The github link here have this whole document, Judpyters noteboo, HTML version and dataset:

<https://github.com/alijaweddelawari/AI-STUDIO/upload>

Binary Classification of eSports Match Outcomes Using Logistic Regression

**Introduction**

In this assignment the writer will be figuring out how data can predict the outcome off the sports and other tournaments. Or in this case, the E sports tournament. The writers focus is on predicting whether a team wins or loses using a powerful statistical method called logistic regression. It's like finding patterns in scores, player stats, and team strategies to uncover what leads to victory. The dataset in here which the essaer is working with is packed with numbers from actual games like kills, gold earned, and which players were in which roles. By analysing these numbers and applying logistic regression, the aim is to discover which factors truly influence the outcome of a match. The assignment in hand isn't just about numbers and formulas it's about understanding how data can provide insights into competitive gaming. The goal is to find on how mathematics can predict outcomes in the dynamic world of gaming.

**Objective**

The writer will use logistic regression as a tool to predict eSports match outcomes based on detailed game statistics. This analysis will uncover key factors influencing success in competitive gaming like esports etc.

**Data Overview**

The code below reads data from three CSV files related to the League of Legends World Championship 2019 player information (wc\_players.csv), match details (wc\_matches.csv), and champion statistics (wc\_champions.csv). the writer is using the Pandas library to loads each file into a DataFrame.

**Players Dataset:** Shows the first 2000 rows of player information.

**Matches Dataset:** Displays the first 1000 rows of match details.

**Champions Dataset:** Prints the first 1000 rows of champion statistics.

This code snippet is useful for getting a quick look at the data structure and initial entries of each dataset related to the 2019 League of Legends World Championship.

import pandas as pd

file\_players = 'wc\_players.csv'

file\_matches = 'wc\_matches.csv'

file\_champions = 'wc\_champions.csv'

df\_players = pd.read\_csv(file\_players)

df\_matches = pd.read\_csv(file\_matches)

df\_champions = pd.read\_csv(file\_champions)

print("Players Dataset:")

print(df\_players.head(2000))

print("\nMatches Dataset:")

print(df\_matches.head(1000))

print("\nChampions Dataset:")

print(df\_champions.head(1000))

Players Dataset:

Unnamed: 0 date side position player team \

0 0 43740.286146 Blue Top Evi DetonatioN FocusMe

1 1 43740.286146 Blue Jungle Steal DetonatioN FocusMe

2 2 43740.286146 Blue Middle Ceros DetonatioN FocusMe

3 3 43740.286146 Blue ADC Yutapon DetonatioN FocusMe

4 4 43740.286146 Blue Support Gaeng DetonatioN FocusMe

... ... ... ... ... ... ...

1185 1421 43779.393137 Red Top Wunder G2 Esports

1186 1422 43779.393137 Red Jungle Jankos G2 Esports

1187 1423 43779.393137 Red Middle Caps G2 Esports

1188 1424 43779.393137 Red ADC Perkz G2 Esports

1189 1425 43779.393137 Red Support Mikyx G2 Esports

champion ban1 ban2 ban3 ... gdat15 xpat10 oppxpat10 \

0 Gnar Qiyana Gangplank Akali ... -798 4530 5051

1 Ekko Qiyana Gangplank Akali ... -1366 3679 3928

2 Nocturne Qiyana Gangplank Akali ... -629 4751 4533

3 Lucian Qiyana Gangplank Akali ... -1829 3526 3532

4 Thresh Qiyana Gangplank Akali ... -1156 2731 2376

... ... ... ... ... ... ... ... ...

1185 Ryze Pantheon Qiyana Rakan ... 692 4330 4611

1186 Jarvan IV Pantheon Qiyana Rakan ... 279 2926 2699

1187 Veigar Pantheon Qiyana Rakan ... -169 4711 4430

1188 Ezreal Pantheon Qiyana Rakan ... -1495 3194 3887

1189 Nautilus Pantheon Qiyana Rakan ... -1004 2485 1766

xpdat10 csat10 oppcsat10 csdat10 csat15 oppcsat15 csdat15

0 -521 76 90 -14 116 122 -6

1 -249 60 64 -4 88 105 -17

2 218 88 85 3 131 150 -19

3 -6 87 76 11 119 134 -15

4 355 4 3 1 6 5 1

... ... ... ... ... ... ... ...

1185 -281 88 65 23 153 105 48

1186 227 48 55 -7 80 81 -1

1187 281 89 78 11 129 122 7

1188 -693 71 90 -19 125 151 -26

1189 719 14 3 11 23 6 17

[1190 rows x 91 columns]

Matches Dataset:

Unnamed: 0 team1 team2 winner \

0 0 Fnatic SK Telecom T1 SK Telecom T1

1 1 Royal Never Give Up Clutch Gaming Royal Never Give Up

2 2 Invictus Gaming ahq eSports Club Invictus Gaming

3 3 DAMWON Gaming Team Liquid Team Liquid

4 4 J Team FunPlus Phoenix J Team

.. ... ... ... ...

76 76 Flamengo eSports Royal Youth Royal Youth

77 77 DAMWON Gaming Lowkey Esports DAMWON Gaming

78 81 Clutch Gaming Royal Youth Clutch Gaming

79 84 Hong Kong Attitude Isurus Hong Kong Attitude

80 88 Splyce Unicorns Of Love Splyce

date pbp\_caster color\_caster mvp \

0 2019-10-12 12:00:00 Atlus Froskurinn, Kobe Faker

1 2019-10-12 13:00:00 Atlus Froskurinn, Kobe Langx

2 2019-10-12 14:00:00 Atlus Froskurinn, Kobe Rookie

3 2019-10-12 15:00:00 Phreak Azael, Spawn Impact

4 2019-10-12 16:00:00 Phreak Azael, Spawn FoFo

.. ... ... ... ...

76 2019-10-05 16:00:00 Drakos Ender Armut

77 2019-10-07 11:00:00 Atlus Jatt ShowMaker

78 2019-10-07 16:00:00 Medic EGym Cody Sun

79 2019-10-08 11:00:00 Medic Spawn Kaiwing

80 2019-10-08 16:00:00 Drakos Ender Kobbe

blue red

0 Fnatic SK Telecom T1

1 Royal Never Give Up Clutch Gaming

2 Invictus Gaming ahq eSports Club

3 DAMWON Gaming Team Liquid

4 J Team FunPlus Phoenix

.. ... ...

76 Royal Youth Flamengo eSports

77 Lowkey Esports DAMWON Gaming

78 Clutch Gaming Royal Youth

79 Hong Kong Attitude Isurus

80 Unicorns Of Love Splyce

[81 rows x 10 columns]

Champions Dataset:

Unnamed: 0 champion sum\_total win\_total lose\_total winrate\_total \

0 1 Kai'Sa 71 37 34 52%

1 2 Lee Sin 58 30 28 52%

2 3 Xayah 57 35 22 61%

3 4 Nautilus 52 28 24 54%

4 5 Gragas 49 32 17 65%

.. ... ... ... ... ... ...

93 94 Viktor 1 0 1 0%

94 95 Volibear 1 0 1 0%

95 96 Xin Zhao 1 0 1 0%

96 97 Ziggs 1 0 1 0%

97 98 Zilean 1 0 1 0%

sum\_blue\_side win\_blue\_side lose\_blue\_side winrate\_blue\_side ... \

0 30 15 15 50% ...

1 21 11 10 52% ...

2 33 21 12 64% ...

3 27 17 10 63% ...

4 26 16 10 62% ...

.. ... ... ... ... ...

93 1 0 1 0% ...

94 1 0 1 0% ...

95 0 0 0 0% ...

96 0 0 0 0% ...

97 0 0 0 0% ...

winrate\_30\_35\_min matches\_35\_40\_min win\_lose\_35\_40\_min \

0 46% 11 4-7

1 45% 10 6-4

2 70% 9 3-6

3 47% 11 7-4

4 59% 7 6-1

.. ... ... ...

93 0% 0 0-0

94 0% 0 0-0

95 0% 0 0-0

96 0% 0 0-0

97 0% 0 0-0

winrate\_35\_40\_min matches\_40\_45\_min win\_lose\_40\_45\_min winrate\_40\_45\_min \

0 36% 5 3-2 60%

1 60% 8 5-3 63%

2 33% 5 2-3 40%

3 64% 5 3-2 60%

4 86% 6 6-0 100%

.. ... ... ... ...

93 0% 0 0-0 0%

94 0% 0 0-0 0%

95 0% 0 0-0 0%

96 0% 0 0-0 0%

97 0% 0 0-0 0%

matches\_more\_45\_min win\_lose\_more\_45\_min winrate\_more\_45\_min

0 1 1-0 100%

1 1 0-1 0%

2 2 2-0 100%

3 2 2-0 100%

4 2 1-1 50%

.. ... ... ...

93 0 0-0 0%

94 0 0-0 0%

95 0 0-0 0%

96 1 0-1 0%

97 0 0-0 0%

[98 rows x 32 columns]

**Exploring Data Details**

The code below, not only notes the data set into Panda daframes from the CSV files, but also provide additional information and insights for each dataset.

**Players Dataset:** this is used below to displays the first few rows of player data using df\_players.head() and provides a summary of the DataFrame structure with extra commonds.

**Matches Dataset:** this one shows the initial rows of match data with df\_matches.head() and also provide an overview of the DataFrame structure and information using df\_matches.info() with is also a commond.

**Champions Dataset:** CD is used here to print the initial row of champion statistics using df\_champions.head() and offers a summary of the DataFrame structure and information using df\_champions.info().This includes everything from number of entries, column names, and data types specific to champion attributes.

file\_players = 'wc\_players.csv'

file\_matches = 'wc\_matches.csv'

file\_champions = 'wc\_champions.csv'

df\_players = pd.read\_csv(file\_players)

df\_matches = pd.read\_csv(file\_matches)

df\_champions = pd.read\_csv(file\_champions)

print("Players Dataset:")

print(df\_players.head())

print(df\_players.info())

print("\nMatches Dataset:")

print(df\_matches.head())

print(df\_matches.info())

print("\nChampions Dataset:")

print(df\_champions.head())

print(df\_champions.info())

Players Dataset:

Unnamed: 0 date side position player team \

0 0 43740.286146 Blue Top Evi DetonatioN FocusMe

1 1 43740.286146 Blue Jungle Steal DetonatioN FocusMe

2 2 43740.286146 Blue Middle Ceros DetonatioN FocusMe

3 3 43740.286146 Blue ADC Yutapon DetonatioN FocusMe

4 4 43740.286146 Blue Support Gaeng DetonatioN FocusMe

champion ban1 ban2 ban3 ... gdat15 xpat10 oppxpat10 xpdat10 \

0 Gnar Qiyana Gangplank Akali ... -798 4530 5051 -521

1 Ekko Qiyana Gangplank Akali ... -1366 3679 3928 -249

2 Nocturne Qiyana Gangplank Akali ... -629 4751 4533 218

3 Lucian Qiyana Gangplank Akali ... -1829 3526 3532 -6

4 Thresh Qiyana Gangplank Akali ... -1156 2731 2376 355

csat10 oppcsat10 csdat10 csat15 oppcsat15 csdat15

0 76 90 -14 116 122 -6

1 60 64 -4 88 105 -17

2 88 85 3 131 150 -19

3 87 76 11 119 134 -15

4 4 3 1 6 5 1

[5 rows x 91 columns]

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1190 entries, 0 to 1189

Data columns (total 91 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Unnamed: 0 1190 non-null int64

1 date 1190 non-null float64

2 side 1190 non-null object

3 position 1190 non-null object

4 player 1190 non-null object

5 team 1190 non-null object

6 champion 1190 non-null object

7 ban1 1190 non-null object

8 ban2 1190 non-null object

9 ban3 1190 non-null object

10 ban4 1190 non-null object

11 ban5 1185 non-null object

12 gamelength 1190 non-null float64

13 result 1190 non-null int64

14 k 1190 non-null int64

15 d 1190 non-null int64

16 a 1190 non-null int64

17 teamkills 1190 non-null int64

18 teamdeaths 1190 non-null int64

19 doubles 1190 non-null int64

20 triples 1190 non-null int64

21 quadras 1190 non-null int64

22 pentas 1190 non-null int64

23 fb 1190 non-null int64

24 fbassist 1190 non-null int64

25 fbvictim 1190 non-null int64

26 fbtime 1190 non-null float64

27 kpm 1190 non-null float64

28 okpm 1190 non-null float64

29 ckpm 1190 non-null float64

30 fd 1190 non-null int64

31 fdtime 1190 non-null float64

32 teamdragkills 1190 non-null int64

33 oppdragkills 1190 non-null int64

34 elementals 1190 non-null int64

35 oppelementals 1190 non-null int64

36 firedrakes 1190 non-null int64

37 waterdrakes 1190 non-null int64

38 earthdrakes 1190 non-null int64

39 airdrakes 1190 non-null int64

40 elders 1190 non-null int64

41 oppelders 1190 non-null int64

42 herald 1190 non-null int64

43 heraldtime 0 non-null float64

44 ft 1190 non-null int64

45 fttime 1190 non-null float64

46 firstmidouter 1190 non-null int64

47 firsttothreetowers 1190 non-null int64

48 teamtowerkills 1190 non-null int64

49 opptowerkills 1190 non-null int64

50 fbaron 1150 non-null float64

51 fbarontime 1150 non-null float64

52 teambaronkills 1190 non-null int64

53 oppbaronkills 1190 non-null int64

54 dmgtochamps 1190 non-null int64

55 dmgtochampsperminute 1190 non-null float64

56 dmgshare 1190 non-null float64

57 earnedgoldshare 1190 non-null float64

58 wards 1190 non-null int64

59 wpm 1190 non-null float64

60 wardshare 1190 non-null float64

61 wardkills 1190 non-null int64

62 wcpm 1190 non-null float64

63 visionwards 1190 non-null int64

64 visionwardbuys 1190 non-null int64

65 visiblewardclearrate 1190 non-null object

66 invisiblewardclearrate 1190 non-null object

67 totalgold 1190 non-null int64

68 earnedgpm 1190 non-null float64

69 goldspent 1190 non-null int64

70 gspd 1190 non-null float64

71 minionkills 1190 non-null int64

72 monsterkills 1190 non-null int64

73 monsterkillsownjungle 1190 non-null int64

74 monsterkillsenemyjungle 1190 non-null int64

75 cspm 1190 non-null float64

76 goldat10 1190 non-null int64

77 oppgoldat10 1190 non-null int64

78 gdat10 1190 non-null int64

79 goldat15 1190 non-null int64

80 oppgoldat15 1190 non-null int64

81 gdat15 1190 non-null int64

82 xpat10 1190 non-null int64

83 oppxpat10 1190 non-null int64

84 xpdat10 1190 non-null int64

85 csat10 1190 non-null int64

86 oppcsat10 1190 non-null int64

87 csdat10 1190 non-null int64

88 csat15 1190 non-null int64

89 oppcsat15 1190 non-null int64

90 csdat15 1190 non-null int64

dtypes: float64(20), int64(59), object(12)

memory usage: 846.1+ KB

None

Matches Dataset:

Unnamed: 0 team1 team2 winner \

0 0 Fnatic SK Telecom T1 SK Telecom T1

1 1 Royal Never Give Up Clutch Gaming Royal Never Give Up

2 2 Invictus Gaming ahq eSports Club Invictus Gaming

3 3 DAMWON Gaming Team Liquid Team Liquid

4 4 J Team FunPlus Phoenix J Team

date pbp\_caster color\_caster mvp \

0 2019-10-12 12:00:00 Atlus Froskurinn, Kobe Faker

1 2019-10-12 13:00:00 Atlus Froskurinn, Kobe Langx

2 2019-10-12 14:00:00 Atlus Froskurinn, Kobe Rookie

3 2019-10-12 15:00:00 Phreak Azael, Spawn Impact

4 2019-10-12 16:00:00 Phreak Azael, Spawn FoFo

blue red

0 Fnatic SK Telecom T1

1 Royal Never Give Up Clutch Gaming

2 Invictus Gaming ahq eSports Club

3 DAMWON Gaming Team Liquid

4 J Team FunPlus Phoenix

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 81 entries, 0 to 80

Data columns (total 10 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Unnamed: 0 81 non-null int64

1 team1 81 non-null object

2 team2 81 non-null object

3 winner 81 non-null object

4 date 81 non-null object

5 pbp\_caster 81 non-null object

6 color\_caster 81 non-null object

7 mvp 81 non-null object

8 blue 81 non-null object

9 red 81 non-null object

dtypes: int64(1), object(9)

memory usage: 6.5+ KB

None

Champions Dataset:

Unnamed: 0 champion sum\_total win\_total lose\_total winrate\_total \

0 1 Kai'Sa 71 37 34 52%

1 2 Lee Sin 58 30 28 52%

2 3 Xayah 57 35 22 61%

3 4 Nautilus 52 28 24 54%

4 5 Gragas 49 32 17 65%

sum\_blue\_side win\_blue\_side lose\_blue\_side winrate\_blue\_side ... \

0 30 15 15 50% ...

1 21 11 10 52% ...

2 33 21 12 64% ...

3 27 17 10 63% ...

4 26 16 10 62% ...

winrate\_30\_35\_min matches\_35\_40\_min win\_lose\_35\_40\_min winrate\_35\_40\_min \

0 46% 11 4-7 36%

1 45% 10 6-4 60%

2 70% 9 3-6 33%

3 47% 11 7-4 64%

4 59% 7 6-1 86%

matches\_40\_45\_min win\_lose\_40\_45\_min winrate\_40\_45\_min \

0 5 3-2 60%

1 8 5-3 63%

2 5 2-3 40%

3 5 3-2 60%

4 6 6-0 100%

matches\_more\_45\_min win\_lose\_more\_45\_min winrate\_more\_45\_min

0 1 1-0 100%

1 1 0-1 0%

2 2 2-0 100%

3 2 2-0 100%

4 2 1-1 50%

[5 rows x 32 columns]

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 98 entries, 0 to 97

Data columns (total 32 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Unnamed: 0 98 non-null int64

1 champion 98 non-null object

2 sum\_total 98 non-null int64

3 win\_total 98 non-null int64

4 lose\_total 98 non-null int64

5 winrate\_total 98 non-null object

6 sum\_blue\_side 98 non-null int64

7 win\_blue\_side 98 non-null int64

8 lose\_blue\_side 98 non-null int64

9 winrate\_blue\_side 98 non-null object

10 sum\_red\_side 98 non-null int64

11 win\_red\_side 98 non-null int64

12 lose\_red\_side 98 non-null int64

13 winrate\_red\_side 98 non-null object

14 matches\_less\_25\_min 98 non-null int64

15 win\_lose\_less\_25\_min 98 non-null object

16 winrate\_less\_25\_min 98 non-null object

17 matches\_25\_to\_30\_min 98 non-null int64

18 win\_lose\_25\_to\_30\_min 98 non-null object

19 winrate\_25\_30\_min 98 non-null object

20 matches\_30\_35\_min 98 non-null int64

21 win\_lose\_30\_35\_min 98 non-null object

22 winrate\_30\_35\_min 98 non-null object

23 matches\_35\_40\_min 98 non-null int64

24 win\_lose\_35\_40\_min 98 non-null object

25 winrate\_35\_40\_min 98 non-null object

26 matches\_40\_45\_min 98 non-null int64

27 win\_lose\_40\_45\_min 98 non-null object

28 winrate\_40\_45\_min 98 non-null object

29 matches\_more\_45\_min 98 non-null int64

30 win\_lose\_more\_45\_min 98 non-null object

31 winrate\_more\_45\_min 98 non-null object

dtypes: int64(16), object(16)

memory usage: 24.6+ KB

None

**Checking Missing Data**

The code below is responsible for finding, identifying and counting the missing values of all three datasets. By roaming these codes, the writer will be able to identify and check the missing data and values. This step is crucial to make sure the data is ready for the next step.

print("Missing values in Players Dataset:")

print(df\_players.isnull().sum())

print("\nMissing values in Matches Dataset:")

print(df\_matches.isnull().sum())

print("\nMissing values in Champions Dataset:")

print(df\_champions.isnull().sum())

Missing values in Players Dataset:

Unnamed: 0 0

date 0

side 0

position 0

player 0

..

oppcsat10 0

csdat10 0

csat15 0

oppcsat15 0

csdat15 0

Length: 91, dtype: int64

Missing values in Matches Dataset:

Unnamed: 0 0

team1 0

team2 0

winner 0

date 0

pbp\_caster 0

color\_caster 0

mvp 0

blue 0

red 0

dtype: int64

Missing values in Champions Dataset:

Unnamed: 0 0

champion 0

sum\_total 0

win\_total 0

lose\_total 0

winrate\_total 0

sum\_blue\_side 0

win\_blue\_side 0

lose\_blue\_side 0

winrate\_blue\_side 0

sum\_red\_side 0

win\_red\_side 0

lose\_red\_side 0

winrate\_red\_side 0

matches\_less\_25\_min 0

win\_lose\_less\_25\_min 0

winrate\_less\_25\_min 0

matches\_25\_to\_30\_min 0

win\_lose\_25\_to\_30\_min 0

winrate\_25\_30\_min 0

matches\_30\_35\_min 0

win\_lose\_30\_35\_min 0

winrate\_30\_35\_min 0

matches\_35\_40\_min 0

win\_lose\_35\_40\_min 0

winrate\_35\_40\_min 0

matches\_40\_45\_min 0

win\_lose\_40\_45\_min 0

winrate\_40\_45\_min 0

matches\_more\_45\_min 0

win\_lose\_more\_45\_min 0

winrate\_more\_45\_min 0

dtype: int64

**Analysis of Gold Difference at 15 Minutes in League of Legends World Championship 2019**

Code below is used by the writer to analyze on the distribution of goal difference at 15 minutes among players participating.

**Data Handling:** Uses Pandas and NumPy for data manipulation and handling and also replaces infinite value.

**Data Merging:**Merges df\_players and selected columns into df\_merged.

Visualization: in the next step, the writer uses Seaborn to visualise.

**Statistical Measures:**Computes statistical measures.

**Additionaly;**

1. dentifies and labels the top 3 teams (top\_teams) with the highest number of players based on team counts in df\_merged.
2. Calculates the mean gdat15 for each top team and annotates these values on the histogram for comparison.

So basically this code provides a visual and statistical of early-game performance metrics.

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

df\_players.replace([np.inf, -np.inf], np.nan, inplace=True)

df\_matches.replace([np.inf, -np.inf], np.nan, inplace=True)

df\_merged = pd.merge(df\_players, df\_matches[['team1', 'team2', 'winner']], left\_on='team', right\_on='team1', how='left')

plt.figure(figsize=(12, 8))

sns.histplot(df\_players['gdat15'].dropna(), bins=30, kde=True, color='lightcoral', edgecolor='black')

plt.title('Distribution of Gold Difference at 15 minutes', fontsize=16)

plt.xlabel('Gold Difference at 15 minutes', fontsize=14)

plt.ylabel('Count', fontsize=14)

plt.grid(True)

mean\_gdat15 = df\_players['gdat15'].mean()

median\_gdat15 = df\_players['gdat15'].median()

q1\_gdat15 = df\_players['gdat15'].quantile(0.25)

q3\_gdat15 = df\_players['gdat15'].quantile(0.75)

plt.axvline(mean\_gdat15, color='darkred', linestyle='--', linewidth=2, label=f'Mean: {mean\_gdat15:.2f}')

plt.axvline(median\_gdat15, color='darkgreen', linestyle='-', linewidth=2, label=f'Median: {median\_gdat15:.2f}')

plt.axvline(q1\_gdat15, color='darkblue', linestyle='-.', linewidth=1.5, label=f'Q1: {q1\_gdat15:.2f}')

plt.axvline(q3\_gdat15, color='darkblue', linestyle='-.', linewidth=1.5, label=f'Q3: {q3\_gdat15:.2f}')

top\_teams = df\_merged['team'].value\_counts().index[:3] # Example: Top 3 teams based on number of players

for team in top\_teams:

team\_data = df\_merged[df\_merged['team'] == team]

mean\_team\_gdat15 = team\_data['gdat15'].mean()

plt.text(mean\_team\_gdat15, 10, team, rotation=90, ha='center', va='bottom', fontsize=10, color='black')

plt.legend(fontsize=12)

plt.tight\_layout()

plt.show()

A graph of a number of different values

Description automatically generated with medium confidence

**Selection and Export of Key Features for  analysis**

In this code the writer selects specific features from the datasets related to the League of Legends World Championship 2019. For player data, he takes metrics like early-game gold difference (gdat15), minion kills (csat10, csat15), experience (xpat10), and associated differences (csdat10, csdat15). From match data, he takes the team identifiers (team1, team2), match outcome (winner), and commentary details (pbp\_caster, color\_caster). From champion statistics it takes overall win rates (winrate\_total), frequency on blue side (sum\_blue\_side), and success rates on the blue side (win\_blue\_side).then the writer creates new DataFrames for each dataset and subset and save them as CSV files (players\_selected.csv, matches\_selected.csv, champions\_selected.csv) for further analysis or reporting purposes. This process makes the data to relevant attributes cimportant for understanding player performances, match dynamics, and also champion strategies during the championship.

performance\_threshold = df\_players['gdat15'].mean()

df\_players['performance\_category'] = np.where(df\_players['gdat15'] >= performance\_threshold, 'Strong', 'Weak')

print(df\_players['performance\_category'].value\_counts())

performance\_category

Weak 595

Strong 595

Name: count, dtype: int64

**Determining Match Outcomes**

The code below assesses the outcome of each match in the League of Legends World Championship 2019 dataset. It creates a new column called match outcome, in the df\_matches DataFrame., the code also checks if team1 is the winner of the match. If true, the match outcome is labeled as 'Win'; otherwise, it is labeled as 'Loss'. Finally, it prints the count of win and loss outcomes using value\_counts(). This helps in understanding the distribution of match results and evaluating team performance throughout the championship.

df\_matches['match\_outcome'] = np.where(df\_matches['team1'] == df\_matches['winner'], 'Win', 'Loss')

print(df\_matches['match\_outcome'].value\_counts())

match\_outcome

Loss 42

Win 39

Name: count, dtype: int64

**Save updated datasets with target variables**

df\_players.to\_csv('players\_with\_target.csv', index=False)

df\_matches.to\_csv('matches\_with\_target.csv', index=False)

**Preparing Data for Machine Learning**

In this code, firsly the writer prepare the League of Legends World Championship 2019 player data for machine learning. Then using the code below he reads the player and match datasets from CSV files. then identify columns like side, position, team, champion, ban1, ban2 and ban3 in the df\_players DataFrame and convert these categorical variables into numerical values using LabelEncoder. After doing all that he will define our feature set X by dropping columns that are not relevant for training and set his target variable y as performance\_category, which classifies player performance. Then after that the writer will split the data into training and testing sets using train\_test\_split, allocating 20% of the data to the test set and setting a random state for reproducibility. Finally, he will print the shapes of the training and testing sets to confirm the split. This process is crucial for preparing the data for machine learning models to predict player performance based on ingame statistics and categorical information.

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler, LabelEncoder

df\_players = pd.read\_csv('players\_with\_target.csv')

df\_matches = pd.read\_csv('matches\_with\_target.csv')

cat\_columns = ['side', 'position', 'team', 'champion', 'ban1', 'ban2', 'ban3']

for col in cat\_columns:

df\_players[col] = LabelEncoder().fit\_transform(df\_players[col])

num\_columns = ['gdat15', 'xpat10', 'oppxpat10', 'xpdat10', 'csat10', 'oppcsat10', 'csdat10', 'csat15', 'oppcsat15', 'csdat15']

scaler = StandardScaler()

df\_players[num\_columns] = scaler.fit\_transform(df\_players[num\_columns])

X = df\_players.drop(['Unnamed: 0', 'date', 'player', 'performance\_category'], axis=1) # Adjust columns as needed

y = df\_players['performance\_category']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

print(f"Training set shape: X\_train {X\_train.shape}, y\_train {y\_train.shape}")

print(f"Testing set shape: X\_test {X\_test.shape}, y\_test {y\_test.shape}")

Training set shape: X\_train (952, 88), y\_train (952,)

Testing set shape: X\_test (238, 88), y\_test (238,)

**File paths or names**

file\_players = 'wc\_players.csv'

file\_matches = 'wc\_matches.csv'

file\_champions = 'wc\_champions.csv'

**Load datasets into pandas DataFrames**

df\_players = pd.read\_csv(file\_players)

df\_matches = pd.read\_csv(file\_matches)

df\_champions = pd.read\_csv(file\_champions)

**Preparing Player Performance Data**

The code the writer is using here preprocesses player performance data from the League of Legends World Championship 2019 for machine learning. First, the essar loads the player dataset from a CSV file to jupyter and then selects specific features like gold difference at 15 minutes.minutes, and their corresponding differences for the input data. The target variable also known as performance category, is encoded into numerical values to make it suitable for machine learning. The numerical features are made standard to make sure they have a consistent scale. The preprocessed data is then split into training and testing sets with an 80-20 ratio. Finally, the code prints the shapes of the training and testing sets to verify the data split. This preprocessing ensures the data is correctly made and scaled for training machine learning models to predict player performance.

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler, LabelEncoder

import numpy as np

df\_players = pd.read\_csv('players\_with\_target.csv')

df\_matches = pd.read\_csv('matches\_with\_target.csv')

X\_players = df\_players[['gdat15', 'csat10', 'xpat10', 'csdat10', 'csat15', 'csdat15']]

y\_players = df\_players['performance\_category']

le\_players = LabelEncoder()

y\_players = le\_players.fit\_transform(y\_players)

scaler\_players = StandardScaler()

X\_players = scaler\_players.fit\_transform(X\_players)

X\_train\_players, X\_test\_players, y\_train\_players, y\_test\_players = train\_test\_split(X\_players, y\_players, test\_size=0.2, random\_state=42)

print(X\_train\_players.shape, X\_test\_players.shape, y\_train\_players.shape, y\_test\_players.shape)

(952, 6) (238, 6) (952,) (238,)

**Preparing Match Data for Machine Learning**

The code below prepares match data from the League of Legends World Championship 2019 for machine learning. It selects features like team identifiers and casters and sets the match outcome as the target variable here. All the feature that are categorical and the target are encoded into numbers and dummy variables are created for the categorical data. The data is then split into training and testing sets with an 80-20 ratio. Finally, it prints the shapes of the training and testing sets to verify the data split, ensuring the data is ready for model training.

X\_matches = df\_matches[['team1', 'team2', 'pbp\_caster', 'color\_caster']]

y\_matches = df\_matches['match\_outcome']

le\_matches = LabelEncoder()

y\_matches = le\_matches.fit\_transform(y\_matches)

X\_matches = pd.get\_dummies(X\_matches, drop\_first=True)

X\_train\_matches, X\_test\_matches, y\_train\_matches, y\_test\_matches = train\_test\_split(X\_matches, y\_matches, test\_size=0.2, random\_state=42)

print(X\_train\_matches.shape, X\_test\_matches.shape, y\_train\_matches.shape, y\_test\_matches.shape)

(64, 69) (17, 69) (64,) (17,)

**Check for missing values or inconsistencies**

print(df\_players.info())

**Data Exploration**

The code reads the player dataset from a CSV file and performs an initial exploration. It loads the data into a DataFrame, then prints the first few rows to give a preview of the dataset's content. Additionally, it prints the dataset's information, including the number of entries, column names, data types, and any missing values. This initial exploration helps in understanding the structure and quality of the data before proceeding with further analysis or preprocessing steps.

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score

file\_players = 'wc\_players.csv'

df\_players = pd.read\_csv(file\_players)

print(df\_players.head())

print(df\_players.info())

Unnamed: 0 date side position player team \

0 0 43740.286146 Blue Top Evi DetonatioN FocusMe

1 1 43740.286146 Blue Jungle Steal DetonatioN FocusMe

2 2 43740.286146 Blue Middle Ceros DetonatioN FocusMe

3 3 43740.286146 Blue ADC Yutapon DetonatioN FocusMe

4 4 43740.286146 Blue Support Gaeng DetonatioN FocusMe

champion ban1 ban2 ban3 ... gdat15 xpat10 oppxpat10 xpdat10 \

0 Gnar Qiyana Gangplank Akali ... -798 4530 5051 -521

1 Ekko Qiyana Gangplank Akali ... -1366 3679 3928 -249

2 Nocturne Qiyana Gangplank Akali ... -629 4751 4533 218

3 Lucian Qiyana Gangplank Akali ... -1829 3526 3532 -6

4 Thresh Qiyana Gangplank Akali ... -1156 2731 2376 355

csat10 oppcsat10 csdat10 csat15 oppcsat15 csdat15

0 76 90 -14 116 122 -6

1 60 64 -4 88 105 -17

2 88 85 3 131 150 -19

3 87 76 11 119 134 -15

4 4 3 1 6 5 1

[5 rows x 91 columns]

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1190 entries, 0 to 1189

Data columns (total 91 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Unnamed: 0 1190 non-null int64

1 date 1190 non-null float64

2 side 1190 non-null object

3 position 1190 non-null object

4 player 1190 non-null object

5 team 1190 non-null object

6 champion 1190 non-null object

7 ban1 1190 non-null object

8 ban2 1190 non-null object

9 ban3 1190 non-null object

10 ban4 1190 non-null object

11 ban5 1185 non-null object

12 gamelength 1190 non-null float64

13 result 1190 non-null int64

14 k 1190 non-null int64

15 d 1190 non-null int64

16 a 1190 non-null int64

17 teamkills 1190 non-null int64

18 teamdeaths 1190 non-null int64

19 doubles 1190 non-null int64

20 triples 1190 non-null int64

21 quadras 1190 non-null int64

22 pentas 1190 non-null int64

23 fb 1190 non-null int64

24 fbassist 1190 non-null int64

25 fbvictim 1190 non-null int64

26 fbtime 1190 non-null float64

27 kpm 1190 non-null float64

28 okpm 1190 non-null float64

29 ckpm 1190 non-null float64

30 fd 1190 non-null int64

31 fdtime 1190 non-null float64

32 teamdragkills 1190 non-null int64

33 oppdragkills 1190 non-null int64

34 elementals 1190 non-null int64

35 oppelementals 1190 non-null int64

36 firedrakes 1190 non-null int64

37 waterdrakes 1190 non-null int64

38 earthdrakes 1190 non-null int64

39 airdrakes 1190 non-null int64

40 elders 1190 non-null int64

41 oppelders 1190 non-null int64

42 herald 1190 non-null int64

43 heraldtime 0 non-null float64

44 ft 1190 non-null int64

45 fttime 1190 non-null float64

46 firstmidouter 1190 non-null int64

47 firsttothreetowers 1190 non-null int64

48 teamtowerkills 1190 non-null int64

49 opptowerkills 1190 non-null int64

50 fbaron 1150 non-null float64

51 fbarontime 1150 non-null float64

52 teambaronkills 1190 non-null int64

53 oppbaronkills 1190 non-null int64

54 dmgtochamps 1190 non-null int64

55 dmgtochampsperminute 1190 non-null float64

56 dmgshare 1190 non-null float64

57 earnedgoldshare 1190 non-null float64

58 wards 1190 non-null int64

59 wpm 1190 non-null float64

60 wardshare 1190 non-null float64

61 wardkills 1190 non-null int64

62 wcpm 1190 non-null float64

63 visionwards 1190 non-null int64

64 visionwardbuys 1190 non-null int64

65 visiblewardclearrate 1190 non-null object

66 invisiblewardclearrate 1190 non-null object

67 totalgold 1190 non-null int64

68 earnedgpm 1190 non-null float64

69 goldspent 1190 non-null int64

70 gspd 1190 non-null float64

71 minionkills 1190 non-null int64

72 monsterkills 1190 non-null int64

73 monsterkillsownjungle 1190 non-null int64

74 monsterkillsenemyjungle 1190 non-null int64

75 cspm 1190 non-null float64

76 goldat10 1190 non-null int64

77 oppgoldat10 1190 non-null int64

78 gdat10 1190 non-null int64

79 goldat15 1190 non-null int64

80 oppgoldat15 1190 non-null int64

81 gdat15 1190 non-null int64

82 xpat10 1190 non-null int64

83 oppxpat10 1190 non-null int64

84 xpdat10 1190 non-null int64

85 csat10 1190 non-null int64

86 oppcsat10 1190 non-null int64

87 csdat10 1190 non-null int64

88 csat15 1190 non-null int64

89 oppcsat15 1190 non-null int64

90 csdat15 1190 non-null int64

dtypes: float64(20), int64(59), object(12)

memory usage: 846.1+ KB

None

**Preparing Data for Binary Outcome Prediction**

The code below prepares data from the player dataset to protect the weather a team wins or loses. It selects features like player position, early-game gold difference etc.then all these feature are extract into data frame. Other data like player position is saved numerically. Numerical features are scaled using StandardScaler to normalize their values. This preprocessing ensures that all features are ready for machine learning model training, improving prediction accuracy for match outcomes.

features = ['position', 'gdat10', 'xpat10', 'csat10', 'csdat10']

target = 'result' # Binary target variable (e.g., 'result' indicating win/loss)

df\_selected = df\_players[features + [target]].copy()

encoder = LabelEncoder()

df\_selected['position'] = encoder.fit\_transform(df\_selected['position'])

X = df\_selected.drop(columns=[target])

y = df\_selected[target]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

Initialize Logistic Regression model

model = LogisticRegression(random\_state=42)

**Train the model**

model.fit(X\_train\_scaled, y\_train)

A close-up of a tag

Description automatically generated

**Evaluating Logistic Regression Model Performance**

The code makes predictions using trained logistic regression model on the data set. Then calculate and evaluation method to assess the model performance and predicting match outcomes based on player statistics from the league. The evaluation matrics computed include accuracy, Percision, recall an F1 score. The results of the matric are printed to provide insight into the model performance. This evaluation step is crucial for understanding the effectiveness and reliability of the Logistic Regression model for this specific prediction task.

y\_pred = model.predict(X\_test\_scaled)

accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred)

recall = recall\_score(y\_test, y\_pred)

f1 = f1\_score(y\_test, y\_pred)

print(f'Accuracy: {accuracy:.2f}, Precision: {precision:.2f}, Recall: {recall:.2f}, F1-score: {f1:.2f}')

Accuracy: 0.62, Precision: 0.55, Recall: 0.72, F1-score: 0.62

**Conclusion**

Throughout this assignment on analyzing data from the League of Legends World Championship 2019 the writer performed essential steps to prepare, model, and evaluate predictive analytics which are the core blocks of this essay. He began by exploring and refining player and match datasets, selecting relevant features, and encoding categorical data. After splitting the data into training and testing sets, he contanuied to train a Logistic Regression model to predict match outcomes based on player statistics. Evaluating the model's performance with accuracy, precision, recall, and F1-score metrics confirmed its effectiveness in predicting win or loss outcomes. These steps underscored the importance of data preprocessing, feature selection, and model evaluation in extracting insights from esports data for decision-making.

DATASET LINK: <https://www.kaggle.com/datasets/ilyadziamidovich/league-of-legends-world-championship-2019/data>

NOTE:The student tried to use TselinMachine but despite Jupyters notebook being able to download it, somehow Mac terminal did not allow it to run.



Assessment Submission Form

|  |  |
| --- | --- |
| **Student Number**  (If this is group work, please  include the student numbers of all group participants) | GH1024093 |
| **Assessment Title** | Binary Classification of eSports Match Outcomes Using Logistic Regression |
| **Module Code** | B143 |
| **Module Title** | AI STUDIO |
| **Module Tutor** | [Amirhossein Jamalian](https://study.gisma.com/courses/3937/users/2053) |
| **Date Submitted** | 03/07/2024 |

**Declaration of Authorship**

I declare that all material in this assessment is my own work except where there is clear acknowledgement and appropriate reference to the work of others.

I fully understand that the unacknowledged inclusion of another person’s writings or ideas or works in this work may be considered plagiarism and that, should a formal investigation process confirms the allegation, I would be subject to the penalties associated with plagiarism, as per GISMA Business School, University of Applied Sciences’ regulations for academic misconduct.

Signed………………Ali Jawed Delawari……………. Date ………03/07/2024………………