# Title

In [239…

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
from sklearn.preprocessing import OrdinalEncoder, OneHotEncoder, MinMaxScaler, StandardScaler
from sklearn.metrics import roc_auc_score, classification_report, confusion_matrix
import shap
import warnings


warnings.simplefilter(action='ignore', category=FutureWarning)
df = pd.read_csv('/Users/mallorygo/Desktop/DATA1030/data1030-fall2025/final reports/ufc-master.csv')
pd.set_option("display.max_columns", None, "display.max_rows", None)

print(df.head())
```

| | RedFighter | BlueFighter | RedOdds | BlueOdds | RedExpectedValue | BlueExpectedValue | Date | Location | Country | Winner | TitleBout | WeightClass | Gender | NumberOfRounds | BlueCurrentLoseStreak | BlueCurrentWinStreak | BlueDraws | BlueAvgSigStrLanded | BlueAvgSigStrPct | BlueAvgSubAtt | BlueAvgTDLanded | BlueAvgTDPct | BlueLongestWinStreak | BlueLosses | BlueTotalRoundsFought | BlueTotalTitleBouts |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Alexandre Pantoja | Kai Asakura | -250.0 | 215.0 | 40.0000 | 215.0 | 2024-12-07 | Las Vegas, Nevada, USA | USA | Red | True | Flyweight | MALE | 5 | 0 | 0 | 0 | 0.00 | 0.00 | 0.0 | 0.00 | 0.00 | 0 | 0 | 0 | 0 |
| 1 | Shavkat Rakhmonov | Ian Machado Garry | -210.0 | 295.0 | 47.6190 | 295.0 | 2024-12-07 | Las Vegas, Nevada, USA | USA | Red | False | Welterweight | MALE | 3 | 0 | 8 | 0 | 5.50 | 0.55 | 0.3 | 0.77 | 0.55 | 8 | 0 | 20 | 0 |
| 2 | Ciryl Gane | Alexander Volkov | -380.0 | 300.0 | 26.3158 | 300.0 | 2024-12-07 | Las Vegas, Nevada, USA | USA | Red | False | Heavyweight | MALE | 3 | 0 | 4 | 0 | 5.13 | 0.57 | 0.2 | 0.45 | 0.63 | 4 | 4 | 44 | 0 |
| 3 | Bryce Mitchell | Kron Gracie | -950.0 | 625.0 | 10.5263 | 625.0 | 2024-12-07 | Las Vegas, Nevada, USA | USA | Red | False | Featherweight | MALE | 3 | 2 | 0 | 0 | 3.74 | 0.44 | 0.5 | 0.47 | 0.25 | 1 | 2 | 7 | 0 |
| 4 | Nate Landwehr | Dooho Choi | -130.0 | 110.0 | 76.9231 | 110.0 | 2024-12-07 | Las Vegas, Nevada, USA | USA | Blue | False | Featherweight | MALE | 3 | 0 | 1 | 1 | 4.41 | 0.53 | 0.8 | 0.75 | 0.37 | 3 | 3 | 15 | 0 |

| | BlueWinsByDecisionMajority | BlueWinsByDecisionSplit | \ |
|---|---|---|---|
| 0 | 0 | 0 | |
| 1 | 0 | 1 | |
| 2 | 0 | 1 | |
| 3 | 0 | 0 | |
| 4 | 0 | 0 | |

| | BlueWinsByDecisionUnanimous | BlueWinsByKO | BlueWinsBySubmission | \ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | |
| 1 | 4 | 3 | 0 | |
| 2 | 4 | 6 | 1 | |
| 3 | 0 | 0 | 1 | |
| 4 | 0 | 4 | 0 | |

| | BlueWinsByTKODoctorStoppage | BlueWins | BlueStance | BlueHeightCms | \ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | Orthodox | 172.72 | |
| 1 | 0 | 8 | Orthodox | 190.50 | |
| 2 | 0 | 12 | Orthodox | 200.66 | |
| 3 | 0 | 1 | Southpaw | 175.26 | |
| 4 | 0 | 4 | Orthodox | 177.80 | |

| | BlueReachCms | BlueWeightLbs | RedCurrentLoseStreak | RedCurrentWinStreak | \ |
|---|---|---|---|---|---|
| 0 | 175.26 | 125 | 0 | 6 | |
| 1 | 187.96 | 170 | 0 | 6 | |
| 2 | 203.20 | 250 | 0 | 1 | |
| 3 | 177.80 | 145 | 1 | 0 | |
| 4 | 177.80 | 145 | 0 | 1 | |

| | RedDraws | RedAvgSigStrLanded | RedAvgSigStrPct | RedAvgSubAtt | \ |
|---|---|---|---|---|---|
| 0 | 0 | 4.41 | 0.49 | 0.8 | |
| 1 | 0 | 4.12 | 0.61 | 1.8 | |
| 2 | 0 | 5.49 | 0.60 | 0.5 | |
| 3 | 0 | 2.30 | 0.58 | 1.6 | |
| 4 | 0 | 6.25 | 0.46 | 1.0 | |

| | RedAvgTDLanded | RedAvgTDPct | RedLongestWinStreak | RedLosses | \ |
|---|---|---|---|---|---|
| 0 | 2.61 | 0.47 | 6 | 3 | |
| 1 | 1.49 | 0.29 | 6 | 0 | |
| 2 | 0.58 | 0.21 | 7 | 2 | |
| 3 | 3.45 | 0.41 | 6 | 2 | |
| 4 | 1.00 | 0.41 | 3 | 3 | |

| | RedTotalRoundsFought | RedTotalTitleBouts | RedWinsByDecisionMajority | RedWinsByDecisionSplit | RedWinsByDecisionUnanimous | RedWinsByKO | RedWinsBySubmission | RedWinsByTKODoctorStoppage | RedWins | RedStance | RedHeightCms | RedReachCms | RedWeightLbs | RedAge | BlueAge | LoseStreakDif | WinStreakDif | LongestWinStreakDif | WinDif | LossDif | TotalRoundDif | TotalTitleBoutDif | KODif | SubDif | HeightDif | ReachDif | AgeDif | SigStrDif |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 42 | 3 | 0 | 2 | 4 | 2 | 4 | 0 | 12 | Orthodox | 165.10 | 170.18 | 125 | 34 | 31 | 0 | -6 | -6 | -12 | -3 | -42 | -3 | -2 | -4 | 7.62 | 5.08 | -3 | -4.41 |
| 1 | 11 | 0 | 0 | 0 | 0 | 1 | 5 | 0 | 6 | Orthodox | 185.42 | 195.58 | 170 | 30 | 27 | 0 | 2 | 2 | 2 | 0 | 9 | 0 | 2 | -5 | 5.08 | -7.62 | -3 | 1.38 |
| 2 | 33 | 3 | 0 | 0 | 3 | 4 | 2 | 0 | 9 | Orthodox | 193.04 | 205.74 | 245 | 34 | 36 | 0 | 3 | -3 | 3 | 2 | 11 | -3 | 2 | -1 | 7.62 | -2.54 | 2 | -0.36 |
| 3 | 22 | 0 | 1 | 0 | 5 | 0 | 1 | 0 | 7 | Southpaw | 177.80 | 177.80 | 145 | 30 | 36 | 1 | 0 | -5 | -6 | 0 | -15 | 0 | 0 | 0 | -2.54 | 0.00 | 6 | 1.44 |
| 4 | 17 | 0 | 1 | 0 | 1 | 1 | 2 | 0 | 5 | Orthodox | 175.26 | 182.88 | 145 | 36 | 33 | 0 | 0 | 0 | -1 | 0 | -2 | 0 | 3 | -2 | 2.54 | -5.08 | -3 | -1.84 |

| | AvgSubAttDif | AvgTDDif | EmptyArena | BMatchWCRank | RMatchWCRank |
|---|---|---|---|---|---|
| 0 | -0.8 | -2.61 | NaN | NaN | 0.0 |
| 1 | -1.5 | -0.72 | NaN | 7.0 | 3.0 |
| 2 | -0.3 | -0.13 | NaN | 3.0 | 2.0 |
| 3 | -1.1 | -2.98 | NaN | NaN | 13.0 |
| 4 | -0.2 | -0.25 | NaN | NaN | NaN |

| | RWFlyweightRank | RWFeatherweightRank | RWStrawweightRank |
|---|---|---|---|
| 0 | NaN | NaN | NaN |
| 1 | NaN | NaN | NaN |
| 2 | NaN | NaN | NaN |
| 3 | NaN | NaN | NaN |
| 4 | NaN | NaN | NaN |

| | RWBantamweightRank | RHeavyweightRank | RLightHeavyweightRank |
|---|---|---|---|
| 0 | NaN | NaN | NaN |
| 1 | NaN | NaN | NaN |
| 2 | NaN | 2.0 | NaN |
| 3 | NaN | NaN | NaN |
| 4 | NaN | NaN | NaN |

| | RMiddleweightRank | RWelterweightRank | RLightweightRank | RFeatherweightRank |
|---|---|---|---|---|
| 0 | NaN | NaN | NaN | NaN |
| 1 | NaN | 3.0 | NaN | NaN |
| 2 | NaN | NaN | NaN | NaN |
| 3 | NaN | NaN | NaN | 13.0 |
| 4 | NaN | NaN | NaN | NaN |

| | RBantamweightRank | RFlyweightRank | RPFPRank | BWFlyweightRank |
|---|---|---|---|---|
| 0 | NaN | 0.0 | 11.0 | NaN |
| 1 | NaN | NaN | NaN | NaN |
| 2 | NaN | NaN | NaN | NaN |
| 3 | NaN | NaN | NaN | NaN |
| 4 | NaN | NaN | NaN | NaN |

| | BWFeatherweightRank | BWStrawweightRank | BWBantamweightRank |
|---|---|---|---|
| 0 | NaN | NaN | NaN |
| 1 | NaN | NaN | NaN |
| 2 | NaN | NaN | NaN |
| 3 | NaN | NaN | NaN |
| 4 | NaN | NaN | NaN |

```
   BHeavyweightRank  BLightHeavyweightRank  BMiddleweightRank  \
0               NaN                    NaN                NaN
1               NaN                    NaN                NaN
2               3.0                    NaN                NaN
3               NaN                    NaN                NaN
4               NaN                    NaN                NaN

   BWelterweightRank  BLightweightRank  BFeatherweightRank  BBantamweightRank  \
0                NaN               NaN                 NaN                NaN
1                7.0               NaN                 NaN                NaN
2                NaN               NaN                 NaN                NaN
3                NaN               NaN                 NaN                NaN
4                NaN               NaN                 NaN                NaN

   BFlyweightRank  BPFPRank BetterRank  Finish     FinishDetails  FinishRound  \
0             NaN       NaN        Red     SUB  Rear Naked Choke          2.0
1             NaN       NaN        Red   U-DEC               NaN          5.0
2             NaN       NaN        Red   S-DEC               NaN          3.0
3             NaN       NaN        Red  KO/TKO            Elbows          3.0
4             NaN       NaN    neither  KO/TKO            Elbows          3.0

  FinishRoundTime  TotalFightTimeSecs  RedDecOdds  BlueDecOdds  RSubOdds  \
0            2:05                425.0       300.0        800.0     150.0
1            5:00               1500.0       250.0        650.0     180.0
2            5:00                900.0      -160.0        450.0    1100.0
3            0:39                639.0      -200.0       1100.0     380.0
4            3:21                801.0       275.0        550.0     500.0

   BSubOdds  RKOOdds  BKOOdds
0    2500.0    400.0    350.0
1    3000.0    240.0    700.0
2    3000.0    350.0   1100.0
3    1400.0    500.0   4000.0
4     700.0    300.0    250.0
```

In [240…
```
import os
output_folder = "/Users/mallorygo/Desktop/DATA1030/data1030-fall2025/data1030_final/figurefolder"
#fig_path = os.path.join(output_folder, f"{fig_name}.png")
```

In [241…
```
pd.set_option("display.max_columns", None, "display.max_rows", None)

print(df.dtypes)
```

```
# How many rows and columns do we have in df_merge?
print(df.shape[0]) # number of rows
print(df.shape[1]) # number of rows

print(len(df))
```

```
RedFighter                    object
BlueFighter                   object
RedOdds                      float64
BlueOdds                     float64
RedExpectedValue             float64
BlueExpectedValue            float64
Date                          object
Location                      object
Country                       object
Winner                        object
TitleBout                       bool
WeightClass                   object
Gender                        object
NumberOfRounds                 int64
BlueCurrentLoseStreak          int64
BlueCurrentWinStreak           int64
BlueDraws                      int64
BlueAvgSigStrLanded          float64
BlueAvgSigStrPct             float64
BlueAvgSubAtt                float64
BlueAvgTDLanded              float64
BlueAvgTDPct                 float64
BlueLongestWinStreak           int64
BlueLosses                     int64
BlueTotalRoundsFought          int64
BlueTotalTitleBouts            int64
BlueWinsByDecisionMajority     int64
BlueWinsByDecisionSplit        int64
BlueWinsByDecisionUnanimous    int64
BlueWinsByKO                   int64
BlueWinsBySubmission           int64
BlueWinsByTKODoctorStoppage    int64
BlueWins                       int64
BlueStance                    object
BlueHeightCms                float64
BlueReachCms                 float64
BlueWeightLbs                  int64
RedCurrentLoseStreak           int64
RedCurrentWinStreak            int64
RedDraws                       int64
RedAvgSigStrLanded           float64
RedAvgSigStrPct              float64
```

```
RedAvgSubAtt                 float64
RedAvgTDLanded               float64
RedAvgTDPct                  float64
RedLongestWinStreak            int64
RedLosses                      int64
RedTotalRoundsFought           int64
RedTotalTitleBouts             int64
RedWinsByDecisionMajority      int64
RedWinsByDecisionSplit         int64
RedWinsByDecisionUnanimous     int64
RedWinsByKO                    int64
RedWinsBySubmission            int64
RedWinsByTKODoctorStoppage     int64
RedWins                        int64
RedStance                     object
RedHeightCms                 float64
RedReachCms                  float64
RedWeightLbs                   int64
RedAge                         int64
BlueAge                        int64
LoseStreakDif                  int64
WinStreakDif                   int64
LongestWinStreakDif            int64
WinDif                         int64
LossDif                        int64
TotalRoundDif                  int64
TotalTitleBoutDif              int64
KODif                          int64
SubDif                         int64
HeightDif                    float64
ReachDif                     float64
AgeDif                         int64
SigStrDif                    float64
AvgSubAttDif                 float64
AvgTDDif                     float64
EmptyArena                   float64
BMatchWCRank                 float64
RMatchWCRank                 float64
RWFlyweightRank              float64
RWFeatherweightRank          float64
RWStrawweightRank            float64
RWBantamweightRank           float64
```

```
RHeavyweightRank              float64
RLightHeavyweightRank         float64
RMiddleweightRank             float64
RWelterweightRank             float64
RLightweightRank              float64
RFeatherweightRank            float64
RBantamweightRank             float64
RFlyweightRank                float64
RPFPRank                      float64
BWFlyweightRank               float64
BWFeatherweightRank           float64
BWStrawweightRank             float64
BWBantamweightRank            float64
BHeavyweightRank              float64
BLightHeavyweightRank         float64
BMiddleweightRank             float64
BWelterweightRank             float64
BLightweightRank              float64
BFeatherweightRank            float64
BBantamweightRank             float64
BFlyweightRank                float64
BPFPRank                      float64
BetterRank                     object
Finish                         object
FinishDetails                  object
FinishRound                   float64
FinishRoundTime                object
TotalFightTimeSecs            float64
RedDecOdds                    float64
BlueDecOdds                   float64
RSubOdds                      float64
BSubOdds                      float64
RKOOdds                       float64
BKOOdds                       float64
dtype: object
6528
118
6528
```

In [242…

```
comm_drop = [
'Date','Location', 'Country'
]
```

```
df.drop(comm_drop, axis=1, inplace = True)


print(df.dtypes)
print(df.shape[0]) # number of rows
print(df.shape[1]) # number of rows

print(len(df))
```

```
RedFighter                    object
BlueFighter                   object
RedOdds                      float64
BlueOdds                     float64
RedExpectedValue             float64
BlueExpectedValue            float64
Winner                        object
TitleBout                       bool
WeightClass                   object
Gender                        object
NumberOfRounds                 int64
BlueCurrentLoseStreak          int64
BlueCurrentWinStreak           int64
BlueDraws                      int64
BlueAvgSigStrLanded          float64
BlueAvgSigStrPct             float64
BlueAvgSubAtt                float64
BlueAvgTDLanded              float64
BlueAvgTDPct                 float64
BlueLongestWinStreak           int64
BlueLosses                     int64
BlueTotalRoundsFought          int64
BlueTotalTitleBouts            int64
BlueWinsByDecisionMajority     int64
BlueWinsByDecisionSplit        int64
BlueWinsByDecisionUnanimous    int64
BlueWinsByKO                   int64
BlueWinsBySubmission           int64
BlueWinsByTKODoctorStoppage    int64
BlueWins                       int64
BlueStance                    object
BlueHeightCms                float64
BlueReachCms                 float64
BlueWeightLbs                  int64
RedCurrentLoseStreak           int64
RedCurrentWinStreak            int64
RedDraws                       int64
RedAvgSigStrLanded           float64
RedAvgSigStrPct              float64
RedAvgSubAtt                 float64
RedAvgTDLanded               float64
RedAvgTDPct                  float64
```

```
RedLongestWinStreak             int64
RedLosses                       int64
RedTotalRoundsFought            int64
RedTotalTitleBouts              int64
RedWinsByDecisionMajority       int64
RedWinsByDecisionSplit          int64
RedWinsByDecisionUnanimous      int64
RedWinsByKO                     int64
RedWinsBySubmission             int64
RedWinsByTKODoctorStoppage      int64
RedWins                         int64
RedStance                      object
RedHeightCms                  float64
RedReachCms                   float64
RedWeightLbs                    int64
RedAge                          int64
BlueAge                         int64
LoseStreakDif                   int64
WinStreakDif                    int64
LongestWinStreakDif             int64
WinDif                          int64
LossDif                         int64
TotalRoundDif                   int64
TotalTitleBoutDif               int64
KODif                           int64
SubDif                          int64
HeightDif                     float64
ReachDif                      float64
AgeDif                          int64
SigStrDif                     float64
AvgSubAttDif                  float64
AvgTDDif                      float64
EmptyArena                    float64
BMatchWCRank                  float64
RMatchWCRank                  float64
RWFlyweightRank               float64
RWFeatherweightRank           float64
RWStrawweightRank             float64
RWBantamweightRank            float64
RHeavyweightRank              float64
RLightHeavyweightRank         float64
RMiddleweightRank             float64
```

```
RWelterweightRank            float64
RLightweightRank             float64
RFeatherweightRank           float64
RBantamweightRank            float64
RFlyweightRank               float64
RPFPRank                     float64
BWFlyweightRank              float64
BWFeatherweightRank          float64
BWStrawweightRank            float64
BWBantamweightRank           float64
BHeavyweightRank             float64
BLightHeavyweightRank        float64
BMiddleweightRank            float64
BWelterweightRank            float64
BLightweightRank             float64
BFeatherweightRank           float64
BBantamweightRank            float64
BFlyweightRank               float64
BPFPRank                     float64
BetterRank                    object
Finish                        object
FinishDetails                 object
FinishRound                  float64
FinishRoundTime               object
TotalFightTimeSecs           float64
RedDecOdds                   float64
BlueDecOdds                  float64
RSubOdds                     float64
BSubOdds                     float64
RKOOdds                      float64
BKOOdds                      float64
dtype: object
6528
115
6528
```

In [243…

```
#Separating the features based on their data types
cat_col = [col for col in df.columns if df[col].dtypes == 'object']
num_col = [col for col in df.columns if col not in cat_col]
print(df['Winner'].value_counts(dropna=False))
```

```
Winner
Red     3787
Blue    2741
Name: count, dtype: int64
```

In [244…
```
#take care of missingness in Winner
print("Unique values in Winner column:", df['Winner'].unique())
# Strip spaces and lowercase all values
df['Winner'] = df['Winner'].astype(str).str.strip().str.capitalize()
print("Cleaned Winner values:", df['Winner'].unique())
df = df[df['Winner'].isin(['Red', 'Blue'])]
print("Rows remaining after filtering:", len(df))
```

```
Unique values in Winner column: ['Red' 'Blue']
Cleaned Winner values: ['Red' 'Blue']
Rows remaining after filtering: 6528
```

In [245…
```
unique_categorieswc = df['WeightClass'].unique()
print(unique_categorieswc)
```

```
['Flyweight' 'Welterweight' 'Heavyweight' 'Featherweight'
 'Light Heavyweight' 'Catch Weight' 'Lightweight' 'Bantamweight'
 "Women's Strawweight" "Women's Flyweight" 'Middleweight'
 "Women's Bantamweight" "Women's Featherweight"]
```

In [246…
```
from sklearn.preprocessing import LabelEncoder
print(df['Winner'].value_counts(dropna=False))
# Convert 'Red' and 'Blue' winner string to binary label (1 = Red wins, 0 = Blue wins)
df = df[df['Winner'].isin(['Red', 'Blue'])]  # filter out 'Draw' or 'No Contest'
print("Rows remaining after filtering:", len(df))

cat_col = ['RedFighter', 'BlueFighter', 'WeightClass', 'Gender']

from sklearn.preprocessing import LabelEncoder

enc = LabelEncoder()

for col in cat_col:
    if col in df.columns:
        df[col] = enc.fit_transform(df[col].astype(str))
    else:
        print(f"Warning: Column {col} not found in dataframe.")
```

```
print(df['Winner'].value_counts())
#Series([], Name: count, dtype: int64)

# result =
# Series([], Name: count, dtype: int64)
# Rows remaining after filtering: 0
# Series([], Name: count, dtype: int64)
```

```
Winner
Red     3787
Blue    2741
Name: count, dtype: int64
Rows remaining after filtering: 6528
Winner
Red     3787
Blue    2741
Name: count, dtype: int64
```

In [247…

```
win_counts = df['Winner'].value_counts()
# Display the frequency table
print("Frequency Table for 'Winner' column:")
print(win_counts)
```

```
Frequency Table for 'Winner' column:
Winner
Red     3787
Blue    2741
Name: count, dtype: int64
```

In [248…

```
for column in df.columns:
    if df[column].isnull().sum()!=0:
        print(f"Nan in {column}: {df[column].isnull().sum()}")
```

```
Nan in RedOdds: 227
Nan in BlueOdds: 226
Nan in RedExpectedValue: 227
Nan in BlueExpectedValue: 226
Nan in BlueAvgSigStrLanded: 930
Nan in BlueAvgSigStrPct: 765
Nan in BlueAvgSubAtt: 832
Nan in BlueAvgTDLanded: 833
Nan in BlueAvgTDPct: 842
Nan in BlueStance: 3
Nan in RedAvgSigStrLanded: 455
Nan in RedAvgSigStrPct: 357
Nan in RedAvgSubAtt: 357
Nan in RedAvgTDLanded: 357
Nan in RedAvgTDPct: 367
Nan in EmptyArena: 1486
Nan in BMatchWCRank: 5328
Nan in RMatchWCRank: 4749
Nan in RWFlyweightRank: 6432
Nan in RWFeatherweightRank: 6519
Nan in RWStrawweightRank: 6382
Nan in RWBantamweightRank: 6374
Nan in RHeavyweightRank: 6342
Nan in RLightHeavyweightRank: 6344
Nan in RMiddleweightRank: 6346
Nan in RWelterweightRank: 6337
Nan in RLightweightRank: 6344
Nan in RFeatherweightRank: 6351
Nan in RBantamweightRank: 6347
Nan in RFlyweightRank: 6340
Nan in RPFPRank: 6275
Nan in BWFlyweightRank: 6455
Nan in BWFeatherweightRank: 6527
Nan in BWStrawweightRank: 6428
Nan in BWBantamweightRank: 6421
Nan in BHeavyweightRank: 6380
Nan in BLightHeavyweightRank: 6408
Nan in BMiddleweightRank: 6391
Nan in BWelterweightRank: 6409
Nan in BLightweightRank: 6408
Nan in BFeatherweightRank: 6404
Nan in BBantamweightRank: 6409
```

```
Nan in BFlyweightRank: 6398
Nan in BPFPRank: 6461
Nan in Finish: 238
Nan in FinishDetails: 3636
Nan in FinishRound: 622
Nan in FinishRoundTime: 622
Nan in TotalFightTimeSecs: 622
Nan in RedDecOdds: 1087
Nan in BlueDecOdds: 1116
Nan in RSubOdds: 1336
Nan in BSubOdds: 1359
Nan in RKOOdds: 1334
Nan in BKOOdds: 1360
```

In [249…

```python
#force numeric WeightClass
df['WeightClass'] = pd.to_numeric(df['WeightClass'], errors='coerce')

weightclass_labels = {
    0: 'Flyweight',
    1: 'Welterweight',
    2: 'Heavyweight',
    3: 'Featherweight',
    4: 'Light Heavyweight',
    5: 'Catch Weight',
    6: 'Lightweight',
    7: 'Bantamweight',
    8: "Women's Strawweight",
    9: "Women's Flyweight",
    10: "Middleweight",
    11: "Women's Bantamweight",
    12: "Women's Featherweight "
}



df['WeightClassLabel'] = df['WeightClass'].map(weightclass_labels)

#check
missing_wtclass = df['WeightClass'].isnull().sum()
print(f"✅ WeightClass mapped. Missing values: {missing_wtclass}")
```

```
✅ WeightClass mapped. Missing values: 0
```

In [250…

```
# Basic descriptive statistics
print(df.describe())

# Check class balance
sns.countplot(x='Winner', data=df)
plt.title('Fight Winner Distribution (Red = 1, Blue = 0)')
fig_name = "fight_winner_dist"
fig_path = os.path.join(output_folder, f"{fig_name}.png")
plt.savefig(fig_path, bbox_inches="tight", dpi=300)
plt.show()
plt.close()

# Missing data heatmap
sns.heatmap(df.isnull(), cbar=False)
plt.title("Missing Data Heatmap")
fig_name = "missing_data_heatmap1"
fig_path = os.path.join(output_folder, f"{fig_name}.png")
plt.savefig(fig_path, bbox_inches="tight", dpi=300)
plt.show()
plt.close()
```

| | RedFighter | BlueFighter | RedOdds | BlueOdds | RedExpectedValue |
|---|---|---|---|---|---|
| count | 6528.000000 | 6528.000000 | 6301.000000 | 6302.000000 | 6301.000000 |
| mean | 819.761795 | 942.707874 | -115.711474 | 59.793240 | 96.658224 |
| std | 476.188466 | 556.368724 | 277.225783 | 253.117416 | 85.891109 |
| min | 0.000000 | 0.000000 | -2100.000000 | -1200.000000 | 4.761900 |
| 25% | 419.000000 | 462.000000 | -255.000000 | -150.000000 | 39.215700 |
| 50% | 806.000000 | 922.500000 | -150.000000 | 130.000000 | 66.666700 |
| 75% | 1241.250000 | 1427.000000 | 130.000000 | 215.000000 | 130.000000 |
| max | 1660.000000 | 1921.000000 | 775.000000 | 1300.000000 | 775.000000 |

| | BlueExpectedValue | WeightClass | Gender | NumberOfRounds |
|---|---|---|---|---|
| count | 6302.000000 | 6528.000000 | 6528.000000 | 6528.000000 |
| mean | 165.054566 | 5.503370 | 0.877298 | 3.185509 |
| std | 137.689177 | 3.200274 | 0.328120 | 0.577441 |
| min | 8.333300 | 0.000000 | 0.000000 | 3.000000 |
| 25% | 66.666700 | 3.000000 | 1.000000 | 3.000000 |
| 50% | 130.000000 | 6.000000 | 1.000000 | 3.000000 |
| 75% | 215.000000 | 8.000000 | 1.000000 | 3.000000 |
| max | 1300.000000 | 12.000000 | 1.000000 | 5.000000 |

| | BlueCurrentLoseStreak | BlueCurrentWinStreak | BlueDraws |
|---|---|---|---|
| count | 6528.000000 | 6528.000000 | 6528.000000 |
| mean | 0.501072 | 0.957567 | 0.023131 |
| std | 0.794303 | 1.406786 | 0.156327 |
| min | 0.000000 | 0.000000 | 0.000000 |
| 25% | 0.000000 | 0.000000 | 0.000000 |
| 50% | 0.000000 | 0.000000 | 0.000000 |
| 75% | 1.000000 | 1.000000 | 0.000000 |
| max | 6.000000 | 12.000000 | 2.000000 |

| | BlueAvgSigStrLanded | BlueAvgSigStrPct | BlueAvgSubAtt | BlueAvgTDLanded |
|---|---|---|---|---|
| count | 5598.000000 | 5763.000000 | 5696.000000 | 5695.000000 |
| mean | 19.841810 | 0.453059 | 0.500202 | 1.320536 |
| std | 20.315307 | 0.110787 | 0.672859 | 1.356491 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 3.880000 | 0.400000 | 0.000000 | 0.330000 |
| 50% | 9.280000 | 0.460000 | 0.300000 | 1.000000 |
| 75% | 32.666700 | 0.513000 | 0.800000 | 1.970000 |
| max | 154.000000 | 1.000000 | 8.400000 | 10.860000 |

| | BlueAvgTDPct | BlueLongestWinStreak | BlueLosses | BlueTotalRoundsFought |
|---|---|---|---|---|
| count | 5686.000000 | 6528.000000 | 6528.000000 | 6528.000000 |

| | | | | |
|---|---|---|---|---|
| mean | 0.325419 | 1.923407 | 1.863664 | 11.872396 |
| std | 0.239174 | 1.949952 | 2.170130 | 13.845139 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 0.150000 | 0.000000 | 0.000000 | 3.000000 |
| 50% | 0.330000 | 1.000000 | 1.000000 | 7.000000 |
| 75% | 0.470000 | 3.000000 | 3.000000 | 17.000000 |
| max | 1.000000 | 17.000000 | 16.000000 | 111.000000 |

| | BlueTotalTitleBouts | BlueWinsByDecisionMajority | \ |
|---|---|---|---|
| count | 6528.000000 | 6528.000000 | |
| mean | 0.251685 | 0.017770 | |
| std | 1.085122 | 0.133278 | |
| min | 0.000000 | 0.000000 | |
| 25% | 0.000000 | 0.000000 | |
| 50% | 0.000000 | 0.000000 | |
| 75% | 0.000000 | 0.000000 | |
| max | 16.000000 | 2.000000 | |

| | BlueWinsByDecisionSplit | BlueWinsByDecisionUnanimous | BlueWinsByKO | \ |
|---|---|---|---|---|
| count | 6528.000000 | 6528.000000 | 6528.000000 | |
| mean | 0.279871 | 1.093597 | 1.066330 | |
| std | 0.595123 | 1.613288 | 1.723951 | |
| min | 0.000000 | 0.000000 | 0.000000 | |
| 25% | 0.000000 | 0.000000 | 0.000000 | |
| 50% | 0.000000 | 0.000000 | 0.000000 | |
| 75% | 0.000000 | 2.000000 | 1.250000 | |
| max | 5.000000 | 11.000000 | 20.000000 | |

| | BlueWinsBySubmission | BlueWinsByTKODoctorStoppage | BlueWins | \ |
|---|---|---|---|---|
| count | 6528.000000 | 6528.000000 | 6528.000000 | |
| mean | 0.626532 | 0.022212 | 3.145680 | |
| std | 1.258249 | 0.153494 | 3.712852 | |
| min | 0.000000 | 0.000000 | 0.000000 | |
| 25% | 0.000000 | 0.000000 | 0.000000 | |
| 50% | 0.000000 | 0.000000 | 2.000000 | |
| 75% | 1.000000 | 0.000000 | 5.000000 | |
| max | 13.000000 | 2.000000 | 31.000000 | |

| | BlueHeightCms | BlueReachCms | BlueWeightLbs | RedCurrentLoseStreak | \ |
|---|---|---|---|---|---|
| count | 6528.000000 | 6528.000000 | 6528.000000 | 6528.000000 | |
| mean | 177.822068 | 182.162155 | 163.183977 | 0.622243 | |
| std | 9.105099 | 11.168891 | 34.599386 | 0.872301 | |

| | | | | |
|---|---|---|---|---|
| min | 152.400000 | 0.000000 | 115.000000 | 0.000000 |
| 25% | 170.180000 | 175.260000 | 135.000000 | 0.000000 |
| 50% | 177.800000 | 182.880000 | 155.000000 | 0.000000 |
| 75% | 185.420000 | 190.500000 | 185.000000 | 1.000000 |
| max | 210.820000 | 213.360000 | 265.000000 | 7.000000 |

| | RedCurrentWinStreak | RedDraws | RedAvgSigStrLanded | RedAvgSigStrPct |
|---|---|---|---|---|
| count | 6528.000000 | 6528.000000 | 6073.000000 | 6171.000000 |
| mean | 1.101562 | 0.031097 | 21.152766 | 0.460321 |
| std | 1.760767 | 0.187999 | 19.882916 | 0.098315 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 0.000000 | 0.000000 | 4.160000 | 0.403000 |
| 50% | 0.000000 | 0.000000 | 15.666700 | 0.460000 |
| 75% | 2.000000 | 0.000000 | 34.000000 | 0.516000 |
| max | 18.000000 | 2.000000 | 141.000000 | 1.000000 |

| | RedAvgSubAtt | RedAvgTDLanded | RedAvgTDPct | RedLongestWinStreak |
|---|---|---|---|---|
| count | 6171.000000 | 6171.000000 | 6161.000000 | 6528.000000 |
| mean | 0.536907 | 1.399962 | 0.341467 | 2.679994 |
| std | 0.693125 | 1.308340 | 0.220986 | 2.242687 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 0.000000 | 0.450000 | 0.200000 | 1.000000 |
| 50% | 0.333300 | 1.030000 | 0.338000 | 2.000000 |
| 75% | 0.800000 | 2.000000 | 0.470000 | 4.000000 |
| max | 8.400000 | 12.500000 | 1.000000 | 18.000000 |

| | RedLosses | RedTotalRoundsFought | RedTotalTitleBouts |
|---|---|---|---|
| count | 6528.000000 | 6528.000000 | 6528.000000 |
| mean | 2.566789 | 17.408548 | 0.553462 |
| std | 2.703546 | 17.846748 | 1.544962 |
| min | 0.000000 | 0.000000 | 0.000000 |
| 25% | 1.000000 | 5.000000 | 0.000000 |
| 50% | 2.000000 | 12.000000 | 0.000000 |
| 75% | 4.000000 | 25.000000 | 0.000000 |
| max | 21.000000 | 448.000000 | 16.000000 |

| | RedWinsByDecisionMajority | RedWinsByDecisionSplit |
|---|---|---|
| count | 6528.000000 | 6528.000000 |
| mean | 0.026961 | 0.400888 |
| std | 0.163862 | 0.702801 |
| min | 0.000000 | 0.000000 |
| 25% | 0.000000 | 0.000000 |

| | | |
|---|---|---|
| 50% | 0.000000 | 0.000000 |
| 75% | 0.000000 | 1.000000 |
| max | 2.000000 | 5.000000 |

| | RedWinsByDecisionUnanimous | RedWinsByKO | RedWinsBySubmission |
|---|---|---|---|
| count | 6528.000000 | 6528.000000 | 6528.000000 |
| mean | 1.612286 | 1.563879 | 0.934589 |
| std | 1.972295 | 2.156770 | 1.608402 |
| min | 0.000000 | 0.000000 | 0.000000 |
| 25% | 0.000000 | 0.000000 | 0.000000 |
| 50% | 1.000000 | 1.000000 | 0.000000 |
| 75% | 2.000000 | 2.000000 | 1.000000 |
| max | 11.000000 | 21.000000 | 16.000000 |

| | RedWinsByTKODoctorStoppage | RedWins | RedHeightCms | RedReachCms |
|---|---|---|---|---|
| count | 6528.000000 | 6528.000000 | 6528.000000 | 6528.000000 |
| mean | 0.035539 | 4.628064 | 177.802399 | 182.410954 |
| std | 0.199492 | 4.505498 | 9.182963 | 11.130395 |
| min | 0.000000 | 0.000000 | 152.400000 | 147.320000 |
| 25% | 0.000000 | 1.000000 | 170.180000 | 175.260000 |
| 50% | 0.000000 | 3.000000 | 177.800000 | 182.880000 |
| 75% | 0.000000 | 7.000000 | 185.420000 | 190.500000 |
| max | 2.000000 | 33.000000 | 210.820000 | 214.630000 |

| | RedWeightLbs | RedAge | BlueAge | LoseStreakDif | WinStreakDif |
|---|---|---|---|---|---|
| count | 6528.000000 | 6528.000000 | 6528.000000 | 6528.000000 | 6528.000000 |
| mean | 163.621324 | 30.359528 | 29.805607 | 0.059283 | -0.143842 |
| std | 34.846543 | 4.180712 | 3.959623 | 1.024000 | 1.874732 |
| min | 115.000000 | 18.000000 | 19.000000 | -6.000000 | -18.000000 |
| 25% | 135.000000 | 27.000000 | 27.000000 | 0.000000 | -1.000000 |
| 50% | 155.000000 | 30.000000 | 30.000000 | 0.000000 | 0.000000 |
| 75% | 185.000000 | 33.000000 | 32.000000 | 0.000000 | 0.000000 |
| max | 265.000000 | 47.000000 | 47.000000 | 6.000000 | 10.000000 |

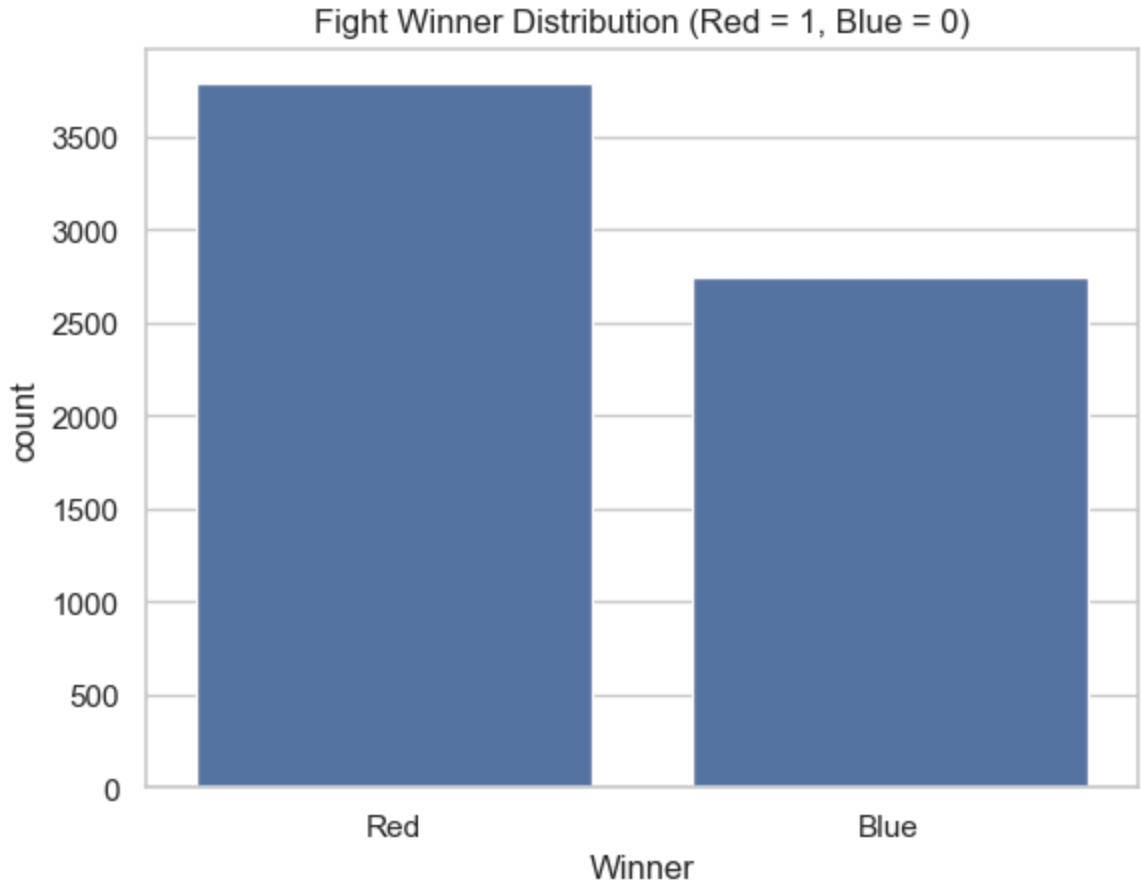
| | LongestWinStreakDif | WinDif | LossDif | TotalRoundDif |
|---|---|---|---|---|
| count | 6528.000000 | 6528.000000 | 6528.000000 | 6528.000000 |
| mean | -0.756587 | -1.482384 | 0.075061 | -5.536152 |
| std | 2.025886 | 4.182192 | 3.129140 | 17.995648 |
| min | -12.000000 | -28.000000 | -20.000000 | -448.000000 |
| 25% | -2.000000 | -3.000000 | -1.000000 | -12.000000 |
| 50% | -1.000000 | -1.000000 | 0.000000 | -3.000000 |
| 75% | 0.000000 | 0.000000 | 2.000000 | 2.000000 |

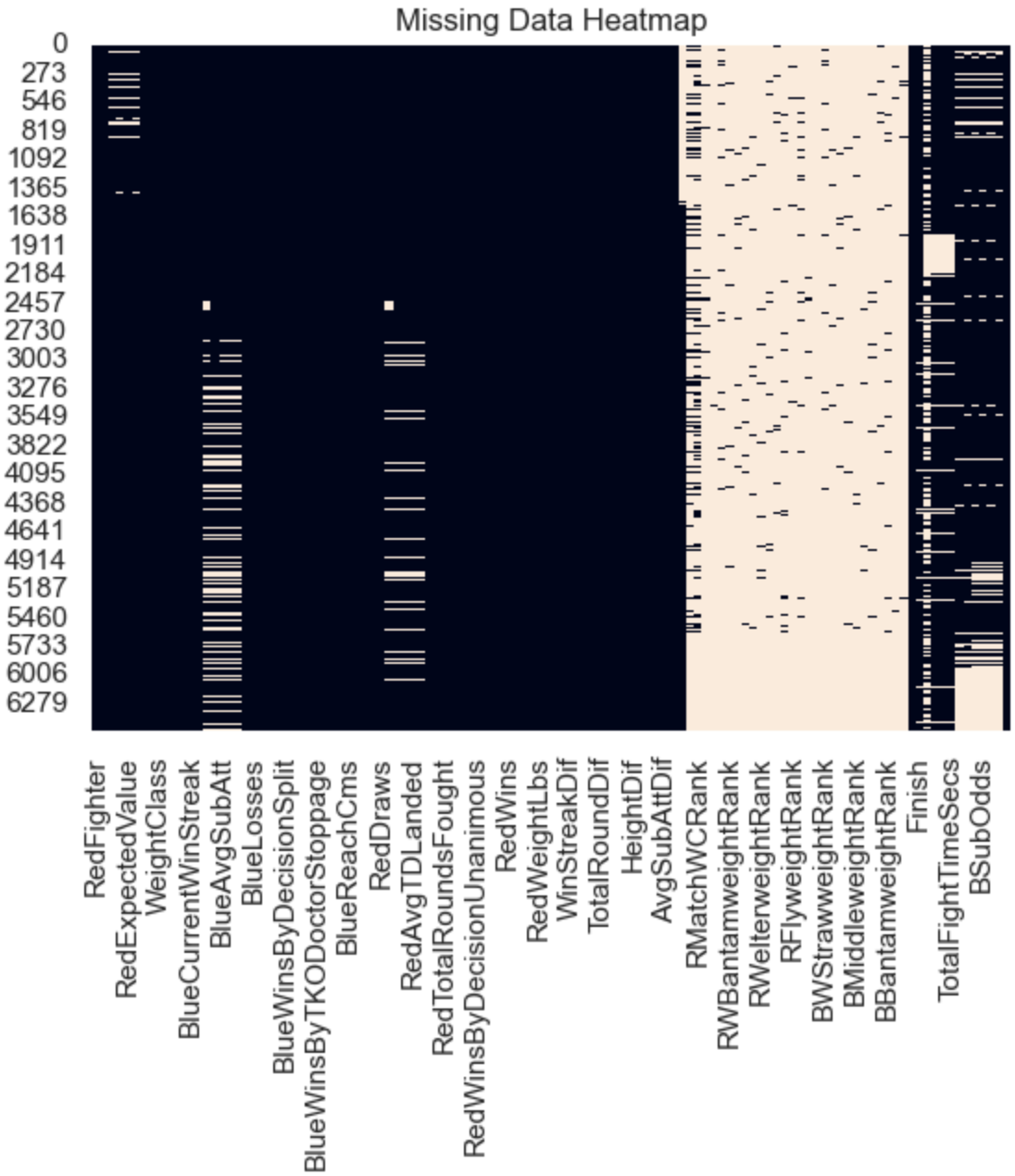
```
max             14.000000    23.000000    16.000000      87.000000

       TotalTitleBoutDif        KODif       SubDif    HeightDif     ReachDif  \
count        6528.000000  6528.000000  6528.000000  6528.000000  6528.000000
mean           -0.301777    -0.510876    -0.308058    -0.006679    -0.299271
std             1.681080     2.149365     1.844810     6.770956     9.132413
min           -16.000000   -21.000000   -16.000000  -187.960000  -187.960000
25%             0.000000    -1.000000    -1.000000    -5.080000    -5.080000
50%             0.000000     0.000000     0.000000     0.000000     0.000000
75%             0.000000     0.000000     0.000000     5.080000     5.080000
max            15.000000    14.000000    10.000000    30.480000    30.480000

            AgeDif    SigStrDif  AvgSubAttDif     AvgTDDif   EmptyArena  \
count  6528.000000  6528.000000   6528.000000  6528.000000  5042.000000
mean      0.096814    -2.663342     -0.071094    -0.171371     0.153114
std       5.201719    19.583493      0.892257     1.754354     0.360133
min     -17.000000  -118.000000     -8.400000   -11.000000     0.000000
25%      -3.000000    -7.895850     -0.444400    -1.010000     0.000000
50%       0.000000    -0.304750      0.000000     0.000000     0.000000
75%       4.000000     2.100000      0.282400     0.700000     0.000000
max      17.000000   128.222200      7.800000    10.860000     1.000000

       BMatchWCRank  RMatchWCRank  RWFlyweightRank  RWFeatherweightRank  \
count   1200.000000   1779.000000        96.000000                  9.0
mean       8.339167      6.953345         7.291667                  0.0
std        4.202839      4.657029         4.975554                  0.0
min        1.000000      0.000000         0.000000                  0.0
25%        5.000000      3.000000         3.000000                  0.0
50%        9.000000      7.000000         7.000000                  0.0
75%       12.000000     11.000000        12.000000                  0.0
max       15.000000     15.000000        15.000000                  0.0

       RWStrawweightRank  RWBantamweightRank  RHeavyweightRank  \
count         146.000000          154.000000        186.000000
mean            7.047945            7.097403          6.881720
std             4.616809            4.803008          4.404787
min             0.000000            0.000000          0.000000
25%             3.000000            3.000000          3.000000
50%             7.000000            7.000000          7.000000
75%            11.000000           11.750000         10.000000
max            15.000000           15.000000         15.000000
```

```
       RLightHeavyweightRank  RMiddleweightRank  RwelterweightRank  \
count             184.000000         182.000000         191.000000
mean                7.119565           7.351648           7.130890
std                 4.638500           4.711274           4.705947
min                 0.000000           0.000000           0.000000
25%                 4.000000           4.000000           3.000000
50%                 7.000000           7.000000           7.000000
75%                11.000000          11.750000          11.000000
max                15.000000          15.000000          15.000000

       RLightweightRank  RFeatherweightRank  RBantamweightRank  \
count        184.000000          177.000000         181.000000
mean           7.032609            6.983051           6.972376
std            4.662257            4.653291           4.709837
min            0.000000            0.000000           0.000000
25%            3.000000            3.000000           3.000000
50%            6.000000            7.000000           7.000000
75%           11.000000           11.000000          11.000000
max           15.000000           15.000000          15.000000

       RFlyweightRank    RPFPRank  BWFlyweightRank  BWFeatherweightRank  \
count      188.000000  253.000000        73.000000                  1.0
mean         6.590426    6.913043         8.410959                  0.0
std          4.621192    4.195655         4.009834                  NaN
min          0.000000    1.000000         1.000000                  0.0
25%          2.000000    3.000000         6.000000                  0.0
50%          7.000000    6.000000         9.000000                  0.0
75%         10.250000   10.000000        12.000000                  0.0
max         15.000000   15.000000        15.000000                  0.0

       BWStrawweightRank  BWBantamweightRank  BHeavyweightRank  \
count         100.000000          107.000000        148.000000
mean            8.170000            8.476636          8.641892
std             4.192478            4.254511          4.090823
min             1.000000            0.000000          1.000000
25%             5.000000            5.000000          5.000000
50%             9.000000            9.000000          9.000000
75%            12.000000           12.000000         12.000000
max            15.000000           15.000000         15.000000

       BLightHeavyweightRank  BMiddleweightRank  BWelterweightRank  \
count             120.000000         137.000000         119.000000
```

```
mean                8.483333           8.554745           8.386555
std                 4.152519           4.311271           4.405523
min                 0.000000           0.000000           1.000000
25%                 6.000000           5.000000           4.500000
50%                 9.000000           9.000000           9.000000
75%                12.000000          12.000000          12.000000
max                15.000000          15.000000          15.000000

       BLightweightRank  BFeatherweightRank  BBantamweightRank  \
count        120.000000          124.000000         119.000000
mean           8.150000            7.967742           8.268908
std            4.076269            4.462919           4.354420
min            1.000000            0.000000           0.000000
25%            5.000000            4.000000           4.000000
50%            8.000000            8.500000           9.000000
75%           11.250000           12.000000          12.000000
max           15.000000           15.000000          15.000000

       BFlyweightRank   BPFPRank  FinishRound  TotalFightTimeSecs  \
count      130.000000  67.000000  5906.000000         5906.000000
mean         8.407692   9.194030     2.424145          657.536234
std          4.307369   4.352773     1.007887          360.383418
min          1.000000   1.000000     1.000000            5.000000
25%          5.000000   5.000000     1.000000          299.000000
50%          8.000000  10.000000     3.000000          900.000000
75%         12.000000  13.500000     3.000000          900.000000
max         15.000000  15.000000     5.000000         1500.000000

        RedDecOdds  BlueDecOdds     RSubOdds     BSubOdds      RKOOdds  \
count  5441.000000  5412.000000  5192.000000  5169.000000  5194.000000
mean    308.333395   425.870288   884.048151  1100.497775   510.891606
std     250.750088   325.940028   601.826547   671.106177   426.563458
min    -440.000000  -200.000000  -370.000000 -1250.000000  -550.000000
25%     170.000000   222.000000   439.750000   600.000000   225.000000
50%     250.000000   350.000000   750.000000  1000.000000   420.000000
75%     400.000000   550.000000  1200.000000  1450.000000   700.000000
max    2400.000000  3000.000000  4665.000000  5000.000000  4000.000000

          BKOOdds
count  5168.000000
mean    636.463235
std     465.014634
```

```
min    -400.000000
25%     310.000000
50%     525.000000
75%     875.000000
max    4000.000000
```






In [251…

```
import numpy as np
df["BlueHeight"] = df["BlueHeightCms"] / 2.54
df['BlueReach']= df['BlueReachCms'] / 2.54
```

```
df['RedHeight'] = df['RedHeightCms'] / 2.54
df['RedReach']= df['RedReachCms'] / 2.54
df.drop(columns = ['BlueHeightCms', 'BlueReachCms', 'RedHeightCms', 'RedReachCms'], axis = 1, inplace = Tru

df['ROver35'] = df['RedAge'].apply(lambda x: int(x > 35))
df['BOver35'] = df['BlueAge'].apply(lambda x: int(x > 35))

df['RWinPct'] = df['RedWins'] / (df['RedWins'] + df['RedLosses'])
df['BWinPct'] = df['BlueWins'] / (df['BlueWins'] + df['BlueLosses'])

df['RedStrikingRatio'] = df['RedAvgSigStrLanded'] / (df['RedAvgSigStrLanded'] + df['BlueAvgSigStrLanded'])
df['BlueStrikingRatio'] = df['BlueAvgSigStrLanded'] / (df['RedAvgSigStrLanded'] + df['BlueAvgSigStrLanded']

df['RedTotalFights'] = df['RedWins'] + df['RedLosses']
df['BlueTotalFights'] = df['BlueWins'] + df['BlueLosses']

df['RedIsGrappler'] = (df['RedAvgTDLanded'] > 3.5).astype(int)
df['BlueIsGrappler'] = (df['BlueAvgTDLanded'] > 3.5).astype(int)

df['RedIsStriker'] = (df['RedAvgSigStrLanded'] > 6.0).astype(int)
df['BlueIsStriker'] = (df['BlueAvgSigStrLanded'] > 6.0).astype(int)

df['RGrapplerVBStriker'] = ((df['RedIsGrappler'] == 1) & (df['BlueIsStriker'] == 1)).astype(int)
df['BGrapplerVRStriker'] = ((df['BlueIsGrappler'] == 1) & (df['RedIsStriker'] == 1)).astype(int)

df['RedStrikingEfficiency'] = df['RedAvgSigStrLanded'] * df['RedAvgSigStrPct']
df['BlueStrikingEfficiency'] = df['BlueAvgSigStrLanded'] * df['BlueAvgSigStrPct']

df['RedTDEfficiency'] = df['RedAvgTDLanded'] * df['RedAvgTDPct']
df['BlueTDEfficiency'] = df['BlueAvgTDLanded'] * df['BlueAvgTDPct']

df['RedEffectiveTD'] = 1 / (df['RedAvgTDLanded'] / df['RedAvgSubAtt'])
df['BlueEffectiveTD'] = 1 / (df['BlueAvgTDLanded'] / df['BlueAvgSubAtt'])
df['EffectiveTDDif'] = df['BlueEffectiveTD'] - df['RedEffectiveTD']

df['RedSize'] = df['RedHeight'] + df['RedReach']
df['BlueSize'] = df['BlueHeight'] + df['BlueReach']

df['Favorite'] = (df['RedOdds'] < df['BlueOdds']).astype(int)

df['Favorite'] = df['Favorite'].map({0 : 'Blue', 1 : 'Red'})
```

```
df['FavoriteWins'] = (df['Favorite'] == df['Winner']).astype(int)
```

In [252…
```
df.head()
```

Out[252…

| | RedFighter | BlueFighter | RedOdds | BlueOdds | RedExpectedValue | BlueExpectedValue | Winner | TitleBout | WeightClass |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 66 | 1009 | -250.0 | 215.0 | 40.0000 | 215.0 | Red | True | 3 |
| **1** | 1441 | 718 | -210.0 | 295.0 | 47.6190 | 295.0 | Red | False | 8 |
| **2** | 307 | 67 | -380.0 | 300.0 | 26.3158 | 300.0 | Red | False | 4 |
| **3** | 221 | 1071 | -950.0 | 625.0 | 10.5263 | 625.0 | Red | False | 2 |
| **4** | 1183 | 524 | -130.0 | 110.0 | 76.9231 | 110.0 | Blue | False | 2 |

In [253…
```
df['draw_diff'] = (df['BlueDraws']-df['RedDraws'])
df['avg_sig_str_pct_diff'] = (df['BlueAvgSigStrPct']-df['RedAvgSigStrPct'])
df['avg_TD_pct_diff'] = (df['BlueAvgTDPct']-df['RedAvgTDPct'])
df['win_by_Decision_Majority_diff'] = (df['BlueWinsByDecisionMajority']-df['RedWinsByDecisionMajority'])
df['win_by_Decision_Split_diff'] = (df['BlueWinsByDecisionSplit']-df['RedWinsByDecisionSplit'])
df['win_by_Decision_Unanimous_diff'] = (df['BlueWinsByDecisionUnanimous']-df['RedWinsByDecisionUnanimous']
df['win_by_TKO_Doctor_Stoppage_diff'] = (df['BlueWinsByTKODoctorStoppage']-df['RedWinsByTKODoctorStoppage'
df['odds_diff'] = (df['BlueOdds']-df['RedOdds'])
df['ev_diff'] = (df['BlueExpectedValue']-df['RedExpectedValue'])


blue_win = df[df['Winner'] == 'Blue']
red_win = df[df['Winner'] == 'Red']

blue_win_percent = len(blue_win) / len(df)
red_win_percent = len(red_win) / len(df)
df.head()
```

Out[253…

| | RedFighter | BlueFighter | RedOdds | BlueOdds | RedExpectedValue | BlueExpectedValue | Winner | TitleBout | WeightClass |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 66 | 1009 | -250.0 | 215.0 | 40.0000 | 215.0 | Red | True | 3 |
| **1** | 1441 | 718 | -210.0 | 295.0 | 47.6190 | 295.0 | Red | False | 8 |
| **2** | 307 | 67 | -380.0 | 300.0 | 26.3158 | 300.0 | Red | False | 4 |
| **3** | 221 | 1071 | -950.0 | 625.0 | 10.5263 | 625.0 | Red | False | 2 |
| **4** | 1183 | 524 | -130.0 | 110.0 | 76.9231 | 110.0 | Blue | False | 2 |

In [254…
```
print(df.shape[0]) # number of rows
print(df.shape[1]) # number of rows
```

```
6528
150
```

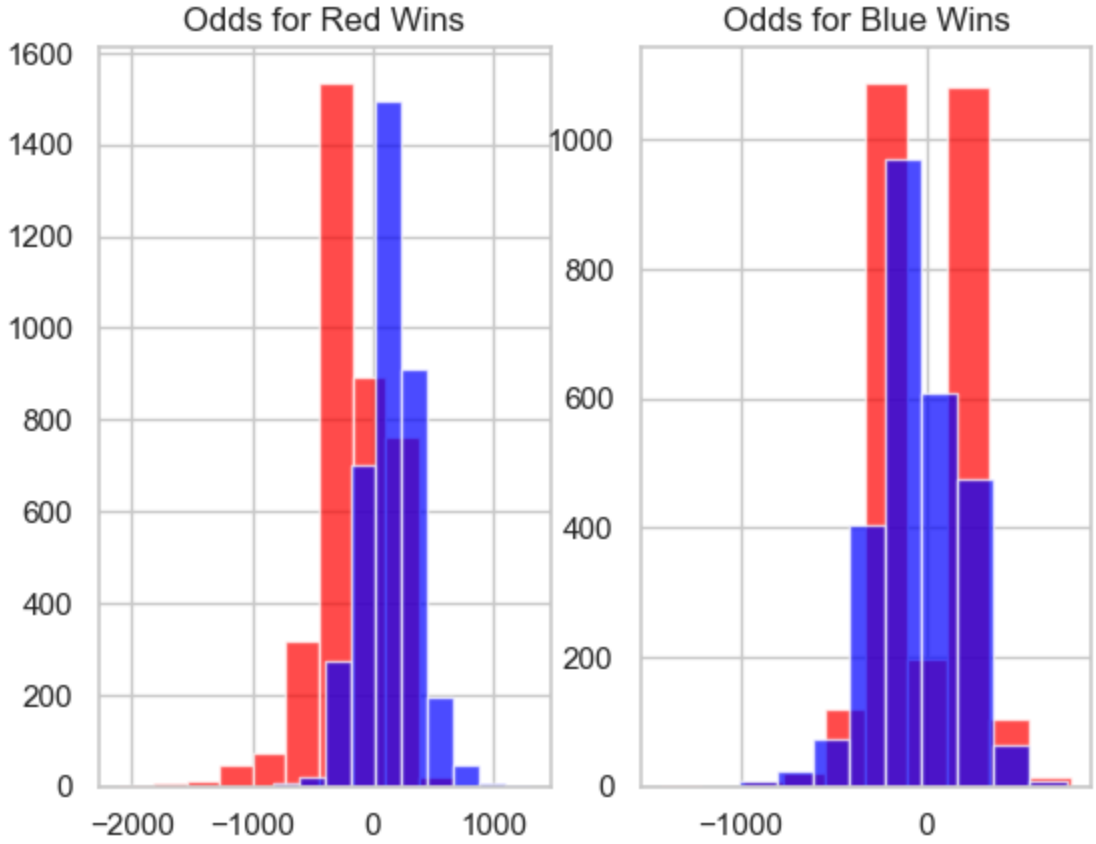
In [255…
```
df.BlueStance.unique()
#It has one spelling mistake
df['BlueStance'].loc[df['BlueStance']=='Switch '] = 'Switch'
#R_Stance doesn't have this error, so we're cool

print(df['BlueStance'].value_counts())
```

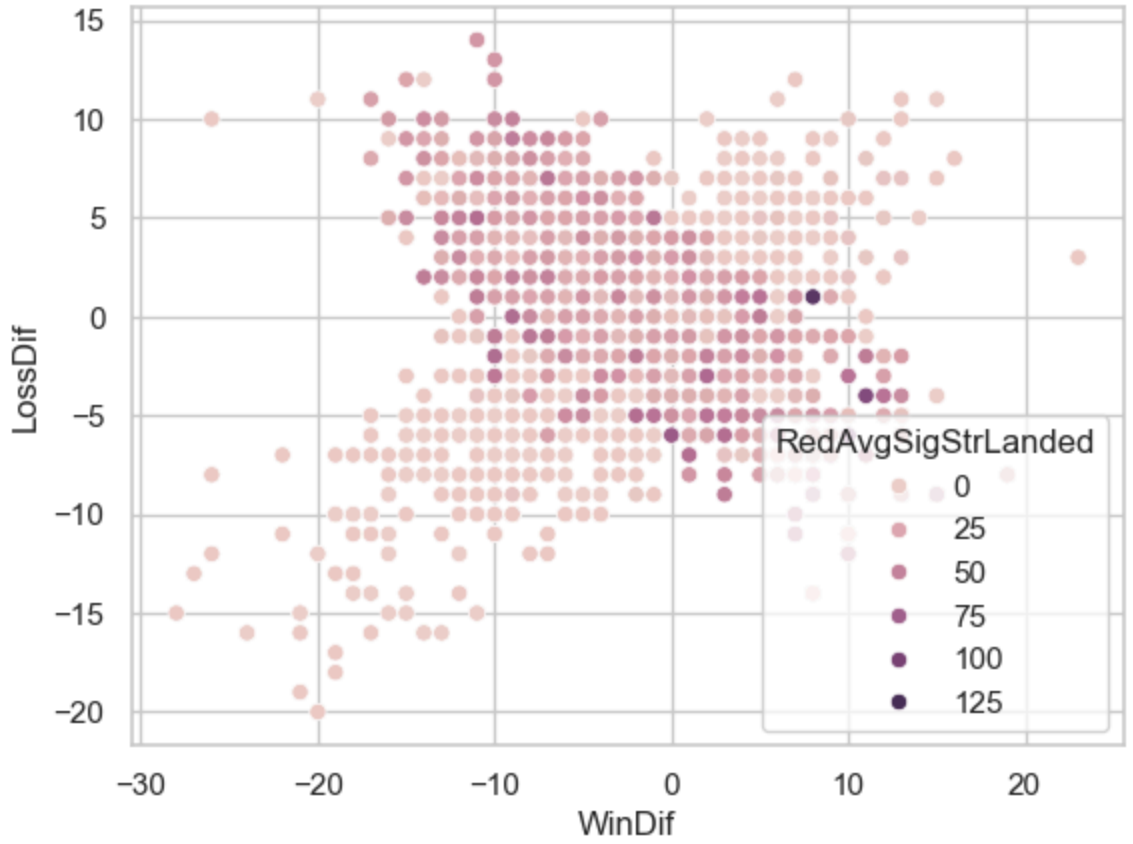
```
BlueStance
Orthodox       4844
Southpaw       1274
Switch          406
Open Stance       1
Name: count, dtype: int64
```

In [256…
```
red_win_percent_high_TDDif = len(red_win[red_win['AvgTDDif'] < -1]) / len(df[df['AvgTDDif'] < -1])
print("Red Win Percentage Increase (w/ +1 more TDs): " + str(red_win_percent_high_TDDif - red_win_percent)

blue_win_percent_high_TDDif = len(blue_win[blue_win['AvgTDDif'] > 1]) / len(df[df['AvgTDDif'] > 1])
print("Blue Win Percentage Increase (w/ +1 more TDs): " + str(blue_win_percent_high_TDDif - blue_win_percer

print('\n')

red_win_percent_highTD_v_lowTD = len(red_win[(red_win['RedAvgTDLanded'] > 2.5) & (red_win['BlueAvgTDLanded
```

```
print("Red Win Percentage Increase (w/ Red TD > 2.5, Blue TD < 1): " + str(red_win_percent_highTD_v_lowTD -

blue_win_percent_highTD_v_lowTD = len(blue_win[(blue_win['BlueAvgTDLanded'] > 2.5) & (blue_win['RedAvgTDLa
print("Blue Win Percentage Increase (w/ Blue TD > 2.5, Red TD < 1): " + str(blue_win_percent_highTD_v_lowT

print('\n')

red_old_win_percent = len(df[(df['ROver35'] == 1) & (df['Winner'] == 'Red')]) / len(df[df['ROver35'] == 1]
print("Decrease in Win Percentage When Red Fighter > 35 y.o.: " + str(red_old_win_percent - red_win_percen
blue_old_win_percent = len(df[(df['BOver35'] == 1) & (df['Winner'] == 'Blue')]) / len(df[df['BOver35'] == 
print("Decrease in Win Percentage When Blue Fighter > 35 y.o.: " + str(blue_old_win_percent - blue_win_per


df.to_csv('df.csv')

print(df['FavoriteWins'].sum() / len(df))
print("Blue Corner Win Percentage: " + str(blue_win_percent))
print("Red Corner Win Percentage: " + str(red_win_percent))


fig, ax = plt.subplots(1, 2)
ax[0].hist(red_win['RedOdds'], alpha=0.7, color='red')
ax[0].hist(red_win['BlueOdds'], alpha=0.7, color='blue')
ax[0].set_title('Odds for Red Wins')
ax[1].hist(blue_win['RedOdds'], alpha=0.7, color='red')
ax[1].hist(blue_win['BlueOdds'], alpha=0.7, color='blue')
ax[1].set_title('Odds for Blue Wins')
fig_name = "odds"
fig_path = os.path.join(output_folder, f"{fig_name}.png")
plt.savefig(fig_path, bbox_inches="tight", dpi=300)
plt.show()
plt.close()

sns.scatterplot(x='WinDif', y='LossDif', data=df, hue='RedAvgSigStrLanded')
fig_name = "strikes_and_wins"
fig_path = os.path.join(output_folder, f"{fig_name}.png")
plt.savefig(fig_path, bbox_inches="tight", dpi=300)
plt.show()
plt.close()
```

```
Red Win Percentage Increase (w/ +1 more TDs): 0.06798639360885117
Blue Win Percentage Increase (w/ +1 more TDs): 0.0755971117904845


Red Win Percentage Increase (w/ Red TD > 2.5, Blue TD < 1): 0.11768577623357035
Blue Win Percentage Increase (w/ Blue TD > 2.5, Red TD < 1): 0.13643041474268885


Decrease in Win Percentage When Red Fighter > 35 y.o.: -0.1215602718360071
Decrease in Win Percentage When Blue Fighter > 35 y.o.: -0.08955866868408013
0.6499693627450981
Blue Corner Win Percentage: 0.41988357843137253
Red Corner Win Percentage: 0.5801164215686274
```






In [257…

```
#calc missing proportions
missing_proportions = df.isnull().sum() / len(df)
print("Proportion of missing values per column:")
print(missing_proportions)

#calc missing percentages
missing_percentages = df.isnull().sum() * 100 / len(df)
print("\nPercentage of missing values per column:")
print(missing_percentages)
```

```
Proportion of missing values per column:
RedFighter                          0.000000
BlueFighter                         0.000000
RedOdds                             0.034773
BlueOdds                            0.034620
RedExpectedValue                    0.034773
BlueExpectedValue                   0.034620
Winner                              0.000000
TitleBout                           0.000000
WeightClass                         0.000000
Gender                              0.000000
NumberOfRounds                      0.000000
BlueCurrentLoseStreak               0.000000
BlueCurrentWinStreak                0.000000
BlueDraws                           0.000000
BlueAvgSigStrLanded                 0.142463
BlueAvgSigStrPct                    0.117188
BlueAvgSubAtt                       0.127451
BlueAvgTDLanded                     0.127604
BlueAvgTDPct                        0.128983
BlueLongestWinStreak                0.000000
BlueLosses                          0.000000
BlueTotalRoundsFought               0.000000
BlueTotalTitleBouts                 0.000000
BlueWinsByDecisionMajority          0.000000
BlueWinsByDecisionSplit             0.000000
BlueWinsByDecisionUnanimous         0.000000
BlueWinsByKO                        0.000000
BlueWinsBySubmission                0.000000
BlueWinsByTKODoctorStoppage         0.000000
BlueWins                            0.000000
BlueStance                          0.000460
BlueWeightLbs                       0.000000
RedCurrentLoseStreak                0.000000
RedCurrentWinStreak                 0.000000
RedDraws                            0.000000
RedAvgSigStrLanded                  0.069700
RedAvgSigStrPct                     0.054688
RedAvgSubAtt                        0.054688
RedAvgTDLanded                      0.054688
RedAvgTDPct                         0.056219
RedLongestWinStreak                 0.000000
```

| | |
|---|---|
| RedLosses | 0.000000 |
| RedTotalRoundsFought | 0.000000 |
| RedTotalTitleBouts | 0.000000 |
| RedWinsByDecisionMajority | 0.000000 |
| RedWinsByDecisionSplit | 0.000000 |
| RedWinsByDecisionUnanimous | 0.000000 |
| RedWinsByKO | 0.000000 |
| RedWinsBySubmission | 0.000000 |
| RedWinsByTKODoctorStoppage | 0.000000 |
| RedWins | 0.000000 |
| RedStance | 0.000000 |
| RedWeightLbs | 0.000000 |
| RedAge | 0.000000 |
| BlueAge | 0.000000 |
| LoseStreakDif | 0.000000 |
| WinStreakDif | 0.000000 |
| LongestWinStreakDif | 0.000000 |
| WinDif | 0.000000 |
| LossDif | 0.000000 |
| TotalRoundDif | 0.000000 |
| TotalTitleBoutDif | 0.000000 |
| KODif | 0.000000 |
| SubDif | 0.000000 |
| HeightDif | 0.000000 |
| ReachDif | 0.000000 |
| AgeDif | 0.000000 |
| SigStrDif | 0.000000 |
| AvgSubAttDif | 0.000000 |
| AvgTDDif | 0.000000 |
| EmptyArena | 0.227635 |
| BMatchWCRank | 0.816176 |
| RMatchWCRank | 0.727482 |
| RWFlyweightRank | 0.985294 |
| RWFeatherweightRank | 0.998621 |
| RWStrawweightRank | 0.977635 |
| RWBantamweightRank | 0.976409 |
| RHeavyweightRank | 0.971507 |
| RLightHeavyweightRank | 0.971814 |
| RMiddleweightRank | 0.972120 |
| RWelterweightRank | 0.970741 |
| RLightweightRank | 0.971814 |
| RFeatherweightRank | 0.972886 |

```
RBantamweightRank          0.972273
RFlyweightRank             0.971201
RPFPRank                   0.961244
BWFlyweightRank            0.988817
BWFeatherweightRank        0.999847
BWStrawweightRank          0.984681
BWBantamweightRank         0.983609
BHeavyweightRank           0.977328
BLightHeavyweightRank      0.981618
BMiddleweightRank          0.979013
BWelterweightRank          0.981771
BLightweightRank           0.981618
BFeatherweightRank         0.981005
BBantamweightRank          0.981771
BFlyweightRank             0.980086
BPFPRank                   0.989737
BetterRank                 0.000000
Finish                     0.036458
FinishDetails              0.556985
FinishRound                0.095282
FinishRoundTime            0.095282
TotalFightTimeSecs         0.095282
RedDecOdds                 0.166513
BlueDecOdds                0.170956
RSubOdds                   0.204657
BSubOdds                   0.208180
RKOOdds                    0.204350
BKOOdds                    0.208333
WeightClassLabel           0.000000
BlueHeight                 0.000000
BlueReach                  0.000000
RedHeight                  0.000000
RedReach                   0.000000
ROver35                    0.000000
BOver35                    0.000000
RWinPct                    0.070619
BWinPct                    0.170037
RedStrikingRatio           0.161152
BlueStrikingRatio          0.161152
RedTotalFights             0.000000
BlueTotalFights            0.000000
RedIsGrappler              0.000000
```

```
BlueIsGrappler                    0.000000
RedIsStriker                      0.000000
BlueIsStriker                     0.000000
RGrapplerVBStriker                0.000000
BGrapplerVRStriker                0.000000
RedStrikingEfficiency             0.069700
BlueStrikingEfficiency            0.142463
RedTDEfficiency                   0.056219
BlueTDEfficiency                  0.128983
RedEffectiveTD                    0.142616
BlueEffectiveTD                   0.250000
EffectiveTDDif                    0.314338
RedSize                           0.000000
BlueSize                          0.000000
Favorite                          0.000000
FavoriteWins                      0.000000
draw_diff                         0.000000
avg_sig_str_pct_diff              0.135876
avg_TD_pct_diff                   0.145987
win_by_Decision_Majority_diff     0.000000
win_by_Decision_Split_diff        0.000000
win_by_Decision_Unanimous_diff    0.000000
win_by_TKO_Doctor_Stoppage_diff   0.000000
odds_diff                         0.036458
ev_diff                           0.036458
dtype: float64

Percentage of missing values per column:
RedFighter                         0.000000
BlueFighter                        0.000000
RedOdds                            3.477328
BlueOdds                           3.462010
RedExpectedValue                   3.477328
BlueExpectedValue                  3.462010
Winner                             0.000000
TitleBout                          0.000000
WeightClass                        0.000000
Gender                             0.000000
NumberOfRounds                     0.000000
BlueCurrentLoseStreak              0.000000
BlueCurrentWinStreak               0.000000
BlueDraws                          0.000000
```

```
BlueAvgSigStrLanded              14.246324
BlueAvgSigStrPct                 11.718750
BlueAvgSubAtt                    12.745098
BlueAvgTDLanded                  12.760417
BlueAvgTDPct                     12.898284
BlueLongestWinStreak              0.000000
BlueLosses                        0.000000
BlueTotalRoundsFought             0.000000
BlueTotalTitleBouts               0.000000
BlueWinsByDecisionMajority        0.000000
BlueWinsByDecisionSplit           0.000000
BlueWinsByDecisionUnanimous       0.000000
BlueWinsByKO                      0.000000
BlueWinsBySubmission              0.000000
BlueWinsByTKODoctorStoppage       0.000000
BlueWins                          0.000000
BlueStance                        0.045956
BlueWeightLbs                     0.000000
RedCurrentLoseStreak              0.000000
RedCurrentWinStreak               0.000000
RedDraws                          0.000000
RedAvgSigStrLanded                6.969975
RedAvgSigStrPct                   5.468750
RedAvgSubAtt                      5.468750
RedAvgTDLanded                    5.468750
RedAvgTDPct                       5.621936
RedLongestWinStreak               0.000000
RedLosses                         0.000000
RedTotalRoundsFought              0.000000
RedTotalTitleBouts                0.000000
RedWinsByDecisionMajority         0.000000
RedWinsByDecisionSplit            0.000000
RedWinsByDecisionUnanimous        0.000000
RedWinsByKO                       0.000000
RedWinsBySubmission               0.000000
RedWinsByTKODoctorStoppage        0.000000
RedWins                           0.000000
RedStance                         0.000000
RedWeightLbs                      0.000000
RedAge                            0.000000
BlueAge                           0.000000
LoseStreakDif                     0.000000
```

```
WinStreakDif                0.000000
LongestWinStreakDif         0.000000
WinDif                      0.000000
LossDif                     0.000000
TotalRoundDif               0.000000
TotalTitleBoutDif           0.000000
KODif                       0.000000
SubDif                      0.000000
HeightDif                   0.000000
ReachDif                    0.000000
AgeDif                      0.000000
SigStrDif                   0.000000
AvgSubAttDif                0.000000
AvgTDDif                    0.000000
EmptyArena                 22.763480
BMatchWCRank               81.617647
RMatchWCRank               72.748162
RWFlyweightRank            98.529412
RWFeatherweightRank        99.862132
RWStrawweightRank          97.763480
RWBantamweightRank         97.640931
RHeavyweightRank           97.150735
RLightHeavyweightRank      97.181373
RMiddleweightRank          97.212010
RWelterweightRank          97.074142
RLightweightRank           97.181373
RFeatherweightRank         97.288603
RBantamweightRank          97.227328
RFlyweightRank             97.120098
RPFPRank                   96.124387
BWFlyweightRank            98.881740
BWFeatherweightRank        99.984681
BWStrawweightRank          98.468137
BWBantamweightRank         98.360907
BHeavyweightRank           97.732843
BLightHeavyweightRank      98.161765
BMiddleweightRank          97.901348
BWelterweightRank          98.177083
BLightweightRank           98.161765
BFeatherweightRank         98.100490
BBantamweightRank          98.177083
BFlyweightRank             98.008578
```

| | |
|---|---|
| BPFPRank | 98.973652 |
| BetterRank | 0.000000 |
| Finish | 3.645833 |
| FinishDetails | 55.698529 |
| FinishRound | 9.528186 |
| FinishRoundTime | 9.528186 |
| TotalFightTimeSecs | 9.528186 |
| RedDecOdds | 16.651348 |
| BlueDecOdds | 17.095588 |
| RSubOdds | 20.465686 |
| BSubOdds | 20.818015 |
| RKOOdds | 20.435049 |
| BKOOdds | 20.833333 |
| WeightClassLabel | 0.000000 |
| BlueHeight | 0.000000 |
| BlueReach | 0.000000 |
| RedHeight | 0.000000 |
| RedReach | 0.000000 |
| ROver35 | 0.000000 |
| BOver35 | 0.000000 |
| RWinPct | 7.061887 |
| BWinPct | 17.003676 |
| RedStrikingRatio | 16.115196 |
| BlueStrikingRatio | 16.115196 |
| RedTotalFights | 0.000000 |
| BlueTotalFights | 0.000000 |
| RedIsGrappler | 0.000000 |
| BlueIsGrappler | 0.000000 |
| RedIsStriker | 0.000000 |
| BlueIsStriker | 0.000000 |
| RGrapplerVBStriker | 0.000000 |
| BGrapplerVRStriker | 0.000000 |
| RedStrikingEfficiency | 6.969975 |
| BlueStrikingEfficiency | 14.246324 |
| RedTDEfficiency | 5.621936 |
| BlueTDEfficiency | 12.898284 |
| RedEffectiveTD | 14.261642 |
| BlueEffectiveTD | 25.000000 |
| EffectiveTDDif | 31.433824 |
| RedSize | 0.000000 |
| BlueSize | 0.000000 |
| Favorite | 0.000000 |

```
FavoriteWins                        0.000000
draw_diff                           0.000000
avg_sig_str_pct_diff               13.587623
avg_TD_pct_diff                    14.598652
win_by_Decision_Majority_diff       0.000000
win_by_Decision_Split_diff          0.000000
win_by_Decision_Unanimous_diff      0.000000
win_by_TKO_Doctor_Stoppage_diff     0.000000
odds_diff                           3.645833
ev_diff                             3.645833
dtype: float64
```

In [258…
```
#count columns with any missing data
columns_with_missing_data = df.isna().any()
number_of_columns_with_missing_data = columns_with_missing_data.sum()
number_of_columns_with_missing_data
```

Out[258…
```
np.int64(70)
```

In [259…
```
print(df.shape[0]) # number of rows
print(df.shape[1]) # number of rows
#% of missing values per column
missing_percentage = df.isnull().sum() * 100 / len(df)
print(missing_percentage)
```

```
6528
150
RedFighter                      0.000000
BlueFighter                     0.000000
RedOdds                         3.477328
BlueOdds                        3.462010
RedExpectedValue                3.477328
BlueExpectedValue               3.462010
Winner                          0.000000
TitleBout                       0.000000
WeightClass                     0.000000
Gender                          0.000000
NumberOfRounds                  0.000000
BlueCurrentLoseStreak           0.000000
BlueCurrentWinStreak            0.000000
BlueDraws                       0.000000
BlueAvgSigStrLanded            14.246324
BlueAvgSigStrPct               11.718750
BlueAvgSubAtt                  12.745098
BlueAvgTDLanded                12.760417
BlueAvgTDPct                   12.898284
BlueLongestWinStreak            0.000000
BlueLosses                      0.000000
BlueTotalRoundsFought           0.000000
BlueTotalTitleBouts             0.000000
BlueWinsByDecisionMajority      0.000000
BlueWinsByDecisionSplit         0.000000
BlueWinsByDecisionUnanimous     0.000000
BlueWinsByKO                    0.000000
BlueWinsBySubmission            0.000000
BlueWinsByTKODoctorStoppage     0.000000
BlueWins                        0.000000
BlueStance                      0.045956
BlueWeightLbs                   0.000000
RedCurrentLoseStreak            0.000000
RedCurrentWinStreak             0.000000
RedDraws                        0.000000
RedAvgSigStrLanded              6.969975
RedAvgSigStrPct                 5.468750
RedAvgSubAtt                    5.468750
RedAvgTDLanded                  5.468750
RedAvgTDPct                     5.621936
```

```
RedLongestWinStreak              0.000000
RedLosses                        0.000000
RedTotalRoundsFought             0.000000
RedTotalTitleBouts               0.000000
RedWinsByDecisionMajority        0.000000
RedWinsByDecisionSplit           0.000000
RedWinsByDecisionUnanimous       0.000000
RedWinsByKO                      0.000000
RedWinsBySubmission              0.000000
RedWinsByTKODoctorStoppage       0.000000
RedWins                          0.000000
RedStance                        0.000000
RedWeightLbs                     0.000000
RedAge                           0.000000
BlueAge                          0.000000
LoseStreakDif                    0.000000
WinStreakDif                     0.000000
LongestWinStreakDif              0.000000
WinDif                           0.000000
LossDif                          0.000000
TotalRoundDif                    0.000000
TotalTitleBoutDif                0.000000
KODif                            0.000000
SubDif                           0.000000
HeightDif                        0.000000
ReachDif                         0.000000
AgeDif                           0.000000
SigStrDif                        0.000000
AvgSubAttDif                     0.000000
AvgTDDif                         0.000000
EmptyArena                      22.763480
BMatchWCRank                    81.617647
RMatchWCRank                    72.748162
RWFlyweightRank                 98.529412
RWFeatherweightRank             99.862132
RWStrawweightRank               97.763480
RWBantamweightRank              97.640931
RHeavyweightRank                97.150735
RLightHeavyweightRank           97.181373
RMiddleweightRank               97.212010
RWelterweightRank               97.074142
RLightweightRank                97.181373
```

| | |
|---|---|
| RFeatherweightRank | 97.288603 |
| RBantamweightRank | 97.227328 |
| RFlyweightRank | 97.120098 |
| RPFPRank | 96.124387 |
| BWFlyweightRank | 98.881740 |
| BWFeatherweightRank | 99.984681 |
| BWStrawweightRank | 98.468137 |
| BWBantamweightRank | 98.360907 |
| BHeavyweightRank | 97.732843 |
| BLightHeavyweightRank | 98.161765 |
| BMiddleweightRank | 97.901348 |
| BWelterweightRank | 98.177083 |
| BLightweightRank | 98.161765 |
| BFeatherweightRank | 98.100490 |
| BBantamweightRank | 98.177083 |
| BFlyweightRank | 98.008578 |
| BPFPRank | 98.973652 |
| BetterRank | 0.000000 |
| Finish | 3.645833 |
| FinishDetails | 55.698529 |
| FinishRound | 9.528186 |
| FinishRoundTime | 9.528186 |
| TotalFightTimeSecs | 9.528186 |
| RedDecOdds | 16.651348 |
| BlueDecOdds | 17.095588 |
| RSubOdds | 20.465686 |
| BSubOdds | 20.818015 |
| RKOOdds | 20.435049 |
| BKOOdds | 20.833333 |
| WeightClassLabel | 0.000000 |
| BlueHeight | 0.000000 |
| BlueReach | 0.000000 |
| RedHeight | 0.000000 |
| RedReach | 0.000000 |
| ROver35 | 0.000000 |
| BOver35 | 0.000000 |
| RWinPct | 7.061887 |
| BWinPct | 17.003676 |
| RedStrikingRatio | 16.115196 |
| BlueStrikingRatio | 16.115196 |
| RedTotalFights | 0.000000 |
| BlueTotalFights | 0.000000 |

```
RedIsGrappler                      0.000000
BlueIsGrappler                     0.000000
RedIsStriker                       0.000000
BlueIsStriker                      0.000000
RGrapplerVBStriker                 0.000000
BGrapplerVRStriker                 0.000000
RedStrikingEfficiency              6.969975
BlueStrikingEfficiency            14.246324
RedTDEfficiency                    5.621936
BlueTDEfficiency                  12.898284
RedEffectiveTD                    14.261642
BlueEffectiveTD                   25.000000
EffectiveTDDif                    31.433824
RedSize                            0.000000
BlueSize                           0.000000
Favorite                           0.000000
FavoriteWins                       0.000000
draw_diff                          0.000000
avg_sig_str_pct_diff              13.587623
avg_TD_pct_diff                   14.598652
win_by_Decision_Majority_diff      0.000000
win_by_Decision_Split_diff         0.000000
win_by_Decision_Unanimous_diff     0.000000
win_by_TKO_Doctor_Stoppage_diff    0.000000
odds_diff                          3.645833
ev_diff                            3.645833
dtype: float64
```

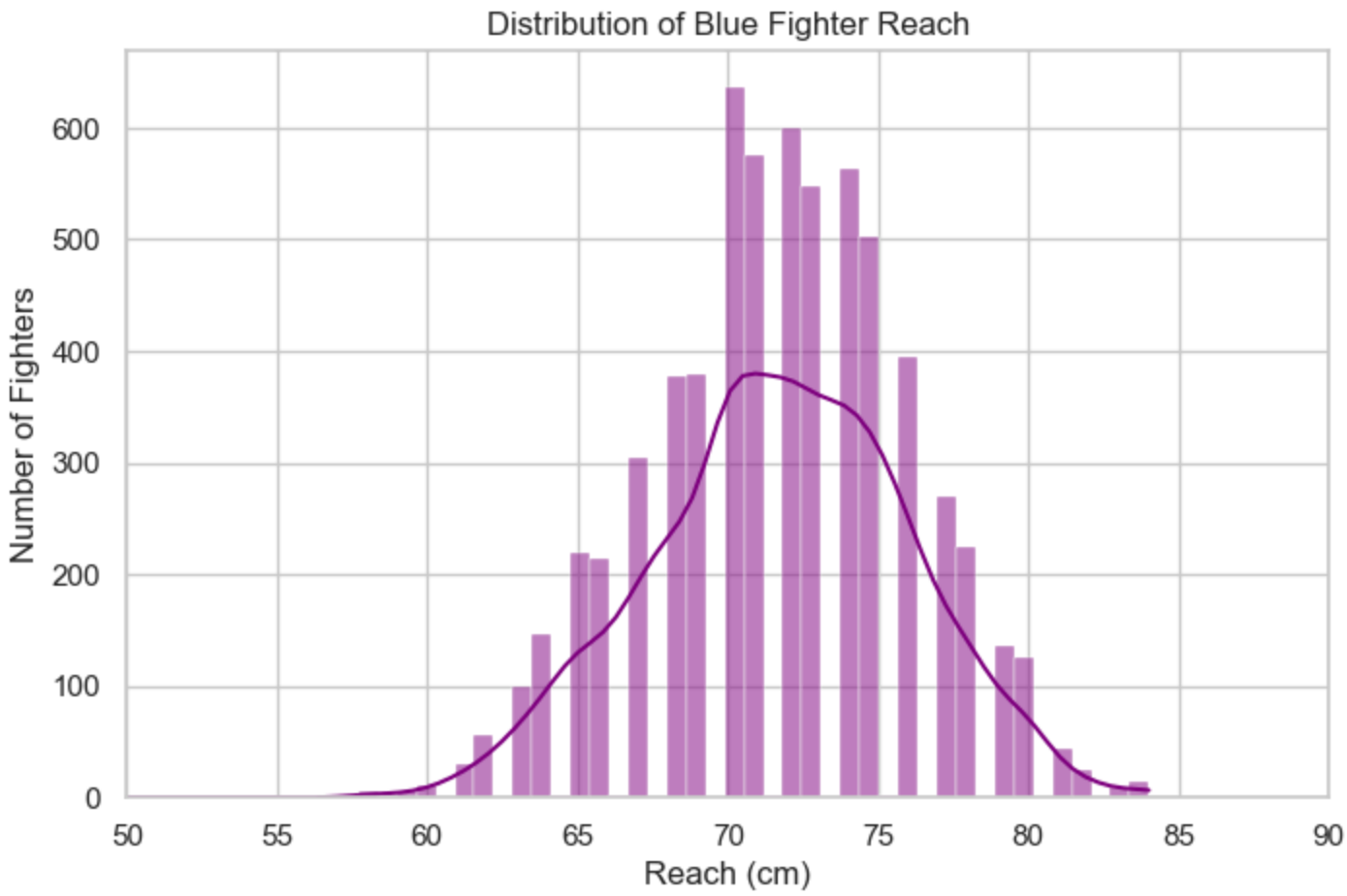
In [260…

```
# y = df['Winner']
# X = df.drop(columns=['Winner'])

# # Encode target variable labels as integers
# le = LabelEncoder()
# y = le.fit_transform(y)
```

In [261…

```
plt.figure(figsize=(8, 5))
sns.histplot(df['BlueReach'], kde=True, color='purple')
plt.title('Distribution of Blue Fighter Reach')
plt.xlabel('Reach (cm)')
plt.ylabel('Number of Fighters')
plt.xlim(50, 90)  # limit x-axis range to be reasonable bc who has a 20in reach?
fig_name = "dist_blue_reach"
```

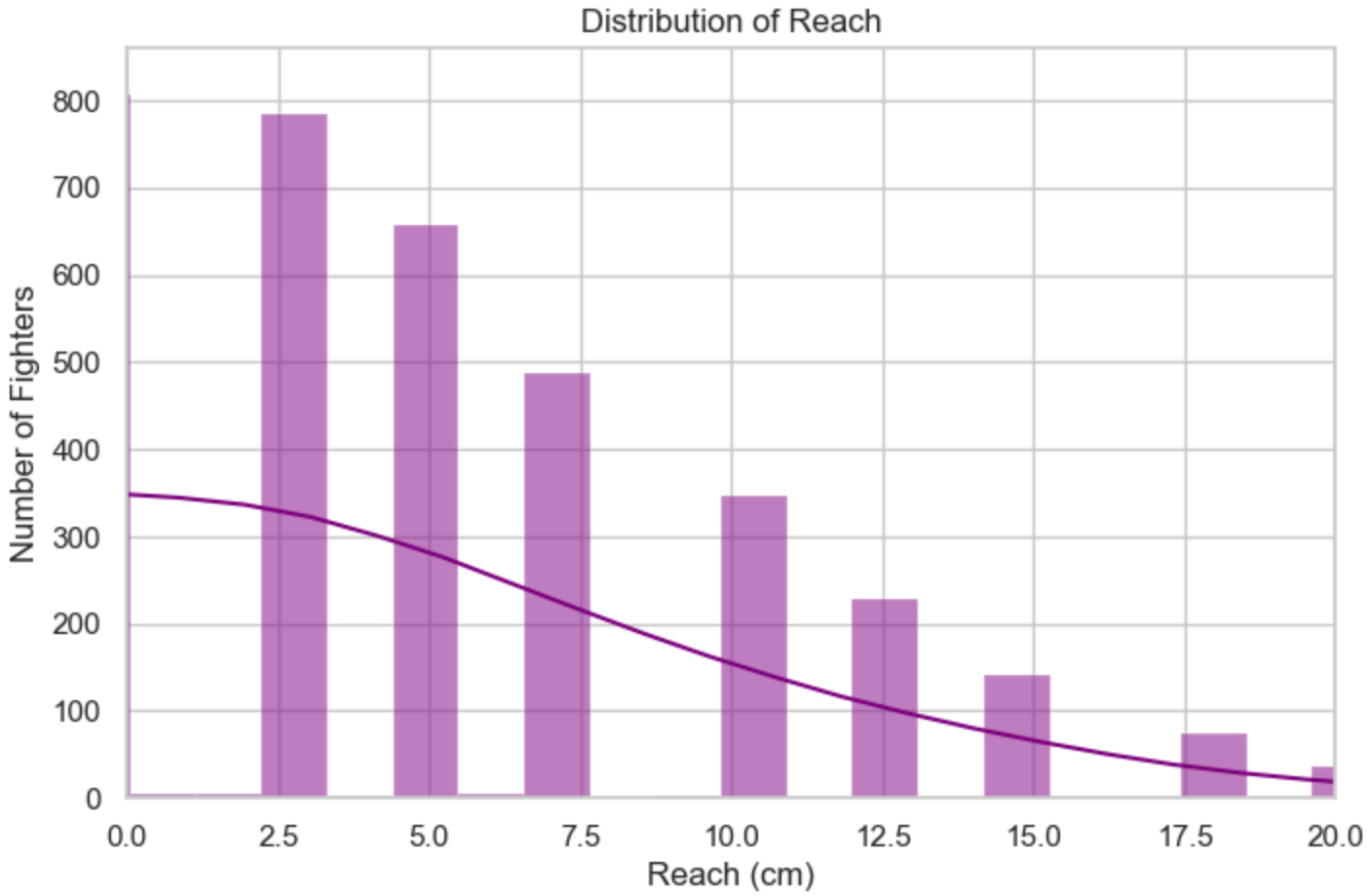
```
fig_path = os.path.join(output_folder, f"{fig_name}.png")
plt.savefig(fig_path, bbox_inches="tight", dpi=300)
plt.show()
plt.close()
```




In [262…

```
plt.figure(figsize=(8, 5))
sns.histplot(df['ReachDif'], kde=True, color='purple')
plt.title('Distribution of Reach')
plt.xlabel('Reach (cm)')
plt.ylabel('Number of Fighters')
plt.xlim(0, 20)  # limit x-axis range to be reasonable bc who has a 20in reach?
fig_name = "dist_reach"
fig_path = os.path.join(output_folder, f"{fig_name}.png")
```

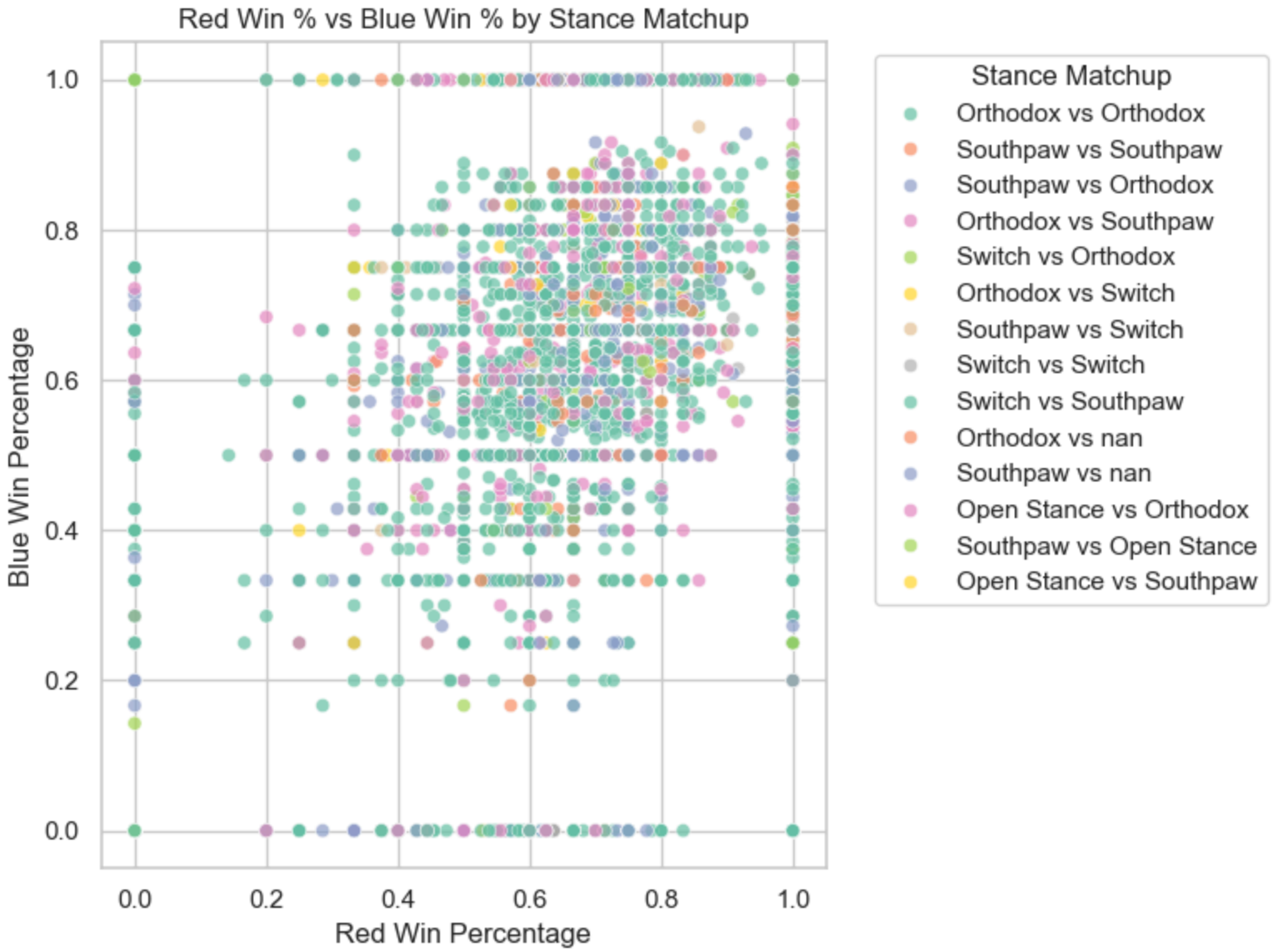
```
plt.savefig(fig_path, bbox_inches="tight", dpi=300)
plt.show()
plt.close()
```




In [263…

```
df['StanceCombo'] = df['RedStance'].astype(str) + ' vs ' + df['BlueStance'].astype(str)

plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='RWinPct', y='BWinPct', hue='StanceCombo', palette='Set2', alpha=0.7)
plt.title('Red Win % vs Blue Win % by Stance Matchup')
plt.xlabel('Red Win Percentage')
plt.ylabel('Blue Win Percentage')
plt.legend(title='Stance Matchup', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
```

```
fig_name = "stance_combo"
fig_path = os.path.join(output_folder, f"{fig_name}.png")
plt.savefig(fig_path, bbox_inches="tight", dpi=300)
plt.show()
plt.close()
```



In [264…
```
print(df.dtypes)
```

```
RedFighter                      int64
BlueFighter                     int64
RedOdds                       float64
BlueOdds                      float64
RedExpectedValue              float64
BlueExpectedValue             float64
Winner                         object
TitleBout                        bool
WeightClass                     int64
Gender                          int64
NumberOfRounds                  int64
BlueCurrentLoseStreak           int64
BlueCurrentWinStreak            int64
BlueDraws                       int64
BlueAvgSigStrLanded           float64
BlueAvgSigStrPct              float64
BlueAvgSubAtt                 float64
BlueAvgTDLanded               float64
BlueAvgTDPct                  float64
BlueLongestWinStreak            int64
BlueLosses                      int64
BlueTotalRoundsFought           int64
BlueTotalTitleBouts             int64
BlueWinsByDecisionMajority      int64
BlueWinsByDecisionSplit         int64
BlueWinsByDecisionUnanimous     int64
BlueWinsByKO                    int64
BlueWinsBySubmission            int64
BlueWinsByTKODoctorStoppage     int64
BlueWins                        int64
BlueStance                     object
BlueWeightLbs                   int64
RedCurrentLoseStreak            int64
RedCurrentWinStreak             int64
RedDraws                        int64
RedAvgSigStrLanded            float64
RedAvgSigStrPct               float64
RedAvgSubAtt                  float64
RedAvgTDLanded                float64
RedAvgTDPct                   float64
RedLongestWinStreak             int64
RedLosses                       int64
```

```
RedTotalRoundsFought               int64
RedTotalTitleBouts                 int64
RedWinsByDecisionMajority          int64
RedWinsByDecisionSplit             int64
RedWinsByDecisionUnanimous         int64
RedWinsByKO                        int64
RedWinsBySubmission                int64
RedWinsByTKODoctorStoppage         int64
RedWins                            int64
RedStance                         object
RedWeightLbs                       int64
RedAge                             int64
BlueAge                            int64
LoseStreakDif                      int64
WinStreakDif                       int64
LongestWinStreakDif                int64
WinDif                             int64
LossDif                            int64
TotalRoundDif                      int64
TotalTitleBoutDif                  int64
KODif                              int64
SubDif                             int64
HeightDif                        float64
ReachDif                         float64
AgeDif                             int64
SigStrDif                        float64
AvgSubAttDif                     float64
AvgTDDif                         float64
EmptyArena                       float64
BMatchWCRank                     float64
RMatchWCRank                     float64
RWFlyweightRank                  float64
RWFeatherweightRank              float64
RWStrawweightRank                float64
RWBantamweightRank               float64
RHeavyweightRank                 float64
RLightHeavyweightRank            float64
RMiddleweightRank                float64
RWelterweightRank                float64
RLightweightRank                 float64
RFeatherweightRank               float64
RBantamweightRank                float64
```

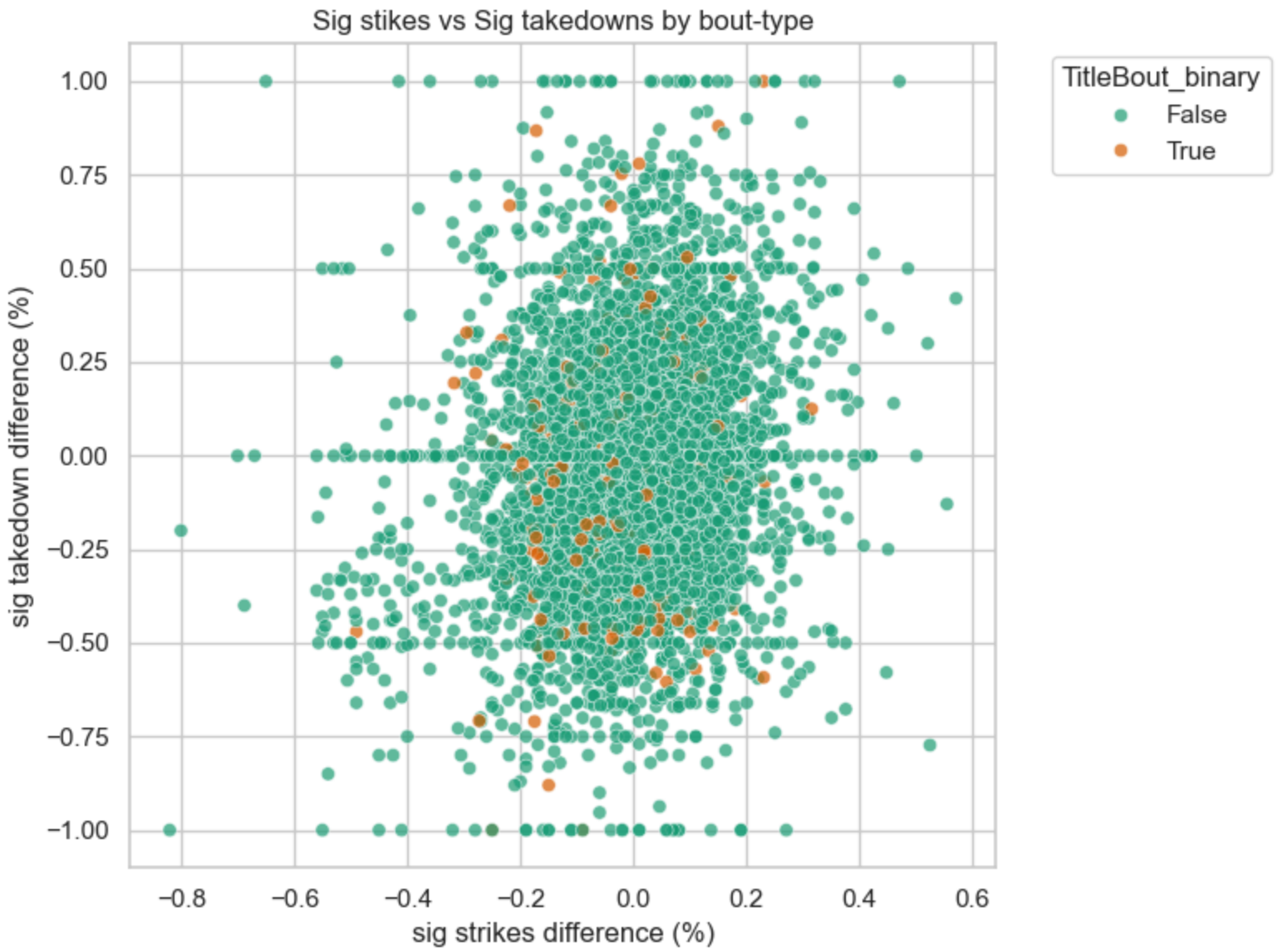
```
RFlyweightRank              float64
RPFPRank                    float64
BWFlyweightRank             float64
BWFeatherweightRank         float64
BWStrawweightRank           float64
BWBantamweightRank          float64
BHeavyweightRank            float64
BLightHeavyweightRank       float64
BMiddleweightRank           float64
BWelterweightRank           float64
BLightweightRank            float64
BFeatherweightRank          float64
BBantamweightRank           float64
BFlyweightRank              float64
BPFPRank                    float64
BetterRank                   object
Finish                       object
FinishDetails                object
FinishRound                 float64
FinishRoundTime              object
TotalFightTimeSecs          float64
RedDecOdds                  float64
BlueDecOdds                 float64
RSubOdds                    float64
BSubOdds                    float64
RKOOdds                     float64
BKOOdds                     float64
WeightClassLabel             object
BlueHeight                  float64
BlueReach                   float64
RedHeight                   float64
RedReach                    float64
ROver35                       int64
BOver35                       int64
RWinPct                     float64
BWinPct                     float64
RedStrikingRatio            float64
BlueStrikingRatio           float64
RedTotalFights                int64
BlueTotalFights               int64
RedIsGrappler                 int64
BlueIsGrappler                int64
```

```
RedIsStriker                        int64
BlueIsStriker                       int64
RGrapplerVBStriker                  int64
BGrapplerVRStriker                  int64
RedStrikingEfficiency             float64
BlueStrikingEfficiency            float64
RedTDEfficiency                   float64
BlueTDEfficiency                  float64
RedEffectiveTD                    float64
BlueEffectiveTD                   float64
EffectiveTDDif                    float64
RedSize                           float64
BlueSize                          float64
Favorite                           object
FavoriteWins                        int64
draw_diff                           int64
avg_sig_str_pct_diff              float64
avg_TD_pct_diff                   float64
win_by_Decision_Majority_diff       int64
win_by_Decision_Split_diff          int64
win_by_Decision_Unanimous_diff      int64
win_by_TKO_Doctor_Stoppage_diff     int64
odds_diff                         float64
ev_diff                           float64
StanceCombo                        object
dtype: object
```

In [265…

```
plt.figure(figsize=(8, 6))
sns.scatterplot(
    data=df,
    x='avg_sig_str_pct_diff',
    y='avg_TD_pct_diff',
    hue='TitleBout',
    palette='Dark2',
    alpha=0.7
)
plt.title('Sig stikes vs Sig takedowns by bout-type')
plt.xlabel('sig strikes difference (%)')
plt.ylabel('sig takedown difference (%)')
plt.legend(title='TitleBout_binary', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
fig_name = "sigstrikes_sigtakedowns"
```

```
fig_path = os.path.join(output_folder, f"{fig_name}.png")
plt.savefig(fig_path, bbox_inches="tight", dpi=300)
plt.show()
plt.close()
```




In [266…

```
# a list of all fighter names from the red and blue corner
all_fighters_df = pd.concat([df.RedFighter, df.BlueFighter], ignore_index = True)
```

```
# how many unique names there are
all_fighters_nbr = all_fighters_df.nunique()
print(all_fighters_nbr)

# a list of all fighter names from the red and blue corner
all_fighters_df = pd.concat([df.RedFighter, df.BlueFighter], ignore_index = True)

#check how many unique names there are
all_fighters_nbr = all_fighters_df.nunique()
print(all_fighters_nbr)

df['WeightClass'].value_counts()
```

```
1922
1922
```

Out[266…

```
WeightClass
6     1074
8     1026
7      783
2      754
0      679
5      501
4      496
3      353
12     320
11     239
9      213
1       61
10      29
Name: count, dtype: int64
```

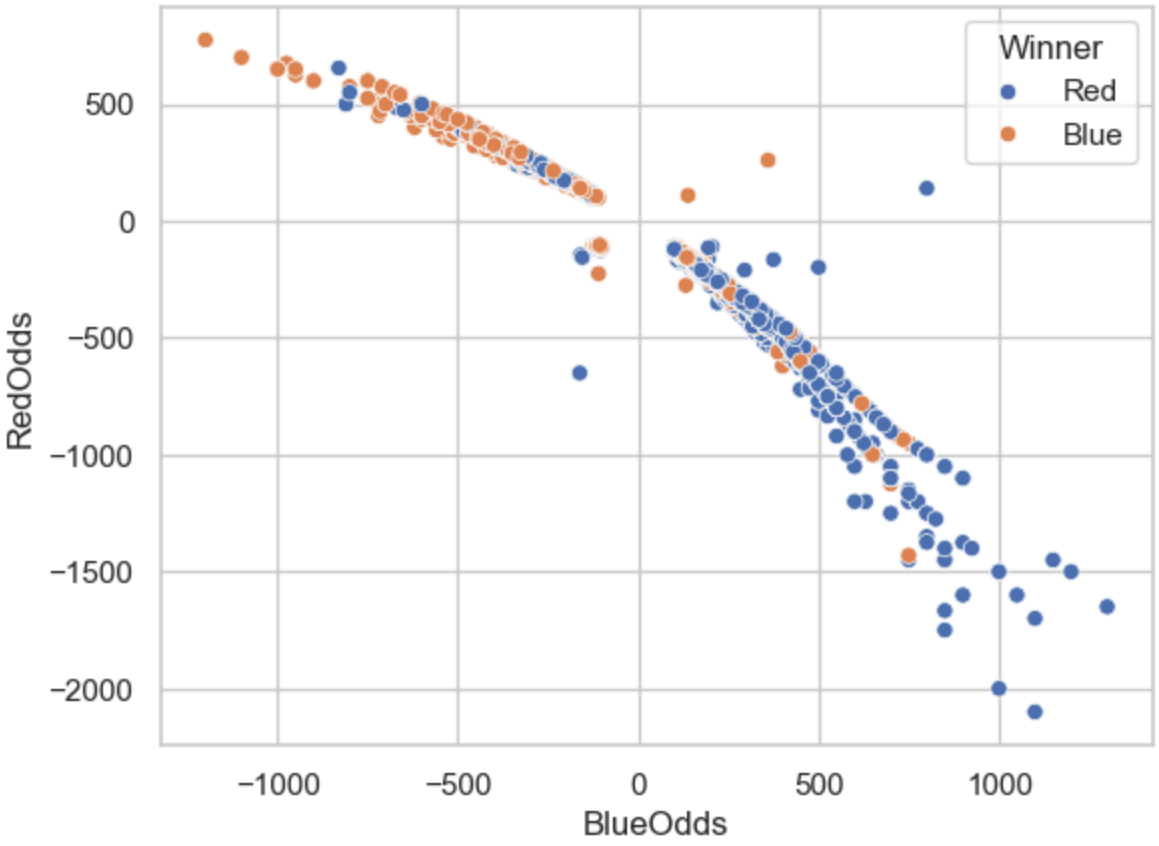
In [267…

```
#remove possible leading and trailing spaces in all 'object' columns
ufc_obj = df.select_dtypes(['object'])
df[ufc_obj.columns] = ufc_obj.apply(lambda x: x.str.strip()) # Apply over every column function lambda
#categorical
print(df['Finish'].value_counts())
#filtering
kos_by_round = df[['Finish', 'FinishRound']].query('Finish == "KO/TKO"') # Create new filtered dataframe
# kos_by_round
```

```
Finish
U-DEC         2404
KO/TKO        2009
SUB           1157
S-DEC          654
M-DEC           46
DQ              18
Overturned       2
Name: count, dtype: int64
```

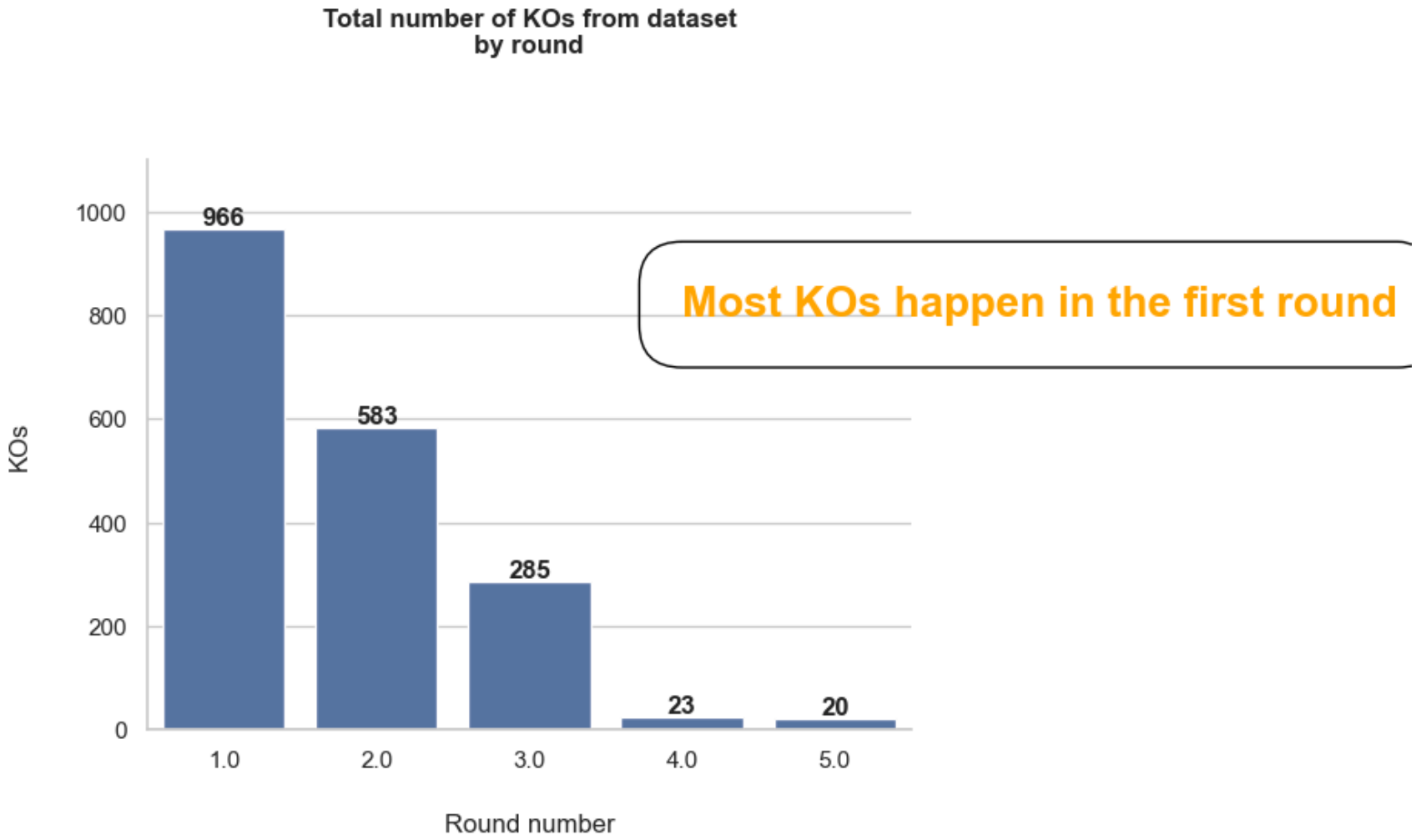
In [268…
```
#Analysing "_odds" variables
sns.scatterplot(x="BlueOdds", y="RedOdds", hue="Winner", data = df)
df["Winner"].loc[df["BlueOdds"]>1].value_counts()
```

Out[268…
```
Winner
Red     2659
Blue    1159
Name: count, dtype: int64
```




In [269…

```
KOs = sns.countplot(x = kos_by_round['FinishRound']);
plt.title('Total number of KOs from dataset\nby round', pad = 50, weight = 'bold') # \n for line break. S:
plt.xlabel('Round number', labelpad = 20) # Label weight set to bold in general settings
plt.ylabel('KOs', labelpad = 20)
sns.despine() # Remove top and right border
plt.ylim([0,1100])
plt.bar_label(KOs.containers[0], weight = 'bold') # Add number labels on top of bars
plt.text(x = 3, y = 800, s = 'Most KOs happen in the first round', fontdict = {'size' : 20, 'weight' : 'bo
```

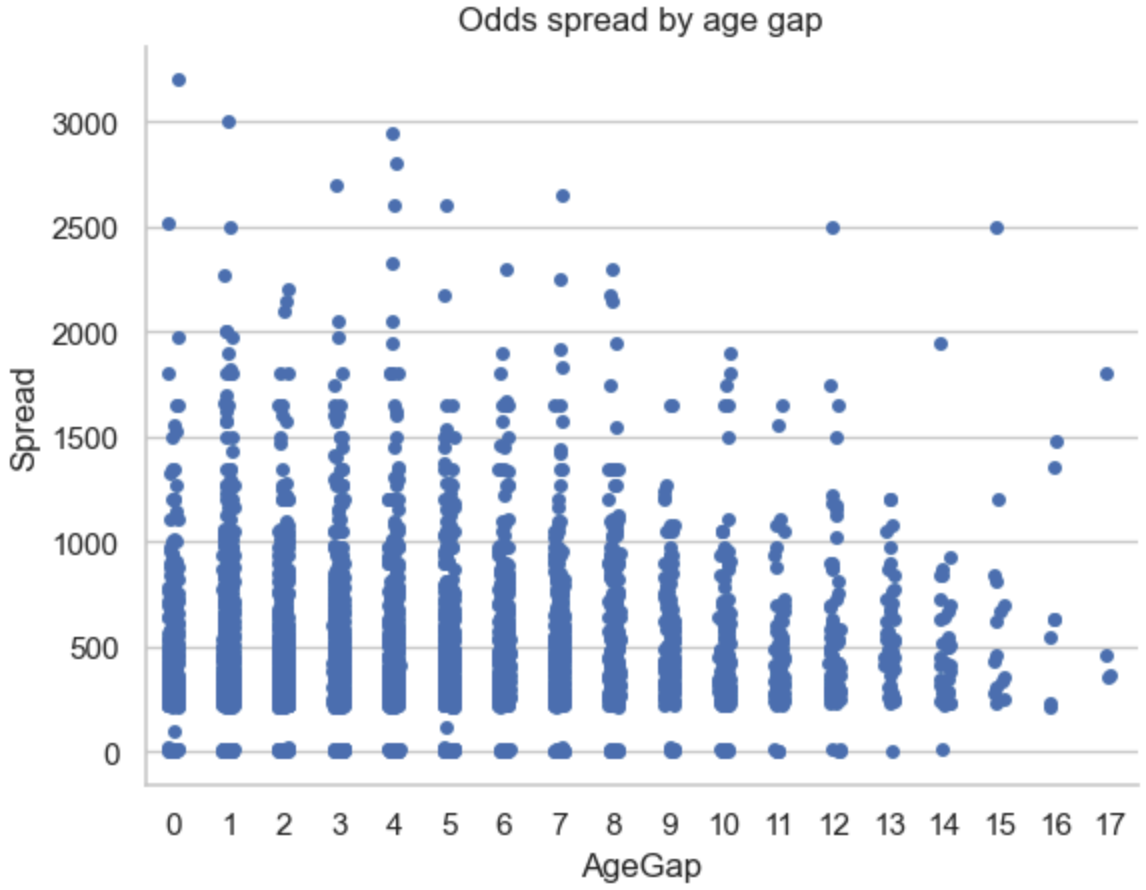



In [270…
```python
# Calculate the age gap
df['AgeGap'] = abs(df['RedAge'] - df['BlueAge'])
# df[['AgeGap', 'RedAge', 'BlueAge']]
```

In [271…
```python
# Create a function to calculate the spread as positive number
def spread_calculation(RedOdds, BlueOdds):
    if RedOdds > 0 and BlueOdds > 0:
        return max(RedOdds, BlueOdds) - min(RedOdds, BlueOdds)
    else:
        return abs(RedOdds - BlueOdds)
```

```
# Create column that holds the spread
df['Spread'] = df.apply(lambda row: spread_calculation(row['RedOdds'], row['BlueOdds']), axis=1)

# Check results
# df[['RedOdds', 'BlueOdds','Spread']]
```

In [272…
```
sns.stripplot(data = df, x = 'AgeGap', y = 'Spread').set(title = 'Odds spread by age gap')
sns.despine()
```




In [273…
```
# Create a frequency table
win_counts = df['Winner'].value_counts()
# Display the frequency table
```

```
print("Frequency Table for 'Winner' column:")
print(win_counts)
```

```
Frequency Table for 'Winner' column:
Winner
Red     3787
Blue    2741
Name: count, dtype: int64
```

In [274…

```
# To get proportions instead of counts
check_win = df['Winner'].value_counts(normalize=True)
print("\nProportion Table for 'Winner' column:")
print(check_win)
```

```
Proportion Table for 'Winner' column:
Winner
Red     0.580116
Blue    0.419884
Name: proportion, dtype: float64
```

In [275…

```
print(df.dtypes)
# How many rows and columns do we have in df_merge?
print(df.shape[0]) # number of rows
print(df.shape[1]) # number of rows

print(len(df))
df.head()
```

```
RedFighter                      int64
BlueFighter                     int64
RedOdds                       float64
BlueOdds                      float64
RedExpectedValue              float64
BlueExpectedValue             float64
Winner                         object
TitleBout                        bool
WeightClass                     int64
Gender                          int64
NumberOfRounds                  int64
BlueCurrentLoseStreak           int64
BlueCurrentWinStreak            int64
BlueDraws                       int64
BlueAvgSigStrLanded           float64
BlueAvgSigStrPct              float64
BlueAvgSubAtt                 float64
BlueAvgTDLanded               float64
BlueAvgTDPct                  float64
BlueLongestWinStreak            int64
BlueLosses                      int64
BlueTotalRoundsFought           int64
BlueTotalTitleBouts             int64
BlueWinsByDecisionMajority      int64
BlueWinsByDecisionSplit         int64
BlueWinsByDecisionUnanimous     int64
BlueWinsByKO                    int64
BlueWinsBySubmission            int64
BlueWinsByTKODoctorStoppage     int64
BlueWins                        int64
BlueStance                     object
BlueWeightLbs                   int64
RedCurrentLoseStreak            int64
RedCurrentWinStreak             int64
RedDraws                        int64
RedAvgSigStrLanded            float64
RedAvgSigStrPct               float64
RedAvgSubAtt                  float64
RedAvgTDLanded                float64
RedAvgTDPct                   float64
RedLongestWinStreak             int64
RedLosses                       int64
```

```
RedTotalRoundsFought               int64
RedTotalTitleBouts                 int64
RedWinsByDecisionMajority          int64
RedWinsByDecisionSplit             int64
RedWinsByDecisionUnanimous         int64
RedWinsByKO                        int64
RedWinsBySubmission                int64
RedWinsByTKODoctorStoppage         int64
RedWins                            int64
RedStance                         object
RedWeightLbs                       int64
RedAge                             int64
BlueAge                            int64
LoseStreakDif                      int64
WinStreakDif                       int64
LongestWinStreakDif                int64
WinDif                             int64
LossDif                            int64
TotalRoundDif                      int64
TotalTitleBoutDif                  int64
KODif                              int64
SubDif                             int64
HeightDif                        float64
ReachDif                         float64
AgeDif                             int64
SigStrDif                        float64
AvgSubAttDif                     float64
AvgTDDif                         float64
EmptyArena                       float64
BMatchWCRank                     float64
RMatchWCRank                     float64
RWFlyweightRank                  float64
RWFeatherweightRank              float64
RWStrawweightRank                float64
RWBantamweightRank               float64
RHeavyweightRank                 float64
RLightHeavyweightRank            float64
RMiddleweightRank                float64
RWelterweightRank                float64
RLightweightRank                 float64
RFeatherweightRank               float64
RBantamweightRank                float64
```

```
RFlyweightRank              float64
RPFPRank                    float64
BWFlyweightRank             float64
BWFeatherweightRank         float64
BWStrawweightRank           float64
BWBantamweightRank          float64
BHeavyweightRank            float64
BLightHeavyweightRank       float64
BMiddleweightRank           float64
BWelterweightRank           float64
BLightweightRank            float64
BFeatherweightRank          float64
BBantamweightRank           float64
BFlyweightRank              float64
BPFPRank                    float64
BetterRank                   object
Finish                       object
FinishDetails                object
FinishRound                 float64
FinishRoundTime              object
TotalFightTimeSecs          float64
RedDecOdds                  float64
BlueDecOdds                 float64
RSubOdds                    float64
BSubOdds                    float64
RKOOdds                     float64
BKOOdds                     float64
WeightClassLabel             object
BlueHeight                  float64
BlueReach                   float64
RedHeight                   float64
RedReach                    float64
ROver35                       int64
BOver35                       int64
RWinPct                     float64
BWinPct                     float64
RedStrikingRatio            float64
BlueStrikingRatio           float64
RedTotalFights                int64
BlueTotalFights               int64
RedIsGrappler                 int64
BlueIsGrappler                int64
```

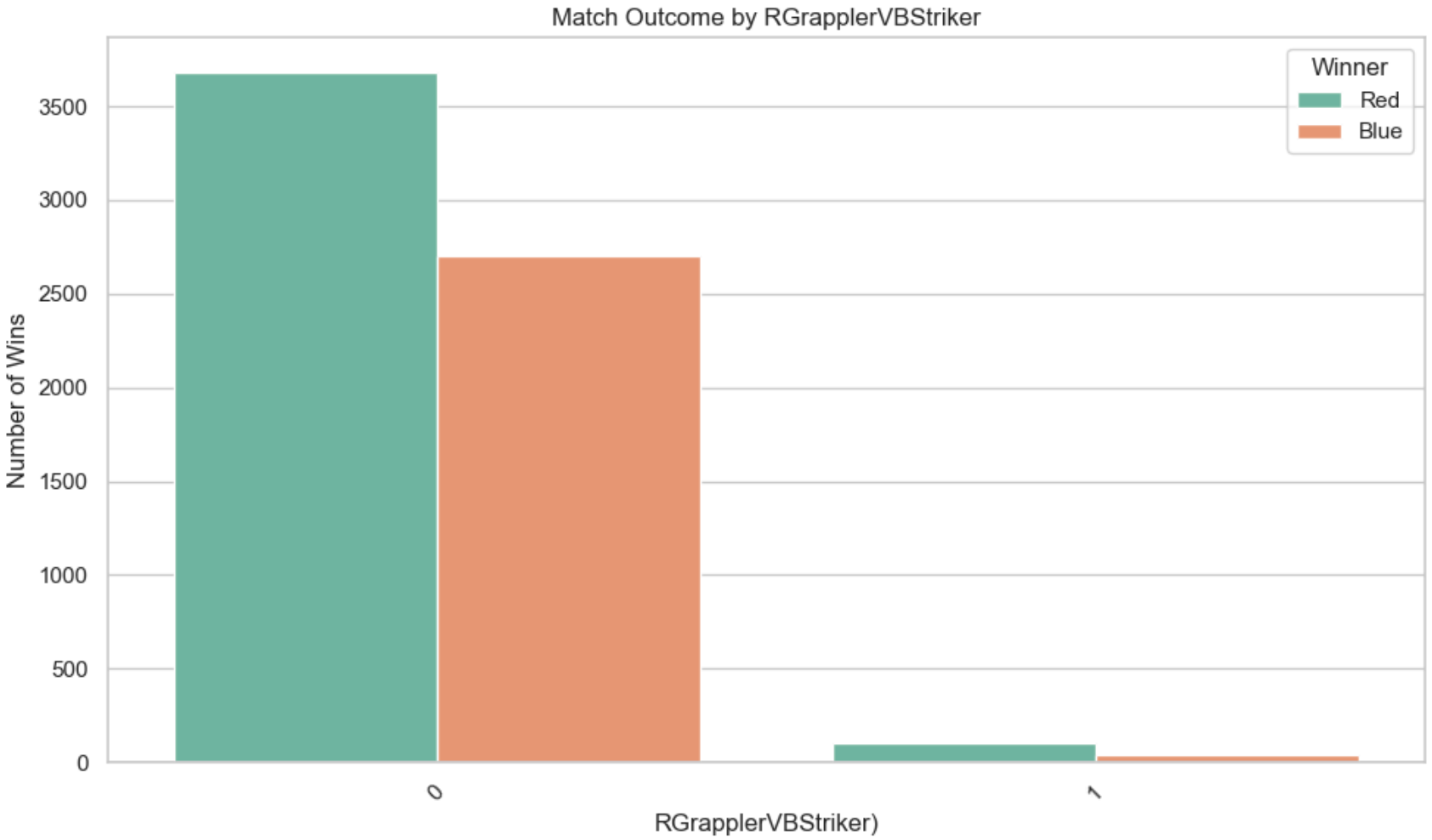
```
RedIsStriker                        int64
BlueIsStriker                       int64
RGrapplerVBStriker                  int64
BGrapplerVRStriker                  int64
RedStrikingEfficiency             float64
BlueStrikingEfficiency            float64
RedTDEfficiency                   float64
BlueTDEfficiency                  float64
RedEffectiveTD                    float64
BlueEffectiveTD                   float64
EffectiveTDDif                    float64
RedSize                           float64
BlueSize                          float64
Favorite                           object
FavoriteWins                        int64
draw_diff                           int64
avg_sig_str_pct_diff              float64
avg_TD_pct_diff                   float64
win_by_Decision_Majority_diff       int64
win_by_Decision_Split_diff          int64
win_by_Decision_Unanimous_diff      int64
win_by_TKO_Doctor_Stoppage_diff     int64
odds_diff                         float64
ev_diff                           float64
StanceCombo                        object
AgeGap                              int64
Spread                            float64
dtype: object
6528
153
6528
```

Out [275…

| | RedFighter | BlueFighter | RedOdds | BlueOdds | RedExpectedValue | BlueExpectedValue | Winner | TitleBout | WeightClass |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 66 | 1009 | -250.0 | 215.0 | 40.0000 | 215.0 | Red | True | 3 |
| **1** | 1441 | 718 | -210.0 | 295.0 | 47.6190 | 295.0 | Red | False | 8 |
| **2** | 307 | 67 | -380.0 | 300.0 | 26.3158 | 300.0 | Red | False | 4 |
| **3** | 221 | 1071 | -950.0 | 625.0 | 10.5263 | 625.0 | Red | False | 2 |
| **4** | 1183 | 524 | -130.0 | 110.0 | 76.9231 | 110.0 | Blue | False | 2 |

In [276…

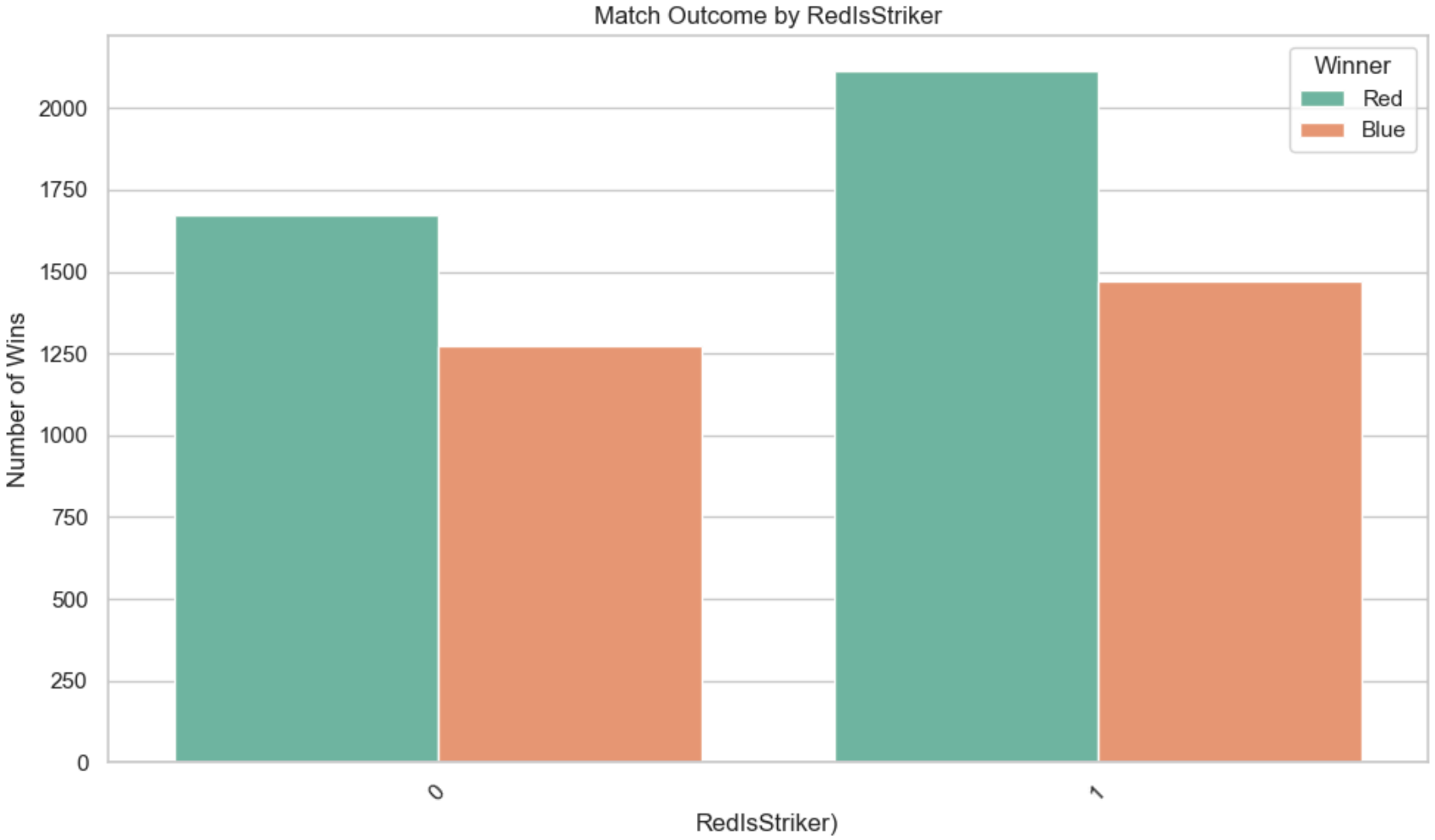
```
# Plot: Count of Winners per Matchup type
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='RGrapplerVBStriker', hue='Winner', palette='Set2')
plt.title('Match Outcome by RGrapplerVBStriker')
plt.xlabel('RGrapplerVBStriker)')
plt.ylabel('Number of Wins')
plt.xticks(rotation=45)
plt.tight_layout()
fig_name = "grappler_v_striker"
fig_path = os.path.join(output_folder, f"{fig_name}.png")
plt.savefig(fig_path, bbox_inches="tight", dpi=300)
plt.show()
plt.close()
```




In [277…

```
# Create a new combined column for Red vs Blue matchup
#df['Matchup'] = df['RGrapplerVBStriker'].astype(str)

# Plot: Count of Winners per Matchup type
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='RedIsStriker', hue='Winner', palette='Set2')
plt.title('Match Outcome by RedIsStriker')
plt.xlabel('RedIsStriker)')
plt.ylabel('Number of Wins')
plt.xticks(rotation=45)
plt.tight_layout()
```

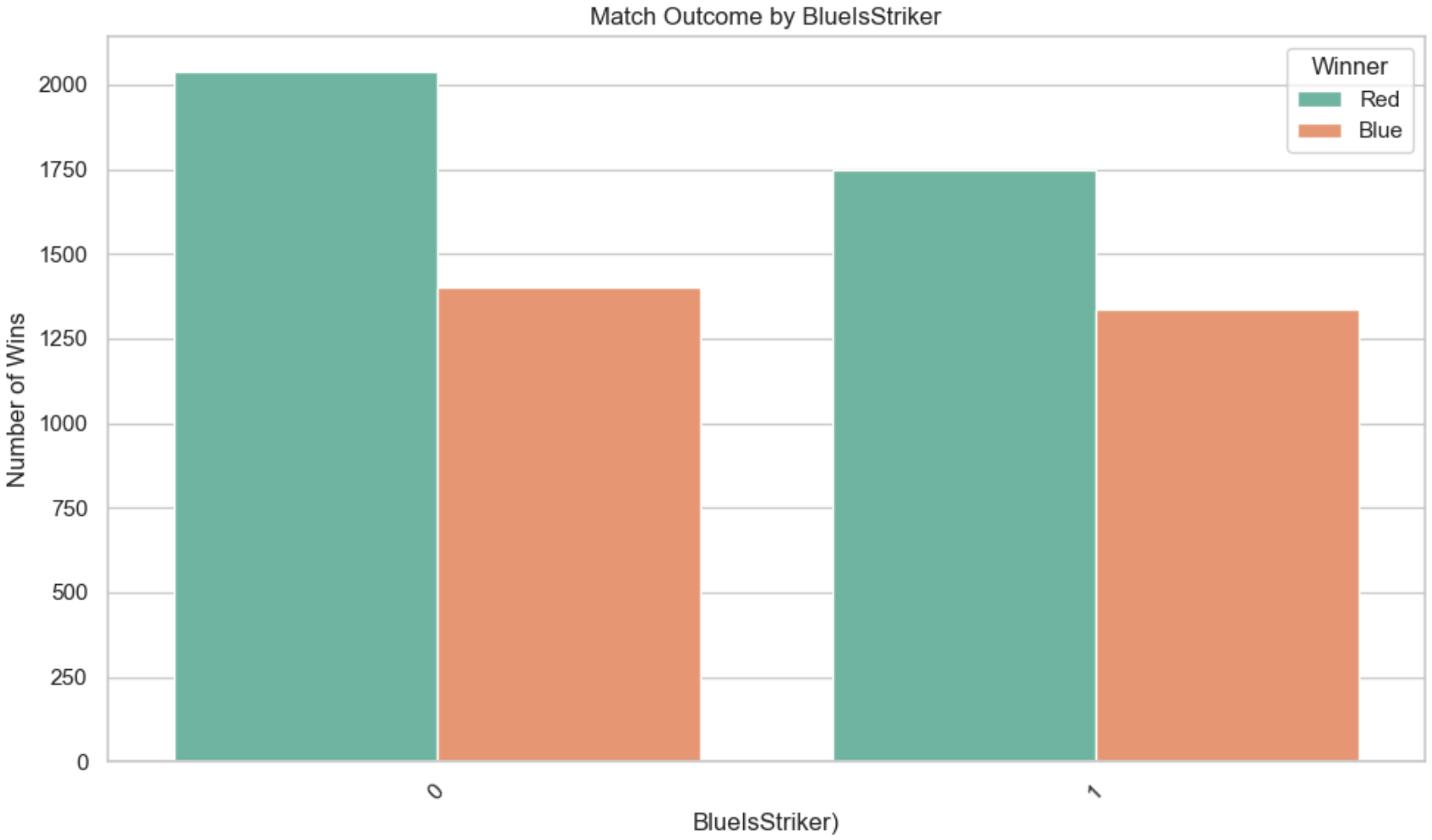
```
fig_name = "redstriker"
fig_path = os.path.join(output_folder, f"{fig_name}.png")
plt.savefig(fig_path, bbox_inches="tight", dpi=300)
plt.show()
plt.close()
```




In [278…

```
# Plot: Count of Winners per Matchup type
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='BlueIsStriker', hue='Winner', palette='Set2')
plt.title('Match Outcome by BlueIsStriker')
plt.xlabel('BlueIsStriker)')
```

```
plt.ylabel('Number of Wins')
plt.xticks(rotation=45)
plt.tight_layout()
fig_name = "bluestriker"
fig_path = os.path.join(output_folder, f"{fig_name}.png")
plt.savefig(fig_path, bbox_inches="tight", dpi=300)
plt.show()
plt.close()
```




In [279…

```
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
```

```
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder, OrdinalEncoder, LabelEncoder
from sklearn.compose import ColumnTransformer
import numpy as np



# Convert TitleBout to string "True"/"False" for encoding
if 'TitleBout' in df.columns:
    df['TitleBout'] = df['TitleBout'].astype(str)


categorical_nominal = [
    'Country', 'BlueStance', 'RedStance', 'BetterRank',
    'Finish', 'FinishDetails', 'FinishRoundTime',
    'Favorite', 'StanceCombo', 'TitleBout', "WeightClass"
]


numeric_features = [
    'RedOdds', 'BlueOdds', 'RedExpectedValue', 'BlueExpectedValue',
    'NumberOfRounds', 'BlueCurrentLoseStreak', 'BlueCurrentWinStreak', 'BlueDraws',
    'BlueAvgSigStrLanded', 'BlueAvgSigStrPct', 'BlueAvgSubAtt', 'BlueAvgTDLanded', 'BlueAvgTDPct',
    'BlueLongestWinStreak', 'BlueLosses', 'BlueTotalRoundsFought', 'BlueTotalTitleBouts',
    'BlueWinsByDecisionMajority', 'BlueWinsByDecisionSplit', 'BlueWinsByDecisionUnanimous',
    'BlueWinsByKO', 'BlueWinsBySubmission', 'BlueWinsByTKODoctorStoppage', 'BlueWins',
    'BlueWeightLbs', 'BlueAge',

    'RedCurrentLoseStreak', 'RedCurrentWinStreak', 'RedDraws',
    'RedAvgSigStrLanded', 'RedAvgSigStrPct', 'RedAvgSubAtt', 'RedAvgTDLanded', 'RedAvgTDPct',
    'RedLongestWinStreak', 'RedLosses', 'RedTotalRoundsFought', 'RedTotalTitleBouts',
    'RedWinsByDecisionMajority', 'RedWinsByDecisionSplit', 'RedWinsByDecisionUnanimous',
    'RedWinsByKO', 'RedWinsBySubmission', 'RedWinsByTKODoctorStoppage', 'RedWins',
    'RedWeightLbs', 'RedAge',

    'LoseStreakDif', 'WinStreakDif', 'LongestWinStreakDif', 'WinDif', 'LossDif', 'TotalRoundDif', 'TotalTi
    'KODif', 'SubDif', 'HeightDif', 'ReachDif', 'AgeDif', 'SigStrDif', 'AvgSubAttDif', 'AvgTDDif',
    'EmptyArena',

    'BMatchWCRank', 'RMatchWCRank',
    'RWFlyweightRank', 'RWFeatherweightRank', 'RWStrawweightRank', 'RWBantamweightRank', 'RHeavyweightRank
    'RLightHeavyweightRank', 'RMiddleweightRank', 'RWelterweightRank', 'RLightweightRank', 'RFeatherweightI
```

```
    'RBantamweightRank', 'RFlyweightRank', 'RPFPRank',
    'BWFlyweightRank', 'BWFeatherweightRank', 'BWStrawweightRank', 'BWBantamweightRank', 'BHeavyweightRank
    'BLightHeavyweightRank', 'BMiddleweightRank', 'BWelterweightRank', 'BLightweightRank', 'BFeatherweightR
    'BBantamweightRank', 'BFlyweightRank', 'BPFPRank',

    'FinishRound', 'TotalFightTimeSecs',
    'RedDecOdds', 'BlueDecOdds', 'RSubOdds', 'BSubOdds', 'RKOOdds', 'BKOOdds',

    'BlueHeight', 'BlueReach', 'RedHeight', 'RedReach',

    'ROver35', 'BOver35',

    'RWinPct', 'BWinPct',
    'RedStrikingRatio', 'BlueStrikingRatio',
    'RedTotalFights', 'BlueTotalFights',
    'RedIsGrappler', 'BlueIsGrappler', 'RedIsStriker', 'BlueIsStriker',
    'RGrapplerVBStriker', 'BGrapplerVRStriker',
    'RedStrikingEfficiency', 'BlueStrikingEfficiency',
    'RedTDEfficiency', 'BlueTDEfficiency',
    'RedEffectiveTD', 'BlueEffectiveTD',
    'EffectiveTDDif',
    'RedSize', 'BlueSize',
    'FavoriteWins', 'draw_diff',
    'avg_sig_str_pct_diff', 'avg_TD_pct_diff',
    'win_by_Decision_Majority_diff', 'win_by_Decision_Split_diff', 'win_by_Decision_Unanimous_diff',
    'win_by_TKO_Doctor_Stoppage_diff', 'odds_diff', 'ev_diff'
]

# Replace infinite values with NaN in numeric features ---
df[numeric_features] = df[numeric_features].replace([np.inf, -np.inf], np.nan)


# numeric_transformer = Pipeline([
#     ('imputer', SimpleImputer(strategy='mean')),
#     ('scaler', StandardScaler())
# ])

# categorical_nominal_transformer = Pipeline([
#     ('imputer', SimpleImputer(strategy='most_frequent')),
#     ('onehot', OneHotEncoder(sparse_output=False, handle_unknown='ignore'))
# ])
```

```
# categorical_ordered_transformer = Pipeline([
#     ('imputer', SimpleImputer(strategy='most_frequent')),
#     # ('ordinal', OrdinalEncoder(categories=ordinal_cats))
# ])

# preprocessor = ColumnTransformer(
#     transformers=[
#         ('num', numeric_transformer, numeric_features),
#         ('cat', categorical_nominal_transformer, categorical_nominal)
#     ])


# prep = Pipeline(steps=[('preprocessor', preprocessor)])
```

In [280…

```
print(df.dtypes)
print(df.shape[0]) # number of rows
print(df.shape[1]) # number of rows
print(len(df))
df.head()
for column in df.columns:
    unique_values = df[column].unique()
    print(f"Unique values in column '{column}': {unique_values}")
    print(f"Number of unique values in column '{column}': {len(unique_values)}\n")
```

```
RedFighter                    int64
BlueFighter                   int64
RedOdds                     float64
BlueOdds                    float64
RedExpectedValue            float64
BlueExpectedValue           float64
Winner                       object
TitleBout                    object
WeightClass                   int64
Gender                        int64
NumberOfRounds                int64
BlueCurrentLoseStreak         int64
BlueCurrentWinStreak          int64
BlueDraws                     int64
BlueAvgSigStrLanded         float64
BlueAvgSigStrPct            float64
BlueAvgSubAtt               float64
BlueAvgTDLanded             float64
BlueAvgTDPct                float64
BlueLongestWinStreak          int64
BlueLosses                    int64
BlueTotalRoundsFought         int64
BlueTotalTitleBouts           int64
BlueWinsByDecisionMajority    int64
BlueWinsByDecisionSplit       int64
BlueWinsByDecisionUnanimous   int64
BlueWinsByKO                  int64
BlueWinsBySubmission          int64
BlueWinsByTKODoctorStoppage   int64
BlueWins                      int64
BlueStance                   object
BlueWeightLbs                 int64
RedCurrentLoseStreak          int64
RedCurrentWinStreak           int64
RedDraws                      int64
RedAvgSigStrLanded          float64
RedAvgSigStrPct             float64
RedAvgSubAtt                float64
RedAvgTDLanded              float64
RedAvgTDPct                 float64
RedLongestWinStreak           int64
RedLosses                     int64
```

```
RedTotalRoundsFought               int64
RedTotalTitleBouts                 int64
RedWinsByDecisionMajority          int64
RedWinsByDecisionSplit             int64
RedWinsByDecisionUnanimous         int64
RedWinsByKO                        int64
RedWinsBySubmission                int64
RedWinsByTKODoctorStoppage         int64
RedWins                            int64
RedStance                         object
RedWeightLbs                       int64
RedAge                             int64
BlueAge                            int64
LoseStreakDif                      int64
WinStreakDif                       int64
LongestWinStreakDif                int64
WinDif                             int64
LossDif                            int64
TotalRoundDif                      int64
TotalTitleBoutDif                  int64
KODif                              int64
SubDif                             int64
HeightDif                        float64
ReachDif                         float64
AgeDif                             int64
SigStrDif                        float64
AvgSubAttDif                     float64
AvgTDDif                         float64
EmptyArena                       float64
BMatchWCRank                     float64
RMatchWCRank                     float64
RWFlyweightRank                  float64
RWFeatherweightRank              float64
RWStrawweightRank                float64
RWBantamweightRank               float64
RHeavyweightRank                 float64
RLightHeavyweightRank            float64
RMiddleweightRank                float64
RWelterweightRank                float64
RLightweightRank                 float64
RFeatherweightRank               float64
RBantamweightRank                float64
```

```
RFlyweightRank              float64
RPFPRank                    float64
BWFlyweightRank             float64
BWFeatherweightRank         float64
BWStrawweightRank           float64
BWBantamweightRank          float64
BHeavyweightRank            float64
BLightHeavyweightRank       float64
BMiddleweightRank           float64
BWelterweightRank           float64
BLightweightRank            float64
BFeatherweightRank          float64
BBantamweightRank           float64
BFlyweightRank              float64
BPFPRank                    float64
BetterRank                   object
Finish                       object
FinishDetails                object
FinishRound                 float64
FinishRoundTime              object
TotalFightTimeSecs          float64
RedDecOdds                  float64
BlueDecOdds                 float64
RSubOdds                    float64
BSubOdds                    float64
RKOOdds                     float64
BKOOdds                     float64
WeightClassLabel             object
BlueHeight                  float64
BlueReach                   float64
RedHeight                   float64
RedReach                    float64
ROver35                       int64
BOver35                       int64
RWinPct                     float64
BWinPct                     float64
RedStrikingRatio            float64
BlueStrikingRatio           float64
RedTotalFights                int64
BlueTotalFights               int64
RedIsGrappler                 int64
BlueIsGrappler                int64
```

```
RedIsStriker                        int64
BlueIsStriker                       int64
RGrapplerVBStriker                  int64
BGrapplerVRStriker                  int64
RedStrikingEfficiency             float64
BlueStrikingEfficiency            float64
RedTDEfficiency                   float64
BlueTDEfficiency                  float64
RedEffectiveTD                    float64
BlueEffectiveTD                   float64
EffectiveTDDif                    float64
RedSize                           float64
BlueSize                          float64
Favorite                           object
FavoriteWins                        int64
draw_diff                           int64
avg_sig_str_pct_diff              float64
avg_TD_pct_diff                   float64
win_by_Decision_Majority_diff       int64
win_by_Decision_Split_diff          int64
win_by_Decision_Unanimous_diff      int64
win_by_TKO_Doctor_Stoppage_diff     int64
odds_diff                         float64
ev_diff                           float64
StanceCombo                        object
AgeGap                              int64
Spread                            float64
dtype: object
6528
153
6528
Unique values in column 'RedFighter': [  66 1441  307 ...  237  487  499]
Number of unique values in column 'RedFighter': 1661

Unique values in column 'BlueFighter': [1009  718   67 ...  781 1667  313]
Number of unique values in column 'BlueFighter': 1922

Unique values in column 'RedOdds': [ -250.  -210.  -380.  -950.  -130.  -650.  -238.   150.  -112.   142.
   700.  -550.  -305.  -198.    nan   215.  -225.   190.   102.   380.
  -310.  -144.  -245.  -265. -1350.   235. -1000.  -105.   195. -1450.
  -170.   250.   550.   205.   360.  -485.  -180.  -155.  -192.  -270.
  -375.   200.  -340.  -166.  -108.  -800.  -575.  -148.  -218.  -162.
```

```
  455.   170.  -258.   330.   240.  -290.   124.  -425.   185.   154.
 -500.   210.   245.   160.  -520.  -125.  -102.  -625.  -360.   105.
  100.  -205.   110.  -535.   225. -2100.   140.   220.  -298.   120.
  260.  -142.   600.  -325.   180.  -900.  -120.  -445.   320.   114.
 -135. -1050.   152.  -720.   280.  -345.  -140.  -160.  -230.  -475.
 -220.  -110.  -175.  -188.  -410.  -115.  -145.  -200. -1750.  -810.
 -395.   230.   450.   136.  -215.  -285.   130.  -122.  -152.  -118.
  390.   295.   145.  -455.  -440.  -185. -1600.   350.   285.   370.
  410.   164.   270.   300.   400.   625.  -355.   500.  -620.  -470.
 -195.   175.  -920.  -350.  -700.  -315.   290.   115.  -240.   125.
 -590.  -104.   126.   188.  -300.   128.  -116.   310.  -172.  -260.
 -138.  -275.  -164.  -124. -1200.   132.  -190.   122.  -235.   166.
 -335.  -158.   106.  -146.  -255.  -174.   116.  -165.  -132.   104.
  184.   194.  -490.  -420.  -156.   108.  -184.  -450.   148.  -390.
  138.   112.   440.  -196.   134.   146.   176.  -126.   172.  -560.
 -850.  -530.  -168.  -600. -1400.  -750.   155.  -320.  -365.   255.
-1250.   165.   675.  -540.  -150.   135.  -280.   340. -1150.  -580.
 -645.  -295.  -330.   265.   275.  -460.   520.  -610.  -435.   650.
  575.  -630.  -760.  -400. -2000.   385.   375.  -525.   435. -1125.
 -675. -1375.  -129.  -107.  -106.  -143.  -136.  -177.  -134.   117.
 -114.  -137.  -178.   127.  -127.  -147.  -117.   157.   178.  -370.
 -430.  -157.  -278.  -253.  -487.  -447.  -103.   325.  -139.  -167.
 -286. -1667.  -835.  -770.   315.   119.   277.  -159.   133.   321.
 -480.   111.  -109.  -186.   470.  -182.  -385.   306. -1430.   131.
 -141.   271.  -715.   168.  -570.  -870.  -133.   655.   129.  -641.
 -181.  -151.  -303.  -640. -1100.  -670.   525.   415.   158.  -515.
  365.   425. -1500.  -840.  -680.   335.   305.   173.   243.   345.
  485. -1165.   475.   775. -1700. -1275.  -980.  -660.  -705.  -925.
 -815.  -975.   540.   167.   505.  -860.   420.  -740.  -405.   480.
 -605.  -510.  -690.   149.  -910.  -935.   460.  -545.   423.  -735.
  177.   355.   248.  -875.  -161.  -131.  -615.  -710.   211.  -171.
 -121. -1650.  -415.  -153.   123.   101.  -191.  -176.   118.  -173.
  121.   163.  -752.   181.  -119.   107.  -780.]
Number of unique values in column 'RedOdds': 377

Unique values in column 'BlueOdds': [  215.   295.   300.   625.   110.  -162.   195.  -180.  -108.  -170.
 -1100.   410.   245.   164.  -230.   750.  -265.   210.  -122.  -490.
   250.   136.   200.   800.  -290.   650.  -115.  -238.   850.   142.
  -310.    nan  -800.  -250.  -470.   370.   150.   130.   160.   220.
  -245.   270.   140.  -112.   550.   425.   124.   180.  -625.  -205.
  -425.  -298.   235.  -148.   330.  -225.  -185.   380.  -258.   185.
  -305.  -192.   390.   105.  -118.   455.   285.  -125.  -120.   170.
```

```
  -130.   400.  -278.  1100.  -166.  -270.   240.  -142.  -325.   120.
  -900.   260.  -218.   600.   100.   310.  -260.  -410.  -145.  -160.
   450.  -395.  -540.  -345.   175.  -110.  -195.   325.   145.   320.
  -105.   114.   154.   500.   205.   290.  -285.  -720.   165.   230.
   102.  -150.   128.  -102.  -520.  -355.  -200.  -280.  -375.  -175.
  -135.   350.   900.  -455.  -155.  -360.  -485.  -550.   190.  -198.
  -340.  -380.  -620.  -950.   280.  -810.   360.  -500.   340.   188.
  -535.   125.  -215.  -240.  -190.   135.   430.  -154.   168.  -405.
   176.  -255.   460.  -152.  -390.   144.   196.   108.   132.   106.
  -156.  -144.   194.   184.   134.  -124.  -275.  -116.   152.   146.
  -136.   112.  -220.  -235.   122.   630.  -510.  -300.   198.   660.
  -168.  -176.   138.  -164.  -132.  -590.  -158.  -174.  -210.  -196.
  -134.  -184.   420.  -106.   590.   186.  -146.   115.  -295.   265.
   155.  -350.   225.  -600.   255.  -975.  -365.   700.   510.  -165.
  -435.  -675.   440.   480.   275.  -320.  -330.  -140.  -315.  -475.
   525.   435.  1000.  -525.   385.  -630. -1000.  -335.  -400.   475.
   575.   580.  -186.   162.   153.   113.   187.  -139.  -167.   355.
  -114.  -177.   375.   117.   157.  -147.   107.   127.  -103.   335.
  -137.   178.   345.   415.   365.   137.   228.  -107.   174.  -129.
  -182.  -117.  -143.  -560.  -159.  -420.   356.   277.   296.   131.
   169.   317.  -141.   103.   342.   246.   151.   253.  -715.   233.
   286.   227.   315.   239.  -109.   387.   333.   148.   161.   129.
   305.   635.   485.   158.   585.   610.  -830.  -151.   484.   159.
   234.   470.  -750.   328.  -460.  -570.  -370.   445.   465.   775.
   570.   490.   313.  -440.  -430.   243.   177.  -670.  -640.  -450.
   163.  1200.  -650.   925.  -700.   383. -1200.   825.   640.  1050.
   540.  -710.   725.   248.   645.  -660.  -178.  -605.   675.   535.
   595.   167.   568.  1150.   123.   505.  -126.  -179.   560.   166.
   405.   710.   735.   423.  -480.   318.   555.   378.   685.   121.
   126.  -101.   258.  -231.   111.  1300.   133.   143.  -133.   680.
   181.  -111.  -530.   348.  -128.  -131.  -445.  -173.  -193.   602.
   109.   620.]
Number of unique values in column 'BlueOdds': 382

Unique values in column 'RedExpectedValue': [ 40.      47.619   26.3158  10.5263  76.9231  15.3846  42.0168
150.
  89.2857 142.     700.      18.1818  32.7869  50.5051      nan 215.
  44.4444 190.     102.     380.      32.2581  69.4444  40.8163  37.7358
   7.4074 235.      10.      95.2381 195.       6.8966  58.8235 250.
 550.     205.     360.      20.6186  55.5556  64.5161  52.0833  37.037
  26.6667 200.      29.4118  60.241   92.5926  12.5     17.3913  67.5676
  45.8716  61.7284 455.     170.      38.7597 330.     240.      34.4828
```

```
 124.      23.5294 185.     154.      20.     210.     245.     160.
  19.2308  80.      98.0392  16.      27.7778 105.     100.      48.7805
 110.      18.6916 225.       4.7619 140.     220.      33.557  120.
 260.      70.4225 600.      30.7692 180.      11.1111  83.3333  22.4719
 320.     114.      74.0741   9.5238 152.      13.8889 280.      28.9855
  71.4286  62.5     43.4783  21.0526  45.4545  90.9091  57.1429  53.1915
  24.3902  86.9565  68.9655  50.       5.7143  12.3457  25.3165 230.
 450.     136.      46.5116  35.0877 130.      81.9672  65.7895  84.7458
 390.     295.     145.      21.978   22.7273  54.0541   6.25   350.
 285.     370.     410.     164.     270.     300.     400.     625.
  28.169  500.      16.129   21.2766  51.2821 175.      10.8696  28.5714
  14.2857  31.746  290.     115.      41.6667 125.      16.9492  96.1538
 126.     188.      33.3333 128.      86.2069 310.      58.1395  38.4615
  72.4638  36.3636  60.9756  80.6452   8.3333 132.      52.6316 122.
  42.5532 166.      29.8507  63.2911 106.      68.4932  39.2157  57.4713
 116.      60.6061  75.7576 104.     184.     194.      20.4082  23.8095
  64.1026 108.      54.3478  22.2222 148.      25.641  138.     112.
 440.      51.0204 134.     146.     176.      79.3651 172.      17.8571
  11.7647  18.8679  59.5238  16.6667   7.1429  13.3333 155.      31.25
  27.3973 255.       8.     165.     675.      18.5185  66.6667 135.
  35.7143 340.       8.6957  17.2414  15.5039  33.8983  30.303  265.
 275.      21.7391 520.      16.3934  22.9885 650.     575.      15.873
  13.1579  25.       5.     385.     375.      19.0476 435.       8.8889
  14.8148   7.2727  77.5194  93.4579  94.3396  69.9301  73.5294  56.4972
  74.6269 117.      87.7193  72.9927  56.1798 127.      78.7402  68.0272
  85.4701 157.     178.      27.027   23.2558  66.      45.      63.
  35.9712  39.5257  20.5339  22.3714  97.0874  63.6943 325.      71.9424
  59.8802  34.965    5.9988  11.976   13.986  315.     119.     277.
  62.8931 133.     321.      20.8333 111.      91.7431  53.7634 470.
  54.9451  25.974  306.       6.993  131.      70.922   12.987  168.
  17.5439  11.4943  75.188  655.     129.      15.6006  55.2486  66.2252
  33.0033  15.625    9.0909  14.9254 525.     415.     158.      19.4175
 365.     425.       6.6667  11.9048  14.7059 335.     305.     173.
 243.     345.     485.       8.5837 475.     775.       5.8824   7.8431
  10.2041  15.1515  14.1844  10.8108  12.2699  10.2564 540.     167.
 505.      11.6279 420.      13.5135  24.6914 480.      16.5289  19.6078
  14.4928 149.      10.989   10.6952 460.      18.3486 423.      13.6054
 177.     355.     248.      11.4286  62.1118  76.3359  16.2602  14.0845
 211.      58.4795  82.6446   6.0606  24.0964  65.3595 123.     101.
  52.356   56.8182 118.      57.8035 121.     163.      13.2979 181.
  84.0336 107.      12.8205]
Number of unique values in column 'RedExpectedValue': 379
```

```
Unique values in column 'BlueExpectedValue': [ 215.      295.      300.      625.      110.       61.7284
195.
   55.5556   92.5926   58.8235    9.0909  410.      245.      164.
   43.4783  750.       37.7358  210.       81.9672   20.4082  250.
  136.      200.      800.       34.4828  650.       86.9565   42.0168
  850.      142.       32.2581       nan   12.5      40.       21.2766
  370.      150.      130.      160.      220.       40.8163  270.
  140.       89.2857  550.      425.      124.      180.       16.
   48.7805   23.5294   33.557   235.       67.5676  330.       44.4444
   54.0541  380.       38.7597  185.       32.7869   52.0833  390.
  105.       84.7458  455.      285.       80.       83.3333  170.
   76.9231  400.       35.9712 1100.       60.241    37.037   240.
   70.4225   30.7692  120.       11.1111  260.       45.8716  600.
  100.      310.       38.4615   24.3902   68.9655   62.5     450.
   25.3165   18.5185   28.9855  175.       90.9091   51.2821  325.
  145.      320.       95.2381  114.      154.      500.      205.
  290.       35.0877   13.8889  165.      230.      102.       66.6667
  128.       98.0392   19.2308   28.169    50.       35.7143   26.6667
   57.1429   74.0741  350.      900.       21.978    64.5161   27.7778
   20.6186   18.1818  190.       50.5051   29.4118   26.3158   16.129
   10.5263  280.       12.3457  360.       20.      340.      188.
   18.6916  125.       46.5116   41.6667   52.6316  135.      430.
   64.9351  168.       24.6914  176.       39.2157  460.       65.7895
   25.641   144.      196.      108.      132.      106.       64.1026
   69.4444  194.      184.      134.       80.6452   36.3636   86.2069
  152.      146.       73.5294  112.       45.4545   42.5532  122.
  630.       19.6078   33.3333  198.      660.       59.5238   56.8182
  138.       60.9756   75.7576   16.9492   63.2911   57.4713   47.619
   51.0204   74.6269   54.3478  420.       94.3396  590.      186.
   68.4932  115.       33.8983  265.      155.       28.5714  225.
   16.6667  255.       10.2564   27.3973  700.      510.       60.6061
   22.9885   14.8148  440.      480.      275.       31.25     30.303
   71.4286   31.746    21.0526  525.      435.     1000.       19.0476
  385.       15.873    10.       29.8507   25.      475.      575.
  580.       53.7634  162.      153.      113.      187.       71.9424
   59.8802  355.       87.7193   56.4972  375.      117.      157.
   68.0272  107.      127.       97.0874  335.       72.9927  178.
  345.      415.      365.       44.       33.       74.       46.
   54.      137.      228.       93.4579  174.       77.5194   54.9451
   85.4701   69.9301   17.8571   62.8931   23.8095  356.      277.
  296.      131.      169.      317.       70.922   103.      342.
```

```
  246.      151.      253.       13.986   233.      286.      227.
  315.      239.       91.7431  387.      333.      148.      161.
  129.      305.      635.      485.      158.      585.      610.
   12.0482   66.2252  484.      159.      234.      470.       13.3333
  328.       21.7391   17.5439   27.027   445.      465.      775.
  570.      490.      313.       22.7273   23.2558  243.      177.
   14.9254   15.625    22.2222  163.     1200.       15.3846  925.
   14.2857  383.        8.3333  825.      640.     1050.      540.
   14.0845  725.      248.      645.       15.1515   56.1798   16.5289
  675.      535.      595.      167.      568.     1150.      123.
  505.       79.3651   55.8659  560.      166.      405.      710.
  735.      423.       20.8333  318.      555.      378.      685.
  121.      126.       99.0099  258.       43.29    111.     1300.
  133.      143.       75.188   680.      181.       90.0901   18.8679
  348.       78.125    76.3359   22.4719   57.8035   51.8135  602.
  109.      620.    ]
Number of unique values in column 'BlueExpectedValue': 387

Unique values in column 'Winner': ['Red' 'Blue']
Number of unique values in column 'Winner': 2

Unique values in column 'TitleBout': ['True' 'False']
Number of unique values in column 'TitleBout': 2

Unique values in column 'WeightClass': [ 3  8  4  2  5  1  6  0 12 11  7  9 10]
Number of unique values in column 'WeightClass': 13

Unique values in column 'Gender': [1 0]
Number of unique values in column 'Gender': 2

Unique values in column 'NumberOfRounds': [5 3 4]
Number of unique values in column 'NumberOfRounds': 3

Unique values in column 'BlueCurrentLoseStreak': [0 2 1 3 4 5 6]
Number of unique values in column 'BlueCurrentLoseStreak': 7

Unique values in column 'BlueCurrentWinStreak': [ 0  8  4  1  3  2  6  5  7 10  9 12]
Number of unique values in column 'BlueCurrentWinStreak': 12

Unique values in column 'BlueDraws': [0 1 2]
Number of unique values in column 'BlueDraws': 3
```

```
Unique values in column 'BlueAvgSigStrLanded': [ 0.      5.5     5.13   ... 32.8333 16.8333 13.7   ]
Number of unique values in column 'BlueAvgSigStrLanded': 1702

Unique values in column 'BlueAvgSigStrPct': [0.    0.55  0.57  0.44  0.53  0.49  0.61  0.52  0.47  0.5   0.
54  0.36
 0.46  0.56  0.43  0.42  0.66  0.45  0.31  0.35  0.37  0.38  0.41  0.48
 0.34  0.39  0.58  0.1   0.4   0.59  0.6   0.51  0.64  0.7   0.8   0.33
 0.23  0.28  0.63  0.65  0.71  0.18  0.29  0.62  0.67  0.27  0.32  0.21
 0.3   0.68  0.69  0.26  0.25  0.79  0.2   0.88  0.73  0.24  0.74  0.72
 0.19  0.77  0.76  0.397 0.529 0.377 0.526 0.522 0.317 0.616 0.556 0.667
 0.348 0.599 0.642 0.362 0.376 0.467 0.445 0.413 0.474 0.391 0.342 0.502
 0.436 0.487 0.559 0.417 0.507 0.412 0.456 0.571 0.503 0.575 0.466 0.399
 0.496 0.499 0.454 0.536 0.567 0.345 0.562 0.587 0.335 0.495 0.515 0.549
 0.597 0.464 0.433 0.463 0.358 0.504 0.398 0.514 0.455 0.477 0.353 0.395
 0.451 0.361   nan 0.605 0.497 0.462 0.434 0.525 0.544 0.367 0.662 0.437
 0.573 0.426 0.472 0.288 0.483 0.475 0.583 0.543 0.405 0.465 0.505 0.664
 0.537 0.488 0.453 0.165 0.435 0.352 0.16  0.457 0.363 0.425 0.371 0.485
 0.545 0.368 0.359 0.527 0.885 0.563 0.427 0.382 0.593 0.366 0.523 0.651
 0.486 0.383 0.442 0.375 0.409 0.447 0.13  0.394 0.566 0.484 0.492 0.374
 0.333 0.347 0.324 0.459 0.612 0.388 0.518 0.452 0.343 0.326 0.423 0.476
 0.513 0.565 0.336 0.432 0.438 0.263 0.552 0.248 0.307 0.542 0.561 0.501
 0.357 0.385 0.655 0.325 0.553 0.319 0.387 0.491 0.618 0.602 0.431 0.332
 0.356 0.75  0.424 0.443 0.378 0.441 0.538 0.386 0.554 0.393 0.471 0.705
 0.574 0.615 0.479 0.428 0.411 0.685 0.578 0.22  0.675 0.235 0.444 0.511
 0.372 0.516 0.603 0.558 0.365 0.531 0.524 0.448 0.392 0.446 0.364 0.577
 0.509 0.401 0.285 0.283 0.601 0.478 0.384 0.429 0.355 0.402 0.346 0.396
 0.403 0.535 0.506 0.473 0.354 0.596 0.389 0.713 0.458 0.407 0.338 0.757
 0.315 0.406 0.555 0.533 0.418 0.275 0.468 0.304 0.416 0.414 0.737 0.337
 0.419 0.339 0.379 0.421 0.493 0.598 0.585 0.517 0.267 0.215 0.519 0.494
 0.489 0.592 0.408 0.586 0.314 0.637 0.373 0.606 0.78  0.308 0.498 0.645
 0.404 0.327 0.607 0.15  0.619 0.693 0.344 0.539 0.481 0.313 0.633 0.551
 0.305 0.259 0.692 0.422 0.609 0.613 0.469 0.625 0.512 0.643 0.415 0.665
 0.12  0.795 0.237 0.568 0.309 0.17  0.635 0.634 0.439 0.541 0.627 0.381
 0.759 0.638 0.564 0.295 0.547 0.743 0.461 0.449 0.294 0.322 0.328 0.175
 0.508 0.534 0.646 0.703 0.207 0.628 0.763 0.296 0.745 0.528 0.653 0.238
 0.11  0.482 0.225 0.591 0.334 0.688 0.331 0.83  0.588 0.608 0.742 0.253
 0.569 0.243 0.589 0.707 0.548 0.695 0.611 0.579 1.    0.81  0.14  0.727
 0.172 0.279 0.297 0.369 0.595 0.715 0.329 0.255 0.623 0.244 0.657 0.302
 0.805 0.268 0.303 0.205 0.725 0.245 0.185 0.157 0.227 0.269 0.323 0.09
 0.223 0.532 0.272 0.252 0.217 0.614 0.648 0.604 0.242 0.318 0.293 0.622]
Number of unique values in column 'BlueAvgSigStrPct': 432
```

```
Unique values in column 'BlueAvgSubAtt': [0.     0.3    0.2    0.5    0.8    0.6    0.7    0.1    2.3    1.
4
 1.     1.9    1.2    1.5    0.4    1.3    0.9    1.1    2.8    1.7
 1.6    2.     2.4    2.2    1.8    2.1    8.4    2.5    3.7    2.6
 3.9    4.3    2.7    3.     4.1    4.     3.4    5.2    3.2    3.6
 3.5    2.9    3.8    5.8    0.64   1.37   1.75   0.24   0.52   0.14
 0.16   0.88   0.15   0.23   0.96   2.07   0.35   0.45   0.25   1.68
 0.78   3.3    3.1    0.3548 0.6667    nan 0.3636 0.3333 0.1667 0.7857
 1.0909 0.7619 0.75   0.3571 0.2609 0.4167 0.5385 0.3667 0.375  0.5714
 0.619  0.1429 0.125  0.2857 0.05   2.3333 0.4667 0.7273 0.0556 0.7143
 0.0909 0.2222 0.4545 0.1111 0.0588 0.2143 0.2727 0.1818 0.8182 1.0952
 6.     0.0625 1.1667 0.0769 0.3793 1.2857 1.1429 0.5556 0.7778 5.
 1.1053 0.8889 0.625  0.5625 0.4286 0.4615 0.4444 0.8571 0.5455 0.7895
 0.5263 0.8333 0.4706 0.55   1.6667 1.125  0.6364 0.6154 0.875  1.3158
 0.6429 0.0714 0.6296 1.25   0.1538 0.3478 0.9167 0.4737 1.3333 0.2083
 0.3704 0.6538 0.2941 0.6087 0.5333 0.7647 0.6923 0.2174 0.5833 0.9231
 0.381  1.4444 1.5714 0.5909 1.1111 0.5294 0.3158 0.4375 0.0667 2.25
 0.72   0.2778 1.2941 1.8333 1.1333 0.1333 1.44   0.5217 1.0455 0.3529
 0.3913 0.5789 1.0833 1.375  1.5217 0.0833 0.7222 0.4211 0.9412 0.2105
 0.4091 0.8235 1.15   0.2632 1.1875 0.5238 0.8077 1.0714 0.2667 0.3077
 0.3889 0.7368 0.2308 0.8667 0.3846 0.9375 0.84   1.4615 1.1538 1.4167
 0.3125 0.2917 0.7059 1.2222 0.3043 0.913  1.4375 0.7333 0.6957 1.2105
 1.2727 0.2381 1.1818 1.4286 1.8235 0.8125 1.2308 0.8462 0.9286 0.0952
 2.5714 0.5882 1.2778 0.8421 1.3125 0.9091 1.6154 0.1053 2.6667 1.4545
 0.1765 0.9333 1.3846 1.0526 1.5385 1.8182 2.3636 1.8571 1.5833 1.5789
 0.4783 7.     1.7273 1.3889 0.1176 0.4118 0.6111 1.4706 1.7143 0.6471]
Number of unique values in column 'BlueAvgSubAtt': 250

Unique values in column 'BlueAvgTDLanded': [ 0.      0.77    0.45    0.47    0.75    0.46    4.89    2.31
0.96
  1.79    1.62    1.45    2.74    1.08    0.56    0.72    4.67    1.6
  1.7     1.74    1.59    1.52    2.79    2.3     1.51    2.16    4.37
  0.44    6.39    1.97    2.68    0.92    0.59    2.17    0.84    1.39
  1.46    1.27    2.5     1.07    0.48    2.34    1.02    1.1     2.18
  2.53    0.25    4.31    0.66    0.99    0.19    1.95    5.36    0.5
  0.81    0.78    1.      2.63    0.23    1.42    5.08    1.13    0.49
  2.73    1.85    0.91    3.38    0.34    3.97    1.04    2.07    2.
  2.48    1.81    1.92    0.83    2.75    1.96    2.4     0.95    1.09
  0.57    1.14    1.67    3.08    3.85    2.39    1.18    0.36    0.8
  2.88    3.53    6.09    0.64    1.21    1.72    0.76    1.2     3.49
  0.4     1.06    4.36    4.8     3.26    3.44    0.41    2.02    9.63
  1.56    2.99    2.2     1.16    2.14    1.57    0.05    0.73    4.52
```

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2.77 | 2.94 | 1.5 | 1.75 | 4.63 | 3.09 | 2.62 | 4.2 | 3.98 |
| 1.68 | 3.35 | 2.95 | 1.23 | 0.63 | 0.65 | 0.85 | 2.28 | 5.69 |
| 0.21 | 1.93 | 0.54 | 0.15 | 2.29 | 0.86 | 2.26 | 1.03 | 1.61 |
| 1.73 | 2.04 | 0.7 | 2.49 | 2.09 | 1.12 | 2.41 | 2.33 | 0.55 |
| 3.91 | 0.98 | 1.33 | 2.37 | 2.15 | 1.38 | 1.82 | 1.78 | 1.01 |
| 3.24 | 0.71 | 0.68 | 0.67 | 3.3 | 1.11 | 0.88 | 0.89 | 2.45 |
| 4.08 | 1.48 | 0.27 | 1.24 | 4.17 | 2.23 | 3. | 1.47 | 1.22 |
| 1.37 | 0.6 | 0.2 | 4.12 | 0.33 | 7.44 | 1.58 | 2.78 | 3.59 |
| 0.38 | 3.6 | 3.25 | 4.42 | 1.89 | 1.53 | 2.84 | 1.17 | 1.69 |
| 0.93 | 1.31 | 1.83 | 2.97 | 4.98 | 6.62 | 1.55 | 0.28 | 0.79 |
| 1.49 | 6.06 | 0.13 | 2.22 | 0.82 | 0.51 | 4.76 | 6.85 | 1.44 |
| 1.65 | 6.68 | 5.29 | 1.26 | 3.06 | 4.65 | 1.86 | 2.54 | 2.32 |
| 3.99 | 1.8 | 1.99 | 2.43 | 0.22 | 0.61 | 0.31 | 3.04 | 4.75 |
| 1.94 | 2.25 | 3.15 | 3.79 | 0.26 | 3.45 | 4.6 | 0.29 | 4.23 |
| 0.39 | 2.19 | 3.11 | 5.19 | 0.17 | 3.27 | 3.73 | 2.86 | 2.65 |
| 2.05 | 3.29 | 1.76 | 3.41 | 5.25 | 1.54 | 0.35 | 3.69 | 0.58 |
| 1.9 | 1.34 | 1.35 | 0.9 | 6.2 | 3.13 | 1.66 | 0.11 | 1.4 |
| 1.19 | 4.07 | 0.69 | 2.61 | 1.15 | 3.61 | 4.43 | 4.34 | 2.12 |
| 0.42 | 0.16 | 1.71 | 3.43 | 0.62 | 3.01 | 4.04 | 1.84 | 0.09 |
| 2.24 | 2.59 | 5.2 | 0.87 | 1.91 | 0.52 | 0.97 | 0.43 | 2.82 |
| 3.96 | 1.77 | 6.43 | 2.27 | 5.4 | 0.94 | 2.72 | 4.55 | 3.72 |
| 6.63 | 2.83 | 3.76 | 5.28 | 1.88 | 5.24 | 0.24 | 1.3 | 3.66 |
| 2.51 | 3.17 | 0.18 | 4.77 | 3.19 | 0.74 | 1.25 | 6. | 3.1 |
| 4.18 | 3.63 | 4.28 | 4.45 | 0.14 | 3.22 | 2.35 | 3.71 | 3.87 |
| 3.14 | 6.75 | 4.16 | 3.93 | 2.9 | 2.85 | 5.03 | 2.11 | 2.13 |
| 2.44 | 3.83 | 3.5 | 7.3 | 2.6 | 3.31 | 5.57 | 4.14 | 2.58 |
| 0.53 | 1.05 | 4.33 | 2.93 | 2.08 | 1.32 | 6.46 | 1.87 | 1.63 |
| 1.36 | 1.29 | 5.35 | 3.81 | 0.3 | 2.01 | 4.83 | 4.11 | 2.03 |
| 2.1 | 3.12 | 3.78 | 6.86 | 2.8 | 10.61 | 3.92 | 4.97 | 7.84 |
| 8.93 | 4.91 | 5.52 | 2.56 | 4. | 2.98 | 3.65 | 1.98 | 10.86 |
| 3.54 | 6.91 | 2.21 | 0.32 | 5.27 | 3.37 | 2.57 | 3.7 | 4.51 |
| 2.64 | 3.28 | 3.16 | 2.55 | 1.28 | 2.76 | 4.27 | 5. | 3.03 |
| 2.71 | 3.02 | 1.41 | 7.17 | 4.24 | 8.67 | 3.82 | 6.77 | 8.7 |
| 4.84 | 3.74 | 3.34 | 1.43 | 3.18 | 4.26 | 2.46 | 9.93 | 5.78 |
| 3.55 | 3.67 | 3.23 | 2.91 | nan | 6.66 | 7.02 | 3.39 | 4.5 |
| 4.48 | 0.37 | 2.42 | 5.48 | 6.53 | 3.32 | 2.69 | 3.84 | 0.6774 |
| 0.3333 | 0.5833 | 1.125 | 0.3636 | 0.8333 | 1.3333 | 2.1667 | 1.5238 | 2.3571 |
| 0.1304 | 2.6667 | 0.375 | 0.6667 | 1.0833 | 2.4231 | 7.3333 | 0.5556 | 0.8571 |
| 1.1667 | 1.1429 | 1.875 | 1.6667 | 0.5333 | 1.0909 | 2.3333 | 0.2778 | 0.5714 |
| 0.6364 | 2.1111 | 0.9091 | 4.4444 | 5.75 | 0.4118 | 0.4286 | 2.4286 | 0.2727 |
| 1.4545 | 1.9091 | 2.9048 | 0.1364 | 0.0625 | 0.0833 | 2.4615 | 0.6897 | 1.2857 |
| 1.5455 | 4.3333 | 3.2 | 0.7273 | 0.9167 | 0.7778 | 0.1429 | 0.1818 | 1.4444 |

```
  0.1667  2.0526  0.4444  1.625   8.      1.1818  0.125   3.0909  1.7692
  1.2222  1.3636  3.6667  0.8889  0.1     1.4211  0.1333  0.1111  0.5652
  1.9474  0.6923  0.7143  4.6667  0.2857  0.875   0.9412  3.75    0.9643
  0.625   4.0909  0.3846  0.6154  1.7778  4.5556  4.8333  3.4     1.3684
  0.0714  3.1111  3.8     1.4286  0.6087  1.375   0.7857  0.1538  2.875
  2.0909  0.2222  2.625   1.1111  0.0909  0.8125  7.      1.8889  2.125
  5.5     3.2308  0.4706  2.0588  1.7391  1.0667  0.5294  2.8571  1.0769
  1.7273  4.1429  1.6154  1.0714  3.2222  5.125   2.2727  2.7143  1.8182
  3.8889  2.5833  0.7083  1.4167  0.8182  6.5     1.8333  1.2308  1.8235
  3.2105  3.3333  4.7143  1.8571  0.6818  5.3333  1.3913  1.1538  4.375
  3.2857  1.2941  3.625   3.1667  2.5556  0.5385  2.2667  1.1304  0.4167
  1.0455  0.2353  5.6667  2.1429  2.8333  1.5714  1.1739  0.4667  1.1053
  0.8235  1.7143  0.7619  0.7692  2.3529  0.5263  1.5385  2.7727  0.2308
  1.2273  2.2941  4.2857  0.1053  2.5714  2.6316  2.3077  0.3571  1.0952
  0.8077  2.5385  1.2381  1.7857  2.4118  1.4667  1.7647  0.5455  0.7222
  1.5263  2.2308  2.1818  2.3846  2.1538  1.1905 10.      0.9333  2.7778
  2.4167  1.8462  1.2143  2.0833  1.6111  0.5789  4.25    0.3158  2.3636
  1.6364  1.9333  1.6957  3.4167  0.8696  0.4545  0.3125  1.2667  0.6111
  3.4783  1.7059  2.0714  0.7895  2.375   2.2857  2.7     1.8667  1.1765
  3.7143  1.3529  1.3889  1.6875  2.2222  3.4615  1.3846  5.8     1.2727
  1.0588  1.4118  0.1765  3.2727  2.8462  1.5556  2.7273  3.6316  2.9167
  0.7368  3.7273  3.1818  1.9286  1.5294  0.3077  0.8667  2.1154  1.9231
  0.5625  0.9286  1.5833  3.4286  0.9231  3.125   3.1429  3.5714  1.5625
  7.5     0.4737  0.8824  3.8333  2.2609  0.7059  1.9412  1.2778  4.3636
  1.2353  0.8462  4.7     3.0833]
Number of unique values in column 'BlueAvgTDLanded': 715

Unique values in column 'BlueAvgTDPct': [0.    0.55  0.63  0.25  0.37  0.26  0.65  0.27  0.23  0.41  0.24
0.5
 0.34  0.32  0.18  0.75  0.58  0.14  1.    0.21  0.2   0.33  0.35  0.38
 0.66  0.39  0.12  0.11  0.43  0.16  0.36  0.4   0.53  0.46  0.52  0.51
 0.6   0.44  0.47  0.45  0.29  0.56  0.3   0.57  0.68  0.64  0.28  0.61
 0.13  0.8   0.48  0.42  0.72  0.54  0.31  0.17  0.22  0.62  0.73  0.59
 0.07  0.76  0.19  0.77  0.83  0.06  0.69  0.49  0.71  0.15  0.9   0.09
 0.05  0.86  0.7   0.08  0.78  0.1   0.79  0.417 0.091 0.518 0.381 0.312
 0.182 0.375   nan 0.225 0.295 0.288 0.233 0.125 0.165 0.241 0.096 0.228
 0.357 0.393 0.455 0.315 0.625 0.415 0.524 0.457 0.094 0.277 0.363 0.343
 0.305 0.266 0.453 0.264 0.323 0.107 0.221 0.333 0.473 0.163 0.254 0.466
 0.335 0.314 0.523 0.329 0.735 0.187 0.066 0.144 0.313 0.583 0.215 0.605
 0.498 0.287 0.258 0.167 0.497 0.194 0.164 0.085 0.123 0.398 0.435 0.553
 0.153 0.106 0.399 0.255 0.162 0.075 0.214 0.433 0.136 0.678 0.166 0.451
 0.318 0.063 0.332 0.387 0.592 0.322 0.319 0.083 0.281 0.325 0.282 0.396
```

```
 0.548 0.235 0.099 0.273 0.428 0.527 0.243 0.047 0.146 0.151 0.374 0.248
 0.195 0.366 0.356 0.267 0.133 0.334 0.193 0.304 0.436 0.239 0.198 0.385
 0.222 0.331 0.143 0.093 0.407 0.081 0.208 0.494 0.262 0.426 0.154 0.486
 0.353 0.276 0.309 0.383 0.493 0.224 0.056 0.347 0.421 0.202 0.437 0.071
 0.249 0.192 0.82  0.487 0.272 0.503 0.362 0.539 0.067 0.253 0.395 0.284
 0.213 0.545 0.365 0.132 0.283 0.124 0.645 0.405 0.256 0.179 0.514 0.246
 0.219 0.394 0.271 0.506 0.612 0.418 0.104 0.328 0.447 0.326 0.403 0.475
 0.563 0.355 0.419 0.321 0.777 0.244 0.595 0.212 0.479 0.432 0.444 0.077
 0.443 0.541 0.481 0.141 0.183 0.567 0.538 0.178 0.055 0.376 0.286 0.316
 0.238 0.369 0.119 0.522 0.307 0.229 0.667 0.234 0.665 0.516 0.535 0.468
 0.301 0.448 0.543 0.172 0.558 0.588 0.411 0.364 0.263 0.373 0.346 0.097
 0.109 0.425 0.943 0.371 0.404 0.729 0.368 0.354 0.293 0.569 0.422 0.577
 0.484 0.439 0.581 0.294 0.408 0.715 0.424 0.292 0.177 0.044 0.261 0.615
 0.298 0.111 0.502 0.041 0.188 0.227 0.653 0.429 0.205 0.412 0.306 0.574
 0.086 0.269 0.454 0.915 0.358 0.647 0.508 0.413 0.643 0.341 0.402 0.472
 0.585 0.175 0.482 0.135 0.291 0.137 0.197 0.265 0.533 0.458 0.626 0.495
 0.714 0.512 0.308 0.467 0.367 0.351 0.456 0.279 0.04  0.103 0.587 0.673
 0.324 0.537 0.207 0.118 0.232 0.388 0.562 0.176 0.278 0.344 0.438 0.105
 0.303 0.159 0.833 0.753 0.465 0.349 0.223 0.074 0.065 0.382 0.631 0.601
 0.131 0.463 0.679 0.218 0.478 0.311 0.477 0.378 0.441 0.632 0.359 0.226
 0.446 0.098 0.505 0.639 0.379 0.676 0.607 0.302 0.174 0.471 0.317 0.337
 0.416 0.339 0.361 0.406 0.474 0.117 0.633 0.705 0.464 0.274 0.529 0.532
 0.336 0.462 0.622 0.126 0.414 0.115 0.148 0.555 0.145 0.184 0.559 0.685
 0.203 0.598 0.623 0.275 0.531 0.606 0.297 0.196 0.517 0.746 0.526 0.372
 0.431 0.231 0.766 0.571 0.513 0.338 0.392 0.573 0.185 0.237 0.576 0.391
 0.85  0.142 0.342 0.627 0.507 0.546 0.245 0.492 0.423 0.808 0.236 0.345
 0.211 0.084 0.088 0.958 0.476 0.169 0.611 0.575 0.057 0.128 0.401 0.856
 0.268 0.247 0.189 0.773 0.578 0.483 0.568 0.289 0.257 0.775 0.445 0.552
 0.663 0.386 0.259 0.867 0.442 0.053 0.677 0.087 0.873 0.191 0.81  0.635
 0.875 0.251 0.296 0.591 0.812 0.525 0.173 0.171 0.501 0.515 0.596 0.452
 0.129 0.618 0.674 0.157 0.742 0.122 0.593 0.629 0.511 0.488 0.534 0.149
 0.803 0.561 0.572]
Number of unique values in column 'BlueAvgTDPct': 543

Unique values in column 'BlueLongestWinStreak': [ 0  8  4  1  3  9  2  5  6 13  7 10 12 15 17 16]
Number of unique values in column 'BlueLongestWinStreak': 16

Unique values in column 'BlueLosses': [ 0  4  2  3 12  1  8  5  6  9  7 14 13 10 16 11 15]
Number of unique values in column 'BlueLosses': 17

Unique values in column 'BlueTotalRoundsFought': [  0  20  44   7  15  63  12  57  16  14  23  22  26   9
13   2   6   4
```

```
   3  11  46  34   8   5  30  37  52  24  17  32  51  18  98  49 111  19
  25  21   1  27  53  29  31  78  45  65  33  10  60  41  48  28  40  54
  75  50  47  71  42  36  93  39  96  61 108  58  70  56  38  59 103  67
  69  55  77  35  64 101  43  80 104  72  62  79  91  94  87  86  89  68
  97  66  83  73]
Number of unique values in column 'BlueTotalRoundsFought': 94

Unique values in column 'BlueTotalTitleBouts': [ 0  1  5  6  8  7  9  4  3 11  2 10 15 13 14 12 16]
Number of unique values in column 'BlueTotalTitleBouts': 17

Unique values in column 'BlueWinsByDecisionMajority': [0 1 2]
Number of unique values in column 'BlueWinsByDecisionMajority': 3

Unique values in column 'BlueWinsByDecisionSplit': [0 1 2 3 5 4]
Number of unique values in column 'BlueWinsByDecisionSplit': 6

Unique values in column 'BlueWinsByDecisionUnanimous': [ 0  4  1  3  7  2  5  8 11  6  9 10]
Number of unique values in column 'BlueWinsByDecisionUnanimous': 12

Unique values in column 'BlueWinsByKO': [ 0  3  6  4  7  1  2  5  9 11  8 12 10 14 20]
Number of unique values in column 'BlueWinsByKO': 15

Unique values in column 'BlueWinsBySubmission': [ 0  1  6  4  2  3  7  5  8 10 12  9 11 13]
Number of unique values in column 'BlueWinsBySubmission': 14

Unique values in column 'BlueWinsByTKODoctorStoppage': [0 1 2]
Number of unique values in column 'BlueWinsByTKODoctorStoppage': 3

Unique values in column 'BlueWins': [ 0  8 12  1  4 15 16  6  7  2 13  3 14  9  5 11 10 22 21 23 26 18 17 2
0
 19 29 31]
Number of unique values in column 'BlueWins': 27

Unique values in column 'BlueStance': ['Orthodox' 'Southpaw' 'Switch' nan 'Open Stance']
Number of unique values in column 'BlueStance': 5

Unique values in column 'BlueWeightLbs': [125 170 250 145 205 185 155 236 135 115 240 254 246 265 252 242 2
60 235
 264 262 249 256 230 245 253 257 243 255 241 140 220 238 258 225 231 248
 247 251 263]
Number of unique values in column 'BlueWeightLbs': 39
```

```
Unique values in column 'RedCurrentLoseStreak': [0 1 2 3 4 7 6 5]
Number of unique values in column 'RedCurrentLoseStreak': 8

Unique values in column 'RedCurrentWinStreak': [ 6  1  0  8  3  2 18  5  4  7 10 12 13  9 11 17 15 14 16]
Number of unique values in column 'RedCurrentWinStreak': 19

Unique values in column 'RedDraws': [0 1 2]
Number of unique values in column 'RedDraws': 3

Unique values in column 'RedAvgSigStrLanded': [ 4.41    4.12    5.49   ... 12.75   30.7273 33.1   ]
Number of unique values in column 'RedAvgSigStrLanded': 2025

Unique values in column 'RedAvgSigStrPct': [0.49  0.61  0.6   0.58  0.46  0.5   0.52  0.48  0.47  0.45  0.4
0.33
 0.54  0.55  0.38  0.39  0.43  0.29  0.63  0.59  0.42  0.41  0.56  0.44
 0.53  0.51  0.65  0.57  0.37  0.64  0.67  0.62  0.36  0.32  0.28  0.34
 0.7   0.24  0.23  0.69  0.35  0.66  0.31  0.3   0.75  0.72  0.68  0.77
 0.    0.86  0.1   0.78  0.95  0.79  0.82  0.27  0.81  0.74  0.399 0.464
 0.533 0.426 0.522 0.574 0.318 0.431 0.466 0.427 0.405 0.541 0.512 0.381
 0.459 0.489 0.505 0.653 0.348 0.368 0.373 0.73  0.457 0.553 0.417 0.789
 0.667 0.25  0.576 0.366 0.545 0.638 0.478 0.403 0.257 0.295 0.385 0.537
 0.456 0.498 0.467   nan 0.465 0.317 0.453 0.434 0.511 0.484 0.555 0.485
 0.408 0.496 0.303 0.443 0.494 0.432 0.428 0.335 0.528 0.445 0.439 0.488
 0.337 0.495 0.582 0.461 0.507 0.523 0.414 0.454 0.413 0.386 0.479 0.372
 0.689 0.525 0.288 0.444 0.387 0.557 0.458 0.527 0.543 0.321 0.343 0.363
 0.449 0.415 0.365 0.605 0.513 0.473 0.452 0.493 0.487 0.378 0.305 0.416
 0.411 0.554 0.435 0.384 0.338 0.536 0.393 0.499 0.22  0.481 0.379 0.383
 0.544 0.369 0.438 0.333 0.519 0.437 0.579 0.425 0.508 0.357 0.347 0.352
 0.565 0.402 0.468 0.418 0.9   0.647 0.514 0.358 0.451 0.578 0.344 0.535
 0.422 0.476 0.552 0.643 0.323 0.595 0.442 0.423 0.521 0.462 0.325 0.354
 0.486 0.327 0.546 0.678 0.725 0.472 0.483 0.855 0.407 0.549 0.695 0.361
 0.455 0.563 0.591 0.312 0.419 0.652 0.475 0.446 0.491 0.376 0.382 0.429
 0.516 0.517 0.396 0.374 0.397 0.547 0.412 0.587 0.394 0.275 0.421 0.404
 0.509 1.    0.469 0.436 0.526 0.581 0.502 0.256 0.328 0.583 0.566 0.538
 0.463 0.265 0.497 0.685 0.353 0.518 0.448 0.395 0.424 0.367 0.377 0.597
 0.406 0.441 0.477 0.388 0.375 0.501 0.359 0.635 0.339 0.534 0.585 0.356
 0.433 0.561 0.326 0.558 0.603 0.515 0.506 0.341 0.297 0.471 0.568 0.331
 0.277 0.607 0.362 0.584 0.524 0.315 0.722 0.503 0.573 0.531 0.389 0.304
 0.84  0.626 0.324 0.319 0.492 0.575 0.572 0.409 0.283 0.371 0.588 0.322
 0.223 0.398 0.655 0.345 0.26  0.621 0.14  0.598 0.474 0.392 0.308 0.482
 0.273 0.567 0.551 0.316 0.401 0.735 0.255 0.569 0.644 0.733 0.355 0.532
 0.656 0.589 0.616 0.447 0.391 0.266 0.609 0.213 0.688 0.306 0.634 0.292
```

```
 0.648 0.195 0.612 0.624 0.604 0.296 0.548 0.539 0.675 0.205 0.628 0.613
 0.08  0.756 0.623 0.683 0.599 0.055 0.267 0.676 0.577 0.723 0.622 0.18
 0.649 0.775 0.197 0.504 0.21  0.145 0.564 0.313 0.346 0.349 0.627 0.264
 0.364 0.278 0.291 0.556 0.713 0.2   0.342 0.596 0.85  0.529 0.668 0.8
 0.307 0.336 0.665 0.727 0.253 0.602 0.17  0.254 0.329 0.677 0.249 0.83
 0.294 0.738 0.651 0.765 0.311 0.703 0.697 0.592 0.282 0.301 0.276 0.773
 0.71  0.704 0.245 0.625 0.562 0.251 0.571 0.825 0.185 0.299 0.334 0.15
 0.332 0.614 0.12  0.698 0.717 0.268 0.606 0.187 0.633 0.235 0.09  0.805
 0.155 0.11  0.718 0.204 0.351 0.617 0.238 0.608 0.314 0.559 0.728 0.286
 0.262 0.284 0.619 0.749 0.793 0.03  0.248 0.298]
Number of unique values in column 'RedAvgSigStrPct': 464

Unique values in column 'RedAvgSubAtt': [0.8    1.8    0.5    1.6    1.     0.3    0.6    0.2    0.9    0.1
 0.     2.3    2.4    2.7    3.6    0.7    1.7    1.5    0.4    1.4
 1.1    1.2    1.3    2.1    5.3    4.4    2.5    7.3    3.9    1.9
 4.3    2.6    2.     2.2    2.8    7.5    5.4    3.     8.4    3.7
 3.8    3.2    3.3    7.2    5.     2.9    8.3    3.1    6.9    4.5
 4.     6.2    6.     0.35   1.25   1.23   1.58   1.35   0.98   0.19
 0.42   0.58   0.26   0.44   0.83   0.62   1.64   0.74   0.71   1.83
 1.71   0.92   0.31   3.93   0.4286 0.75   0.6667 0.1667 0.3333 0.1818
    nan 0.5556 0.6154 1.2609 0.25   0.0833 0.2308 0.24   0.4615 0.2143
 0.125  0.0909 1.1429 0.4667 0.5294 0.1786 1.2903 0.1429 0.3684 0.2778
 0.05   1.375  0.2857 0.1111 0.4783 1.7273 0.5263 0.3448 0.7143 0.6429
 0.9524 0.5833 0.2222 0.4545 0.4444 0.6842 0.1538 0.8571 1.2727 0.5714
 0.48   0.8462 0.0526 1.3333 0.3889 0.5385 0.0588 0.45   0.56   0.2667
 0.1176 0.2632 0.3846 0.5455 0.3571 1.5556 0.7692 1.625  0.625  1.0714
 0.2941 0.6316 0.0667 0.4118 0.0625 0.1923 1.3448 0.0556 1.3571 0.7059
 0.1333 0.7778 0.2727 1.125  0.2353 1.1667 0.375  0.7222 0.3704 0.6364
 0.0769 0.2105 1.1579 1.7143 0.8889 0.32   0.8947 0.4348 1.3929 0.875
 0.5789 1.16   0.8333 1.0526 0.4167 0.6923 1.3077 0.3636 1.1765 2.6667
 1.1111 1.2778 0.6111 1.0435 1.2083 0.1852 0.9231 0.9091 0.9444 0.3125
 0.7273 0.4375 0.5238 0.6087 0.2273 0.4706 0.4762 1.75   1.2174 1.3636
 0.5625 1.8571 0.2381 1.2273 0.9167 2.1429 0.3529 0.8235 0.5333 1.2632
 0.3077 0.5882 1.4583 1.1176 0.8125 1.2857 1.1875 1.2381 1.1364 1.0909
 1.0952 1.2143 1.4444 1.5714 0.7826 1.5909 1.6667 0.9545 2.3333 0.8182
 0.3158 0.8667 0.6471 1.1333 1.3684 0.55   1.2667 0.64   3.5    0.1765
 0.1053 0.95   1.2222 1.7895 0.2174 0.6875 0.7857 0.4211 1.7222 0.65
 1.5294 1.0833 0.4737 0.7619 1.875  0.8421 1.4667 1.9333 0.3182 0.9286
 1.0625 1.2941 1.1818 0.0714 2.25   0.7647 2.0769 1.6364 1.7778 1.8182
 0.1875 1.8889 0.7333 1.8333 1.0556 1.1538 1.4286 0.2609 1.5333 1.4167
 1.4615 1.5455 0.6957 2.4286 0.4091 0.6818 1.3125 5.5    1.0667]
Number of unique values in column 'RedAvgSubAtt': 289
```

```
Unique values in column 'RedAvgTDLanded': [ 2.61    1.49    0.58    3.45    1.      0.35    4.75    0.72
3.44
  4.36    3.35    3.13    1.61    0.92    0.42    1.16    0.5     0.
  1.65    0.37    2.42    0.84    1.89    2.25    3.61    1.86    0.38
  1.52    0.73    1.42    1.15    2.21    0.93    2.12    2.01    0.78
  1.8     0.31    0.7     1.01    2.17    3.46    1.53    1.59    0.52
  0.21    2.27    0.16    0.34    3.22    4.49    0.68    2.02    0.8
  1.78    0.56    3.84    3.12    3.97    1.6     3.95    6.64    0.33
  2.86    1.22    1.88    1.23    1.29    2.03    0.75    0.74    3.65
  0.59    5.02    4.89    1.11    2.95    0.14    0.83    0.49    1.33
  1.39    0.25    0.28    1.54    1.32    3.28    1.87    0.88    1.05
  2.47    1.28    4.04    3.21    1.7     0.27    5.36    2.24    0.39
  2.11    0.29    0.44    1.07    0.18    1.13    0.15    2.26    4.02
  2.1     2.23    1.17    1.45    1.85    5.15    3.54    3.91    0.76
  2.15    0.63    4.08    3.66    3.04    0.61    5.25    1.19    0.43
  5.66    2.34    0.26    0.97    1.68    1.2     2.33    0.51    0.62
  1.03    0.77    2.63    1.31    1.25    3.27    1.56    1.84    5.29
  2.67    1.94    0.9     2.43    4.65    0.69    1.55    0.67    2.92
  1.69    1.62    2.39    3.41    0.66    0.99    0.46    0.64    1.04
  5.45    5.2     1.1     0.45    1.97    5.5     2.04    0.55    4.8
  5.08    2.5     2.      0.89    0.36    1.47    1.79    2.22    3.19
  6.85    1.18    2.65    3.11    1.99    0.41    0.57    1.96    0.96
  0.54    2.99    1.27    3.1     1.73    1.14    2.73    0.65    1.9
  2.83    0.22    0.17    2.51    0.11    7.46    0.87    0.79    2.37
  1.74    1.06    1.82    4.52    1.91    2.07    4.45    3.38    2.05
  2.29    4.07    4.2     1.24    2.77    1.02    3.      9.69    2.14
  5.03    1.77    1.48    2.62    5.69    4.17    3.98    1.38    0.2
  1.66    2.32    3.26    1.37    1.75    3.17    3.43    3.76    6.43
  6.62    1.4     3.72    4.18    1.26    1.76    3.01    2.49    1.71
  2.48    0.94    1.72    3.96    0.91    3.71    3.73    1.83    2.35
  3.29    1.58    1.09    1.08    5.24    0.82    2.59    4.12    2.82
  0.98    2.85    0.23    1.12    4.39    2.09    1.5     0.05    0.95
  2.31    1.51    4.76    2.41    3.6     0.86    2.74    3.59    1.67
  2.68    3.08    4.6     0.24    3.03    3.85    4.84    3.18    1.3
  3.25    1.81    1.92    2.28    0.47    0.6     1.93    3.3     2.56
  3.14    4.23    6.2     6.      2.19    2.13    1.21    3.63    0.13
  0.09    2.4     4.15    4.16    0.71    5.4     3.99    1.46    0.81
  6.9     0.06    3.16    1.95    6.75    4.43    1.34    4.34    1.44
  3.53    2.78    2.6     4.38    5.64    4.06    3.79    4.63    1.35
  2.45    2.54    2.55    4.32    3.39    4.48    1.57    0.53    6.27
  2.9     7.17    0.19    4.56    4.13    3.15    4.47    0.4     4.55
```

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0.07 | 2.76 | 0.12 | 3.24 | 2.36 | 3.62 | 2.66 | 0.48 | 2.75 |
| 3.2 | 3.31 | 5.77 | 2.16 | 2.58 | 3.47 | 2.06 | 4. | 2.89 |
| 2.64 | 2.57 | 4.7 | 2.44 | 2.7 | 3.69 | 3.02 | 7.38 | 6.96 |
| 7.1 | 6.11 | 1.63 | 2.79 | 3.9 | 4.28 | 1.41 | 4.01 | 2.69 |
| 2.38 | 2.81 | 3.7 | 7.07 | 6.05 | 6.61 | 0.32 | 4.98 | 5. |
| 0.3 | 0.85 | 6.41 | 1.64 | 5.35 | 1.98 | 3.32 | 3.33 | 8.4 |
| 8.33 | 4.91 | 4.81 | 2.2 | 4.59 | 2.08 | 2.8 | 4.5 | 5.09 |
| 4.29 | 4.31 | 2.98 | 4.67 | 6.63 | 7.8 | 3.58 | 2.52 | 4.14 |
| 6.53 | 3.23 | 2.71 | 12.5 | 4.77 | 5.23 | 3.74 | 2.72 | 7.05 |
| 2.46 | 3.93 | 5.22 | 1.43 | 3.06 | 4.42 | 3.4 | 3.5 | 2.53 |
| 3.42 | 3.34 | 6.77 | 2.4286 | 0.625 | 0.1111 | 1.0667 | 0.1667 | 0.2727 |
| 0.3333 | nan | 0.5556 | 0.7143 | 1.7308 | 2.7778 | 1.3043 | 2.8333 | 0.125 |
| 0.9231 | 0.0909 | 0.8889 | 0.6667 | 0.9286 | 0.6875 | 0.1818 | 1.3333 | 0.8571 |
| 1.0833 | 1.8571 | 1.3125 | 0.1176 | 0.3214 | 2.7143 | 1.1613 | 1.7857 | 0.2667 |
| 2.1667 | 1.5714 | 1.6667 | 0.2632 | 0.6429 | 4.1667 | 6.1667 | 0.3846 | 4.125 |
| 1.4167 | 0.4286 | 0.3125 | 0.2857 | 0.913 | 0.7273 | 0.2222 | 1.3636 | 1.9474 |
| 0.4167 | 0.8421 | 0.7778 | 0.931 | 1.4286 | 3.75 | 1.1071 | 8. | 2.3333 |
| 0.3571 | 3.4545 | 0.4737 | 2.0909 | 0.8333 | 1.4667 | 1.1667 | 2.2667 | 0.375 |
| 2.1429 | 1.3182 | 0.5417 | 2.1111 | 1.0625 | 1.1429 | 2.875 | 0.4667 | 1.8462 |
| 0.5385 | 2.4615 | 0.3684 | 3.875 | 2.3 | 1.4444 | 0.2941 | 0.4118 | 3.9091 |
| 0.4706 | 4.7143 | 0.2143 | 2.1579 | 1.7692 | 3.7273 | 1.2222 | 0.6923 | 1.1111 |
| 2.7692 | 0.4211 | 3.1111 | 1.7778 | 4.9 | 0.5294 | 2.6316 | 0.0667 | 1.3571 |
| 0.9412 | 0.4375 | 0.2692 | 8.5 | 0.4545 | 2.5385 | 5.6667 | 3.1667 | 1.037 |
| 1.2069 | 0.4444 | 4.2941 | 0.7333 | 0.1429 | 1.7143 | 1.0909 | 1.375 | 2.6667 |
| 1.2353 | 1.6429 | 0.5833 | 0.9167 | 0.9375 | 0.7037 | 0.5625 | 0.8182 | 0.2308 |
| 3.8571 | 2.4091 | 1.2105 | 3.1053 | 2.2778 | 1.625 | 2.2222 | 0.875 | 1.6842 |
| 0.3889 | 2.913 | 1.8333 | 1.2143 | 11. | 1.2632 | 0.9545 | 4.7778 | 3.0455 |
| 0.3158 | 2.0556 | 1.9167 | 1.4615 | 0.1538 | 3.3333 | 0.7308 | 0.5455 | 1.8889 |
| 0.6522 | 0.6364 | 3.8889 | 2.7857 | 1.3077 | 2.2727 | 1.0385 | 2.1765 | 4.875 |
| 0.4615 | 1.5556 | 0.8462 | 0.5714 | 0.2353 | 0.5333 | 3.2778 | 1.0556 | 0.963 |
| 3.05 | 1.1818 | 2.3077 | 3.2857 | 2.8571 | 4.0625 | 0.0833 | 1.125 | 2.1875 |
| 3.5556 | 2.4167 | 2.9231 | 3.6667 | 1.0769 | 1.6316 | 2.6522 | 0.1364 | 3.1429 |
| 0.3636 | 4.375 | 1.0714 | 1.2174 | 1.8235 | 4.2667 | 0.5238 | 1.1875 | 1.875 |
| 2.7273 | 4.1429 | 3.3889 | 0.9091 | 3.4706 | 1.2857 | 0.7391 | 1.4091 | 1.2273 |
| 2.125 | 3.2353 | 0.1333 | 0.7727 | 3.4286 | 4.2143 | 1.2941 | 2.2941 | 5.2857 |
| 1.6364 | 1.381 | 1.2727 | 0.8824 | 1.4375 | 2.2857 | 1.4545 | 3.375 | 0.1 |
| 3.7143 | 2.3913 | 1.0588 | 1.0952 | 4.3333 | 2.0714 | 0.6842 | 2.0526 | 0.2778 |
| 1.2667 | 2.3636 | 2.0625 | 4.5385 | 5.8333 | 0.7222 | 3.8 | 1.6923 | 0.1579 |
| 2.0769 | 0.6471 | 1.1176 | 1.9375 | 2.381 | 0.0769 | 0.3077 | 2.2308 | 4.5833 |
| 0.5882 | 0.5263 | 1.2778 | 1.1304 | 1.1579 | 0.7647 | 3.2941 | 0.7368 | 2.2632 |
| 3.2143 | 1.3529 | 2.5833 | 2.5455 | 2.6364 | 1.2308 | 1.4211 | 3.0769 | 3.0625 |
| 0.7692 | 0.7857 | 7. | 3.4091 | 1.1538 | 2.8421 | 3.381 | 3.0667 | 4.4444 |

```
  1.5833  1.9333  1.9286  1.3684  1.3158  2.375   0.8125  0.8636  2.5714
  2.625   0.8095  3.55    2.3846  2.8824  1.7727  0.8667  2.4545  1.5882
  0.9048  0.1875  0.3529  2.1818  3.7222  0.8235  3.9474  1.7619  3.4118
  3.8235  3.8333  1.5385  1.5333  0.8947  1.2381  0.6154  3.7778  3.9333
  0.9333  3.1818  3.125   1.7895  3.4615  2.0833  3.4667  2.5556  4.0714
  1.9091  3.5714  1.9583  2.3478  2.0667  3.1538  0.7059  4.1538  3.8824
  1.3846  4.25    2.0435  4.4167  4.0909  3.625   2.3182  3.2727  2.1364
  9.      1.8125  2.2381  4.6667]
Number of unique values in column 'RedAvgTDLanded': 805

Unique values in column 'RedAvgTDPct': [0.47  0.29  0.21  0.41  0.28  0.53  0.46  0.39  0.42  0.5   0.51
0.36
 0.45  0.49  0.73  0.3   0.    0.75  0.33  0.16  0.31  0.25  0.4   0.43
 0.35  0.44  0.2   0.32  0.24  0.38  1.    0.22  0.57  0.37  0.55  0.56
 0.14  0.61  0.54  0.64  0.34  0.23  0.11  0.18  0.26  0.65  0.8   0.27
 0.52  0.66  0.62  0.69  0.12  0.15  0.48  0.63  0.17  0.71  0.91  0.72
 0.59  0.6   0.77  0.68  0.58  0.9   0.07  0.09  0.76  0.19  0.13  0.06
 0.78  0.05  0.1   0.7   0.83  0.81  0.85  0.08  0.88  0.319 0.222 0.539
 0.313 0.714 0.429 0.506 0.833 0.387 0.167 0.462 0.571 0.125 0.433   nan
 0.458 0.601 0.277 0.104 0.818 0.418 0.056 0.301 0.111 0.303 0.091 0.132
 0.333 0.231 0.151 0.297 0.483 0.108 0.478 0.638 0.219 0.067 0.558 0.183
 0.375 0.316 0.875 0.215 0.102 0.379 0.121 0.268 0.364 0.118 0.129 0.453
 0.423 0.278 0.267 0.368 0.341 0.529 0.79  0.211 0.226 0.312 0.127 0.237
 0.398 0.348 0.193 0.063 0.415 0.546 0.513 0.141 0.584 0.357 0.555 0.274
 0.448 0.221 0.438 0.292 0.165 0.208 0.163 0.244 0.171 0.225 0.305 0.557
 0.342 0.351 0.135 0.463 0.615 0.248 0.214 0.161 0.468 0.228 0.503 0.119
 0.275 0.587 0.282 0.073 0.254 0.198 0.362 0.454 0.467 0.665 0.367 0.509
 0.216 0.189 0.389 0.383 0.577 0.194 0.338 0.261 0.466 0.329 0.249 0.444
 0.437 0.152 0.224 0.255 0.535 0.643 0.288 0.608 0.289 0.296 0.286 0.472
 0.206 0.455 0.397 0.526 0.308 0.227 0.114 0.197 0.667 0.179 0.287 0.449
 0.475 0.683 0.419 0.583 0.291 0.542 0.339 0.517 0.371 0.354 0.294 0.373
 0.143 0.173 0.556 0.374 0.245 0.238 0.882 0.298 0.602 0.285 0.311 0.095
 0.259 0.425 0.356 0.314 0.235 0.213 0.496 0.113 0.565 0.138 0.083 0.432
 0.479 0.229 0.417 0.353 0.204 0.212 0.035 0.473 0.404 0.071 0.525 0.186
 0.242 0.304 0.393 0.498 0.177 0.474 0.416 0.344 0.693 0.647 0.403 0.192
 0.241 0.093 0.323 0.302 0.349 0.201 0.166 0.355 0.188 0.322 0.625 0.153
 0.265 0.258 0.047 0.064 0.887 0.154 0.262 0.271 0.377 0.203 0.413 0.414
 0.508 0.281 0.136 0.915 0.566 0.327 0.209 0.688 0.243 0.395 0.232 0.543
 0.279 0.236 0.318 0.381 0.122 0.055 0.572 0.434 0.402 0.385 0.392 0.409
 0.039 0.564 0.376 0.345 0.205 0.272 0.147 0.328 0.594 0.361 0.037 0.346
 0.251 0.596 0.284 0.176 0.435 0.427 0.412 0.239 0.233 0.384 0.785 0.456
 0.172 0.247 0.257 0.407 0.568 0.253 0.677 0.042 0.264 0.476 0.502 0.306
```

```
 0.452 0.283 0.062 0.406 0.494 0.378 0.156 0.075 0.307 0.653 0.382 0.553
 0.648 0.428 0.273 0.694 0.234 0.332 0.343 0.446 0.045 0.669 0.712 0.115
 0.185 0.369 0.325 0.408 0.266 0.436 0.592 0.295 0.522 0.411 0.646 0.405
 0.719 0.336 0.359 0.605 0.358 0.493 0.386 0.459 0.581 0.324 0.256 0.451
 0.223 0.182 0.148 0.133 0.066 0.053 0.464 0.337 0.365 0.699 0.317 0.331
 0.335 0.426 0.162 0.293 0.372 0.347 0.058 0.527 0.532 0.175 0.263 0.658
 0.276 0.441 0.321 0.733 0.269 0.181 0.528 0.545 0.246 0.465 0.611 0.094
 0.309 0.164 0.586 0.515 0.158 0.054 0.137 0.863 0.187 0.424 0.363 0.67
 0.604 0.721 0.077 0.773 0.334 0.352 0.421 0.447 0.442 0.178 0.168 0.533
 0.326 0.388 0.461 0.603 0.481 0.202 0.817 0.613 0.574 0.486 0.443 0.707
 0.184 0.491 0.749 0.471 0.445 0.765 0.117 0.678 0.497 0.128 0.523 0.718
 0.505 0.492 0.401 0.562 0.612 0.504 0.641 0.217 0.422 0.777 0.144 0.485
 0.632 0.544 0.623 0.576 0.366 0.507 0.196 0.074 0.736 0.801 0.521 0.534
 0.431 0.131 0.146 0.516 0.139 0.487 0.775 0.768 0.519 0.488 0.74  0.107
 0.394 0.633 0.661 0.396 0.315 0.484 0.563 0.155 0.595 0.573 0.674 0.514
 0.145 0.123 0.098 0.585 0.551 0.793 0.606 0.391 0.207 0.743 0.569 0.511
 0.469 0.579 0.645 0.399 0.758 0.94  0.626 0.607 0.537 0.618 0.616 0.659
 0.518 0.652 0.644 0.567 0.614 0.116 0.499 0.495 0.489 0.157 0.547 0.477
 0.591 0.682 0.524 0.748 0.593 0.457 0.675 0.627 0.82  0.575 0.767 0.716
 0.126 0.552 0.666 0.554 0.843 0.541 0.092 0.732 0.751]
Number of unique values in column 'RedAvgTDPct': 609

Unique values in column 'RedLongestWinStreak': [ 6  7  3  8  4  9  2  1 18 11  5  0 15 12 10 13 17 14 16]
Number of unique values in column 'RedLongestWinStreak': 19

Unique values in column 'RedLosses': [ 3  0  2  4  6  5  7 19  1 10 17  8 11  9 12 13 21 15 16 20 18 14]
Number of unique values in column 'RedLosses': 22

Unique values in column 'RedTotalRoundsFought': [ 42  11  33  22  17  21  47  24  43  46  41 105  31  45  3
4  23   2   8
   6   4   1  12  74  63  35  39  99  14  49  88   5  13  40  26  37  53
  19   9  29  36  20  15  16  64  30  10  25  72   3  65   7  54  90  60
  44  59  57   0  73  87  18  48  27  66  58  75  28  79 103  56  76  62
  32  61  38  51  71  84  93 100  55 102  52  68  69  50  81  96  78  92
  98  85  70  89  67  97  77  94  95  91  83  86  80 448  82]
Number of unique values in column 'RedTotalRoundsFought': 105

Unique values in column 'RedTotalTitleBouts': [ 3  0  2  5  1  4 16  6 15  7  8  9 11 10 12 14 13]
Number of unique values in column 'RedTotalTitleBouts': 17

Unique values in column 'RedWinsByDecisionMajority': [0 1 2]
Number of unique values in column 'RedWinsByDecisionMajority': 3
```

```
Unique values in column 'RedWinsByDecisionSplit': [2 0 1 4 3 5]
Number of unique values in column 'RedWinsByDecisionSplit': 6

Unique values in column 'RedWinsByDecisionUnanimous': [ 4  0  3  5  1  7  6  9  2 10 11  8]
Number of unique values in column 'RedWinsByDecisionUnanimous': 12

Unique values in column 'RedWinsByKO': [ 2  1  4  0  8  3  6  5 11  7 10  9 14 13 12 21 20 19]
Number of unique values in column 'RedWinsByKO': 18

Unique values in column 'RedWinsBySubmission': [ 4  5  2  1  0  3  7  6 16 12 11  8  9 10 15 14 13]
Number of unique values in column 'RedWinsBySubmission': 17

Unique values in column 'RedWinsByTKODoctorStoppage': [0 1 2]
Number of unique values in column 'RedWinsByTKODoctorStoppage': 3

Unique values in column 'RedWins': [12  6  9  7  5 15  8 13 20  2  1  3 21 22 10 26  4 17 18  0 11 14 19 25
 16 24 27 23 29 33 32]
Number of unique values in column 'RedWins': 31

Unique values in column 'RedStance': ['Orthodox' 'Southpaw' 'Switch' 'Open Stance']
Number of unique values in column 'RedStance': 4

Unique values in column 'RedWeightLbs': [125 170 245 145 205 185 155 135 115 248 249 250 262 240 264 254 26
5 256
 239 257 260 235 242 253 246 230 243 255 258 225 140 238 231 241 263 247]
Number of unique values in column 'RedWeightLbs': 36

Unique values in column 'RedAge': [34 30 36 33 40 37 42 32 31 35 26 28 29 25 24 41 39 27 38 19 23 22 45 20
 44 18 21 43 47]
Number of unique values in column 'RedAge': 29

Unique values in column 'BlueAge': [31 27 36 33 35 30 37 23 39 25 32 29 40 34 28 22 42 38 24 26 41 21 43 20
 44 45 19 46 47]
Number of unique values in column 'BlueAge': 29

Unique values in column 'LoseStreakDif': [ 0  1 -1 -2  2 -3  3 -4 -6  4  5 -5  6]
Number of unique values in column 'LoseStreakDif': 13

Unique values in column 'WinStreakDif': [ -6   2   3   0  -1   4  -7  -2 -18  -5   5  -3   1  -4 -10   8 -1
2   6
   7  -9 -11  -8  10 -16   9 -13]
```

```
Number of unique values in column 'WinStreakDif': 26

Unique values in column 'LongestWinStreakDif': [ -6   2  -3  -5   0  -2   1  -1  -7   3 -12  -8   6  -4  -9
4   5   9
   7 -10   8  11 -11  14  12  10]
Number of unique values in column 'LongestWinStreakDif': 26

Unique values in column 'WinDif': [-12   2   3  -6  -1   8 -11  -7  -4  -2 -13   4   1   0 -18 -20  -9   7
  -5 -10  -3   5  15  13  -8 -14 -17   6   9 -15  10 -21  12 -16 -24 -19
 -26  11  16 -22 -27  23 -28  14  19]
Number of unique values in column 'WinDif': 45

Unique values in column 'LossDif': [ -3   0   2   8  -5   4  -4   1 -16  -1   3  -7 -12  -2  -8 -11  -9  -6
   5   7  10   6 -10 -20  12  11 -14 -19   9 -15 -13 -17 -18  14  16  13]
Number of unique values in column 'LossDif': 36

Unique values in column 'TotalRoundDif': [ -42    9   11  -15   -2   42  -35   33  -27  -10    3  -82  -17
-1
  -12    7  -18    0   -9  -28  -49   28  -34  -69  -14  -43  -13  -81
   24   12  -31  -40  -21  -24   -5   -4   34  -22    4    1    5   82
  -52  -11   29   81    6  -25  -16  -45  -41   -7   25    8  -70   32
  -32  -29   56   10  -30   17   -3   -8   23   -6   14  -38  -44    2
   15  -54  -74   20  -19  -33   16  -65   13  -20  -37   18  -90  -60
  -26   41   19   31  -61   60  -47  -23  -53   26  -51  -46  -50   65
   22  -48   37  -62   87   21  -68  -79  -67  -89   49   51   44  -59
   36   53   43  -36   46   63   27   39  -66  -92  -75  -58  -39   62
  -57  -64  -77  -72  -55   40   50   38  -83   52  -76  -86   45  -80
   54   80   48  -71   59   30 -448  -56   47  -73   55  -84   35   57
   58]
Number of unique values in column 'TotalRoundDif': 155

Unique values in column 'TotalTitleBoutDif': [ -3   0  -1   5  -5  -2   2  -8  -6   7   8  -4   4   1 -15
9  10  -7
   6  -9 -10 -11 -13   3  15 -12  13  14 -14  11 -16]
Number of unique values in column 'TotalTitleBoutDif': 31

Unique values in column 'KODif': [ -2   2   0   3  -7  -6  -1   1   4  -5  -3   7  -4 -10   6   5  -8   9
 -14  10  -9  12 -12 -20   8 -11 -17 -21  14 -13  11]
Number of unique values in column 'KODif': 31

Unique values in column 'SubDif': [ -4  -5  -1   0  -2   5   4  -3  -7   1  -6 -15   2 -10 -11   7   3   8
   9 -16  -8  -9   6  10 -13 -12]
```

```
Number of unique values in column 'SubDif': 26

Unique values in column 'HeightDif': [   7.62    5.08   -2.54    2.54    0.     -5.08   15.24  -10.16   -7.
62
  -12.7    12.7    10.16  -20.32   17.78  -15.24   20.32  -17.78   22.86
  -22.86 -187.96   -5.5   -25.4   -27.94  -33.02   30.48   27.94]
Number of unique values in column 'HeightDif': 26

Unique values in column 'ReachDif': [ 5.0800e+00 -7.6200e+00 -2.5400e+00  0.0000e+00 -5.0800e+00  2.5400e+0
0
  1.0160e+01 -1.2700e+01 -1.7780e+01  1.5240e+01 -1.0160e+01  2.0320e+01
  7.6200e+00  1.2700e+01  1.7780e+01 -1.5240e+01 -2.0320e+01 -2.2860e+01
  2.2860e+01 -2.5400e+01  2.5400e+01 -1.8796e+02 -1.6002e+02 -1.3400e+00
 -2.3400e+00  8.8900e+00 -1.3970e+01 -1.9050e+01  6.3500e+00  1.2700e+00
 -1.6510e+01 -6.3500e+00 -3.8100e+00 -1.2700e+00 -1.7960e+01  4.0000e-02
 -4.2800e+00  2.5800e+00 -8.9800e+00 -4.2000e-01 -8.1000e+00  7.6000e+00
 -1.0440e+01  2.7400e+00  1.6600e+00  1.0640e+01 -1.8260e+01  2.7940e+01
 -8.0400e+00 -9.9600e+00 -4.9800e+00 -3.0480e+01 -2.7940e+01  8.9300e+00
  1.2820e+01  1.0000e-01 -5.1800e+00 -9.6000e-01  7.2000e+00  1.8000e-01
  5.2800e+00  2.5000e+00  2.8000e-01 -5.4200e+00  3.6600e+00  2.3400e+00
 -5.2000e+00 -2.8200e+00 -9.8000e+00 -5.5000e+00  2.0000e-01 -7.5000e+00
 -7.5800e+00 -4.8800e+00 -1.0420e+01 -4.9600e+00  2.2800e+00  1.1200e+00
 -7.4200e+00  4.9000e+00 -4.8000e+00  1.0500e+01  2.0580e+01  1.0000e+01
  4.6600e+00  5.2000e+00 -1.3120e+01 -7.2600e+00 -7.8800e+00 -3.4000e-01
  8.0400e+00 -1.2260e+01 -1.0040e+01  7.2800e+00  1.2960e+01  1.5120e+01
 -1.0120e+01  5.5000e+00 -1.2800e+00  1.2900e+01 -5.0400e+00 -1.4800e+01
 -2.5800e+00 -3.3020e+01  4.0000e+00  2.6600e+00 -1.5280e+01  1.2000e-01
 -2.0000e+00 -2.0580e+01  1.2500e+01  1.8200e+01  1.4040e+01  4.5800e+00
  1.5280e+01 -1.5340e+01 -4.0000e-02 -2.1200e+00  1.4740e+01 -7.7200e+00
  3.0480e+01 -1.3000e+00]
Number of unique values in column 'ReachDif': 122

Unique values in column 'AgeDif': [ -3   2   6   0   5  -4 -10 -17  -6  -1   8  -8  -2   7   3  -7   4  -5
   1 -15  -9   9 -11  12 -12  10  13  11 -13 -16  14  15 -14  16  17]
Number of unique values in column 'AgeDif': 35

Unique values in column 'SigStrDif': [ -4.41     1.38    -0.36   ...  -7.1091 -15.55   -22.35  ]
Number of unique values in column 'SigStrDif': 3007

Unique values in column 'AvgSubAttDif': [-8.0000e-01 -1.5000e+00 -3.0000e-01 -1.1000e+00 -2.0000e-01  3.000
0e-01
  0.0000e+00  5.0000e-01 -1.0000e-01 -7.0000e-01 -9.0000e-01  1.7000e+00
```

```
 1.3000e+00  1.0000e-01 -2.0000e+00 -5.0000e-01 -2.2000e+00  1.0000e+00
-2.1000e+00  1.2000e+00  4.0000e-01 -6.0000e-01  8.0000e-01  7.0000e-01
-4.0000e-01  1.4000e+00  9.0000e-01  2.0000e-01  1.1000e+00  2.8000e+00
-2.4000e+00 -1.2000e+00  6.0000e-01 -1.3000e+00 -4.8000e+00 -2.5000e+00
-7.3000e+00  1.8000e+00 -1.6000e+00  2.4000e+00 -3.9000e+00  1.9000e+00
-1.7000e+00  2.6000e+00 -2.3000e+00 -1.0000e+00 -3.6000e+00  7.8000e+00
-1.8000e+00 -2.8000e+00 -6.6000e+00  2.1000e+00  3.3000e+00 -1.4000e+00
 1.5000e+00 -5.0000e+00  2.2000e+00 -3.0000e+00  3.4000e+00  2.0000e+00
-8.4000e+00 -2.6000e+00 -1.9000e+00  1.6000e+00 -4.6000e+00 -7.5000e+00
-4.3000e+00  2.5000e+00  3.5000e+00 -5.4000e+00 -7.4000e+00  3.8000e+00
-2.9000e+00 -3.8000e+00  2.7000e+00  2.3000e+00  4.1000e+00 -2.7000e+00
 2.9000e+00  5.2000e+00  3.1000e+00 -3.2000e+00 -6.4000e+00  3.6000e+00
-8.3000e+00 -3.3000e+00 -4.9000e+00  3.0000e+00  4.8000e+00 -4.0000e+00
-5.9000e+00 -6.0000e+00  2.9000e-01 -1.2500e+00  1.4000e-01 -1.5800e+00
-9.8000e-01  2.4000e-01  1.0000e-02 -4.4000e-01  6.2000e-01 -6.8000e-01
-3.9000e-01 -7.4000e-01 -7.1000e-01  2.0700e+00 -1.4800e+00 -1.2600e+00
-6.7000e-01 -1.5000e-01 -2.2500e+00  7.8000e-01 -5.1000e+00  2.7140e-01
-6.4520e-01  2.5000e-01  1.6670e-01 -8.3300e-02 -6.6670e-01  3.3330e-01
 3.0300e-02 -1.8180e-01 -2.2220e-01  1.7030e-01  1.3333e+00 -1.7000e-01
-7.5000e-01 -2.5000e-01  6.7860e-01 -3.9290e-01  2.6090e-01  2.6920e-01
-2.4000e-01  8.1820e-01 -3.3333e+00  7.6900e-02  1.1670e-01 -2.1430e-01
 7.5000e-02 -9.0900e-02 -1.1429e+00 -2.9170e-01 -5.2940e-01 -1.7860e-01
-4.2860e-01 -9.5700e-01  1.4290e-01  2.3800e-02 -3.3330e-01  2.5060e-01
 5.7100e-02 -2.3800e-02  3.2220e-01 -1.6670e-01  1.2500e-01 -5.0000e-02
 2.8570e-01 -1.1670e-01 -8.3330e-01 -1.3750e+00 -4.1670e-01  6.6670e-01
 1.8333e+00 -8.7500e-01  4.6670e-01  2.2730e-01  3.8100e-01  5.5600e-02
 6.0320e-01 -4.0910e-01 -2.5600e-01 -5.4550e-01 -1.2273e+00 -1.4290e-01
-1.2500e-01 -1.2630e-01 -2.7800e-02  8.8890e-01  7.2730e-01  6.5520e-01
-7.1430e-01  1.1110e-01  1.1571e+00 -2.4120e-01 -5.7740e-01 -5.8330e-01
 4.8050e-01 -7.9000e-03  1.3556e+00 -6.8420e-01 -1.9440e-01  8.0950e-01
-1.3330e-01  2.2700e-02  5.0000e+00 -3.9375e+00  1.1667e+00  9.6200e-02
 8.3300e-02 -8.5710e-01 -1.2727e+00  6.8890e-01 -2.6670e-01 -1.7310e-01
 4.2860e-01  3.7930e-01 -2.8570e-01  3.5710e-01 -1.9840e-01  6.0600e-02
 4.4620e-01  2.0000e-02 -1.5380e-01 -5.7340e-01  2.4740e-01 -4.7600e-02
-1.1110e-01 -6.3330e-01  6.6700e-02  7.5400e-01  1.6150e-01 -1.3750e-01
 5.7780e-01 -5.8800e-02  3.7780e-01 -4.5000e-01  2.1210e-01 -4.4440e-01
-3.1000e-01  7.3330e-01 -1.1760e-01 -2.6320e-01  3.9100e-01 -3.8460e-01
 6.3890e-01 -6.2500e-01  3.4290e-01 -3.5710e-01 -1.5556e+00 -4.2000e-02
 1.0260e-01 -9.5830e-01  5.6250e-01 -7.1400e-02  2.2220e-01 -5.2600e-02
 3.5040e-01  4.4440e-01  2.0590e-01 -3.4290e-01  9.0900e-02  4.6840e-01
 7.7780e-01 -6.6700e-02  1.3330e-01 -9.6670e-01  3.7770e-01 -8.1820e-01
 7.5000e-01  9.3750e-01  7.7000e-03  5.2630e-01  8.3330e-01 -1.2020e+00
```

```
 4.4400e-02  8.5700e-02  2.1538e+00  4.7060e-01  6.5000e-01 -1.1071e+00
-5.8090e-01 -2.7780e-01  5.5000e-01  8.4400e-02  1.5000e-01  1.5833e+00
 1.4700e-02 -6.2500e-02 -8.5700e-02 -5.5000e-01 -3.3300e-02  3.2500e-01
-1.1667e+00 -3.7500e-01  6.3640e-01 -1.8460e-01  4.0480e-01  4.0280e-01
-7.2220e-01  1.8520e-01  3.6360e-01  5.5560e-01  4.3333e+00 -3.6110e-01
-7.6900e-02 -2.8210e-01  2.1430e-01  7.5580e-01  4.3230e-01  7.1400e-02
-1.1579e+00  6.2960e-01 -4.6430e-01  2.7800e-02 -5.7140e-01 -6.0320e-01
-1.0710e-01  6.8060e-01 -5.4620e-01  1.6667e+00 -8.9470e-01  8.6110e-01
-2.9190e-01 -5.7890e-01 -9.7400e-02 -1.3571e+00 -3.2630e-01 -6.0710e-01
-1.4440e-01 -1.1600e+00 -3.2140e-01  8.7880e-01 -8.4430e-01  8.7500e-01
-3.5560e-01 -5.2780e-01  1.7040e-01 -2.7380e-01 -4.4400e-02  4.3590e-01
-6.9230e-01 -7.3630e-01  2.6667e+00 -2.1000e-02 -1.8060e-01 -2.7270e-01
-3.6360e-01  1.1429e+00 -2.0830e-01  7.8570e-01 -7.4600e-01  1.4091e+00
-9.6200e-02 -1.1176e+00 -4.4120e-01 -2.1667e+00 -3.9820e-01 -1.1111e+00
 6.0870e-01  2.1540e-01  5.7140e-01  1.0462e+00  3.7500e-01  3.6670e-01
-1.1180e-01  4.1270e-01 -9.4440e-01 -1.0333e+00 -1.1900e-02  7.6470e-01
-1.7140e-01 -3.5120e-01  2.1740e-01 -2.1970e-01  9.1010e-01  9.2310e-01
-2.5000e-02  1.0820e-01 -6.7310e-01 -5.7100e-02  1.0444e+00  8.7180e-01
-3.6670e-01  9.4640e-01 -8.7900e-02 -9.0910e-01  2.1370e-01 -3.7880e-01
-2.3810e-01  1.7780e-01  1.8180e-01  2.7840e-01  3.8890e-01  1.1111e+00
-1.1900e-01  1.2500e+00  7.0830e-01  5.6670e-01  9.1670e-01 -1.3333e+00
 7.6000e-01  4.1670e-01  1.6727e+00 -9.1670e-01 -1.1030e-01 -4.8480e-01
 2.0190e-01 -4.3750e-01  2.6980e-01 -7.7780e-01 -1.5910e-01  1.1364e+00
-1.5480e-01 -5.5560e-01 -2.0800e-01  2.4840e-01 -2.2730e-01 -1.8490e-01
 3.7140e-01  7.1430e-01  1.7140e-01 -1.5710e-01  1.0420e-01  2.7540e-01
-3.2480e-01  3.2100e-02  1.7860e-01 -4.6150e-01  6.9050e-01 -1.7500e+00
 3.8670e-01 -6.1740e-01  9.4290e-01  2.5000e-02  3.3300e-02 -3.4720e-01
 1.2941e+00  1.3462e+00  2.6670e-01  4.3750e-01  1.4583e+00 -3.4620e-01
 6.4290e-01  3.6400e-02  1.0710e-01 -2.3100e-02  6.3330e-01  4.6150e-01
 4.7600e-02 -3.0710e-01  8.7750e-01  3.2140e-01 -1.7650e-01  7.9170e-01
-1.8571e+00  1.4860e-01  1.1900e-02  8.6670e-01 -5.2530e-01  1.1778e+00
-8.7830e-01  2.7270e-01 -1.5290e-01 -8.2350e-01  3.2000e+00 -1.2632e+00
 5.9090e-01  4.1700e-02 -4.1700e-02  8.5710e-01  7.3810e-01 -1.0770e-01
 4.0110e-01  4.1180e-01 -3.0300e-02  1.5290e-01 -4.0900e-02 -5.1790e-01
-8.7000e-03  1.1230e-01 -1.1583e+00  4.8330e-01 -3.0560e-01  4.3060e-01
 1.0180e-01 -9.5100e-01  2.3480e-01  5.2730e-01  1.1540e-01 -1.0667e+00
 2.0630e-01 -8.1250e-01 -6.6070e-01 -3.9000e-02 -1.1875e+00  1.9440e-01
 7.9000e-03  5.8330e-01  2.8360e-01  1.9050e-01 -1.1364e+00  1.0590e-01
-3.4920e-01 -8.9290e-01 -5.1950e-01 -8.4520e-01 -2.5710e-01 -6.4290e-01
-3.5000e-01 -2.7500e-01 -1.4444e+00 -1.0714e+00 -4.1430e-01  4.6100e-02
 5.1740e-01 -1.5909e+00  9.4120e-01 -5.6250e-01  2.2500e+00 -6.3160e-01
-1.8950e-01 -1.5714e+00  1.0833e+00  3.6540e-01  4.2060e-01 -3.1250e-01
```

```
-4.4290e-01 -1.0303e+00 -2.2500e-01 -3.5470e-01  1.0910e-01  4.9020e-01
 3.1430e-01 -6.1900e-01  1.9550e-01  2.6320e-01  1.1875e+00 -4.0280e-01
-2.1150e-01 -3.0360e-01 -2.1333e+00 -1.6667e+00  2.8890e-01  3.5700e-02
 6.1540e-01 -3.7300e-01  5.2380e-01 -2.6667e+00  7.1680e-01  1.0952e+00
-7.2730e-01  6.9230e-01  8.4850e-01 -1.0750e-01 -2.4440e-01  3.8160e-01
-2.4680e-01  7.6200e-02 -8.6670e-01 -3.0770e-01 -1.8400e-02 -4.5450e-01
-2.3330e-01 -3.8500e-02  2.3330e-01 -7.6670e-01  1.4100e-01  4.5450e-01
 1.1760e-01  2.7780e-01 -1.9230e-01  1.1500e+00  1.7430e-01 -9.0260e-01
 4.4000e-02 -2.6260e-01  8.2310e-01 -5.0180e-01 -1.6920e-01 -9.8570e-01
 1.8460e-01 -2.6920e-01 -5.3330e-01  2.0240e-01 -1.7900e-02 -3.1820e-01
-2.7500e+00 -3.0670e-01  2.7080e-01  1.3640e-01 -3.5000e+00 -1.9050e-01
-7.6000e-03  5.4590e-01  1.0449e+00  7.0910e-01  1.5690e-01  8.9470e-01
-8.5000e-01 -8.8890e-01 -1.5395e+00  1.8200e-02  9.3650e-01  6.2500e-01
-4.0480e-01 -7.7220e-01 -9.3570e-01  7.8950e-01 -1.3890e-01 -3.3040e-01
 5.1950e-01 -6.3680e-01  2.7380e-01  4.2900e-02 -4.5390e-01 -4.1180e-01
 1.3070e-01  4.5710e-01 -1.3640e-01  3.8330e-01  2.5670e-01 -1.7780e-01
 3.7270e-01  9.7200e-02  3.2890e-01  5.6800e-02 -9.2220e-01  7.7140e-01
-2.2630e-01 -6.5000e-01 -3.9470e-01 -4.8610e-01  7.9370e-01  7.3600e-02
 1.2154e+00  9.1300e-01  1.2375e+00  1.1250e+00  1.0667e+00 -1.1008e+00
-6.3640e-01 -1.1540e-01  3.8070e-01  6.9570e-01  8.2350e-01 -5.8970e-01
 1.9840e-01  1.3500e+00 -9.3940e-01  4.6050e-01  1.4615e+00 -1.9190e-01
 1.7500e+00  7.9900e-01 -2.3210e-01  3.5000e-01 -1.2857e+00  9.2300e-02
 4.5500e-02 -1.1500e+00  9.1430e-01 -3.8890e-01 -4.6670e-01 -5.9500e-02
-6.0600e-02 -3.3640e-01 -8.4620e-01  5.5680e-01  5.3850e-01 -7.8570e-01
 7.4440e-01  1.4389e+00 -3.6190e-01  1.4550e-01  6.8750e-01 -1.3490e-01
 1.6071e+00 -2.6300e-02 -1.0909e+00  1.2090e-01  4.7500e-01 -4.1070e-01
-8.8900e-02  8.3080e-01  2.8600e-02 -1.4550e-01  8.4620e-01 -1.8750e+00
 1.3750e+00  7.9400e-02 -9.2300e-02  3.3090e-01  2.2333e+00 -1.0833e+00
-1.0202e+00  1.2120e-01  9.2860e-01 -3.4210e-01  9.5200e-02  2.9170e-01
 6.5710e-01  1.7381e+00  4.0000e+00 -3.5700e-02  1.0500e+00  5.8820e-01
-5.8170e-01  1.6250e+00 -1.2167e+00 -6.4710e-01 -6.8330e-01 -4.7500e-01
-4.7620e-01  5.3600e-02  1.1900e-01  1.1905e+00  1.4375e+00 -2.2080e-01
 5.8550e-01 -6.4550e-01  3.0770e-01  8.9090e-01  5.1590e-01 -1.0420e-01
-7.8800e-02  1.5150e-01 -1.0625e+00 -1.0227e+00  9.6000e-03  4.5000e+00
-4.7730e-01  5.2940e-01 -1.5240e-01  4.2100e-02  5.2780e-01 -2.9090e-01
-1.7143e+00  1.1154e+00 -4.4020e-01 -5.8820e-01 -9.2310e-01  4.2500e-01
 6.9640e-01  1.5450e-01  2.3939e+00  2.8330e-01 -4.7060e-01  3.8180e-01
 1.4545e+00  2.9370e-01  5.1500e-02 -1.2941e+00 -1.2143e+00  1.4286e+00
-1.5790e-01  5.9520e-01 -1.1818e+00  2.0910e-01  3.2860e-01 -2.0833e+00
 1.2820e-01  1.6860e-01  5.0980e-01  8.2910e-01 -1.4580e-01 -1.0910e-01
-2.0590e-01 -7.1800e-02 -5.2630e-01  6.9440e-01  5.0000e-02  2.2860e-01
 7.6920e-01 -9.5200e-02 -1.4300e-02  8.4440e-01 -1.9090e-01 -1.0530e-01
```

```
  1.0556e+00  5.2860e-01 -2.0800e-02 -1.2250e+00 -1.6364e+00 -1.6970e-01
  2.0800e-02  9.5240e-01 -2.7450e-01 -1.5682e+00  7.6190e-01 -1.8750e-01
 -2.2860e-01 -2.6370e-01  1.6042e+00  1.1100e-02  1.3940e-01  5.4170e-01
  2.1667e+00  1.0714e+00  1.0526e+00  1.6429e+00  1.5380e-01  2.7500e-01
 -1.0278e+00 -4.5830e-01 -2.3080e-01  1.7143e+00  1.2260e+00  3.0300e-01
  1.3182e+00 -1.8889e+00  6.0200e-02  2.2970e+00  1.0909e+00  1.0536e+00
 -1.9000e-02 -1.4333e+00 -5.5600e-02 -7.0000e-02 -6.9400e-02 -1.1250e+00
  5.8570e-01 -7.6920e-01  3.2310e-01  3.0360e-01  1.8286e+00 -4.3180e-01
 -8.6360e-01  2.3080e-01  5.6667e+00 -1.1538e+00  4.6210e-01  2.0510e-01
  2.8210e-01 -4.9210e-01  4.1667e+00  1.1504e+00 -1.2820e-01  3.4560e-01
  1.0273e+00  7.2500e-02  1.2857e+00 -7.0830e-01 -1.5333e+00  8.1430e-01
 -7.3330e-01 -8.0950e-01  1.2222e+00  1.2143e+00  5.6410e-01 -3.2170e-01
  1.0889e+00 -3.3770e-01  6.0000e+00  1.9700e-01  8.6060e-01 -3.0130e-01
 -1.4143e+00  1.3889e+00 -5.4900e-01  1.9167e+00 -4.7270e-01  5.5710e-01
 -9.6320e-01 -8.9010e-01  4.9230e-01  1.0222e+00 -4.3430e-01 -8.4500e-02
 -1.1786e+00  9.3180e-01 -1.8940e-01  4.7140e-01  4.6030e-01 -6.1540e-01
 -7.5890e-01  7.0140e-01  3.8640e-01  2.4240e-01  8.8100e-01  6.9090e-01
 -2.4286e+00  3.9710e-01  1.7900e-02 -3.7690e-01 -5.1280e-01  9.6360e-01
  6.9320e-01  1.8750e-01 -5.5000e+00  7.4550e-01  9.8570e-01 -4.0180e-01
  6.0710e-01  2.3333e+00 -1.2667e+00  1.4444e+00 -2.8330e-01  1.5910e-01
  1.2730e-01]
Number of unique values in column 'AvgSubAttDif': 889

Unique values in column 'AvgTDDif': [-2.61   -0.72   -0.13   ...  1.9545 -0.9818 -4.6667]
Number of unique values in column 'AvgTDDif': 1796

Unique values in column 'EmptyArena': [nan  0.  1.]
Number of unique values in column 'EmptyArena': 3

Unique values in column 'BMatchWCRank': [nan  7.  3. 13.  9.  5. 10.  8. 11. 14. 15.  2. 12.  1.  6.  4.]
Number of unique values in column 'BMatchWCRank': 16

Unique values in column 'RMatchWCRank': [ 0.  3.  2. 13. nan 12. 14.  5.  8.  9. 15.  1. 10.  7.  6. 11.
4.]
Number of unique values in column 'RMatchWCRank': 17

Unique values in column 'RWFlyweightRank': [nan  3. 13.  0. 15.  6. 12.  2.  8.  4.  9. 14. 11.  5. 10.  7.
1.]
Number of unique values in column 'RWFlyweightRank': 17

Unique values in column 'RWFeatherweightRank': [nan  0.]
Number of unique values in column 'RWFeatherweightRank': 2
```

```
Unique values in column 'RWStrawweightRank': [nan  2. 13.  6.  9.  7.  3. 14. 11.  0.  5. 10. 12. 15.  8.
4.  1.]
Number of unique values in column 'RWStrawweightRank': 17

Unique values in column 'RWBantamweightRank': [nan  0.  2. 15.  5. 12.  8.  3. 11.  6. 13.  4.  9.  7.  1.
14. 10.]
Number of unique values in column 'RWBantamweightRank': 17

Unique values in column 'RHeavyweightRank': [nan  2.  0.  9. 13. 10.  8.  1.  3.  7. 12.  5. 15.  6. 14.
4. 11.]
Number of unique values in column 'RHeavyweightRank': 17

Unique values in column 'RLightHeavyweightRank': [nan 12.  8.  1.  0. 14. 10.  7. 15. 11.  3.  9. 13.  5.
6.  2.  4.]
Number of unique values in column 'RLightHeavyweightRank': 17

Unique values in column 'RMiddleweightRank': [nan  3. 13. 10.  4.  5.  0.  1.  6. 15. 11.  8. 14.  2.  7. 1
2.  9.]
Number of unique values in column 'RMiddleweightRank': 17

Unique values in column 'RWelterweightRank': [nan  3. 14. 15. 10.  9.  6. 12.  0.  7. 11. 13.  4.  8.  5.
1.  2.]
Number of unique values in column 'RWelterweightRank': 17

Unique values in column 'RLightweightRank': [nan  2. 11.  5. 15.  0.  1. 10.  3.  6. 13.  4. 12. 14.  8.
7.  9.]
Number of unique values in column 'RLightweightRank': 17

Unique values in column 'RFeatherweightRank': [nan 13.  5.  0. 12.  3.  6.  8.  4. 11. 10.  1.  9. 15.  2.
14.  7.]
Number of unique values in column 'RFeatherweightRank': 17

Unique values in column 'RBantamweightRank': [nan  3. 13. 10.  0.  2.  4.  8.  7. 15.  6.  9. 11. 12.  1.
5. 14.]
Number of unique values in column 'RBantamweightRank': 17

Unique values in column 'RFlyweightRank': [ 0. nan 14.  2.  7.  1. 11.  4.  6.  5.  9. 12. 10.  3. 15.  8.
13.]
Number of unique values in column 'RFlyweightRank': 17

Unique values in column 'RPFPRank': [11. nan  3.  7.  4.  2.  8.  6.  1. 12. 14. 10. 15.  5. 13.  9.]
```

```
Number of unique values in column 'RPFPRank': 16

Unique values in column 'BWFlyweightRank': [nan 11.  5. 14.  1.  8. 13.  3.  6.  9.  4. 12. 10. 15.  7.
2.]
Number of unique values in column 'BWFlyweightRank': 16

Unique values in column 'BWFeatherweightRank': [nan  0.]
Number of unique values in column 'BWFeatherweightRank': 2

Unique values in column 'BWStrawweightRank': [nan 10. 14. 11.  5. 15. 12.  1.  6.  7. 13.  2.  9.  4.  8.
3.]
Number of unique values in column 'BWStrawweightRank': 16

Unique values in column 'BWBantamweightRank': [nan  3.  8. 14. 11.  7. 10.  9. 13. 15.  5.  6. 12.  1.  4.
2.  0.]
Number of unique values in column 'BWBantamweightRank': 17

Unique values in column 'BHeavyweightRank': [nan  3.  8. 15. 12.  9.  4.  5. 13. 10.  7. 14.  1. 11.  6.
2.]
Number of unique values in column 'BHeavyweightRank': 16

Unique values in column 'BLightHeavyweightRank': [nan 13. 10.  8. 15.  9.  1.  7. 11.  3. 12.  5.  6. 14.
4.  2.  0.]
Number of unique values in column 'BLightHeavyweightRank': 17

Unique values in column 'BMiddleweightRank': [nan 13. 14.  8. 12.  2. 10.  7.  6. 11.  5.  4.  1. 15.  9.
3.  0.]
Number of unique values in column 'BMiddleweightRank': 17

Unique values in column 'BWelterweightRank': [nan  7. 15. 11.  8.  2. 14. 10.  3.  6.  4. 13.  5.  1.  9. 1
2.]
Number of unique values in column 'BWelterweightRank': 16

Unique values in column 'BLightweightRank': [nan  7.  5. 15. 11.  4. 13.  8.  9.  3. 12.  6. 10.  1. 14.
2.]
Number of unique values in column 'BLightweightRank': 16

Unique values in column 'BFeatherweightRank': [nan  9.  2. 14. 13. 10.  4.  3.  0. 12.  8. 15.  1.  5.  6.
7. 11.]
Number of unique values in column 'BFeatherweightRank': 17

Unique values in column 'BBantamweightRank': [nan  5. 12. 11.  1. 10.  6.  2.  7.  3. 15. 14. 13.  8.  9.
```

```
4.  0.]
Number of unique values in column 'BBantamweightRank': 17

Unique values in column 'BFlyweightRank': [nan  3. 13. 14.  5. 15.  7.  8. 10. 12.  2. 11.  1.  6.  9.  4.]
Number of unique values in column 'BFlyweightRank': 16

Unique values in column 'BPFPRank': [nan  6. 11.  3. 13. 10. 14. 15.  8.  2.  4.  5.  1.  9.  7. 12.]
Number of unique values in column 'BPFPRank': 16

Unique values in column 'BetterRank': ['Red' 'neither' 'Blue']
Number of unique values in column 'BetterRank': 3

Unique values in column 'Finish': ['SUB' 'U-DEC' 'S-DEC' 'KO/TKO' 'M-DEC' 'DQ' nan 'Overturned']
Number of unique values in column 'Finish': 8

Unique values in column 'FinishDetails': ['Rear Naked Choke' nan 'Elbows' 'Punches' 'Anaconda Choke' 'Armba
r'
 'Punch' 'Kick' 'Elbow' 'Spinning Back Kick' 'Guillotine Choke'
 'Arm Triangle' "D'Arce Choke" 'Spinning Back Fist' 'Knee' 'Heel Hook'
 'Triangle Choke' 'Knees' 'Triangle Armbar' 'Spinning Back Elbow'
 'Flying Knee' 'Neck Crank' 'Kimura' 'Slam' 'Kicks' 'Ezekiel Choke'
 'Twister' 'Other - Lock' 'Kneebar' 'Inverted Triangle' 'Von Flue Choke'
 'Keylock' 'Scarf Hold' 'Straight Armbar' 'Ankle Lock' 'Other - Choke'
 'Peruvian Necktie' 'Omoplata' 'North-South Choke' 'Injury']
Number of unique values in column 'FinishDetails': 40

Unique values in column 'FinishRound': [ 2.  5.  3.  1.  4. nan]
Number of unique values in column 'FinishRound': 6

Unique values in column 'FinishRoundTime': ['2:05' '5:00' '0:39' '3:21' '4:46' '0:52' '4:51' '1:56' '3:41'
'3:49'
 '2:25' '2:36' '0:46' '1:28' '4:29' '2:44' '2:55' '0:40' '4:50' '1:44'
 '1:15' '0:51' '4:19' '1:23' '1:20' '2:28' '4:28' '3:43' '0:59' '1:34'
 '3:34' '4:52' '1:30' '3:14' '4:27' '2:22' '4:33' '1:42' '2:50' '4:06'
 '2:15' '2:12' '4:32' '2:17' '1:35' '3:19' '3:47' '0:27' '2:59' '3:52'
 '4:02' '3:02' '3:59' '0:29' '4:14' '3:28' '1:14' '1:49' '4:39' '4:12'
 '1:02' '3:38' '4:04' '4:56' '0:48' '2:39' '1:55' '3:44' '3:36' '3:18'
 '4:45' '1:00' '3:22' '3:12' '3:23' '4:48' '1:36' '2:58' '1:05' '3:58'
 '4:49' '0:20' '0:18' '3:27' '1:46' '0:13' '1:58' '1:25' '1:22' '0:19'
 '2:27' '1:52' '1:50' '2:00' '4:47' '2:42' '2:47' '3:50' '0:37' '3:16'
 '0:49' '0:12' '3:00' '0:54' '2:10' '3:35' '4:42' '2:16' '2:09' '1:29'
 '4:59' '3:17' '1:47' '4:11' '2:13' '3:06' '2:18' '4:00' '4:43' '0:22'
```

```
 '1:32' '0:21' '2:45' '2:52' '4:08' '3:54' '4:23' '1:43' '1:38' '0:15'
 '2:32' '0:36' '1:01' '0:26' '4:41' '1:57' '4:36' '0:58' '2:04' '0:44'
 '3:32' '1:41' '1:18' '2:43' '0:55' '4:03' '0:32' '2:03' '4:21' '1:17'
 '3:25' '4:55' '3:29' '1:03' '4:31' '3:42' '4:25' '0:56' '1:04' '2:49'
 '4:10' '4:17' '1:11' '0:38' '3:53' '4:24' '1:31' '1:09' '3:15' '3:08'
 '4:01' '2:14' '2:07' '1:13' '3:03' '2:26' '1:06' '2:23' '0:33' '4:40'
 '4:15' '3:26' '4:44' '4:37' '3:46' '2:20' '3:51' '0:23' '1:24' '3:01'
 '4:16' '2:40' '2:19' '3:11' '4:54' '2:37' '2:11' '4:57' '4:26' '4:05'
 '1:07' '0:34' '3:10' '0:17' '4:20' '2:54' '4:13' '1:48' '1:45' '2:57'
 '4:09' '0:14' '1:10' '1:40' '4:34' '4:22' '2:51' '1:37' '3:13' '2:02'
 '3:05' '2:56' '0:28' '3:04' '2:29' '4:30' '3:20' '3:24' '1:27' '2:01'
 '3:37' '2:30' '3:33' '0:47' '0:08' '4:07' '1:33' '0:30' '3:55' '1:53'
 '2:21' '2:24' '0:16' '3:39' '1:39' '2:31' '3:57' '3:09' '2:46' '4:38'
 '0:45' '0:42' '4:58' '1:08' '1:12' '2:38' '4:35' '1:16' '2:35' '0:41'
 '2:48' '3:56' '2:34' '3:45' '2:33' '0:57' '2:06' '0:25' '3:07' '1:21'
 '3:31' '0:53' '1:51' '3:48' '1:54' '3:40' '0:35' '1:59' '0:07' nan '4:18'
 '2:53' '2:41' '1:26' '0:05' '0:09' '0:50' '0:43' '1:19' '4:53' '2:08'
 '0:24' '3:30' '0:31' '0:11']
Number of unique values in column 'FinishRoundTime': 295

Unique values in column 'TotalFightTimeSecs': [ 425. 1500.  900.  639.  801.  586.   52.  591.  716.  221.
291.  229.
  529.  145.  156.  646.  688.  869.  164.  600.  175.  640.  290.  704.
   75.   51.  259.   83.  380.  748.  268.  523.  359.  694.  214.  592.
   90.  794.  267. 1342.  873.  402.  170.  246.  435.  432. 1172.  300.
  737.   95.  199.  527.  327.  779.  232.  242.  182.  239.  329.  554.
  559.  208.  374.  409.  390.  279.  552.   62. 1118.  244.  542.  296.
  648.  159.  104.  415.  224.  816.  498.  285.   60.  202.  792.  203.
  588.   96.  478.  518.   65.  238.  289.  620.   18.  207.  406.  313.
  418.   85.  382.   19.  109.  747.  148.  675.  479.  712.  710.  994.
  120.  442.   20.  287. 1362.   94.  147.  167.  230.  223.  637.  496.
  649.   12.  891.  480.   37.   54.  430.  515.  282.  395.  436.  129.
  756.  273.  689.  194. 1499.  497.  407.   89.  551.  599.  433.  486.
  738.  198.  240.  283.  420.  622.  692.  339.  321.  410.  165.  135.
  472.  248.  234.  563.  403.   98.  615.  452.  823.  336.   61. 1200.
  326.  881.  417.  276.  658.  106.  724.  137.   44.  512.  503.  101.
  678.  163.  655.  288.  543.  218.  197.  299.   32.  892.  885.  492.
  123.  861.  377.  462.  805.  295.  509.  252.   63.  571.  596.  117.
  222.  378.  565.  212.  356.  656.  539.  361.  355.   64.  169.  703.
  550.  557.  371.  638.  178.  833.  534.  889.  564.   91.  548.   69.
  495.  188.   49.  573.  241.  363.  434.  186.  127.   73.  783.  780.
  446.   66.  743.   33.  280.  423.  572.  192.  255.  506.  584.  577.
```

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 82. | 132. | 499. | 59. | 524. | 226. | 740. | 231. | 623. | 84. | 181. | 256. |
| 589. | 351. | 103. | 459. | 160. | 522. | 139. | 251. | 821. | 227. | 391. | 491. |
| 360. | 894. | 333. | 457. | 731. | 115. | 580. | 597. | 566. | 206. | 245. | 585. |
| 367. | 34. | 859. | 443. | 38. | 822. | 190. | 17. | 560. | 77. | 55. | 174. |
| 797. | 553. | 456. | 408. | 110. | 899. | 250. | 366. | 23. | 105. | 661. | 555. |
| 475. | 177. | 429. | 249. | 715. | 14. | 860. | 140. | 448. | 130. | 501. | 1270. |
| 469. | 388. | 131. | 215. | 157. | 125. | 400. | 853. | 561. | 796. | 195. | 470. |
| 544. | 567. | 872. | 112. | 136. | 883. | 383. | 124. | 1174. | 857. | 862. | 865. |
| 471. | 595. | 397. | 193. | 133. | 422. | 344. | 185. | 791. | 260. | 253. | 545. |
| 476. | 437. | 438. | 328. | 196. | 733. | 184. | 465. | 530. | 749. | 763. | 490. |
| 581. | 570. | 500. | 404. | 546. | 806. | 362. | 663. | 204. | 687. | 1321. | 365. |
| 720. | 121. | 200. | 80. | 217. | 150. | 654. | 813. | 67. | 725. | 294. | 47. |
| 308. | 349. | 484. | 92. | 247. | 587. | 866. | 88. | 122. | 93. | 30. | 568. |
| 488. | 257. | 387. | 235. | 713. | 439. | 1072. | 741. | 277. | 444. | 863. | 669. |
| 316. | 594. | 1444. | 274. | 219. | 1037. | 699. | 213. | 179. | 1051. | 721. | 837. |
| 116. | 874. | 489. | 690. | 513. | 15. | 466. | 578. | 508. | 1218. | 790. | 493. |
| 739. | 45. | 392. | 42. | 598. | 68. | 72. | 1472. | 152. | 74. | 541. | 458. |
| 275. | 485. | 832. | 671. | 142. | 502. | 76. | 317. | 134. | 155. | 41. | 851. |
| 78. | 168. | 836. | 590. | 698. | 945. | 237. | 454. | 225. | 153. | 57. | 243. |
| 58. | 126. | 325. | 582. | 644. | 338. | 346. | 507. | 389. | 700. | 264. | 421. |
| 569. | 708. | 504. | 278. | 16. | 146. | 216. | 778. | 113. | 645. | 487. | 662. |
| 201. | 48. | 381. | 270. | 108. | 830. | 318. | 482. | 211. | 314. | 886. | 511. |
| 742. | 755. | 777. | 653. | 583. | 254. | 411. | 1140. | 746. | 875. | 158. | 450. |
| 265. | 776. | 879. | 895. | 205. | 272. | 284. | 676. | 455. | 528. | 431. | 525. |
| 1058. | 714. | 820. | 35. | 401. | 451. | 574. | 330. | 97. | 774. | 119. | 7. |
| nan | 393. | 626. | 269. | 657. | 189. | 372. | 258. | 449. | 1404. | 141. | 758. |
| 824. | 773. | 161. | 764. | 876. | 162. | 22. | 114. | 370. | 111. | 183. | 39. |
| 1264. | 520. | 817. | 1419. | 628. | 154. | 414. | 398. | 40. | 789. | 517. | 616. |
| 70. | 1450. | 1496. | 633. | 376. | 849. | 86. | 629. | 841. | 29. | 723. | 834. |
| 166. | 494. | 771. | 726. | 266. | 579. | 1149. | 176. | 709. | 706. | 46. | 187. |
| 5. | 399. | 505. | 71. | 642. | 674. | 9. | 419. | 1127. | 107. | 447. | 804. |
| 50. | 25. | 854. | 172. | 481. | 757. | 292. | 36. | 441. | 26. | 233. | 53. |
| 354. | 236. | 722. | 852. | 100. | 759. | 341. | 643. | 679. | 261. | 845. | 427. |
| 1056. | 286. | 463. | 766. | 1083. | 379. | 660. | 843. | 81. | 461. | 342. | 118. |
| 526. | 43. | 87. | 882. | 297. | 593. | 369. | 352. | 533. | 138. | 775. | 761. |
| 271. | 1356. | 220. | 730. | 1338. | 180. | 460. | 933. | 736. | 998. | 228. | 263. |
| 752. | 453. | 298. | 768. | 632. | 293. | 650. | 102. | 839. | 473. | 870. | 728. |
| 685. | 718. | 128. | 819. | 842. | 1395. | 79. | 753. | 24. | 445. | 556. | 171. |
| 576. | 810. | 686. | 562. | 825. | 751. | 424. | 474. | 343. | 353. | 1131. | 1267. |
| 28. | 607. | 516. | 210. | 467. | 719. | 21. | 659. | 826. | 799. | 149. | 324. |
| 890. | 634. | 765. | 477. | 1122. | 385. | 426. | 831. | 732. | 624. | 262. | 781. |
| 880. | 143. | 745. | 618. | 13. | 340. | 538. | 682. | 173. | 871. | 347. | 696. |

```
 1410.  394.  334.  691.  386.  323.  782.  697.  850. 1092. 1345.  558.
  191.  627.  651.  549.  898.  647.   99.  760.  935.   56. 1260.  877.
  701.  348.  670.  281.  384. 1367.  795.  695.  838.  858.    8.  547.
 1138.  514.  897.  510.  846.  413.   31.  357.  412.  681. 1340.  416.
  151.  320.  827.  957.  521.  144.  856.  829.  976.  767.  625. 1346.
  786.  803.  350.  612.  440. 1389.  531.  335.  798.  808.  630.  954.
 1020. 1423.  537.  744.  887.  641.  613.  884. 1096.  693.  540. 1137.
  364.  536.  575.  664.  711.  619.  967.  311.  332.  532.  668.  814.
 1134.  464.   27.  717.  735. 1390.  375.  322.  893.  772.]
Number of unique values in column 'TotalFightTimeSecs': 862

Unique values in column 'RedDecOdds': [ 300.  250. -160. -200.  275.  450. -135.  400.  150.  200. 1200.  1
10.
   nan -110.  500.  800.  480.  185.  700.  180.  215.  550.  350. 1100.
  600.  130.  140.  225. -115. -175.  120.  650.  100. -105.  750.  240.
 -150. 1400. -140.  165. 1000.  900.  175.  330. -165.  430. -120.  125.
  370.  380. -125.  425. 1600. 2000.  320. 2200. 1300.  850.  280.  210.
  410.  360.  420.  460.  190.  440.  105.  270.  135.  260.  195.  220.
  310.  160.  170.  390.  340. 2400.  290.  230. 1500.  950.  470.  145.
 1800.  475.  188.  333.  235.  525.  285.  385.  375.  575. -145.  155.
  115.  325. -180.  625. -130. -185.  515.  245. 1050.  305.  211.  580.
  710.  265.  539.  167.  345.  255.  181.  437.  925.  395.  349.  151.
  560.  545.  490.  335.  129.  355. -121.  491. 1065. -107.  178.  465.
 -155.  163. -132.  317.  205.  855.  442.  168.  509.  510.  415.  315.
 -103.  405.  153.  690.  635.  685.  246. -225.  429.  256.  313.  177.
  520.  540.  585.  257.  579.  112.  670.  142.  660.  162.  219.  147.
  179.  156.  199.  365.  334.  378. -104.  201.  174.  229.  553.  488.
  463. -195.  286.  166.  262.  328. -325.  284.  547.  222.  207.  566.
  109.  137.  251.  234.  228.  134.  128.  352.  226.  157.  148. 1460.
  169.  412.  454.  223.  432.  164.  149.  435.  455.  306.  565.  504.
  294. 1265.  118.  574.  448.  655.  570.  709.  214.  172.  309.  447.
  705.  116.  189.  249.  610.  485.  316. 1040. -220. 1475.  106.  196.
 1094.  213.  278.  139. -172.  399.  108.  495.  252.  640.  271.  517.
  216.  457.  775.  184.  382.  362.  287.  247.  727.  595.  860.  242.
  562. -116.  323.  237.  603.  667.  176. -153. -117.  738. -108.  103.
  445.  459. -114.  561.  221.  675.  825.  423. -101.  743.  292.  535.
  538. -170. 1310.  208.  131. -127.  891.  505.  204.  544. 1125.  696.
  940.  588.  119.  526.  496.  209.  173.  755. 1075. -112.  268.  481.
  680. -154.  605.  363.  725.  144.  203.  620.  132.  331.  785.  159.
  277. -138.  133.  111.  702.  354.  197.  383.  146.  346.  626. -123.
  875.  730.  438.  830.  227. 1350.  302.  530.  318.  659.  307.  101.
  648.  348.  324.  263.  259.  114.  182.  198.  254.  296.  295.  276.
```

```
 -118.  272. -146.  244.  154.  507.  113.  122.  117.  404.  187.  123.
  723.  143.  261.  304.  243.  141. -113.  206.  506.  413.  549.  436.
  885. -128.  787.  569.  314. 1190.  477. 1209. 1175.  347. 1201.  386.
  634.  529.  751.  381.  493.  192.  359.  351.  630.  232.  961. -148.
  718.  449.  303.  543.  499. -111.  297.  183.  804.  191.  161. -102.
  236.  472.  291. 1390. 1450. 1080.  138.  451.  233.  554.  642. 1550.
  289. -124.  288. -137.  373.  706.  282. 1420.  311.  818.  194.  171.
  274.  258.  422. 1025.  434.  337.  212.  293.  476.  721.  224.  102.
  609.  152.  193.  584.  469. 1440. 1250.  253. 1008.  158.  266. 1090.
  845.  267.  202.  564.  443.  651.  589.  674.  557.  124.  444.  518.
  336.  398.  541.  394.  401.  890.  353.  241.  645.  416. 1150.  951.
  358.  406.  441.  555.  615.  218.  633.  599.  372.  327.  767.  369.
  301.  753.  498.  567.  798.  377.  239.  299.  894.  269. -440. 1365.
 1085.  473.  217.  341.  402.  656.  321.  417.  840.  468.  329.  552.
  548. 1195.  503.  631.  281.  388.  339.  487.  248.  708.  666.  613.
  464.  357. 1019.  121.  868.  387.  644.  127.  695.  497.  638.  606.
 1003.  572.  264.  596.  136.  471.  581.  524. 1358. 1005.  186.  652.
  104.  733.  686.  407.  322.  837.  716.  424.  326.  344.  384.  534.
 1260. 1700.]
Number of unique values in column 'RedDecOdds': 578

Unique values in column 'BlueDecOdds': [ 800.  650.  450. 1100.  550.  750.  130.  275.  185.  200.  250.
225.
  700.  215.  175. 1000.  240. 1400.   nan  150. -195. 3000.  100. 1200.
  300. -150.  180.  500. -110.  350.  400.  140. -125. -115. 1800.  120.
  165.  110. -160.  330. 2800.  600. -120.  230.  900. 2000. -105.  195.
 -135.  125.  425.  320. 1600.  850. 2200.  475.  380. 1300.  220.  310.
  155.  270.  170.  420.  210.  340.  950. 1500.  460.  260.  410.  360.
  290.  190.  280.  430.  470.  115.  135.  145.  480. 1900. 2600. -190.
  105. -200. 2500. -140.  375. -165.  525.  625.  325.  255.  675.  235.
  575.  295.  365. -175.  160.  725.  383.  645.  740.  285.  248. 1040.
  925. 1166. -170. 1250.  423. -185. -145. 1075.  345.  355.  251.  540.
  655. 1165.  265.  485.  415.  510.  524.  570.  177.  560.  724.  258.
  245.  422.  257.  945.  335.  435.  975.  205. -182.  168.  565.  244.
  315.  515.  535.  382.  440.  370.  674.  520.  445.  390.  783. -130.
  211. 1150.  279. 1217.  385.  465.  317.  588.  223.  163.  326. -128.
  585.  127.  432.  875.  506.  316.  690.  259.  274.  156.  349. -123.
  640.  138.  143.  698.  328.  158.  284.  442.  174.  218.  169. 1309.
  760.  348.  286.  494.  192.  221.  775.  241.  484.  592.  141.  839.
  323.  281.  159.  299.  735.  395.  391.  705.  289.  503.  514.  519.
  183.  882.  294.  293.  116.  424.  273.  178.  555. 1025.  715.  243.
  237.  171.  202. -142. -136.  249.  329.  176.  204.  630.  499.  233.
```

```
 -172.  680.  509.  660.  474.  920.  942.  137.  479. 1350.  589.  726.
  635.  108.  217.  572.  164. 1700.  444. -155.  336.  333.  551.  402.
  505.  102.  426.  271.  371.  522.  157.  363. 1050.  327.  236.  405.
  111.  246.  287.  865.  610.  711.  161.  148.  153.  187.  825.  377.
  782. 1235. 1750. -103.  539.  688.  347.  254.  563.  512.  227.  264.
  139.  427.  128.  679.  129.  282.  203. 1255.  229. -104.  890.  272.
  835.  107.  529.  595.  748.  112.  576.  481.  777.  232.  126.  252.
 2300. 1650.  940. 1225.  785.  401.  916.  545.  437.  678.  214. -112.
  278.  860. 1125.  603.  222.  228.  314.  791.  692.  172.  490.  452.
  431.  376.  590.  805.  301. -121.  720.  730.  197.  142.  304.  662.
  792. 1110.  253.  298.  312.  580. -101.  491.  283.  318.  477.  234.
  343.  615.  547.  710.  291.  344. -114.  351.  483.  605.  242.  626.
  219.  467.  369.  247.  616.  937.  268.  922.  536.  352.  461.  541.
  324.  308.  917. 1108.  904.  508. 1450.  121.  332.  446.  364. 1270.
  179. 1015.  419.  266.  454.  561.  118. 1189.  334.  831.  584.  397.
  632.  403.  408.  362.  830.  620.  753.  502.  543.  113.  429. 1271.
  486.  822. 1149.  166.  305.  199.  418. 2100.  765. 1441.  434.  463.
  338. 1420.  146. -180.  109.  269.  495.  594.  386.  778.  207. 1005.
 1487.  880.  577.  530. 1262.  648.  378.  346. 1065.  162.  353.  361.
  411. 1002.  216. 1550.  149.  517.  224.  154.  196.  593.  191.  553.
 1055.  358.  459.  381.  621.  321.  606.  456.  262.  554. 1440.  194.
  990.  763.  201. 1090.  492.  957.  910.  609.  104.  357. 1133. 1155.
 1252.  421.  167.  319.  331. 1272.  671.  614.  416.  379.  438. 1030.
  523.  556.  842.  528.  855.  845. 1395.  374.  721.  354. -124.  624.
  209.  677.  723.  685.  549.  433.  436.  297. 1850. 1185.  658.  114.
  356.  288. 1081. 1160. -161.  306.  885. 1372.  181.  747.  193.  738.
 1175.  516.  768.  847.  802.  184.  261.  367.  341.  188.  695.  776.
  404.  591.  231.  546.  573.  417.  239. 1370.  277. 1445.  569.  173.
  226.  406. -108.  412.  716.  206.  359. 1022.  368.  389.  147.  366.
  568.  471. 1385.  256.  722.  769.  303.  384.  428.  119. 1394.  117.
 1335.  761.  488.  396.  526.  238.  263.  829.  489.  212.  439.  213.
 1006.  372. 1406. 1142.  322. 1430. 1312. 1117.  186. 1123.  394.  578.
  513.  814. 1950.  123.  732.  874.  665.  739.]
Number of unique values in column 'BlueDecOdds': 632

Unique values in column 'RSubOdds': [ 150.  180. 1100.  380.  500. 1200.  450. 1600. 1000.  550.  350. 280
0.
  700. 2000.   nan  460.  600. 2100.  800.  490.  200.  130.  175. 1400.
  900.  275.  300. 3000. 2200. 1800.  215. 2500.  400. 3500.  650.  250.
  440.  240.  750.  110.  165.  850.  280. -135.  330.  120. 4000.  225.
  100. -145.  140. -200. -125.  950.  230. 2900.  420. 2600.  170.  360.
  210. 1300. 1500.  220. 1700.  340.  125. -110.  270. 2300.  310.  260.
```

```
 145.  290.  185.  155. -105.  430. -175. -115.  475. -140.  333.  575.
 375.  625.  675.  775.  425.  325.  285.  725.  525.  135. -240.  235.
1475. 1325.  730. 1275. 1415.  652.  115.  190. 1125. 1250.  205.  160.
-150. 1150. 1020.  630.  435. 1365.  470.  610.  567.  421.  432.  448.
1030. 1375. 1083. 1015.  580.  405. -160.  948.  659. 4665. 1440.  617.
1025. 1050.  540.  825. 1090.  602. 2050.  581.  505.  845.  365.  320.
 385.  642.  667.  245. 1850. -230. -170. 1008. 1422. 2750.  545.  680.
1065. 1425. 1340.  315. 2105. 2250. 2550. 1075. 1905.  312.  515.  172.
 410. 1483. 1575. 1465.  465. 1120.  587.  585.  940. 1240. 1533.  885.
1650.  975.  595. 1265.  565. 1355.  335. 1825. 1720. 1550. -130.  705.
 765.  925.  715. 1428. 1403. 1350. 1675.  685.  970. 1450.  664.  345.
 690.  434. 1445.  875.  179.  677. 1190. 1225.  382.  755. 2400. 1460.
 645.  265.  746.  760.  859. 1655.  520.  559. 1183.  429. 1467. 1278.
 571. 1195. 1805.  815.  217.  615.  334. 1285. 1055. 2150.  560.  614.
2625.  927.  739. 1915.  337.  796.  870.  485.  535.  355. 1950.  779.
 105.  711. 1525. 2025. 1545. 1419. 2665.  880. 2320.  480.  901.  370.
 424.  542.  267.  684.  284. 1625. 1390. 1105.  920.  720. 1167. 2390.
 770. 1555.  751. 1160. 1208.  665. 1018.  865.  510.  431. -280. 1645.
1900. 1145. 1042. -165.  634. 1175.  419.  701.  569. 1245.  745. 1060.
2850. 2450. 1615. 1750.  154. 1875. 1205. 1735.  945. 1855. 1070.  524.
 794.  955.  539.  930. 1035. 1315. 3200.  735. 1455.  820. 1180.  852.
1710.  412.  795.  459. 1865.  889. 1085.  402. 2160.  734. -120. 2350.
 390.  683. 1288. 1385.  519.  157.  740.  197.  169. 1165. 1435.  537.
 935.  830.  501. 1520.  219. 1155.  837. 1530.  635. 2125.  268. 2700.
1135.  305. 1092. 1215.  278. 1490. 1010. 1320.  495. 1005.  679.  392.
 778.  826.  468. 1880.  551.  882. 1775. 1360.  620.  951.  854. 1347.
 504.  749.  805. 1255. 2225.  781.  176.  660.  457. 1095.  572. 1370.
1995.  789.  563.  914.  960.  824.  924.  804. 1170.  349.  827.  195.
 511. 1910. -155. 1363.  556.  621. 1130. 1405. 1142.  674. 1755.  483.
 454.  941.  967. 1185.  487.  627.  761.  699.  554.  547. 1670.  639.
 892.  931.  631.  633.  703. 1080.  590.  328.  243.  810.  724.  273.
 493.  952.  916. 1117.  478.  586.  527.  295.  913. 1140.  806.  404.
1115.  840.  514.  456.  544.  502.  822.  121. 1253.  655.  777. 1920.
 331.  488.  754.  710. 2005.  708. 1108. 1335. 1343.  965.  876.  442.
1310.  536.  605.  415. 1510.  561. 1975.  194.  771.  244. 1045. 1620.
 426. -111.  474.  939. 1695.  811. 1955. 1725.  681.  607.  758.  646.
1220.  351.  928.  414.  449.  910.  670.  202.  768. 1940.  721.  788.
1303. 1470.  682.  932.  526. -185.  445. 1705.  366.  453.  718. 2055.
 329. 2075.  416.  521. 2115. -180.  254.  286.  640.  626. 1408.  596.
 451.  255.  531. 1570. 1480.  649. 1505.  241.  878. 1290.  354. 1110.
 582.  594.  362. 1295.  706.  689.  693. 1820. 1760.  861.  417.  324.
1595.  401.  619.  588. 1067. 1447.  439.  395.  506.  530.  766.  253.
```

```
  344.  628. 1980.  443.  376.  306.  386. 1430.  507.  943.  784.  313.
  346. 1481.  576. 1990.  555.  743.  477.  558.  471.  629.  713.  222.
  570. 1260. 1860. 1630. 1330. 2770.  455.  656.  304. 1305.  562. 1270.
  688.  589.  842. 1230.  311. 1685.  801. 1790. 1835.  484.  654.  722.
  906.  942.  785.  422. 1235.  835. 1970.  479. 2760. 1268.  807.  178.
 1040. 1845.  319.  601.  873. 1210. 1380. 1895. 1478. -275.  695.  606.
 1665.  926. 2020.  897.  896. 1680. 2640. 1795. 1433. -300.  129.  272.
  428.  258. 1640.  336. 1456. 1410.  874.  134.  199. 3300.  361. 1770.
 -370.  323.  973. 1387.  242.  387. 2295.  564. 2830.  989. 2470. 1263.
  236.  676.  637. 1133. 1280. 3450.  522. 1367.]
Number of unique values in column 'RSubOdds': 692

Unique values in column 'BSubOdds': [ 2500.  3000.  1400.   700.  1200.   900.  1100.   550.  1800.   800.
  -165.  1600.    nan   400.  1000.  3500.   300.  4000.  2200.   165.
   110.   600.   450.  2000.  2800.   240.   350.   500.   225.   650.
  5000.   175.   250.   120.   750.   275.   125.   180.  1900.  1500.
   140.  4500.   200.   100.   330.   150.   950.   460.  1300.   280.
  2600.   380.   260.   390.  2300.   850.  1700.  3100.   210.  -105.
   440.   270.   320.   470.  3200.   340.  3300.   370.   420.   190.
  -110.  3400.   215.  -125.   925.   425.   375.   325.   675.   575.
   170.   525.   475.   775.   825.   625.   185.  1625.  1465.   445.
  1125.   875.  1250.   605.  1365.  -145.  2150.   725.   785.   970.
  1750.  2400.   244.   405.   567.  1370.  1045.   955.   534.  -140.
   455.  1285.  1350.  2050.  4785.   860.  3550.   505.   385.   885.
   614.   930.  1278.   335.  1150.   535.  4200.  1225.  2550.   360.
  1450.  1440.  1575.  1360.   665.   590.   502.  2250.  1815.  1310.
   730.  1065.  1995.   830.  1550.  2875.   640.  1675.  1095.  1408.
  1008.  3600.   965.   195.   935.  1660.  1050.  1670.   715.   410.
  1422.  1330.  1075.  2475.   602.   720.   585.  2750.  2320.  1055.
  1305.   555.  -115.   205.   934.  1425.  1475.  1005.   680.  1615.
   285.  1480.   160.  1795.   310.   705.  1275.   515.   685.  1175.
  1240.  2090.  1745.   615.   545.   435.   315.  1680.   815.   542.
   464.   645.  1565.   510.   760.   635.   855.  1165.   977.   954.
   915.  2100.  1270.  1155.  1705.   601.   489.   577.  1655.  1025.
  1505.  1263.   451.   895.  1755.   387.  1455.  1850.  1185.  1340.
   364.   840.  2450.  1015.   941.   962.   757.  1535.   735.  1820.
  1375.  1315.  2015.   173.  -175.   245.   740.   660.   920.   522.
   557.  1431.  1650.   743.   698.  1040.   345.  1515.   395.  1458.
  1397.   619.  1980.   710.  1260.   869.   584.  1145.   520.  1833.
   905.  3425.  1115.  1355.  1120.   485.   690.  1325.   870.  2900.
  1495.  1785.  1780.  1445.   220.  1570.  1135.   365.   611.   620.
   457.  1845.  1265.  1345.   610.   789.   290.   995.  1435.   820.
```

```
  940.  1205.  -130.   810.   465.   865.   570.   630.   415.  1983.
 1387.  1530.   530.   495.   898.   265.  -120.   701.  2350.   958.
 1430.  1486.   416.   787.   129.   684.   441.   540.   765.  1060.
  906.   564.   695.   115.  1010.   980.  1410.  1775.   493.  1245.
  945.  1085.   755.   724.   772.   158.   541.   776.   366.  1195.
 1220.  1825.   617.  1630.   382.  1695.   823.   277.  1067.   975.
 2020.  1170.   226.   586.  1353.   878.   430.  1335.  1390.  1190.
  321.   131.   230.  1470.   174.   295.   433.   556.  1405.   155.
  655.   429.   921.  1255.   880.  1585.   539.  1110.   862.  1142.
 1160.  2165.   686.   612.  1080.   722.   444.  1605.   346.  2700.
  717.   459.   714.   349.   446.   319.   177.   235.   419.   842.
 1725.   311.   835.  1683.  1105.   609.   595.   507.   571.   504.
  946.   514.  1235.   592.  1092.   846.   872.   652.   606.   558.
  516.  1580.   582.   537.   845.   524.   481.   960.  1280.   436.
 1950.   422.  1210.   916.   745.   756.   939.   716.   363.  1295.
 1378.   670.  2275.  1945.   837.   682.  1130.   983.   371.   649.
  284.  2950.  1665.   953.   565.   910.   985.  2675.   866.   432.
  490.   636.   467.  1485.  1510.   461.  1930.  1520.  2825.   911.
 1635.   769.  2045.  1905.  2255.  1920.  1720.   576.   859.  1955.
 2155.  1290.  1208.   754.   768.  2425.   974.   951.   316.  1090.
 1467.   651.  1318.  2295.  1395.   604.   468.  1033.  1030.  1380.
  672.   713.  1180.   562.   731.   942.   795.   642.   304.   546.
 1765.  2740.   737.  1835.   676.   358.   709.   914.   384.   551.
  770.  1545.   626.   353.   952. -1250.  2095.  1975.  1908.  2640.
  766.   634.  2615.   326.   791.  2105.   889.  1420.   873.  1215.
  808.   781.  1935.  1385.  1970.  1880.   874.  2355.   482.   792.
  538.   381.  2360.  1307.   664.  2665.   531.   414.  2365.  2530.
  479.  1610.  2860.  1462.  1640.  1460.   105.   856.   628.   454.
 1770.  1965.   959.  2170.   805.   135.   574.  1860.  1303.   242.
 2030.   403.  2120.  1910.   834.   580.  2040.  1915.  1790.  2540.
  972.  1540.   790.   202.  2650.   563.  1620.  1035.  1472.   431.
  596.   990.  2785.  2830.   402.   944.   913.  1865.  1242.  1042.
  501.  1140.   599.   931.   832.   578.   788.  1805.  1735.   526.
  145.   305.  1415.   666.   764.  2065.   484.  1685.  2220.  2240.
 1608.   809.  2525.   591.   294.   423.   469.   763.   739.   271.
 2205.   264.  2125.  2575.   411.   573.   583.   899.   497.  2445.
 1369.   547.  3375.  2775.   747.   699.   632.   631.  2625.   332.]
Number of unique values in column 'BSubOdds': 670

Unique values in column 'RKOOdds': [ 400.  240.  350.  500.  300. -125.  600.  550.  450.  650.  900. 1600.
 2200. -150.  200.   nan  280. -185. 1100. 3600.  130.  700.  120.  215.
 2000. -120. 1200. 1400.  380. -250.  800.  225.  165.  150. -175.  250.
```

```
 180. -225. -295. 4000. -200.  185.  275. 1000.  340.  750.  100.  175.
1800.  110.  330. -165. -140. -190. -135.  140. 2500. -300. -110. -145.
-230.  950.  115.  850.  135. -130.  290.  270.  470.  195.  420.  155.
1700.  460.  145.  210.  320.  390.  310.  125. -180.  160.  360.  170.
-115.  440.  480.  260.  410.  370. 1500.  190.  230.  220. 1300.  490.
 475.  425.  333.  163.  525.  295.  325.  375.  235.  435.  625.  285.
 575.  675. -160. -275.  105.  205.  315.  725. 1025.  825. 2100.  585.
 305.  615. 1125. -155. 1250. -210. -170.  780. 1405. 1450.  560. -105.
 960.  926.  495.  255.  508. -245.  655.  584. 1230.  515.  642. -122.
 875. 1050. 1350.  686.  705.  124.  640.  620.  539. 1180.  201.  415.
 364. 1035.  405.  245.  794.  935. 1265. 1565.  667. 1145. 1150.  870.
 660.  434.  499. 1675.  565.  775.  317. 1175. 1325. 1900.  513.  685.
1550.  830.  605.  737.  335. 1020.  430.  485.  925. 1375.  915.  510.
 599. 1075. 1215.  690.  276.  520. -365. 1275.  529.  942.  277.  302.
-260.  710.  681.  626.  213.  975.  612.  835. 1345. 1390.  692. 1415.
 534. 1315.  545. 1635.  439.  540.  899. 1318. 1010. 1208.  643.  999.
 590.  251.  421.  595.  342.  306. 1293.  254. 2255.  630.  166.  868.
 283.  552.  473.  102.  236.  345.  384. 1040.  311.  507.  758.  402.
 933.  970.  385. 1695.  815.  535. 1490.  714.  265. 1080.  381.  127.
-147. 1290.  613.  521.  592. -400. -111. 1260.  580. 1445.  796.  506.
1110. 1380. 1655. 1875.  484. 1220.  249.  884.  514.  527.  111.  109.
-107. -128.  294.  777. 2335.  708.  178.  422.  855. 1245.  173. 1650.
1105.  985.  444.  680.  365. 1030.  411. 1225. 1065.  412. 1330. 1425.
 940.  129.  805.  635. 1465.  252.  792.  303.  324.  890.  781. 1015.
1665. 1235.  148.  319.  447.  765.  820.  912.  464.  454. 2075. 1158.
1965. 1460.  610. 1280.  346.  945. 1540.  279.  119.  957.  670.  930.
1710.  395.  658.  357. 1625.  799.  366.  530. 1170. 1070. 1447.  811.
1085.  562.  810.  272.  158.  790.  952.  860.  171.  452. -108.  465.
 845. 1130.  401.  489. 1268.  459.  449.  882.  596. -162. 1115.  570.
 911.  574. 1120.  547.  683.  224.  437.  924.  394. 1408. 1745. 1045.
1455.  445.  716.  207.  631.  665.  840.  271.  348.  397.  467.  463.
 505.  524.  542.  941. -240.  461. 1090.  259.  879.  588.  770.  719.
 956.  877. 1210.  910.  706. 1008.  668.  378.  664.  387.  334.  785.
 322.  501.  955.  312.  177.  307.  809.  273.  865.  423.  257.  455.
 466.  885.  488.  688. 1417. -146.  907.  114.  204.  731.  715.  571.
1670. 1520.  226. -270. -103.  696.  676. 1135. 1475.  281.  554.  991.
 608.  332.  798.  581.  229. 2305. 2300. 1530.  451. 1060.  179. 2675.
 652. 2370. 1340.  904.  474.  172.  301.  377.  286.  755. 1195. 1033.
 349.  274.  555. 1410.  151. 1610. 1420. 1160.  189.  482.  771.  632.
 379.  242.  424. 1255.  448.  169.  347. 1295. 1780.  734.  678.  704.
 462. 1486. 1525.  363. -320.  438.  543.  654.  821.  849.  645. -550.
 483.  778.  181. 1005. 1285.  323.  371.  353. 1385. -131.  786. 2225.
```

```
  601. 1880. 1720.  289. 1055.  814.  112.  813.  457.  735.  243. 1185.
 1431.  905. 1360. -127. 2195.  627. -119.  328. 1095.  760. 1740.  901.
 1915.  839. 2475.  745. 1042.  880. 2650.  304.  187. 1750. 1242. -380.
  589. 1440.  436. 1320.  182. -106.  293.  856. 1155.  578.  149. 1515.
 1117.  672.  416.  874.  359.  533.  282. -265.  649. 1690.  427. 1270.
 -121.  843. 1392.  604.  634.  990. 1480.  628.  253.  893.  141.  894.
  689.]
Number of unique values in column 'RKOOdds': 613

Unique values in column 'BKOOdds': [ 350.  700. 1100. 4000.  250.  800.  450. 1600.  300.  380.  225. 1200.
 1000. 1800.  200. 2200.  130.   nan  550.  750. 3000.  500.  165. -400.
  650.  175.  240.  900.  330.  600. -250.  280. 1400. 2000. -110.  120.
 2500.  400.  180.  275. -160.  230.  475. -280. -190.  110.  185. -120.
  320.  425.  215.  140. -225.  150.  105. -175. -125. -105.  100. -150.
  440.  260.  220.  420.  135.  190.  470.  460.  950.  310.  210.  125.
  360.  270.  850. 1300.  390. 2100.  115.  340.  160. 1700.  370.  155.
 1500.  145.  290. 2300. 2800. -165. -135.  170.  333.  265. 1250. -145.
  775.  325.  285.  235.  205.  725.  375.  575.  625. -130.  315.  525.
 -220.  675.  365.  435. 1085.  402.  737.  515.  921. 1375.  540.  795.
 1125. 1900.  417. 1145.  295. -115.  483.  245. 1120.  247.  410.  865.
  805.  683.  353.  335.  695.  560.  905.  305.  492.  382. 1180.  965.
  890.  765.  780.  520. -116.  767.  570. 1405.  860.  490. 1385. 1745.
 1050.  415. 1265. 1450.  455.  640. 1425. 1190.  345. 1675.  445.  480.
  545.  505.  880.  590. 1750.  430.  510. 2150. -210.  499.  875.  960.
  935. 1195. 1369. 1215.  534.  355. 1280. 1150.  715. 1350.  820.  404.
 1230. 1075.  485. 1550. 1565. 1715. 1010.  576. 1315.  385. 1485.  484.
  620.  177. 1515. 1160. 1387.  565.  138.  178.  508.  594. 1030. 1055.
 1590.  985.  255.  316.  663.  779.  945. 1263.  278.  555.  328.  405.
  853.  660. 1170. 1472.  845.  236. 1525.  760.  495. -138. 1183.  331.
 1353.  482.  352.  467.  302.  228. 1275.  599.  710.  645.  434. -122.
 1135. 1025. 2065. 1365. 1095. 1950. 1940. 1282.  615.  535.  119.  870.
  790. 1455.  199.  753.  585. 1430.  248. 2060. -140.  273. 1775. 1475.
  579. 1645.  942.  610. 1270.  971.  451.  121. 2020.  179.  658.  401.
 1245. 2975.  569. 1020.  927.  649. 2400. -215. 1445. 1260.  453. 3200.
  168.  258.  637.  798. 1760. 1325.  764. 1105. 1340. 1885. 1130.  118.
 1360. -126.  840.  785. 1855.  770.  395.  925.  641.  446.  712. 1285.
  617.  665. -155.  915.  655. 1155.  609.  735.  293.  472.  137.  132.
  626. 1035. 1140.  670. 1845.  825.  174.  259.  465. 1380. 1370. 1440.
  990.  975. -230.  142.  881. 1288. -169.  702.  616.  206. -185. -149.
 1330.  509.  481. 1740. 1175.  885.  372.  436.  463.  716.  195. 1045.
  676.  516.  136. 2550.  263.  878.  479. 1335.  471.  842. -123.  815.
 1165.  244.  639.  830.  911.  448. -172.  751.  685.  855. 2700.  396.
```

```
 2350.  605.  634.  595.  117.  872.  728.  506. 1560.  524.  251.  112.
 1635.  745. 1255. 1090. 1185. 1520. -200.  332.  541.  948.  602.  653.
 1060.  644.  601.  636. 1225.  632.  920.  816. 1205.  504.  257.  547.
  167.  201.  421.  172.  877.  409.  354.  684. 1220.  679.  294.  588.
  635.  862.  329.  692.  774.  580.  503.  813.  720.  539. -240.  680.
  951. 1395.  216.  787.  243.  910. 1410.  464.  727. 2105.  607.  930.
  686.  705. 1875. 1725. 1615.  582.  542.  348.  530.  571. 1390. 1108.
  690. 1625. 2455. -180. 1435.  731. 1850. 2005.  981. 2170.  242. 1005.
 1305.  474. 1650.  391. 1080. 1930.  598.  980. 1995.  687.  755. 1555.
 1320.  689. 1575.  486. 1580. 1680.  847.  841.  317.  630.  109. 2250.
  859. 1378. -275. 1040.  894. 1065.  810. 2075.  442.  253.  307.  698.
 2050. 1655. -131. 1470.  707.  769.  718. 2025.  603.  361. 2110. 1730.
  336.  749.  721. 2525.  217. 1925. 2255. 2650.  730. 1412. 1115. 1905.
 2225.  212. 1257.  426. 2145.  827.  869.  462. -195. 1420.  627.  187.
 1332.  783.  662. -154.  276.  928.  291. 2470.  424.  514.  561.  366.
  574.  529.  606.  586.  564. 1167.  254.  171. 1240.  469. 1431. 1486.
  706. 2630.]
Number of unique values in column 'BKOOdds': 590

Unique values in column 'WeightClassLabel': ['Featherweight' "Women's Strawweight" 'Light Heavyweight' 'Hea
vyweight'
 'Catch Weight' 'Welterweight' 'Lightweight' 'Flyweight'
 "Women's Featherweight" "Women's Bantamweight" 'Bantamweight'
 "Women's Flyweight" 'Middleweight']
Number of unique values in column 'WeightClassLabel': 13

Unique values in column 'BlueHeight': [68.         75.         79.         69.         70.         76.
 73.         67.         65.         71.         61.         66.
 72.         64.         62.         78.         74.         63.
 77.         60.         67.71653543 83.         80.        ]
Number of unique values in column 'BlueHeight': 23

Unique values in column 'BlueReach': [69.         74.         80.         70.         76.         77.
 71.         75.         65.         78.         68.         61.
 66.         79.         67.         73.         72.         63.
 81.         62.         84.         58.         64.         60.
 59.         83.         82.         67.71653543  0.         70.47244094
 70.07874016 75.5        68.5        71.5        77.5        78.5
 74.01574803 80.31496063 72.83464567 61.81102362 62.99212598 71.65354331
 72.04724409 64.96062992 75.98425197 68.11023622 70.86614173 72.44094488
 66.14173228 68.8976378  66.92913386 81.1023622  77.16535433 64.17322835
 79.52755906 74.80314961 69.5        70.98425197 74.48818898]
```

```
Number of unique values in column 'BlueReach': 59

Unique values in column 'RedHeight': [65.         73.         76.         70.         69.         71.
 67.         75.         74.         77.         72.         66.
 62.         64.         68.         63.         78.         61.
 60.         79.         83.         77.16535433]
Number of unique values in column 'RedHeight': 22

Unique values in column 'RedReach': [67.         77.         81.         70.         72.         75.
 78.         83.         63.         71.         66.         73.
 60.         84.         74.         76.         68.         64.
 80.         69.         62.         79.         65.         82.
 59.         61.         58.         78.5        74.5        70.5
 67.5        66.5        75.5        72.5        84.5        71.5
 83.07086614 66.53543307 68.11023622 61.81102362 77.16535433 64.96062992
 76.37795276 66.92913386 70.07874016 72.04724409 75.98425197 70.86614173
 66.14173228 72.83464567 68.50393701 74.01574803]
Number of unique values in column 'RedReach': 52

Unique values in column 'ROver35': [0 1]
Number of unique values in column 'ROver35': 2

Unique values in column 'BOver35': [0 1]
Number of unique values in column 'BOver35': 2

Unique values in column 'RWinPct': [0.8        1.         0.81818182 0.77777778 0.625      0.63636364
 0.71428571 0.72222222 0.63157895 0.6        0.51282051 0.64285714
 0.69230769 0.72727273 0.57142857 0.66666667 0.5        0.95454545
 0.6875     0.54545455 0.60465116 0.52941176 0.53846154 0.33333333
 0.85714286 0.41666667 0.46153846 0.75       0.77272727 0.90909091
 0.61538462 0.58823529 0.55555556 0.7        0.         0.25
 0.88888889 0.75862069 0.44444444 0.45454545 0.4        0.91666667
 0.68181818 0.62962963 0.83333333        nan 0.48       0.53333333
 0.64705882 0.76923077 0.65217391 0.875      0.5862069  0.58333333
 0.42857143 0.14285714 0.60869565 0.57692308 0.61290323 0.54347826
 0.76190476 0.93333333 0.61904762 0.70588235 0.9        0.62068966
 0.78571429 0.5625     0.68       0.375      0.70967742 0.47619048
 0.61111111 0.16666667 0.60606061 0.52631579 0.73333333 0.65
 0.46666667 0.86666667 0.46428571 0.67741935 0.6097561  0.42105263
 0.84615385 0.43478261 0.92857143 0.88235294 0.3        0.68421053
 0.74074074 0.56521739 0.73684211 0.2        0.65384615 0.92307692
 0.78947368 0.68965517 0.84210526 0.41176471 0.56818182 0.59090909
```

```
 0.56       0.55172414 0.82352941 0.8125     0.70833333 0.73076923
 0.59259259 0.54054054 0.95238095 0.675      0.52777778 0.30769231
 0.57894737 0.7037037  0.58139535 0.72413793 0.64516129 0.60526316
 0.35714286 0.76       0.28571429 0.45833333 0.64       0.54285714
 0.76470588 0.72       0.63333333 0.59459459 0.73913043 0.56097561
 0.60714286 0.69565217 0.38461538 0.55       0.46875    0.47058824
 0.65909091 0.4375     0.47826087 0.36363636 0.65517241 0.71052632
 0.52173913 0.68085106 0.59375    0.52380952 0.61764706 0.7027027
 0.5483871  0.80952381 0.9375     0.48387097 0.80769231 0.95
 0.70731707 0.47368421 0.67857143 0.94736842 0.65625    0.58064516
 0.89473684 0.51724138 0.54166667 0.35294118 0.82608696 0.7826087 ]
Number of unique values in column 'RWinPct': 168

Unique values in column 'BWinPct': [       nan 1.         0.75       0.33333333 0.57142857 0.55555556
 0.8        0.85714286 0.5        0.7        0.8125     0.63636364
 0.875      0.         0.6        0.77777778 0.52941176 0.54545455
 0.4        0.66666667 0.64705882 0.6875     0.375      0.53846154
 0.52631579 0.625      0.71428571 0.28571429 0.75862069 0.61111111
 0.44444444 0.81818182 0.41666667 0.42857143 0.65       0.53333333
 0.64285714 0.69230769 0.46153846 0.61764706 0.5625     0.53571429
 0.86666667 0.83333333 0.55       0.61904762 0.61538462 0.58333333
 0.2        0.82352941 0.76470588 0.45454545 0.16666667 0.3
 0.52380952 0.73333333 0.72727273 0.25       0.74193548 0.68421053
 0.76923077 0.63157895 0.45833333 0.78947368 0.68181818 0.27272727
 0.59259259 0.92857143 0.60714286 0.57692308 0.38461538 0.73684211
 0.57894737 0.9        0.58823529 0.56       0.48148148 0.52
 0.62962963 0.9375     0.88888889 0.91666667 0.72       0.84615385
 0.76       0.64444444 0.42105263 0.61290323 0.78571429 0.65384615
 0.60869565 0.73913043 0.57575758 0.47058824 0.54166667 0.63333333
 0.68       0.76190476 0.53125    0.7037037  0.65217391 0.62068966
 0.75609756 0.77272727 0.46666667 0.9047619  0.70833333 0.7826087
 0.6744186  0.73076923 0.6969697  0.90909091 0.36363636 0.56666667
 0.72222222 0.47619048 0.70588235 0.72413793 0.51851852 0.14285714
 0.69565217 0.79166667 0.56521739 0.64       0.52173913 0.59090909
 0.80952381 0.94117647 0.47368421]
Number of unique values in column 'BWinPct': 129

Unique values in column 'RedStrikingRatio': [1.         0.42827443 0.51694915 ... 0.79846449 0.54915217 0.3
3098615]
Number of unique values in column 'RedStrikingRatio': 5189

Unique values in column 'BlueStrikingRatio': [0.         0.57172557 0.48305085 ... 0.20153551 0.45084783 0.
```

```
66901385]
Number of unique values in column 'BlueStrikingRatio': 5192

Unique values in column 'RedTotalFights': [15  6 11  9  8 21 18 19 10 39 14 13  2  3  1  4  5 22 32 43 20 1
2 33 17
  7 28 24 29 16 27  0 25 23 26 31 46 30 41 45 38 35 40 44 37 36 34 42 48
 47]
Number of unique values in column 'RedTotalFights': 49

Unique values in column 'BlueTotalFights': [ 0  8 16  3  7 27  5 20 10  6 11  1  2 18 17  4 13 19 29  9 35
12 15 14
 22 34 28 21 31 24 42 26 33 25 32 45 23 30 41 43]
Number of unique values in column 'BlueTotalFights': 40

Unique values in column 'RedIsGrappler': [0 1]
Number of unique values in column 'RedIsGrappler': 2

Unique values in column 'BlueIsGrappler': [0 1]
Number of unique values in column 'BlueIsGrappler': 2

Unique values in column 'RedIsStriker': [0 1]
Number of unique values in column 'RedIsStriker': 2

Unique values in column 'BlueIsStriker': [0 1]
Number of unique values in column 'BlueIsStriker': 2

Unique values in column 'RGrapplerVBStriker': [0 1]
Number of unique values in column 'RGrapplerVBStriker': 2

Unique values in column 'BGrapplerVRStriker': [0 1]
Number of unique values in column 'BGrapplerVRStriker': 2

Unique values in column 'RedStrikingEfficiency': [ 2.1609     2.5132     3.294     ...  9.9       16.4025
9.2120196]
Number of unique values in column 'RedStrikingEfficiency': 4748

Unique values in column 'BlueStrikingEfficiency': [ 0.        3.025     2.9241   ...  3.57975  11.305017 1
4.566682]
Number of unique values in column 'BlueStrikingEfficiency': 4240

Unique values in column 'RedTDEfficiency': [1.2267    0.4321    0.1218    ... 0.5625368 3.686693  0.120828
5]
```

```
Number of unique values in column 'RedTDEfficiency': 3145

Unique values in column 'BlueTDEfficiency': [0.      0.4235  0.2835  ... 1.3125  0.6864  0.82875]
Number of unique values in column 'BlueTDEfficiency': 2543

Unique values in column 'RedEffectiveTD': [0.30651341 1.20805369 0.86206897 ... 0.5000175  0.16663611 0.214
28418]
Number of unique values in column 'RedEffectiveTD': 1767

Unique values in column 'BlueEffectiveTD': [       nan 0.38961039 0.44444444 ... 0.29731132 0.58333333 0.06
668   ]
Number of unique values in column 'BlueEffectiveTD': 1423

Unique values in column 'EffectiveTDDif': [        nan -0.8184433  -0.41762452 ...  0.82352941 -1.93362003
 -1.200012  ]
Number of unique values in column 'EffectiveTDDif': 3180

Unique values in column 'RedSize': [132.         150.         157.         140.         141.
 153.         146.         139.         152.         134.
 148.         137.         160.         128.         143.
 149.         122.         138.         131.         144.
 145.         155.         136.         124.         127.
 130.         147.         135.         154.         133.
 142.         158.         125.         156.         124.
 159.         123.         151.         126.         129.
 121.         163.         167.         154.5        146.5
 141.5        132.5        131.5        148.5        143.5
 167.5        139.5        149.5        161.         160.23622047
 142.5        132.53543307 120.         135.11023622 128.81102362
 152.16535433 129.96062992 150.37795276 131.92913386 136.07874016
 143.04724409 149.98425197 143.86614173 133.14173228 144.04724409
 143.83464567 135.50393701 149.16535433 146.01574803 141.86614173
 138.07874016 145.01574803 145.83464567]
Number of unique values in column 'RedSize': 78

Unique values in column 'BlueSize': [137.         149.         159.         139.         140.
 152.         150.         138.         148.         130.
 147.         154.         133.         122.         141.
 132.         153.         155.         131.         135.
 134.         136.         156.         151.         145.
 142.         144.         128.         125.         157.
 146.         129.         126.         163.         143.
```

```
 127.         121.         119.         123.         160.
 124.         158.         124.         135.43307087  66.
 138.47244094 135.07874016 149.5        133.5        137.5
 151.5        154.5        134.5        135.5        144.01574803
 148.01574803 155.31496063 140.83464567 128.81102362 126.99212598
 167.         140.07874016 161.         142.65354331 120.
 139.07874016 146.01574803 143.04724409 129.96062992 144.83464567
 137.07874016 149.98425197 135.11023622 143.86614173 143.44094488
 132.14173228 140.04724409 136.8976378  142.04724409 135.8976378
 138.07874016 131.92913386 159.1023622  141.83464567 137.8976378
 141.04724409 145.83464567 133.14173228 139.86614173 151.16535433
 130.17322835 142.07874016 145.01574803 139.8976378  158.52755906
 146.80314961 131.96062992 150.5        140.5        137.98425197
 148.48818898 152.16535433]
Number of unique values in column 'BlueSize': 102

Unique values in column 'Favorite': ['Red' 'Blue']
Number of unique values in column 'Favorite': 2

Unique values in column 'FavoriteWins': [1 0]
Number of unique values in column 'FavoriteWins': 2

Unique values in column 'draw_diff': [ 0  1 -1 -2  2]
Number of unique values in column 'draw_diff': 5

Unique values in column 'avg_sig_str_pct_diff': [-0.49  -0.06  -0.03  ... -0.134 -0.138 -0.128]
Number of unique values in column 'avg_sig_str_pct_diff': 1206

Unique values in column 'avg_TD_pct_diff': [-0.47   0.26   0.42  ...  0.361  0.161  0.572]
Number of unique values in column 'avg_TD_pct_diff': 1669

Unique values in column 'win_by_Decision_Majority_diff': [ 0 -1  1 -2]
Number of unique values in column 'win_by_Decision_Majority_diff': 4

Unique values in column 'win_by_Decision_Split_diff': [-2  1  0 -1  2 -4 -3  3  4  5 -5]
Number of unique values in column 'win_by_Decision_Split_diff': 11

Unique values in column 'win_by_Decision_Unanimous_diff': [ -4   4   1  -5  -1   0   2  -2  -3  -7  -6 -10
6  -9  10  -8   3   5
   7   8 -11  11   9]
Number of unique values in column 'win_by_Decision_Unanimous_diff': 23
```

```
Unique values in column 'win_by_TKO_Doctor_Stoppage_diff': [ 0 -1  1 -2  2]
Number of unique values in column 'win_by_TKO_Doctor_Stoppage_diff': 5

Unique values in column 'odds_diff': [ 4.650e+02  5.050e+02  6.800e+02  1.575e+03  2.400e+02  4.880e+02
  4.330e+02 -3.300e+02  4.000e+00 -3.120e+02 -1.800e+03  9.600e+02
  5.500e+02  3.620e+02         nan  1.700e+03 -4.800e+02  4.350e+02
 -4.200e+02 -2.240e+02 -8.700e+02  5.600e+02  2.800e+02  4.450e+02
  4.800e+02  2.150e+03 -5.250e+02  1.650e+03 -1.000e+01 -4.330e+02
  2.300e+03  3.120e+02 -5.600e+02 -1.350e+03 -4.550e+02 -8.300e+02
  8.550e+02  3.300e+02  2.850e+02  3.520e+02 -4.400e+02  4.900e+02
  6.700e+02 -4.450e+02  6.100e+02  3.060e+02 -4.000e+00  1.350e+03
  1.000e+03  2.720e+02  3.980e+02  2.980e+02 -1.080e+03 -3.750e+02
  4.680e+02 -7.550e+02 -5.380e+02  5.250e+02 -2.720e+02  7.550e+02
 -4.100e+02 -3.390e+02  8.800e+02 -4.680e+02  4.100e+02 -5.500e+02
 -3.520e+02  9.100e+02  2.300e+02 -1.600e+01  1.080e+03  6.450e+02
 -2.300e+02 -2.200e+02  3.750e+02 -2.400e+02  9.350e+02 -5.030e+02
  3.200e+03 -3.060e+02 -4.900e+02  5.380e+02 -2.620e+02 -5.850e+02
  2.620e+02 -1.500e+03  5.850e+02 -3.980e+02  1.500e+03  2.200e+02
 -5.180e+02 -7.300e+02 -2.590e+02 -2.840e+02 -3.320e+02  7.250e+02
  1.170e+03 -6.750e+02  5.950e+02 -9.000e+02  2.500e+02  2.960e+02
 -5.950e+02  2.200e+03  2.840e+02  4.050e+02  1.000e+01  4.700e+02
  8.000e+02  3.900e+02  3.200e+02  3.330e+02  0.000e+00  7.300e+02
  2.590e+02 -2.000e+01  3.540e+02 -4.180e+02  2.490e+02  2.600e+03
  4.180e+02  1.310e+03  4.550e+02  7.050e+02 -3.450e+02  7.000e+02
 -3.200e+02 -5.150e+02 -1.170e+03 -2.980e+02  6.200e+02  3.800e+02
  5.180e+02  5.150e+02 -2.960e+02  2.240e+02 -2.700e+02  1.600e+01
 -9.100e+02 -6.350e+02 -3.540e+02 -6.700e+02 -2.490e+02  8.050e+02
  3.390e+02 -4.080e+02  2.500e+03 -8.050e+02 -2.850e+02 -6.450e+02
 -8.550e+02  2.890e+02  6.600e+02 -9.600e+02  4.200e+02 -3.620e+02
 -6.100e+02 -6.800e+02 -1.020e+03 -1.575e+03  6.350e+02  6.980e+02
 -1.310e+03  1.020e+03  8.300e+02  3.450e+02  1.150e+02 -8.800e+02
 -4.050e+02  1.470e+03  3.550e+02  5.700e+02  1.200e+03  8.700e+02
  6.750e+02  7.800e+02  5.450e+02  4.060e+02 -9.350e+02  2.700e+02
 -4.000e+02 -2.500e+02  6.550e+02  3.250e+02  9.000e+02 -3.250e+02
  2.600e+02 -2.800e+02 -3.500e+02  2.900e+02 -8.000e+00 -2.740e+02
  3.680e+02 -4.130e+02 -7.250e+02  3.860e+02 -4.650e+02  1.400e+01
 -7.000e+02  4.130e+02  3.160e+02  4.560e+02  2.380e+02  2.460e+02
  4.070e+02  3.440e+02  1.950e+03 -2.880e+02  3.500e+02 -2.660e+02
  4.290e+02 -3.640e+02  6.050e+02  4.040e+02  2.920e+02 -5.000e+02
 -1.400e+01  3.320e+02  3.950e+02  2.220e+02 -2.520e+02  2.730e+02
  1.000e+02  2.440e+02 -2.260e+02 -4.040e+02 -4.290e+02  2.660e+02
 -2.220e+02  1.830e+03 -5.450e+02  7.500e+02  2.860e+02  4.380e+02
```

```
-2.440e+02  1.660e+03 -3.100e+02  3.380e+02  7.900e+02 -3.240e+02
 6.300e+02  2.800e+01 -3.020e+02  3.020e+02 -1.030e+03 -2.180e+02
 3.600e+02 -2.920e+02 -2.380e+02 -3.860e+02 -3.600e+02  2.340e+02
 2.880e+02 -2.480e+02  8.100e+02  4.750e+02 -3.770e+02 -4.500e+02
-3.380e+02  9.800e+02  1.440e+03  3.150e+02  8.900e+02  4.310e+02
 3.100e+02  1.050e+03  2.250e+03  3.050e+02 -4.600e+02 -4.250e+02
-7.400e+02  1.300e+03  4.000e+02 -3.150e+02  9.300e+02 -3.350e+02
-5.400e+02 -3.900e+02  4.600e+02  6.400e+02  6.650e+02  3.350e+02
-6.400e+02 -1.050e+03  3.700e+02  2.050e+03 -1.650e+03 -6.650e+02
 1.750e+03  7.100e+02  1.450e+03  7.650e+02 -2.900e+02 -3.800e+02
 5.750e+02 -7.650e+02  4.150e+02 -6.900e+02  4.400e+02 -4.150e+02
 1.210e+03 -3.050e+02 -3.700e+02  4.500e+02  1.900e+03 -7.850e+02
 6.900e+02 -1.175e+03  1.125e+03  5.400e+02 -5.050e+02 -6.200e+02
-6.050e+02 -2.600e+02 -5.750e+02  1.600e+03 -1.240e+03  1.070e+03
 7.850e+02  9.900e+02  4.250e+02 -6.300e+02 -1.600e+03 -1.375e+03
 3.000e+02  5.200e+02 -7.200e+02  4.950e+02  1.285e+03 -3.000e+02
 8.250e+02 -3.650e+02  1.035e+03 -4.350e+02  3.850e+02 -4.700e+02
 3.000e+03  1.015e+03 -8.850e+02  7.200e+02 -4.950e+02 -1.035e+03
 5.350e+02  8.500e+02  5.550e+02 -5.550e+02  7.450e+02 -1.500e+01
-3.850e+02  2.350e+02 -2.950e+02 -2.750e+02 -5.350e+02  1.825e+03
-7.450e+02 -5.200e+02  1.150e+03 -5.700e+02  5.900e+02 -1.220e+03
 2.175e+03  1.425e+03  9.500e+02  9.850e+02  2.290e+02  1.580e+03
-3.400e+02 -3.360e+02 -1.200e+01  3.480e+02  2.560e+02  4.270e+02
 6.500e+02  3.170e+02 -2.250e+02 -3.310e+02 -2.510e+02  2.750e+02
 2.390e+02 -4.850e+02 -2.650e+02  4.280e+02 -6.000e+00  2.270e+02
 2.950e+02  1.200e+01 -3.170e+02 -2.390e+02  3.090e+02  2.540e+02
 4.300e+02  2.740e+02 -3.550e+02 -3.340e+02 -2.640e+02  3.040e+02
 5.800e+02 -3.930e+02  5.650e+02 -2.540e+02  5.100e+02  3.930e+02
 7.750e+02  6.000e+02  5.000e+02  9.400e+02  2.940e+02  4.930e+02
 3.400e+02  2.000e+01  5.060e+02  3.790e+02  4.630e+02 -3.220e+02
 8.520e+02  7.870e+02 -2.150e+02 -2.410e+02  2.580e+02  2.250e+02
-2.580e+02 -2.350e+02 -9.500e+02 -2.680e+02 -2.870e+02  9.050e+02
 3.220e+02 -2.890e+02  9.750e+02  2.510e+02  4.850e+02  2.470e+02
-5.800e+02  5.110e+02  2.517e+03  2.970e+02  6.950e+02  1.360e+03
 2.420e+02  1.270e+03 -7.500e+02  8.560e+02  6.520e+02 -2.670e+02
 6.960e+02 -6.950e+02  3.130e+02  3.840e+02 -6.520e+02  7.520e+02
-2.560e+02 -3.110e+02  3.650e+02 -7.760e+02  2.280e+02  8.220e+02
 5.810e+02 -5.000e+00  3.370e+02  5.880e+02 -1.185e+03 -2.370e+02
-3.010e+02 -2.470e+02  6.710e+02  5.220e+02 -7.260e+02 -2.970e+02
 3.270e+02  5.540e+02  5.000e+00  4.010e+02  5.030e+02 -4.280e+02
 2.180e+03 -2.330e+02  9.470e+02  7.880e+02  8.200e+02  3.340e+02
 6.250e+02  2.370e+02 -3.270e+02  4.980e+02  3.610e+02 -2.930e+02
```

```
  2.530e+02 -6.460e+02  1.190e+03 -2.420e+02 -4.300e+02  3.240e+02
  1.435e+03  1.060e+03 -3.630e+02 -6.000e+02  2.640e+02  2.100e+02
  3.140e+02  1.455e+03  1.530e+03 -1.485e+03  2.430e+02  5.370e+02
  8.450e+02  1.110e+03 -5.100e+02  1.800e+03  1.155e+03  1.005e+03
 -6.600e+02  1.275e+03 -1.275e+03  1.065e+03  6.930e+02 -6.500e+02
 -9.400e+02  9.250e+02 -8.250e+02 -5.650e+02 -8.450e+02 -1.005e+03
 -9.750e+02  1.095e+03  1.975e+03  1.410e+03  2.275e+03 -5.900e+02
  6.880e+02 -3.780e+02 -7.900e+02  7.400e+02 -7.950e+02 -5.280e+02
 -7.750e+02  5.280e+02  3.690e+02 -1.155e+03 -1.110e+03 -8.100e+02
 -6.850e+02  7.670e+02  3.630e+02  2.700e+03  1.230e+03  1.915e+03
 -2.100e+02 -1.125e+03  2.325e+03 -1.200e+03  8.630e+02 -1.975e+03
  2.800e+03  2.100e+03 -7.100e+02  1.620e+03  2.650e+03 -1.285e+03
  1.460e+03  1.535e+03 -8.950e+02  1.185e+03  1.335e+03  7.950e+02
  8.950e+02  1.268e+03  6.850e+02  1.100e+03  2.000e+03 -3.460e+02
  1.250e+03 -3.080e+02  3.430e+02  8.600e+02  1.670e+03 -1.000e+03
  9.030e+02  1.010e+03 -1.225e+03 -9.030e+02  1.325e+03 -3.690e+02
  6.680e+02  1.235e+03 -8.500e+02  8.030e+02  1.560e+03  2.520e+02
  5.300e+02  8.000e+00  5.430e+02 -4.420e+02  2.320e+02  2.780e+02
  2.950e+03  2.760e+02  5.730e+02  2.260e+02  1.550e+03  3.770e+02
 -2.120e+02  3.670e+02 -2.940e+02 -9.850e+02  3.420e+02 -2.860e+02
  7.380e+02 -2.460e+02  3.360e+02 -2.000e+00  3.180e+02 -1.100e+03
 -8.000e+02  6.000e+00 -3.710e+02  1.354e+03  4.160e+02  1.225e+03
  1.400e+03]
Number of unique values in column 'odds_diff': 619

Unique values in column 'ev_diff': [175.     247.381  273.6842 ... 607.1795 459.6154 534.6154]
Number of unique values in column 'ev_diff': 1067

Unique values in column 'StanceCombo': ['Orthodox vs Orthodox' 'Southpaw vs Southpaw' 'Southpaw vs Orthodo
x'
 'Orthodox vs Southpaw' 'Switch vs Orthodox' 'Orthodox vs Switch'
 'Southpaw vs Switch' 'Switch vs Switch' 'Switch vs Southpaw'
 'Orthodox vs nan' 'Southpaw vs nan' 'Open Stance vs Orthodox'
 'Southpaw vs Open Stance' 'Open Stance vs Southpaw']
Number of unique values in column 'StanceCombo': 14

Unique values in column 'AgeGap': [ 3  2  6  0  5  4 10 17  1  8  7 15  9 11 12 13 16 14]
Number of unique values in column 'AgeGap': 18

Unique values in column 'Spread': [4.650e+02 5.050e+02 6.800e+02 1.575e+03 2.400e+02 4.880e+02 4.330e+02
 3.300e+02 4.000e+00 3.120e+02 1.800e+03 9.600e+02 5.500e+02 3.620e+02
       nan 1.700e+03 4.800e+02 4.350e+02 4.200e+02 2.240e+02 8.700e+02
```

```
5.600e+02 2.800e+02 4.450e+02 2.150e+03 5.250e+02 1.650e+03 1.000e+01
2.300e+03 1.350e+03 4.550e+02 8.300e+02 8.550e+02 2.850e+02 3.520e+02
4.400e+02 4.900e+02 6.700e+02 6.100e+02 3.060e+02 1.000e+03 2.720e+02
3.980e+02 2.980e+02 1.080e+03 3.750e+02 4.680e+02 7.550e+02 5.380e+02
4.100e+02 3.390e+02 8.800e+02 9.100e+02 2.300e+02 1.600e+01 6.450e+02
2.200e+02 9.350e+02 5.030e+02 3.200e+03 2.620e+02 5.850e+02 1.500e+03
5.180e+02 7.300e+02 2.590e+02 2.840e+02 3.320e+02 7.250e+02 1.170e+03
6.750e+02 5.950e+02 9.000e+02 2.500e+02 2.960e+02 2.200e+03 4.050e+02
4.700e+02 8.000e+02 3.900e+02 3.200e+02 3.330e+02 0.000e+00 2.000e+01
3.540e+02 4.180e+02 2.490e+02 2.600e+03 1.310e+03 7.050e+02 3.450e+02
7.000e+02 5.150e+02 6.200e+02 3.800e+02 2.700e+02 6.350e+02 8.050e+02
4.080e+02 2.500e+03 2.890e+02 6.600e+02 1.020e+03 6.980e+02 1.150e+02
1.470e+03 3.550e+02 5.700e+02 1.200e+03 7.800e+02 5.450e+02 4.060e+02
4.000e+02 6.550e+02 3.250e+02 2.600e+02 3.500e+02 2.900e+02 8.000e+00
2.740e+02 3.680e+02 4.130e+02 3.860e+02 1.400e+01 3.160e+02 4.560e+02
2.380e+02 2.460e+02 4.070e+02 3.440e+02 1.950e+03 2.880e+02 2.660e+02
4.290e+02 3.640e+02 6.050e+02 4.040e+02 2.920e+02 5.000e+02 3.950e+02
2.220e+02 2.520e+02 2.730e+02 1.000e+02 2.440e+02 2.260e+02 1.830e+03
7.500e+02 2.860e+02 4.380e+02 1.660e+03 3.100e+02 3.380e+02 7.900e+02
3.240e+02 6.300e+02 2.800e+01 3.020e+02 1.030e+03 2.180e+02 3.600e+02
2.340e+02 2.480e+02 8.100e+02 4.750e+02 3.770e+02 4.500e+02 9.800e+02
1.440e+03 3.150e+02 8.900e+02 4.310e+02 1.050e+03 2.250e+03 3.050e+02
4.600e+02 4.250e+02 7.400e+02 1.300e+03 9.300e+02 3.350e+02 5.400e+02
6.400e+02 6.650e+02 3.700e+02 2.050e+03 1.750e+03 7.100e+02 1.450e+03
7.650e+02 5.750e+02 4.150e+02 6.900e+02 1.210e+03 1.900e+03 7.850e+02
1.175e+03 1.125e+03 1.600e+03 1.240e+03 1.070e+03 9.900e+02 1.375e+03
3.000e+02 5.200e+02 7.200e+02 4.950e+02 1.285e+03 8.250e+02 3.650e+02
1.035e+03 3.850e+02 3.000e+03 1.015e+03 8.850e+02 5.350e+02 8.500e+02
5.550e+02 7.450e+02 1.500e+01 2.350e+02 2.950e+02 2.750e+02 1.825e+03
1.150e+03 5.900e+02 1.220e+03 2.175e+03 1.425e+03 9.500e+02 9.850e+02
2.290e+02 1.580e+03 3.400e+02 3.360e+02 1.200e+01 3.480e+02 2.560e+02
4.270e+02 6.500e+02 3.170e+02 2.250e+02 3.310e+02 2.510e+02 2.390e+02
4.850e+02 2.650e+02 4.280e+02 6.000e+00 2.270e+02 3.090e+02 2.540e+02
4.300e+02 3.340e+02 2.640e+02 3.040e+02 5.800e+02 3.930e+02 5.650e+02
5.100e+02 7.750e+02 6.000e+02 9.400e+02 2.940e+02 4.930e+02 5.060e+02
3.790e+02 4.630e+02 3.220e+02 8.520e+02 7.870e+02 2.150e+02 2.410e+02
2.580e+02 2.680e+02 2.870e+02 9.050e+02 9.750e+02 2.470e+02 5.110e+02
2.517e+03 2.970e+02 6.950e+02 1.360e+03 2.420e+02 1.270e+03 8.560e+02
6.520e+02 2.670e+02 6.960e+02 3.130e+02 3.840e+02 7.520e+02 3.110e+02
7.760e+02 2.280e+02 8.220e+02 5.810e+02 5.000e+00 3.370e+02 5.880e+02
1.185e+03 2.370e+02 3.010e+02 6.710e+02 5.220e+02 7.260e+02 3.270e+02
5.540e+02 4.010e+02 2.180e+03 2.330e+02 9.470e+02 7.880e+02 8.200e+02
```

```
6.250e+02 4.980e+02 3.610e+02 2.930e+02 2.530e+02 6.460e+02 1.190e+03
1.435e+03 1.060e+03 3.630e+02 2.100e+02 3.140e+02 1.455e+03 1.530e+03
1.485e+03 2.430e+02 5.370e+02 8.450e+02 1.110e+03 1.155e+03 1.005e+03
1.275e+03 1.065e+03 6.930e+02 9.250e+02 1.095e+03 1.975e+03 1.410e+03
2.275e+03 6.880e+02 3.780e+02 7.950e+02 5.280e+02 3.690e+02 6.850e+02
7.670e+02 2.700e+03 1.230e+03 1.915e+03 2.325e+03 8.630e+02 2.800e+03
2.100e+03 1.620e+03 2.650e+03 1.460e+03 1.535e+03 8.950e+02 1.335e+03
1.268e+03 1.100e+03 2.000e+03 3.460e+02 1.250e+03 3.080e+02 3.430e+02
8.600e+02 1.670e+03 9.030e+02 1.010e+03 1.225e+03 1.325e+03 6.680e+02
1.235e+03 8.030e+02 1.560e+03 5.300e+02 5.430e+02 4.420e+02 2.320e+02
2.780e+02 2.950e+03 2.760e+02 5.730e+02 1.550e+03 2.120e+02 3.670e+02
3.420e+02 7.380e+02 2.000e+00 3.180e+02 3.710e+02 1.354e+03 4.160e+02
1.400e+03]
Number of unique values in column 'Spread': 400
```

In [281…

```
df.head()
```

Out[281…

| | RedFighter | BlueFighter | RedOdds | BlueOdds | RedExpectedValue | BlueExpectedValue | Winner | TitleBout | WeightClass |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 66 | 1009 | -250.0 | 215.0 | 40.0000 | 215.0 | Red | True | 3 |
| **1** | 1441 | 718 | -210.0 | 295.0 | 47.6190 | 295.0 | Red | False | 8 |
| **2** | 307 | 67 | -380.0 | 300.0 | 26.3158 | 300.0 | Red | False | 4 |
| **3** | 221 | 1071 | -950.0 | 625.0 | 10.5263 | 625.0 | Red | False | 2 |
| **4** | 1183 | 524 | -130.0 | 110.0 | 76.9231 | 110.0 | Blue | False | 2 |

In [282…

```
pd.set_option('display.max_columns', None)

print(df.columns)
print(df.columns.tolist())
```

```
Index(['RedFighter', 'BlueFighter', 'RedOdds', 'BlueOdds', 'RedExpectedValue',
       'BlueExpectedValue', 'Winner', 'TitleBout', 'WeightClass', 'Gender',
       ...
       'avg_TD_pct_diff', 'win_by_Decision_Majority_diff',
       'win_by_Decision_Split_diff', 'win_by_Decision_Unanimous_diff',
       'win_by_TKO_Doctor_Stoppage_diff', 'odds_diff', 'ev_diff',
       'StanceCombo', 'AgeGap', 'Spread'],
      dtype='object', length=153)
['RedFighter', 'BlueFighter', 'RedOdds', 'BlueOdds', 'RedExpectedValue', 'BlueExpectedValue', 'Winner', 'Ti
tleBout', 'WeightClass', 'Gender', 'NumberOfRounds', 'BlueCurrentLoseStreak', 'BlueCurrentWinStreak', 'Blue
Draws', 'BlueAvgSigStrLanded', 'BlueAvgSigStrPct', 'BlueAvgSubAtt', 'BlueAvgTDLanded', 'BlueAvgTDPct', 'Blu
eLongestWinStreak', 'BlueLosses', 'BlueTotalRoundsFought', 'BlueTotalTitleBouts', 'BlueWinsByDecisionMajori
ty', 'BlueWinsByDecisionSplit', 'BlueWinsByDecisionUnanimous', 'BlueWinsByKO', 'BlueWinsBySubmission', 'Blu
eWinsByTKODoctorStoppage', 'BlueWins', 'BlueStance', 'BlueWeightLbs', 'RedCurrentLoseStreak', 'RedCurrentWi
nStreak', 'RedDraws', 'RedAvgSigStrLanded', 'RedAvgSigStrPct', 'RedAvgSubAtt', 'RedAvgTDLanded', 'RedAvgTDP
ct', 'RedLongestWinStreak', 'RedLosses', 'RedTotalRoundsFought', 'RedTotalTitleBouts', 'RedWinsByDecisionMa
jority', 'RedWinsByDecisionSplit', 'RedWinsByDecisionUnanimous', 'RedWinsByKO', 'RedWinsBySubmission', 'Red
WinsByTKODoctorStoppage', 'RedWins', 'RedStance', 'RedWeightLbs', 'RedAge', 'BlueAge', 'LoseStreakDif', 'Wi
nStreakDif', 'LongestWinStreakDif', 'WinDif', 'LossDif', 'TotalRoundDif', 'TotalTitleBoutDif', 'KODif', 'Su
bDif', 'HeightDif', 'ReachDif', 'AgeDif', 'SigStrDif', 'AvgSubAttDif', 'AvgTDDif', 'EmptyArena', 'BMatchWCR
ank', 'RMatchWCRank', 'RWFlyweightRank', 'RWFeatherweightRank', 'RWStrawweightRank', 'RWBantamweightRank',
'RHeavyweightRank', 'RLightHeavyweightRank', 'RMiddleweightRank', 'RWelterweightRank', 'RLightweightRank',
'RFeatherweightRank', 'RBantamweightRank', 'RFlyweightRank', 'RPFPRank', 'BWFlyweightRank', 'BWFeatherweigh
tRank', 'BWStrawweightRank', 'BWBantamweightRank', 'BHeavyweightRank', 'BLightHeavyweightRank', 'BMiddlewei
ghtRank', 'BWelterweightRank', 'BLightweightRank', 'BFeatherweightRank', 'BBantamweightRank', 'BFlyweightRa
nk', 'BPFPRank', 'BetterRank', 'Finish', 'FinishDetails', 'FinishRound', 'FinishRoundTime', 'TotalFightTime
Secs', 'RedDecOdds', 'BlueDecOdds', 'RSubOdds', 'BSubOdds', 'RKOOdds', 'BKOOdds', 'WeightClassLabel', 'Blue
Height', 'BlueReach', 'RedHeight', 'RedReach', 'ROver35', 'BOver35', 'RWinPct', 'BWinPct', 'RedStrikingRati
o', 'BlueStrikingRatio', 'RedTotalFights', 'BlueTotalFights', 'RedIsGrappler', 'BlueIsGrappler', 'RedIsStri
ker', 'BlueIsStriker', 'RGrapplerVBStriker', 'BGrapplerVRStriker', 'RedStrikingEfficiency', 'BlueStrikingEf
ficiency', 'RedTDEfficiency', 'BlueTDEfficiency', 'RedEffectiveTD', 'BlueEffectiveTD', 'EffectiveTDDif', 'R
edSize', 'BlueSize', 'Favorite', 'FavoriteWins', 'draw_diff', 'avg_sig_str_pct_diff', 'avg_TD_pct_diff', 'w
in_by_Decision_Majority_diff', 'win_by_Decision_Split_diff', 'win_by_Decision_Unanimous_diff', 'win_by_TKO_
Doctor_Stoppage_diff', 'odds_diff', 'ev_diff', 'StanceCombo', 'AgeGap', 'Spread']
```

In [283…

```python
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import LabelEncoder

#  Convert 'Winner' to binary (Red=1, Blue=0), and drop others
df = df[df['Winner'].isin(['Red', 'Blue'])]  # Keep only valid outcomes
```

```
df['Winner'] = df['Winner'].map({'Red': 1, 'Blue': 0})  # Binary conversion

# Check value counts for the 'Winner' column (including NaNs)
print(df['Winner'].value_counts(dropna=False))

# Basic descriptive statistics
print(df.describe())


# ------------------------------------------------
# columns with excessive missingness
# ------------------------------------------------
cols_missing = [
    'BMatchWCRank', 'RMatchWCRank', 'RWFlyweightRank', 'RWFeatherweightRank',
    'RWStrawweightRank', 'RWBantamweightRank', 'RHeavyweightRank',
    'RLightHeavyweightRank', 'RMiddleweightRank', 'RWelterweightRank',
    'RLightweightRank', 'RFeatherweightRank', 'RBantamweightRank',
    'RFlyweightRank', 'RPFPRank', 'BWFlyweightRank', 'BWFeatherweightRank',
    'BWStrawweightRank', 'BWBantamweightRank', 'BHeavyweightRank',
    'BLightHeavyweightRank', 'BMiddleweightRank', 'BWelterweightRank',
    'BLightweightRank', 'BFeatherweightRank', 'BBantamweightRank',
    'BFlyweightRank', 'BPFPRank'
]

# ------------------------------------------------
# columns to drop due to data leakage
# ------------------------------------------------
cols_leakage = [
# Red fighter stats
'RedAvgSigStrLanded', 'RedAvgSigStrPct', 'RedAvgSubAtt',
'RedAvgTDLanded', 'RedAvgTDPct', 'RedLongestWinStreak',
'RedTotalRoundsFought', 'RedTotalTitleBouts',
'RedWinsByDecisionMajority', 'RedWinsByDecisionSplit', 'RedWinsByDecisionUnanimous',
'RedWinsByKO', 'RedWinsBySubmission', 'RedWinsByTKODoctorStoppage',
'RedWins', 'RedStrikingRatio', 'RedStrikingEfficiency', 'RedTDEfficiency',
'RedEffectiveTD', 'RedExpectedValue', 'RedDecOdds', 'RSubOdds', 'RKOOdds',

# Blue fighter stats
'BlueAvgSigStrLanded', 'BlueAvgSigStrPct', 'BlueAvgSubAtt',
'BlueAvgTDLanded', 'BlueAvgTDPct', 'BlueLongestWinStreak',
'BlueTotalRoundsFought', 'BlueTotalTitleBouts',
'BlueWinsByDecisionMajority', 'BlueWinsByDecisionSplit', 'BlueWinsByDecisionUnanimous',
```

```
'BlueWinsByKO', 'BlueWinsBySubmission', 'BlueWinsByTKODoctorStoppage',
'BlueWins', 'BlueStrikingRatio', 'BlueStrikingEfficiency', 'BlueTDEfficiency',
'BlueEffectiveTD', 'BlueExpectedValue', 'BlueDecOdds', 'BSubOdds', 'BKOOdds',
'Finish', 'FinishDetails', 'FinishRound', 'FinishRoundTime', 'TotalFightTimeSecs', 'Favorite', 'FavoriteWi

# Derived features (includes current fight outcome)
'SigStrDif', 'AvgSubAttDif', 'AvgTDDif', 'LoseStreakDif', 'WinStreakDif', 'LongestWinStreakDif',
'WinDif', 'LossDif', 'TotalRoundDif', 'TotalTitleBoutDif', 'KODif', 'SubDif',
'NumberOfRounds', 'EffectiveTDDif', 'draw_diff', 'avg_sig_str_pct_diff', 'avg_TD_pct_diff',
'win_by_Decision_Majority_diff', 'win_by_Decision_Split_diff', 'win_by_Decision_Unanimous_diff', 'win_by_T
]

# ------------------------------------------------------
# REMOVE LEAKAGE / FUTURE-ONLY FEATURES FROM TRAINING DATA
# ------------------------------------------------------

all_leakage = cols_missing + cols_leakage

df = df.drop(columns=all_leakage, errors='ignore')

print("\nTraining dataset shape after leakage removal:", df.shape)
```

```
Winner
1    3787
0    2741
Name: count, dtype: int64
        RedFighter  BlueFighter      RedOdds     BlueOdds  RedExpectedValue  \
count  6528.000000  6528.000000  6301.000000  6302.000000       6301.000000
mean    819.761795   942.707874  -115.711474    59.793240         96.658224
std     476.188466   556.368724   277.225783   253.117416         85.891109
min       0.000000     0.000000 -2100.000000 -1200.000000          4.761900
25%     419.000000   462.000000  -255.000000  -150.000000         39.215700
50%     806.000000   922.500000  -150.000000   130.000000         66.666700
75%    1241.250000  1427.000000   130.000000   215.000000        130.000000
max    1660.000000  1921.000000   775.000000  1300.000000        775.000000

       BlueExpectedValue       Winner  WeightClass       Gender  \
count        6302.000000  6528.000000  6528.000000  6528.000000
mean          165.054566     0.580116     5.503370     0.877298
std           137.689177     0.493577     3.200274     0.328120
min             8.333300     0.000000     0.000000     0.000000
25%            66.666700     0.000000     3.000000     1.000000
50%           130.000000     1.000000     6.000000     1.000000
75%           215.000000     1.000000     8.000000     1.000000
max          1300.000000     1.000000    12.000000     1.000000

       NumberOfRounds  BlueCurrentLoseStreak  BlueCurrentWinStreak  \
count     6528.000000            6528.000000           6528.000000
mean         3.185509               0.501072              0.957567
std          0.577441               0.794303              1.406786
min          3.000000               0.000000              0.000000
25%          3.000000               0.000000              0.000000
50%          3.000000               0.000000              0.000000
75%          3.000000               1.000000              1.000000
max          5.000000               6.000000             12.000000

         BlueDraws  BlueAvgSigStrLanded  BlueAvgSigStrPct  BlueAvgSubAtt  \
count  6528.000000          5598.000000       5763.000000    5696.000000
mean      0.023131            19.841810          0.453059       0.500202
std       0.156327            20.315307          0.110787       0.672859
min       0.000000             0.000000          0.000000       0.000000
25%       0.000000             3.880000          0.400000       0.000000
50%       0.000000             9.280000          0.460000       0.300000
75%       0.000000            32.666700          0.513000       0.800000
```

```
max       2.000000          154.000000          1.000000       8.400000

       BlueAvgTDLanded  BlueAvgTDPct  BlueLongestWinStreak   BlueLosses  \
count      5695.000000   5686.000000           6528.000000  6528.000000
mean          1.320536      0.325419              1.923407     1.863664
std           1.356491      0.239174              1.949952     2.170130
min           0.000000      0.000000              0.000000     0.000000
25%           0.330000      0.150000              0.000000     0.000000
50%           1.000000      0.330000              1.000000     1.000000
75%           1.970000      0.470000              3.000000     3.000000
max          10.860000      1.000000             17.000000    16.000000

       BlueTotalRoundsFought  BlueTotalTitleBouts  BlueWinsByDecisionMajority  \
count            6528.000000          6528.000000                 6528.000000
mean               11.872396             0.251685                    0.017770
std                13.845139             1.085122                    0.133278
min                 0.000000             0.000000                    0.000000
25%                 3.000000             0.000000                    0.000000
50%                 7.000000             0.000000                    0.000000
75%                17.000000             0.000000                    0.000000
max               111.000000            16.000000                    2.000000

       BlueWinsByDecisionSplit  BlueWinsByDecisionUnanimous  BlueWinsByKO  \
count              6528.000000                  6528.000000   6528.000000
mean                  0.279871                     1.093597      1.066330
std                   0.595123                     1.613288      1.723951
min                   0.000000                     0.000000      0.000000
25%                   0.000000                     0.000000      0.000000
50%                   0.000000                     0.000000      0.000000
75%                   0.000000                     2.000000      1.250000
max                   5.000000                    11.000000     20.000000

       BlueWinsBySubmission  BlueWinsByTKODoctorStoppage     BlueWins  \
count           6528.000000                  6528.000000  6528.000000
mean               0.626532                     0.022212     3.145680
std                1.258249                     0.153494     3.712852
min                0.000000                     0.000000     0.000000
25%                0.000000                     0.000000     0.000000
50%                0.000000                     0.000000     2.000000
75%                1.000000                     0.000000     5.000000
max               13.000000                     2.000000    31.000000
```

```
       BlueWeightLbs  RedCurrentLoseStreak  RedCurrentWinStreak     RedDraws  \
count    6528.000000           6528.000000          6528.000000  6528.000000
mean      163.183977              0.622243             1.101562     0.031097
std        34.599386              0.872301             1.760767     0.187999
min       115.000000              0.000000             0.000000     0.000000
25%       135.000000              0.000000             0.000000     0.000000
50%       155.000000              0.000000             0.000000     0.000000
75%       185.000000              1.000000             2.000000     0.000000
max       265.000000              7.000000            18.000000     2.000000

       RedAvgSigStrLanded  RedAvgSigStrPct  RedAvgSubAtt  RedAvgTDLanded  \
count         6073.000000      6171.000000   6171.000000     6171.000000
mean            21.152766         0.460321      0.536907        1.399962
std             19.882916         0.098315      0.693125        1.308340
min              0.000000         0.000000      0.000000        0.000000
25%              4.160000         0.403000      0.000000        0.450000
50%             15.666700         0.460000      0.333300        1.030000
75%             34.000000         0.516000      0.800000        2.000000
max            141.000000         1.000000      8.400000       12.500000

       RedAvgTDPct  RedLongestWinStreak    RedLosses  RedTotalRoundsFought  \
count  6161.000000          6528.000000  6528.000000           6528.000000
mean      0.341467             2.679994     2.566789             17.408548
std       0.220986             2.242687     2.703546             17.846748
min       0.000000             0.000000     0.000000              0.000000
25%       0.200000             1.000000     1.000000              5.000000
50%       0.338000             2.000000     2.000000             12.000000
75%       0.470000             4.000000     4.000000             25.000000
max       1.000000            18.000000    21.000000            448.000000

       RedTotalTitleBouts  RedWinsByDecisionMajority  RedWinsByDecisionSplit  \
count         6528.000000                6528.000000             6528.000000
mean             0.553462                   0.026961                0.400888
std              1.544962                   0.163862                0.702801
min              0.000000                   0.000000                0.000000
25%              0.000000                   0.000000                0.000000
50%              0.000000                   0.000000                0.000000
75%              0.000000                   0.000000                1.000000
max             16.000000                   2.000000                5.000000

       RedWinsByDecisionUnanimous  RedWinsByKO  RedWinsBySubmission  \
count                6528.000000  6528.000000          6528.000000
```

```
mean                    1.612286     1.563879            0.934589
std                     1.972295     2.156770            1.608402
min                     0.000000     0.000000            0.000000
25%                     0.000000     0.000000            0.000000
50%                     1.000000     1.000000            0.000000
75%                     2.000000     2.000000            1.000000
max                    11.000000    21.000000           16.000000

       RedWinsByTKODoctorStoppage      RedWins  RedWeightLbs       RedAge  \
count                6528.000000  6528.000000   6528.000000  6528.000000
mean                    0.035539     4.628064    163.621324    30.359528
std                     0.199492     4.505498     34.846543     4.180712
min                     0.000000     0.000000    115.000000    18.000000
25%                     0.000000     1.000000    135.000000    27.000000
50%                     0.000000     3.000000    155.000000    30.000000
75%                     0.000000     7.000000    185.000000    33.000000
max                     2.000000    33.000000    265.000000    47.000000

          BlueAge  LoseStreakDif  WinStreakDif  LongestWinStreakDif  \
count  6528.000000    6528.000000   6528.000000          6528.000000
mean     29.805607       0.059283     -0.143842            -0.756587
std       3.959623       1.024000      1.874732             2.025886
min      19.000000      -6.000000    -18.000000           -12.000000
25%      27.000000       0.000000     -1.000000            -2.000000
50%      30.000000       0.000000      0.000000            -1.000000
75%      32.000000       0.000000      0.000000             0.000000
max      47.000000       6.000000     10.000000            14.000000

           WinDif      LossDif  TotalRoundDif  TotalTitleBoutDif  \
count  6528.000000  6528.000000    6528.000000        6528.000000
mean     -1.482384     0.075061      -5.536152          -0.301777
std       4.182192     3.129140      17.995648           1.681080
min     -28.000000   -20.000000    -448.000000         -16.000000
25%      -3.000000    -1.000000     -12.000000           0.000000
50%      -1.000000     0.000000      -3.000000           0.000000
75%       0.000000     2.000000       2.000000           0.000000
max      23.000000    16.000000      87.000000          15.000000

            KODif       SubDif    HeightDif     ReachDif       AgeDif  \
count  6528.000000  6528.000000  6528.000000  6528.000000  6528.000000
mean     -0.510876    -0.308058    -0.006679    -0.299271     0.096814
std       2.149365     1.844810     6.770956     9.132413     5.201719
```

| | | | | | |
|---|---|---|---|---|---|
| min | -21.000000 | -16.000000 | -187.960000 | -187.960000 | -17.000000 |
| 25% | -1.000000 | -1.000000 | -5.080000 | -5.080000 | -3.000000 |
| 50% | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 75% | 0.000000 | 0.000000 | 5.080000 | 5.080000 | 4.000000 |
| max | 14.000000 | 10.000000 | 30.480000 | 30.480000 | 17.000000 |

| | SigStrDif | AvgSubAttDif | AvgTDDif | EmptyArena | BMatchWCRank | \ |
|---|---|---|---|---|---|---|
| count | 6528.000000 | 6528.000000 | 6528.000000 | 5042.000000 | 1200.000000 | |
| mean | -2.663342 | -0.071094 | -0.171371 | 0.153114 | 8.339167 | |
| std | 19.583493 | 0.892257 | 1.754354 | 0.360133 | 4.202839 | |
| min | -118.000000 | -8.400000 | -11.000000 | 0.000000 | 1.000000 | |
| 25% | -7.895850 | -0.444400 | -1.010000 | 0.000000 | 5.000000 | |
| 50% | -0.304750 | 0.000000 | 0.000000 | 0.000000 | 9.000000 | |
| 75% | 2.100000 | 0.282400 | 0.700000 | 0.000000 | 12.000000 | |
| max | 128.222200 | 7.800000 | 10.860000 | 1.000000 | 15.000000 | |

| | RMatchWCRank | RWFlyweightRank | RWFeatherweightRank | RWStrawweightRank | \ |
|---|---|---|---|---|---|
| count | 1779.000000 | 96.000000 | 9.0 | 146.000000 | |
| mean | 6.953345 | 7.291667 | 0.0 | 7.047945 | |
| std | 4.657029 | 4.975554 | 0.0 | 4.616809 | |
| min | 0.000000 | 0.000000 | 0.0 | 0.000000 | |
| 25% | 3.000000 | 3.000000 | 0.0 | 3.000000 | |
| 50% | 7.000000 | 7.000000 | 0.0 | 7.000000 | |
| 75% | 11.000000 | 12.000000 | 0.0 | 11.000000 | |
| max | 15.000000 | 15.000000 | 0.0 | 15.000000 | |

| | RWBantamweightRank | RHeavyweightRank | RLightHeavyweightRank | \ |
|---|---|---|---|---|
| count | 154.000000 | 186.000000 | 184.000000 | |
| mean | 7.097403 | 6.881720 | 7.119565 | |
| std | 4.803008 | 4.404787 | 4.638500 | |
| min | 0.000000 | 0.000000 | 0.000000 | |
| 25% | 3.000000 | 3.000000 | 4.000000 | |
| 50% | 7.000000 | 7.000000 | 7.000000 | |
| 75% | 11.750000 | 10.000000 | 11.000000 | |
| max | 15.000000 | 15.000000 | 15.000000 | |

| | RMiddleweightRank | RWelterweightRank | RLightweightRank | \ |
|---|---|---|---|---|
| count | 182.000000 | 191.000000 | 184.000000 | |
| mean | 7.351648 | 7.130890 | 7.032609 | |
| std | 4.711274 | 4.705947 | 4.662257 | |
| min | 0.000000 | 0.000000 | 0.000000 | |
| 25% | 4.000000 | 3.000000 | 3.000000 | |

```
50%             7.000000           7.000000          6.000000
75%            11.750000          11.000000         11.000000
max            15.000000          15.000000         15.000000

       RFeatherweightRank  RBantamweightRank  RFlyweightRank    RPFPRank  \
count          177.000000         181.000000      188.000000  253.000000
mean             6.983051           6.972376        6.590426    6.913043
std              4.653291           4.709837        4.621192    4.195655
min              0.000000           0.000000        0.000000    1.000000
25%              3.000000           3.000000        2.000000    3.000000
50%              7.000000           7.000000        7.000000    6.000000
75%             11.000000          11.000000       10.250000   10.000000
max             15.000000          15.000000       15.000000   15.000000

       BWFlyweightRank  BWFeatherweightRank  BWStrawweightRank  \
count        73.000000                  1.0         100.000000
mean          8.410959                  0.0           8.170000
std           4.009834                  NaN           4.192478
min           1.000000                  0.0           1.000000
25%           6.000000                  0.0           5.000000
50%           9.000000                  0.0           9.000000
75%          12.000000                  0.0          12.000000
max          15.000000                  0.0          15.000000

       BWBantamweightRank  BHeavyweightRank  BLightHeavyweightRank  \
count          107.000000        148.000000             120.000000
mean             8.476636          8.641892               8.483333
std              4.254511          4.090823               4.152519
min              0.000000          1.000000               0.000000
25%              5.000000          5.000000               6.000000
50%              9.000000          9.000000               9.000000
75%             12.000000         12.000000              12.000000
max             15.000000         15.000000              15.000000

       BMiddleweightRank  BWelterweightRank  BLightweightRank  \
count         137.000000         119.000000        120.000000
mean            8.554745           8.386555          8.150000
std             4.311271           4.405523          4.076269
min             0.000000           1.000000          1.000000
25%             5.000000           4.500000          5.000000
50%             9.000000           9.000000          8.000000
75%            12.000000          12.000000         11.250000
```

```
max            15.000000          15.000000         15.000000

       BFeatherweightRank  BBantamweightRank  BFlyweightRank   BPFPRank  \
count          124.000000         119.000000      130.000000  67.000000
mean             7.967742           8.268908        8.407692   9.194030
std              4.462919           4.354420        4.307369   4.352773
min              0.000000           0.000000        1.000000   1.000000
25%              4.000000           4.000000        5.000000   5.000000
50%              8.500000           9.000000        8.000000  10.000000
75%             12.000000          12.000000       12.000000  13.500000
max             15.000000          15.000000       15.000000  15.000000

       FinishRound  TotalFightTimeSecs   RedDecOdds  BlueDecOdds     RSubOdds  \
count  5906.000000         5906.000000  5441.000000  5412.000000  5192.000000
mean      2.424145          657.536234   308.333395   425.870288   884.048151
std       1.007887          360.383418   250.750088   325.940028   601.826547
min       1.000000            5.000000  -440.000000  -200.000000  -370.000000
25%       1.000000          299.000000   170.000000   222.000000   439.750000
50%       3.000000          900.000000   250.000000   350.000000   750.000000
75%       3.000000          900.000000   400.000000   550.000000  1200.000000
max       5.000000         1500.000000  2400.000000  3000.000000  4665.000000

          BSubOdds      RKOOdds      BKOOdds   BlueHeight    BlueReach  \
count  5169.000000  5194.000000  5168.000000  6528.000000  6528.000000
mean   1100.497775   510.891606   636.463235    70.008688    71.717384
std     671.106177   426.563458   465.014634     3.584685     4.397201
min   -1250.000000  -550.000000  -400.000000    60.000000     0.000000
25%     600.000000   225.000000   310.000000    67.000000    69.000000
50%    1000.000000   420.000000   525.000000    70.000000    72.000000
75%    1450.000000   700.000000   875.000000    73.000000    75.000000
max    5000.000000  4000.000000  4000.000000    83.000000    84.000000

         RedHeight     RedReach      ROver35      BOver35      RWinPct  \
count  6528.000000  6528.000000  6528.000000  6528.000000  6067.000000
mean     70.000944    71.815336     0.114583     0.084865     0.631346
std       3.615340     4.382045     0.318543     0.278702     0.256596
min      60.000000    58.000000     0.000000     0.000000     0.000000
25%      67.000000    69.000000     0.000000     0.000000     0.500000
50%      70.000000    72.000000     0.000000     0.000000     0.666667
75%      73.000000    75.000000     0.000000     0.000000     0.777778
max      83.000000    84.500000     1.000000     1.000000     1.000000
```

```
           BWinPct  RedStrikingRatio  BlueStrikingRatio  RedTotalFights  \
count  5418.000000       5476.000000        5476.000000     6528.000000
mean      0.591404          0.513428           0.486572        7.194853
std       0.298916          0.158090           0.158090        6.761721
min       0.000000          0.000000           0.000000        0.000000
25%       0.500000          0.414774           0.395959        2.000000
50%       0.625000          0.507965           0.492035        5.000000
75%       0.800000          0.604041           0.585226       10.000000
max       1.000000          1.000000           1.000000       48.000000

       BlueTotalFights  RedIsGrappler  BlueIsGrappler  RedIsStriker  \
count      6528.000000    6528.000000     6528.000000   6528.000000
mean          5.009344       0.062653        0.059743      0.548866
std           5.502321       0.242357        0.237028      0.497644
min           0.000000       0.000000        0.000000      0.000000
25%           1.000000       0.000000        0.000000      0.000000
50%           3.000000       0.000000        0.000000      1.000000
75%           7.000000       0.000000        0.000000      1.000000
max          45.000000       1.000000        1.000000      1.000000

       BlueIsStriker  RGrapplerVBStriker  BGrapplerVRStriker  \
count    6528.000000         6528.000000         6528.000000
mean        0.472733            0.022212            0.025429
std         0.499294            0.147384            0.157436
min         0.000000            0.000000            0.000000
25%         0.000000            0.000000            0.000000
50%         0.000000            0.000000            0.000000
75%         1.000000            0.000000            0.000000
max         1.000000            1.000000            1.000000

       RedStrikingEfficiency  BlueStrikingEfficiency  RedTDEfficiency  \
count            6073.000000             5598.000000      6161.000000
mean                9.729212                9.149099         0.626869
std                 9.535491                9.800691         0.756785
min                 0.000000                0.000000         0.000000
25%                 1.906100                1.761800         0.110000
50%                 6.600000                4.076800         0.380800
75%                15.500186               14.755537         0.888000
max                81.180000               78.540000         8.330000

       BlueTDEfficiency  RedEffectiveTD  BlueEffectiveTD  EffectiveTDDif  \
count       5686.000000     5347.000000      4608.000000     4078.000000
```

| | | | | |
|---|---|---|---|---|
| mean | 0.605154 | 0.552095 | 0.538713 | -0.010168 |
| std | 0.810902 | 0.902846 | 0.876956 | 1.243102 |
| min | 0.000000 | 0.000000 | 0.000000 | -10.732888 |
| 25% | 0.062500 | 0.052627 | 0.000000 | -0.380948 |
| 50% | 0.330000 | 0.293332 | 0.285700 | 0.000000 |
| 75% | 0.831017 | 0.666667 | 0.666667 | 0.353956 |
| max | 9.000000 | 11.212121 | 11.212121 | 11.170016 |

| | RedSize | BlueSize | FavoriteWins | draw_diff |
|---|---|---|---|---|
| count | 6528.000000 | 6528.000000 | 6528.000000 | 6528.000000 |
| mean | 141.816281 | 141.726072 | 0.649969 | -0.007966 |
| std | 7.797873 | 7.744036 | 0.477016 | 0.232091 |
| min | 120.000000 | 66.000000 | 0.000000 | -2.000000 |
| 25% | 136.000000 | 136.000000 | 0.000000 | 0.000000 |
| 50% | 142.000000 | 142.000000 | 1.000000 | 0.000000 |
| 75% | 147.000000 | 147.000000 | 1.000000 | 0.000000 |
| max | 167.500000 | 167.000000 | 1.000000 | 2.000000 |

| | avg_sig_str_pct_diff | avg_TD_pct_diff | win_by_Decision_Majority_diff |
|---|---|---|---|
| count | 5641.000000 | 5575.000000 | 6528.000000 |
| mean | -0.008139 | -0.019948 | -0.009191 |
| std | 0.138355 | 0.311780 | 0.208392 |
| min | -0.820000 | -1.000000 | -2.000000 |
| 25% | -0.080000 | -0.210000 | 0.000000 |
| 50% | 0.000000 | 0.000000 | 0.000000 |
| 75% | 0.077000 | 0.161000 | 0.000000 |
| max | 0.570000 | 1.000000 | 1.000000 |

| | win_by_Decision_Split_diff | win_by_Decision_Unanimous_diff |
|---|---|---|
| count | 6528.000000 | 6528.000000 |
| mean | -0.121017 | -0.518689 |
| std | 0.873700 | 2.107779 |
| min | -5.000000 | -11.000000 |
| 25% | 0.000000 | -1.000000 |
| 50% | 0.000000 | 0.000000 |
| 75% | 0.000000 | 0.000000 |
| max | 5.000000 | 11.000000 |

| | win_by_TKO_Doctor_Stoppage_diff | odds_diff | ev_diff | AgeGap |
|---|---|---|---|---|
| count | 6528.000000 | 6290.000000 | 6290.000000 | 6528.000000 |
| mean | -0.013327 | 175.464229 | 68.482524 | 4.142463 |
| std | 0.247506 | 525.482605 | 203.186608 | 3.147162 |

```
min                         -2.000000 -1975.000000  -766.666700     0.000000
25%                          0.000000  -280.000000   -63.333300     2.000000
50%                          0.000000   280.000000    63.333300     3.000000
75%                          0.000000   470.000000   175.784300     6.000000
max                          2.000000  3200.000000  1293.939400    17.000000

             Spread
count  6290.000000
mean    457.442607
std     312.470074
min       0.000000
25%     280.000000
50%     370.000000
75%     550.000000
max    3200.000000

Training dataset shape after leakage removal: (6528, 51)
```

In [284…
```
#shiyu
# To follow up on our previous discussion:
# You can keep all 152 variables — that's perfectly fine.
# Just make sure to check for highly correlated features and remove or combine
# any that show strong correlations (for example, |r| > 0.9),
# so your model doesn't suffer from redundancy.
```

In [285…
```
# for i in random_states:

#    - split the data
#    - preprocess it
#    - decide which hyperparameters you'll tune and what values you'll try
#    - for combo in hyperparameters:
#        - train your ML algo
#        - calculate training scores
#        - calculate validation scores
#    - select best model based on the mean and std validation scores
#    - predict the test set using the best model
#    - return your test score (generalization error)
#    - return the best model
```

In [286…
```
df.isnull().sum().sort_values(ascending=False)
```

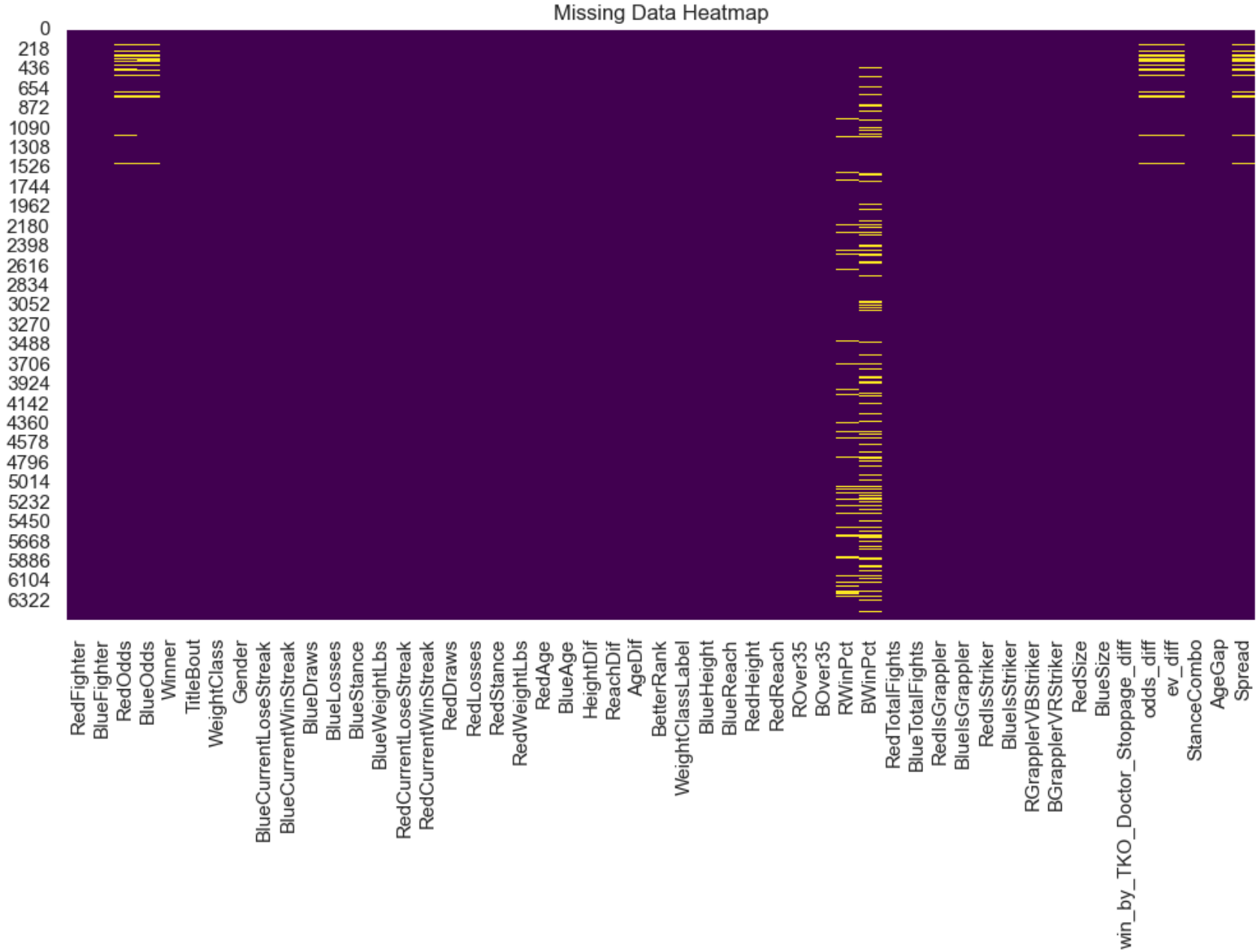
Out[286…

| | |
|---|---|
| BWinPct | 1110 |
| RWinPct | 461 |
| Spread | 238 |
| ev_diff | 238 |
| odds_diff | 238 |
| RedOdds | 227 |
| BlueOdds | 226 |
| BlueStance | 3 |
| BlueIsGrappler | 0 |
| RedHeight | 0 |
| RedReach | 0 |
| ROver35 | 0 |
| BOver35 | 0 |
| RedTotalFights | 0 |
| BlueTotalFights | 0 |
| RedIsGrappler | 0 |
| RGrapplerVBStriker | 0 |
| RedIsStriker | 0 |
| BlueIsStriker | 0 |
| BlueHeight | 0 |
| BGrapplerVRStriker | 0 |
| RedSize | 0 |
| BlueSize | 0 |
| win_by_TKO_Doctor_Stoppage_diff | 0 |
| StanceCombo | 0 |
| AgeGap | 0 |
| BlueReach | 0 |
| RedFighter | 0 |
| WeightClassLabel | 0 |
| RedCurrentLoseStreak | 0 |
| Winner | 0 |
| TitleBout | 0 |
| WeightClass | 0 |
| Gender | 0 |
| BlueCurrentLoseStreak | 0 |
| BlueCurrentWinStreak | 0 |
| BlueDraws | 0 |
| BlueLosses | 0 |
| BlueWeightLbs | 0 |
| RedCurrentWinStreak | 0 |
| BlueFighter | 0 |
| RedDraws | 0 |

```
RedLosses                       0
RedStance                       0
RedWeightLbs                    0
RedAge                          0
BlueAge                         0
HeightDif                       0
ReachDif                        0
AgeDif                          0
BetterRank                      0
dtype: int64
```

In [287…

```python
# Plot missing values heatmap
plt.figure(figsize=(12, 6))
sns.heatmap(df.isnull(), cbar=False, cmap="viridis")
plt.title("Missing Data Heatmap")
fig_name = "missing_data_heatmap2"
fig_path = os.path.join(output_folder, f"{fig_name}.png")
plt.savefig(fig_path, bbox_inches="tight", dpi=300)
plt.show()
plt.close()

# Missing values per column (%)
missing_percent = df.isnull().sum() * 100 / len(df)
print("Missing percentages per column:\n", missing_percent)

# Class balance
plt.figure(figsize=(6, 4))
sns.countplot(x='Winner', data=df)
plt.title('Class Balance: Winner')
fig_name = "class_balance_winner"
fig_path = os.path.join(output_folder, f"{fig_name}.png")
plt.savefig(fig_path, bbox_inches="tight", dpi=300)
plt.show()
plt.close()

# Display proportions
print("Class proportions:")
print(df['Winner'].value_counts(normalize=True))
```


Missing Data Heatmap
0
218
436
654
872
1090
1308
1526
1744
1962
2180
2398
2616
2834
3052
3270
3488
3706
3924
4142
4360
4578
4796
5014
5232
5450
5668
5886
6104
6322
RedFighter
BlueFighter
RedOdds
BlueOdds
Winner
TitleBout
WeightClass
Gender
BlueCurrentLoseStreak
BlueCurrentWinStreak
BlueDraws
BlueLosses
BlueStance
BlueWeightLbs
RedCurrentLoseStreak
RedCurrentWinStreak
RedDraws
RedLosses
RedStance
RedWeightLbs
RedAge
BlueAge
HeightDif
ReachDif
AgeDif
BetterRank
WeightClassLabel
BlueHeight
BlueReach
RedHeight
RedReach
ROver35
BOver35
RWinPct
BWinPct
RedTotalFights
BlueTotalFights
RedIsGrappler
BlueIsGrappler
RedIsStriker
BlueIsStriker
RGrapplerVBStriker
BGrapplerVRStriker
RedSize
BlueSize
win_by_TKO_Doctor_Stoppage_diff
odds_diff
ev_diff
StanceCombo
AgeGap
Spread

```
Missing percentages per column:
 RedFighter                  0.000000
BlueFighter                  0.000000
RedOdds                      3.477328
BlueOdds                     3.462010
Winner                       0.000000
TitleBout                    0.000000
WeightClass                  0.000000
Gender                       0.000000
BlueCurrentLoseStreak        0.000000
BlueCurrentWinStreak         0.000000
BlueDraws                    0.000000
BlueLosses                   0.000000
BlueStance                   0.045956
BlueWeightLbs                0.000000
RedCurrentLoseStreak         0.000000
RedCurrentWinStreak          0.000000
RedDraws                     0.000000
RedLosses                    0.000000
RedStance                    0.000000
RedWeightLbs                 0.000000
RedAge                       0.000000
BlueAge                      0.000000
HeightDif                    0.000000
ReachDif                     0.000000
AgeDif                       0.000000
BetterRank                   0.000000
WeightClassLabel             0.000000
BlueHeight                   0.000000
BlueReach                    0.000000
RedHeight                    0.000000
RedReach                     0.000000
ROver35                      0.000000
BOver35                      0.000000
RWinPct                      7.061887
BWinPct                     17.003676
RedTotalFights               0.000000
BlueTotalFights              0.000000
RedIsGrappler                0.000000
BlueIsGrappler               0.000000
RedIsStriker                 0.000000
BlueIsStriker                0.000000
```

```
RGrapplerVBStriker                0.000000
BGrapplerVRStriker                0.000000
RedSize                           0.000000
BlueSize                          0.000000
win_by_TKO_Doctor_Stoppage_diff   0.000000
odds_diff                         3.645833
ev_diff                           3.645833
StanceCombo                       0.000000
AgeGap                            0.000000
Spread                            3.645833
dtype: float64
```


```
Class proportions:
Winner
1    0.580116
0    0.419884
Name: proportion, dtype: float64
```

In [288…
```
# Separate categorical and numerical columns
cat_cols = df.select_dtypes(include=['object']).columns.tolist()
num_cols = df.select_dtypes(include=['int64', 'float64']).columns.tolist()
num_cols.remove('Winner') if 'Winner' in num_cols else None

print("Categorical columns:", cat_cols)
print("Numerical columns:", num_cols)
```

```
Categorical columns: ['TitleBout', 'BlueStance', 'RedStance', 'BetterRank', 'WeightClassLabel', 'StanceComb
o']
Numerical columns: ['RedFighter', 'BlueFighter', 'RedOdds', 'BlueOdds', 'WeightClass', 'Gender', 'BlueCurre
ntLoseStreak', 'BlueCurrentWinStreak', 'BlueDraws', 'BlueLosses', 'BlueWeightLbs', 'RedCurrentLoseStreak',
'RedCurrentWinStreak', 'RedDraws', 'RedLosses', 'RedWeightLbs', 'RedAge', 'BlueAge', 'HeightDif', 'ReachDi
f', 'AgeDif', 'BlueHeight', 'BlueReach', 'RedHeight', 'RedReach', 'ROver35', 'BOver35', 'RWinPct', 'BWinPc
t', 'RedTotalFights', 'BlueTotalFights', 'RedIsGrappler', 'BlueIsGrappler', 'RedIsStriker', 'BlueIsStrike
r', 'RGrapplerVBStriker', 'BGrapplerVRStriker', 'RedSize', 'BlueSize', 'win_by_TKO_Doctor_Stoppage_diff',
'odds_diff', 'ev_diff', 'AgeGap', 'Spread']
```

In [289…
```
#DummyClassifier = baseline to compare better model's performance later
# base_model = DummyClassifier(random_state=42)
# base_model.fit(X_train,y_train)
# preds = base_model.predict(X_valid)
# accuracy_score(y_valid, preds)
```

In [290…
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix

# ------------------------------------------------------------
# drop strongly corr ftrs  (|r| ≥ 0.50)
# ------------------------------------------------------------

num_df = df.select_dtypes(include=['int64','float64'])
corr_matrix = num_df.corr().abs()

# Keep only upper triangle
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(bool))

# Strong correlation pairs
strong_pairs = (
    upper.stack()
```

```
        .reset_index()
        .rename(columns={"level_0": "Feature_1", "level_1": "Feature_2", 0: "Correlation"})
)
strong_pairs = strong_pairs[strong_pairs["Correlation"] >= 0.50]

print("\nSTRONGLY CORRELATED PAIRS (|r| ≥ 0.50)")
print(strong_pairs.sort_values("Correlation", ascending=False))

# Choose lower variance feature from each pair
to_drop_corr = set()
for f1, f2, corr_val in strong_pairs.values:
    var1 = num_df[f1].var()
    var2 = num_df[f2].var()
    drop_feature = f1 if var1 < var2 else f2
    to_drop_corr.add(drop_feature)

print("\nDropping correlated features:")
print(to_drop_corr)

# Drop from df_clean
df_clean = df.drop(columns=list(to_drop_corr), errors='ignore')

# Rebuild num_df using cleaned df
num_df = df_clean.select_dtypes(include=['int64','float64'])

# ---------------------
# Load features + target
# ---------------------
X = df_clean.drop(columns=['Winner'])
y = df_clean['Winner']

print("Raw feature columns:")
print(X.columns.tolist())



print("\nFeatures considered for modeling:")
print(X.columns.tolist())

# ----------------------------------------
#                EDA
# ----------------------------------------
```

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

sns.set(style="whitegrid")

print("\n==============================")
print(" BASIC DATA OVERVIEW")
print("==============================")
print(df_clean.head())
print("\nShape:", df_clean.shape)
print("\nData types:\n", df_clean.dtypes)
print("\nMissing values per column:\n", df_clean.isna().sum().sort_values(ascending=False))


# ------------------------------------------------------------
# Target variable distribution
# ------------------------------------------------------------
plt.figure(figsize=(6,4))
sns.countplot(x=df_clean['Winner'])
plt.title("Distribution of Winner")
plt.xlabel("Winner (0 = Blue, 1 = Red)")
fig_name = "dist_winner_rb"
fig_path = os.path.join(output_folder, f"{fig_name}.png")
plt.savefig(fig_path, bbox_inches="tight", dpi=300)
plt.show()
plt.close()

num_df = X.select_dtypes(include=['int64','float64'])
cat_df = X.select_dtypes(include=['object', 'category'])

print("\nUpdated feature count after removing correlated vars:")
print("Numeric:", len(num_df.columns))
print("Categorical:", len(cat_df.columns))

# rebuild cleaned num_df
num_df = df_clean.select_dtypes(include=['int64','float64'])



plt.figure(figsize=(10,4))
sns.histplot(num_df.isna().sum(), bins=20)
```

```
plt.title("Missing Values Distribution (Numeric Columns)")
fig_name = "numeric_missing"
fig_path = os.path.join(output_folder, f"{fig_name}.png")
plt.savefig(fig_path, bbox_inches="tight", dpi=300)
plt.show()
plt.close()

print("\nNumeric Summary Stats:\n")
display(num_df.describe().T)



# ------------------------------------------------------------
# Correlation Heatmap (top 20 strongest correlations)
# ------------------------------------------------------------
plt.figure(figsize=(14,12))
corr = num_df.corr()

# keep strongest correlations with target
if 'Winner' in corr.columns:
    target_corr = corr['Winner'].abs().sort_values(ascending=False)[1:21]
    top_corr_cols = target_corr.index.tolist()
else:
    top_corr_cols = num_df.columns[:20]

sns.heatmap(num_df[top_corr_cols].corr(), annot=False, cmap='viridis')
plt.title("Correlation Heatmap (Top Correlated Numeric Features)")
fig_name = "corr_heatmap_num"
fig_path = os.path.join(output_folder, f"{fig_name}.png")
plt.savefig(fig_path, bbox_inches="tight", dpi=300)
plt.show()
plt.close()


# ------------------------------------------------------------
# Distribution plots for key variables
# ------------------------------------------------------------

cols_to_plot = [
    'ReachDif', 'HeightDif', 'Oddsdiff',
    'ev_diff', 'AgeGap'
]
```

```
cols_to_plot = [c for c in cols_to_plot if c in df_clean.columns]

for col in cols_to_plot:
    plt.figure(figsize=(7,4))
    sns.histplot(df_clean[col], kde=True)
    plt.title(f"Distribution: {col}")
    fig_name = "dist_cols_to_plot"
    fig_path = os.path.join(output_folder, f"{fig_name}.png")
    plt.savefig(fig_path, bbox_inches="tight", dpi=300)
    plt.show()
    plt.close()


# ------------------------------------------------------------
# Boxplots of numeric variables by Winner
# ------------------------------------------------------------
for col in cols_to_plot:
    plt.figure(figsize=(7,4))
    sns.boxplot(x=df_clean["Winner"], y=df_clean[col])
    plt.title(f"{col} by Winner")
    fig_name = "boxplot_winner"
    fig_path = os.path.join(output_folder, f"{fig_name}.png")
    plt.savefig(fig_path, bbox_inches="tight", dpi=300)
    plt.show()
    plt.close()


# ------------------------------------------------------------
# Categorical feature frequencies
# ------------------------------------------------------------
cat_df = df_clean.select_dtypes(include=['object','category'])

for col in cat_df.columns:
    plt.figure(figsize=(9,4))
    sns.countplot(y=df_clean[col], order=df_clean[col].value_counts().index)
    plt.title(f"Value Counts: {col}")
    plt.tight_layout()
    fig_name = "last_eda"
    fig_path = os.path.join(output_folder, f"{fig_name}.png")
    plt.savefig(fig_path, bbox_inches="tight", dpi=300)
    plt.show()
    plt.close()
```

```
print("\n-----------------")
print(" EDA COMPLETE")
print("-----------------")
```

STRONGLY CORRELATED PAIRS (|r| ≥ 0.50)

| | Feature_1 | Feature_2 | Correlation |
|---|---|---|---|
| 125 | RedOdds | odds_diff | 0.991816 |
| 166 | BlueOdds | odds_diff | 0.990186 |
| 984 | odds_diff | ev_diff | 0.984505 |
| 812 | RedReach | RedSize | 0.979474 |
| 126 | RedOdds | ev_diff | 0.976921 |
| 774 | BlueReach | BlueSize | 0.975792 |
| 167 | BlueOdds | ev_diff | 0.974324 |
| 793 | RedHeight | RedSize | 0.969695 |
| 433 | BlueWeightLbs | RedWeightLbs | 0.965231 |
| 87 | RedOdds | BlueOdds | 0.964238 |
| 753 | BlueHeight | BlueSize | 0.963343 |
| 780 | RedHeight | RedReach | 0.900542 |
| 569 | RedLosses | RedTotalFights | 0.895224 |
| 415 | BlueLosses | BlueTotalFights | 0.887705 |
| 737 | BlueHeight | BlueReach | 0.881352 |
| 605 | RedWeightLbs | RedSize | 0.809872 |
| 591 | RedWeightLbs | RedHeight | 0.806387 |
| 456 | BlueWeightLbs | BlueSize | 0.803452 |
| 439 | BlueWeightLbs | BlueHeight | 0.802761 |
| 455 | BlueWeightLbs | RedSize | 0.799574 |
| 441 | BlueWeightLbs | RedHeight | 0.794270 |
| 606 | RedWeightLbs | BlueSize | 0.793032 |
| 589 | RedWeightLbs | BlueHeight | 0.792587 |
| 592 | RedWeightLbs | RedReach | 0.775875 |
| 442 | BlueWeightLbs | RedReach | 0.767546 |
| 969 | RedSize | BlueSize | 0.767207 |
| 752 | BlueHeight | RedSize | 0.764074 |
| 440 | BlueWeightLbs | BlueReach | 0.760556 |
| 794 | RedHeight | BlueSize | 0.759768 |
| 738 | BlueHeight | RedHeight | 0.758014 |
| 590 | RedWeightLbs | BlueReach | 0.750499 |
| 813 | RedReach | BlueSize | 0.738414 |
| 739 | BlueHeight | RedReach | 0.734285 |
| 773 | BlueReach | RedSize | 0.728262 |
| 759 | BlueReach | RedHeight | 0.720101 |
| 760 | BlueReach | RedReach | 0.701839 |
| 927 | BlueIsGrappler | BGrapplerVRStriker | 0.640823 |
| 935 | RedIsStriker | BlueIsStriker | 0.638930 |
| 620 | RedAge | ROver35 | 0.636553 |
| 665 | HeightDif | ReachDif | 0.627953 |

```
210            WeightClass              Gender     0.622688
647                BlueAge             BOver35     0.593329
915          RedIsGrappler  RGrapplerVBStriker     0.582975
624                RedAge      RedTotalFights     0.547067
556             RedLosses              RedAge     0.544640
308  BlueCurrentLoseStreak             BWinPct     0.544207
344   BlueCurrentWinStreak             BWinPct     0.538211
128                RedOdds              Spread     0.533459
281                 Gender            BlueSize     0.533012
280                 Gender             RedSize     0.521222
265                 Gender           BlueReach     0.521174
267                 Gender            RedReach     0.518583
988                ev_diff              Spread     0.517156
264                 Gender          BlueHeight     0.512167

Dropping correlated features:
{'ROver35', 'ev_diff', 'HeightDif', 'RedSize', 'BWinPct', 'Gender', 'RedLosses', 'BlueReach', 'BGrapplerVRS
triker', 'BlueWeightLbs', 'RedAge', 'BlueOdds', 'BOver35', 'BlueSize', 'BlueLosses', 'BlueHeight', 'RGrappl
erVBStriker', 'RedHeight', 'RedIsStriker', 'RedReach', 'RedOdds'}
Raw feature columns:
['RedFighter', 'BlueFighter', 'TitleBout', 'WeightClass', 'BlueCurrentLoseStreak', 'BlueCurrentWinStreak',
'BlueDraws', 'BlueStance', 'RedCurrentLoseStreak', 'RedCurrentWinStreak', 'RedDraws', 'RedStance', 'RedWeig
htLbs', 'BlueAge', 'ReachDif', 'AgeDif', 'BetterRank', 'WeightClassLabel', 'RWinPct', 'RedTotalFights', 'Bl
ueTotalFights', 'RedIsGrappler', 'BlueIsGrappler', 'BlueIsStriker', 'win_by_TKO_Doctor_Stoppage_diff', 'odd
s_diff', 'StanceCombo', 'AgeGap', 'Spread']

Features considered for modeling:
['RedFighter', 'BlueFighter', 'TitleBout', 'WeightClass', 'BlueCurrentLoseStreak', 'BlueCurrentWinStreak',
'BlueDraws', 'BlueStance', 'RedCurrentLoseStreak', 'RedCurrentWinStreak', 'RedDraws', 'RedStance', 'RedWeig
htLbs', 'BlueAge', 'ReachDif', 'AgeDif', 'BetterRank', 'WeightClassLabel', 'RWinPct', 'RedTotalFights', 'Bl
ueTotalFights', 'RedIsGrappler', 'BlueIsGrappler', 'BlueIsStriker', 'win_by_TKO_Doctor_Stoppage_diff', 'odd
s_diff', 'StanceCombo', 'AgeGap', 'Spread']

==============================
 BASIC DATA OVERVIEW
==============================
   RedFighter  BlueFighter  Winner TitleBout  WeightClass  \
0          66         1009       1      True            3
1        1441          718       1     False            8
2         307           67       1     False            4
3         221         1071       1     False            2
4        1183          524       0     False            2
```

| | BlueCurrentLoseStreak | BlueCurrentWinStreak | BlueDraws | BlueStance |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Orthodox |
| 1 | 0 | 8 | 0 | Orthodox |
| 2 | 0 | 4 | 0 | Orthodox |
| 3 | 2 | 0 | 0 | Southpaw |
| 4 | 0 | 1 | 1 | Orthodox |

| | RedCurrentLoseStreak | RedCurrentWinStreak | RedDraws | RedStance |
|---|---|---|---|---|
| 0 | 0 | 6 | 0 | Orthodox |
| 1 | 0 | 6 | 0 | Orthodox |
| 2 | 0 | 1 | 0 | Orthodox |
| 3 | 1 | 0 | 0 | Southpaw |
| 4 | 0 | 1 | 0 | Orthodox |

| | RedWeightLbs | BlueAge | ReachDif | AgeDif | BetterRank | WeightClassLabel |
|---|---|---|---|---|---|---|
| 0 | 125 | 31 | 5.08 | -3 | Red | Featherweight |
| 1 | 170 | 27 | -7.62 | -3 | Red | Women's Strawweight |
| 2 | 245 | 36 | -2.54 | 2 | Red | Light Heavyweight |
| 3 | 145 | 36 | 0.00 | 6 | Red | Heavyweight |
| 4 | 145 | 33 | -5.08 | -3 | neither | Heavyweight |

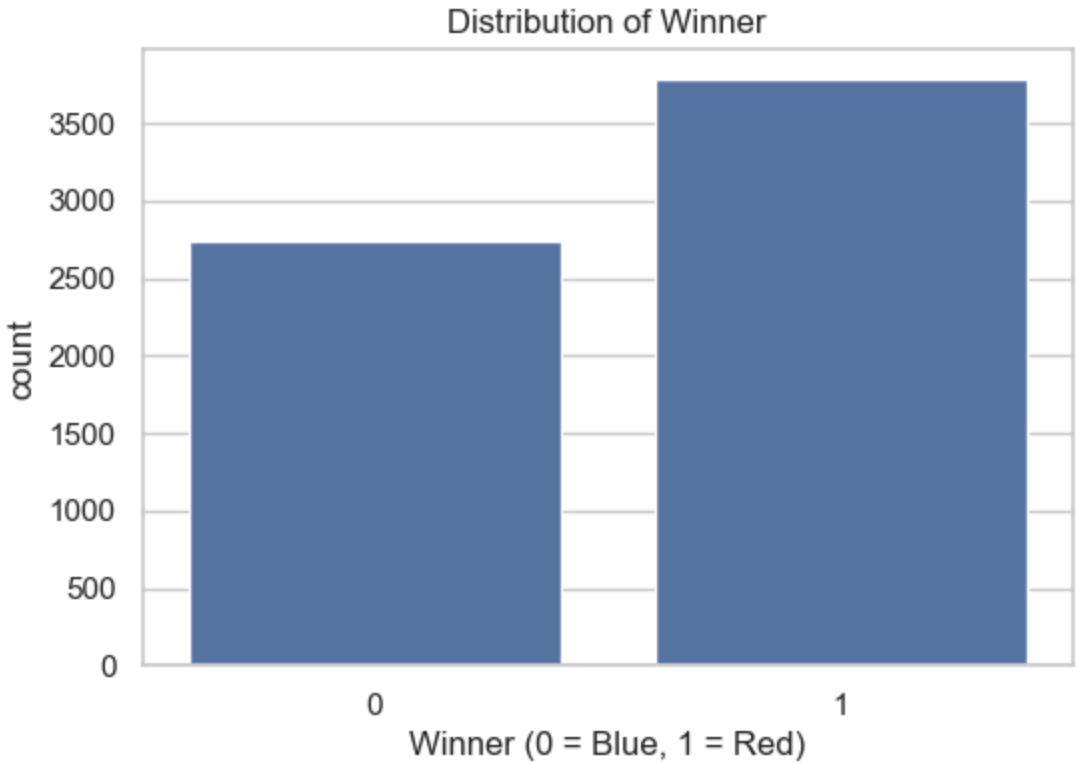
| | RWinPct | RedTotalFights | BlueTotalFights | RedIsGrappler | BlueIsGrappler |
|---|---|---|---|---|---|
| 0 | 0.800000 | 15 | 0 | 0 | 0 |
| 1 | 1.000000 | 6 | 8 | 0 | 0 |
| 2 | 0.818182 | 11 | 16 | 0 | 0 |
| 3 | 0.777778 | 9 | 3 | 0 | 0 |
| 4 | 0.625000 | 8 | 7 | 0 | 0 |

| | BlueIsStriker | win_by_TKO_Doctor_Stoppage_diff | odds_diff |
|---|---|---|---|
| 0 | 0 | 0 | 465.0 |
| 1 | 0 | 0 | 505.0 |
| 2 | 0 | 0 | 680.0 |
| 3 | 0 | 0 | 1575.0 |
| 4 | 0 | 0 | 240.0 |

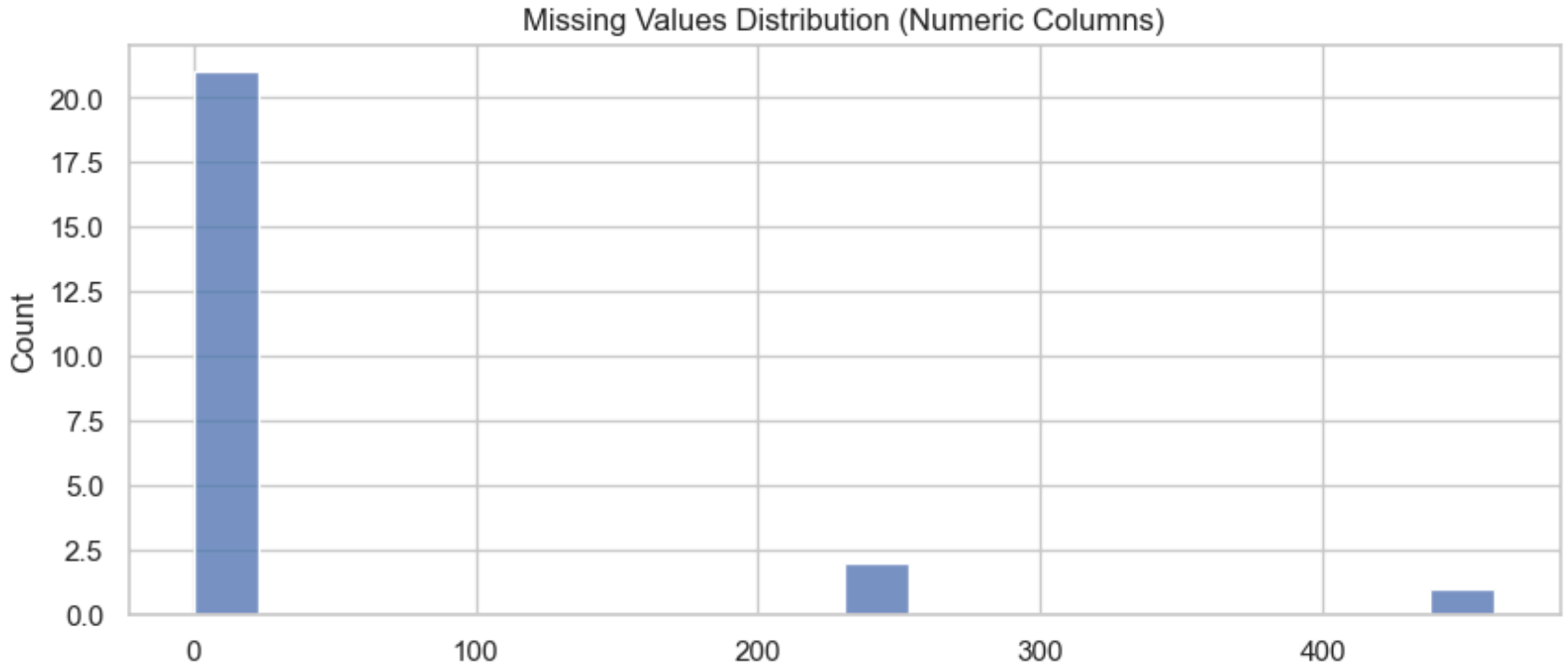
| | StanceCombo | AgeGap | Spread |
|---|---|---|---|
| 0 | Orthodox vs Orthodox | 3 | 465.0 |
| 1 | Orthodox vs Orthodox | 3 | 505.0 |
| 2 | Orthodox vs Orthodox | 2 | 680.0 |
| 3 | Southpaw vs Southpaw | 6 | 1575.0 |
| 4 | Orthodox vs Orthodox | 3 | 240.0 |

```
Shape: (6528, 30)

Data types:
 RedFighter                      int64
BlueFighter                      int64
Winner                           int64
TitleBout                       object
WeightClass                      int64
BlueCurrentLoseStreak            int64
BlueCurrentWinStreak             int64
BlueDraws                        int64
BlueStance                      object
RedCurrentLoseStreak             int64
RedCurrentWinStreak              int64
RedDraws                         int64
RedStance                       object
RedWeightLbs                     int64
BlueAge                          int64
ReachDif                       float64
AgeDif                           int64
BetterRank                      object
WeightClassLabel                object
RWinPct                        float64
RedTotalFights                   int64
BlueTotalFights                  int64
RedIsGrappler                    int64
BlueIsGrappler                   int64
BlueIsStriker                    int64
win_by_TKO_Doctor_Stoppage_diff  int64
odds_diff                      float64
StanceCombo                     object
AgeGap                           int64
Spread                         float64
dtype: object

Missing values per column:
 RWinPct                         461
Spread                           238
odds_diff                        238
BlueStance                         3
AgeDif                             0
```

```
AgeGap                            0
StanceCombo                       0
win_by_TKO_Doctor_Stoppage_diff   0
BlueIsStriker                     0
BlueIsGrappler                    0
RedIsGrappler                     0
BlueTotalFights                   0
RedTotalFights                    0
WeightClassLabel                  0
BetterRank                        0
RedFighter                        0
BlueFighter                       0
BlueAge                           0
RedWeightLbs                      0
RedStance                         0
RedDraws                          0
RedCurrentWinStreak               0
RedCurrentLoseStreak              0
BlueDraws                         0
BlueCurrentWinStreak              0
BlueCurrentLoseStreak             0
WeightClass                       0
TitleBout                         0
Winner                            0
ReachDif                          0
dtype: int64
```



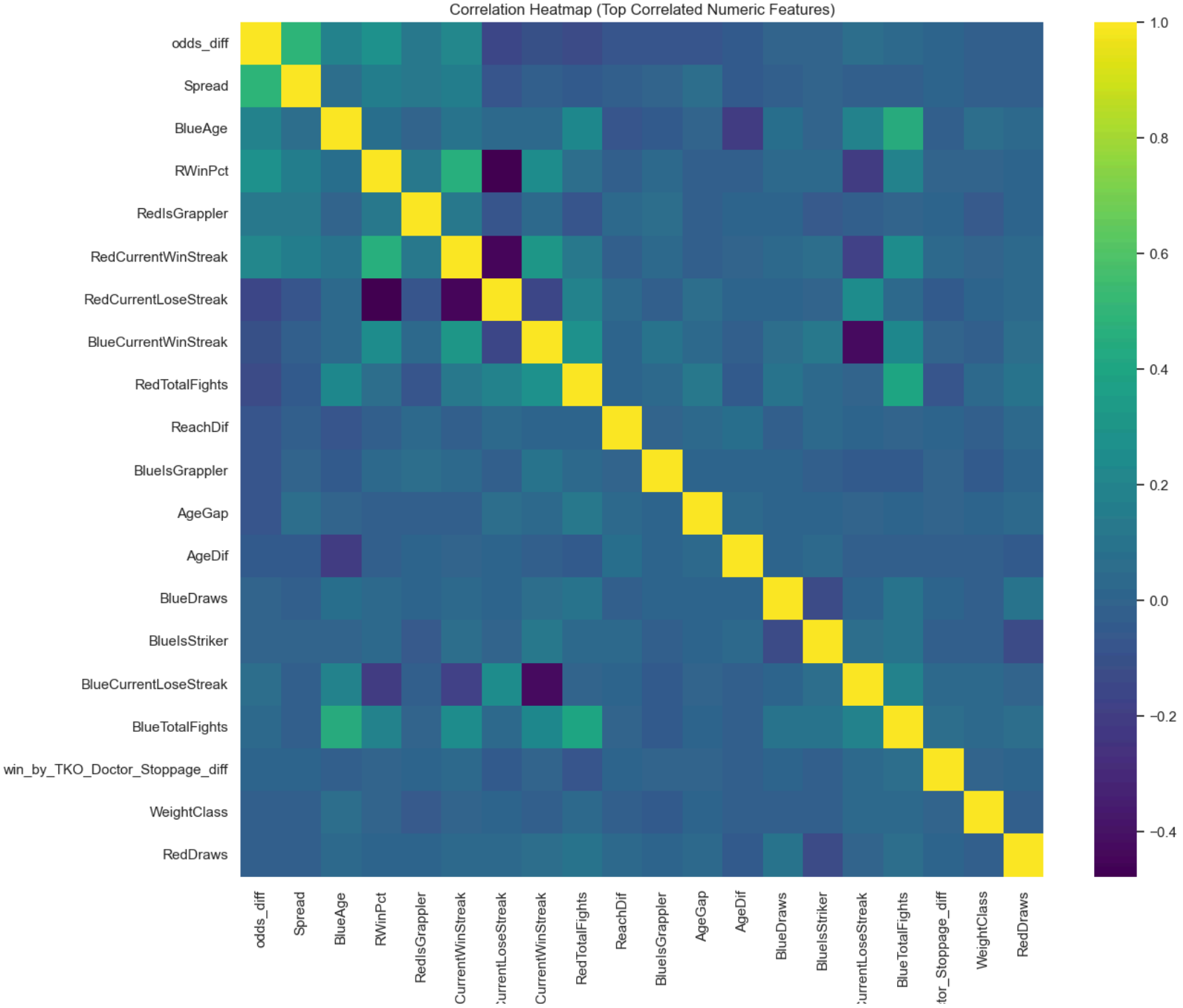

```
Updated feature count after removing correlated vars:
Numeric: 23
Categorical: 6
```



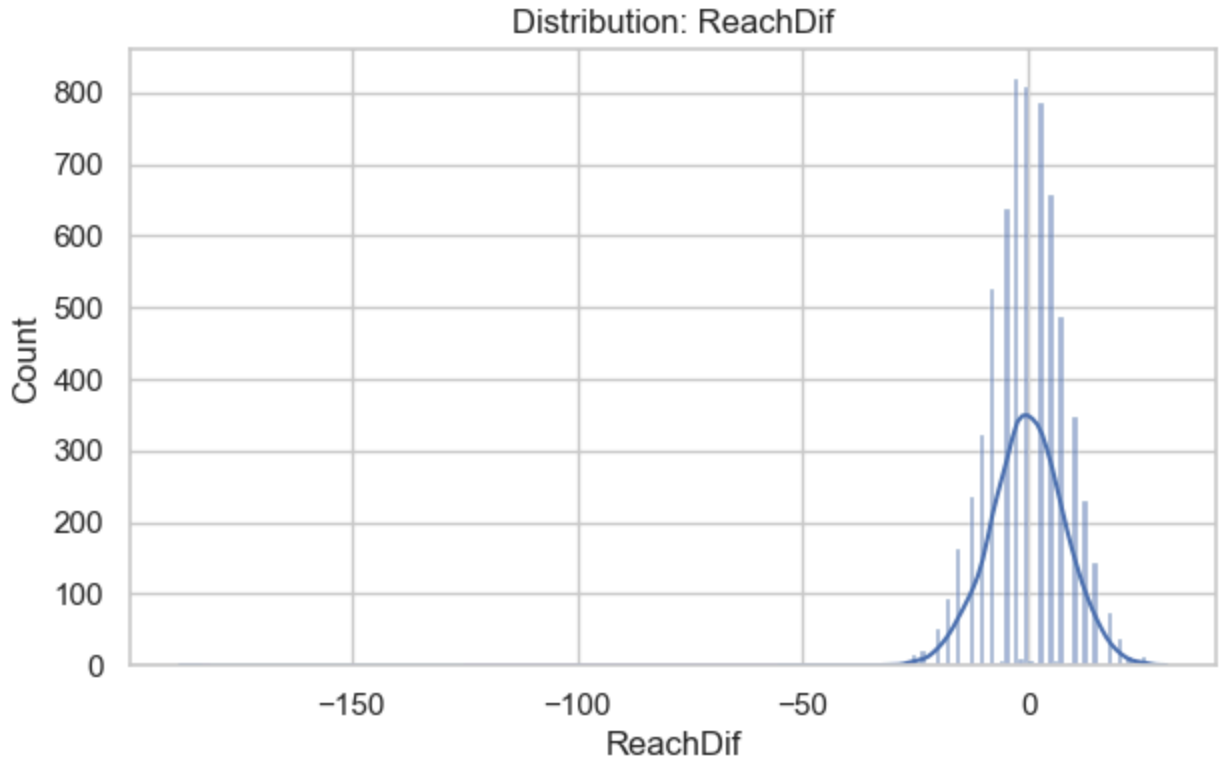

Numeric Summary Stats:

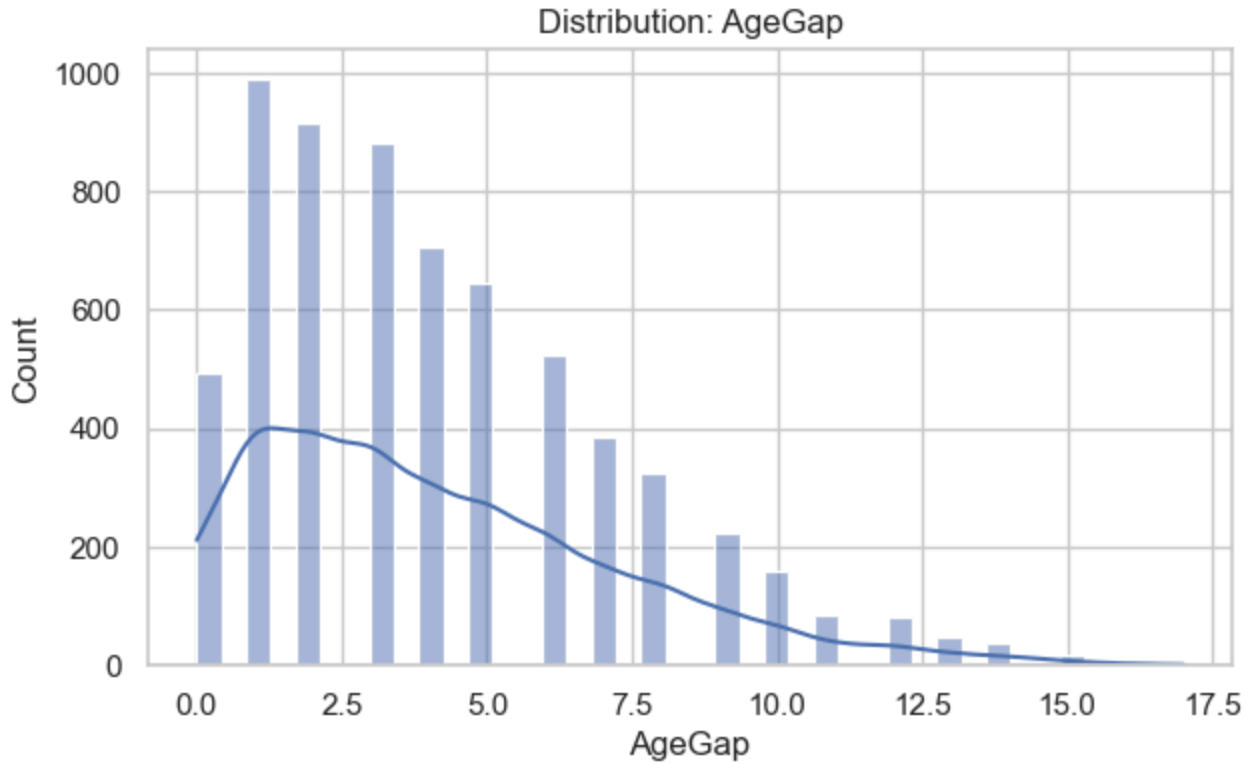
| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **RedFighter** | 6528.0 | 819.761795 | 476.188466 | 0.00 | 419.00 | 806.000000 | 1241.250000 | 1660.00 |
| **BlueFighter** | 6528.0 | 942.707874 | 556.368724 | 0.00 | 462.00 | 922.500000 | 1427.000000 | 1921.00 |
| **Winner** | 6528.0 | 0.580116 | 0.493577 | 0.00 | 0.00 | 1.000000 | 1.000000 | 1.00 |
| **WeightClass** | 6528.0 | 5.503370 | 3.200274 | 0.00 | 3.00 | 6.000000 | 8.000000 | 12.00 |
| **BlueCurrentLoseStreak** | 6528.0 | 0.501072 | 0.794303 | 0.00 | 0.00 | 0.000000 | 1.000000 | 6.00 |
| **BlueCurrentWinStreak** | 6528.0 | 0.957567 | 1.406786 | 0.00 | 0.00 | 0.000000 | 1.000000 | 12.00 |
| **BlueDraws** | 6528.0 | 0.023131 | 0.156327 | 0.00 | 0.00 | 0.000000 | 0.000000 | 2.00 |
| **RedCurrentLoseStreak** | 6528.0 | 0.622243 | 0.872301 | 0.00 | 0.00 | 0.000000 | 1.000000 | 7.00 |
| **RedCurrentWinStreak** | 6528.0 | 1.101562 | 1.760767 | 0.00 | 0.00 | 0.000000 | 2.000000 | 18.00 |
| **RedDraws** | 6528.0 | 0.031097 | 0.187999 | 0.00 | 0.00 | 0.000000 | 0.000000 | 2.00 |
| **RedWeightLbs** | 6528.0 | 163.621324 | 34.846543 | 115.00 | 135.00 | 155.000000 | 185.000000 | 265.00 |
| **BlueAge** | 6528.0 | 29.805607 | 3.959623 | 19.00 | 27.00 | 30.000000 | 32.000000 | 47.00 |
| **ReachDif** | 6528.0 | -0.299271 | 9.132413 | -187.96 | -5.08 | 0.000000 | 5.080000 | 30.48 |
| **AgeDif** | 6528.0 | 0.096814 | 5.201719 | -17.00 | -3.00 | 0.000000 | 4.000000 | 17.00 |
| **RWinPct** | 6067.0 | 0.631346 | 0.256596 | 0.00 | 0.50 | 0.666667 | 0.777778 | 1.00 |
| **RedTotalFights** | 6528.0 | 7.194853 | 6.761721 | 0.00 | 2.00 | 5.000000 | 10.000000 | 48.00 |
| **BlueTotalFights** | 6528.0 | 5.009344 | 5.502321 | 0.00 | 1.00 | 3.000000 | 7.000000 | 45.00 |
| **RedIsGrappler** | 6528.0 | 0.062653 | 0.242357 | 0.00 | 0.00 | 0.000000 | 0.000000 | 1.00 |
| **BlueIsGrappler** | 6528.0 | 0.059743 | 0.237028 | 0.00 | 0.00 | 0.000000 | 0.000000 | 1.00 |
| **BlueIsStriker** | 6528.0 | 0.472733 | 0.499294 | 0.00 | 0.00 | 0.000000 | 1.000000 | 1.00 |
| **win_by_TKO_Doctor_Stoppage_diff** | 6528.0 | -0.013327 | 0.247506 | -2.00 | 0.00 | 0.000000 | 0.000000 | 2.00 |
| **odds_diff** | 6290.0 | 175.464229 | 525.482605 | -1975.00 | -280.00 | 280.000000 | 470.000000 | 3200.00 |
| **AgeGap** | 6528.0 | 4.142463 | 3.147162 | 0.00 | 2.00 | 3.000000 | 6.000000 | 17.00 |

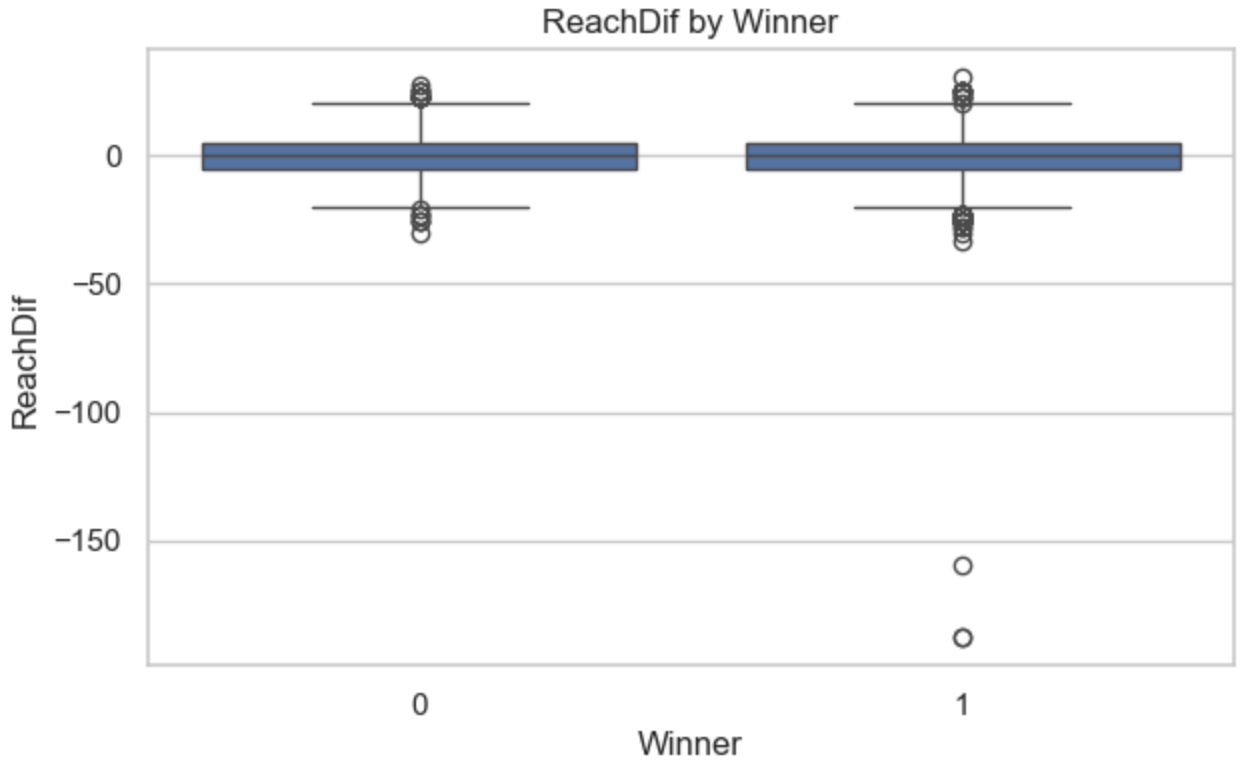
| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **Spread** | 6290.0 | 457.442607 | 312.470074 | 0.00 | 280.00 | 370.000000 | 550.000000 | 3200.00 |

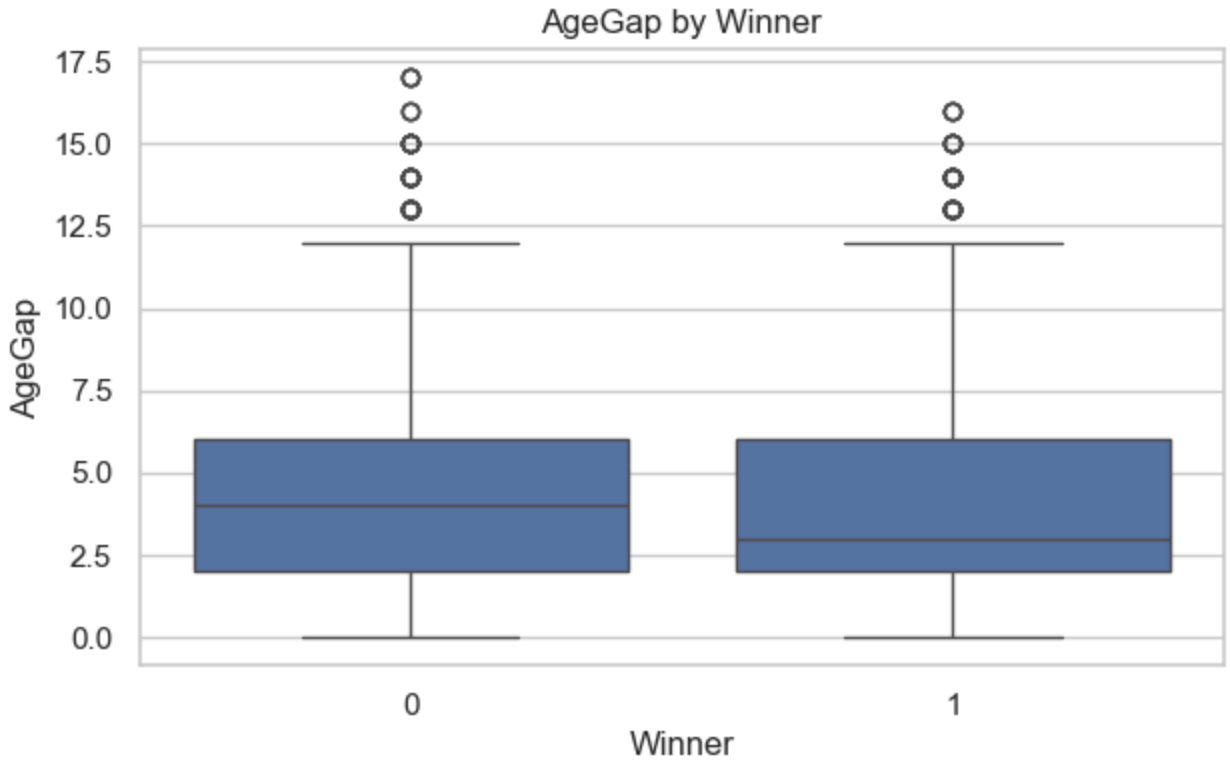

Correlation Heatmap (Top Correlated Numeric Features)
odds_diff
Spread
BlueAge
RWinPct
RedIsGrappler
RedCurrentWinStreak
RedCurrentLoseStreak
BlueCurrentWinStreak
RedTotalFights
ReachDif
BlueIsGrappler
AgeGap
AgeDif
BlueDraws
BlueIsStriker
BlueCurrentLoseStreak
BlueTotalFights
win_by_TKO_Doctor_Stoppage_diff
WeightClass
RedDraws
odds_diff
Spread
BlueAge
RWinPct
RedIsGrappler
CurrentWinStreak
urrentLoseStreak
CurrentWinStreak
RedTotalFights
ReachDif
BlueIsGrappler
AgeGap
AgeDif
BlueDraws
BlueIsStriker
urrentLoseStreak
BlueTotalFights
tor_Stoppage_diff
WeightClass
RedDraws
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

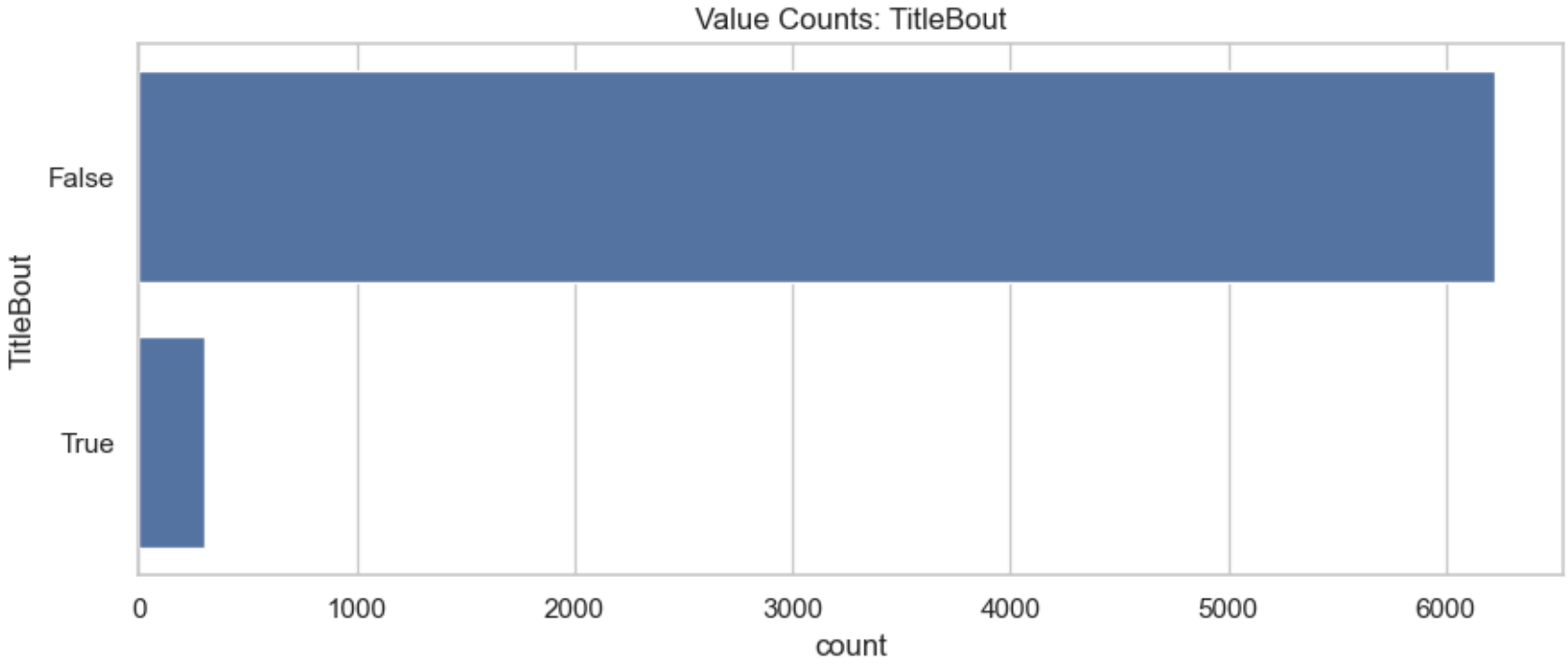
Red RedC Blue BlueC win_by_TKO_Doc

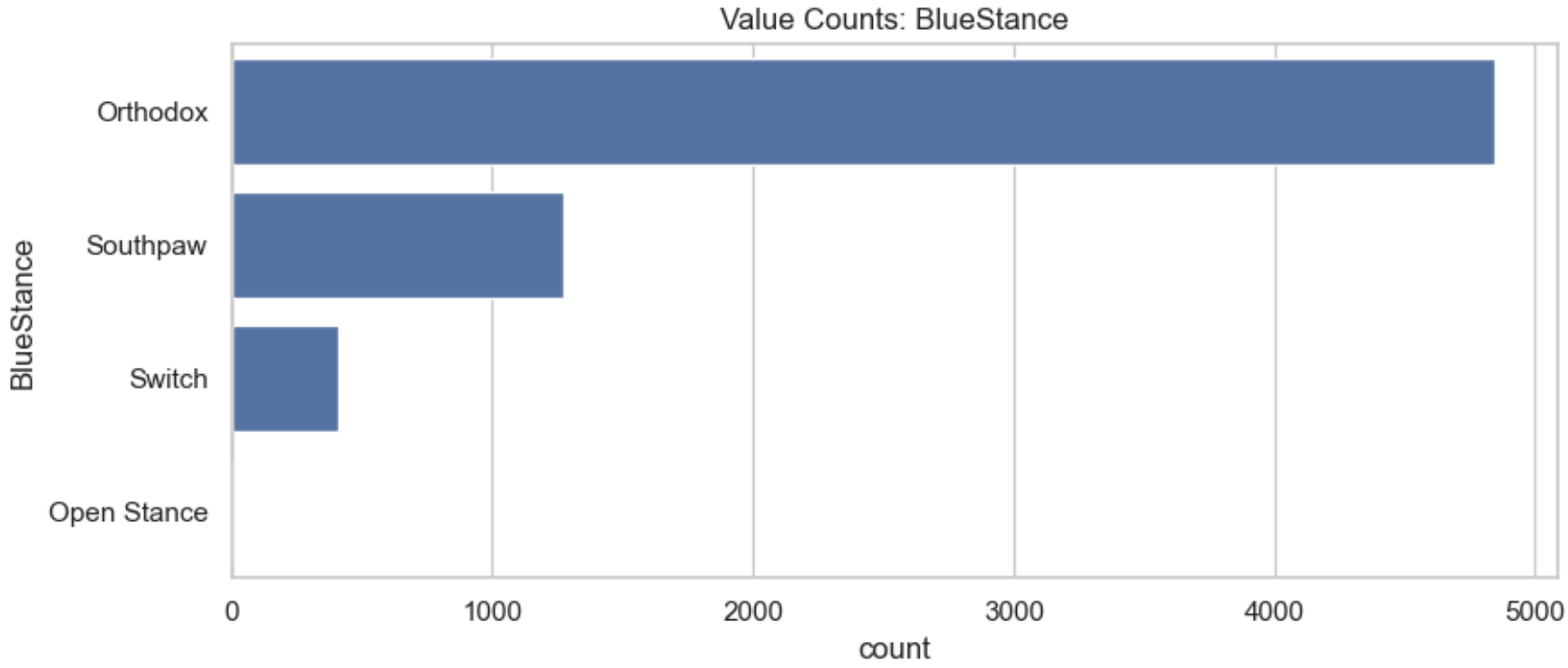


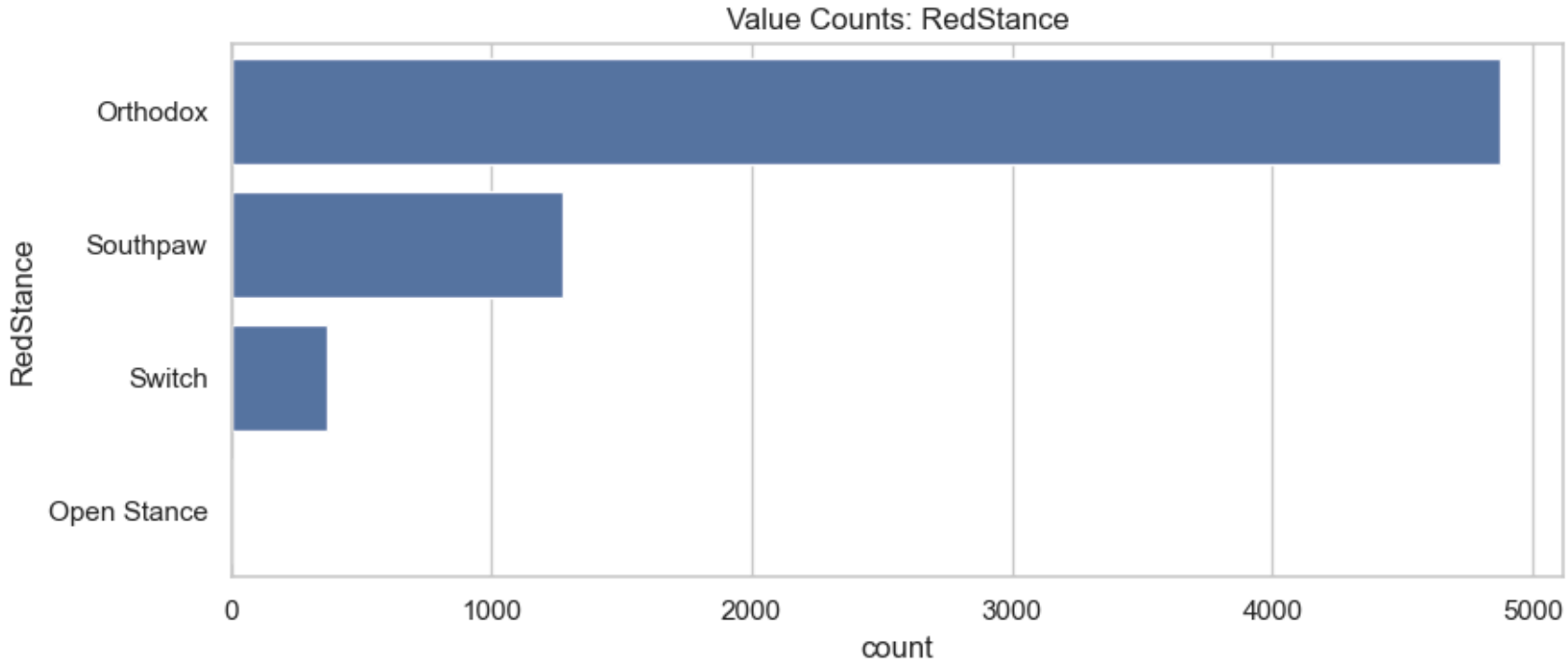

Distribution: AgeGap
Count
1000
800
600
400
200
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
AgeGap

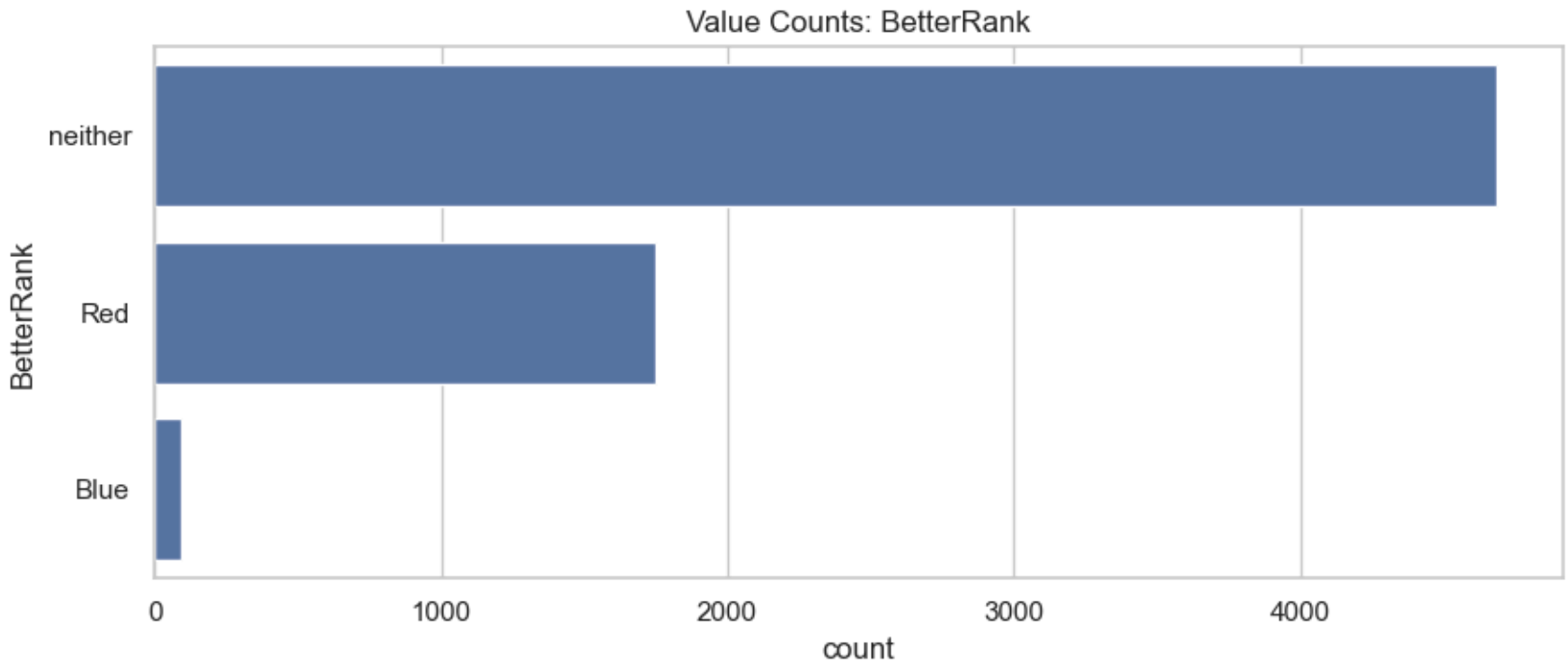

ReachDif by Winner
0
−50
−100
−150
ReachDif
0
1
Winner

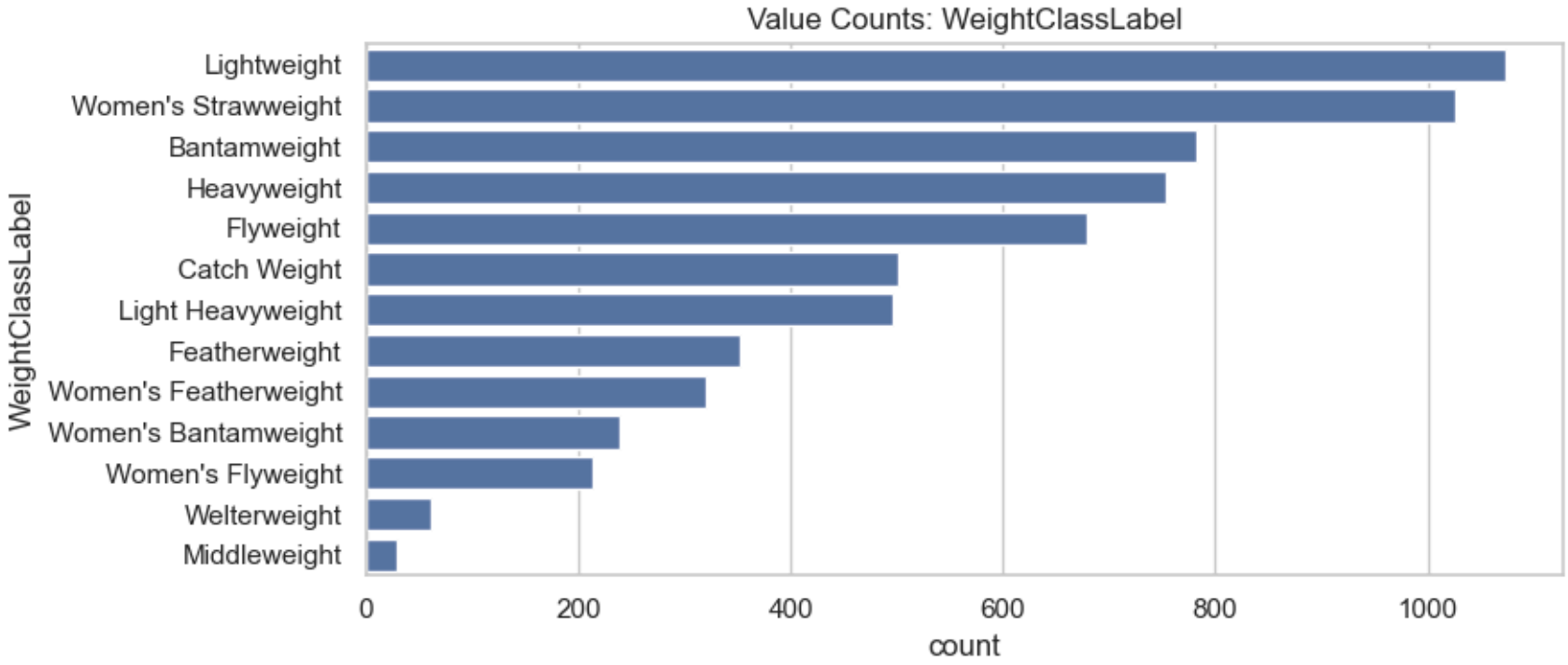

AgeGap by Winner
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
AgeGap
0
1
Winner

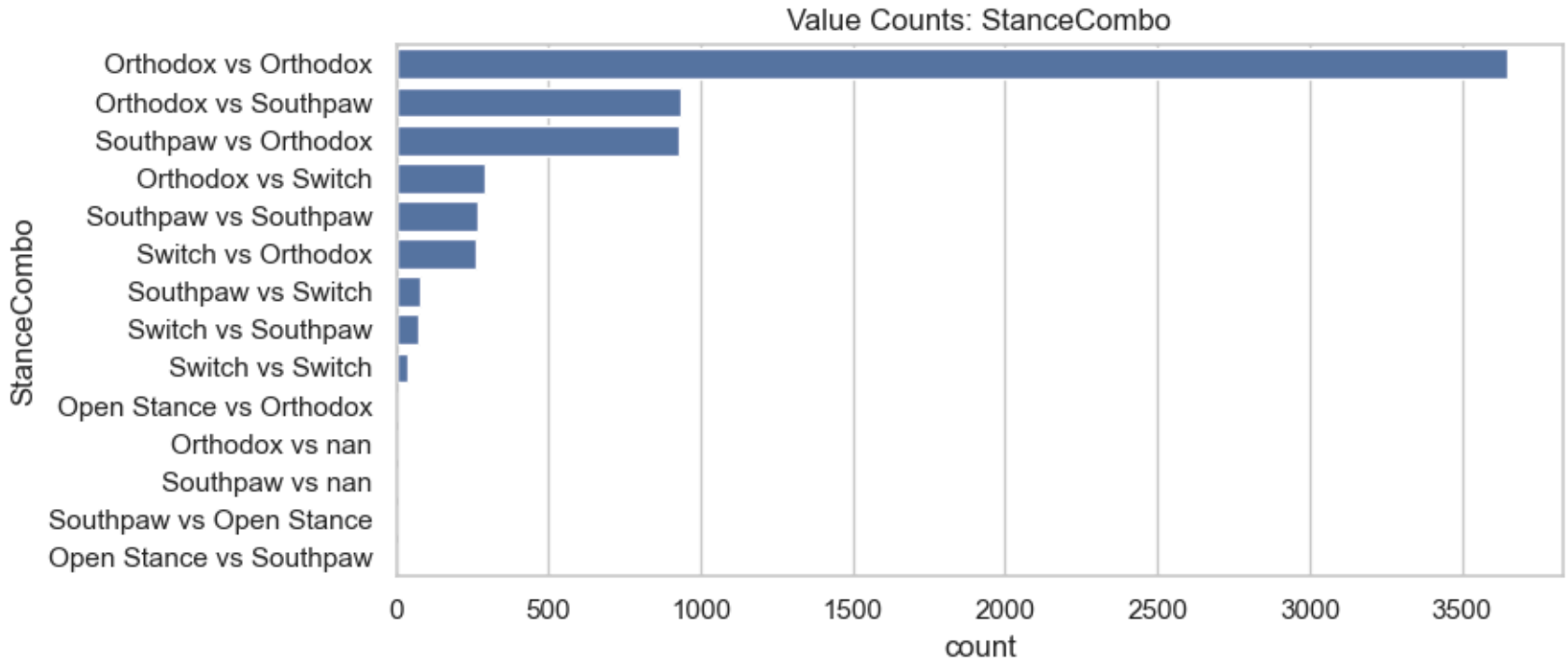

Value Counts: TitleBout
TitleBout
False
True
0
1000
2000
3000
4000
5000
6000
count


Value Counts: BlueStance
BlueStance
Orthodox
Southpaw
Switch
Open Stance
0
1000
2000
3000
4000
5000
count


Value Counts: RedStance
RedStance
Orthodox
Southpaw
Switch
Open Stance
0
1000
2000
3000
4000
5000
count


Value Counts: BetterRank
neither
Red
Blue
BetterRank
0
1000
2000
3000
4000
count


Value Counts: WeightClassLabel
WeightClassLabel
Lightweight
Women's Strawweight
Bantamweight
Heavyweight
Flyweight
Catch Weight
Light Heavyweight
Featherweight
Women's Featherweight
Women's Bantamweight
Women's Flyweight
Welterweight
Middleweight
0
200
400
600
800
1000
count




```
-----------------
 EDA COMPLETE
-----------------
```

In [291…

```python
# ----------------------------------------
# MICE and preprocess
# ----------------------------------------
# ----------------------------------------
#   split — matchup safe — for leakage
# ----------------------------------------
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import GroupShuffleSplit

print("\n----------------------------------------")
print(" Creating matchup-safe split")
print("----------------------------------------")

# Encode target
le = LabelEncoder()
y_enc = le.fit_transform(df_clean["Winner"])
```

```
# Build matchup ID
df_clean["MatchupID"] = df_clean.apply(
    lambda row: "_".join(sorted([
        str(row.get("RedFighter", "")),
        str(row.get("BlueFighter", ""))
    ])),
    axis=1
)

# Build X and y
X = df_clean.drop(columns=["Winner"])
y = y_enc

# Group-based split
gss = GroupShuffleSplit(n_splits=1, test_size=0.20, random_state=42)
train_idx, test_idx = next(gss.split(X, y, groups=df_clean["MatchupID"]))

X_train = X.iloc[train_idx].reset_index(drop=True)
X_test  = X.iloc[test_idx].reset_index(drop=True)
y_train = y[train_idx]
y_test  = y[test_idx]

# Drop identifiers (not allowed in model)
drop_cols = ["RedFighter", "BlueFighter", "MatchupID", "win_by_TKO_Doctor_Stoppage_diff"] #win_by_TKO_Docto
X_train = X_train.drop(columns=[c for c in drop_cols if c in X_train.columns])
X_test  = X_test.drop(columns=[c for c in drop_cols if c in X_test.columns])

# ----------------------------------------
# Recompute num and cat cols
# ----------------------------------------
num_cols = X_train.select_dtypes(include=["int64", "float64"]).columns.tolist()
cat_cols = X_train.select_dtypes(include=["object", "category"]).columns.tolist()

print("\nUpdated numeric columns:", len(num_cols))
print("Updated categorical columns:", len(cat_cols))

# ----------------------------------------
#   MICEFOREST — for num training data
# ----------------------------------------
import miceforest as mf
```

```
print("\nRunning MICE-Forest Imputation on TRAINING numeric columns...")

# Extract numeric-only
X_train_num = X_train[num_cols].reset_index(drop=True)
X_test_num  = X_test[num_cols].reset_index(drop=True)

# Ensure column names strings
X_train_num.columns = X_train_num.columns.astype(str)
X_test_num.columns  = X_test_num.columns.astype(str)

# ---- FIT MICE ON TRAIN ONLY ----
kernel = mf.ImputationKernel(
    data=X_train_num,
    save_all_iterations_data=True,
    random_state=42
)

kernel.mice(
    iterations=3,
    n_estimators=50
)

# ---- COMPLETE TRAIN ----
completed_train = kernel.complete_data()

# ---- IMPUTE TEST USING THE TRAINING KERNEL ----
completed_test = kernel.impute_new_data(X_test_num).complete_data()

# Put imputed values back
X_train[num_cols] = completed_train.values
X_test[num_cols]  = completed_test.values

print("MICE Imputation Completed (train-fit only).\n")

# ----------------------------------------
#   preprocessing pipeline
# ----------------------------------------
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
```

```
num_transformer = Pipeline(steps=[
    ('scaler', StandardScaler())
])

cat_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='constant', fill_value='Missing')),
    ('onehot', OneHotEncoder(handle_unknown='ignore', sparse_output=False))
])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', num_transformer, num_cols),
        ('cat', cat_transformer, cat_cols)
    ]
)
```

```
----------------------------------------
 Creating matchup-safe split
----------------------------------------

Updated numeric columns: 20
Updated categorical columns: 6

Running MICE-Forest Imputation on TRAINING numeric columns...
MICE Imputation Completed (train-fit only).
```

In [292…

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import shap
import warnings
warnings.filterwarnings('ignore')

from sklearn.base import clone
from sklearn.model_selection import StratifiedKFold, GridSearchCV
from sklearn.model_selection import (
    train_test_split, GridSearchCV, KFold
)
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, OneHotEncoder
```

```
from sklearn.compose import ColumnTransformer
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.metrics import (
    accuracy_score, confusion_matrix, roc_auc_score, classification_report
)
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.model_selection import GroupKFold


# ----------------------------------------------------------------------
#  Machine Learning Pipelines
# ----------------------------------------------------------------------
pipelines = {
    'LogisticRegression': Pipeline(steps=[
        ('preprocessor', preprocessor),
        ('classifier', LogisticRegression(max_iter=1000))
    ]),

    'RandomForest': Pipeline(steps=[
        ('preprocessor', preprocessor),
        ('classifier', RandomForestClassifier(random_state=42))
    ]),

    'SVM': Pipeline(steps=[
        ('preprocessor', preprocessor),
        ('classifier', SVC(probability=True))
    ]),

    'XGBoost': Pipeline(steps=[
        ('preprocessor', preprocessor),
        ('classifier', XGBClassifier(eval_metric='logloss'))
    ])
}

# ----------------------------------------------------------------------
#  Hyperparameter Grids
# ----------------------------------------------------------------------
#shifted LR parameters down as 0.1 was best performing in first run
#ended up shifting several parameters down post-ppt
```

```
param_grids = {
    'LogisticRegression': {'classifier__C': [0.0001, 0.001, 0.01, 0.1]},
    'RandomForest': {
        'classifier__n_estimators': [100, 300],
        'classifier__max_depth': [None, 10, 20],
        'classifier__min_samples_split': [5,10]
    },

    'SVM': {
        'classifier__C': [0.001, 0.01, 0.1],
        'classifier__kernel': ['rbf', 'linear']
    },

    'XGBoost': {
        'classifier__n_estimators': [100, 300],
        'classifier__max_depth': [1, 3],
        'classifier__learning_rate': [0.001, 0.01]
    }
}
```

In [293…

```
# ----------------------------------------
# (3 seeds × 5-fold CV)
# ----------------------------------------
import joblib

nr_states = 3
kf_splits = 5
results_summary = {}

for name, base_pipeline in pipelines.items():

    print(f"\n----------------------------------------")
    print(f"Training {name}")
    print(f"----------------------------------------")

    test_scores = []
    best_param_list = []
    final_models = []

    for seed in range(1, nr_states + 1):

        print(f"\n--- Random State {seed} ---")
```

```
        # Clone pipeline to ensure fresh model each iteration
        pipeline = clone(base_pipeline)

        # Ensure reproducibility for models that support random_state
        if name in ["RandomForest", "XGBoost", "LogisticRegression"]:
            pipeline.named_steps["classifier"].set_params(random_state=seed)

        # CV strategy
        cv = StratifiedKFold(
            n_splits=kf_splits, shuffle=True, random_state=seed
        )

        # Grid search
        grid = GridSearchCV(
            estimator=pipeline,
            param_grid=param_grids[name],
            scoring="accuracy",
            cv=cv,
            n_jobs=-1
        )

        grid.fit(X_train, y_train)

        # Store best estimator for this seed
        best_model = grid.best_estimator_
        final_models.append(best_model)
        best_param_list.append(grid.best_params_)

        # Evaluate on test set
        y_pred = best_model.predict(X_test)
        acc = accuracy_score(y_test, y_pred)
        test_scores.append(acc)

        print(f"Best params: {grid.best_params_}")
        print(f"CV accuracy: {grid.best_score_:.4f}")
        print(f"Test accuracy: {acc:.4f}")

    # ------------------------------------------
    # Aggregate results across random seeds - need average
    # ------------------------------------------
    avg_test_score = np.mean(test_scores)
```

```
    std_test_score = np.std(test_scores)
    best_test_score = np.max(test_scores)

    print(f"\nAverage Test Accuracy for {name}: {avg_test_score:.4f}")

    results_summary[name] = {
        "test_scores": test_scores,
        "avg_test_score": avg_test_score,
        "std_test_score": std_test_score,
        "best_test_score": best_test_score,
        "best_params": best_param_list,
        "models": final_models
    }

    # ------------------------------------------------
    # Save best model (pick the seed with highest test accuracy) for results in git repo
    # ------------------------------------------------
    best_model_idx = np.argmax(test_scores)
    best_model_overall = final_models[best_model_idx]

    save_path = f"best_model_{name}.pkl"
    joblib.dump(best_model_overall, save_path)

    print(f"Saved best {name} model → {save_path}")


# ----------------------------------------
#              COMPILE FINAL RESULTS TABLE
# ----------------------------------------

results_table = []

for model_name, info in results_summary.items():

    rows = {
        "Algorithm": model_name,
        "Best params (seed 1)": info["best_params"][0],
        "Mean test accuracy": round(info["avg_test_score"], 4),
        "Std test accuracy": round(info["std_test_score"], 4),
        "Best test accuracy (across seeds)": round(info["best_test_score"], 4)
    }
```

```
    results_table.append(rows)

results_table = pd.DataFrame(results_table)

print("\n\n----------------------------------------")
print("       FINAL RESULTS TABLE")
print("----------------------------------------\n")
print(results_table)
```

```
----------------------------------------
Training LogisticRegression
----------------------------------------

--- Random State 1 ---
Best params: {'classifier__C': 0.1}
CV accuracy: 0.6560
Test accuracy: 0.6458

--- Random State 2 ---
Best params: {'classifier__C': 0.01}
CV accuracy: 0.6595
Test accuracy: 0.6489

--- Random State 3 ---
Best params: {'classifier__C': 0.01}
CV accuracy: 0.6556
Test accuracy: 0.6489

Average Test Accuracy for LogisticRegression: 0.6478
Saved best LogisticRegression model → best_model_LogisticRegression.pkl

----------------------------------------
Training RandomForest
----------------------------------------

--- Random State 1 ---
Best params: {'classifier__max_depth': 10, 'classifier__min_samples_split': 5, 'classifier__n_estimators':
300}
CV accuracy: 0.6512
Test accuracy: 0.6473

--- Random State 2 ---
Best params: {'classifier__max_depth': 10, 'classifier__min_samples_split': 10, 'classifier__n_estimators':
300}
CV accuracy: 0.6508
Test accuracy: 0.6450

--- Random State 3 ---
Best params: {'classifier__max_depth': 20, 'classifier__min_samples_split': 10, 'classifier__n_estimators':
300}
CV accuracy: 0.6455
```

```
Test accuracy: 0.6405

Average Test Accuracy for RandomForest: 0.6443
Saved best RandomForest model → best_model_RandomForest.pkl

----------------------------------------
Training SVM
----------------------------------------

--- Random State 1 ---
Best params: {'classifier__C': 0.1, 'classifier__kernel': 'linear'}
CV accuracy: 0.6564
Test accuracy: 0.6557

--- Random State 2 ---
Best params: {'classifier__C': 0.01, 'classifier__kernel': 'linear'}
CV accuracy: 0.6564
Test accuracy: 0.6534

--- Random State 3 ---
Best params: {'classifier__C': 0.1, 'classifier__kernel': 'linear'}
CV accuracy: 0.6539
Test accuracy: 0.6557

Average Test Accuracy for SVM: 0.6550
Saved best SVM model → best_model_SVM.pkl

----------------------------------------
Training XGBoost
----------------------------------------

--- Random State 1 ---
Best params: {'classifier__learning_rate': 0.01, 'classifier__max_depth': 3, 'classifier__n_estimators': 30
0}
CV accuracy: 0.6562
Test accuracy: 0.6534

--- Random State 2 ---
Best params: {'classifier__learning_rate': 0.01, 'classifier__max_depth': 1, 'classifier__n_estimators': 30
0}
CV accuracy: 0.6548
Test accuracy: 0.6618
```

```
--- Random State 3 ---
Best params: {'classifier__learning_rate': 0.01, 'classifier__max_depth': 1, 'classifier__n_estimators': 10
0}
CV accuracy: 0.6543
Test accuracy: 0.6618

Average Test Accuracy for XGBoost: 0.6590
Saved best XGBoost model → best_model_XGBoost.pkl


----------------------------------------
       FINAL RESULTS TABLE
----------------------------------------

            Algorithm                                Best params (seed 1)  \
0  LogisticRegression                              {'classifier__C': 0.1}
1        RandomForest  {'classifier__max_depth': 10, 'classifier__min...
2                 SVM  {'classifier__C': 0.1, 'classifier__kernel': '...
3             XGBoost  {'classifier__learning_rate': 0.01, 'classifie...

   Mean test accuracy  Std test accuracy  Best test accuracy (across seeds)
0              0.6478             0.0014                             0.6489
1              0.6443             0.0029                             0.6473
2              0.6550             0.0011                             0.6557
3              0.6590             0.0040                             0.6618
```

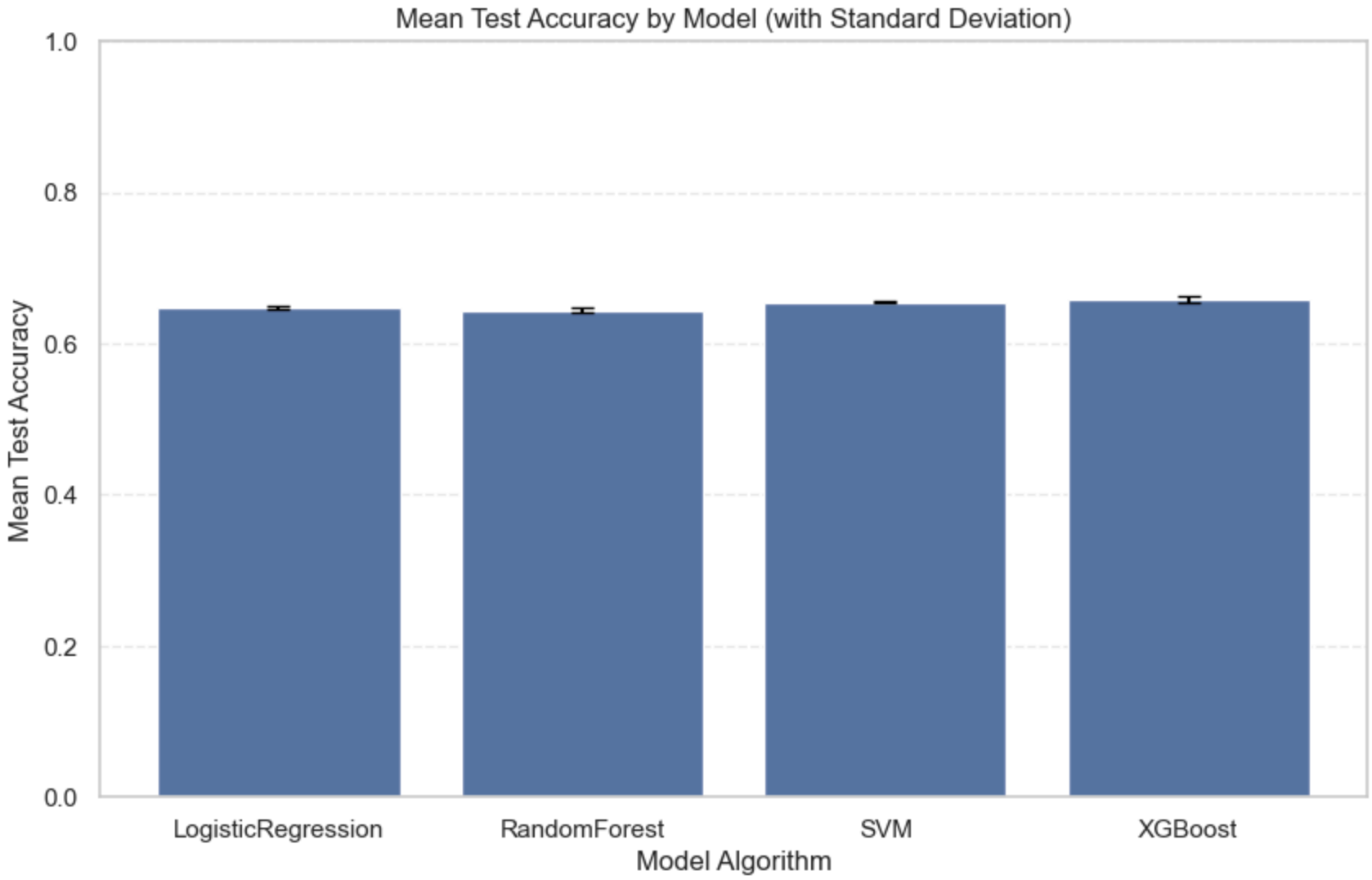
In [294…

```python
# ----------------------------------------------------------------------
# Plot: Mean Test Accuracy by Model Algorithm
# ----------------------------------------------------------------------
import seaborn as sns
import matplotlib.pyplot as plt

print(results_table.columns)
print(results_table.head())
plt.figure(figsize=(10,6))
sns.barplot(
    data=results_table,
    x="Algorithm",              # check corrected column name - check
    y="Mean test accuracy",     # check corrected column name - check
    ci=None
)
```

```
# Add error bars manually using standard deviation
for i, row in results_table.iterrows():
    plt.errorbar(
        i,
        row["Mean test accuracy"],  # check corrected column name - check
        yerr=row["Std test accuracy"],  # check corrected column name - check
        fmt="none",
        capsize=5,
        color="black"
    )

plt.title("Mean Test Accuracy by Model (with Standard Deviation)")
plt.xlabel("Model Algorithm")
plt.ylabel("Mean Test Accuracy")
plt.ylim(0, 1)
plt.grid(axis='y', linestyle='--', alpha=0.4)
fig_name = "test_acc_mean"
fig_path = os.path.join(output_folder, f"{fig_name}.png")
plt.savefig(fig_path, bbox_inches="tight", dpi=300)
plt.show()
plt.close()
```

```
Index(['Algorithm', 'Best params (seed 1)', 'Mean test accuracy',
       'Std test accuracy', 'Best test accuracy (across seeds)'],
      dtype='object')
```

| | Algorithm | Best params (seed 1) | Mean test accuracy | Std test accuracy | Best test accuracy (across seeds) |
|---|---|---|---|---|---|
| 0 | LogisticRegression | {'classifier__C': 0.1} | 0.6478 | 0.0014 | 0.6489 |
| 1 | RandomForest | {'classifier__max_depth': 10, 'classifier__min... | 0.6443 | 0.0029 | 0.6473 |
| 2 | SVM | {'classifier__C': 0.1, 'classifier__kernel': '... | 0.6550 | 0.0011 | 0.6557 |
| 3 | XGBoost | {'classifier__learning_rate': 0.01, 'classifie... | 0.6590 | 0.0040 | 0.6618 |




In [295…

```
# ----------------------------------------
#   metrics
# ----------------------------------------

import numpy as np
import shap
from sklearn.metrics import accuracy_score, fbeta_score, f1_score
from sklearn.inspection import permutation_importance
```

```
print("\n\n----------------------------------------")
print("     BASELINES + CHECK SHAP/PERMUTATION INPUTS")
print("----------------------------------------n")

# ------------------------------------------------------
# FINAL FITTED PREPROCESSOR of the BEST MODEL
# ------------------------------------------------------
best_global_model_name = max(results_summary, key=lambda m: results_summary[m]["avg_test_score"])
best_pipeline = results_summary[best_global_model_name]["models"][0]
fitted_preprocessor = best_pipeline.named_steps["preprocessor"]

# Feature names from trained preprocessor
try:
    feature_names = fitted_preprocessor.get_feature_names_out()
except:
    feature_names = X_train.columns.tolist()

# Preprocessed matrices (same one SHAP will use)
X_train_trans = fitted_preprocessor.transform(X_train)
X_test_trans  = fitted_preprocessor.transform(X_test)

# Convert sparse → dense only if necessary as per error messages
X_train_dense = X_train_trans.toarray() if hasattr(X_train_trans, "toarray") else np.asarray(X_train_trans
X_test_dense  = X_test_trans.toarray()  if hasattr(X_test_trans,  "toarray") else np.asarray(X_test_trans)

print("Shapes after preprocessing:")
print("  X_train_trans:", X_train_trans.shape)
print("  X_test_trans :", X_test_trans.shape)


# ------------------------------------------------------
# baseline metrics
# ------------------------------------------------------
(unique, counts) = np.unique(y_train, return_counts=True)
majority_label = unique[np.argmax(counts)]
baseline_preds = np.full_like(y_test, fill_value=majority_label)

baseline_acc  = accuracy_score(y_test, baseline_preds)
baseline_f05 = fbeta_score(y_test, baseline_preds, beta=0.5, average='macro', zero_division=0)
baseline_f1  = f1_score(y_test, baseline_preds, average='macro', zero_division=0)
baseline_f2  = fbeta_score(y_test, baseline_preds, beta=2, average='macro', zero_division=0)
```

```
print("\n=== BASELINE (majority class) ===")
print(f"Majority label: {majority_label}")
print(f"Accuracy: {baseline_acc:.4f}")
print(f"F0.5 (macro): {baseline_f05:.4f}")
print(f"F1   (macro): {baseline_f1:.4f}")
print(f"F2   (macro): {baseline_f2:.4f}")


# Non-1D columns would break SHAP/permutation importance
bad_cols = [
    c for c in X_train.columns
    if isinstance(X_train[c].iloc[0], (list, dict, tuple))
]

if bad_cols:
    print("⚠️ Removing non-1D columns:", bad_cols)
    X_train = X_train.drop(columns=bad_cols, errors="ignore")
    X_test  = X_test.drop(columns=bad_cols, errors="ignore")


# ------------------------------------------------------
# shiyu note from meeting: check shape
# ------------------------------------------------------
for model_name, info in results_summary.items():
    model_check = info["models"][0]
    try:
        preds_check = model_check.predict(X_test)
        print(f"{model_name} prediction length: {len(preds_check)}  | y_test: {len(y_test)}")
        if len(preds_check) != len(y_test):
            print("❌ MISMATCH! Predictions don't align with y_test.")
        else:
            print("✅ OK — predictions match test labels.")
    except Exception as e:
        print(f"{model_name} prediction failed: {e}")

print("\n------- Completed SHAP/Permutation Prep Block -------\n")
```

```
----------------------------------------
     BASELINES + CHECK SHAP/PERMUTATION INPUTS
----------------------------------------n
Shapes after preprocessing:
  X_train_trans: (5218, 60)
  X_test_trans : (1310, 60)

=== BASELINE (majority class) ===
Majority label: 1
Accuracy: 0.5947
F0.5 (macro): 0.3236
F1   (macro): 0.3729
F2   (macro): 0.4400
LogisticRegression prediction length: 1310  | y_test: 1310
✅ OK — predictions match test labels.
RandomForest prediction length: 1310  | y_test: 1310
✅ OK — predictions match test labels.
SVM prediction length: 1310  | y_test: 1310
✅ OK — predictions match test labels.
XGBoost prediction length: 1310  | y_test: 1310
✅ OK — predictions match test labels.

------- Completed SHAP/Permutation Prep Block -------
```

In [296…

```python
# ------------------------------------------------------------------
# Compare models to baseline; compute permutation importance + impurity + SHAP
# ------------------------------------------------------------------
summary_rows = []
global_importances_all = []

for model_name, info in results_summary.items():
    print(f"\n\n--------- ANALYZING {model_name} ---------")

    # Best model (first index in stored list)
    pipeline_best = info["models"][0]
    classifier = pipeline_best.named_steps["classifier"]

    # ------------------------- Metrics -------------------------
    y_test_pred = pipeline_best.predict(X_test)
```

```
    acc = accuracy_score(y_test, y_test_pred)
    f05 = fbeta_score(y_test, y_test_pred, beta=0.5, average="macro", zero_division=0)
    f1  = f1_score(y_test, y_test_pred, average="macro", zero_division=0)
    f2  = fbeta_score(y_test, y_test_pred, beta=2, average="macro", zero_division=0)

    mean_test = info.get("avg_test_score", np.mean(info["test_scores"]))
    std_test  = np.std(info["test_scores"]) if "test_scores" in info else np.nan

    z_score = (mean_test - baseline_acc) / std_test if std_test and std_test > 0 else np.nan

    print(f"Mean CV/Test (avg across states): {mean_test:.4f} (std: {std_test:.4f})")
    print(f"Test accuracy: {acc:.4f}")
    print(f"F0.5/F1/F2: {f05:.4f} / {f1:.4f} / {f2:.4f}")
    print(f"Z-score above baseline: {z_score if not np.isnan(z_score) else 'N/A'}")

    # ------------------------- Impurity Importance --------------------------
    impurity_df = None
    if hasattr(classifier, "feature_importances_"):
        try:
            impurity_df = (
                pd.DataFrame({
                    "feature": feature_names,
                    "impurity_importance": classifier.feature_importances_
                }).sort_values("impurity_importance", ascending=False)
            )
            print("Impurity importance computed.")
        except Exception as e:
            print("Impurity importance failed:", e)

    # ------------------------- Permutation Importance -------------------------
    perm_df = None
    try:
        classifier = pipeline_best.named_steps["classifier"]
        preprocessor = pipeline_best.named_steps["preprocessor"]

        X_test_trans = preprocessor.transform(X_test)

        #parameters for times sake
        perm = permutation_importance(
            classifier,
            X_test_trans,
            y_test,
```

```
            n_repeats=20,
            random_state=42,
            n_jobs=-1,
            scoring="accuracy"
        )

        feature_names = preprocessor.get_feature_names_out()

        perm_df = pd.DataFrame({
            "feature": feature_names,
            "perm_mean": perm.importances_mean,
            "perm_std": perm.importances_std
        }).sort_values("perm_mean", ascending=False)

        print("Permutation importance computed.")
        print("\nTop 10 permutation importances:")
        print(perm_df.head(10))
    except Exception as e:
        print("Permutation importance failed:", e)


    # ------------------------- Transform Train/Test Once -------------------------
    pre = pipeline_best.named_steps["preprocessor"]

    X_train_trans = pre.transform(X_train)
    X_test_trans  = pre.transform(X_test)

    # convert to dense
    X_train_dense = X_train_trans.toarray() if hasattr(X_train_trans, "toarray") else X_train_trans
    X_test_dense  = X_test_trans.toarray()  if hasattr(X_test_trans, "toarray") else X_test_trans

    # ------------------------- SHAP Global Importance -------------------------
    shap_df = None
    explainer = None

    try:
        X_shap_background = X_train_dense[:200]

        explainer = shap.Explainer(
            classifier,
            X_shap_background,
            feature_names=feature_names
```

```
        )

        shap_values = explainer(X_train_dense[:500])
        vals = shap_values.values

        # multiclass = list of arrays → average abs val over classes
        if isinstance(vals, list) or (hasattr(vals, "dtype") and vals.dtype == object):
            arrs = [np.abs(v).mean(axis=0) for v in vals]
            mean_abs_shap = np.mean(arrs, axis=0)
        else:
            mean_abs_shap = np.abs(vals).mean(axis=0)

        shap_df = pd.DataFrame({
            "feature": feature_names,
            "shap_mean_abs": mean_abs_shap
        }).sort_values("shap_mean_abs", ascending=False)

        print("SHAP global importance computed.")
    except Exception as e:
        print("SHAP global importances failed:", e)

    # ------------------------- Combine importances -------------------------
    imp_combined = pd.DataFrame({"feature": feature_names})

    if impurity_df is not None:
        imp_combined = imp_combined.merge(impurity_df, on="feature", how="left")

    if perm_df is not None:
        imp_combined = imp_combined.merge(perm_df[["feature", "perm_mean"]], on="feature", how="left")

    if shap_df is not None:
        imp_combined = imp_combined.merge(shap_df, on="feature", how="left")

    # Normalize
    for col in ["impurity_importance", "perm_mean", "shap_mean_abs"]:
        if col in imp_combined.columns:
            col_vals = imp_combined[col].fillna(0).values
            imp_combined[col + "_norm"] = (
                col_vals / col_vals.max() if col_vals.max() > 0 else 0
            )

    norm_cols = [c for c in imp_combined.columns if c.endswith("_norm")]
```

```
    if norm_cols:
        imp_combined["aggregate_score"] = imp_combined[norm_cols].mean(axis=1)
        imp_combined = imp_combined.sort_values("aggregate_score", ascending=False)

    global_importances_all.append((model_name, imp_combined))

    # ------------------------- Local SHAP ---------------------------
    local_shap_out = None
    if explainer is not None and shap_values is not None:
        try:
            n_local = min(3, X_test_dense.shape[0])
            local_vals = explainer(X_test_dense[:n_local])
            local_shap_out = {
                "instance_indices": list(range(n_local)),
                "local_values": local_vals.values,
                "expected_value": local_vals.base_values
            }
            print(f"Computed local SHAP for {n_local} samples.")

            # Print top 5 features per instance
            for i in range(n_local):
                instance_df = pd.DataFrame({
                    "feature": feature_names,
                    "shap_value": local_vals.values[i]
                }).sort_values("shap_value", key=abs, ascending=False)
                print(f"\nInstance {i} top 5 SHAP features:")
                print(instance_df.head(5))
        except Exception as e:
            print("Local SHAP failed:", e)

    # ------------------------- Collect summary row ---------------------------
    summary_rows.append({
        "Algorithms": model_name,
        "Best parameters (example)": info["best_params"][0] if info.get("best_params") else None,
        "Mean test score": round(mean_test, 4),
        "Std test score": round(std_test, 4) if not np.isnan(std_test) else None,
        "Test accuracy (best model)": round(acc, 4),
        "Test F0.5": round(f05, 4),
        "Test F1": round(f1, 4),
        "Test F2": round(f2, 4),
        "Z_score_above_baseline": round(z_score, 4) if not np.isnan(z_score) else None,
        "Top feature (aggregate)": imp_combined["feature"].iloc[0] if not imp_combined.empty else None,
```

```
        "Least feature (aggregate)": imp_combined["feature"].iloc[-1] if not imp_combined.empty else None,
        "local_shap_example": local_shap_out
    })
```

```
--------- ANALYZING LogisticRegression ---------
Mean CV/Test (avg across states): 0.6478 (std: 0.0014)
Test accuracy: 0.6458
F0.5/F1/F2: 0.6282 / 0.6265 / 0.6254
Z-score above baseline: 36.9463293169982
Permutation importance computed.

Top 10 permutation importances:
                                      feature  perm_mean  perm_std
17                              num__odds_diff   0.092137  0.010744
32                         cat__BetterRank_Red   0.002481  0.001944
50         cat__StanceCombo_Orthodox vs Switch   0.002214  0.000634
11                                num__RWinPct   0.001183  0.001610
5                     num__RedCurrentWinStreak   0.001031  0.001965
34          cat__WeightClassLabel_Bantamweight   0.001031  0.001624
14                           num__RedIsGrappler   0.000802  0.002334
21                          cat__TitleBout_True   0.000725  0.000743
10                                  num__AgeDif   0.000649  0.000437
43  cat__WeightClassLabel_Women's Bantamweight   0.000611  0.000857
SHAP global importance computed.
Computed local SHAP for 3 samples.

Instance 0 top 5 SHAP features:
                       feature  shap_value
17             num__odds_diff    0.735632
2   num__BlueCurrentWinStreak   -0.140735
8                num__BlueAge    0.139484
32        cat__BetterRank_Red    0.106626
7           num__RedWeightLbs    0.080405

Instance 1 top 5 SHAP features:
                     feature  shap_value
17           num__odds_diff    1.941565
19              num__Spread    0.224957
8               num__BlueAge    0.139484
29  cat__RedStance_Southpaw    0.120034
32      cat__BetterRank_Red    0.106626

Instance 2 top 5 SHAP features:
                      feature  shap_value
17            num__odds_diff   -1.352854
```

```
8               num__BlueAge   -0.238962
9              num__ReachDif   -0.121905
0           num__WeightClass   -0.105519
25  cat__BlueStance_Southpaw    0.097518


--------- ANALYZING RandomForest ---------
Mean CV/Test (avg across states): 0.6443 (std: 0.0029)
Test accuracy: 0.6473
F0.5/F1/F2: 0.6243 / 0.6189 / 0.6171
Z-score above baseline: 17.37198072430765
Impurity importance computed.
Permutation importance computed.

Top 10 permutation importances:
                                      feature  perm_mean  perm_std
17                               num__odds_diff   0.081870  0.009784
46   cat__WeightClassLabel_Women's Strawweight   0.002634  0.001698
16                           num__BlueIsStriker   0.002252  0.001765
26                        cat__BlueStance_Switch   0.001870  0.000511
24                     cat__BlueStance_Orthodox   0.001489  0.001038
30                        cat__RedStance_Switch   0.001489  0.000818
50         cat__StanceCombo_Orthodox vs Switch   0.001412  0.000651
34          cat__WeightClassLabel_Bantamweight   0.001260  0.000735
48       cat__StanceCombo_Orthodox vs Orthodox   0.001107  0.001265
43  cat__WeightClassLabel_Women's Bantamweight   0.001069  0.000611
 99%|===================| 988/1000 [00:29<00:00]
```

```
SHAP global importances failed: Per-column arrays must each be 1-dimensional
Computed local SHAP for 3 samples.
Local SHAP failed: Per-column arrays must each be 1-dimensional


--------- ANALYZING SVM ---------
Mean CV/Test (avg across states): 0.6550 (std: 0.0011)
Test accuracy: 0.6557
F0.5/F1/F2: 0.6328 / 0.6264 / 0.6243
Z-score above baseline: 55.861435713737386
Permutation importance computed.

Top 10 permutation importances:
                             feature  perm_mean  perm_std
17                    num__odds_diff   0.147748  0.010390
19                       num__Spread   0.015458  0.006694
40  cat__WeightClassLabel_Lightweight   0.002634  0.000614
10                       num__AgeDif   0.001832  0.000978
18                       num__AgeGap   0.001756  0.000545
16                 num__BlueIsStriker   0.001603  0.000867
5            num__RedCurrentWinStreak   0.001336  0.000585
0                    num__WeightClass   0.001221  0.001217
9                       num__ReachDif   0.001107  0.000918
14                 num__RedIsGrappler   0.000916  0.000890
SHAP global importance computed.
Computed local SHAP for 3 samples.

Instance 0 top 5 SHAP features:
                                  feature  shap_value
17                          num__odds_diff    1.093114
19                             num__Spread   -0.247060
7                         num__RedWeightLbs    0.037935
48    cat__StanceCombo_Orthodox vs Orthodox    0.030171
39  cat__WeightClassLabel_Light Heavyweight   -0.029046

Instance 1 top 5 SHAP features:
                     feature  shap_value
17            num__odds_diff    2.885071
19               num__Spread   -1.909985
29   cat__RedStance_Southpaw    0.076576
24  cat__BlueStance_Orthodox    0.042962
28   cat__RedStance_Orthodox   -0.038900
```

```
Instance 2 top 5 SHAP features:
                                  feature  shap_value
17                         num__odds_diff   -2.010276
19                            num__Spread   -0.600084
24                cat__BlueStance_Orthodox    0.042962
48  cat__StanceCombo_Orthodox vs Orthodox   -0.038399
8                            num__BlueAge   -0.033158


--------- ANALYZING XGBoost ---------
Mean CV/Test (avg across states): 0.6590 (std: 0.0040)
Test accuracy: 0.6534
F0.5/F1/F2: 0.6318 / 0.6268 / 0.6249
Z-score above baseline: 16.263455967290525
Impurity importance computed.
Permutation importance computed.

Top 10 permutation importances:
                                feature  perm_mean  perm_std
17                        num__odds_diff   0.119885  0.010302
8                           num__BlueAge   0.001641  0.002969
9                           num__ReachDif   0.001603  0.001205
0                       num__WeightClass   0.001489  0.000886
29                cat__RedStance_Southpaw   0.000802  0.000614
16                    num__BlueIsStriker   0.000534  0.000350
12                  num__RedTotalFights   0.000496  0.000774
11                           num__RWinPct   0.000382  0.002027
2              num__BlueCurrentWinStreak   0.000267  0.001493
57  cat__StanceCombo_Switch vs Orthodox   0.000153  0.000305
SHAP global importance computed.
Computed local SHAP for 3 samples.

Instance 0 top 5 SHAP features:
                 feature  shap_value
17       num__odds_diff    0.854119
19          num__Spread    0.294339
12  num__RedTotalFights   -0.035756
8           num__BlueAge    0.025057
10           num__AgeDif   -0.022076

Instance 1 top 5 SHAP features:
```

```
                    feature  shap_value
17            num__odds_diff    1.143060
19               num__Spread    0.560455
15       num__BlueIsGrappler    0.029522
24  cat__BlueStance_Orthodox   -0.029412
12       num__RedTotalFights   -0.029180

Instance 2 top 5 SHAP features:
                    feature  shap_value
17            num__odds_diff   -1.061218
19               num__Spread   -0.339018
5   num__RedCurrentWinStreak    0.083631
11              num__RWinPct    0.056465
8               num__BlueAge   -0.041240
```

In [297…

```python
# ---------------------------
# Summary DataFrame
# ---------------------------
model_perf_df = pd.DataFrame(summary_rows)
print("\n\n=== Model performance summary vs baseline ===")
display(model_perf_df)

# ---------------------------
# Barplot of mean test score with error bars
# ---------------------------
plt.figure(figsize=(8,5))

# Create barplot
sns.barplot(
    data=model_perf_df,
    x='Algorithms',
    y='Mean test score',
    ci=None,
    palette='Set2'
)

# Add error bars
y_err = model_perf_df['Std test score'].fillna(0).values
plt.errorbar(
    x=np.arange(len(model_perf_df)),
    y=model_perf_df['Mean test score'].values,
    yerr=y_err,
```

```
    fmt='none',
    capsize=5,
    color='black'
)

# Baseline accuracy line
plt.axhline(baseline_acc, color='red', linestyle='--', label=f'Baseline acc={baseline_acc:.3f}')

# Labels and styling
plt.title("Model Mean Test Score (with SD) vs Baseline")
plt.ylabel("Mean Test Accuracy")
plt.xlabel("Model Algorithm")
plt.ylim(0, 1)
plt.grid(axis='y', linestyle='--', alpha=0.4)
plt.legend()
plt.tight_layout()
fig_name = "mean_test_acc_model"
fig_path = os.path.join(output_folder, f"{fig_name}.png")
plt.savefig(fig_path, bbox_inches="tight", dpi=300)
plt.show()
plt.close()
```

```
=== Model performance summary vs baseline ===
```

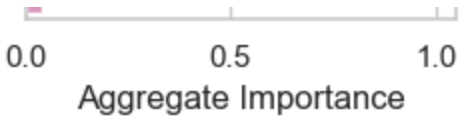
| | Algorithms | Best parameters (example) | Mean test score | Std test score | Test accuracy (best model) | Test F0.5 | Test F1 | Test F2 | Z_score_above_baseline |
|---|---|---|---|---|---|---|---|---|---|
| **0** | LogisticRegression | {'classifier__C': 0.1} | 0.6478 | 0.0014 | 0.6458 | 0.6282 | 0.6265 | 0.6254 | 36.9463 |
| **1** | RandomForest | {'classifier__max_depth': 10, 'classifier__min... | 0.6443 | 0.0029 | 0.6473 | 0.6243 | 0.6189 | 0.6171 | 17.3720 |
| **2** | SVM | {'classifier__C': 0.1, 'classifier__kernel': '... | 0.6550 | 0.0011 | 0.6557 | 0.6328 | 0.6264 | 0.6243 | 55.8614 |
| **3** | XGBoost | {'classifier__learning_rate': 0.01, 'classifie... | 0.6590 | 0.0040 | 0.6534 | 0.6318 | 0.6268 | 0.6249 | 16.2635 |


In [298…

```
# ------------------------------------------------------------------------
# Top features across models (aggregate)
# ------------------------------------------------------------------------
import os
top_feats_list = []

for model_name, imp_df in global_importances_all:
    # Use aggregate_score if available, otherwise fallback to SHAP
    score_col = 'aggregate_score' if 'aggregate_score' in imp_df.columns else 'shap_mean_abs'

    if score_col in imp_df.columns:
        top_feats = (
```

```
            imp_df.nlargest(10, score_col)[['feature', score_col]]
            .rename(columns={score_col: 'aggregate_score'})
            .assign(model=model_name)
        )
        top_feats_list.append(top_feats)

if top_feats_list:
    top_feats_all = pd.concat(top_feats_list, ignore_index=True)

    plt.figure(figsize=(8,8))
    sns.barplot(
        data=top_feats_all,
        x='aggregate_score',
        y='feature',
        hue='model',
        palette='Set2'
    )
    plt.title("Top Features per Model (Top 10 by Aggregate Score)")
    plt.xlabel("Aggregate Importance")
    plt.ylabel("Feature")
    plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
    plt.tight_layout()
    fig_name = "top_features_per_model"
    fig_path = os.path.join(output_folder, f"{fig_name}.png")
    plt.savefig(fig_path, bbox_inches="tight", dpi=300)
    plt.show()
    plt.close()
else:
    print("No feature importance data available to plot.")
```


Top Features per Model (Top 10 by Aggregate Score)
Feature
num__odds_diff
num__RedIsGrappler
cat__BetterRank_Red
cat__StanceCombo_Orthodox vs Orthodox
num__RedCurrentLoseStreak
num__BlueAge
num__BlueIsGrappler
num__BlueDraws
num__ReachDif
num__BlueCurrentWinStreak
num__Spread
num__RedTotalFights
num__RWinPct
num__AgeGap
num__BlueTotalFights
num__AgeDif
cat__RedStance_Orthodox
cat__WeightClassLabel_Lightweight
cat__BlueStance_Orthodox
num__BlueCurrentLoseStreak
LogisticRegression
RandomForest
SVM
XGBoost




In [299…

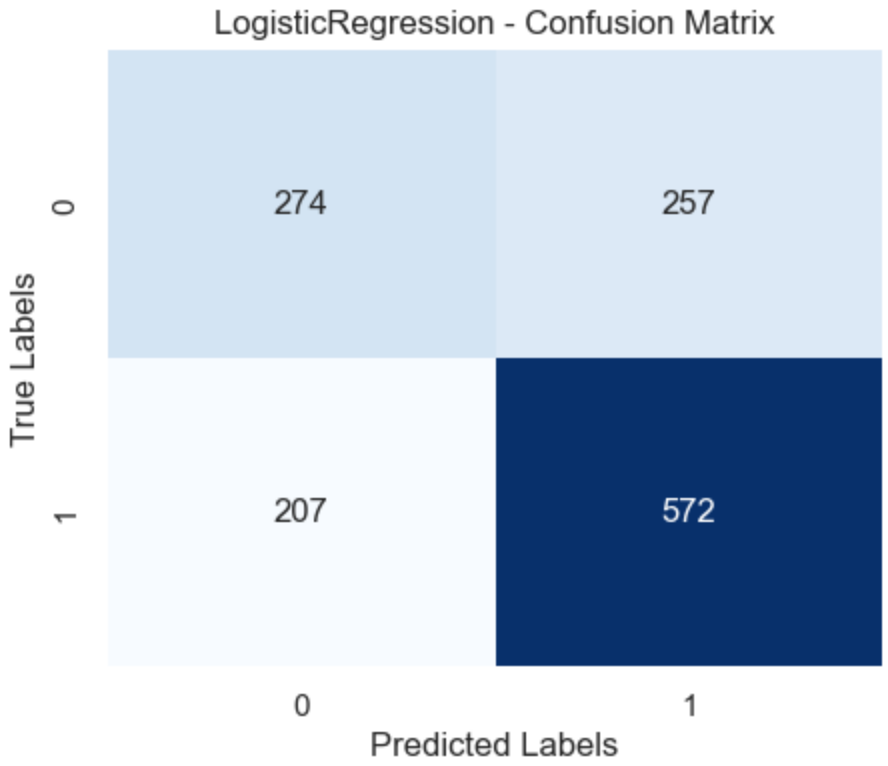
```python
# ----------------------------------------------------------------------
# Ensure dense inputs for SHAP/tree models - may be redundant but i'm paranoid
# ----------------------------------------------------------------------
X_train_dense = X_train_trans.toarray() if hasattr(X_train_trans, "toarray") else np.asarray(X_train_trans
X_test_dense  = X_test_trans.toarray()  if hasattr(X_test_trans, "toarray")  else np.asarray(X_test_trans)


# ----------------------------------------------------------------------
# SHAP fx
# ----------------------------------------------------------------------
def run_shap(pipeline_model, classifier, model_name, feature_names, X_sample=None):
    if X_sample is None:
        X_sample = X_test_dense[:200]  # subset for speed

    try:
        if model_name in ["RandomForest", "XGBoost"]:
            explainer = shap.TreeExplainer(classifier)
            shap_values = explainer.shap_values(X_sample)
        elif model_name == "LogisticRegression":
            masker = shap.maskers.Independent(X_sample)
            explainer = shap.KernelExplainer(classifier.predict_proba, masker)
            shap_values = explainer.shap_values(X_sample, nsamples=200)
        elif model_name == "SVM":
            print("Skipping SHAP for SVM - KernelExplainer too slow.")
            return
        else:
            print(f"SHAP not supported for {model_name}.")
            return

        shap.summary_plot(shap_values, X_sample, feature_names=feature_names, show=True)
        plt.title(f"SHAP Summary Plot - {model_name}")
        fig_name = "shap_summary_plot_{model_name}"
        fig_path = os.path.join(output_folder, f"{fig_name}.png")
        plt.savefig(fig_path, bbox_inches="tight", dpi=300)
        plt.show()
        plt.close()
    except Exception as e:
```

```
        print(f"SHAP failed for {model_name}: {e}")



# ------------------------------------------------------------------
# Ftr importance for rubric
# ------------------------------------------------------------------
def extract_feature_importance(classifier, model_name, feature_names):
    if hasattr(classifier, "feature_importances_"):    # Tree-based
        importances = classifier.feature_importances_
    elif hasattr(classifier, "coef_"):                 # Linear models
        importances = np.abs(classifier.coef_).flatten()
    else:
        print(f"{model_name} does not support feature importances.")
        return None

    return pd.DataFrame({
        "Feature": feature_names,
        "Importance": importances,
        "Model": model_name
    })


# ------------------------------------------------------------------
# CONFUSION MATRIX + AUROC
# ------------------------------------------------------------------
def run_confusion(pipeline_model, model_name):
    preds = pipeline_model.predict(X_test)
    cm = confusion_matrix(y_test, preds)
    plt.figure(figsize=(5,4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
    plt.xlabel("Predicted Labels")
    plt.ylabel("True Labels")
    plt.title(f"{model_name} - Confusion Matrix")
    fig_name = "{model_name}_confusion_matrix"
    fig_path = os.path.join(output_folder, f"{fig_name}.png")
    plt.savefig(fig_path, bbox_inches="tight", dpi=300)
    plt.show()
    plt.close()


from sklearn.preprocessing import LabelBinarizer
```

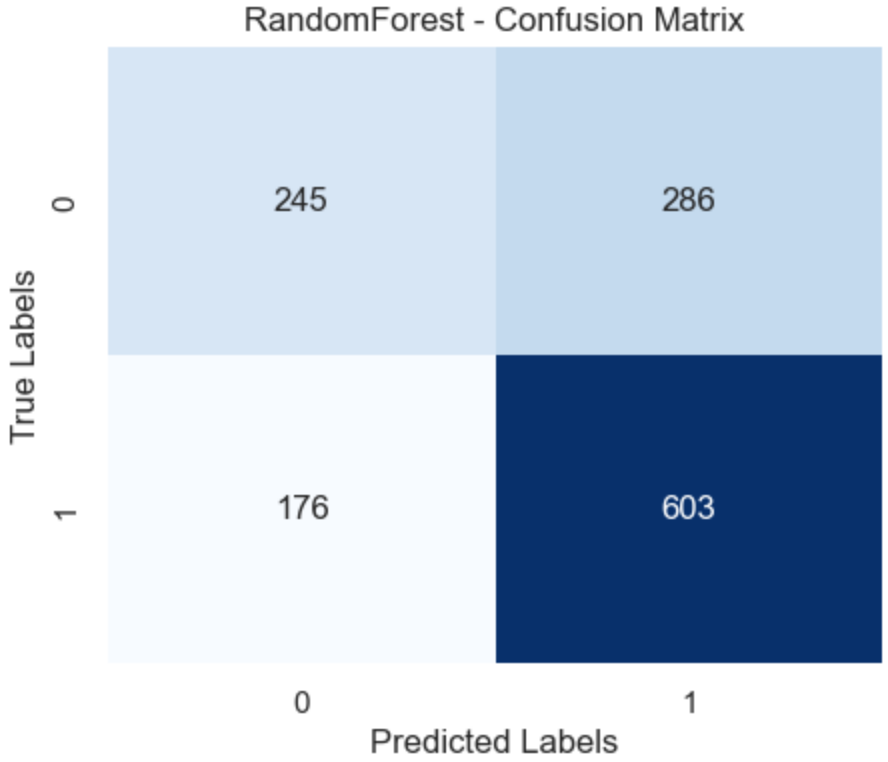
```
def run_auroc(pipeline_model, model_name):
    y_true = y_test.copy()  # 1D integer labels
    if y_true.ndim > 1:
        #y_test is one-hot, convert to single integer labels
        y_true = np.argmax(y_true, axis=1)

    try:
        if hasattr(pipeline_model, "predict_proba"):
            probs = pipeline_model.predict_proba(X_test)
            if probs.shape[1] > 2:
                # multiclass
                auc = roc_auc_score(y_true, probs, multi_class="ovr")
            else:
                auc = roc_auc_score(y_true, probs[:,1])
        else:
            probs = pipeline_model.decision_function(X_test)
            auc = roc_auc_score(y_true, probs)

        print(f"{model_name} AUROC: {auc:.4f}")
        print(classification_report(y_true, pipeline_model.predict(X_test)))
    except Exception as e:
        print(f"{model_name} AUROC failed: {e}")


# ---------------------------------------------------------------------
# RUN ANALYSIS FOR ALL MODELS - home stretch
# ---------------------------------------------------------------------
all_fi = []

for model_name, info in results_summary.items():
    print(f"\n\n----------------------------------------")
    print(f" FINAL ANALYSIS: {model_name}")
    print("----------------------------------------")

    pipeline_model = info["models"][0]  # best model
    classifier = pipeline_model.named_steps["classifier"]

    # Feature names from pipeline
    try:
        feature_names = pipeline_model.named_steps["preprocessor"].get_feature_names_out()
    except:
```

```
        feature_names = X_train.columns.tolist()

    # Confusion matrix & AUROC
    run_confusion(pipeline_model, model_name)
    run_auroc(pipeline_model, model_name)

    # Feature importance
    fi_df = extract_feature_importance(classifier, model_name, feature_names)
    if fi_df is not None:
        all_fi.append(fi_df)

    # SHAP summary
    run_shap(pipeline_model, classifier, model_name, feature_names)
```

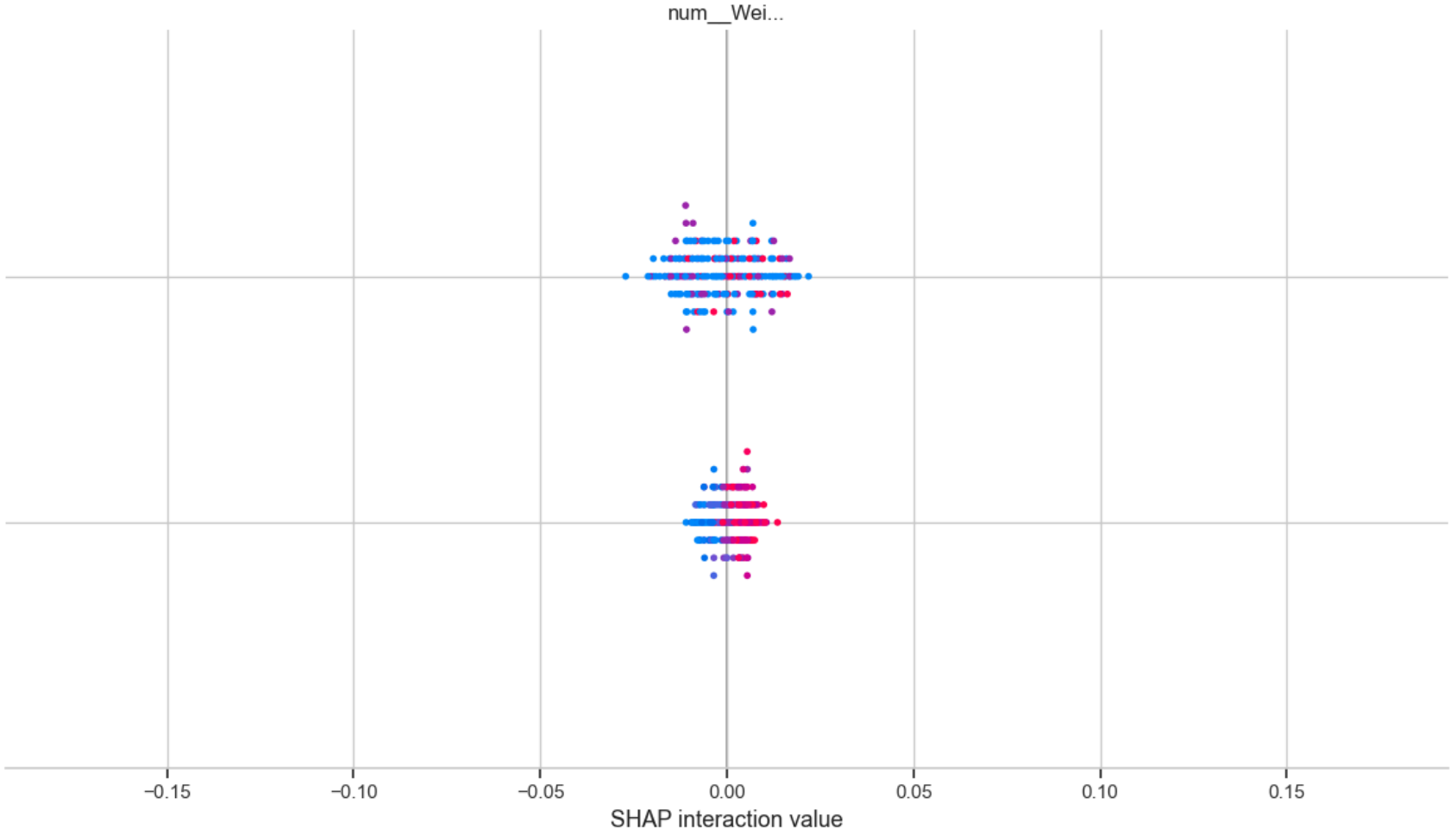
```
----------------------------------------
 FINAL ANALYSIS: LogisticRegression
----------------------------------------
```

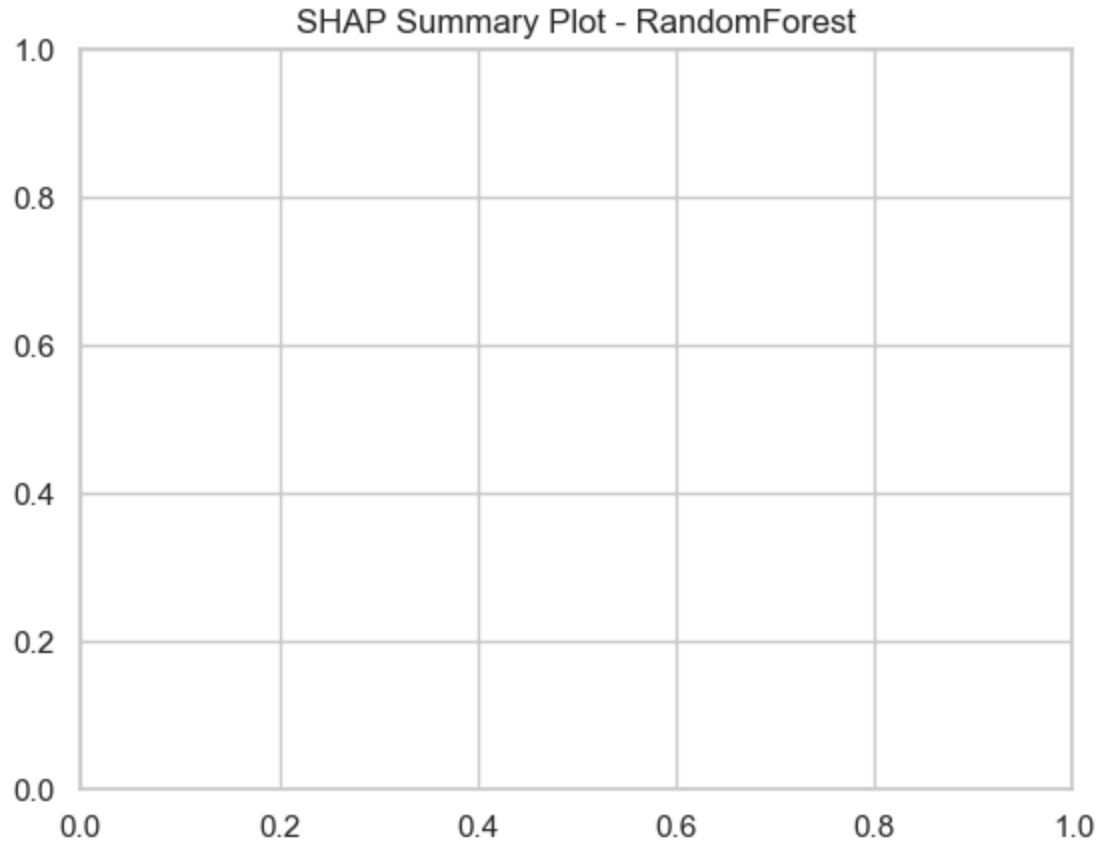


```
LogisticRegression AUROC: 0.6974
              precision    recall  f1-score   support

           0       0.57      0.52      0.54       531
           1       0.69      0.73      0.71       779

    accuracy                           0.65      1310
   macro avg       0.63      0.63      0.63      1310
weighted avg       0.64      0.65      0.64      1310

SHAP failed for LogisticRegression: Unknown type passed as data object: <class 'shap.maskers._tabular.Indep
endent'>


--------------------------------------
 FINAL ANALYSIS: RandomForest
--------------------------------------
```

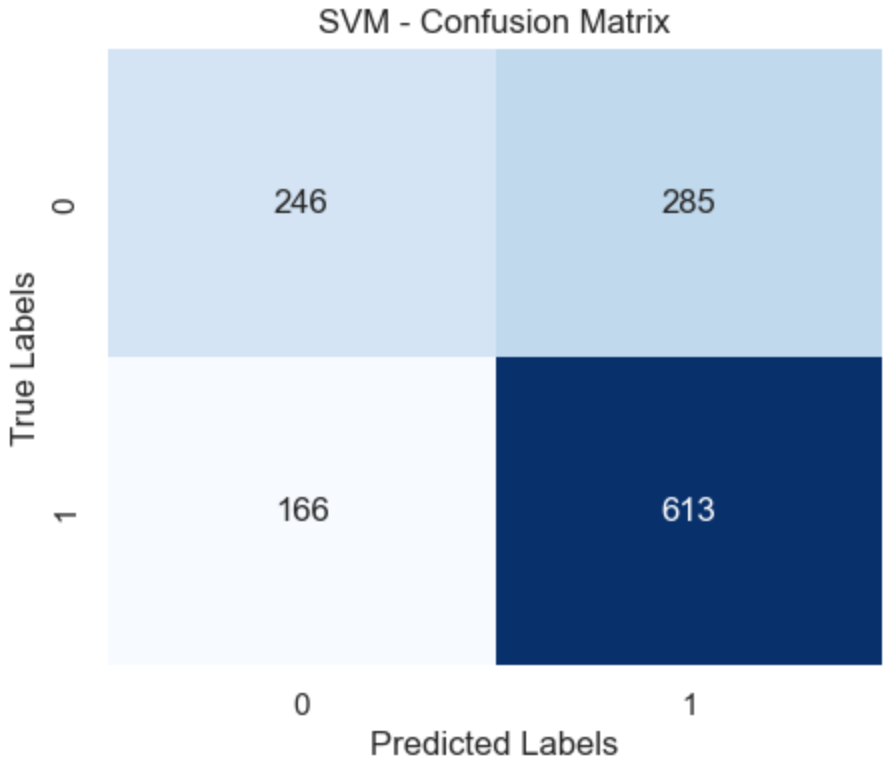


```
RandomForest AUROC: 0.6928
              precision    recall  f1-score   support

           0       0.58      0.46      0.51       531
           1       0.68      0.77      0.72       779

    accuracy                           0.65      1310
   macro avg       0.63      0.62      0.62      1310
weighted avg       0.64      0.65      0.64      1310

<Figure size 640x480 with 0 Axes>
```

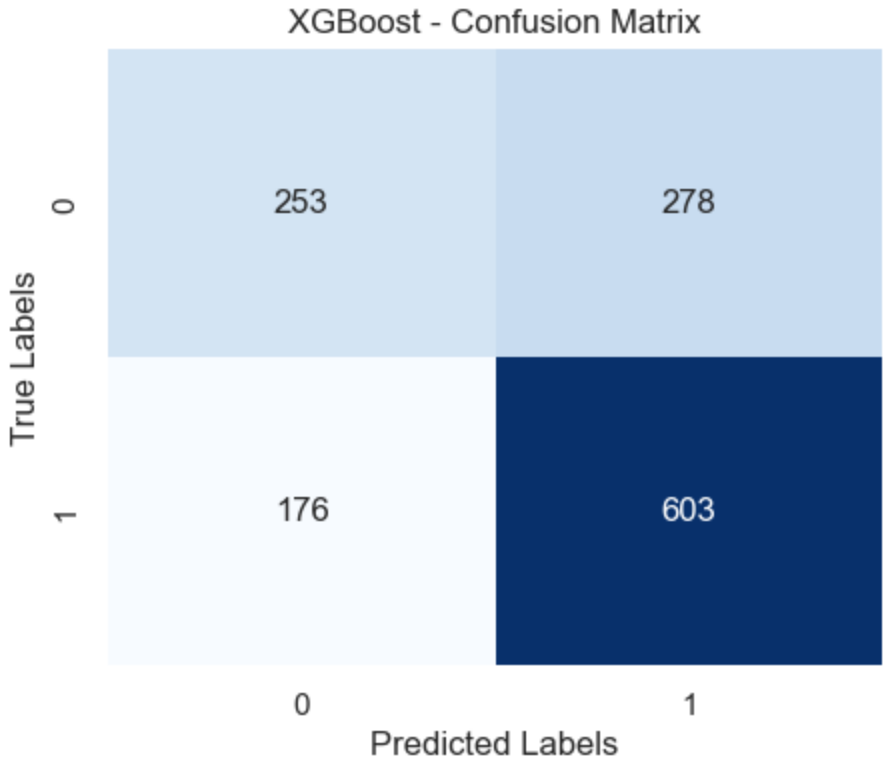




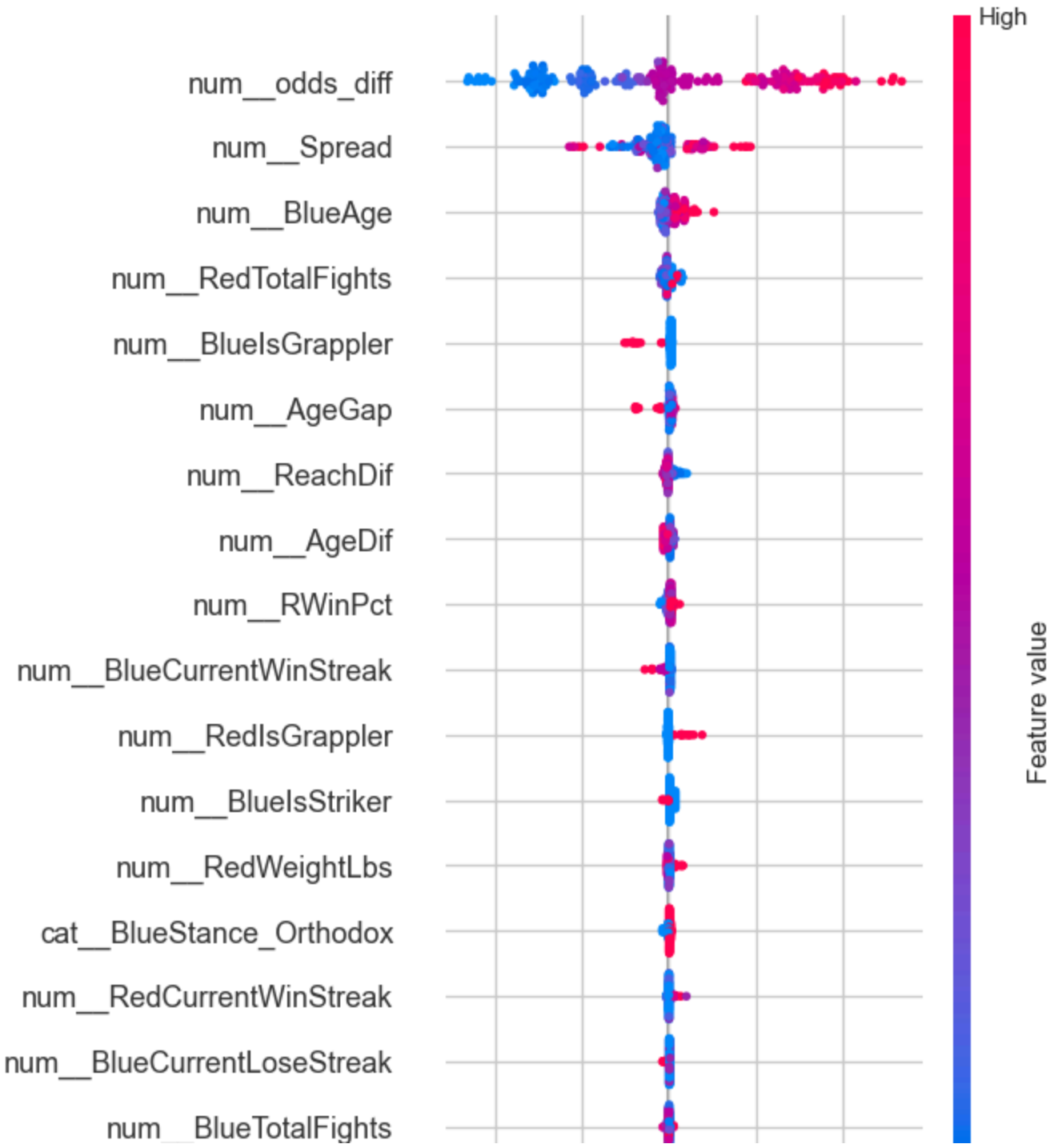

```
----------------------------------------
 FINAL ANALYSIS: SVM
----------------------------------------
```

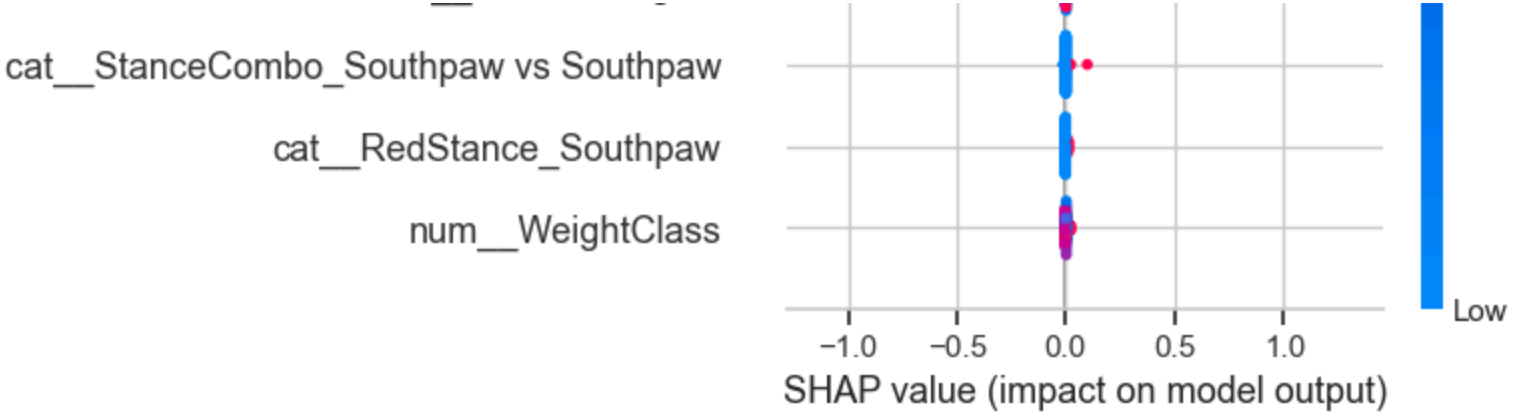


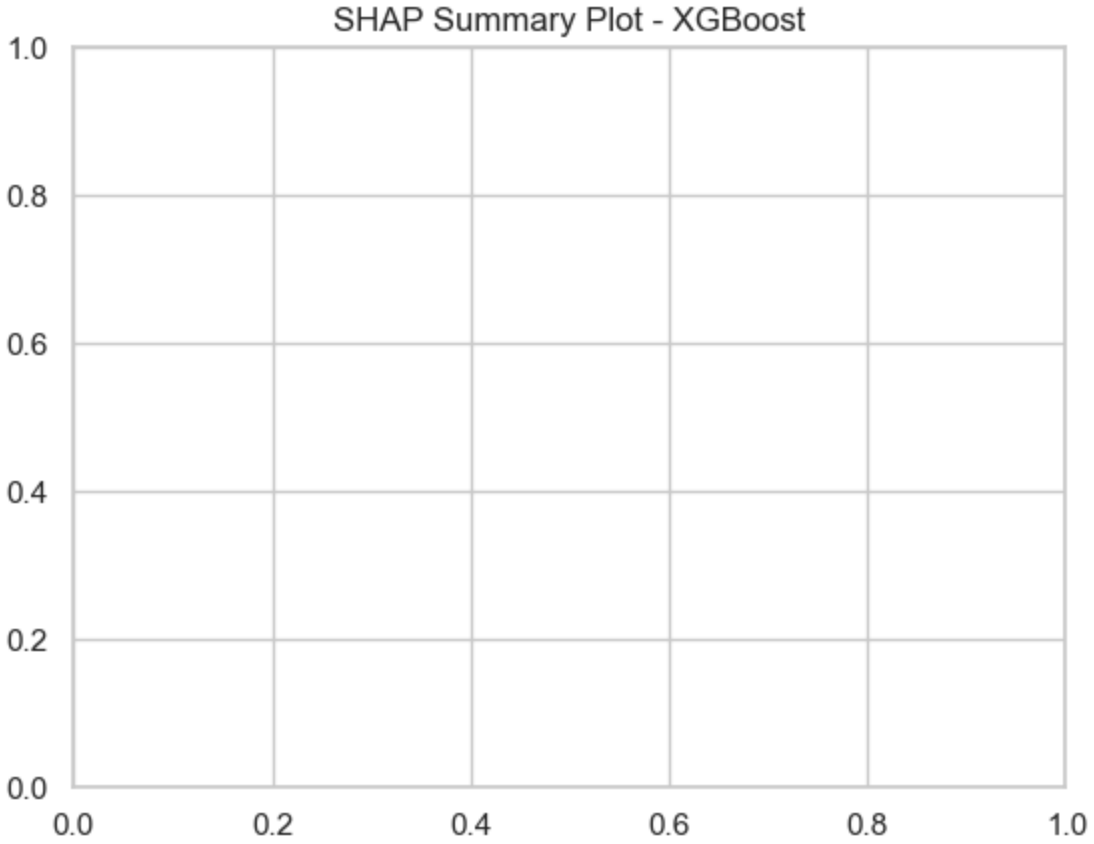

```
SVM AUROC: 0.6921
              precision    recall  f1-score   support

           0       0.60      0.46      0.52       531
           1       0.68      0.79      0.73       779

    accuracy                           0.66      1310
   macro avg       0.64      0.63      0.63      1310
weighted avg       0.65      0.66      0.65      1310

Skipping SHAP for SVM — KernelExplainer too slow.


----------------------------------------
 FINAL ANALYSIS: XGBoost
----------------------------------------
```




```
XGBoost AUROC: 0.6998
              precision    recall  f1-score   support

           0       0.59      0.48      0.53       531
           1       0.68      0.77      0.73       779

    accuracy                           0.65      1310
   macro avg       0.64      0.63      0.63      1310
weighted avg       0.65      0.65      0.65      1310
```


num__odds_diff
num__Spread
num__BlueAge
num__RedTotalFights
num__BlueIsGrappler
num__AgeGap
num__ReachDif
num__AgeDif
num__RWinPct
num__BlueCurrentWinStreak
num__RedIsGrappler
num__BlueIsStriker
num__RedWeightLbs
cat__BlueStance_Orthodox
num__RedCurrentWinStreak
num__BlueCurrentLoseStreak
num__BlueTotalFights
High
Feature value




In [300…

```
# ------------------------------------------------------------------
# XGBOOST-SPECIFIC ADVANCED ANALYSIS - BEST MODEL I KNEW IT
```

```
# ------------------------------------------------------------------
if "XGBoost" in results_summary:
    model_name = "XGBoost"
    pipeline_model = results_summary[model_name]["models"][0]
    classifier = pipeline_model.named_steps["classifier"]
    preprocessor = pipeline_model.named_steps["preprocessor"]

    # Get feature names (post-preprocessing)
    feature_names = preprocessor.get_feature_names_out()

    print("\n========== XGBoost — Baseline vs Model ==========")

    # ---- majority baseline ----
    unique, counts = np.unique(y_train, return_counts=True)
    majority_class = unique[counts.argmax()]

    baseline_preds = np.full(len(y_test), majority_class)
    baseline_acc = accuracy_score(y_test, baseline_preds)
    baseline_f1  = f1_score(y_test, baseline_preds, average="macro")

    model_preds = pipeline_model.predict(X_test)
    model_acc = accuracy_score(y_test, model_preds)
    model_f1  = f1_score(y_test, model_preds, average="macro")

    print(f"Baseline acc: {baseline_acc:.4f}, Model acc: {model_acc:.4f}")
    print(f"Baseline F1:  {baseline_f1:.4f}, Model F1:  {model_f1:.4f}")

    # ---- PERMUTATION IMPORTANCE (must use transformed X_test) ----
    X_test_trans = preprocessor.transform(X_test)
    X_test_dense = (
        X_test_trans.toarray() if hasattr(X_test_trans, "toarray")
        else np.asarray(X_test_trans)
    )

    perm = permutation_importance(
        classifier,
        X_test_dense,
        y_test,
        n_repeats=15,
        random_state=42,
        n_jobs=-1,
        scoring="accuracy"
```

```
    )

    perm_df = pd.DataFrame({
        "Feature": feature_names,
        "Importance": perm.importances_mean,
        "Std": perm.importances_std
    }).sort_values("Importance", ascending=False)

    print("\nTop 15 Permutation Importances:")
    print(perm_df.head(15))

    # ---- XGBoost Internal Importance - I chose: gain / cover / weight ----
    booster = classifier.get_booster()

    gain = booster.get_score(importance_type="gain")
    cover = booster.get_score(importance_type="cover")
    weight = booster.get_score(importance_type="weight")

    def map_booster_keys(score_dict):
        return {
            feature_names[int(k[1:])]: v
            for k, v in score_dict.items()
        }

    gain_map   = map_booster_keys(gain)
    cover_map  = map_booster_keys(cover)
    weight_map = map_booster_keys(weight)

    gain_df = pd.DataFrame({
        "Feature": list(gain_map.keys()),
        "Gain": list(gain_map.values()),
        "Cover": [cover_map.get(f, 0) for f in gain_map.keys()],
        "Weight": [weight_map.get(f, 0) for f in gain_map.keys()],
    }).sort_values("Gain", ascending=False)

    print("\nXGBoost Gain / Cover / Weight:")
    print(gain_df.head(20))

    # ---- SHAP Global ----
    X_shap_global = X_test_dense[:200]

    explainer = shap.TreeExplainer(classifier)
```

```
========== XGBoost — Baseline vs Model ==========
Baseline acc: 0.5947, Model acc: 0.6534
Baseline F1:  0.3729, Model F1:  0.6268

Top 15 Permutation Importances:
                                    Feature  Importance       Std
17                          num__odds_diff    0.121221  0.009547
9                            num__ReachDif    0.001730  0.001291
0                         num__WeightClass    0.001374  0.000798
8                             num__BlueAge    0.001272  0.002811
29                 cat__RedStance_Southpaw    0.000763  0.000683
16                      num__BlueIsStriker    0.000560  0.000338
12                     num__RedTotalFights    0.000458  0.000873
2                 num__BlueCurrentWinStreak    0.000356  0.001692
57     cat__StanceCombo_Switch vs Orthodox    0.000102  0.000259
1               num__BlueCurrentLoseStreak    0.000051  0.000190
55     cat__StanceCombo_Southpaw vs Switch    0.000000  0.000000
45  cat__WeightClassLabel_Women's Flyweight    0.000000  0.000000
34      cat__WeightClassLabel_Bantamweight    0.000000  0.000000
35      cat__WeightClassLabel_Catch Weight    0.000000  0.000000
37         cat__WeightClassLabel_Flyweight    0.000000  0.000000

XGBoost Gain / Cover / Weight:
                                    Feature       Gain       Cover  Weight
17                          num__odds_diff  57.722713  883.203125   537.0
19                             num__Spread  14.390915  468.717590   441.0
18                             num__AgeGap   7.698019  451.328674    71.0
8                             num__BlueAge   7.328346  453.039642   158.0
1               num__BlueCurrentLoseStreak   6.852878  281.897278    18.0
15                    num__BlueIsGrappler   6.804818  827.483093    71.0
7                       num__RedWeightLbs   6.611917  409.421082    44.0
28    cat__StanceCombo_Switch vs Orthodox   6.583469  320.493195     9.0
26    cat__StanceCombo_Orthodox vs Switch   6.393301  271.613617     7.0
12                     num__RedTotalFights   6.311788  350.288300    87.0
14                      num__RedIsGrappler   6.126726  662.121399    48.0
2                num__BlueCurrentWinStreak   6.066645  437.866760    49.0
25     cat__WeightClassLabel_Lightweight   5.799694   79.841141     1.0
21                cat__RedStance_Southpaw   5.544939  775.471252     6.0
27  cat__StanceCombo_Southpaw vs Southpaw   5.305297  124.942085    54.0
9                            num__ReachDif   5.227875  376.921967   110.0
10                             num__AgeDif   5.171468  182.773819    79.0
20               cat__BlueStance_Orthodox   5.062984  202.173096    22.0
```

| | | | | |
|---|---|---|---|---|
| 13 | num__BlueTotalFights | 5.008529 | 91.695442 | 37.0 |
| 24 | cat__WeightClassLabel_Heavyweight | 4.836949 | 300.554871 | 6.0 |