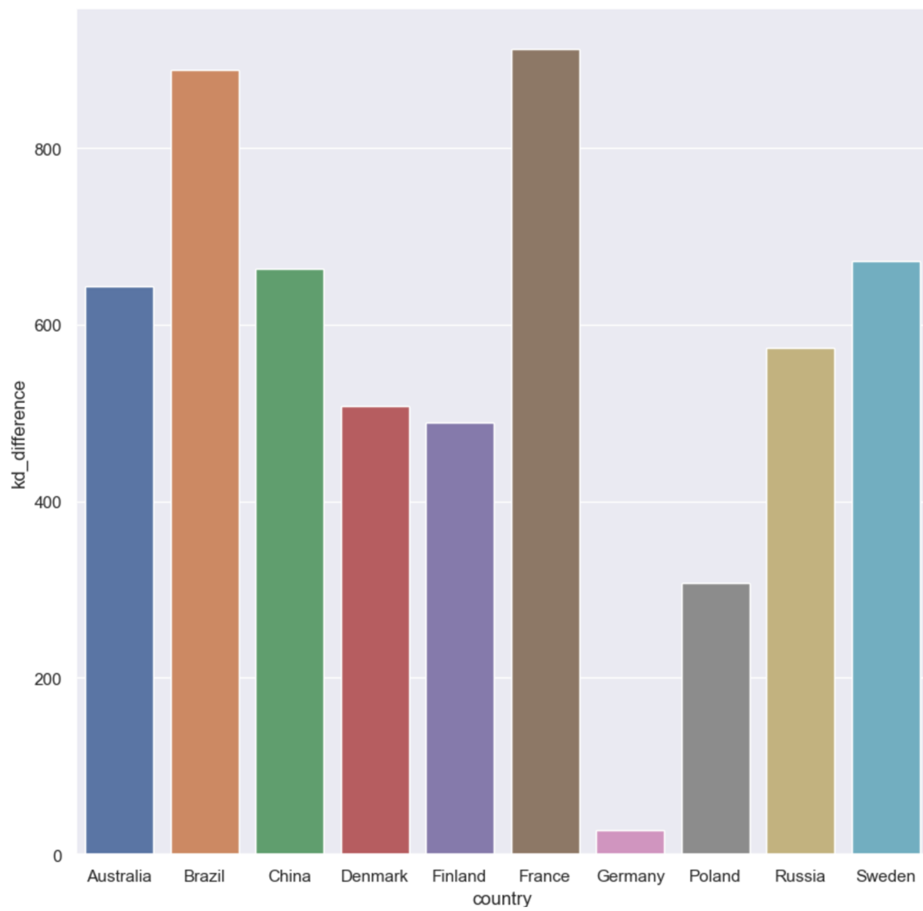

Fig 4.4.2.3. The output is a bar graph with the ten countries with the most significant number of players in alphabetical order

4.4.3. Plotting a Correlation Matrix

The columns are rearranged to facilitate an efficient way to determine the correlation using a heatmap between the numerical columns in the input data.

```
In [23]: rearrangement_cols = ['maps_played','rounds_played','kd_difference','kd_ratio','total_kills','headshot_percentage','
          'grenade_damage_per_round','kills_per_round','assists_per_round','deaths_per_round','teammate_saved_per_round','sav
          'kast','impact','rating'] # Rearrangement of the columns
         df_numeric = df_numeric[rearrangement_cols]
```

```
In [24]: correlation_matrix = df_numeric.corr() # Calculate the correlation matrix
         sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap="coolwarm") # Create a heatmap of the correlation matrix
         plt.title("Correlation Matrix of Numerical Variables") # Set the title of the plot
         plt.show() # Display the plot
```

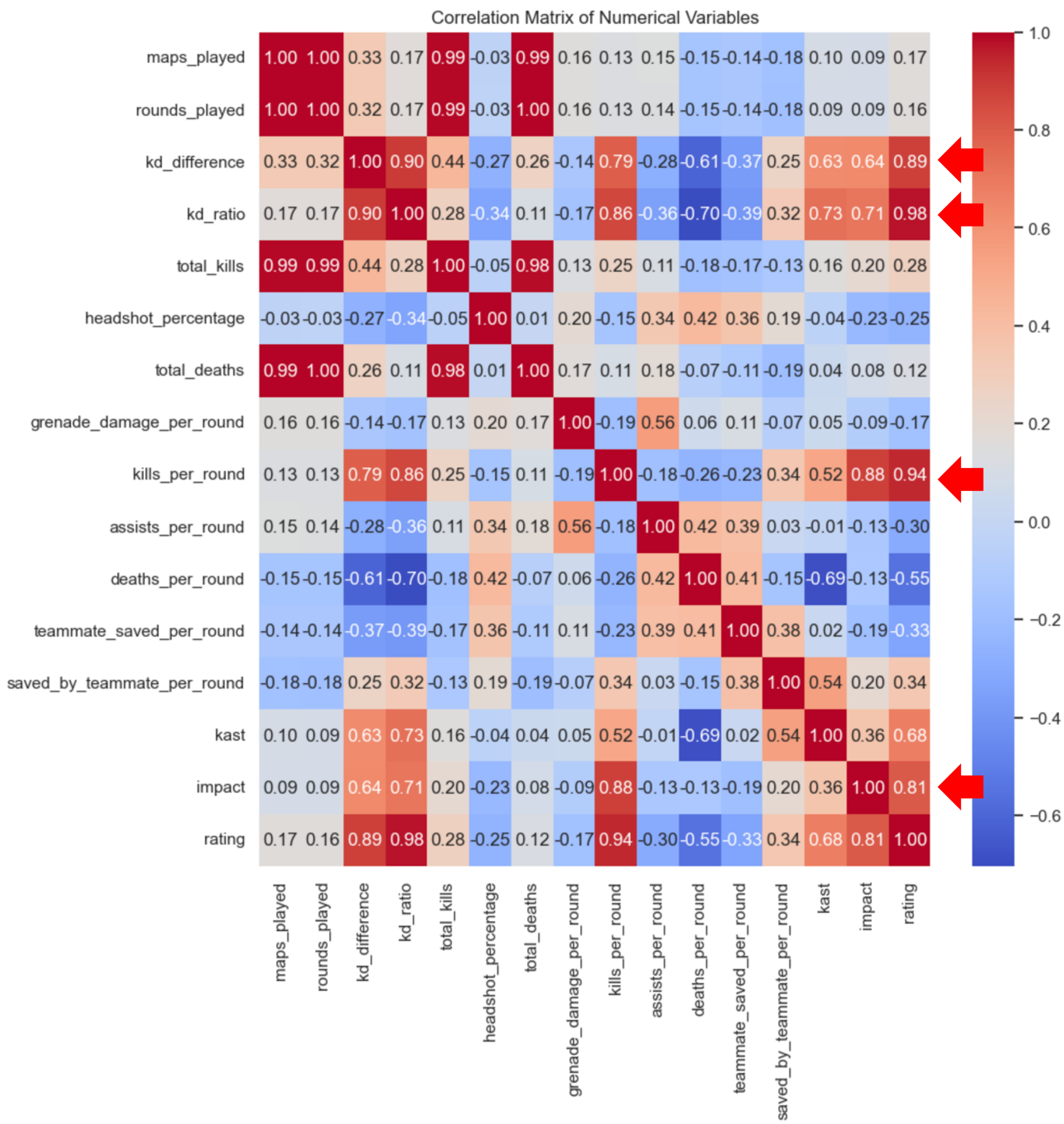

Fig 4.4.3.1. The heatmap is plotted with values indicating the correlation between the ratings and all the numerical values. From the output, we conclude that kd_difference, kd_ratio, kills_per_round, the impact has the highest correlation with ratings and they are selected as the features

4.5. Regression Analysis

4.5.1. Feature Selection and Fitting the Model

Using the heatmap results, the columns with the highest correlation are selected as the feature set to build the regression model. The selected features are converted to numeric format, and the missing values are removed. The model is then built with the selected features and the rating (to be predicted) as the target, with 60% of the input values to train.

```
In [25]: selected_features = ['rating', 'kd_ratio', 'kills_per_round', 'kd_difference', 'impact'] # Select the features with
         selected_features_cleaned = df[selected_features].apply(pd.to_numeric, errors='coerce').dropna() # Convert the selec

In [26]: X = selected_features_cleaned[['kd_ratio', 'kills_per_round', 'kd_difference', 'impact']] # Features
         y = selected_features_cleaned['rating']    # Target variable

In [27]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42) # Split the data into trai

In [28]: model = LinearRegression() # Train Linear Regression model
         model.fit(X_train, y_train)

Out[28]: ▾ LinearRegression
         LinearRegression()
```

Fig 4.5.1.1. The features are selected, converted to numeric, and prepared for training and testing, and the training data fits the linear regression model.

4.5.1. Predicting and Plotting the Results

The target values are then predicted with the model built for the test data from the input.

```
In [29]: y_pred = model.predict(X_test) # Predict on test data

In [30]: # Scatter plot of actual vs predicted values
         plt.figure(figsize=(8, 5))
         plt.scatter(y_test, y_pred, alpha=0.7, color='blue')
         plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linestyle='--')
         plt.title('Actual vs Predicted Ratings')
         plt.xlabel('Actual Ratings')
         plt.ylabel('Predicted Ratings')
         plt.show()
```

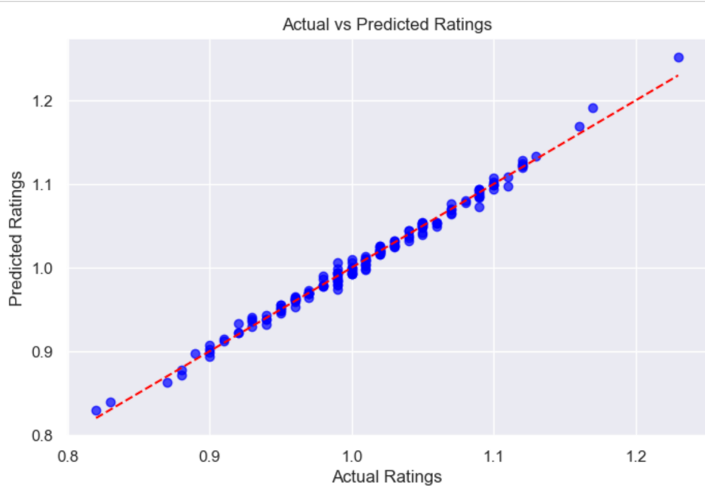

Fig 4.5.1.2. The actual ratings from the input and predicted ratings are plotted as blue and red plots, respectively

5

Results Review & Conclusion

The model built is evaluated by calculating the model coefficients, mean squared error, and the model's coefficient of determination.

```
In [36]: # Model Evaluation
         print("\nModel Coefficients:")
         print(model.coef_)
         print("\nMean Squared Error (MSE):", mean_squared_error(y_test, y_pred))
         print("R-squared (R²):", r2_score(y_test, y_pred))

         Model Coefficients:
         [4.66022327e-01 5.31377843e-01 2.61404548e-07 1.93661277e-02]

         Mean Squared Error (MSE): 3.610684947552941e-05
         R-squared (R²): 0.9918141377813222
```

Fig 5.1. The Output of the Evaluation.

The model coefficients represent the weights assigned to each feature in the model. These coefficients indicate the strength and direction of the relationship between each feature and the target variable.

The first coefficient (0.466) and second coefficient (0.531) suggest that the first feature has a positive relationship with the target variable. The third coefficient (2.614e-07) is close to zero, implying that the third feature has a negligible effect on the target variable. The fourth coefficient (0.019) indicates a slight positive relationship between the fourth feature and the target variable.

The Mean Squared Error (MSE) measures the average squared difference between the actual and predicted values. It indicates the model's prediction accuracy. A lower MSE value indicates better model performance. The MSE value of 3.61 achieved is minimal, suggesting that the model's predictions are very close to the actual values, indicating high accuracy.

The R-squared (R^2) value measures how well the model explains the variability of the target variable. It ranges from 0 to 1, with higher values indicating better model performance. The R-squared value [0.9918141377813222] is close to 1, indicating that the model explains approximately 99.18% of the variability in the target variable. This suggests that the model fits the data very well.

Hence, the model coefficients indicate the strength and direction of the relationship between each feature and the target variable. The meager MSE value suggests the model's highly accurate predictions. The high R-squared value indicates that the model explains a significant portion of the variability in the target variable, suggesting a good fit. Through this experiment the first hypothesis of Player performance metrics having strong positive correlation with their professional ratings is proven.

6

References

[1] 'Pro Players Data Analysis and Rating Prediction' by Saad Azam at Kaggle.
[Pro Players Data Analysis and Rating Prediction]

[2] [GitHub Repo of this Project]

End of Document

Sacred Heart University
Jack Welch College of Business & Technology
www.sacredheart.edu