Forecasting and Analyzing Baseball

Double Play Analytics

Overview

This project analyzes the factors associated with forecasting baseball game outcomes and provides key recommendations for success to industry leaders.

Business Understanding

In the highly competitive and financially driven landscape of professional baseball, accurately predicting game outcomes can significantly impact team performance, fan engagement, and revenue generation. By leveraging historical game data our predictive modeling approach aims to provide teams, broadcasters, and stakeholders with actionable insights into future game outcomes. This predictive analytics tool will empower teams to optimize their strategies, such as player selection, pitching rotations, and in-game tactics, leading to improved win rates and overall performance. Ultimately, a reliable predictive model for baseball game outcomes has the potential to drive fan excitement, increase ticket sales, and attract valuable sponsorships, thus contributing to the long-term success and profitability of baseball organizations.

Data and Analysis Limitations

This analysis spanned the COVID-19 pandemic. The pandemic introduced unique challenges for analyzing baseball data, requiring careful consideration of context, data quality, and potential biases when interpreting and modeling data from 2020 and beyond. In 2020 there were 60 regular season games per team, less than half of the standard 162. The shortened and interrupted season impacted player salaries in 2020 and may have affected player performance metrics such as batting averages, earned run averages (ERAs), and fielding percentages. Some players may have performed better or worse than expected due to factors like altered training routines, health concerns, or changes in game dynamics. COVID-19 outbreaks and health protocols may have impacted player availability due to positive cases, contact tracing, or precautionary measures. This could affect team rosters, playing time, and lineup strategies, leading to shifts in player statistics and team performance. Many sports events

in 2020 were held without spectators or with limited attendance to comply with public health guidelines. The absence of fans in stadiums and arenas could have affected player morale, home field advantage, and game dynamics, potentially influencing game outcomes and performance metrics.

Baseball has one of the richest data caches in sports, offering a plethora of statistics and metrics for analysis. Nonetheless, navigating this vast sea of information poses challenges, demanding a focused approach, organization, and an understanding of big data modeling techniques to extract meaningful insights from the wealth

Data and Analysis Preparation

In [1]:
```
# Bringing in packages for EDA, pre-processing, modeling, and visualizations
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline


from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.impute import MissingIndicator, SimpleImputer
from sklearn.dummy import DummyClassifier

from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn import tree
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score, precision_score, confusion_matrix, ConfusionMatri

import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, TensorDataset
from torchvision import transforms
import torch.nn.functional as F
```

Create a single dataset

In [2]:
```
# Each year is saved in a separate .txt file on retrosheets.org
file_paths = ["data/gl2010.txt", "data/gl2011.txt", "data/gl2012.txt", "data/gl2013.txt", "da
              "data/gl2015.txt", "data/gl2016.txt", "data/gl2017.txt", "data/gl2018.txt", "da
              "data/gl2020.txt", "data/gl2021.txt", "data/gl2022.txt", "data/gl2023.txt"]
```

In [3]:
```
dfs=[]
```

In [4]:
```
# Pulling together the individual year files
for file_path in file_paths:
    with open(file_path, "r") as f:
        data = f.readlines()
        data_split = [line.strip().split(",") for line in data]
        df_initial = pd.DataFrame(data_split)
        dfs.append(df_initial)
```

In [5]:
```
# Concatenate all DataFrames in the list
df = pd.concat(dfs, ignore_index=True)
```

In [6]:
```
df.to_csv('data/DF.csv')
```

General Data Understanding

This section does a high level over view of the data. It looks at the initial head, tail, shape, description, missing values, data types, and counts.

In [7]:
```
# Setting up display
# Set the display width to accommodate more characters per row
pd.set_option('display.width', 1000)  # Adjust as needed

# Set display options
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)

# Display the head of the DataFrame
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32484 entries, 0 to 32483
Columns: 179 entries, 0 to 178
dtypes: object(179)
memory usage: 44.4+ MB
```

In [8]:
```
df.head()
```

Out[8]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	
0	"20100404"	"0"	"Sun"	"NYA"	"AL"	1	"BOS"	"AL"	1	7	9	51	"N"	""	""	""	"BOS07"	37440	226	"02030
1	"20100405"	"0"	"Mon"	"MIN"	"AL"	1	"ANA"	"AL"	1	3	6	51	"N"	""	""	""	"ANA01"	43504	180	"02001
2	"20100405"	"0"	"Mon"	"CLE"	"AL"	1	"CHA"	"AL"	1	0	6	51	"D"	""	""	""	"CHI12"	38935	144	"00000
3	"20100405"	"0"	"Mon"	"DET"	"AL"	1	"KCA"	"AL"	1	8	4	54	"D"	""	""	""	"KAN06"	40052	185	"10000
4	"20100405"	"0"	"Mon"	"SEA"	"AL"	1	"OAK"	"AL"	1	5	3	54	"N"	""	""	""	"OAK01"	30686	167	"11100

In [9]:
```
df.tail()
```

Out[9]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	1
32479	"20231001"	"0"	"Sun"	"SDN"	"NL"	162	"CHA"	"AL"	162	2	1	66	"D"	""	""	""	"CHI12"	20588	18
32480	"20231001"	"0"	"Sun"	"CLE"	"AL"	162	"DET"	"AL"	162	2	5	51	"D"	""	""	""	"DET05"	41425	14
32481	"20231001"	"0"	"Sun"	"NYA"	"AL"	162	"KCA"	"AL"	162	2	5	51	"D"	""	""	""	"KAN06"	20662	14
32482	"20231001"	"0"	"Sun"	"TEX"	"AL"	162	"SEA"	"AL"	162	0	1	51	"D"	""	""	""	"SEA03"	43997	12
32483	"20231001"	"0"	"Sun"	"TBA"	"AL"	162	"TOR"	"AL"	162	12	8	54	"D"	""	""	""	"TOR02"	42058	16

In [10]:
```
df.describe()
```

Out[10]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
count	32484	32484	32484	32484	32484	32484	32484	32484	32484	32484	32484	32484	32484	32484
unique	2417	3	7	31	2	163	31	2	163	27	26	64	2	38
top	"20200904"	"0"	"Sat"	"TOR"	"NL"	36	"PHI"	"NL"	19	3	3	51	"N"	""
freq	20	31518	5352	1091	16479	222	1088	16491	225	4625	4547	14655	21409	32446

In [11]:
```
df.shape
```

Out[11]:
```
(32484, 179)
```

In [12]:
```
nan_values = df.isna().any()
nan_values
```

Out[12]:
```
0      False
1      False
2      False
3      False
4      False
5      False
6      False
7      False
8      False
9      False
10     False
11     False
12     False
13     False
14     False
15     False
16     False
17     False
18     False
```

In [13]:
```
df.dtypes
```

Out[13]:
```
0      object
1      object
2      object
3      object
4      object
5      object
6      object
7      object
8      object
9      object
10     object
11     object
12     object
13     object
14     object
15     object
16     object
17     object
18     object
```

In [14]:
```
df.count()
```

Out[14]:
```
0      32484
1      32484
2      32484
3      32484
4      32484
5      32484
6      32484
7      32484
8      32484
9      32484
10     32484
11     32484
12     32484
13     32484
14     32484
15     32484
16     32484
17     32484
18     32484
```

Feature preparation and data manipulation

This section renames columns, drops rows with missing column values of interest, drops columns, converts objects to numeric columns, and strips characters from object columns.

In [15]:
```
# Dropping the columns with minimalinformation at the END of the dataset
# Dropping and individual player, manager, and umpire information; this analysis will focus o
df.drop(columns=df.columns[77:179], inplace=True)
```

In [16]:
```
# Renaming the numeric columns to something more descriptive
new_column_names = {
    0: 'date',
    1: 'num_games',
    2: 'day_of_week',
    3: 'team_visiting',
    4: 'league_visiting',
    5: 'game_num_visiting',
    6: 'team_home',
    7: 'league_home',
    8: 'game_num_home',
    9: 'score_visiting',
    10: 'score_home',
    11: 'outs_in_game',
    12: 'time_of_day',
    13: 'game_completed',
    14: 'forfeit',
    15: 'protest',
    16: 'park_id',
    17: 'attendance',
    18: 'length_min',
    19: 'line_score_visiting',
    20: 'line_score_home',

    # Offense - Visiting
    21: 'at_bats_visiting',
    22: 'hits_visiting',
    23: 'double_visiting',
    24: 'triple_visiting',
    25: 'home_run_visiting',
    26: 'rbi_visiting',
    27: 'sacrifice_hit_visiting',
    28: 'sacrifice_fly_visiting',
    29: 'hit_by_pitch_visiting',
    30: 'walk_visiting',
    31: 'intent_walk_visiting',
    32: 'strikeout_visiting',
    33: 'stolen_base_visiting',
    34: 'caught_stealing_visiting',
    35: 'grounded_into_double_plays_visiting',
    36: 'first_catcher_interfere_visiting',
    37: 'left_on_base_visiting',

    # Pitching - Visiting
    38: 'pitchers_used_visiting',
    39: 'individual_earned_runs_visiting',
    40: 'team_earned_runs_visiting',
    41: 'wild_pitches_visiting',
    42: 'balks_visiting',

    # Defense - Visiting
    43: 'putouts_visiting',
    44: 'assists_visiting',
    45: 'errors_visiting',
    46: 'passed_balls_visiting',
    47: 'double_def_visiting',
    48: 'triple_def_visiting',

    # Offense - Home
    49: 'at_bats_home',
    50: 'hits_home',
    51: 'double_home',
    52: 'triple_home',
    53: 'home_run_home',
```

```
    54: 'rbi_home',
    55: 'sacrifice_hit_home',
    56: 'sacrifice_fly_home',
    57: 'hit_by_pitch_home',
    58: 'walk_home',
    59: 'intent_walk_home',
    60: 'strikeout_home',
    61: 'stolen_base_home',
    62: 'caught_stealing_home',
    63: 'grounded_into_double_plays_home',
    64: 'first_catcher_interfere_home',
    65: 'left_on_base_home',

    # Pitching - Home
    66: 'pitchers_used_home',
    67: 'individual_earned_runs_home',
    68: 'team_earned_runs_home',
    69: 'wild_pitches_home',
    70: 'balks_home',

    # Defense - Home
    71: 'putouts_home',
    72: 'assists_home',
    73: 'errors_home',
    74: 'passed_balls_home',
    75: 'double_def_home',
    76: 'triple_def_home',
}

# Rename columns
df.rename(columns=new_column_names, inplace=True)
```

In [17]:
```
# Removing quotations around strings--
columns_to_convert_strings = [
  'num_games',
  'date',
  'day_of_week',
  'team_visiting',
  'league_visiting',
  'team_home',
  'league_home',
  'time_of_day',
  'park_id',
  'line_score_visiting',
  'line_score_home']
```

In [18]:
```
df[columns_to_convert_strings] = df[columns_to_convert_strings].applymap(lambda x: x.strip('"
```

In [19]:
```
# Converting most numeric columns from strings to integers
columns_to_convert = [
    "num_games",
    "game_num_visiting",
    "game_num_home",
    "score_visiting",
    'game_num_home',
    'score_visiting',
    'score_home',
    "outs_in_game",
    "attendance",
    "length_min",
    "at_bats_visiting",
    "hits_visiting",
    "double_visiting",
    "triple_visiting",
    "home_run_visiting",
    "rbi_visiting",
    "sacrifice_hit_visiting",
    'sacrifice_fly_visiting',
    'hit_by_pitch_visiting',
    'walk_visiting',
    'intent_walk_visiting',
    'strikeout_visiting',
    'stolen_base_visiting',
    'caught_stealing_visiting',
    'grounded_into_double_plays_visiting',
    'first_catcher_interfere_visiting',
    'left_on_base_visiting',
    'pitchers_used_visiting',
    'individual_earned_runs_visiting',
    'team_earned_runs_visiting',
    'wild_pitches_visiting',
    'balks_visiting',
    'putouts_visiting',
    'assists_visiting',
    'errors_visiting',
    'passed_balls_visiting',
    'double_def_visiting',
    'triple_def_visiting',
    'at_bats_home',
    'hits_home',
    'double_home',
    'triple_home',
    'home_run_home',
    'rbi_home',
    'sacrifice_hit_home',
    'sacrifice_fly_home',
    'hit_by_pitch_home',
    'walk_home',
    'intent_walk_home',
    'strikeout_home',
    'stolen_base_home',
    'caught_stealing_home',
    'grounded_into_double_plays_home',
    'first_catcher_interfere_home',
    'left_on_base_home',
    'pitchers_used_home',
    'individual_earned_runs_home',
    'team_earned_runs_home',
    'wild_pitches_home',
    'balks_home',
    'putouts_home',
    'assists_home',
```

```
    'errors_home',
    'passed_balls_home',
    'double_def_home',
    'triple_def_home']
```

In [20]:
```
# Convert columns to numeric, coercing errors to NaN
for column in columns_to_convert:
    df[column] = pd.to_numeric(df[column], errors='coerce')
```

In [21]:
```
df.head()
```

Out[21]:

	date	num_games	day_of_week	team_visiting	league_visiting	game_num_visiting	team_home	league_hom
0	20100404	0	Sun	NYA	AL	1	BOS	A
1	20100405	0	Mon	MIN	AL	1	ANA	A
2	20100405	0	Mon	CLE	AL	1	CHA	A
3	20100405	0	Mon	DET	AL	1	KCA	A
4	20100405	0	Mon	SEA	AL	1	OAK	A

In [22]:
```
df.describe()
```

Out[22]:

	num_games	game_num_visiting	game_num_home	score_visiting	score_home	outs_in_game	attendanc
count	32484.000000	32484.000000	32484.000000	32484.000000	32484.000000	32484.000000	31549.00000
mean	0.044607	80.078654	80.078408	4.356976	4.472663	53.434121	28737.32086
std	0.268992	46.946529	46.954208	3.157740	3.080320	4.870765	11018.86656
min	0.000000	1.000000	1.000000	0.000000	0.000000	27.000000	0.00000
25%	0.000000	39.000000	39.000000	2.000000	2.000000	51.000000	20244.00000
50%	0.000000	79.000000	79.000000	4.000000	4.000000	54.000000	29112.00000
75%	0.000000	121.000000	121.000000	6.000000	6.000000	54.000000	37705.00000
max	2.000000	163.000000	163.000000	28.000000	29.000000	120.000000	59659.00000

In [23]:
```
df.count()
```

Out[23]:

date	32484
num_games	32484
day_of_week	32484
team_visiting	32484
league_visiting	32484
game_num_visiting	32484
team_home	32484
league_home	32484
game_num_home	32484
score_visiting	32484
score_home	32484
outs_in_game	32484
time_of_day	32484
game_completed	32484
forfeit	32484
protest	32484
park_id	32484
attendance	31549
length_min	32446
line_score_visiting	32484
line_score_home	32484
at_bats_visiting	32479
hits_visiting	32484
double_visiting	32446
triple_visiting	32446
home_run_visiting	32484
rbi_visiting	32484
sacrifice_hit_visiting	32484
sacrifice_fly_visiting	32484
hit_by_pitch_visiting	32484
walk_visiting	32484
intent_walk_visiting	32484
strikeout_visiting	32484
stolen_base_visiting	32484
caught_stealing_visiting	32484
grounded_into_double_plays_visiting	32484
first_catcher_interfere_visiting	32484
left_on_base_visiting	32484
pitchers_used_visiting	32484
individual_earned_runs_visiting	32484
team_earned_runs_visiting	32484
wild_pitches_visiting	32484
balks_visiting	32484
putouts_visiting	32484
assists_visiting	32484
errors_visiting	32484
passed_balls_visiting	32484
double_def_visiting	32484
triple_def_visiting	32484
at_bats_home	32484
hits_home	32484
double_home	32484
triple_home	32484
home_run_home	32484
rbi_home	32484
sacrifice_hit_home	32484
sacrifice_fly_home	32484
hit_by_pitch_home	32484
walk_home	32484
intent_walk_home	32484
strikeout_home	32484
stolen_base_home	32484
caught_stealing_home	32484
grounded_into_double_plays_home	32484

```
first_catcher_interfere_home          32484
left_on_base_home                     32484
pitchers_used_home                    32484
individual_earned_runs_home           32484
team_earned_runs_home                 32484
wild_pitches_home                     32484
balks_home                            32484
putouts_home                          32484
assists_home                          32484
errors_home                           32484
passed_balls_home                     32484
double_def_home                       32484
triple_def_home                       32484
dtype: int64
```

In [24]:
```
df.dtypes
```

Out[24]:

date	object
num_games	int64
day_of_week	object
team_visiting	object
league_visiting	object
game_num_visiting	int64
team_home	object
league_home	object
game_num_home	int64
score_visiting	int64
score_home	int64
outs_in_game	int64
time_of_day	object
game_completed	object
forfeit	object
protest	object
park_id	object
attendance	float64
length_min	float64
line_score_visiting	object
line_score_home	object
at_bats_visiting	float64
hits_visiting	int64
double_visiting	float64
triple_visiting	float64
home_run_visiting	int64
rbi_visiting	int64
sacrifice_hit_visiting	int64
sacrifice_fly_visiting	int64
hit_by_pitch_visiting	int64
walk_visiting	int64
intent_walk_visiting	int64
strikeout_visiting	int64
stolen_base_visiting	int64
caught_stealing_visiting	int64
grounded_into_double_plays_visiting	int64
first_catcher_interfere_visiting	int64
left_on_base_visiting	int64
pitchers_used_visiting	int64
individual_earned_runs_visiting	int64
team_earned_runs_visiting	int64
wild_pitches_visiting	int64
balks_visiting	int64
putouts_visiting	int64
assists_visiting	int64
errors_visiting	int64
passed_balls_visiting	int64
double_def_visiting	int64
triple_def_visiting	int64
at_bats_home	int64
hits_home	int64
double_home	int64
triple_home	int64
home_run_home	int64
rbi_home	int64
sacrifice_hit_home	int64
sacrifice_fly_home	int64
hit_by_pitch_home	int64
walk_home	int64
intent_walk_home	int64
strikeout_home	int64
stolen_base_home	int64
caught_stealing_home	int64
grounded_into_double_plays_home	int64

```
first_catcher_interfere_home            int64
left_on_base_home                       int64
pitchers_used_home                      int64
individual_earned_runs_home             int64
team_earned_runs_home                   int64
wild_pitches_home                       int64
balks_home                              int64
putouts_home                            int64
assists_home                            int64
errors_home                             int64
passed_balls_home                       int64
double_def_home                         int64
triple_def_home                         int64
dtype: object
```

In [25]:
```
# Dropping rows with missing values in the listed columns
# at_bats_visiting - 5 rows with missing data
# double_visiting - 38 rows with missing data
# triple_visiting - 38 rows with missing data
# A total of 38 rows will be deleted; there are 43 duplicates in the 81 rows identified
df.dropna(subset=["at_bats_visiting", "double_visiting", "triple_visiting"], inplace=True)
```

In [26]:
```
# Drop the one row where the game ended in a tie; found this obs because when calculated run
df.drop(index=16957, inplace=True)
```

In [27]:
```
# Drop rows where there was a protest
# A total of 11 rows will be deleted
df.dropna(subset=['protest'], inplace=True)
df = df[(df['protest'] != '"V"') & (df['protest'] != '"H"')]
```

In [28]:
```
# This drops the partial 2020 season; not sure I want to drop it. It was an odd year,
# but unless I'm looking at year as a factor what is the impact of keeping it in?
# df = df[df['year'] != 2020]
```

In [29]:
```
df.drop(columns=["attendance", "num_games", "length_min", "game_completed", "forfeit", "prote
```

In [30]:
```
# FLO/MIA is the only team that switched three letter codes during this time period. Combinin
df['team_visiting'].replace({"FLO": "MIA"}, inplace=True)
df['team_home'].replace({"FLO": "MIA"}, inplace=True)
```

In [31]:
```
df.head()
```

Out[31]:

	date	team_visiting	team_home	score_visiting	score_home	outs_in_game	at_bats_visiting	hits_visiting	d
0	20100404	NYA	BOS	7	9	51	37.0	12	
1	20100405	MIN	ANA	3	6	51	32.0	7	
2	20100405	CLE	CHA	0	6	51	30.0	4	
3	20100405	DET	KCA	8	4	54	39.0	12	
4	20100405	SEA	OAK	5	3	54	31.0	6	

In [32]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 32434 entries, 0 to 32483
Data columns (total 62 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   date                                  32434 non-null  object
 1   team_visiting                         32434 non-null  object
 2   team_home                             32434 non-null  object
 3   score_visiting                        32434 non-null  int64
 4   score_home                            32434 non-null  int64
 5   outs_in_game                          32434 non-null  int64
 6   at_bats_visiting                      32434 non-null  float64
 7   hits_visiting                         32434 non-null  int64
 8   double_visiting                       32434 non-null  float64
 9   triple_visiting                       32434 non-null  float64
 10  home_run_visiting                     32434 non-null  int64
 11  rbi_visiting                          32434 non-null  int64
 12  sacrifice_hit_visiting                32434 non-null  int64
 13  sacrifice_fly_visiting                32434 non-null  int64
 14  hit_by_pitch_visiting                 32434 non-null  int64
 15  walk_visiting                         32434 non-null  int64
 16  intent_walk_visiting                  32434 non-null  int64
 17  strikeout_visiting                    32434 non-null  int64
 18  stolen_base_visiting                  32434 non-null  int64
 19  caught_stealing_visiting              32434 non-null  int64
 20  grounded_into_double_plays_visiting   32434 non-null  int64
 21  first_catcher_interfere_visiting      32434 non-null  int64
 22  left_on_base_visiting                 32434 non-null  int64
 23  pitchers_used_visiting                32434 non-null  int64
 24  individual_earned_runs_visiting       32434 non-null  int64
 25  team_earned_runs_visiting             32434 non-null  int64
 26  wild_pitches_visiting                 32434 non-null  int64
 27  balks_visiting                        32434 non-null  int64
 28  putouts_visiting                      32434 non-null  int64
 29  assists_visiting                      32434 non-null  int64
 30  errors_visiting                       32434 non-null  int64
 31  passed_balls_visiting                 32434 non-null  int64
 32  double_def_visiting                   32434 non-null  int64
 33  triple_def_visiting                   32434 non-null  int64
 34  at_bats_home                          32434 non-null  int64
 35  hits_home                             32434 non-null  int64
 36  double_home                           32434 non-null  int64
 37  triple_home                           32434 non-null  int64
 38  home_run_home                         32434 non-null  int64
 39  rbi_home                              32434 non-null  int64
 40  sacrifice_hit_home                    32434 non-null  int64
 41  sacrifice_fly_home                    32434 non-null  int64
 42  hit_by_pitch_home                     32434 non-null  int64
 43  walk_home                             32434 non-null  int64
 44  intent_walk_home                      32434 non-null  int64
 45  strikeout_home                        32434 non-null  int64
 46  stolen_base_home                      32434 non-null  int64
 47  caught_stealing_home                  32434 non-null  int64
 48  grounded_into_double_plays_home       32434 non-null  int64
 49  first_catcher_interfere_home          32434 non-null  int64
 50  left_on_base_home                     32434 non-null  int64
 51  pitchers_used_home                    32434 non-null  int64
 52  individual_earned_runs_home           32434 non-null  int64
 53  team_earned_runs_home                 32434 non-null  int64
 54  wild_pitches_home                     32434 non-null  int64
 55  balks_home                            32434 non-null  int64
 56  putouts_home                          32434 non-null  int64
 57  assists_home                          32434 non-null  int64
 58  errors_home                           32434 non-null  int64
```

```
 59  passed_balls_home                    32434 non-null  int64
 60  double_def_home                      32434 non-null  int64
 61  triple_def_home                      32434 non-null  int64
dtypes: float64(3), int64(56), object(3)
memory usage: 15.6+ MB
```

Create new features

This section creates several new features: year, winning team, losing team, run differential, total runs, and on base percentage (OBP).

In [33]:
```python
# Creating an integer four-digit year feature
# Extract the first four characters
year_digits = df['date'].str[0:4]

# Convert the extracted substring to numeric
df['year'] = pd.to_numeric(year_digits)
df['year'] = df['year'].astype(int)
```

In [34]:
```python
# Creating a feature for the LOSING team for each game/row
def compare_and_get_value(row):
    if row['score_visiting'] < row['score_home']:
        return row['team_visiting']
    else:
        return row['team_home']

# Create a new feature based on the comparison
df['losing_team'] = df.apply(compare_and_get_value, axis=1)
```

In [35]:
```
df['losing_team'].value_counts()
```

Out[35]:
```
MIA    1204
COL    1186
KCA    1182
PIT    1154
CHA    1151
BAL    1150
SDN    1138
ARI    1138
DET    1133
CIN    1128
MIN    1125
SEA    1112
OAK    1093
ANA    1093
PHI    1090
NYN    1090
CHN    1086
TEX    1084
TOR    1067
WAS    1066
HOU    1057
SFN    1047
MIL    1041
CLE    1025
BOS    1020
ATL     992
TBA     981
SLN     979
NYA     936
LAN     886
Name: losing_team, dtype: int64
```

In [36]:
```
# Creating a feature for the WINNING team for each game/row
def compare_and_get_value(row):
    if row['score_visiting'] > row['score_home']:
        return row['team_visiting']
    else:
        return row['team_home']

# Create a new feature based on the comparison
df['winning_team'] = df.apply(compare_and_get_value, axis=1)
```

In [37]:
```
df['winning_team'].value_counts()
```

Out[37]:
```
LAN    1277
NYA    1226
TBA    1183
SLN    1182
ATL    1170
BOS    1142
CLE    1136
MIL    1126
SFN    1115
HOU    1106
TOR    1097
WAS    1090
TEX    1083
CHN    1077
PHI    1073
ANA    1070
OAK    1070
NYN    1068
SEA    1052
MIN    1039
CIN    1031
ARI    1027
DET    1025
SDN    1023
BAL    1015
CHA    1009
PIT    1008
COL     979
KCA     978
MIA     957
Name: winning_team, dtype: int64
```

In [38]:
```
# Creating a feature for run differential for each game/row
def calculate_run_differential(row):
    if row['winning_team'] == row['team_home']:
        return row['score_home'] - row['score_visiting']
    else:
        return row['score_visiting'] - row['score_home']

# Apply the function to create a new run differential column
df['run_differential'] = df.apply(calculate_run_differential, axis=1)
```

In [39]:
```
df['run_differential'].value_counts()
```

Out[39]:
```
1     9308
2     5916
3     4622
4     3817
5     2695
6     1974
7     1356
8      919
9      647
10     432
11     269
12     162
13     139
14      72
15      40
16      28
17      14
18       9
19       5
20       4
21       4
23       1
24       1
Name: run_differential, dtype: int64
```

In [40]:
```
# Total Runs per Game
df['total_runs'] = df['score_home'] + df['score_visiting']
df['total_runs'].value_counts()
```

Out[40]:
```
7     3551
9     3347
5     3175
8     2555
11    2389
6     2379
10    2133
3     2065
4     1655
13    1650
12    1554
15    1068
14    1060
2      662
1      628
17     608
16     606
18     353
19     289
20     200
21     174
22      94
23      87
25      47
24      47
26      19
28       9
27       9
29       9
30       4
31       3
32       3
38       1
33       1
Name: total_runs, dtype: int64
```

In [41]:
```
# OBP = (Hits + Walks + Hit by Pitch) ÷ (At Bats + Walks + Hit by Pitch + Sacrifice Flies)
df['obp_home'] = (df['hits_home'] + df['walk_home'] + df['hit_by_pitch_home']) / (df['at_bats
df['obp_home'].describe()
```

Out[41]:
```
count    32434.000000
mean         0.318961
std          0.083758
min          0.000000
25%          0.261905
50%          0.322581
75%          0.378378
max          0.603448
Name: obp_home, dtype: float64
```

In [42]:
```
# OBP = (Hits + Walks + Hit by Pitch) ÷ (At Bats + Walks + Hit by Pitch + Sacrifice Flies)
df['obp_visiting'] = (df['hits_visiting'] + df['walk_visiting'] + df['hit_by_pitch_visiting']
df['obp_visiting'].describe()
```

Out[42]:
```
count    32434.000000
mean         0.305073
std          0.081107
min          0.000000
25%          0.250000
50%          0.305556
75%          0.361111
max          0.593220
Name: obp_visiting, dtype: float64
```

In [43]:
```
df.head()
```

Out[43]:

	date	team_visiting	team_home	score_visiting	score_home	outs_in_game	at_bats_visiting	hits_visiting	d
0	20100404	NYA	BOS	7	9	51	37.0	12	
1	20100405	MIN	ANA	3	6	51	32.0	7	
2	20100405	CLE	CHA	0	6	51	30.0	4	
3	20100405	DET	KCA	8	4	54	39.0	12	
4	20100405	SEA	OAK	5	3	54	31.0	6	

Dataframe manipulation

In this section we change the dataset from one row per game to two rows per game, one for each team for each game. Each paired row receives the same game ID to ensure pairs are kept together in the train_test_split to prevent data leakage.

In [44]:
```
# Add unique numeric IDs
df['ID'] = range(1, len(df) + 1)
df['ID'] = df['ID'].astype(str) + '_w'  # Adding suffix '_w' to original IDs

# Duplicate rows and modify IDs for duplicates
duplicates = df.copy()
duplicates['ID'] = duplicates['ID'].apply(lambda x: x.replace('_w', '_l'))  # Changing suffix

# Concatenate original DataFrame and duplicates
result = pd.concat([df, duplicates], ignore_index=True)
```

In [45]:
```
result['win'] = result['ID'].apply(lambda x: 1 if x.endswith('_w') else 0)
```

In [46]:
```
#result.info()
```

In [47]:
```
result_sorted = result.sort_values(by='ID')
result_sorted.head(6)
```

Out[47]:

	date	team_visiting	team_home	score_visiting	score_home	outs_in_game	at_bats_visiting	hits_visitin
42433	20140422	NYA	BOS	9	3	54	41.0	1
9999	20140422	NYA	BOS	9	3	54	41.0	1
42434	20140422	KCA	CLE	8	2	54	38.0	1
10000	20140422	KCA	CLE	8	2	54	38.0	1
42435	20140422	CHA	DET	6	8	51	37.0	1
10001	20140422	CHA	DET	6	8	51	37.0	1

In [48]:
```
# Since I divided the winning team and losing team from each row and stacked them on top of o
# Create four new DataFrames:
# (1) Home team wins
# (2) Visiting team wins
# (3) Home team loses
# (4) Visiting team loses

home_win_df = result[(result['win'] == 1) & (result['winning_team'] == result['team_home'])].
visiting_win_df = result[(result['win'] == 1) & (result['winning_team'] == result['team_visit
home_lose_df = result[(result['win'] == 0) & (result['losing_team'] == result['team_home'])].
visiting_lose_df = result[(result['win'] == 0) & (result['losing_team'] == result['team_visit
```

In [49]:
```
home_win_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 17359 entries, 0 to 32432
Data columns (total 71 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   date                                  17359 non-null  object
 1   team_visiting                         17359 non-null  object
 2   team_home                             17359 non-null  object
 3   score_visiting                        17359 non-null  int64
 4   score_home                            17359 non-null  int64
 5   outs_in_game                          17359 non-null  int64
 6   at_bats_visiting                      17359 non-null  float64
 7   hits_visiting                         17359 non-null  int64
 8   double_visiting                       17359 non-null  float64
 9   triple_visiting                       17359 non-null  float64
 10  home_run_visiting                     17359 non-null  int64
 11  rbi_visiting                          17359 non-null  int64
 12  sacrifice_hit_visiting                17359 non-null  int64
 13  sacrifice_fly_visiting                17359 non-null  int64
 14  hit_by_pitch_visiting                 17359 non-null  int64
 15  walk_visiting                         17359 non-null  int64
 16  intent_walk_visiting                  17359 non-null  int64
 17  strikeout_visiting                    17359 non-null  int64
 18  stolen_base_visiting                  17359 non-null  int64
 19  caught_stealing_visiting              17359 non-null  int64
 20  grounded_into_double_plays_visiting   17359 non-null  int64
 21  first_catcher_interfere_visiting      17359 non-null  int64
 22  left_on_base_visiting                 17359 non-null  int64
 23  pitchers_used_visiting                17359 non-null  int64
 24  individual_earned_runs_visiting       17359 non-null  int64
 25  team_earned_runs_visiting             17359 non-null  int64
 26  wild_pitches_visiting                 17359 non-null  int64
 27  balks_visiting                        17359 non-null  int64
 28  putouts_visiting                      17359 non-null  int64
 29  assists_visiting                      17359 non-null  int64
 30  errors_visiting                       17359 non-null  int64
 31  passed_balls_visiting                 17359 non-null  int64
 32  double_def_visiting                   17359 non-null  int64
 33  triple_def_visiting                   17359 non-null  int64
 34  at_bats_home                          17359 non-null  int64
 35  hits_home                             17359 non-null  int64
 36  double_home                           17359 non-null  int64
 37  triple_home                           17359 non-null  int64
 38  home_run_home                         17359 non-null  int64
 39  rbi_home                              17359 non-null  int64
 40  sacrifice_hit_home                    17359 non-null  int64
 41  sacrifice_fly_home                    17359 non-null  int64
 42  hit_by_pitch_home                     17359 non-null  int64
 43  walk_home                             17359 non-null  int64
 44  intent_walk_home                      17359 non-null  int64
 45  strikeout_home                        17359 non-null  int64
 46  stolen_base_home                      17359 non-null  int64
 47  caught_stealing_home                  17359 non-null  int64
 48  grounded_into_double_plays_home       17359 non-null  int64
 49  first_catcher_interfere_home          17359 non-null  int64
 50  left_on_base_home                     17359 non-null  int64
 51  pitchers_used_home                    17359 non-null  int64
 52  individual_earned_runs_home           17359 non-null  int64
 53  team_earned_runs_home                 17359 non-null  int64
 54  wild_pitches_home                     17359 non-null  int64
 55  balks_home                            17359 non-null  int64
 56  putouts_home                          17359 non-null  int64
 57  assists_home                          17359 non-null  int64
 58  errors_home                           17359 non-null  int64
```

```
 59  passed_balls_home                   17359 non-null  int64
 60  double_def_home                     17359 non-null  int64
 61  triple_def_home                     17359 non-null  int64
 62  year                                17359 non-null  int32
 63  losing_team                         17359 non-null  object
 64  winning_team                        17359 non-null  object
 65  run_differential                    17359 non-null  int64
 66  total_runs                          17359 non-null  int64
 67  obp_home                            17359 non-null  float64
 68  obp_visiting                        17359 non-null  float64
 69  ID                                  17359 non-null  object
 70  win                                 17359 non-null  int64
dtypes: float64(5), int32(1), int64(59), object(6)
memory usage: 9.5+ MB
```

In [50]:
```
visiting_win_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15075 entries, 3 to 32433
Data columns (total 71 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   date                                  15075 non-null  object
 1   team_visiting                         15075 non-null  object
 2   team_home                             15075 non-null  object
 3   score_visiting                        15075 non-null  int64
 4   score_home                            15075 non-null  int64
 5   outs_in_game                          15075 non-null  int64
 6   at_bats_visiting                      15075 non-null  float64
 7   hits_visiting                         15075 non-null  int64
 8   double_visiting                       15075 non-null  float64
 9   triple_visiting                       15075 non-null  float64
 10  home_run_visiting                     15075 non-null  int64
 11  rbi_visiting                          15075 non-null  int64
 12  sacrifice_hit_visiting                15075 non-null  int64
 13  sacrifice_fly_visiting                15075 non-null  int64
 14  hit_by_pitch_visiting                 15075 non-null  int64
 15  walk_visiting                         15075 non-null  int64
 16  intent_walk_visiting                  15075 non-null  int64
 17  strikeout_visiting                    15075 non-null  int64
 18  stolen_base_visiting                  15075 non-null  int64
 19  caught_stealing_visiting              15075 non-null  int64
 20  grounded_into_double_plays_visiting   15075 non-null  int64
 21  first_catcher_interfere_visiting      15075 non-null  int64
 22  left_on_base_visiting                 15075 non-null  int64
 23  pitchers_used_visiting                15075 non-null  int64
 24  individual_earned_runs_visiting       15075 non-null  int64
 25  team_earned_runs_visiting             15075 non-null  int64
 26  wild_pitches_visiting                 15075 non-null  int64
 27  balks_visiting                        15075 non-null  int64
 28  putouts_visiting                      15075 non-null  int64
 29  assists_visiting                      15075 non-null  int64
 30  errors_visiting                       15075 non-null  int64
 31  passed_balls_visiting                 15075 non-null  int64
 32  double_def_visiting                   15075 non-null  int64
 33  triple_def_visiting                   15075 non-null  int64
 34  at_bats_home                          15075 non-null  int64
 35  hits_home                             15075 non-null  int64
 36  double_home                           15075 non-null  int64
 37  triple_home                           15075 non-null  int64
 38  home_run_home                         15075 non-null  int64
 39  rbi_home                              15075 non-null  int64
 40  sacrifice_hit_home                    15075 non-null  int64
 41  sacrifice_fly_home                    15075 non-null  int64
 42  hit_by_pitch_home                     15075 non-null  int64
 43  walk_home                             15075 non-null  int64
 44  intent_walk_home                      15075 non-null  int64
 45  strikeout_home                        15075 non-null  int64
 46  stolen_base_home                      15075 non-null  int64
 47  caught_stealing_home                  15075 non-null  int64
 48  grounded_into_double_plays_home       15075 non-null  int64
 49  first_catcher_interfere_home          15075 non-null  int64
 50  left_on_base_home                     15075 non-null  int64
 51  pitchers_used_home                    15075 non-null  int64
 52  individual_earned_runs_home           15075 non-null  int64
 53  team_earned_runs_home                 15075 non-null  int64
 54  wild_pitches_home                     15075 non-null  int64
 55  balks_home                            15075 non-null  int64
 56  putouts_home                          15075 non-null  int64
 57  assists_home                          15075 non-null  int64
 58  errors_home                           15075 non-null  int64
```

```
 59  passed_balls_home                   15075 non-null  int64
 60  double_def_home                     15075 non-null  int64
 61  triple_def_home                     15075 non-null  int64
 62  year                                15075 non-null  int32
 63  losing_team                         15075 non-null  object
 64  winning_team                        15075 non-null  object
 65  run_differential                    15075 non-null  int64
 66  total_runs                          15075 non-null  int64
 67  obp_home                            15075 non-null  float64
 68  obp_visiting                        15075 non-null  float64
 69  ID                                  15075 non-null  object
 70  win                                 15075 non-null  int64
dtypes: float64(5), int32(1), int64(59), object(6)
memory usage: 8.2+ MB
```

In [51]:
```
home_lose_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15075 entries, 32437 to 64867
Data columns (total 71 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   date                                  15075 non-null  object
 1   team_visiting                         15075 non-null  object
 2   team_home                             15075 non-null  object
 3   score_visiting                        15075 non-null  int64
 4   score_home                            15075 non-null  int64
 5   outs_in_game                          15075 non-null  int64
 6   at_bats_visiting                      15075 non-null  float64
 7   hits_visiting                         15075 non-null  int64
 8   double_visiting                       15075 non-null  float64
 9   triple_visiting                       15075 non-null  float64
 10  home_run_visiting                     15075 non-null  int64
 11  rbi_visiting                          15075 non-null  int64
 12  sacrifice_hit_visiting                15075 non-null  int64
 13  sacrifice_fly_visiting                15075 non-null  int64
 14  hit_by_pitch_visiting                 15075 non-null  int64
 15  walk_visiting                         15075 non-null  int64
 16  intent_walk_visiting                  15075 non-null  int64
 17  strikeout_visiting                    15075 non-null  int64
 18  stolen_base_visiting                  15075 non-null  int64
 19  caught_stealing_visiting              15075 non-null  int64
 20  grounded_into_double_plays_visiting   15075 non-null  int64
 21  first_catcher_interfere_visiting      15075 non-null  int64
 22  left_on_base_visiting                 15075 non-null  int64
 23  pitchers_used_visiting                15075 non-null  int64
 24  individual_earned_runs_visiting       15075 non-null  int64
 25  team_earned_runs_visiting             15075 non-null  int64
 26  wild_pitches_visiting                 15075 non-null  int64
 27  balks_visiting                        15075 non-null  int64
 28  putouts_visiting                      15075 non-null  int64
 29  assists_visiting                      15075 non-null  int64
 30  errors_visiting                       15075 non-null  int64
 31  passed_balls_visiting                 15075 non-null  int64
 32  double_def_visiting                   15075 non-null  int64
 33  triple_def_visiting                   15075 non-null  int64
 34  at_bats_home                          15075 non-null  int64
 35  hits_home                             15075 non-null  int64
 36  double_home                           15075 non-null  int64
 37  triple_home                           15075 non-null  int64
 38  home_run_home                         15075 non-null  int64
 39  rbi_home                              15075 non-null  int64
 40  sacrifice_hit_home                    15075 non-null  int64
 41  sacrifice_fly_home                    15075 non-null  int64
 42  hit_by_pitch_home                     15075 non-null  int64
 43  walk_home                             15075 non-null  int64
 44  intent_walk_home                      15075 non-null  int64
 45  strikeout_home                        15075 non-null  int64
 46  stolen_base_home                      15075 non-null  int64
 47  caught_stealing_home                  15075 non-null  int64
 48  grounded_into_double_plays_home       15075 non-null  int64
 49  first_catcher_interfere_home          15075 non-null  int64
 50  left_on_base_home                     15075 non-null  int64
 51  pitchers_used_home                    15075 non-null  int64
 52  individual_earned_runs_home           15075 non-null  int64
 53  team_earned_runs_home                 15075 non-null  int64
 54  wild_pitches_home                     15075 non-null  int64
 55  balks_home                            15075 non-null  int64
 56  putouts_home                          15075 non-null  int64
 57  assists_home                          15075 non-null  int64
 58  errors_home                           15075 non-null  int64
```

```
 59  passed_balls_home                   15075 non-null  int64
 60  double_def_home                     15075 non-null  int64
 61  triple_def_home                     15075 non-null  int64
 62  year                                15075 non-null  int32
 63  losing_team                         15075 non-null  object
 64  winning_team                        15075 non-null  object
 65  run_differential                    15075 non-null  int64
 66  total_runs                          15075 non-null  int64
 67  obp_home                            15075 non-null  float64
 68  obp_visiting                        15075 non-null  float64
 69  ID                                  15075 non-null  object
 70  win                                 15075 non-null  int64
dtypes: float64(5), int32(1), int64(59), object(6)
memory usage: 8.2+ MB
```

In [52]:
```
visiting_lose_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 17359 entries, 32434 to 64866
Data columns (total 71 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   date                                  17359 non-null  object
 1   team_visiting                         17359 non-null  object
 2   team_home                             17359 non-null  object
 3   score_visiting                        17359 non-null  int64
 4   score_home                            17359 non-null  int64
 5   outs_in_game                          17359 non-null  int64
 6   at_bats_visiting                      17359 non-null  float64
 7   hits_visiting                         17359 non-null  int64
 8   double_visiting                       17359 non-null  float64
 9   triple_visiting                       17359 non-null  float64
 10  home_run_visiting                     17359 non-null  int64
 11  rbi_visiting                          17359 non-null  int64
 12  sacrifice_hit_visiting                17359 non-null  int64
 13  sacrifice_fly_visiting                17359 non-null  int64
 14  hit_by_pitch_visiting                 17359 non-null  int64
 15  walk_visiting                         17359 non-null  int64
 16  intent_walk_visiting                  17359 non-null  int64
 17  strikeout_visiting                    17359 non-null  int64
 18  stolen_base_visiting                  17359 non-null  int64
 19  caught_stealing_visiting              17359 non-null  int64
 20  grounded_into_double_plays_visiting   17359 non-null  int64
 21  first_catcher_interfere_visiting      17359 non-null  int64
 22  left_on_base_visiting                 17359 non-null  int64
 23  pitchers_used_visiting                17359 non-null  int64
 24  individual_earned_runs_visiting       17359 non-null  int64
 25  team_earned_runs_visiting             17359 non-null  int64
 26  wild_pitches_visiting                 17359 non-null  int64
 27  balks_visiting                        17359 non-null  int64
 28  putouts_visiting                      17359 non-null  int64
 29  assists_visiting                      17359 non-null  int64
 30  errors_visiting                       17359 non-null  int64
 31  passed_balls_visiting                 17359 non-null  int64
 32  double_def_visiting                   17359 non-null  int64
 33  triple_def_visiting                   17359 non-null  int64
 34  at_bats_home                          17359 non-null  int64
 35  hits_home                             17359 non-null  int64
 36  double_home                           17359 non-null  int64
 37  triple_home                           17359 non-null  int64
 38  home_run_home                         17359 non-null  int64
 39  rbi_home                              17359 non-null  int64
 40  sacrifice_hit_home                    17359 non-null  int64
 41  sacrifice_fly_home                    17359 non-null  int64
 42  hit_by_pitch_home                     17359 non-null  int64
 43  walk_home                             17359 non-null  int64
 44  intent_walk_home                      17359 non-null  int64
 45  strikeout_home                        17359 non-null  int64
 46  stolen_base_home                      17359 non-null  int64
 47  caught_stealing_home                  17359 non-null  int64
 48  grounded_into_double_plays_home       17359 non-null  int64
 49  first_catcher_interfere_home          17359 non-null  int64
 50  left_on_base_home                     17359 non-null  int64
 51  pitchers_used_home                    17359 non-null  int64
 52  individual_earned_runs_home           17359 non-null  int64
 53  team_earned_runs_home                 17359 non-null  int64
 54  wild_pitches_home                     17359 non-null  int64
 55  balks_home                            17359 non-null  int64
 56  putouts_home                          17359 non-null  int64
 57  assists_home                          17359 non-null  int64
 58  errors_home                           17359 non-null  int64
```

```
 59  passed_balls_home                  17359 non-null  int64
 60  double_def_home                    17359 non-null  int64
 61  triple_def_home                    17359 non-null  int64
 62  year                               17359 non-null  int32
 63  losing_team                        17359 non-null  object
 64  winning_team                       17359 non-null  object
 65  run_differential                   17359 non-null  int64
 66  total_runs                         17359 non-null  int64
 67  obp_home                           17359 non-null  float64
 68  obp_visiting                       17359 non-null  float64
 69  ID                                 17359 non-null  object
 70  win                                17359 non-null  int64
dtypes: float64(5), int32(1), int64(59), object(6)
memory usage: 9.5+ MB
```

In [53]:
```
team_home_df = pd.concat([home_win_df, home_lose_df], ignore_index=True)
```

In [54]:
```
team_visiting_df = pd.concat([visiting_win_df, visiting_lose_df], ignore_index=True)
```

In [55]:
```
team_home_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32434 entries, 0 to 32433
Data columns (total 71 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   date                                  32434 non-null  object
 1   team_visiting                         32434 non-null  object
 2   team_home                             32434 non-null  object
 3   score_visiting                        32434 non-null  int64
 4   score_home                            32434 non-null  int64
 5   outs_in_game                          32434 non-null  int64
 6   at_bats_visiting                      32434 non-null  float64
 7   hits_visiting                         32434 non-null  int64
 8   double_visiting                       32434 non-null  float64
 9   triple_visiting                       32434 non-null  float64
 10  home_run_visiting                     32434 non-null  int64
 11  rbi_visiting                          32434 non-null  int64
 12  sacrifice_hit_visiting                32434 non-null  int64
 13  sacrifice_fly_visiting                32434 non-null  int64
 14  hit_by_pitch_visiting                 32434 non-null  int64
 15  walk_visiting                         32434 non-null  int64
 16  intent_walk_visiting                  32434 non-null  int64
 17  strikeout_visiting                    32434 non-null  int64
 18  stolen_base_visiting                  32434 non-null  int64
 19  caught_stealing_visiting              32434 non-null  int64
 20  grounded_into_double_plays_visiting   32434 non-null  int64
 21  first_catcher_interfere_visiting      32434 non-null  int64
 22  left_on_base_visiting                 32434 non-null  int64
 23  pitchers_used_visiting                32434 non-null  int64
 24  individual_earned_runs_visiting       32434 non-null  int64
 25  team_earned_runs_visiting             32434 non-null  int64
 26  wild_pitches_visiting                 32434 non-null  int64
 27  balks_visiting                        32434 non-null  int64
 28  putouts_visiting                      32434 non-null  int64
 29  assists_visiting                      32434 non-null  int64
 30  errors_visiting                       32434 non-null  int64
 31  passed_balls_visiting                 32434 non-null  int64
 32  double_def_visiting                   32434 non-null  int64
 33  triple_def_visiting                   32434 non-null  int64
 34  at_bats_home                          32434 non-null  int64
 35  hits_home                             32434 non-null  int64
 36  double_home                           32434 non-null  int64
 37  triple_home                           32434 non-null  int64
 38  home_run_home                         32434 non-null  int64
 39  rbi_home                              32434 non-null  int64
 40  sacrifice_hit_home                    32434 non-null  int64
 41  sacrifice_fly_home                    32434 non-null  int64
 42  hit_by_pitch_home                     32434 non-null  int64
 43  walk_home                             32434 non-null  int64
 44  intent_walk_home                      32434 non-null  int64
 45  strikeout_home                        32434 non-null  int64
 46  stolen_base_home                      32434 non-null  int64
 47  caught_stealing_home                  32434 non-null  int64
 48  grounded_into_double_plays_home       32434 non-null  int64
 49  first_catcher_interfere_home          32434 non-null  int64
 50  left_on_base_home                     32434 non-null  int64
 51  pitchers_used_home                    32434 non-null  int64
 52  individual_earned_runs_home           32434 non-null  int64
 53  team_earned_runs_home                 32434 non-null  int64
 54  wild_pitches_home                     32434 non-null  int64
 55  balks_home                            32434 non-null  int64
 56  putouts_home                          32434 non-null  int64
 57  assists_home                          32434 non-null  int64
 58  errors_home                           32434 non-null  int64
```

```
 59  passed_balls_home                   32434 non-null  int64
 60  double_def_home                     32434 non-null  int64
 61  triple_def_home                     32434 non-null  int64
 62  year                                32434 non-null  int32
 63  losing_team                         32434 non-null  object
 64  winning_team                        32434 non-null  object
 65  run_differential                    32434 non-null  int64
 66  total_runs                          32434 non-null  int64
 67  obp_home                            32434 non-null  float64
 68  obp_visiting                        32434 non-null  float64
 69  ID                                  32434 non-null  object
 70  win                                 32434 non-null  int64
dtypes: float64(5), int32(1), int64(59), object(6)
memory usage: 17.4+ MB
```

In [56]:
```
team_visiting_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32434 entries, 0 to 32433
Data columns (total 71 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   date                                  32434 non-null  object
 1   team_visiting                         32434 non-null  object
 2   team_home                             32434 non-null  object
 3   score_visiting                        32434 non-null  int64
 4   score_home                            32434 non-null  int64
 5   outs_in_game                          32434 non-null  int64
 6   at_bats_visiting                      32434 non-null  float64
 7   hits_visiting                         32434 non-null  int64
 8   double_visiting                       32434 non-null  float64
 9   triple_visiting                       32434 non-null  float64
 10  home_run_visiting                     32434 non-null  int64
 11  rbi_visiting                          32434 non-null  int64
 12  sacrifice_hit_visiting                32434 non-null  int64
 13  sacrifice_fly_visiting                32434 non-null  int64
 14  hit_by_pitch_visiting                 32434 non-null  int64
 15  walk_visiting                         32434 non-null  int64
 16  intent_walk_visiting                  32434 non-null  int64
 17  strikeout_visiting                    32434 non-null  int64
 18  stolen_base_visiting                  32434 non-null  int64
 19  caught_stealing_visiting              32434 non-null  int64
 20  grounded_into_double_plays_visiting   32434 non-null  int64
 21  first_catcher_interfere_visiting      32434 non-null  int64
 22  left_on_base_visiting                 32434 non-null  int64
 23  pitchers_used_visiting                32434 non-null  int64
 24  individual_earned_runs_visiting       32434 non-null  int64
 25  team_earned_runs_visiting             32434 non-null  int64
 26  wild_pitches_visiting                 32434 non-null  int64
 27  balks_visiting                        32434 non-null  int64
 28  putouts_visiting                      32434 non-null  int64
 29  assists_visiting                      32434 non-null  int64
 30  errors_visiting                       32434 non-null  int64
 31  passed_balls_visiting                 32434 non-null  int64
 32  double_def_visiting                   32434 non-null  int64
 33  triple_def_visiting                   32434 non-null  int64
 34  at_bats_home                          32434 non-null  int64
 35  hits_home                             32434 non-null  int64
 36  double_home                           32434 non-null  int64
 37  triple_home                           32434 non-null  int64
 38  home_run_home                         32434 non-null  int64
 39  rbi_home                              32434 non-null  int64
 40  sacrifice_hit_home                    32434 non-null  int64
 41  sacrifice_fly_home                    32434 non-null  int64
 42  hit_by_pitch_home                     32434 non-null  int64
 43  walk_home                             32434 non-null  int64
 44  intent_walk_home                      32434 non-null  int64
 45  strikeout_home                        32434 non-null  int64
 46  stolen_base_home                      32434 non-null  int64
 47  caught_stealing_home                  32434 non-null  int64
 48  grounded_into_double_plays_home       32434 non-null  int64
 49  first_catcher_interfere_home          32434 non-null  int64
 50  left_on_base_home                     32434 non-null  int64
 51  pitchers_used_home                    32434 non-null  int64
 52  individual_earned_runs_home           32434 non-null  int64
 53  team_earned_runs_home                 32434 non-null  int64
 54  wild_pitches_home                     32434 non-null  int64
 55  balks_home                            32434 non-null  int64
 56  putouts_home                          32434 non-null  int64
 57  assists_home                          32434 non-null  int64
 58  errors_home                           32434 non-null  int64
```

```
 59  passed_balls_home                   32434 non-null  int64
 60  double_def_home                     32434 non-null  int64
 61  triple_def_home                     32434 non-null  int64
 62  year                                32434 non-null  int32
 63  losing_team                         32434 non-null  object
 64  winning_team                        32434 non-null  object
 65  run_differential                    32434 non-null  int64
 66  total_runs                          32434 non-null  int64
 67  obp_home                            32434 non-null  float64
 68  obp_visiting                        32434 non-null  float64
 69  ID                                  32434 non-null  object
 70  win                                 32434 non-null  int64
dtypes: float64(5), int32(1), int64(59), object(6)
memory usage: 17.4+ MB
```

In [57]:
```
#team_home_df.to_csv('data/Home_DF.csv')
```

In [58]:
```
#team_visiting_df.to_csv('data/Visiting_DF.csv')
```

In [59]:
```
# When I stacked the dataframes on top of one another they all had to have the same feature na
# The _home and _visiting were removed from the features.

team_home_df.drop(columns=["at_bats_visiting", "hits_visiting", "double_visiting", "triple_vi
            "sacrifice_hit_visiting", 'sacrifice_fly_visiting', 'hit_by_pitch_visiting', 'wal
            'strikeout_visiting', 'stolen_base_visiting', 'caught_stealing_visiting', 'ground
            'first_catcher_interfere_visiting', 'left_on_base_visiting', 'pitchers_used_visit
            'team_earned_runs_visiting', 'wild_pitches_visiting', 'balks_visiting', 'putouts_
            'errors_visiting', 'passed_balls_visiting', 'double_def_visiting', 'triple_def_vi

team_visiting_df.drop(columns=["at_bats_home", "hits_home", "double_home", "triple_home", "ho
            "sacrifice_hit_home", 'sacrifice_fly_home', 'hit_by_pitch_home', 'walk_home', 'in
            'strikeout_home', 'stolen_base_home', 'caught_stealing_home', 'grounded_into_doub
            'first_catcher_interfere_home', 'left_on_base_home', 'pitchers_used_home', 'indiv
            'team_earned_runs_home', 'wild_pitches_home', 'balks_home', 'putouts_home', 'assi
            'errors_home', 'passed_balls_home', 'double_def_home', 'triple_def_home', 'obp_ho
```

In [60]:
```
rename_dict_home = {

            'at_bats_home': 'at_bats',
            'hits_home': 'hits',
            'double_home': 'double',
            'triple_home': 'triple',
            'home_run_home': 'home_run',
            'rbi_home': 'rbi',
            'sacrifice_hit_home': 'sacrifice_hit',
            'sacrifice_fly_home': 'sacrifice_fly',
            'hit_by_pitch_home': 'hit_by_pitch',
            'walk_home': 'walk',
            'intent_walk_home': 'intent_walk',
            'strikeout_home': "strikeout",
            'stolen_base_home': 'stolen_base',
            'caught_stealing_home': 'caught_stealing',
            'grounded_into_double_plays_home': 'grounded_into_double_plays',
            'first_catcher_interfere_home': 'first_catcher_interfere',
            'left_on_base_home': 'left_on_base',
            'pitchers_used_home': 'pitchers_used',
            'individual_earned_runs_home': 'individual_earned_runs',
            'team_earned_runs_home': 'team_earned_runs',
            'wild_pitches_home': 'wild_pitches',
            'balks_home': 'balks',
            'putouts_home': 'putouts',
            'assists_home': 'assists',
            'errors_home': 'errors',
            'passed_balls_home': 'passed_balls',
            'double_def_home': 'double_def',
            'triple_def_home': 'triple_def',
            'obp_home': 'obp'
}

# Rename the columns using the rename() function
team_home_df.rename(columns=rename_dict_home, inplace=True)
```

In [61]:
```
rename_dict_visiting = {

            "at_bats_visiting": "at_bats",
            "hits_visiting": "hits",
            "double_visiting": "double",
            "triple_visiting": "triple",
            "home_run_visiting": "home_run",
            "rbi_visiting": "rbi",
            "sacrifice_hit_visiting": "sacrifice_hit",
            'sacrifice_fly_visiting': "sacrifice_fly",
            'hit_by_pitch_visiting': "hit_by_pitch",
            'walk_visiting': 'walk',
            'intent_walk_visiting': 'intent_walk',
            'strikeout_visiting': 'strikeout',
            'stolen_base_visiting': 'stolen_base',
            'caught_stealing_visiting': 'caught_stealing',
            'grounded_into_double_plays_visiting': 'grounded_into_double_plays',
            'first_catcher_interfere_visiting': 'first_catcher_interfere',
            'left_on_base_visiting': 'left_on_base',
            'pitchers_used_visiting': "pitchers_used",
            'individual_earned_runs_visiting': "individual_earned_runs",
            'team_earned_runs_visiting': 'team_earned_runs',
            'wild_pitches_visiting': 'wild_pitches',
            'balks_visiting': 'balks',
            'putouts_visiting': 'putouts',
            'assists_visiting': 'assists',
            'errors_visiting': 'errors',
            'passed_balls_visiting': 'passed_balls',
            'double_def_visiting': 'double_def',
            'triple_def_visiting': 'triple_def',
            'obp_visiting': 'obp'
}

# Rename the columns using the rename() function
team_visiting_df.rename(columns=rename_dict_visiting, inplace=True)
```

In [62]:
```
combo_df = pd.concat([team_home_df, team_visiting_df], ignore_index=True)
```

In [63]:
```
# this is so that when I train test split I can do an exact match on ID to ensure the two tea
combo_df['ID'] = combo_df['ID'].str[:-2]
```

In [64]:
```
combo_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 64868 entries, 0 to 64867
Data columns (total 41 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   date                         64868 non-null  object
 1   team_visiting                64868 non-null  object
 2   team_home                    64868 non-null  object
 3   score_visiting               64868 non-null  int64
 4   score_home                   64868 non-null  int64
 5   outs_in_game                 64868 non-null  int64
 6   at_bats                      64868 non-null  float64
 7   hits                         64868 non-null  int64
 8   double                       64868 non-null  float64
 9   triple                       64868 non-null  float64
 10  home_run                     64868 non-null  int64
 11  rbi                          64868 non-null  int64
 12  sacrifice_hit                64868 non-null  int64
 13  sacrifice_fly                64868 non-null  int64
 14  hit_by_pitch                 64868 non-null  int64
 15  walk                         64868 non-null  int64
 16  intent_walk                  64868 non-null  int64
 17  strikeout                    64868 non-null  int64
 18  stolen_base                  64868 non-null  int64
 19  caught_stealing              64868 non-null  int64
 20  grounded_into_double_plays   64868 non-null  int64
 21  first_catcher_interfere      64868 non-null  int64
 22  left_on_base                 64868 non-null  int64
 23  pitchers_used                64868 non-null  int64
 24  individual_earned_runs       64868 non-null  int64
 25  team_earned_runs             64868 non-null  int64
 26  wild_pitches                 64868 non-null  int64
 27  balks                        64868 non-null  int64
 28  putouts                      64868 non-null  int64
 29  assists                      64868 non-null  int64
 30  errors                       64868 non-null  int64
 31  passed_balls                 64868 non-null  int64
 32  double_def                   64868 non-null  int64
 33  triple_def                   64868 non-null  int64
 34  year                         64868 non-null  int32
 35  losing_team                  64868 non-null  object
 36  winning_team                 64868 non-null  object
 37  run_differential             64868 non-null  int64
 38  obp                          64868 non-null  float64
 39  ID                           64868 non-null  object
 40  win                          64868 non-null  int64
dtypes: float64(4), int32(1), int64(30), object(6)
memory usage: 20.0+ MB
```

In [65]:
```
combo_df.head()
```

Out[65]:

	date	team_visiting	team_home	score_visiting	score_home	outs_in_game	at_bats	hits	double	triple	hc
0	20100404	NYA	BOS	7	9	51	34.0	12	3.0	1.0	
1	20100405	MIN	ANA	3	6	51	33.0	9	0.0	0.0	
2	20100405	CLE	CHA	0	6	51	28.0	6	2.0	0.0	
3	20100405	TOR	TEX	4	5	52	31.0	6	2.0	0.0	
4	20100405	SDN	ARI	3	6	51	33.0	8	0.0	0.0	

In [66]:
```
combo_df.to_csv('data/combo_df.csv')
```

Exploratory Data Analysis

This section runs through a set of features looking at counts, descriptive statistics, crosstabs, and histograms.

In [67]:
```
# This function runs through a list of features of interest and provides the value counts of e
def get_value_counts(combo_df, features):
    value_counts = {}
    for feature in features:
        value_counts[feature] = combo_df[feature].value_counts()
    return value_counts

# List of features to get value counts for
features_to_check = ['at_bats',
   'hits',
   'double',
   'triple',
   'home_run',
   'sacrifice_hit',
   'sacrifice_fly',
   'hit_by_pitch',
   'walk',
   'intent_walk',
   'strikeout',
   'stolen_base',
   'caught_stealing',
   'grounded_into_double_plays',
   'first_catcher_interfere',
   'left_on_base',
   'pitchers_used',
   'wild_pitches',
   'balks',
   'assists',
   'errors',
   'passed_balls',
   'double_def',
   'triple_def'
      ]

# Get the value counts
value_counts = get_value_counts(combo_df, features_to_check)

# Print the value counts
for feature, counts in value_counts.items():
    print(f"\nValue counts for '{feature}':")
    print(counts)
```

```
Value counts for 'at_bats':
33.0    7448
32.0    7213
34.0    6961
31.0    6336
35.0    5872
30.0    4997
36.0    4779
37.0    3808
29.0    3345
38.0    2798
39.0    2045
28.0    1946
40.0    1500
41.0    1025
27.0     922
42.0     774
43.0     582
```

In [68]:
```python
# This function runs through a list of features of interest and provides the descriptive stat
def get_descriptive_statistics(df, features):
    descriptive_stats = {}
    for feature in features:
        descriptive_stats[feature] = df[feature].describe()
    return descriptive_stats

# List of features to get descriptive statistics for
features_to_check = ['at_bats',
   'hits',
   'double',
   'triple',
   'home_run',
   'sacrifice_hit',
   'sacrifice_fly',
   'hit_by_pitch',
   'walk',
   'intent_walk',
   'strikeout',
   'stolen_base',
   'caught_stealing',
   'grounded_into_double_plays',
   'first_catcher_interfere',
   'left_on_base',
   'pitchers_used',
   'wild_pitches',
   'balks',
   'assists',
   'errors',
   'passed_balls',
   'double_def',
   'triple_def',
    'obp']

# Get the descriptive statistics
descriptive_stats = get_descriptive_statistics(combo_df, features_to_check)

# Print the descriptive statistics
for feature, stats in descriptive_stats.items():
    print(f"\nDescriptive statistics for '{feature}':")
    print(stats)
```

```
Descriptive statistics for 'at_bats':
count    64868.000000
mean        33.946353
std          4.271191
min         16.000000
25%         31.000000
50%         33.000000
75%         36.000000
max         76.000000
Name: at_bats, dtype: float64

Descriptive statistics for 'hits':
count    64868.000000
mean         8.537430
std          3.431466
min          0.000000
25%          6.000000
50%          8.000000
```

In [69]:
```
combo_df['run_differential'].describe()
```

Out[69]:
```
count    64868.000000
mean         3.454770
std          2.667129
min          1.000000
25%          1.000000
50%          3.000000
75%          5.000000
max         24.000000
Name: run_differential, dtype: float64
```

In [70]:
```
combo_df['obp'].describe()
```

Out[70]:
```
count    64868.000000
mean         0.312017
std          0.082735
min          0.000000
25%          0.257143
50%          0.314286
75%          0.370968
max          0.603448
Name: obp, dtype: float64
```

In [71]:
```
combo_df['at_bats'].describe()
```

Out[71]:
```
count    64868.000000
mean        33.946353
std          4.271191
min         16.000000
25%         31.000000
50%         33.000000
75%         36.000000
max         76.000000
Name: at_bats, dtype: float64
```

In [72]:
```
combo_df['walk'].describe()
```

Out[72]:
```
count    64868.000000
mean         3.128569
std          2.019414
min          0.000000
25%          2.000000
50%          3.000000
75%          4.000000
max         18.000000
Name: walk, dtype: float64
```

In [73]:
```
combo_df['left_on_base'].describe()
```

Out[73]:
```
count    64868.000000
mean         6.814932
std          2.627345
min          0.000000
25%          5.000000
50%          7.000000
75%          8.000000
max         25.000000
Name: left_on_base, dtype: float64
```

In [74]:
```
combo_df['home_run'].describe()
```

Out[74]:
```
count    64868.000000
mean         1.097290
std          1.122336
min          0.000000
25%          0.000000
50%          1.000000
75%          2.000000
max          8.000000
Name: home_run, dtype: float64
```

In [75]:
```
combo_df['pitchers_used'].describe()
```

Out[75]:
```
count    64868.000000
mean         4.149935
std          1.377971
min          1.000000
25%          3.000000
50%          4.000000
75%          5.000000
max         13.000000
Name: pitchers_used, dtype: float64
```

In [76]:
```
combo_df['double'].describe()
```

Out[76]:
```
count    64868.000000
mean         1.693716
std          1.369302
min          0.000000
25%          1.000000
50%          1.000000
75%          2.000000
max         11.000000
Name: double, dtype: float64
```

In [77]:
```
combo_df['strikeout'].describe()
```

Out[77]:
```
count    64868.000000
mean         8.008124
std          2.987852
min          0.000000
25%          6.000000
50%          8.000000
75%         10.000000
max         26.000000
Name: strikeout, dtype: float64
```

In [78]:
```
combo_df['intent_walk'].describe()
```

Out[78]:
```
count    64868.000000
mean         0.182833
std          0.462936
min          0.000000
25%          0.000000
50%          0.000000
75%          0.000000
max          8.000000
Name: intent_walk, dtype: float64
```

In [79]:
```
cross_tab = pd.crosstab(combo_df['run_differential'], combo_df['year'])
print(cross_tab)
```

```
year              2010  2011  2012  2013  2014  2015  2016  2017  2018  2019  2020  2021  20
22  2023
run_differential
1                 1458  1502  1394  1506  1472  1410  1368  1292  1334  1288   494  1352  14
00  1346
2                  894   878   912   908   898   940   886   846   876   868   290   886   9
04   846
3                  596   708   692   662   694   696   728   686   728   694   262   694   6
88   716
4                  570   590   638   554   544   536   552   570   560   556   216   564   5
64   620
5                  400   362   398   416   434   372   404   414   418   430   176   398   3
62   406
6                  352   236   358   258   298   302   270   310   302   288   108   288   3
14   264
7                  168   216   164   194   166   206   242   236   204   236    70   220   1
98   192
8                  154   118   110   122   132   148   136   176   146   162    42   136   1
34   122
9                   90   100    70    90    82    72    90   104   110   108    44   102   1
08   124
10                  62    56    46    66    58    56    70    90    50    72    20    72
76    70
11                  40    30    28    34    32    48    48    52    46    44    22    34
30    50
12                  34    24    26    16    16    14    14    38    20    22    16    34
28    22
13                  12    14     8    18     8    26    22    16    22    32    14    22
28    36
14                  10    12     0     2     6    12     6    10    12    26     6    20
10    12
15                   4     2     6     8     2     0     2    10    10    12     4     8
2     10
16                   0     4     4     4     2     8     4     4    10     6     0     2
6      2
17                   2     0     2     0     2     2     2     2     2     2     2     4
2      4
18                   2     2     2     0     0     0     0     4     2     0     2     2
0      2
19                   0     0     0     0     0     0     2     0     2     0     2     2
0      2
20                   2     0     0     0     0     0     0     0     2     2     2     0
0      0
21                   0     0     0     0     0     0     0     0     2     2     0     2
2      0
23                   0     0     0     0     0     0     0     0     0     0     0     0
2      0
24                   0     0     0     0     0     0     0     0     0     0     0     0
0      2
```

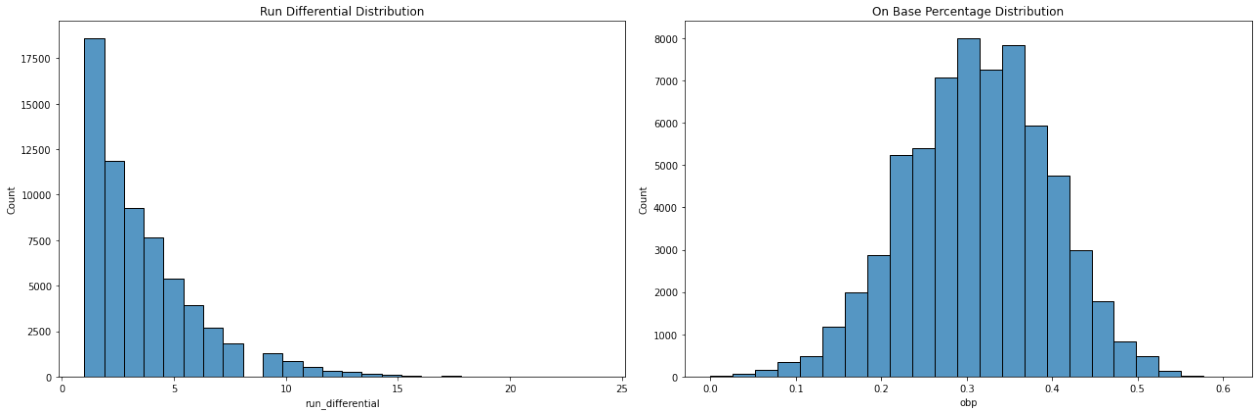
In [80]:
```
cross_tab2 = pd.crosstab(combo_df['strikeout'], combo_df['year'])
print(cross_tab2)
```

year	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	202
3														
strikeout														
0	8	7	7	6	5	3	2	3	0	0	0	0	0	
4														
1	43	41	19	22	28	25	15	19	7	8	5	4	11	1
4														
2	140	134	81	80	76	86	88	60	44	25	8	30	43	3
4														
3	291	254	225	217	174	168	138	105	115	108	43	78	100	9
5														
4	424	444	331	366	327	314	299	242	230	173	75	209	225	22
0														
5	583	594	537	494	499	508	436	424	376	335	127	347	398	35
0														
6	717	664	681	687	637	659	562	561	492	466	183	481	543	47
8														
7	634	693	680	672	696	690	661	646	630	607	208	614	631	60
0														
8	578	616	681	648	633	629	636	674	648	629	246	649	667	67
7														
9	506	487	515	542	554	548	580	578	611	624	232	602	623	59
5														
10	369	361	397	394	428	440	473	490	541	570	182	563	514	51
1														
11	221	234	288	299	296	300	346	407	460	434	165	447	388	47
1														
12	155	145	182	177	203	183	281	263	281	316	124	357	288	31
8														
13	87	80	106	107	123	116	136	161	187	208	83	194	171	21
6														
14	47	44	58	55	79	77	81	105	97	147	53	120	127	12
0														
15	27	24	40	36	41	42	53	60	60	93	30	78	68	6
1														
16	10	13	17	28	24	39	27	23	40	45	22	37	43	4
5														
17	6	10	7	10	16	5	13	24	16	29	3	19	12	2
8														
18	1	4	3	7	6	10	11	4	12	9	2	12	2	
9														
19	2	4	1	7	1	6	4	5	4	9	1	1	2	
2														
20	1	1	1	3	0	0	2	1	6	5	0	0	2	
0														
21	0	0	1	1	0	0	0	0	0	0	0	0	0	
0														
22	0	0	0	0	0	0	1	1	0	2	0	0	0	
0														
23	0	0	0	0	0	0	1	1	0	2	0	0	0	
0														
24	0	0	0	0	0	0	0	1	1	4	0	0	0	
0														
25	0	0	0	0	0	0	0	0	0	1	0	0	0	
0														
26	0	0	0	0	0	0	0	2	0	1	0	0	0	
0														

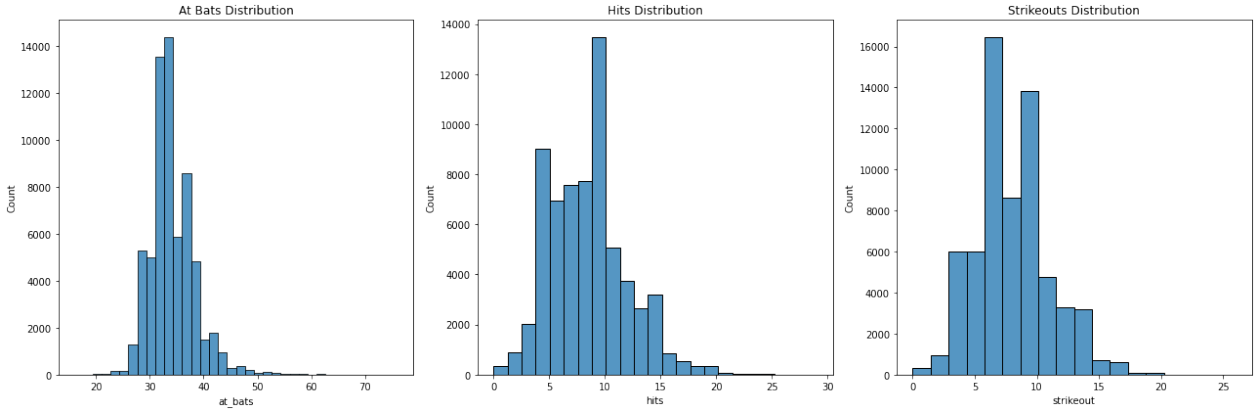
In [81]:
```
mean_obp = combo_df.groupby('win')['obp'].mean().reset_index()
median_obp = combo_df.groupby('win')['obp'].median().reset_index()
print(f"\nMean {mean_obp}")
print(f"\nMedian {median_obp}")
```

```
Mean    win       obp
0    0  0.271706
1    1  0.352328

Median    win       obp
0    0  0.272727
1    1  0.352941
```

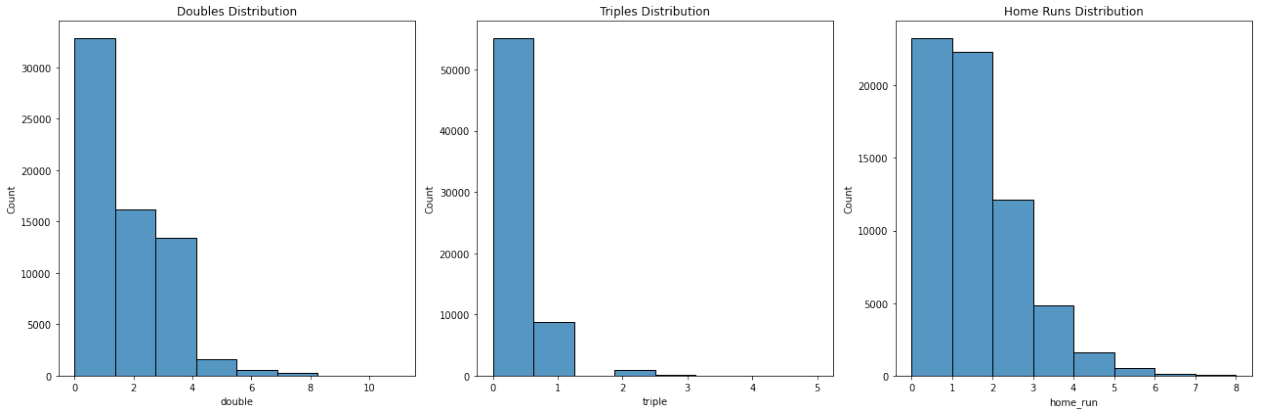
In [82]:
```
fig, axes = plt.subplots(1, 2, figsize=(18, 6))

# Plot 1:
sns.histplot(combo_df['run_differential'], bins=26, ax=axes[0])
axes[0].set_title('Run Differential Distribution')

# Plot 2:
sns.histplot(combo_df['obp'], bins=23, ax=axes[1])
axes[1].set_title('On Base Percentage Distribution')

# Adjust layout
plt.tight_layout()

# Display the plots
plt.show()
```



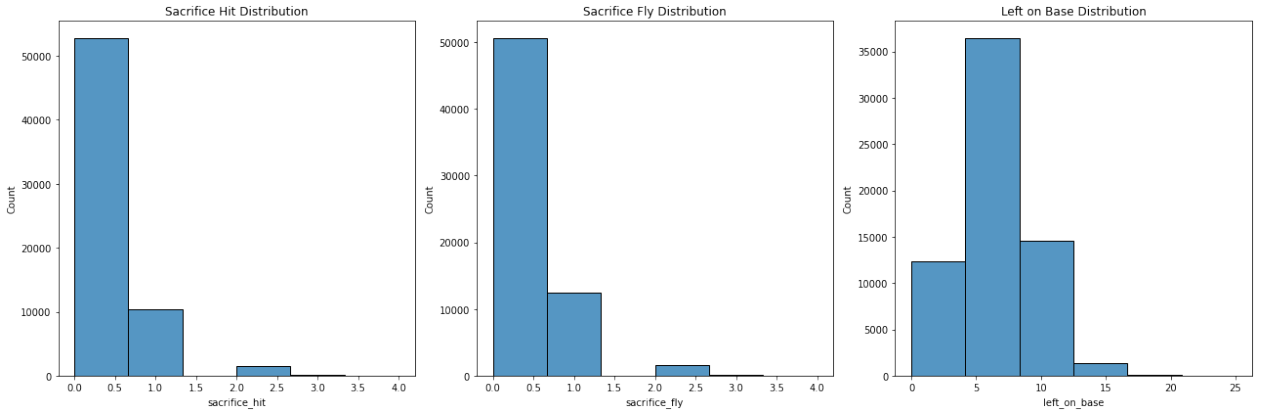
In [83]:
```
fig, axes = plt.subplots(1, 3, figsize=(18, 6))

# Plot 1:
sns.histplot(combo_df['at_bats'], bins=36, ax=axes[0])
axes[0].set_title('At Bats Distribution')

# Plot 2:
sns.histplot(combo_df['hits'], bins=23, ax=axes[1])
axes[1].set_title('Hits Distribution')

# Plot 3:
sns.histplot(combo_df['strikeout'], bins=18, ax=axes[2])
axes[2].set_title('Strikeouts Distribution')

# Adjust layout
plt.tight_layout()

# Display the plots
plt.show()
```



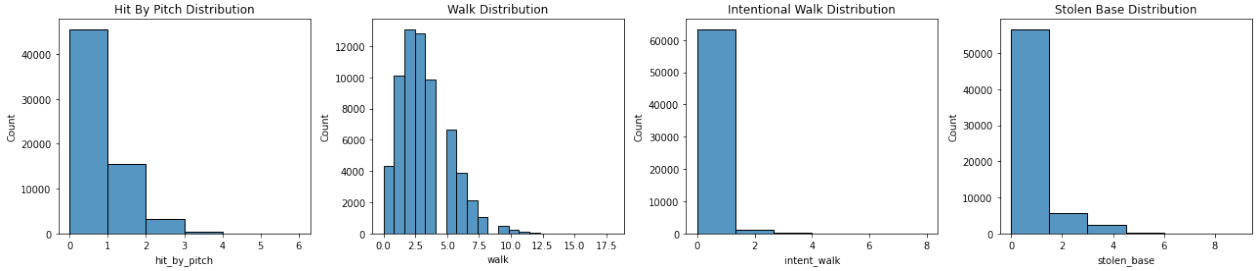
In [84]:
```
#NOTE: 0 is included in these figures
fig, axes = plt.subplots(1, 3, figsize=(18, 6))

# Plot 1:
sns.histplot(combo_df['double'], bins=8, ax=axes[0])
axes[0].set_title('Doubles Distribution')

# Plot 2:
sns.histplot(combo_df['triple'], bins=8, ax=axes[1])
axes[1].set_title('Triples Distribution')

# Plot 3:
sns.histplot(combo_df['home_run'], bins=8, ax=axes[2])
axes[2].set_title('Home Runs Distribution')

# Adjust layout
plt.tight_layout()

# Display the plots
plt.show()
```



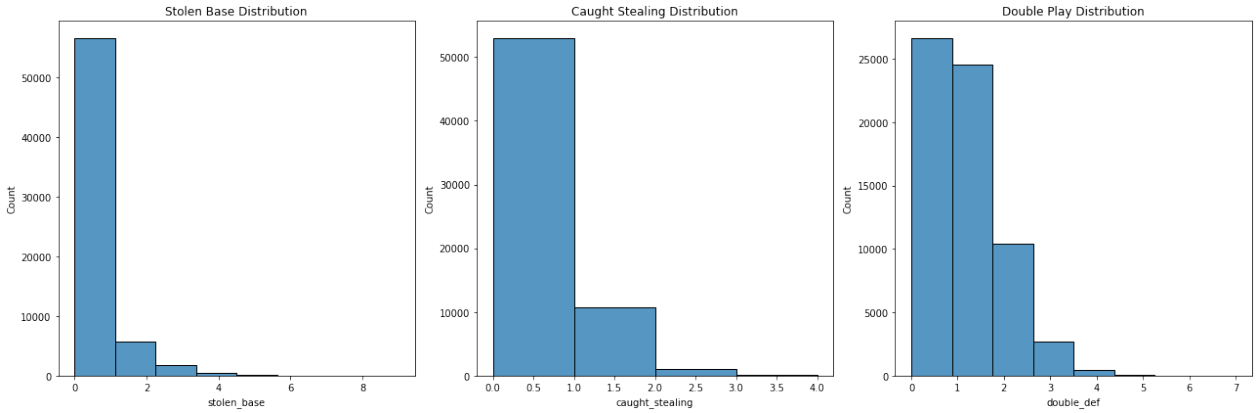
In [85]:
```
#NOTE: 0 is included in these figures
fig, axes = plt.subplots(1, 3, figsize=(18, 6))

# Plot 1:
sns.histplot(combo_df['sacrifice_hit'], bins=6, ax=axes[0])
axes[0].set_title('Sacrifice Hit Distribution')

# Plot 2:
sns.histplot(combo_df['sacrifice_fly'], bins=6, ax=axes[1])
axes[1].set_title('Sacrifice Fly Distribution')

# Plot 3:
sns.histplot(combo_df['left_on_base'], bins=6, ax=axes[2])
axes[2].set_title('Left on Base Distribution')

# Adjust layout
plt.tight_layout()

# Display the plots
plt.show()
```



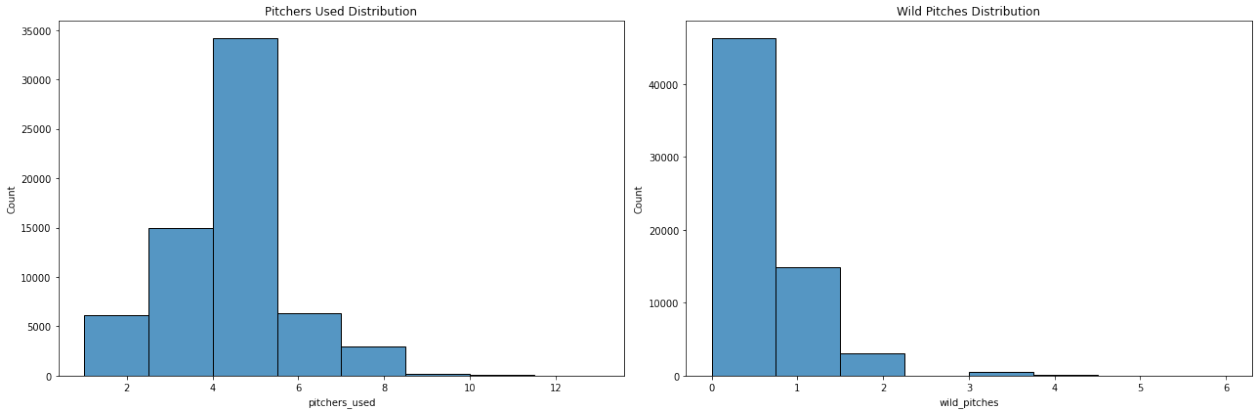
In [86]:
```
#NOTE: 0 is included in these figures
fig, axes = plt.subplots(1, 4, figsize=(18, 4))

# Plot 1:
sns.histplot(combo_df['hit_by_pitch'], bins=6, ax=axes[0])
axes[0].set_title('Hit By Pitch Distribution')

# Plot 2:
sns.histplot(combo_df['walk'], bins=22, ax=axes[1])
axes[1].set_title('Walk Distribution')

# Plot 3:
sns.histplot(combo_df['intent_walk'], bins=6, ax=axes[2])
axes[2].set_title('Intentional Walk Distribution')

# Plot 4:
sns.histplot(combo_df['stolen_base'], bins=6, ax=axes[3])
axes[3].set_title('Stolen Base Distribution')

# Adjust layout
plt.tight_layout()

# Display the plots
plt.show()
```



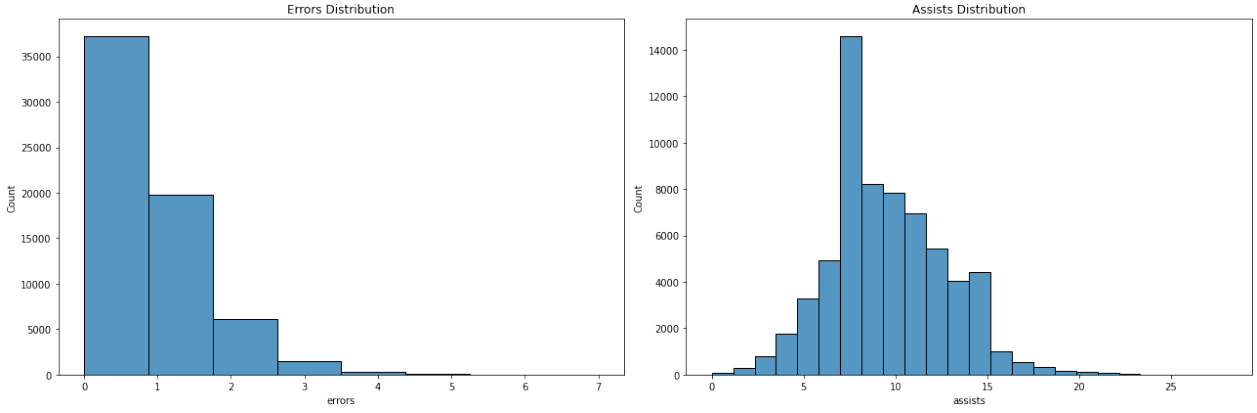
In [87]:
```
#NOTE: 0 is included in these figures
fig, axes = plt.subplots(1, 3, figsize=(18, 6))

# Plot 1:
sns.histplot(combo_df['stolen_base'], bins=8, ax=axes[0])
axes[0].set_title('Stolen Base Distribution')

# Plot 2:
sns.histplot(combo_df['caught_stealing'], bins=4, ax=axes[1])
axes[1].set_title('Caught Stealing Distribution')

# Plot 3:
sns.histplot(combo_df['double_def'], bins=8, ax=axes[2])
axes[2].set_title('Double Play Distribution')

# Adjust layout
plt.tight_layout()

# Display the plots
plt.show()
```



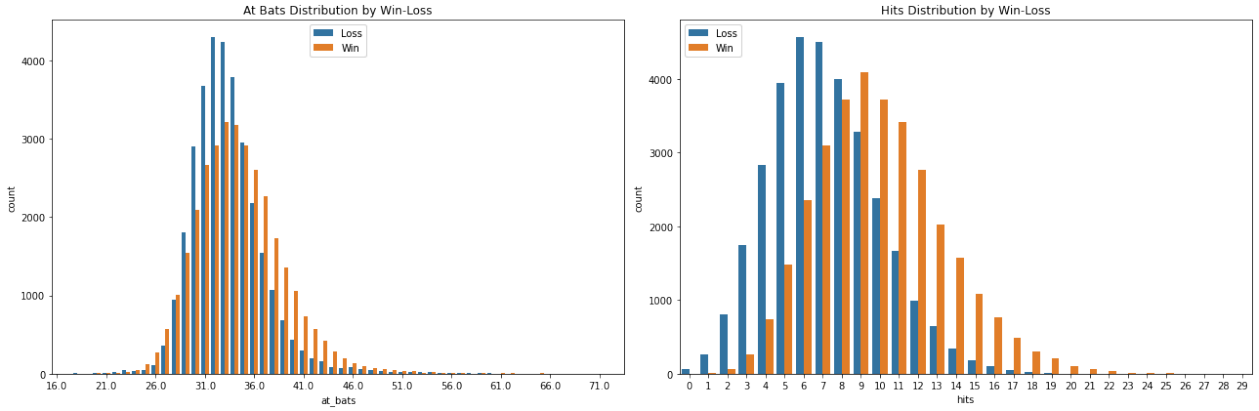
In [88]:
```
#NOTE: 0 is included in the Wild Pitches figure
fig, axes = plt.subplots(1, 2, figsize=(18, 6))

# Plot 1:
sns.histplot(combo_df['pitchers_used'], bins=8, ax=axes[0])
axes[0].set_title('Pitchers Used Distribution')

# Plot 2:
sns.histplot(combo_df['wild_pitches'], bins=8, ax=axes[1])
axes[1].set_title('Wild Pitches Distribution')

# Adjust layout
plt.tight_layout()

# Display the plots
plt.show()
```



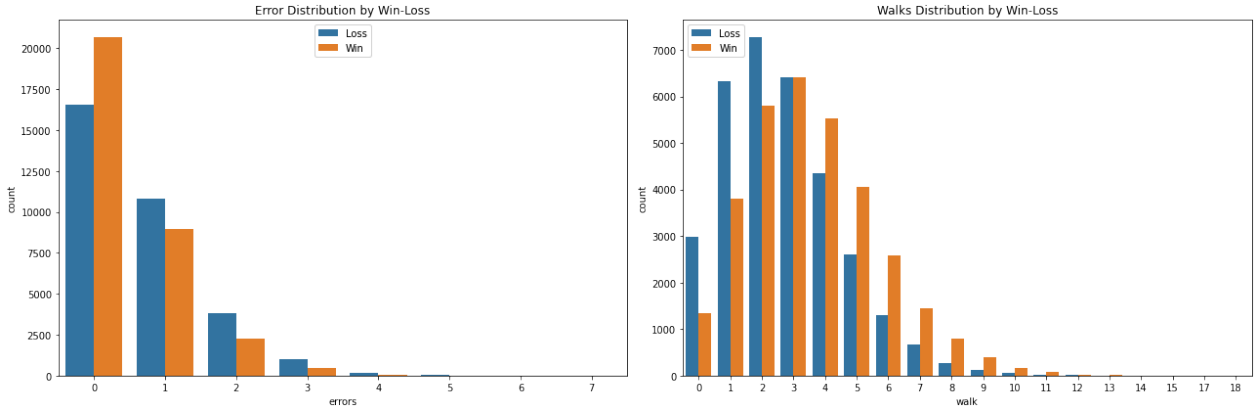
In [89]:
```
fig, axes = plt.subplots(1, 2, figsize=(18, 6))

# Plot 1:
sns.histplot(combo_df['errors'], bins=8, ax=axes[0])
axes[0].set_title('Errors Distribution')

# Plot 2:
sns.histplot(combo_df['assists'], bins=24, ax=axes[1])
axes[1].set_title('Assists Distribution')

# Adjust layout
plt.tight_layout()

# Display the plots
plt.show()
```



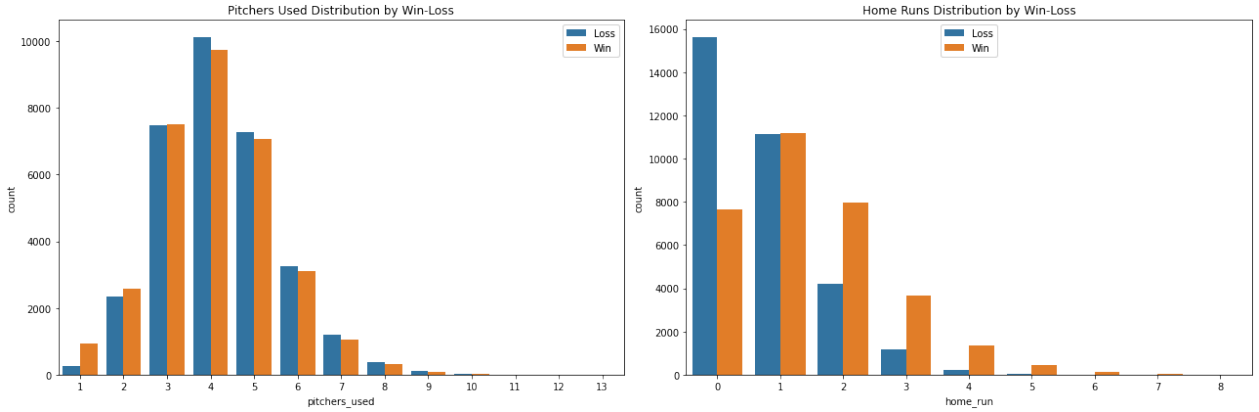
In [90]:
```
fig, axes = plt.subplots(1, 2, figsize=(18, 6))

# Plot 1:
sns.countplot(x='at_bats', hue='win', data=combo_df, ax=axes[0])
axes[0].set_title('At Bats Distribution by Win-Loss')
handles, labels = axes[0].get_legend_handles_labels()
axes[0].legend(handles, ['Loss', 'Win'])

ticks = axes[0].get_xticks()
axes[0].set_xticks(ticks[::5])

# Plot 2:
sns.countplot(x='hits', hue='win', data=combo_df, ax=axes[1])
axes[1].set_title('Hits Distribution by Win-Loss')
handles, labels = axes[1].get_legend_handles_labels()
axes[1].legend(handles, ['Loss', 'Win'])

# Adjust layout
plt.tight_layout()

# Display the plots
plt.show()
```



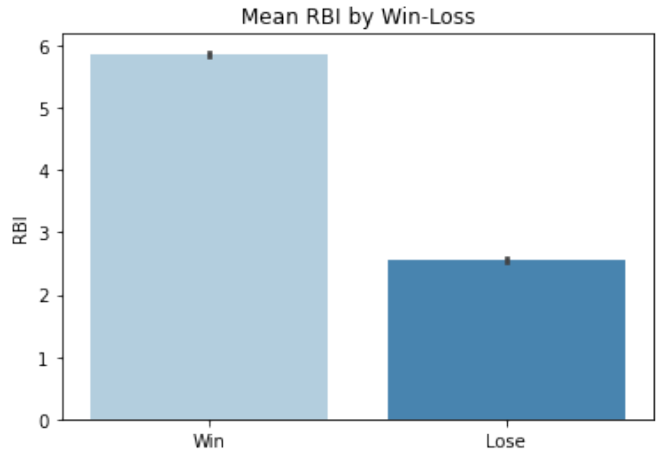
In [91]:
```
fig, axes = plt.subplots(1, 2, figsize=(18, 6))

# Plot 1:
sns.countplot(x='errors', hue='win', data=combo_df, ax=axes[0])
axes[0].set_title('Error Distribution by Win-Loss')
handles, labels = axes[0].get_legend_handles_labels()
axes[0].legend(handles, ['Loss', 'Win'])

# Plot 2:
sns.countplot(x='walk', hue='win', data=combo_df, ax=axes[1])
axes[1].set_title('Walks Distribution by Win-Loss')
handles, labels = axes[1].get_legend_handles_labels()
axes[1].legend(handles, ['Loss', 'Win'])

# Adjust layout
plt.tight_layout()

# Display the plots
plt.show()
```



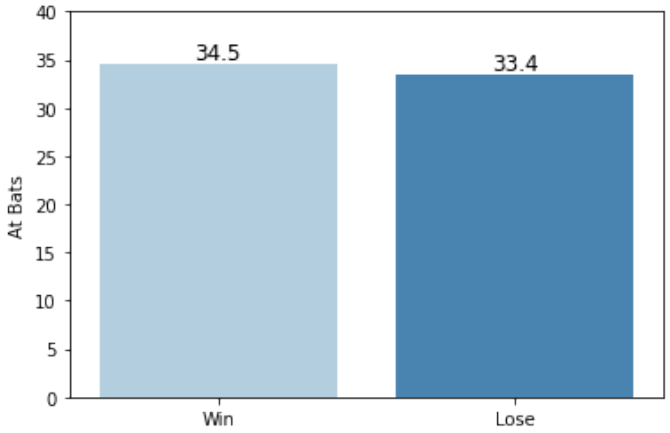
In [92]:
```
fig, axes = plt.subplots(1, 2, figsize=(18, 6))

# Plot 1:
sns.countplot(x='pitchers_used', hue='win', data=combo_df, ax=axes[0])
axes[0].set_title('Pitchers Used Distribution by Win-Loss')
handles, labels = axes[0].get_legend_handles_labels()
axes[0].legend(handles, ['Loss', 'Win'])

# Plot 2:
sns.countplot(x='home_run', hue='win', data=combo_df, ax=axes[1])
axes[1].set_title('Home Runs Distribution by Win-Loss')
handles, labels = axes[1].get_legend_handles_labels()
axes[1].legend(handles, ['Loss', 'Win'])

# Adjust layout
plt.tight_layout()

# Display the plots
plt.show()
```



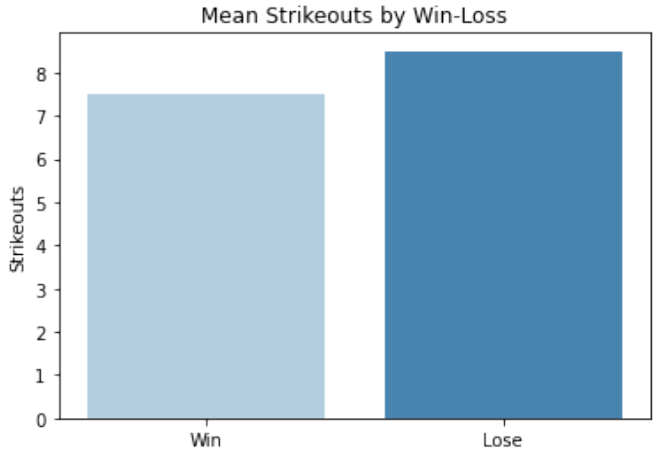
In [93]:
```
combo_df['win'] = combo_df['win'].map({0: 'Lose', 1: 'Win'})
sns.barplot(x='win', y='rbi', data=combo_df, palette='Blues')

# Add labels and title
plt.xlabel('')
plt.ylabel('RBI')
plt.title('Mean RBI by Win-Loss')

# Show plot
plt.show()
```



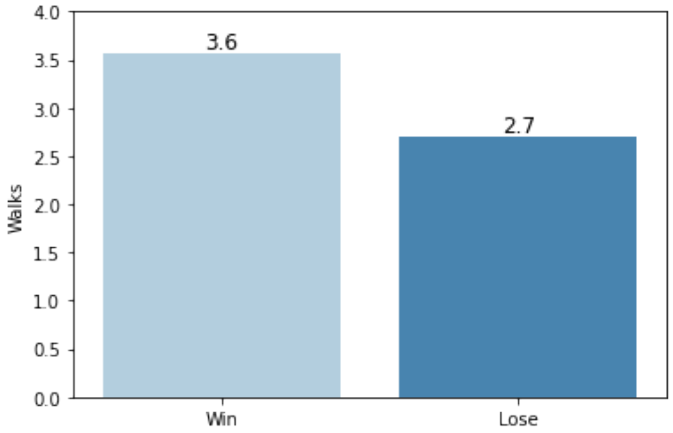
In [94]:
```
ax=sns.barplot(x='win', y='at_bats', data=combo_df, ci=None, palette="Blues")

# Add labels and title
plt.xlabel('')
plt.ylabel('At Bats')
plt.title('')

for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height:.1f}',
                (p.get_x() + p.get_width() / 2., height),
                ha='center', va='bottom',
                fontsize=12, color='black')

ax.set_ylim(0, 40)

# Show plot
plt.show()
```



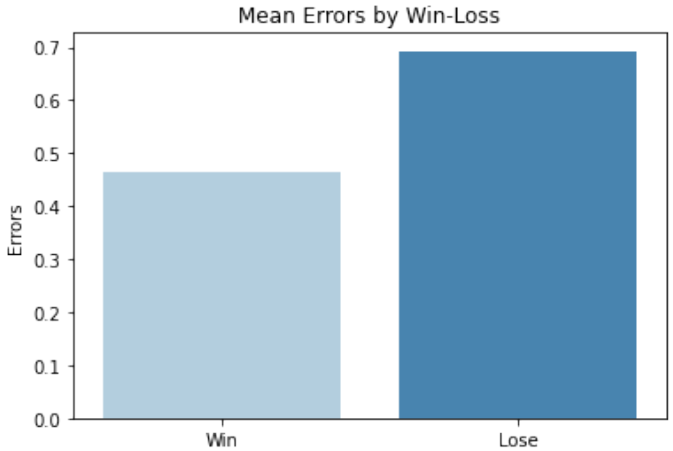
In [95]:
```
sns.barplot(x='win', y='strikeout', data=combo_df, ci=None, palette='Blues')

# Add labels and title
plt.xlabel('')
plt.ylabel('Strikeouts')
plt.title('Mean Strikeouts by Win-Loss')

# Show plot
plt.show()
```



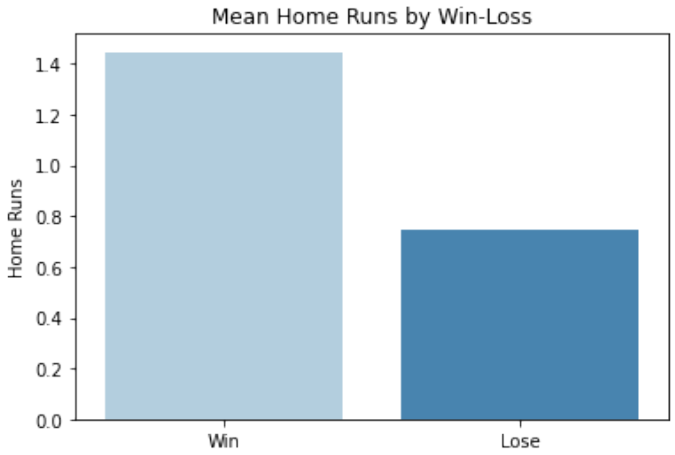
In [96]:
```
ax=sns.barplot(x='win', y='walk', data=combo_df, ci=None, palette='Blues')

# Add labels and title
plt.xlabel('')
plt.ylabel('Walks')
plt.title('')

for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height:.1f}',
                (p.get_x() + p.get_width() / 2., height),
                ha='center', va='bottom',
                fontsize=12, color='black')

ax.set_ylim(0, 4)

# Show plot
plt.show()
```



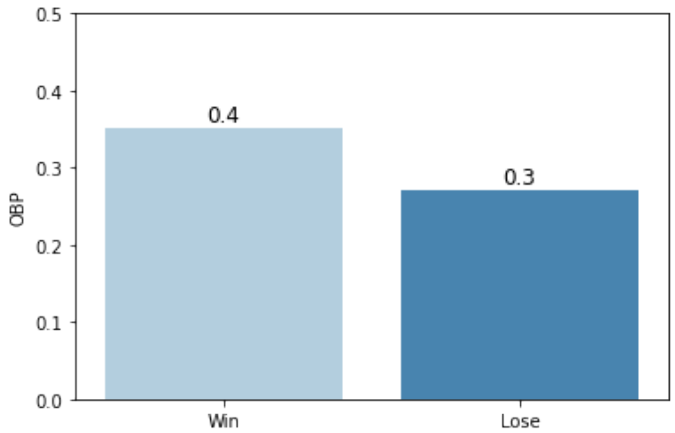
In [97]:
```
sns.barplot(x='win', y='errors', data=combo_df, ci=None, palette='Blues')

# Add labels and title
plt.xlabel('')
plt.ylabel('Errors')
plt.title('Mean Errors by Win-Loss')

# Show plot
plt.show()
```



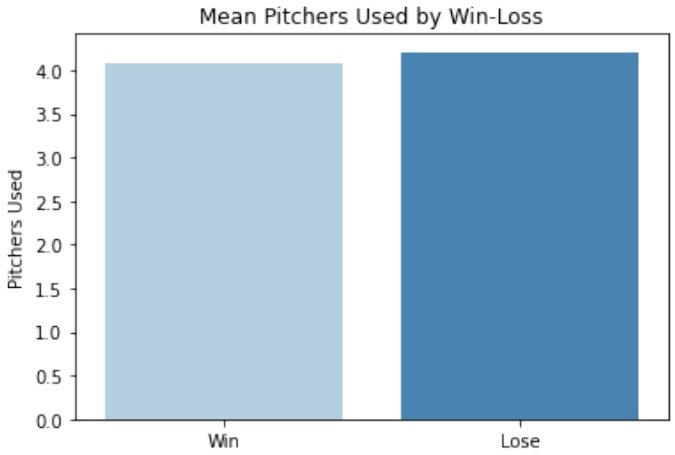
In [98]:
```
sns.barplot(x='win', y='home_run', data=combo_df, ci=None, palette='Blues')

# Add labels and title
plt.xlabel('')
plt.ylabel('Home Runs')
plt.title('Mean Home Runs by Win-Loss')

# Show plot
plt.show()
```



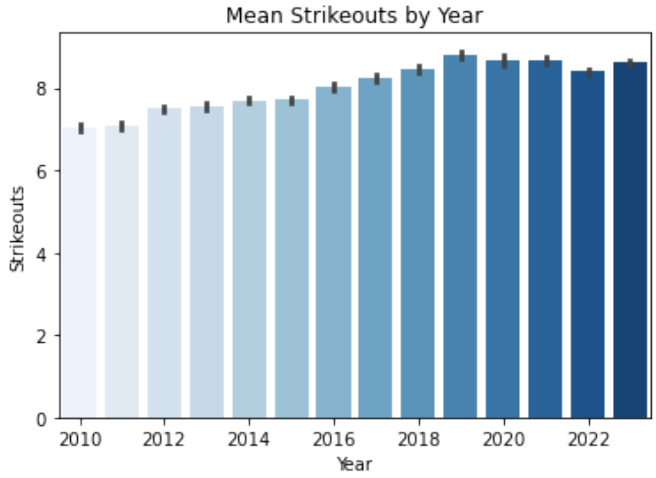
In [99]:
```
ax=sns.barplot(x='win', y='obp', data=combo_df, ci=None, palette='Blues')

# Add labels and title
plt.xlabel('')
plt.ylabel('OBP')
plt.title('')

for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height:.1f}',
                (p.get_x() + p.get_width() / 2., height),
                ha='center', va='bottom',
                fontsize=12, color='black')

ax.set_ylim(0, .5)

# Show plot
plt.show()
```



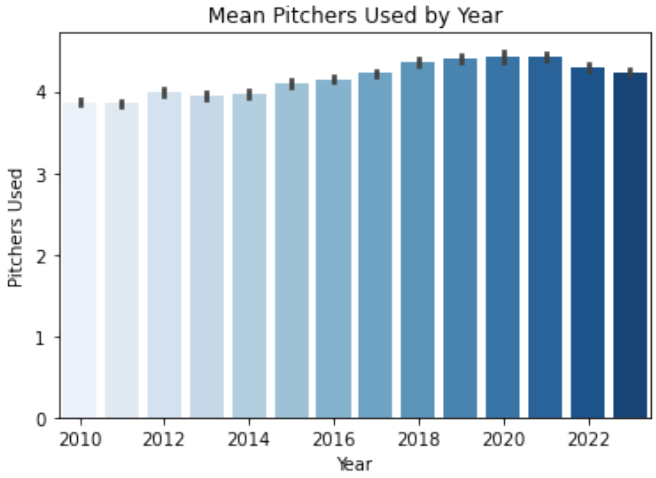
In [100]:
```
sns.barplot(x='win', y='pitchers_used', data=combo_df, ci=None, palette='Blues')

# Add labels and title
plt.xlabel('')
plt.ylabel('Pitchers Used')
plt.title('Mean Pitchers Used by Win-Loss')

# Show plot
plt.show()
```



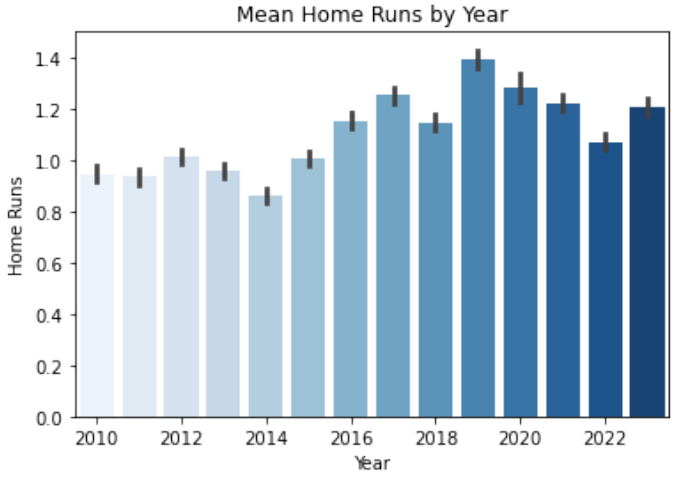
In [101]:
```
sns.barplot(x='year', y='strikeout', data=combo_df, palette='Blues')

# Add labels and title
plt.xlabel('Year')
plt.ylabel('Strikeouts')
plt.title('Mean Strikeouts by Year')

current_ticks = plt.gca().get_xticks()
plt.xticks(ticks=current_ticks[::2])

# Show plot
plt.show()
```



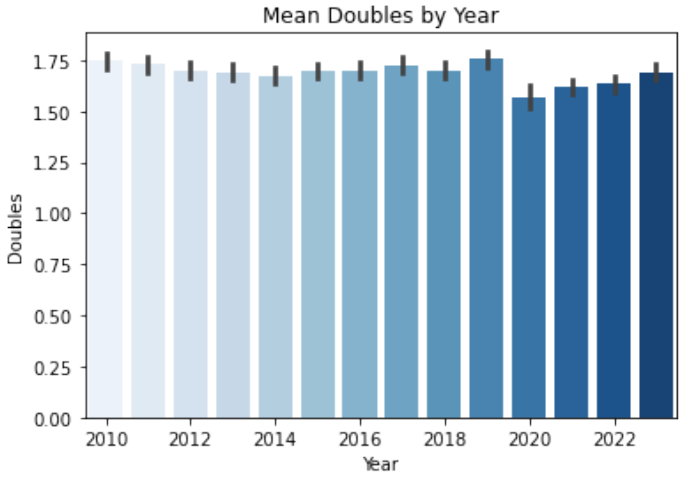
In [102]:
```
sns.barplot(x='year', y='pitchers_used', data=combo_df, palette='Blues')

# Add labels and title
plt.xlabel('Year')
plt.ylabel('Pitchers Used')
plt.title('Mean Pitchers Used by Year')

current_ticks = plt.gca().get_xticks()
plt.xticks(ticks=current_ticks[::2])

# Show plot
plt.show()
```



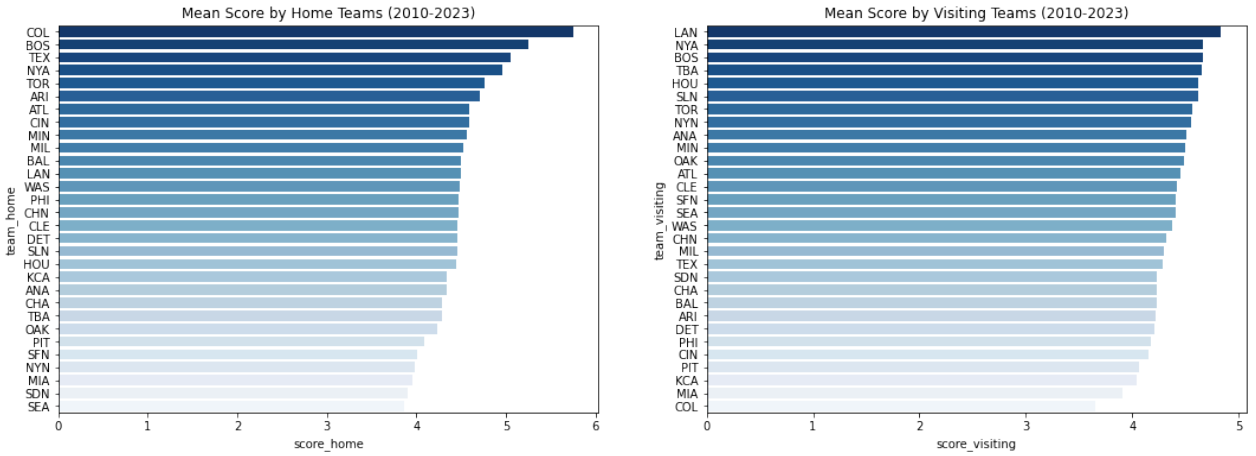
In [103]:
```
sns.barplot(x='year', y='home_run', data=combo_df, palette='Blues')

# Add labels and title
plt.xlabel('Year')
plt.ylabel('Home Runs')
plt.title('Mean Home Runs by Year')

current_ticks = plt.gca().get_xticks()
plt.xticks(ticks=current_ticks[::2])

# Show plot
plt.show()
```



In [104]:
```
sns.barplot(x='year', y='double', data=combo_df, palette='Blues')

# Add labels and title
plt.xlabel('Year')
plt.ylabel('Doubles')
plt.title('Mean Doubles by Year')

current_ticks = plt.gca().get_xticks()
plt.xticks(ticks=current_ticks[::2])

# Show plot
plt.show()
```



In [105]:
```
# Calculate the mean 'score_home' for each 'team_home'
mean_score_home = combo_df.groupby('team_home')['score_home'].mean().reset_index()

# Sort the data by mean 'score_home'
mean_score_home_sorted = mean_score_home.sort_values(by='score_home', ascending=False)

# Calculate the mean 'score_visiting' for each 'team_visiting'
mean_score_visiting = combo_df.groupby('team_visiting')['score_visiting'].mean().reset_index(

# Sort the data by mean 'score_visiting'
mean_score_visiting_sorted = mean_score_visiting.sort_values(by='score_visiting', ascending=F

# Create a figure with 1 row and 2 columns of subplots
fig, axes = plt.subplots(1, 2, figsize=(18, 6))

# Plot 1: Bar plot of mean 'score_home' by 'team_home'
sns.barplot(x='score_home', y='team_home', data=mean_score_home_sorted, ax=axes[0], palette='
axes[0].set_title('Mean Score by Home Teams (2010-2023)')

# Plot 2: Bar plot of mean 'score_visiting' by 'team_visiting'
sns.barplot(x='score_visiting', y='team_visiting', data=mean_score_visiting_sorted, ax=axes[1
axes[1].set_title('Mean Score by Visiting Teams (2010-2023)')

# Show plot
plt.show()
```



FUN NOTE: Colorado on average score more runs at home than any other team and on average score the least runs compared to any other team on the road. Does the thin air really make that much of a difference? Something of interest to explore in a future analysis!

In [106]:
```
combo_df.head()
```

Out[106]:

	date	team_visiting	team_home	score_visiting	score_home	outs_in_game	at_bats	hits	double	triple	ho
0	20100404	NYA	BOS	7	9	51	34.0	12	3.0	1.0	
1	20100405	MIN	ANA	3	6	51	33.0	9	0.0	0.0	
2	20100405	CLE	CHA	0	6	51	28.0	6	2.0	0.0	
3	20100405	TOR	TEX	4	5	52	31.0	6	2.0	0.0	
4	20100405	SDN	ARI	3	6	51	33.0	8	0.0	0.0	

In [107]:
```
combo_df['win'] = combo_df['win'].map({'Lose':0, 'Win': 1})
```

In [108]:
```
combo_df.to_csv('data/Combo_DF.csv')
```

Modeling

Data are split into a training set and test set for modeling. Games are paired so that features for the winning team and losing team of a game are not split between the training and test set to prevent data leakage. Features are standard scale standardized. No additional preprocessing is used. There are no categorical features, no missing data, and the target is 50:50 balanced.

The primary evaluation metric used for these models is accuracy. With a balanced dataset and equal interest in classification of wins as much as losses, accuracy is an appropriate evaluation metric.

X features varied by model and included: 'at_bats', 'hits', 'obp', 'double', 'triple', 'home_run', 'sacrifice_hit', 'sacrifice_fly', 'hit_by_pitch', 'walk', 'intent_walk', 'strikeout', 'stolen_base', 'caught_stealing', 'left_on_base', 'pitchers_used', 'wild_pitches', 'assists', 'errors', 'double_def'.

In [109]:
```
# Group by Game_ID
grouped = combo_df.groupby('ID')

# Convert groups to a list of DataFrames
game_list = [group for _, group in grouped]

# Perform train-test split on the list of games
train_games, test_games = train_test_split(game_list, test_size=0.2, random_state=13)

# Concatenate the DataFrames back into train and test sets
train_df = pd.concat(train_games).reset_index(drop=True)
test_df = pd.concat(test_games).reset_index(drop=True)
```

In [110]:
```
# The X features were selected based on EDA and domain interest
X_train= train_df[[
   'double',
   'triple',
   'home_run',
   'sacrifice_hit',
   'sacrifice_fly',
   'hit_by_pitch',
   'walk',
   'intent_walk',
   'strikeout',
   'stolen_base',
   'caught_stealing',
   'left_on_base',
   'pitchers_used',
   'wild_pitches',
   'assists',
   'errors',
   'double_def',
   'obp',
   'hits',
   'at_bats'
   ]]

y_train= train_df['win']

X_test= test_df[[
   'double',
   'triple',
   'home_run',
   'sacrifice_hit',
   'sacrifice_fly',
   'hit_by_pitch',
   'walk',
   'intent_walk',
   'strikeout',
   'stolen_base',
   'caught_stealing',
   'left_on_base',
   'pitchers_used',
   'wild_pitches',
   'assists',
   'errors',
   'double_def',
   'obp',
   'hits',
   'at_bats'
   ]]

y_test= test_df['win']
```

In [111]:
```
X_train.shape
```

Out[111]: (51894, 20)

In [112]:
```
y_train.shape
```

Out[112]: (51894,)

In [113]:
```
X_test.shape
```

Out[113]:
```
(12974, 20)
```

In [114]:
```
y_test.shape
```

Out[114]:
```
(12974,)
```

In [115]:
```
X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51894 entries, 0 to 51893
Data columns (total 20 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   double           51894 non-null  float64
 1   triple           51894 non-null  float64
 2   home_run         51894 non-null  int64  
 3   sacrifice_hit    51894 non-null  int64  
 4   sacrifice_fly    51894 non-null  int64  
 5   hit_by_pitch     51894 non-null  int64  
 6   walk             51894 non-null  int64  
 7   intent_walk      51894 non-null  int64  
 8   strikeout        51894 non-null  int64  
 9   stolen_base      51894 non-null  int64  
 10  caught_stealing  51894 non-null  int64  
 11  left_on_base     51894 non-null  int64  
 12  pitchers_used    51894 non-null  int64  
 13  wild_pitches     51894 non-null  int64  
 14  assists          51894 non-null  int64  
 15  errors           51894 non-null  int64  
 16  double_def       51894 non-null  int64  
 17  obp              51894 non-null  float64
 18  hits             51894 non-null  int64  
 19  at_bats          51894 non-null  float64
dtypes: float64(4), int64(16)
memory usage: 7.9 MB
```

Preprocessing

Data were standard scaled. X features were all numeric. There were no missing data. There was 50/50 class balance. No other preproccessors were used.

In [116]:
```
scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Baseline (Dummy) Model

The first model is a Dummy Classifier. As expected using the most frequent strategy the accuracy score of this balanced data set is 50%.

In [117]:
```
# Instantiate the model using most_frequent
dummy_model = DummyClassifier(strategy="most_frequent")

# fitting the model
dummy_model.fit(X_train_scaled, y_train)

# predicting
y_pred_dummy = dummy_model.predict(X_test_scaled)
dummy_model.score(X_train_scaled, y_train)
```

Out[117]:
```
0.5
```

If the model were to predict 1 (a win) each time, it would be accurate 50% of the time. This serves as a basic baseline to use as a comparison for models.

In [118]:
```
print(y_train.value_counts(normalize=True))
print(y_test.value_counts(normalize=True))
```

```
1    0.5
0    0.5
Name: win, dtype: float64
0    0.5
1    0.5
Name: win, dtype: float64
```

Decision Tree and Random Forest

The second model is a simple Decision Tree Classifier using parameters for criterion, maximum depth, and the full set of X features listed above. The accuracy score for this model is 75%. The model identifies: obp, left_on_base, intent_walk, at_bats, home_run, sacrifice_fly, and double as the seven most important features in order of importance.

Next a Random Forest Classifier uses a grid search for the estimator, and maxium depth and features. The best parameters are used in the model with the full set of X features listed above. The accuracy score for this model is 80%. The model identifies: home_run', 'obp', 'hits', 'left_on_base', 'errors', 'intent_walk', 'at_bats' as the seven most important features.

In [119]:
```
# Initialize Decision Tree
tree = DecisionTreeClassifier(random_state=13)
```

In [120]:
```
# Train the model on the training data
tree.fit(X_train_scaled, y_train)

tree = DecisionTreeClassifier(criterion='gini', max_depth=4, random_state=13)
tree.fit(X_train_scaled, y_train)
y_pred = tree.predict(X_test_scaled)
precision_score(y_test, y_pred, average="weighted")
```

Out[120]:
```
0.748970306240084
```

In [121]:
```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.7488053029135193
```

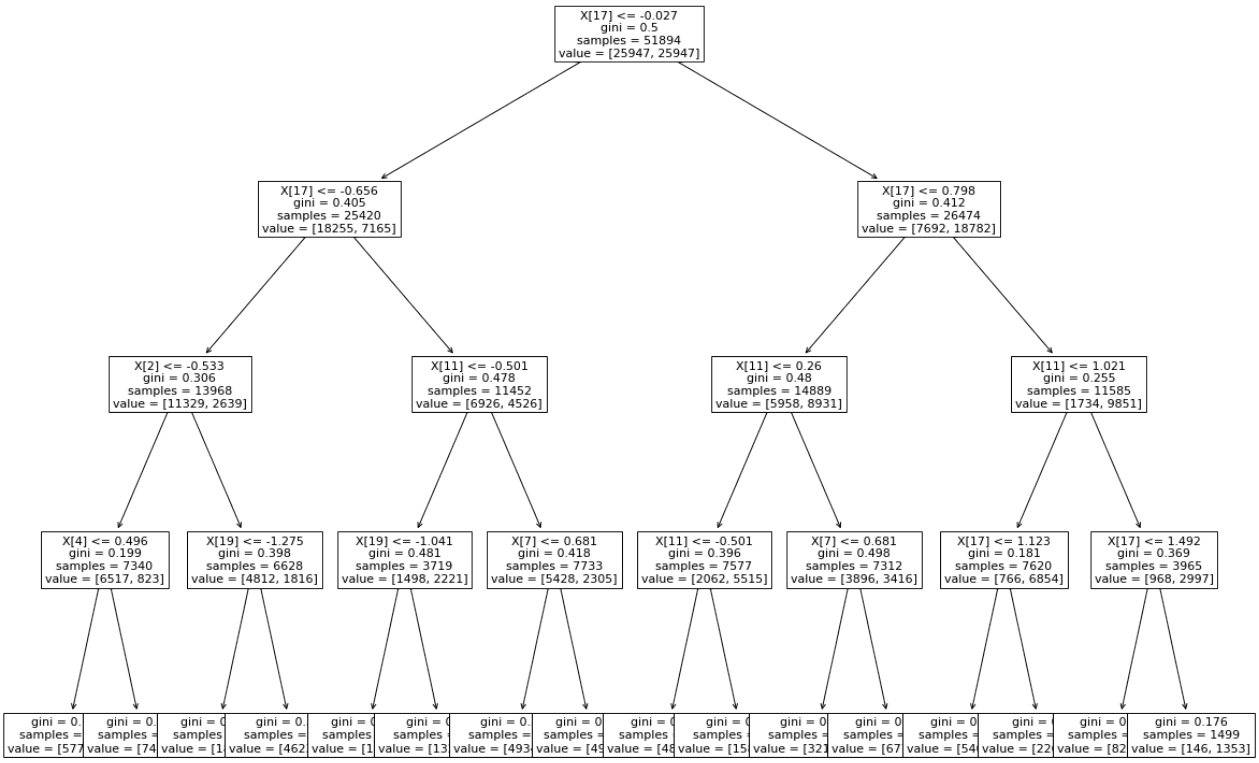
In [122]:
```
X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51894 entries, 0 to 51893
Data columns (total 20 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   double           51894 non-null  float64
 1   triple           51894 non-null  float64
 2   home_run         51894 non-null  int64  
 3   sacrifice_hit    51894 non-null  int64  
 4   sacrifice_fly    51894 non-null  int64  
 5   hit_by_pitch     51894 non-null  int64  
 6   walk             51894 non-null  int64  
 7   intent_walk      51894 non-null  int64  
 8   strikeout        51894 non-null  int64  
 9   stolen_base      51894 non-null  int64  
 10  caught_stealing  51894 non-null  int64  
 11  left_on_base     51894 non-null  int64  
 12  pitchers_used    51894 non-null  int64  
 13  wild_pitches     51894 non-null  int64  
 14  assists          51894 non-null  int64  
 15  errors           51894 non-null  int64  
 16  double_def       51894 non-null  int64  
 17  obp              51894 non-null  float64
 18  hits             51894 non-null  int64  
 19  at_bats          51894 non-null  float64
dtypes: float64(4), int64(16)
memory usage: 7.9 MB
```

In [123]:
```
# Get feature importances
feature_importances = tree.feature_importances_

# Get the original feature names
feature_names = X_train.columns

# Map the indices to feature names
feature_indices = tree.tree_.feature
for idx in feature_indices:
    if idx != -2:  # -2 is a special value in the tree structure indicating no feature (leaf n
        print(f'Feature index {idx} corresponds to feature name: {feature_names[idx]}')

# Alternatively, get the feature importance along with the feature names
importances = list(zip(feature_names, feature_importances))
importances = sorted(importances, key=lambda x: x[1], reverse=True)
print("Feature importances (sorted):")
for feature, importance in importances:
    print(f'{feature}: {importance:.4f}')
```

```
Feature index 17 corresponds to feature name: obp
Feature index 17 corresponds to feature name: obp
Feature index 2 corresponds to feature name: home_run
Feature index 4 corresponds to feature name: sacrifice_fly
Feature index 19 corresponds to feature name: at_bats
Feature index 11 corresponds to feature name: left_on_base
Feature index 19 corresponds to feature name: at_bats
Feature index 7 corresponds to feature name: intent_walk
Feature index 17 corresponds to feature name: obp
Feature index 11 corresponds to feature name: left_on_base
Feature index 11 corresponds to feature name: left_on_base
Feature index 7 corresponds to feature name: intent_walk
Feature index 11 corresponds to feature name: left_on_base
Feature index 17 corresponds to feature name: obp
Feature index 17 corresponds to feature name: obp
Feature importances (sorted):
obp: 0.7723
left_on_base: 0.1422
intent_walk: 0.0343
at_bats: 0.0235
home_run: 0.0226
sacrifice_fly: 0.0051
double: 0.0000
triple: 0.0000
sacrifice_hit: 0.0000
hit_by_pitch: 0.0000
walk: 0.0000
strikeout: 0.0000
stolen_base: 0.0000
caught_stealing: 0.0000
pitchers_used: 0.0000
wild_pitches: 0.0000
assists: 0.0000
errors: 0.0000
double_def: 0.0000
hits: 0.0000
```

In [124]:
```
# plotting the figure
plt.figure(figsize=(20,15))  # set plot size (denoted in inches)
plot_tree(tree, fontsize=11)
plt.show();
```



In [125]:
```
# Initialize Random Forest model
rf = RandomForestClassifier(random_state=13)
```

In [126]:
```
# Creating a grid search to find the best hyperparameters to include in the model
grid = {
    'n_estimators': [1000, 2000],
    'max_depth': [8, 10, 12],
    'max_features': [8, 10, 12]
       }
```

In [127]:
```
grid_search = GridSearchCV(estimator=rf, param_grid=grid, cv=5, scoring='accuracy')
grid_search.fit(X_train_scaled, y_train)
```

Out[127]:
```
GridSearchCV(cv=5, estimator=RandomForestClassifier(random_state=13),
             param_grid={'max_depth': [8, 10, 12], 'max_features': [8, 10, 12],
                         'n_estimators': [1000, 2000]},
             scoring='accuracy')
```

In [128]:
```
grid_search.best_params_
```

Out[128]:
```
{'max_depth': 12, 'max_features': 8, 'n_estimators': 2000}
```

In [129]:
```
grid_search.best_score_
```

Out[129]:
```
0.7932131941190306
```

In [130]:
```
# Train the model on the training data
rf.fit(X_train_scaled, y_train)

rf = RandomForestClassifier(n_estimators=2000, max_features=8, max_depth=12, random_state=13)
rf.fit(X_train_scaled, y_train)
y_pred = rf.predict(X_test_scaled)
```

In [131]:
```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.7953599506705719
```

In [132]:
```
# Get feature importances
feature_importances = rf.feature_importances_

# Get the original feature names
feature_names = X_train.columns

# Feature importance with feature names
importances = list(zip(feature_names, feature_importances))
importances = sorted(importances, key=lambda x: x[1], reverse=True)
print("Feature importances (sorted):")
for feature, importance in importances:
    print(f'{feature}: {importance:.4f}')
```

```
Feature importances (sorted):
obp: 0.3447
hits: 0.1143
at_bats: 0.0950
left_on_base: 0.0789
home_run: 0.0767
intent_walk: 0.0360
errors: 0.0318
pitchers_used: 0.0315
assists: 0.0284
strikeout: 0.0273
walk: 0.0249
double: 0.0227
stolen_base: 0.0202
wild_pitches: 0.0139
double_def: 0.0111
sacrifice_fly: 0.0104
sacrifice_hit: 0.0100
caught_stealing: 0.0081
hit_by_pitch: 0.0078
triple: 0.0063
```

Logistic Regression

Next, a logistic regression was performed to classify each game based on the predictors. Logistic Regression is a good model to use for classification. In this instance, a binary classification was performed to determine whether a game was won (1) or lost (0) for each team. The model uses the full set of X features listed above. The accuracy score for this model is 78%.

The second Logistic Regression model uses the subset of X features identified as the most important features

Log Model 1

In [133]:
```
# instantiating the model
logreg_full = LogisticRegression(random_state=13)

# fitting the model to our scaled training set
logreg_full.fit(X_train_scaled, y_train)
```

Out[133]:
```
LogisticRegression(random_state=13)
```

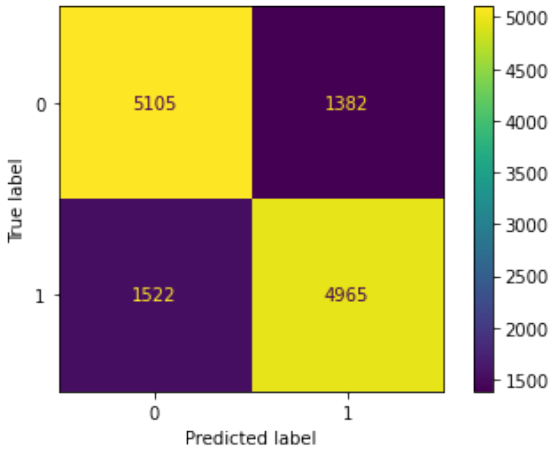
In [134]:
```
# predicting our target
y_pred = logreg_full.predict(X_test_scaled)
y_pred
```

Out[134]:
```
array([0, 1, 1, ..., 1, 1, 0], dtype=int64)
```

In [135]:
```
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.3f}')
```

```
Accuracy: 0.776
```

In [136]:
```
plot_confusion_matrix(logreg_full, X_test_scaled, y_test);
```



In [137]:
```
# printing out a full classification report to check precision and recall
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.77      0.79      0.78      6487
           1       0.78      0.77      0.77      6487

    accuracy                           0.78     12974
   macro avg       0.78      0.78      0.78     12974
weighted avg       0.78      0.78      0.78     12974
```

Log Model 2

In [138]:
```
# The X features were selected based on the Random Forest Feature Importance

X_train_DT= train_df[[
   'home_run',
   'obp',
   'hits',
   'left_on_base',
   'errors',
   'intent_walk',
   'at_bats'
       ]]

y_train= train_df['win']

X_test_DT= test_df[[
   'home_run',
   'obp',
   'hits',
   'left_on_base',
   'errors',
   'intent_walk',
   'at_bats'
   ]]

y_test= test_df['win']
```

In [139]:
```
scaler = StandardScaler()

X_train_DT_scaled = scaler.fit_transform(X_train_DT)
X_test_DT_scaled = scaler.transform(X_test_DT)
```

In [140]:
```
# instantiating the model
logreg_DT = LogisticRegression(random_state=13)

# fitting the model to our scaled training set
logreg_DT.fit(X_train_DT_scaled, y_train)
```

Out[140]:
```
LogisticRegression(random_state=13)
```

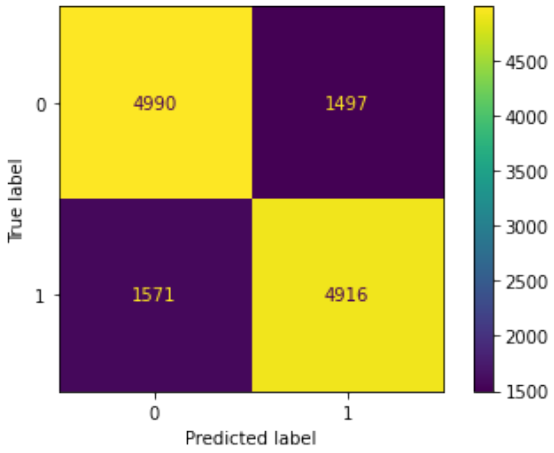
In [141]:
```
# predicting our target
y_pred = logreg_DT.predict(X_test_DT_scaled)
y_pred
```

Out[141]:
```
array([0, 1, 1, ..., 1, 1, 0], dtype=int64)
```

In [142]:
```
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.3f}')
```

```
Accuracy: 0.764
```

In [143]:
```
plot_confusion_matrix(logreg_DT, X_test_DT_scaled, y_test);
```



In [144]:
```
# printing out a full classification report to check precision and recall
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.76      0.77      0.76      6487
           1       0.77      0.76      0.76      6487

    accuracy                           0.76     12974
   macro avg       0.76      0.76      0.76     12974
weighted avg       0.76      0.76      0.76     12974
```

In [145]:
```
# printing out the coefficients
coefficients = logreg_DT.coef_
print("Coefficients:", coefficients)
```

```
Coefficients: [[ 0.30424077  1.69254943  0.22157442 -0.85614584 -0.38152523  0.46123072
  -0.20512428]]
```

In [146]:
```
# converting from log-odds to probabilities using a list comprehension from this source: https
probabilities = [np.exp(x)/(1 + np.exp(x)) for x in coefficients[0]]
probabilities
```

Out[146]:
```
[0.5754788783709657,
 0.8445591401367665,
 0.5551680813516696,
 0.29814521820637147,
 0.405759081682477,
 0.6133060956734895,
 0.44889798541277864]
```

Neural Network

The last model is a Neural Network. This model is built using extensive support from ChatGPT. The full set of X features listed above are used in the model. The accuracy score for this model is 81%; the best performing model. The seven most important feature identified in order of importance are: at_bats, left_on_base, walk, obp, pitchers_used, hits, and strikeouts, in order of importance.

In [147]:
```
X = X_train_scaled
y = y_train.values

# Convert to PyTorch tensors
X_tensor = torch.tensor(X, dtype=torch.float32)
y_tensor = torch.tensor(y, dtype=torch.long)

# Create a TensorDataset and DataLoader
dataset = TensorDataset(X_tensor, y_tensor)
dataloader = DataLoader(dataset, batch_size=32, shuffle=True)
```

In [148]:

```python
# Define a simple neural network model
class SimpleNN(nn.Module):
    def __init__(self):
        super(SimpleNN, self).__init__()
        self.fc1 = nn.Linear(20, 50)  # Adjust input size if necessary
        self.fc2 = nn.Linear(50, 2)   # Assuming binary classification (2 output classes)

    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = self.fc2(x)
        return x

# Initialize the model, loss function, and optimizer
model = SimpleNN()
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# Train the model
num_epochs = 20
for epoch in range(num_epochs):
    model.train()
    running_loss = 0.0
    for inputs, labels in dataloader:
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item()
    print(f"Epoch {epoch+1}, Loss: {running_loss/len(dataloader)}")

# Evaluate the model on the test set
model.eval()
X_test_tensor = torch.tensor(X_test_scaled, dtype=torch.float32)
y_test_tensor = torch.tensor(y_test.values, dtype=torch.long)
with torch.no_grad():
    outputs = model(X_test_tensor)
    probabilities = F.softmax(outputs, dim=1)
    _, predicted = torch.max(outputs, 1)
    accuracy = (predicted == y_test_tensor).sum().item() / y_test_tensor.size(0)
    print(f"Accuracy: {accuracy * 100:.2f}%")
print("Class probabilities:")
print(probabilities.numpy())
```

```
Epoch 1, Loss: 0.4630338629172557
Epoch 2, Loss: 0.43959598962571267
Epoch 3, Loss: 0.4296541095386304
Epoch 4, Loss: 0.4213777183659568
Epoch 5, Loss: 0.41721565913955616
Epoch 6, Loss: 0.41453444114375204
Epoch 7, Loss: 0.41235958035167136
Epoch 8, Loss: 0.4087538426666166
Epoch 9, Loss: 0.40436785060880215
Epoch 10, Loss: 0.40040836180072237
Epoch 11, Loss: 0.3968811105068603
Epoch 12, Loss: 0.3935023108003466
Epoch 13, Loss: 0.3911033780227901
Epoch 14, Loss: 0.3894945754222893
Epoch 15, Loss: 0.3879959057148376
Epoch 16, Loss: 0.38699366903775423
Epoch 17, Loss: 0.38628196898668676
Epoch 18, Loss: 0.3856687323453247
Epoch 19, Loss: 0.38497564842426085
Epoch 20, Loss: 0.3842244464467692
Accuracy: 81.72%
Class probabilities:
[[0.9104444  0.08955562]
 [0.35291407 0.647086  ]
 [0.02048325 0.97951674]
 ...
 [0.35026073 0.64973927]
 [0.05132128 0.94867873]
 [0.97275835 0.02724163]]
```

In [149]:
```
# Extract feature importances from the first layer of the model
importances = model.fc1.weight.detach().numpy()

# Calculate the mean absolute importance for each feature
feature_importance = np.mean(np.abs(importances), axis=0)

# Create a list of (feature_name, importance) tuples
feature_importance_tuples = [(name, importance) for name, importance in zip(X_train.columns,

# Sort the list of tuples by importance in descending order
sorted_feature_importance = sorted(feature_importance_tuples, key=lambda x: x[1], reverse=Tru

# Print the sorted feature importances
print("Feature Importances (most to least important):")
for name, importance in sorted_feature_importance:
    print(f"{name}: {importance:.4f}")
```

```
Feature Importances (most to least important):
at_bats: 0.4346
left_on_base: 0.2709
walk: 0.2453
obp: 0.2317
pitchers_used: 0.1740
hits: 0.1637
home_run: 0.1547
intent_walk: 0.1507
sacrifice_fly: 0.1310
triple: 0.1288
assists: 0.1264
caught_stealing: 0.1258
sacrifice_hit: 0.1246
stolen_base: 0.1222
errors: 0.1178
double: 0.1140
double_def: 0.1094
hit_by_pitch: 0.1086
strikeout: 0.1014
wild_pitches: 0.0889
```

In [150]:
```
#!pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu11
```