```
import pandas as pd
df_qb= pd.read_csv("passing_summary.csv")
df_qb
```

	player	player_id	position	team_name	player_game_count	accuracy_percent	aimed_passes	attempts	avg_depth_of_target	avg_time_to_throw	...	saci
0	Tom Brady	698	QB	TB	18	74.8	743	799	7.3	2.30	...	
1	Patrick Mahomes	11765	QB	KC	20	78.0	685	747	7.5	2.85	...	
2	Justin Herbert	28237	QB	LAC	18	78.8	673	743	6.9	2.74	...	
3	Joe Burrow	28022	QB	CIN	19	77.3	660	715	7.6	2.50	...	
4	Kirk Cousins	7102	QB	MIN	18	76.7	621	682	7.9	2.69	...	
...	...	...	...	...	...	...	...	...	...	...	...	
101	Tommy Townsend	33007	P	KC	1	0.0	1	1	1.0	3.40	...	
102	Jack Fox	34735	P	DET	1	100.0	1	1	6.0	2.00	...	
103	Cooper Kupp	11824	WR	LA	1	0.0	1	1	25.0	3.30	...	
104	Chase Claypool	42312	WR	CHI	1	100.0	1	1	5.0	2.20	...	
105	DeeJay Dallas	55632	HB	SEA	1	0.0	1	1	14.0	3.00	...	

106 rows × 42 columns

```
import pandas as pd

records = {'Arizona Cardinals': '4 - 13',
 'Atlanta Falcons': '7 - 10',
 'Baltimore Ravens': '10 - 8',
 'Buffalo Bills': '14 - 4',
 'Carolina Panthers': '7 - 10',
 'Chicago Bears': '3 - 14',
 'Cincinnati Bengals': '14 - 5',
 'Cleveland Browns': '7 - 10',
 'Dallas Cowboys': '13 - 6',
 'Denver Broncos': '5 - 12',
 'Detroit Lions': '9 - 8',
 'Green Bay Packers': '8 - 9',
 'Houston Texans': '3 - 13 - 1',
 'Indianapolis Colts': '4 - 12 - 1',
 'Jacksonville Jaguars': '10 - 9',
 'Kansas City Chiefs': '17 - 3',
 'Las Vegas Raiders': '6 - 11',
 'Los Angeles Chargers': '10 - 8',
 'Los Angeles Rams': '5 - 12',
 'Miami Dolphins': '9 - 9',
 'Minnesota Vikings': '13 - 5',
 'New England Patriots': '8 - 9',
 'New Orleans Saints': '7 - 10',
 'New York Giants': '10 - 8 - 1',
 'New York Jets': '7 - 10',
 'Philadelphia Eagles': '16 - 4',
 'Pittsburgh Steelers': '9 - 8',
 'San Francisco 49ers': '15 - 5',
 'Seattle Seahawks': '9 - 9',
 'Tampa Bay Buccaneers': '8 - 10',
 'Tennessee Titans': '7 - 10',
 'Washington Commanders': '8 - 8 - 1'}

team_records = []
for team, record in records.items():
    parts = record.split(' - ')
    wins = int(parts[0])
```

```
team_records = []
for team, record in records.items():
    parts = record.split(' - ')
    wins = int(parts[0])
    losses = int(parts[1])
    ties = 0
    if len(parts) == 3:
        ties = int(parts[2])
    wins += 0.5 * ties
    losses += 0.5 * ties
    total_games = wins + losses
    win_pct = wins / total_games
    team_records.append((team, wins, losses, ties, win_pct))

df_WINS = pd.DataFrame(team_records, columns=['Team', 'Wins', 'Losses', 'Ties', 'Win Percentage'])
print(df_WINS)
```

	Team	Wins	Losses	Ties	Win Percentage
0	Arizona Cardinals	4.0	13.0	0	0.235294
1	Atlanta Falcons	7.0	10.0	0	0.411765
2	Baltimore Ravens	10.0	8.0	0	0.555556
3	Buffalo Bills	14.0	4.0	0	0.777778
4	Carolina Panthers	7.0	10.0	0	0.411765
5	Chicago Bears	3.0	14.0	0	0.176471
6	Cincinnati Bengals	14.0	5.0	0	0.736842
7	Cleveland Browns	7.0	10.0	0	0.411765
8	Dallas Cowboys	13.0	6.0	0	0.684211
9	Denver Broncos	5.0	12.0	0	0.294118
10	Detroit Lions	9.0	8.0	0	0.529412
11	Green Bay Packers	8.0	9.0	0	0.470588
12	Houston Texans	3.5	13.5	1	0.205882
13	Indianapolis Colts	4.5	12.5	1	0.264706
14	Jacksonville Jaguars	10.0	9.0	0	0.526316
15	Kansas City Chiefs	17.0	3.0	0	0.850000
16	Las Vegas Raiders	6.0	11.0	0	0.352941
17	Los Angeles Chargers	10.0	8.0	0	0.555556
18	Los Angeles Rams	5.0	12.0	0	0.294118
19	Miami Dolphins	9.0	9.0	0	0.500000
20	Minnesota Vikings	13.0	5.0	0	0.722222

```
name_to_letter = {
    'Arizona Cardinals': 'ARZ',
    'Atlanta Falcons': 'ATL',
    'Baltimore Ravens': 'BLT',
    'Buffalo Bills': 'BUF',
    'Carolina Panthers': 'CAR',
    'Chicago Bears': 'CHI',
    'Cincinnati Bengals': 'CIN',
    'Cleveland Browns': 'CLV',
    'Dallas Cowboys': 'DAL',
    'Denver Broncos': 'DEN',
    'Detroit Lions': 'DET',
    'Green Bay Packers': 'GB',
    'Houston Texans': 'HST',
    'Indianapolis Colts': 'IND',
    'Jacksonville Jaguars': 'JAX',
    'Kansas City Chiefs': 'KC',
    'Las Vegas Raiders': 'LV',
    'Los Angeles Chargers': 'LAC',
    'Los Angeles Rams': 'LA',
    'Miami Dolphins': 'MIA',
    'Minnesota Vikings': 'MIN',
    'New England Patriots': 'NE',
    'New Orleans Saints': 'NO',
    'New York Giants': 'NYG',
    'New York Jets': 'NYJ',
    'Philadelphia Eagles': 'PHI',
    'Pittsburgh Steelers': 'PIT',
    'San Francisco 49ers': 'SF',
    'Seattle Seahawks': 'SEA',
    'Tampa Bay Buccaneers': 'TB',
    'Tennessee Titans': 'TEN',
    'Washington Commanders': 'WAS'
}
```

```
playoff_dict = {
    "ARZ": 0,
    "ATL": 0,
    "BLT": 1,
    "BUF": 1,
    "CAR": 0,
    "CHI": 0,
    "CIN": 1,
    "CLV": 0,
    "DAL": 1,
    "DEN": 0,
    "DET": 0,
    "GB": 0,
    "HST": 0,
    "IND": 0,
    "JAX": 1,
    "KC": 1,
    "LV": 0,
    "LAC": 1,
    "LA": 0,
    "MIA": 1,
    "MIN": 1,
    "NE": 0,
    "NO": 0,
    "NYG": 1,
    "NYJ": 0,
    "PHI": 1,
    "PIT": 0,
    "SF": 1,
    "SEA": 0,
    "TB": 1,
    "TEN": 0,
    "WAS": 0
}
```

```
max_yards_qbs = df_qb.groupby('team_name')['yards'].idxmax().apply(lambda x: df_qb.loc[x]['player'])
max_yards_qbs
```

```
team_name
ARZ       Kyler Murray
ATL     Marcus Mariota
BLT      Lamar Jackson
BUF         Josh Allen
CAR        Sam Darnold
CHI      Justin Fields
CIN         Joe Burrow
CLV    Jacoby Brissett
DAL       Dak Prescott
DEN     Russell Wilson
DET         Jared Goff
GB       Aaron Rodgers
HST        Davis Mills
IND          Matt Ryan
JAX    Trevor Lawrence
KC     Patrick Mahomes
LA      Baker Mayfield
LAC     Justin Herbert
LV          Derek Carr
MIA     Tua Tagovailoa
MIN       Kirk Cousins
NE            Mac Jones
NO          Andy Dalton
NYG       Daniel Jones
NYJ        Zach Wilson
PHI        Jalen Hurts
PIT      Kenny Pickett
SEA         Geno Smith
SF     Jimmy Garoppolo
TB            Tom Brady
TEN     Ryan Tannehill
WAS    Taylor Heinicke
Name: yards, dtype: object
```

```
qb_names = ['Kyler Murray', 'Marcus Mariota', 'Lamar Jackson', 'Josh Allen', 'Sam Darnold', 'Justin Fields', 'Joe Burrc

df_filtered = df_qb[df_qb['player'].isin(qb_names)]
```

```
df_filtered
```

	player	player_id	position	team_name	player_game_count	accuracy_percent	aimed_passes	attempts	avg_depth_of_target	avg_time_to_throw	...	sack
0	Tom Brady	698	QB	TB	18	74.8	743	799	7.3	2.30	...	
1	Patrick Mahomes	11765	QB	KC	20	78.0	685	747	7.5	2.85	...	
2	Justin Herbert	28237	QB	LAC	18	78.8	673	743	6.9	2.74	...	
3	Joe Burrow	28022	QB	CIN	19	77.3	660	715	7.6	2.50	...	
4	Kirk Cousins	7102	QB	MIN	18	76.7	621	682	7.9	2.69	...	
5	Trevor Lawrence	77632	QB	JAX	19	76.3	624	670	7.9	2.50	...	
6	Josh Allen	46601	QB	BUF	18	74.2	596	648	10.2	2.91	...	
7	Geno Smith	7820	QB	SEA	18	77.7	565	607	8.3	2.79	...	
8	Jared Goff	10635	QB	DET	17	77.8	531	587	7.6	2.69	...	
9	Jalen Hurts	40291	QB	PHI	18	77.4	500	547	8.7	2.86	...	
10	Aaron Rodgers	2241	QB	GB	17	75.8	501	542	8.5	2.67	...	
11	Daniel Jones	39395	QB	NYG	18	80.1	482	534	6.5	3.01	...	
12	Derek Carr	8671	QB	LV	15	70.8	455	502	9.7	2.84	...	

```
import pandas as pd

playoff_df = pd.DataFrame.from_dict(playoff_dict, orient='index', columns=['playoff'])
playoff_df
```

	playoff
ARZ	0
ATL	0
BLT	1
BUF	1
CAR	0
CHI	0
CIN	1
CLV	0
DAL	1
DEN	0
DET	0
GB	0
HST	0
IND	0
JAX	1
KC	1
LV	0
LAC	1
LA	0
MIA	1

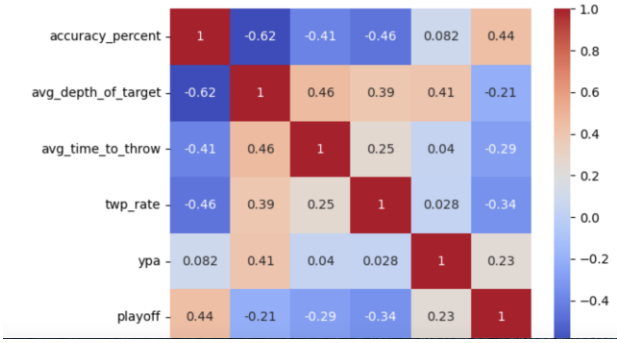
```
import pandas as pd

playoff_dict = {
    "ARZ": 0,
    "ATL": 0,
    "BLT": 1,
    "BUF": 1,
    "CAR": 0,
    "CHI": 0,
    "CIN": 1,
    "CLV": 0,
    "DAL": 1,
    "DEN": 0,
    "DET": 0,
    "GB": 0,
    "HST": 0,
    "IND": 0,
    "JAX": 1,
    "KC": 1,
    "LV": 0,
    "LAC": 1,
    "LA": 0,
    "MIA": 1,
    "MIN": 1,
    "NE": 0,
    "NO": 0,
    "NYG": 1,
    "NYJ": 0,
    "PHI": 1,
    "PIT": 0,
    "SF": 1,
    "SEA": 0,
    "TB": 1,
    "TEN": 0,
    "WAS": 0
}

playoff_df = pd.DataFrame.from_dict(playoff_dict, orient='index', columns=['playoff'])
```

```
combined_df = df_filtered.merge(playoff_df, on='team_name')
combined_df
```

	player	player_id	position	team_name	player_game_count	accuracy_percent	aimed_passes	attempts	avg_depth_of_target	avg_time_to_throw	...	sack
0	Tom Brady	698	QB	TB	18	74.8	743	799	7.3	2.30	...	2
1	Patrick Mahomes	11765	QB	KC	20	78.0	685	747	7.5	2.85	...	2
2	Justin Herbert	28237	QB	LAC	18	78.8	673	743	6.9	2.74	...	4
3	Joe Burrow	28022	QB	CIN	19	77.3	660	715	7.6	2.50	...	5
4	Kirk Cousins	7102	QB	MIN	18	76.7	621	682	7.9	2.69	...	4
5	Trevor Lawrence	77632	QB	JAX	19	76.3	624	670	7.9	2.50	...	3
6	Josh Allen	46601	QB	BUF	18	74.2	596	648	10.2	2.91	...	4
7	Geno Smith	7820	QB	SEA	18	77.7	565	607	8.3	2.79	...	4
8	Jared Goff	10635	QB	DET	17	77.8	531	587	7.6	2.69	...	2
9	Jalen Hurts	40291	QB	PHI	18	77.4	500	547	8.7	2.86	...	4
10	Aaron Rodgers	2241	QB	GB	17	75.8	501	542	8.5	2.67	...	3
11	Daniel Jones	39395	QB	NYG	18	80.1	482	534	6.5	3.01	...	5
12	Derek Carr	8671	QB	LV	15	70.8	455	502	9.7	2.84	...	2
13	Russell Wilson	7077	QB	DEN	15	74.1	432	484	9.5	2.94	...	5
14	Davis Mills	52269	QB	HST	15	69.9	439	479	8.4	2.66	...	3

```
import seaborn as sns

cols = ['accuracy_percent', 'avg_depth_of_target', 'avg_time_to_throw', 'twp_rate', 'ypa', 'playoff']

# Create a new dataframe
heatmap_df = combined_df[cols]

# make correlation matrix
corr = heatmap_df.corr()

# Create a heatmap
sns.heatmap(corr, annot=True, cmap='coolwarm')
```

```
<AxesSubplot:>
```




```
import pandas as pd
df_rb= pd.read_csv("rushing_summary.csv")
df_rb
```

	player	player_id	position	team_name	player_game_count	attempts	avoided_tackles	breakaway_attempts	breakaway_percent	breakaway_yards	...	sc
0	Derrick Henry	10679	HB	TEN	16	349	69	15	27.6	424	...	
1	Josh Jacobs	45953	HB	LV	17	339	90	15	23.4	386	...	
2	Saquon Barkley	45791	HB	NYG	18	313	41	21	37.3	532	...	
3	Nick Chubb	45783	HB	CLV	17	302	83	23	34.6	527	...	
4	Miles Sanders	40555	HB	PHI	20	294	52	13	21.6	306	...	
...	...	...	...	...	...	...	...	...	...	...	...	
365	Marquez Valdes-Scantling	47809	WR	KC	20	1	1	0	0.0	0	...	
366	Darius Slayton	25578	WR	NYG	18	1	0	0	0.0	0	...	
367	Keenan Allen	7857	WR	LAC	11	1	0	0	0.0	0	...	
368	Denzel Mims	47800	WR	NYJ	10	1	0	0	0.0	0	...	
369	Scotty Miller	47481	WR	TB	15	1	0	0	0.0	0	...	

370 rows × 47 columns

```
max_yards_rbs = df_rb.groupby('team_name')['yards'].idxmax().apply(lambda x: df_rb.loc[x]['player'])
max_yards_rbs
```

```
team_name
ARZ            James Conner
ATL          Tyler Allgeier
BLT           Lamar Jackson
BUF        Devin Singletary
CAR          D'Onta Foreman
CHI           Justin Fields
CIN              Joe Mixon
CLV             Nick Chubb
DAL           Tony Pollard
DEN         Latavius Murray
DET         Jamaal Williams
GB              Aaron Jones
HST           Dameon Pierce
IND         Jonathan Taylor
JAX          Travis Etienne
KC            Isiah Pacheco
LA               Cam Akers
LAC           Austin Ekeler
LV              Josh Jacobs
MIA          Raheem Mostert
MIN             Dalvin Cook
NE      Rhamondre Stevenson
NO             Alvin Kamara
NYG          Saquon Barkley
NYJ             Breece Hall
PHI           Miles Sanders
PIT            Najee Harris
SEA      Kenneth Walker III
SF      Christian McCaffrey
TB        Leonard Fournette
TEN           Derrick Henry
WAS      Brian Robinson Jr.
```

```
rb_names = ['James Conner', 'Tyler Allgeier', 'Lamar Jackson', 'Devin Singletary', 'D\'Onta Foreman', 'Justin Fields',

rb_filtered = df_rb[df_rb['player'].isin(rb_names)]
rb_filtered
```

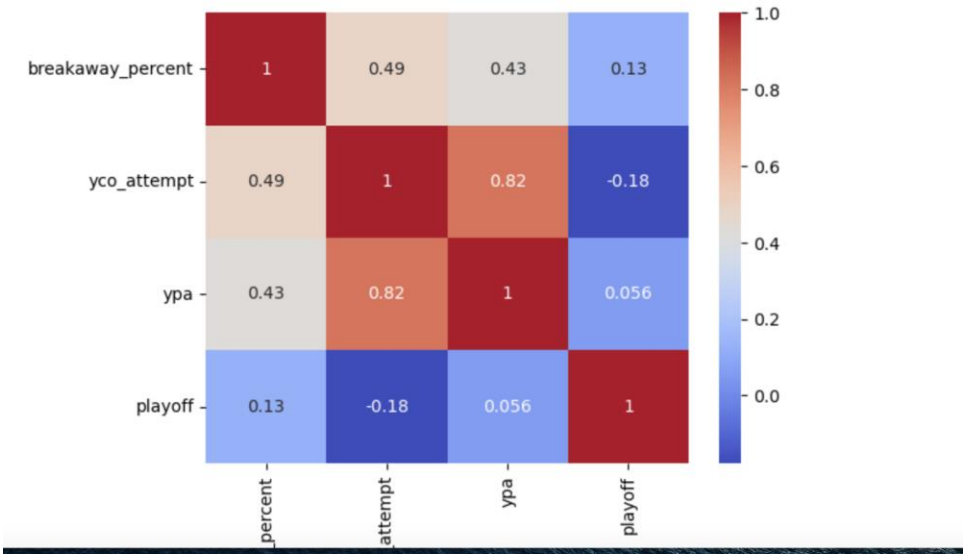
	player	player_id	position	team_name	player_game_count	attempts	avoided_tackles	breakaway_attempts	breakaway_percent	breakaway_yards	...	s
0	Derrick Henry	10679	HB	TEN	16	349	69	15	27.6	424	...	
1	Josh Jacobs	45953	HB	LV	17	339	90	15	23.4	386	...	
2	Saquon Barkley	45791	HB	NYG	18	313	41	21	37.3	532	...	
3	Nick Chubb	45783	HB	CLV	17	302	83	23	34.6	527	...	
4	Miles Sanders	40555	HB	PHI	20	294	52	13	21.6	306	...	
5	Christian McCaffrey	11763	HB	SF	20	284	48	16	32.5	447	...	
6	Dalvin Cook	11796	HB	MIN	18	280	52	12	27.7	344	...	
7	Najee Harris	57121	HB	PIT	17	272	55	7	13.0	135	...	
8	Jamaal Williams	11889	HB	DET	17	262	30	7	18.9	201	...	
10	Travis Etienne	57185	HB	JAX	19	251	64	18	38.3	495	...	
11	Joe Mixon	11803	HB	CIN	17	249	26	7	17.6	172	...	
12	Kenneth Walker III	97199	HB	SEA	16	243	50	17	43.2	481	...	
13	Alvin Kamara	11822	HB	NO	15	223	34	6	13.4	120	...	
14	Dameon	83659	HB	HST	13	220	62	9	29.6	278	...	

```
rb_playoff_df = pd.merge(rb_filtered, playoff_df, on='team_name')
rb_playoff_df
```

	player	player_id	position	team_name	player_game_count	attempts	avoided_tackles	breakaway_attempts	breakaway_percent	breakaway_yards	...	t
0	Derrick Henry	10679	HB	TEN	16	349	69	15	27.6	424	...	
1	Josh Jacobs	45953	HB	LV	17	339	90	15	23.4	386	...	
2	Saquon Barkley	45791	HB	NYG	18	313	41	21	37.3	532	...	
3	Nick Chubb	45783	HB	CLV	17	302	83	23	34.6	527	...	
4	Miles Sanders	40555	HB	PHI	20	294	52	13	21.6	306	...	
5	Christian McCaffrey	11763	HB	SF	20	284	48	16	32.5	447	...	
6	Dalvin Cook	11796	HB	MIN	18	280	52	12	27.7	344	...	
7	Najee Harris	57121	HB	PIT	17	272	55	7	13.0	135	...	
8	Jamaal Williams	11889	HB	DET	17	262	30	7	18.9	201	...	
9	Travis Etienne	57185	HB	JAX	19	251	64	18	38.3	495	...	
10	Joe Mixon	11803	HB	CIN	17	249	26	7	17.6	172	...	
11	Kenneth Walker III	97199	HB	SEA	16	243	50	17	43.2	481	...	
12	Alvin Kamara	11822	HB	NO	15	223	34	6	13.4	120	...	
13	Dameon Pierce	83659	HB	HST	13	220	62	9	29.6	278	...	
14	Austin Ekeler	12164	HB	LAC	18	217	44	7	24.5	233	...	

```
import seaborn as sns

cols = ['breakaway_percent', 'yco_attempt', 'ypa', 'playoff']

heatmap_df = rb_playoff_df[cols]

corr = heatmap_df.corr()

sns.heatmap(corr, annot=True, cmap='coolwarm')
```

```
<AxesSubplot:>
```



```python
import statsmodels.api as sm

# Define predictor variables and outcome variable
X = combined_df[['accuracy_percent', 'avg_depth_of_target', 'avg_time_to_throw', 'twp_rate', 'ypa']]
y = combined_df['playoff']

# Add a constant to the predictor variables
X = sm.add_constant(X)

# Create a linear regression model
model = sm.OLS(y, X).fit()

print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                playoff   R-squared:                       0.278
Model:                            OLS   Adj. R-squared:                  0.139
Method:                 Least Squares   F-statistic:                     1.998
Date:                Fri, 12 May 2023   Prob (F-statistic):              0.112
Time:                        14:28:29   Log-Likelihood:                -17.450
No. Observations:                  32   AIC:                             46.90
Df Residuals:                      26   BIC:                             55.69
Df Model:                           5
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
const                     -3.1165      3.300     -0.944      0.354      -9.899       3.666
accuracy_percent           0.0444      0.036      1.219      0.234      -0.030       0.119
avg_depth_of_target        0.0145      0.125      0.116      0.908      -0.242       0.271
avg_time_to_throw         -0.2966      0.408     -0.727      0.474      -1.135       0.542
twp_rate                  -0.0863      0.089     -0.970      0.341      -0.269       0.097
ypa                        0.1668      0.168      0.991      0.331      -0.179       0.513
==============================================================================
Omnibus:                       10.625   Durbin-Watson:                   1.914
Prob(Omnibus):                  0.005   Jarque-Bera (JB):                2.472
Skew:                          -0.052   Prob(JB):                        0.291
Kurtosis:                       1.642   Cond. No.                     3.05e+03
```

```python
import statsmodels.api as sm

X = combined_df[['accuracy_percent', 'avg_time_to_throw', 'twp_rate', 'ypa']]
y = combined_df['playoff']

X = sm.add_constant(X)

model = sm.OLS(y, X).fit()

print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                playoff   R-squared:                       0.277
Model:                            OLS   Adj. R-squared:                  0.170
Method:                 Least Squares   F-statistic:                     2.589
Date:                Fri, 12 May 2023   Prob (F-statistic):             0.0592
Time:                        14:28:29   Log-Likelihood:                -17.459
No. Observations:                  32   AIC:                             44.92
Df Residuals:                      27   BIC:                             52.25
Df Model:                           4
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
const                     -2.9343      2.851     -1.029      0.312      -8.784       2.915
accuracy_percent           0.0419      0.029      1.448      0.159      -0.017       0.101
avg_time_to_throw         -0.2827      0.383     -0.738      0.467      -1.069       0.503
twp_rate                  -0.0850      0.087     -0.981      0.335      -0.263       0.093
ypa                        0.1782      0.134      1.327      0.196      -0.097       0.454
==============================================================================
Omnibus:                        9.690   Durbin-Watson:                   1.927
Prob(Omnibus):                  0.008   Jarque-Bera (JB):                2.376
Skew:                          -0.037   Prob(JB):                        0.305
Kurtosis:                       1.667   Cond. No.                     2.67e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

```
import statsmodels.api as sm

X = combined_df[['accuracy_percent', 'twp_rate', 'ypa']]
y = combined_df['playoff']

X = sm.add_constant(X)

model = sm.OLS(y, X).fit()

print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                playoff   R-squared:                       0.263
Model:                            OLS   Adj. R-squared:                  0.184
Method:                 Least Squares   F-statistic:                     3.324
Date:                Fri, 12 May 2023   Prob (F-statistic):             0.0339
Time:                        14:28:29   Log-Likelihood:                -17.778
No. Observations:                  32   AIC:                             43.56
Df Residuals:                      28   BIC:                             49.42
Df Model:                           3
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
const                -4.2030      2.256     -1.863      0.073      -8.824       0.418
accuracy_percent      0.0492      0.027      1.828      0.078      -0.006       0.104
twp_rate             -0.0897      0.086     -1.047      0.304      -0.265       0.086
ypa                   0.1708      0.133      1.286      0.209      -0.101       0.443
==============================================================================
Omnibus:                       11.909   Durbin-Watson:                   1.767
Prob(Omnibus):                  0.003   Jarque-Bera (JB):                2.574
Skew:                           0.017   Prob(JB):                        0.276
Kurtosis:                       1.611   Cond. No.                     2.13e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.13e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

```
import statsmodels.api as sm

X = combined_df[['accuracy_percent', 'ypa']]
y = combined_df['playoff']

X = sm.add_constant(X)

model = sm.OLS(y, X).fit()

print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                playoff   R-squared:                       0.234
Model:                            OLS   Adj. R-squared:                  0.181
Method:                 Least Squares   F-statistic:                     4.424
Date:                Fri, 12 May 2023   Prob (F-statistic):             0.0210
Time:                        14:28:29   Log-Likelihood:                -18.392
No. Observations:                  32   AIC:                             42.78
Df Residuals:                      29   BIC:                             47.18
Df Model:                           2
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
const                -5.3936      1.952     -2.764      0.010      -9.385      -1.402
accuracy_percent      0.0623      0.024      2.609      0.014       0.013       0.111
ypa                   0.1604      0.133      1.209      0.237      -0.111       0.432
==============================================================================
Omnibus:                       11.735   Durbin-Watson:                   1.749
Prob(Omnibus):                  0.003   Jarque-Bera (JB):                2.558
Skew:                          -0.010   Prob(JB):                        0.278
Kurtosis:                       1.615   Cond. No.                     1.83e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.83e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

```python
import pandas as pd
from sklearn.linear_model import Lasso
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer


# Define the predictor variables and the outcome variable
cols = ['breakaway_percent', 'yco_attempt', 'ypa', 'yprr', 'playoff']
rb_playoff_df = rb_playoff_df[cols]
X = rb_playoff_df.drop(columns=['playoff'])
y = rb_playoff_df['playoff']

# Handle missing values by imputing the mean of each column
imputer = SimpleImputer(strategy='mean')
X = imputer.fit_transform(X)

# Standardize predictor variables
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Create a Lasso regression model
model = Lasso(alpha=0.1)
model.fit(X, y)

# Print model coefficients
print('Intercept:', model.intercept_)
print('Coefficients:', model.coef_)
```

```
Intercept: 0.40625
Coefficients: [ 0. -0.  0.  0.]
```

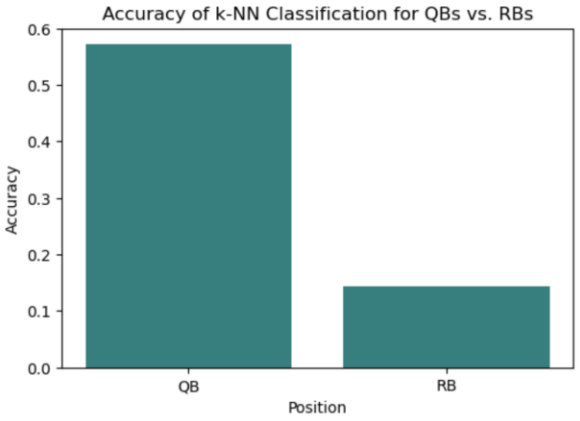
```python
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

X = combined_df[['accuracy_percent', 'avg_depth_of_target', 'avg_time_to_throw', 'twp_rate', 'ypa']]
y = combined_df['playoff']


# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a k-NN classifier
k = 5  # number nearest neighbors
clf = KNeighborsClassifier(n_neighbors=k)
clf.fit(X_train, y_train)

# Make predictions on test data
y_pred = clf.predict(X_test)

# Calculate the accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.5714285714285714
```

```python
cols = ['breakaway_percent', 'yco_attempt', 'ypa', 'playoff']
data = rb_playoff_df[cols]

X = data.drop('playoff', axis=1)
y = data['playoff']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = KNeighborsClassifier(n_neighbors=5)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.14285714285714285
```

```
: import matplotlib.pyplot as plt

  accuracies = [0.5714285714285714, 0.14285714285714285]
  labels = ['QB', 'RB']

  fig = plt.figure(figsize=(6, 4))
  plt.bar(labels, accuracies, color='teal')

  plt.xlabel('Position')
  plt.ylabel('Accuracy')
  plt.title('Accuracy of k-NN Classification for QBs vs. RBs')

  plt.show()
```




```
import statsmodels.api as sm

X = rb_playoff_df[['breakaway_percent', 'yco_attempt', 'ypa']]
y = rb_playoff_df['playoff']

X = sm.add_constant(X)

model = sm.OLS(y, X).fit()

print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                playoff   R-squared:                       0.204
Model:                            OLS   Adj. R-squared:                  0.119
Method:                 Least Squares   F-statistic:                     2.398
Date:                Fri, 12 May 2023   Prob (F-statistic):             0.0892
Time:                        14:28:32   Log-Likelihood:                -18.994
No. Observations:                  32   AIC:                             45.99
Df Residuals:                      28   BIC:                             51.85
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.7888      0.562      1.404      0.171      -0.362       1.939
breakaway_percent     0.0139      0.010      1.356      0.186      -0.007       0.035
yco_attempt   -0.7920      0.309     -2.567      0.016      -1.424      -0.160
ypa            0.3720      0.188      1.982      0.057      -0.013       0.757
==============================================================================
Omnibus:                        5.540   Durbin-Watson:                   2.226
Prob(Omnibus):                  0.063   Jarque-Bera (JB):                2.211
Skew:                           0.287   Prob(JB):                        0.331
Kurtosis:                       1.848   Cond. No.                         201.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

```
import statsmodels.api as sm

# Define predictor variables and outcome variable
X = combined_df[['accuracy_percent', 'avg_depth_of_target', 'avg_time_to_throw', 'twp_rate', 'ypa
y = combined_df['playoff']

# Add a constant to the predictor variables
X = sm.add_constant(X)

# Create a logistic regression model
model = sm.Logit(y, X).fit()

print(model.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.485779
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:                playoff   No. Observations:                   32
Model:                          Logit   Df Residuals:                       26
Method:                           MLE   Df Model:                            5
Date:                Fri, 12 May 2023   Pseudo R-squ.:                  0.2808
Time:                        22:20:13   Log-Likelihood:                -15.545
converged:                       True   LL-Null:                       -21.615
Covariance Type:            nonrobust   LLR p-value:                   0.03292
==============================================================================
                             coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const                   -35.0903     25.894     -1.355      0.175     -85.842      15.662
accuracy_percent          0.4690      0.316      1.482      0.138      -0.151       1.089
avg_depth_of_target       0.5615      0.787      0.714      0.476      -0.981       2.104
avg_time_to_throw        -2.4381      2.390     -1.020      0.308      -7.123       2.247
twp_rate                 -0.8651      0.625     -1.384      0.166      -2.090       0.360
ypa                       0.5698      0.846      0.674      0.501      -1.088       2.228
==============================================================================
```

```
import statsmodels.api as sm

# Define predictor variables and outcome variable
X = rb_playoff_df[['breakaway_percent', 'yco_attempt', 'ypa']]
y = rb_playoff_df['playoff']

# Add a constant to the predictor variables
X = sm.add_constant(X)

# Create a logistic regression model
model = sm.Logit(y, X).fit()

print(model.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.564479
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                playoff   No. Observations:                   32
Model:                          Logit   Df Residuals:                       28
Method:                           MLE   Df Model:                            3
Date:                Fri, 12 May 2023   Pseudo R-squ.:                  0.1643
Time:                        22:21:20   Log-Likelihood:                -18.063
converged:                       True   LL-Null:                       -21.615
Covariance Type:            nonrobust   LLR p-value:                   0.06868
==============================================================================
                             coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const                     1.5743      3.123      0.504      0.614      -4.548       7.696
breakaway_percent         0.0675      0.055      1.238      0.216      -0.039       0.174
yco_attempt              -3.9878      1.800     -2.216      0.027      -7.515      -0.461
ypa                       1.8685      1.145      1.631      0.103      -0.376       4.113
==============================================================================
```