



SC1015

Mini-Project

NBA Regular Season & Championship

FCSA Team 4 members:

- Ho Shang Ji Jason
- Zou Xuehao
- Gupta Mannan Mithun

Context on the NBA


**3rd most popular sport globally with
2.2 billion fans**

**Average Of 1.6 MILLION VIEWERS For
the Regular Season(US)**

**10.58 billion U.S. dollars in
Revenue (2022/2023)**



With the goal of winning the NBA Championship



Problem Formulation

Can variables from the regular season
predict whether the team becomes the NBA
Champion?

Problem Definition

- 01** Which variables from the regular season have a relationship with whether the team becomes/is a NBA Champion?
- 02** Is there a change in the league's playstyle over the seasons?
- 03** Can we use Machine Learning techniques to predict NBA Champion using regular season data?

Sample Collection



<https://www.basketball-reference.com/>

2022-23 NBA Season

Standings

Schedule and Results

Leaders

Coaches

Player Stats

Other

2023 Playoffs Summary

Back to top

Western Conference First Round

Phoenix Suns over Los Angeles Clippers (4-1)

Series Stats

Per Game Stats

Share & Export

Glossary

Team	Opponent																								
Rk	Team	G	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%	FT	FTA	FT%	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS	
1	Sacramento Kings*	82	241.8	43.6	88.2	.494	13.8	37.3	.369	29.8	50.9	.586	19.8	25.1	.790	9.5	32.9	42.5	27.3	7.0	3.4	13.5	19.7	120.7	
2	Golden State Warriors*	82	241.8	43.1	90.2	.479	16.6	43.2	.385	26.5	47.0	.564	16.0	20.2	.794	10.5	34.1	44.6	29.8	7.2	3.9	16.3	21.4	118.9	
3	Atlanta Hawks*	82	242.1	44.6	92.4	.483	10.8	30.5	.352	33.9	61.8	.548	18.5	22.6	.818	11.2	33.2	44.4	25.0	7.1	4.9	12.9	18.8	118.4	
4	Boston Celtics*	82	243.7	42.2	88.8	.475	16.0	42.6	.377	26.2	46.2	.567	17.5	21.6	.812	9.7	35.6	45.3	26.7	6.4	5.2	13.4	18.8	117.9	
5	Oklahoma City Thunder*	82	242.1	43.1	92.6	.465	12.1	34.1	.356	31.0	58.5	.530	19.2	23.7	.809	11.4	32.3	43.6	24.4	8.2	4.2	13.0	21.0	117.5	
6	Los Angeles Lakers*	82	242.4	42.9	89.0	.482	10.8	31.2	.346	32.1	57.8	.555	20.6	26.6	.775	10.0	35.7	45.7	25.3	6.4	4.6	14.1	17.9	117.2	
7	Utah Jazz	82	241.5	42.5	89.8	.473	13.3	37.8	.353	29.2	52.0	.560	18.7	23.8	.786	11.8	34.1	45.9	26.0	6.1	5.2	15.4	20.5	117.1	
8	Memphis Grizzlies*	82	241.2	43.7	92.1	.475	12.0	34.2	.351	31.7	57.9	.548	17.5	23.8	.733	12.0	34.6	46.6	26.0	8.3	5.8	13.6	20.0	116.9	
9	Milwaukee Bucks*	82	241.8	42.7	90.4	.473	14.8	40.3	.368	27.9	50.1	.557	16.6	22.4	.743	11.1	37.5	48.6	25.8	6.4	4.9	14.6	18.1	116.9	
10	Indiana Pacers	82	240.9	42.0	89.6	.469	13.6	37.0	.367	28.4	52.6	.540	18.7	23.7	.790	10.1	31.4	41.5	27.0	7.7	5.8	14.9	21.2	116.3	
11	New York Knicks*	82	243.4	42.0	89.4	.470	12.6	35.7	.354	29.4	53.6	.547	19.4	25.5	.761	12.6	34.0	46.6	22.9	6.4	4.1	13.0	20.3	116.0	
12	Denver Nuggets*	82	240.9	43.6	86.4	.504	11.8	31.2	.379	31.8	55.2	.575	16.8	22.4	.751	10.1	32.9	43.0	28.9	7.5	4.5	14.5	18.6	115.8	
13	Minnesota Timberwolves*	82	241.8	42.9	87.4	.490	12.2	33.3	.365	30.7	54.1	.568	17.9	23.7	.755	9.1	32.8	41.9	26.2	8.0	5.4	15.3	21.6	115.8	
14	Philadelphia 76ers*	82	242.4	40.8	83.8	.487	12.6	32.6	.387	28.2	51.2	.551	21.0	25.1	.835	8.7	32.2	40.9	25.2	7.7	4.7	13.7	20.4	115.2	
15	New Orleans Pelicans*	82	242.1	42.0	87.6	.480	11.0	30.1	.364	31.1	57.5	.541	19.3	24.4	.793	10.6	33.1	43.7	25.9	8.3	4.1	14.6	20.5	114.4	
16	Dallas Mavericks	82	243.0	40.0	84.3	.475	15.2	41.0	.371	24.8	43.3	.574	19.0	25.1	.755	7.6	31.2	38.8	22.9	6.3	3.7	12.2	20.7	114.2	
17	Phoenix Suns*	82	241.2	42.1	90.1	.467	12.2	32.6	.374	29.9	57.5	.520	17.2	21.7	.793	11.8	32.4	44.2	27.3	7.1	5.3	13.5	21.2	113.6	
18	Los Angeles Clippers*	82	241.8	41.1	86.1	.477	12.7	33.4	.381	28.4	52.7	.539	18.7	23.9	.781	9.8	33.4	43.2	23.9	7.1	4.4	14.2	19.5	113.6	
19	Portland Trail Blazers	82	240.6	40.5	85.4	.474	12.9	35.3	.365	27.6	50.1	.550	19.6	24.6	.796	9.4	31.1	40.5	24.2	6.7	4.6	14.5	20.0	113.4	
20	Brooklyn Nets*	82	240.6	41.5	85.1	.487	12.8	33.8	.378	28.7	51.3	.559	17.7	22.1	.800	8.2	32.3	40.5	25.5	7.1	6.2	13.7	21.1	113.4	
21	Washington Wizards	82	240.9	42.1	86.9	.485	11.3	31.7	.356	30.9	55.2	.559	17.6	22.1	.800	9.4	34.2	43.6	25.4	6.8	5.2	14.1	18.8	113.2	
22	Chicago Bulls*	82	242.7	42.5	86.8	.490	10.4	28.9	.361	32.1	57.9	.555	17.6	21.8	.809	8.5	33.9	42.4	24.5	7.9	4.5	13.4	18.9	113.1	



NBAdata																																
Rk	Team	Year	G	MP	W	L	WIN%	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%	FT	FTA	FT%	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS	Champion	Division	MVP	
17	Atlanta Hawks*	2023	82	242.1	41	41		0.5	44.6	92.4	0.483	10.8	30.5	0.352	33.9	61.8	0.548	18.5	22.6	0.818	11.2	33.2	44.4	25	7.1	4.9	12.9	18.8	118.4		Southeast Division	
2	Boston Celtics*	2023	82	243.7	57	25	0.695121951	42.2	88.8	0.475	16	42.6	0.377	26.2	46.2	0.567	17.5	21.6	0.812	9.7	35.6	45.3	26.7	6.4	5.2	13.4	18.8	117.9		Atlantic Division		
9	Brooklyn Nets*	2023	82	240.6	45	37	0.548780488	41.5	85.1	0.487	12.8	33.8	0.378	28.7	51.3	0.559	17.7	22.1	0.8	8.2	32.3	40.5	25.5	7.1	6.2	13.7	21.1	113.4		Atlantic Division		
7	Charlotte Hornets	2023	82	241.8	27	55	0.329268293	41.3	90.4	0.457	10.7	32.5	0.33	30.5	57.9	0.528	17.6	23.6	0.749	11	33.5	44.5	25.1	7.7	5.2	14.2	20.3	111		Southeast Division		
19	Chicago Bulls*	2023	82	242.7	40	42	0.487804878	42.5	86.8	0.49	10.4	28.9	0.361	32.1	57.9	0.555	17.6	21.8	0.809	8.5	33.9	42.4	24.5	7.9	4.5	13.4	18.9	113.1		Central Division		
5	Cleveland Cavaliers*	2023	82	242.4	51	31	0.62195122	41.6	85.2	0.488	11.6	31.6	0.367	30	53.6	0.559	17.5	22.5	0.78	9.7	31.4	41.1	24.9	7.1	4.7	13.3	19	112.3		Central Division		
21	Dallas Mavericks	2023	82	243	38	44	0.463414634	40	84.3	0.475	15.2	41	0.371	24.8	43.3	0.574	19	25.1	0.755	7.6	31.2	38.8	22.9	6.3	3.7	12.2	20.7	114.2		Southwest Division		
4	Denver Nuggets*	2023	82	240.9	53	29	0.646341463	43.6	86.4	0.504	11.8	31.2	0.379	31.8	55.2	0.575	16.8	22.4	0.751	10.1	32.9	43	28.9	7.5	4.5	14.5	18.6	115.8	Yes	Northwest Division		
30	Detroit Pistons	2023	82	241.5	17	65	0.207317073	39.6	87.1	0.454	11.4	32.4	0.351	28.2	54.6	0.516	19.8	25.7	0.771	11.2	31.3	42.4	23	7	3.8	15.1	22.1	110.3		Central Division		
11	Golden State Warriors*	2023	82	241.8	44	38	0.536585366	43.1	90.2	0.479	16.6	43.2	0.385	26.5	47	0.564	16	20.2	0.794	10.5	34.1	44.6	29.8	7.2	3.9	16.3	21.4	118.9		Pacific Division		
28	Houston Rockets	2023	82	240.9	22	60	0.268292683	40.6	88.9	0.457	10.4	31.9	0.327	30.2	56.9	0.53	19.1	25.3	0.754	13.4	32.9	46.3	22.4	7.3	4.6	16.2	20.5	110.7		Southwest Division		
23	Indiana Pacers	2023	82	240.9	35	47	0.426829268	42	89.6	0.469	13.6	37	0.367	28.4	52.6	0.54	18.7	23.7	0.79	10.1	31.4	41.5	27	7.7	5.8	14.9	21.2	116.3		Central Division		
	League Average	2023	82	241.8			#DIV/0!	42	88.3	0.475	12.3	34.2	0.361	29.6	54.1	0.548	18.4	23.5	0.782	10.4	33	43.4	25.3	7.3	4.7	14.1	20	114.7				
12	Los Angeles Clippers*	2023	82	241.8	44	38	0.536585366	41.1	86.1	0.477	12.7	33.4	0.381	28.4	52.7	0.539	18.7	23.9	0.781	9.8	33.4	43.2	23.9	7.1	4.4	14.2	19.5	113.6		Pacific Division		
14	Los Angeles Lakers*	2023	82	242.4	43	39	0.524390244	42.9	89	0.482	10.8	31.2	0.346	32.1	57.8	0.556	20.6	26.6	0.775	10	35.7	45.7	25.3	6.4	4.6	14.1	17.9	117.2		Pacific Division		
6	Memphis Grizzlies*	2023	82	241.2	51	31	0.62195122	43.7	92.1	0.475	12	34.2	0.351	31.7	57.9	0.548	17.5	23.8	0.733	12	34.6	46.6	26	8.3	5.8	13.6	20	116.9		Southwest Division		
13	Miami Heat*	2023	82	241.5	44	38	0.536585366	39.2	85.3	0.46	12	34.8	0.344	27.3	50.5	0.54	19.1	23	0.831	9.7	30.9	40.6	23.8	8	3	13.5	18.5	109.5		Southeast Division		
1	Milwaukee Bucks*	2023	82	241.8	58	24	0.707317073	42.7	90.4	0.473	14.8	40.3	0.368	27.9	50.1	0.557	16.6	22.4	0.743	11.1	37.5	48.6	25.8	6.4	4.9	14.6	18.1	116.9		Central Division		
15	Minnesota Timberwolves*	2023	82	241.8	42	40	0.512195122	42.9	87.4	0.49	12.2	33.3	0.365	30.7	54.1	0.568	17.9	23.7	0.755	9.1	32.8	41.9	26.2	8	5.4	15.3	21.6	115.8		Northwest Division		
16	New Orleans Pelicans*	2023	82	242.1	42	40	0.512195122	42	87.6	0.48	11	30.1	0.364	31.1	57.5	0.541	19.3	24.4	0.793	10.6	33.1	43.7	25.9	8.3	4.1	14.6	20.5	114.4		Southwest Division		
8	New York Knicks*	2023	82	243.4	47	35	0.5737170732	42	89.4	0.47	12.6	35.7	0.354	29.4	53.6	0.547	19.4	25.5	0.761	12.6	34	46.6	22.9	6.4	4.1	13	20.3	116		Atlantic Division		
20	Oklahoma City Thunder*	2023	82	242.1	40	42	0.487804878	43.1	92.6	0.465	12.1	34.1	0.356	31	58.5	0.53	19.2	23.7	0.809	11.4	32.3	43.8	24.4	8.2	4.2	13	21	117.5		Northwest Division		
25	Orlando Magic	2023	82	241.2	34	48	0.414634146	40.5	86.3	0.47	10.8	31.1	0.346	29.8	52.5	0.539	19.6	25	0.784	10.2	33.1	43.2	23.2	7.4	4.7	15.1	20.1	114.3		Southwest Division		
3	Philadelphia 76ers*	2023	82	242.4	54	28	0.658536585	40.8	83.8	0.487	12.6	32.6	0.387	28.2	51.2	0.551	21	25.1	0.835	8.7	32.2	40.9	25.2	7.7	4.7	13.7	20.4	115.2		Atlantic Division	Yes	

Data Preparation & Cleaning

Rk	Team	Year	G	MP	W	L	WIN%	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%	FT	FTA	FT%	ORB	DRB	TRB	AST	STL
17.0	Atlanta Hawks*	2023	82	242.1	41.0	41.0	0.5	44.6	92.4	0.483	10.8	30.5	0.352	33.9	61.8	0.548	18.5	22.6	0.818	11.2	33.2	44.4	25.0	7.1
2.0	Boston Celtics*	2023	82	243.7	57.0	25.0	0.695121951	42.2	88.8	0.475	16.0	42.6	0.377	26.2	46.2	0.567	17.5	21.6	0.812	9.7	35.6	45.3	26.7	6.4
9.0	Brooklyn Nets*	2023	82	240.6	45.0	37.0	0.548780488	41.5	85.1	0.487	12.8	33.8	0.378	28.7	51.3	0.559	17.7	22.1	0.800	8.2	32.3	40.5	25.5	7.1
27.0	Charlotte Hornets	2023	82	241.8	27.0	55.0	0.329268293	41.3	90.4	0.457	10.7	32.5	0.330	30.5	57.9	0.528	17.6	23.6	0.749	11.0	33.5	44.5	25.1	7.7
19.0	Chicago Bulls*	2023	82	242.7	40.0	42.0	0.487804878	42.5	86.8	0.490	10.4	28.9	0.361	32.1	57.9	0.555	17.6	21.8	0.809	8.5	33.9	42.4	24.5	7.9



Data
Shape

330 Rows

29 Columns

Legend for variables

G - Games Played

MP - Minutes Played

W - Wins

L - Losses

WIN% - Win Percentage

FG - Field Goals Made

FGA - Field Goals Attempted

FG% - Field Goals Percentage

3P - 3 Pointers Made

3PA - 3 Pointers Attempted

3P% - 3 Pointers Percentage

2P - 2 Pointers Made

2PA - 2 Pointers Attempted

2P% - 2 Pointers Percentage

FT - Free Throws Made

FTA - Free Throws Attempted

FT% - Free Throws Percentage

ORB - Offensive Rebounds

DRB - Defensive Rebounds

TRB - Total Rebounds

AST - Assists

STL - Steals

BLK - Blocks

TOV - Turnovers

PF - Personal Fouls

PTS - Points Scored

Champion - Is Champion or not

Division - Division Team belongs in

MVP - Does the Team has the reigning MVP?

Data Preparation & Cleaning

Data Kept

- Rank
- Year
- Minutes Played
- Win%
- FGA
- FG%
- 3PA
- 3P%
- 2PA
- 2P%
- FTA
- FT%
- ORB
- DRB
- TRB
- AST
- BLK
- TOV
- PF
- PTS
- Champion
- Division
- MVP

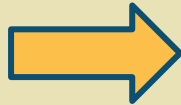
Data Dropped

- Games Played
- Wins
- Loss
- 3-Point Total
- 2-Point Total
- Free Throw Total
- Field Goal Total

Data Preparation & Cleaning

Fill in NaN values and One-Hot Encoding:

Champion	Division	MVP
NaN	Southeast Division	NaN
NaN	Atlantic Division	NaN
NaN	Atlantic Division	NaN
NaN	Southeast Division	NaN
NaN	Central Division	NaN



Champion	Division	MVP
0	Southeast Division	0
0	Atlantic Division	0
0	Atlantic Division	0
0	Southeast Division	0
0	Central Division	0

Standardize all remaining
Team Names and removal
of 'League Average'

```
Team
League Average      11
Boston Celtics*     10
Sacramento Kings    10
Los Angeles Clippers*  9
Orlando Magic        9
Milwaukee Bucks*    9
Detroit Pistons      9
Golden State Warriors* 9
Toronto Raptors*     8
Minnesota Timberwolves 8
Oklahoma City Thunder* 8
New York Knicks      8
Miami Heat*          8
Memphis Grizzlies*   8
Atlanta Hawks*       8
Portland Trail Blazers* 8
Houston Rockets*     8
Phoenix Suns         8
Charlotte Hornets     7
Brooklyn Nets*       7
Indiana Pacers*       7
San Antonio Spurs*   7
Los Angeles Lakers    7
New Orleans Pelicans  6
Utah Jazz*           6
Cleveland Cavaliers  6
Washington Wizards    6
Dallas Mavericks*    6
Denver Nuggets*      6
Philadelphia 76ers*   6
Cleveland Cavaliers* 5
Washington Wizards*   5
Dallas Mavericks      5
Chicago Bulls         5
Utah Jazz             5
Denver Nuggets        5
Philadelphia 76ers    4
New Orleans Pelicans* 4
Los Angeles Lakers*   4
Indiana Pacers        4
San Antonio Spurs     4
Miami Heat            3
Memphis Grizzlies     3
Brooklyn Nets         3
Atlanta Hawks         3
Phoenix Suns*         3
Minnesota Timberwolves* 3
Portland Trail Blazers 3
Oklahoma City Thunder 3
Houston Rockets       3
New York Knicks*      3
Los Angeles Clippers  2
Milwaukee Bucks       2
Toronto Raptors       2
Detroit Pistons*      2
Orlando Magic*        2
Golden State Warriors 1
Sacramento Kings*     1
Charlotte Hornets*    1
Boston Celtics        1
Charlotte Bobcats*    1
Charlotte Bobcats     1
New Orleans Hornets   1
Name: count, dtype: int64
```

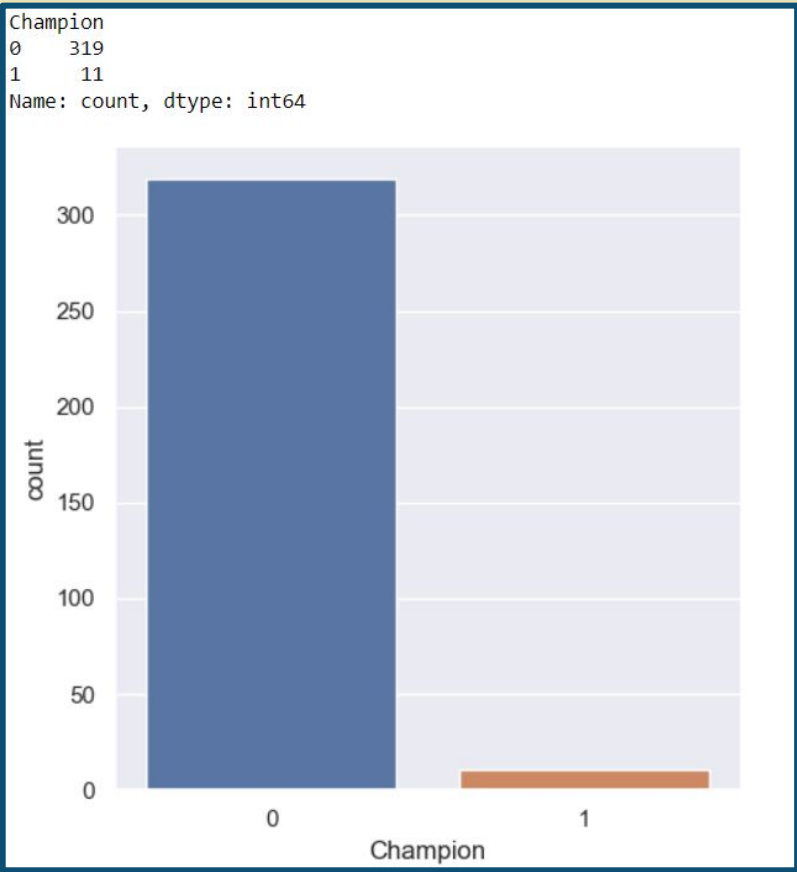


```
Team
Atlanta Hawks      11
Boston Celtics      11
Utah Jazz          11
Toronto Raptors     11
San Antonio Spurs   11
Sacramento Kings    11
Portland Trail Blazers 11
Phoenix Suns        11
Philadelphia 76ers   11
Orlando Magic        11
Oklahoma City Thunder 11
New York Knicks     11
New Orleans Pelicans 11
Minnesota Timberwolves 11
Milwaukee Bucks     11
Miami Heat          11
Memphis Grizzlies   11
Los Angeles Lakers   11
Los Angeles Clippers 11
Indiana Pacers       11
Houston Rockets      11
Golden State Warriors 11
Detroit Pistons      11
Denver Nuggets       11
Dallas Mavericks     11
Cleveland Cavaliers  11
Chicago Bulls        11
Charlotte Hornets     11
Brooklyn Nets        11
Washington Wizards   11
Name: count, dtype: int64
```




Exploratory Data Analysis & Visualization

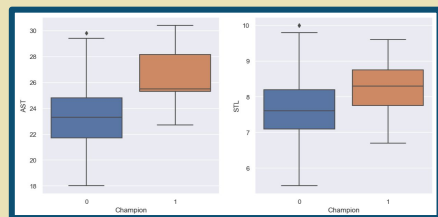




Here we have
319: Non-Champion
11: Champion

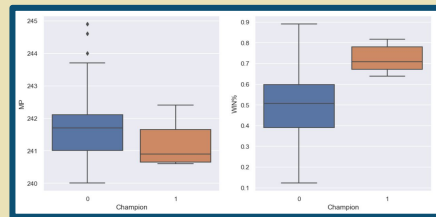
There is an imbalance in our data, 29:1,
hence we will have to deal with that later
on when doing Machine Learning

Here we plot numeric variables against Champion variable (Categorical)



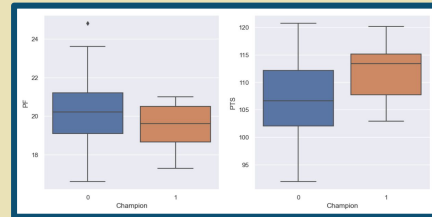
AST

STL



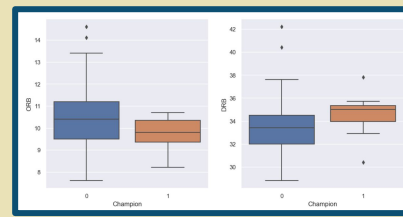
MP

WIN%



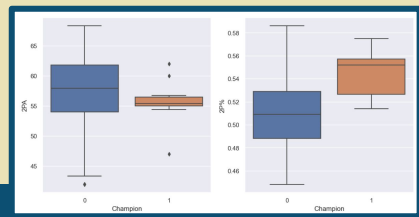
PF

PTS



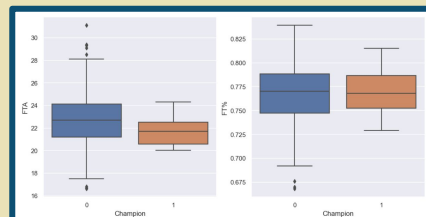
ORB

DRB



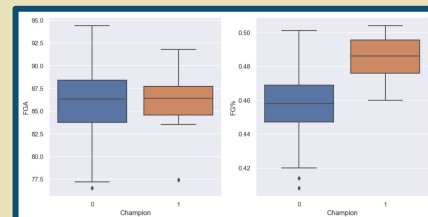
2PA

2P%



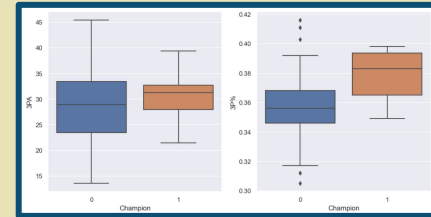
FTA

FT%



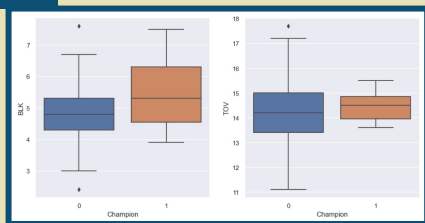
FGA

FG%



3PA

3P%



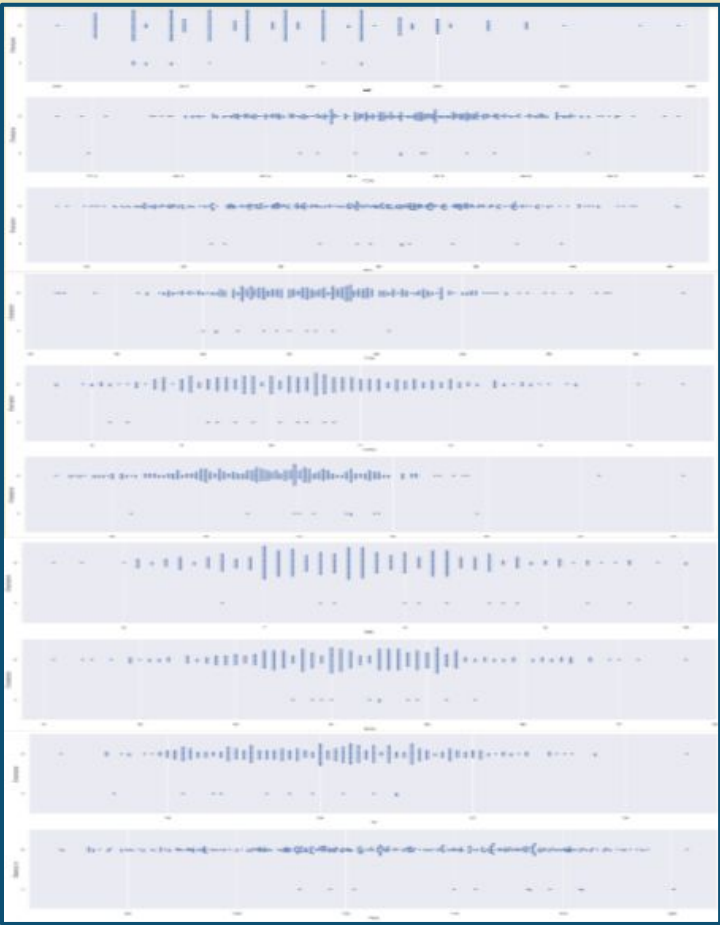
BLK

TOV

There are some clear cut relationship between the variable(s) and Champion. For example this few variables are in general higher for Champions than Non-Champions: WIN%,FG%,3P%,2P%,AST,BLK.

Exploratory Data Analysis & Visualization

There are some ambiguous relationship from the Box plot, so we will use more detailed plots (Swarm and Violin) to analyse the remaining variables



MP

FGA

3PA

2PA

FTA

ORB

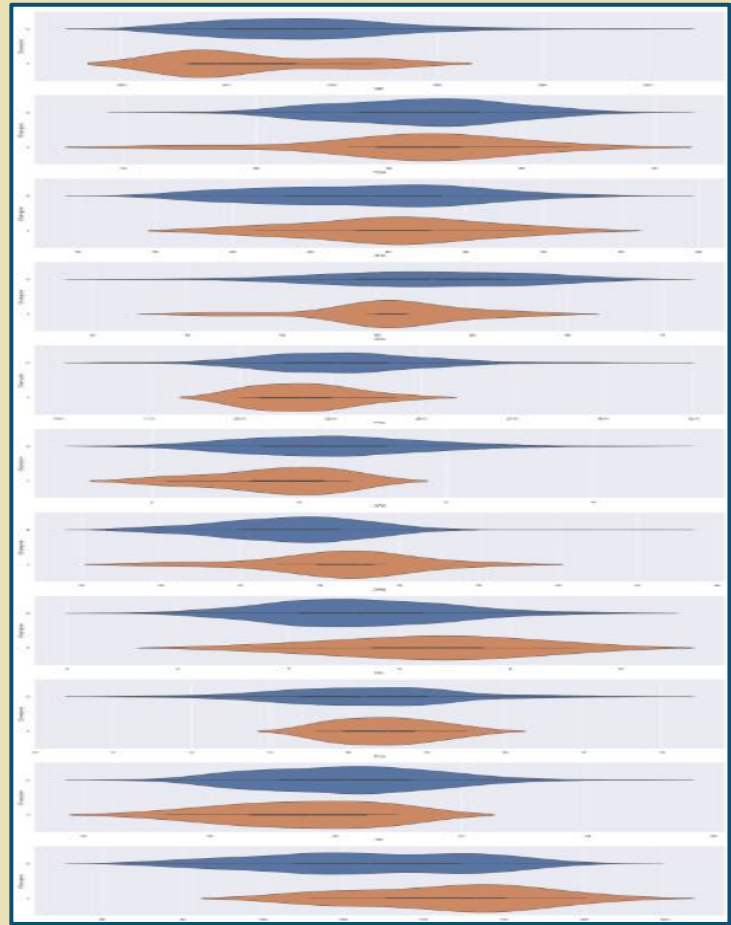
DRB

STL

TOV

PF

PTS



In general we can see that the variables DRB,MP,STL,PTS might have a relationship with the variable Champion



From the above, we decided to narrow our scope and believe that these few variables have a significant relationship with Champion:

WIN%,FG%,3P%,2P%,AST,BLK,DRB,MP,STL,PTS

But we do need further analysis to confirm:

Point-Biserial Correlation

```
# Calculate Point-Biserial Correlation coefficient for FG%
point_biserial_corr, p_value = stats.pointbiserialr(DATA['Champion'], DATA['FG%'])

print("Point-Biserial Correlation Coefficient:", point_biserial_corr)
print("P-value:", p_value)
```

Point-Biserial Correlation Coefficient: 0.2975178098004865
P-value: 3.60000425809278e-08

The Point-Biserial Correlation Coefficient is about 0.3 which mean there is a moderate positive correlation between FG% and Champion.

The P-value is also very close to 0 and less than 0.05, meaning that the Point-Biserial Correlation Coefficient is statistically significant.

```
# Calculate Point-Biserial Correlation coefficient for 3P%
point_biserial_corr, p_value = stats.pointbiserialr(DATA['Champion'], DATA['3P%'])

print("Point-Biserial Correlation Coefficient:", point_biserial_corr)
print("P-value:", p_value)
```

Point-Biserial Correlation Coefficient: 0.24255112271987178
P-value: 8.344103047801263e-06

The Point-Biserial Correlation Coefficient is about 0.25 which mean there is a moderate positive correlation between 3P% and Champion.

The P-value is also very close to 0 and less than 0.05, meaning that the Point-Biserial Correlation Coefficient is statistically significant.


From using Point-Biserial Correlation, we determine that these variables have a strong enough relationship with Champion:

FG%,WIN%,3P%,2P%,AST,BLK,STL,DRB




Data Driven Insights

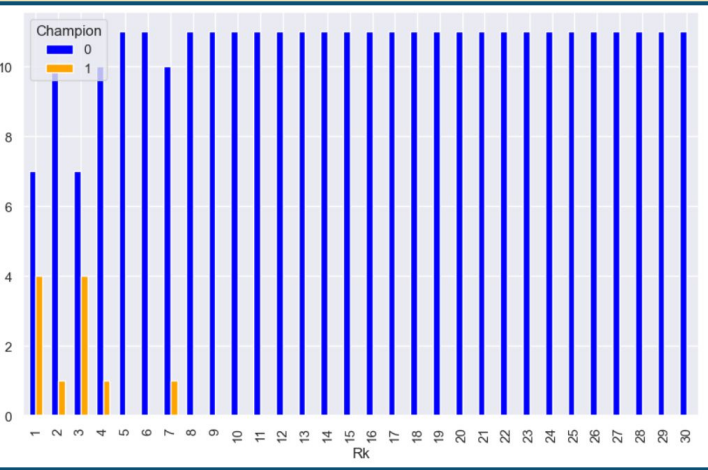
From observing the box-plots, swarm & violin plots and using deeper analysis using Point-Biserial Correlation, we have come to some insights below:



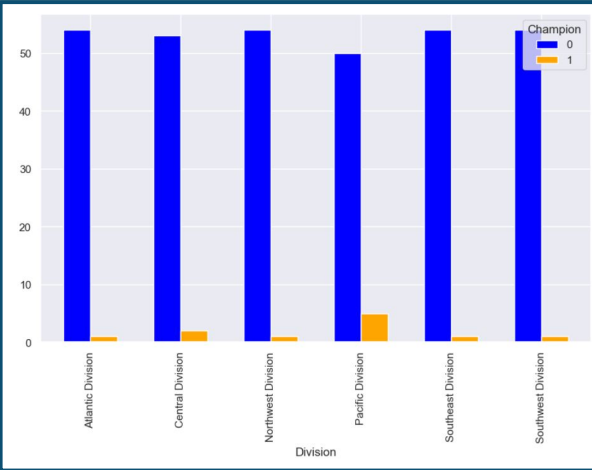
In general, the higher the values of these variables (FG%,WIN%,3P%,2P%,AST,BLK,STL,DRB), the higher odd of the team becoming/is an NBA Champion



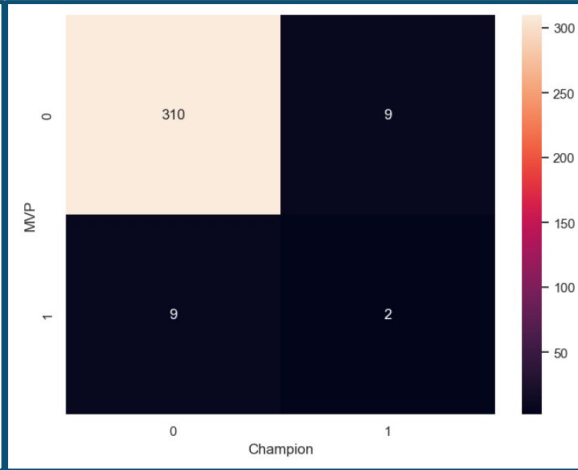
Here we plot categorical variables against Champion variable (Categorical)



Rank



Division



MVP

Higher Rank, Higher number of champions

Seems like no relationship

Rate of Champion (having MVP) higher than (not having MVP)

Further Analysis

As per usual, we need further analysis to confirm the relationship with Champion

Hence we will be looking at using **Phi's Coefficient** & **Cramér's V**

```
#V =  $\sqrt{X^2/N}$  /  $\min(C-1, R-1)$ 
#X2: It is the Chi-square statistic
#N: It represents the total sample size
#R: It is equal to the number of rows
#C: It is equal to the number of columns
```

```
X2 = stats.chi2_contingency(table)[0]
N = np.sum(table).sum()
minimum_dimension = min(table.shape)-1
```

```
# Calculate Cramer's V
V = np.sqrt((X2/(N*minimum_dimension)))
```

```
# Print the result
print(V)
```

```
0.5145364820822323
```

With Cramér's V being about 0.51, it shows that there is a moderately strong association between Rk and Champion, and following-up from the above, in general the lower the Rk the higher odd of being Champion

```
#Obtain the values in each box
```

```
b00 = table.iloc[0, 0]
b01 = table.iloc[0, 1]
b10 = table.iloc[1, 0]
b11 = table.iloc[1, 1]

nume = (b11 * b00) - (b01 * b10)
deno = np.sqrt((b11 + b10) * (b00 + b01) * (b10 + b00) * (b01 + b11))
phi = nume / deno
print("Phi Coefficient: ")
print(phi)
```

```
Phi Coefficient:
0.1536050156739812
```

A Phi Coefficient of 0.154 suggests a weak positive association between the 2 variables.

A positive value indicates that when one variable, MVP, is true, the other variable, Champion, is also more likely to be true, and vice versa.

With Cramér's V being about 0.51, it shows that there is a moderately strong association between Rk and Champion, and following-up from the above, in general the lower the Rk the higher odd of being Champion

```
#Making the table
table = pd.crosstab(DATA['Division'], DATA['Champion'])
```

```
#V =  $\sqrt{X^2/N}$  /  $\min(C-1, R-1)$ 
#X2: It is the Chi-square statistic
#N: It represents the total sample size
#R: It is equal to the number of rows
#C: It is equal to the number of columns
```

```
X2 = stats.chi2_contingency(table)[0]
N = np.sum(table).sum()
minimum_dimension = min(table.shape)-1
```

```
# Calculate Cramer's V
V = np.sqrt((X2/(N*minimum_dimension)))
```

```
# Print the result
print("Cramer's V: ")
print(V)
print("P-value: ")
print(stats.chi2_contingency(table)[1])
```

```
Cramer's V:
0.14813363449166975
P-value:
0.20529863908011306
```

From the above we can see the Cramér's V is around 0.15, which will mean a weak association between Division and Champion, but, as you can see the P-value is way above 0.05, indicating that there is no statistically significant association between the Division and Champion variables.


A Phi Coefficient of 0.154 suggests a weak positive association between the 2 variables. A positive value indicates that when one variable, MVP, is true, the other variable, Champion, is also more likely to be true, and vice versa.

From the above we can see the Cramér's V is around 0.15, which will mean a weak association between Division and Champion, but, as you can see the P-value is way above 0.05, indicating that there is no statistically significant association between the Division and Champion variables.




Data Driven Insights

From observing the relevant plots and using deeper analysis using Phi's Coefficient & Cramér's V, we have come to some insights below:

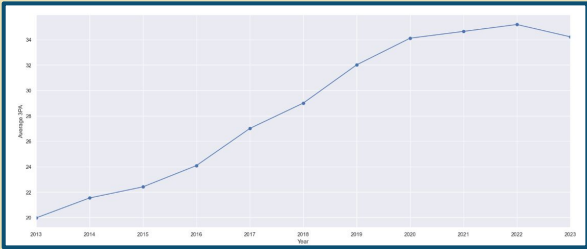


In general, the lower the Rank, and having the MVP contributes to higher odds of winning the NBA Championship

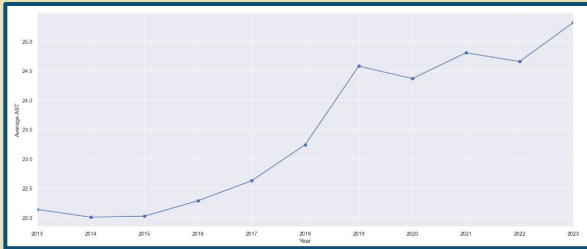


Time-Series

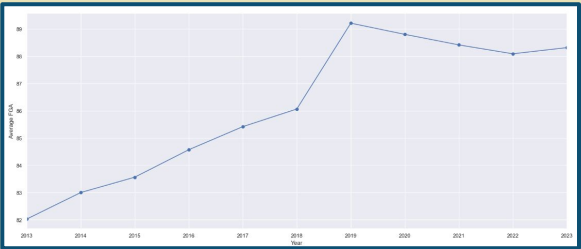
To observe if there is a trend of team play styles throughout the years



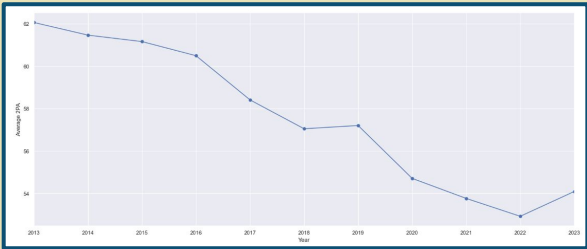
3PA



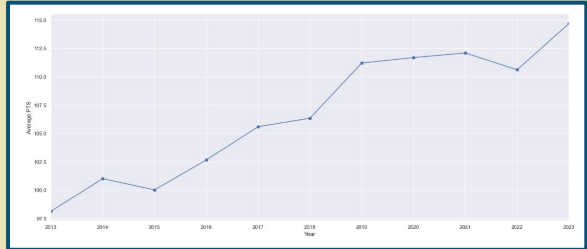
AST



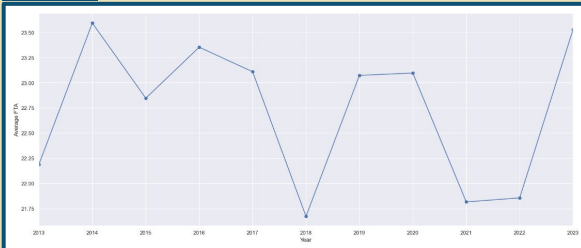
FGA



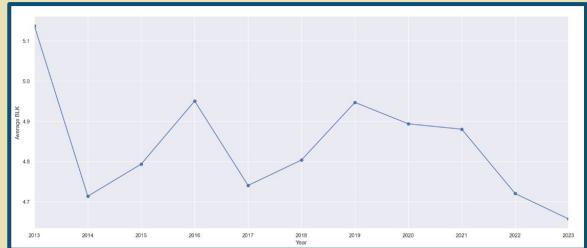
2PA



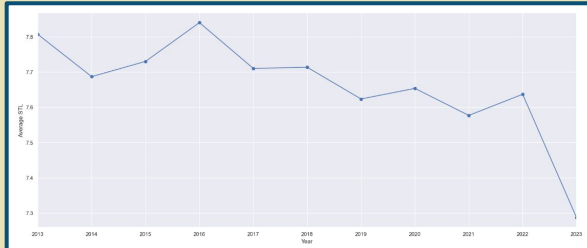
PTS



FTA



BLK



STL



Analysis of all time-series & Data Driven Insights

Upward trend: FGA, 3PA, AST, PTS

Teams have placed more importance onto these variables across the years, this could mean they are paramount to winning an NBA Championship





Downward trend: 2PA, BLK, STL


Teams have placed less importance onto these variables across the years, they might be less important in contributing to a team winning the Championship

No trend: FTA






Machine Learning





Machine Learning

- Multivariate Classification Tree
 - Random Forest
 - Cost-Sensitive Support Vector Machine
- 



Multivariate Classification Tree Model


Creates a binary tree to classify data into different classes
(Champion or Not Champion)

Response Variable : **Champion**



Predictor Feature :

**Rk,MVP,FG%,WIN%,3P%,2P%,AST,BLK,S
TL,DRB**



In order to obtain the most optimal depth for our tree, we used k-fold cross-validation to do so.

The tree produced with depth = 6

```
#In order to see which depth is the most optimal, we will compare the k-fold cross-validation(k = 5) from depths 1 to 7
i = 1
maxdepth = 7
bestscore = 0

for i in range(1,maxdepth):

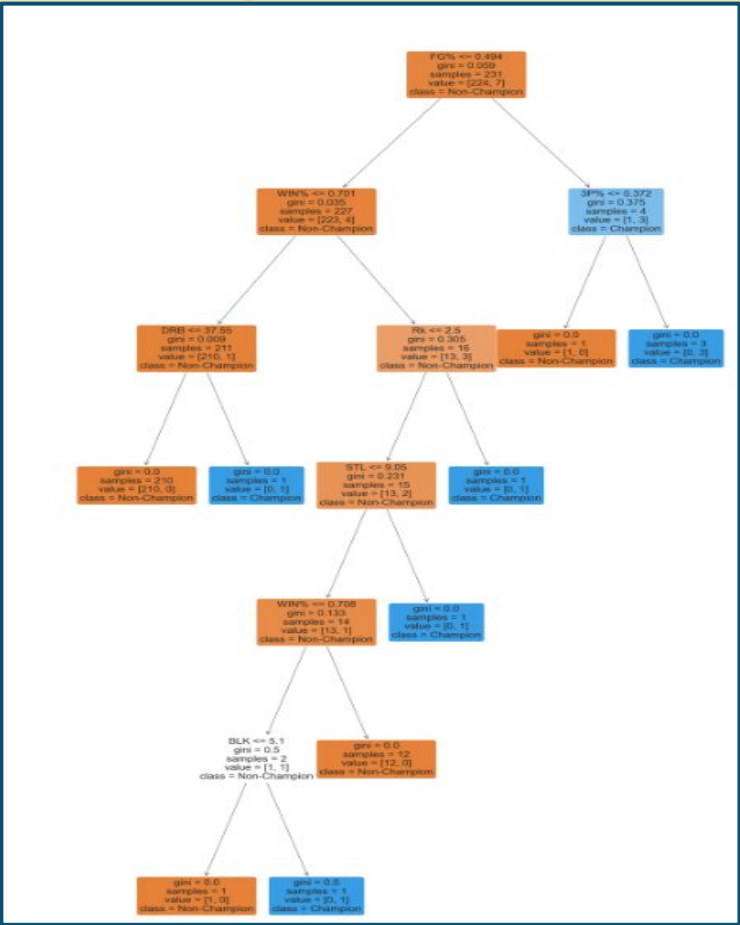
    # Create and train decision tree classifier
    dtree = DecisionTreeClassifier(max_depth=i)

    # 5-fold cross-validation and its mean
    scores = cross_val_score(dtree, X_train, y_train, cv=5)
    meanscore = np.mean(scores)

    # Update best depth if current depth gives better mean-score
    if (meanscore > bestscore):
        bestdepth = i

# Step 4: Select optimal depth
print(f"Optimal Depth: {bestdepth}")

Optimal Depth: 6
```



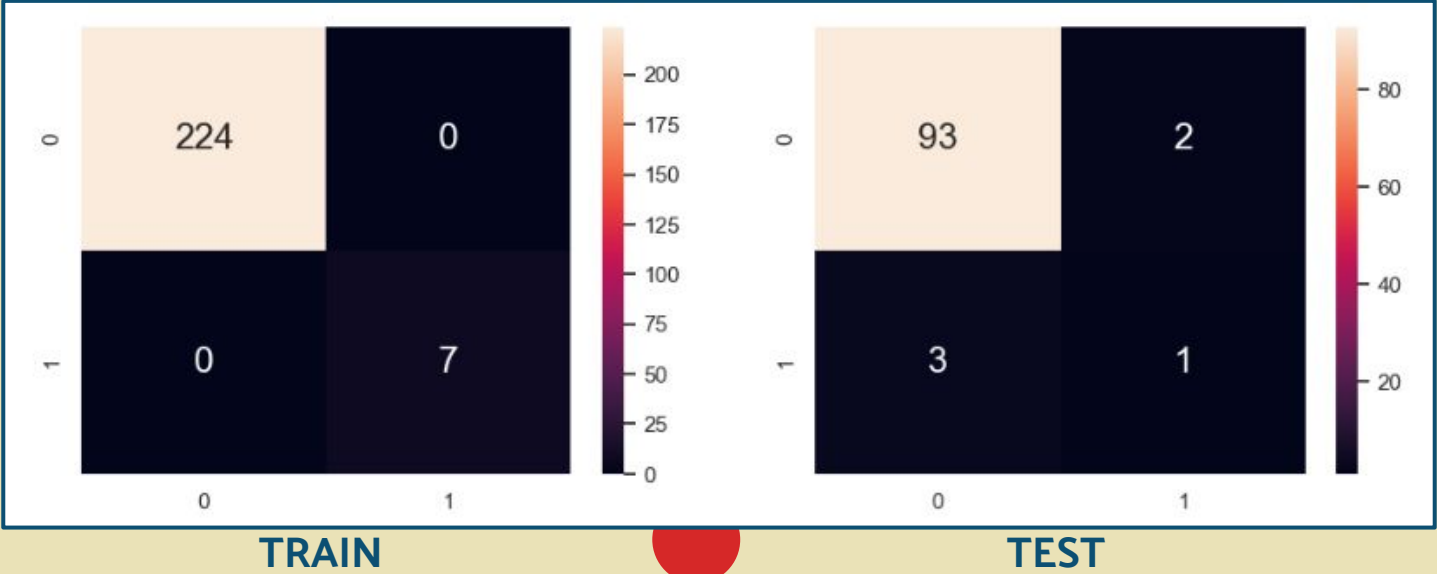
Analysis

Goodness of Fit of Model	Test Dataset
Classification Accuracy:	0.9494949494949495
True Positive Rate :	0.25
True Negative Rate :	0.9789473684210527
False Positive Rate:	0.021052631578947368
False Negative Rate:	0.75

We can evidently see that the Classification accuracy is very high, infact hovering above **0.9** on the test dataset.

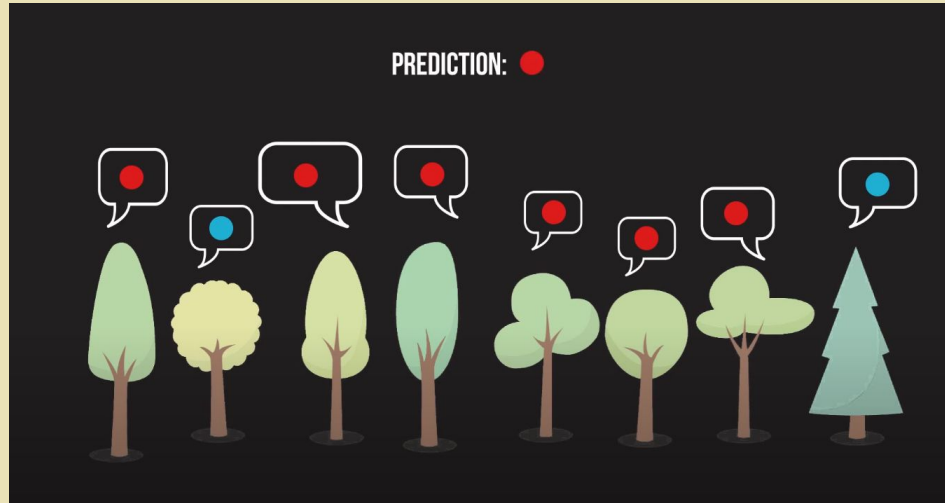
But the True Positive rate is around **0.25**, which is very low and not optimal. This indicates that our model is correctly identifying only **25%** of the positive instances in the test dataset.

On the other hand, the False Negative Rate is too high at around **0.75**, this means that the model is failing to detect a significant portion, **75%**, of the positive instances.



Random Forest Classification Model

Creates an ensemble of independent decision trees using randomly selected subsets of the training data





Random Forest Classification Model



1. One-Hot-Encoding

2. Resampling

3. Apply the Random Forest Classification Model



Random Forest Classification Model

1. One-Hot-Encoding

Variables that are not useful for this model

- Team
- Year
- Division

```
Index: 330 entries, 0 to 340
Data columns (total 25 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Rk           330 non-null    int64
1   Team        330 non-null    object
2   Year        330 non-null    datetime64[ns]
3   MP           330 non-null    float64
4   WIN%         330 non-null    float64
5   FGA          330 non-null    float64
6   FG%          330 non-null    float64
7   3PA          330 non-null    float64
8   3P%          330 non-null    float64
9   2PA          330 non-null    float64
10  2P%          330 non-null    float64
11  FTA          330 non-null    float64
12  FT%          330 non-null    float64
13  ORB          330 non-null    float64
14  DRB          330 non-null    float64
15  TRB          330 non-null    float64
16  AST          330 non-null    float64
17  STL          330 non-null    float64
18  BLK          330 non-null    float64
19  TOV          330 non-null    float64
20  PF           330 non-null    float64
21  PTS          330 non-null    float64
22  Champion     330 non-null    int64
23  Division    330 non-null    object
24  MVP          330 non-null    int64
dtypes: datetime64[ns](1), float64(19), int64(3), object(2)
```



Random Forest Classification Model

1. One-Hot-Encoding

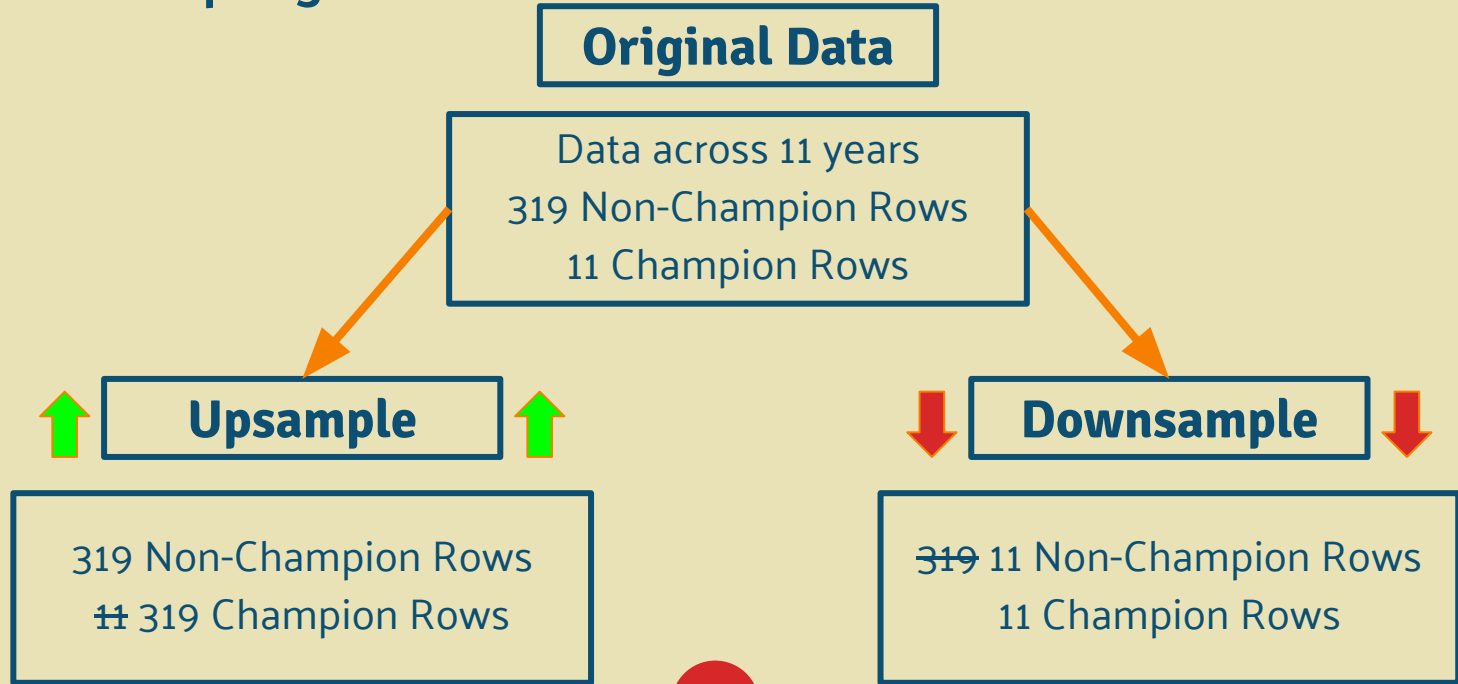
2. Resampling

3. Apply the Random Forest Classification Model



Random Forest Classification Model

2. Resampling





Random Forest Classification Model

1. One-Hot-Encoding
 2. Resampling
 3. Apply the Random Forest Classification Model
- 

Random Forest Classification Model

3. Apply the Random Forest Classification Model

- Train/Test Split - 70:30

```
# Split the Dataset into Train and Test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3) path 4
```

- RandomForestClassifier

RandomForestClassifier(max_depth=4, n_estimators=1000)
- ```
Get feature importances
feature_importances = pd.Series(rforest.feature_importances_, index=X.columns)

Predict the Response corresponding to Predictors
y_train_pred = rforest.predict(X_train)

Print the Classification Accuracy
print("Train Data")
print("Accuracy : \t", rforest.score(X_train, y_train))
print()

Print the Accuracy Measures from the Confusion Matrix
cmTrain = confusion_matrix(y_train, y_train_pred)
tpTrain = cmTrain[1][1] # True Positives : Good (1) predicted Good (1)
fpTrain = cmTrain[0][1] # False Positives : Bad (0) predicted Good (1)
tnTrain = cmTrain[0][0] # True Negatives : Bad (0) predicted Bad (0)
fnTrain = cmTrain[1][0] # False Negatives : Good (1) predicted Bad (0)

es.sort_values(ascending=False)
```

# Random Forest Classification Model

## 3. Apply the Random Forest Classification Model



**Upsampled Data**



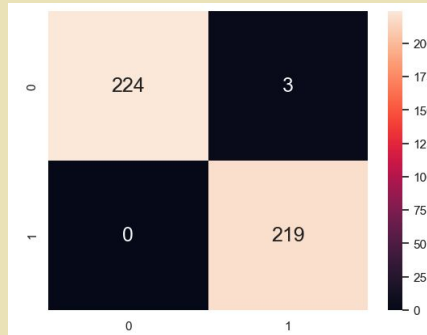
### Top 5 Variables

```
WIN% 0.242806
Rk 0.213070
FG% 0.112286
AST 0.078121
2P% 0.060992
dtype: float64
```

```
Train Data
Accuracy : 0.9932735426008968

TPR Train : 1.0
TNR Train : 0.986784140969163

FPR Train : 0.013215859030837005
FNR Train : 0.0
```

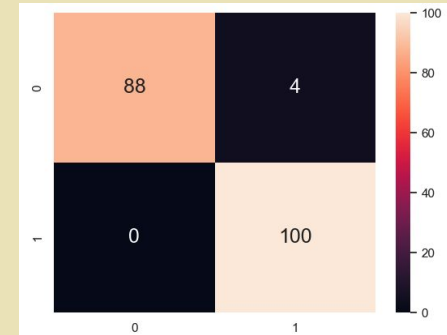


### Prediction Accuracy

```
Test Data
Accuracy : 0.9791666666666666

TPR Test : 1.0
TNR Test : 0.9565217391304348

FPR Test : 0.043478260869565216
FNR Test : 0.0
```





# Random Forest Classification Model

## 3. Apply the Random Forest Classification Model

Downsampled Data

### Top 5 Variables

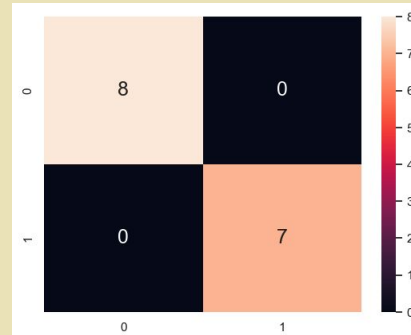
```
WIN% 0.197609
Rk 0.193465
FG% 0.106884
DRB 0.070316
3P% 0.062216
dtype: float64
```

### Prediction Accuracy

```
Train Data
Accuracy : 1.0

TPR Train : 1.0
TNR Train : 1.0

FPR Train : 0.0
FNR Train : 0.0
```



```
Test Data
Accuracy : 0.8571428571428571

TPR Test : 1.0
TNR Test : 0.6666666666666666

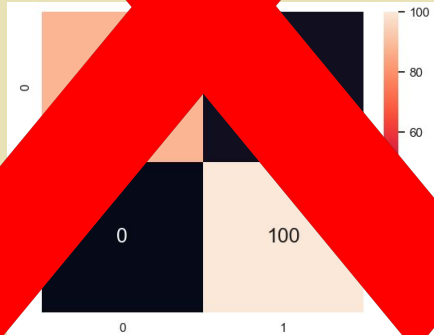
FPR Test : 0.3333333333333333
FNR Test : 0.0
```



# Random Forest Classification Model

**Upsampled Data**

Test Data Accuracy : 0.979166  
TPR Test : 1.0  
TNR Test : 0.94348  
FPR Test : 0.0869565216  
FNR Test : 0.0



Analysis

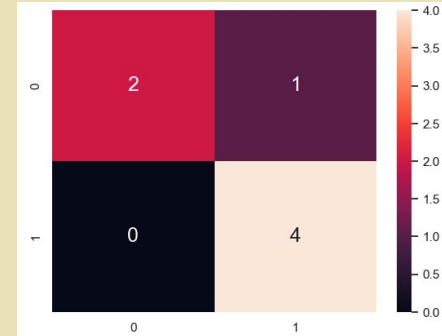
**Introduce Bias  
& Noise**

**High accuracy**

**Can be improved by  
expanding dataset**

**Downsampled Data**

Test Data Accuracy : 0.8571428571428571  
TPR Test : 1.0  
TNR Test : 0.6666666666666666  
FPR Test : 0.3333333333333333  
FNR Test : 0.0



# Cost-sensitive support vector machines

A cost-sensitive Support Vector Machine adjusts misclassification costs to reflect real-world imbalances, allowing different penalties for different types of errors, thus optimizing classification in scenarios where the costs of misclassification vary.

## Advantages:

**Increased Sensitivity to Rare Events:** Helps in predicting rare events-in this case 'champions'

**Improved Performance for Imbalanced Data:** Cost-sensitive SVMs are particularly advantageous when dealing with imbalanced datasets where one class significantly outnumbers the other

**Predicting Features:** "Rk", "MVP", "FG%", "WIN%", "3P%", "2P%", "AST", "BLK", "STL", "DRB"

# Applying cost sensitive SVM

## 1. Copying 'DATA'

```
#Copy of DATA
df = DATA

selected_features = ["Rk", "MVP", "FG%", "WIN%", "3P%", "2P%", "AST", "BLK", "STL", "DRB"]
X = df[selected_features]
y = df['Champion']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)
```

## 2. Scaling

```
This ensures that all features are on the same scale and prevents the features with larger scales
#from dominating those with smaller scales during model training (good for SVM)
scaler = StandardScaler()
scaler.fit(X_train) # Fit scaler on training data only
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

### 3.Setting weight

```
Here, we are setting the weight for class 1, which is Champions, to be 10 times higher than the weight for class 0, Non-Champi
cost_weights = {0: 1, 1: 29}
#The above is to make our SVM more cost-sensitive, infact we made 'Champions' "29 times heavier" to make our SVM more optimal
```

### 4.Training the model

```
Train the Cost-Sensitive SVM Model using the kernel=poly and class_weight = cost_weights
model = SVC(kernel='poly', class_weight=cost_weights)
model.fit(X_train_scaled, y_train)
```

```
SVC(class_weight={0: 1, 1: 29}, kernel='poly')
```

# Classification report

```
y_pred = model.predict(X_test_scaled)

Generate the classification report
report = classification_report(y_test, y_pred)
print("Classification Report for weighted Data:\n", report)
```

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0            | 1.00      | 0.98   | 0.99     | 82      |
| 1            | 0.33      | 1.00   | 0.50     | 1       |
| accuracy     |           |        | 0.98     | 83      |
| macro avg    | 0.67      | 0.99   | 0.74     | 83      |
| weighted avg | 0.99      | 0.98   | 0.98     | 83      |

## Analysis

- The overall accuracy of the model on the test set is approximately 98%, which indicates that the majority of predictions made by the model are correct.
- For class 0 (Non-Champions), the precision is 1.00 and the recall is 0.98. This means that among the instances predicted as Non-Champions, 100% are actually Non-Champions, and the model correctly identifies 98% of the actual Non-Champions.
- For class 1 (Champions), the recall is 1.00. This indicates that the model correctly identifies ALL 100% of the actual Champions.
- The weighted average F1-score, which takes into account the class imbalance, is approximately 0.98. This score reflects the balance between precision and recall, indicating that the model performs very well in terms of both minimizing false positives and false negatives.
- In conclusion, the model demonstrates strong performance in accurately classifying instances of both classes, especially when considering the class imbalance. The high F1-score and accuracy indicate that the model is effective in distinguishing between Champions and Non-Champions.

# Visualisation and Importance

THE FOLLOWING IS JUST TO SHOW A SIMPLIFIED VISUALIZATION OF A REDUCED MODEL.

To give a visual representation of our model, we are going to be basing of only those 2 variables to give a rough idea and visualization. (As trying to use all 10 variables will result in a 10-Dimensional plot which in our case is impossible to plot properly without diving into very complex theory)

Finding top 2 features:

```
#Code to GET top 2 features:

Get the indices of support vectors
support_vector_indices = model.support_

Get the support vectors
support_vectors = X_train_scaled[support_vector_indices]

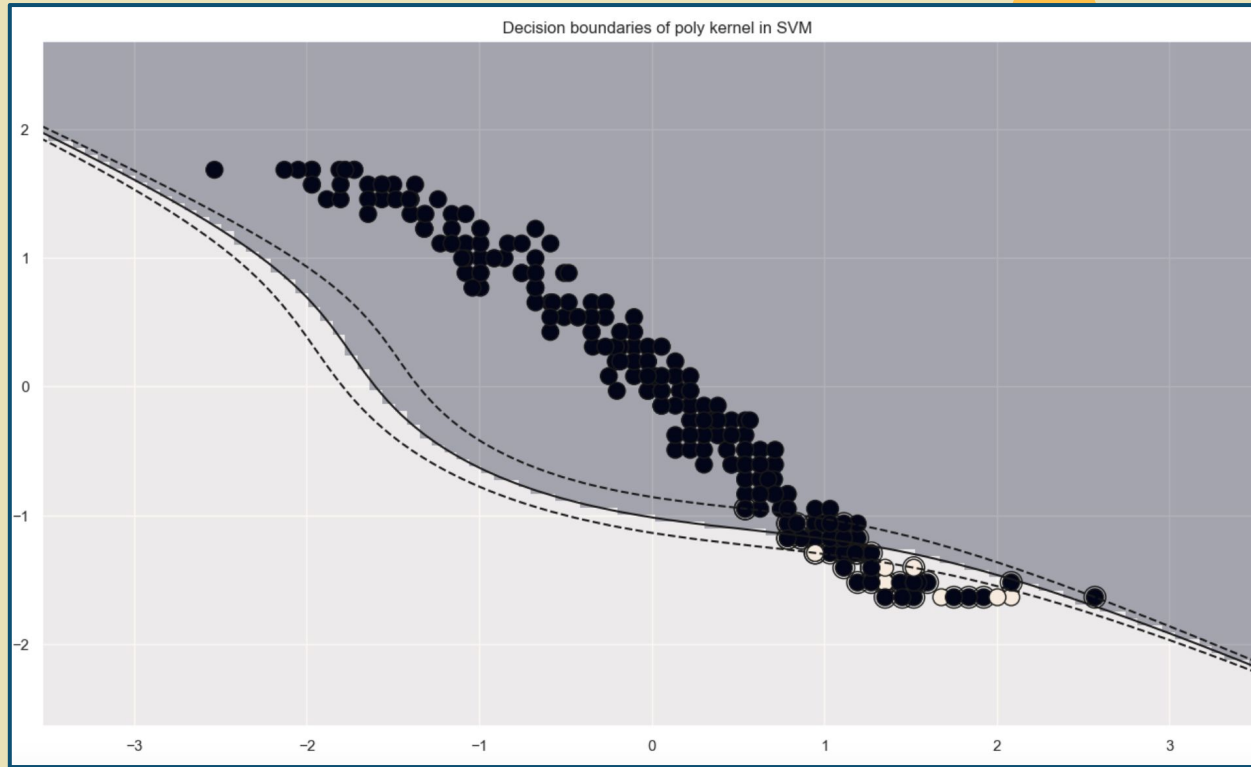
Calculate the importance of features based on the magnitude of support vectors
feature_importance = np.abs(support_vectors).mean(axis=0)

Get the indices of top 2 features
top2_indices = np.argsort(feature_importance)[-2:]

Map indices to feature names
top2_features = [selected_features[i] for i in top2_indices]

print("Top 2 features:", top2_features)

Top 2 features: ['WIN%', 'RK']
```



**This is to show how SVM does classification using our TRAIN data. The black circles are Non-Champions and the white circles are Champions. The polynomial curve shows the classification done by our model to try to classify the circles(teams) into Champions or Non-champions.(On the TEST data of course)**





## Upsampled Data



### Report

| Classification Report for Upsampled Data: |           |        |          |         |
|-------------------------------------------|-----------|--------|----------|---------|
|                                           | precision | recall | f1-score | support |
| 0                                         | 1.00      | 0.93   | 0.96     | 81      |
| 1                                         | 0.93      | 1.00   | 0.96     | 79      |
| accuracy                                  |           |        | 0.96     | 160     |
| macro avg                                 | 0.96      | 0.96   | 0.96     | 160     |
| weighted avg                              | 0.97      | 0.96   | 0.96     | 160     |



## Downsampled Data



### Report

| Classification Report for Downsampled Data: |           |        |          |         |
|---------------------------------------------|-----------|--------|----------|---------|
|                                             | precision | recall | f1-score | support |
| 0                                           | 0.33      | 0.50   | 0.40     | 2       |
| 1                                           | 0.67      | 0.50   | 0.57     | 4       |
| accuracy                                    |           |        | 0.50     | 6       |
| macro avg                                   | 0.50      | 0.50   | 0.49     | 6       |
| weighted avg                                | 0.56      | 0.50   | 0.51     | 6       |

## Upsampled Data

- Overall accuracy of the model on the test set: approximately **96%**, indicating a majority of correct predictions.
- Class 0 (Non-Champions) metrics:
  - Precision: 1.00 Recall: 0.93
  - Interpretation: 100% of predictions as Non-Champions are correct, and 93% of actual Non-Champions are identified.
- Class 1 (Champions) metrics:
  - Precision: 0.93 Recall: 1.00
  - Interpretation: 93% of predictions as Champions are correct, and all actual Champions are identified.

Weighted average F1-score: around 0.96, reflecting a balanced trade-off between precision and recall considering class imbalance.

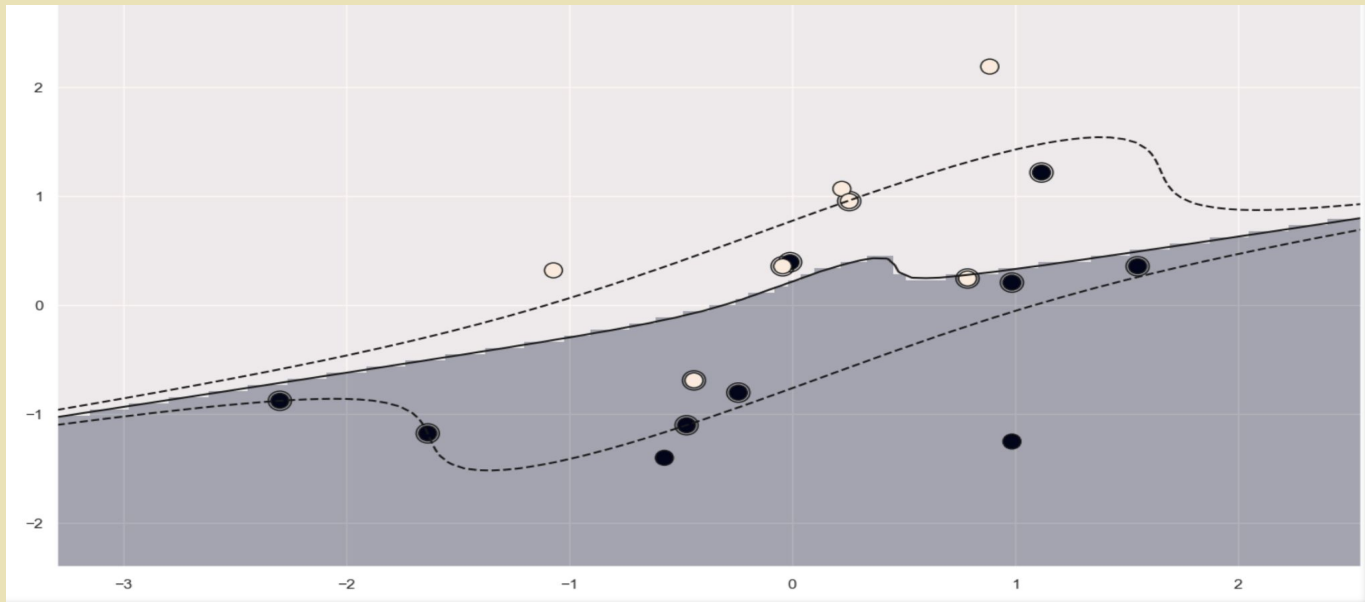
- Conclusion: The model demonstrates strong performance in accurately classifying both classes, particularly considering class imbalance. It exhibits high F1-score and accuracy, effectively distinguishing between Champions and Non-Champions. Comparing with a model using cost\_weights, this model may perform even better, with superior precision, recall, and F1-score for class 1 (Champions).

## Downsampled Data

- Overall accuracy of the model on the test set: approximately **50%**, indicating only half of predictions are correct.
- Class 0 (Non-Champions) metrics:
  - Precision: 0.33 Recall: 0.50
  - Interpretation: 33% of predictions as Non-Champions are correct, and the model identifies only 50% of actual Non-Champions.
- Class 1 (Champions) metrics:
  - Precision: 0.67 Recall: 0.50
  - Interpretation: 67% of predictions as Champions are correct, and the model identifies only 50% of actual Champions.
- Weighted average F1-score: approximately 0.51, indicating poor balance between precision and recall considering class imbalance.
- Conclusion: The model demonstrates mediocre to poor performance in accurately classifying both classes, particularly when considering class imbalance. Both F1-score and accuracy suggest the model is ineffective in distinguishing between Champions and Non-Champions. Compared to the previous models, this one performs the worst.

# Visualisation and Importance

Decision boundaries of poly kernel in SVM



By looking at this plot, perhaps when we down sample, it might be better to use lesser variables to predict Champions as the plot, from a visual perspective, seem to perform better than our original model using all 10 features.

# Overall Analysis




- Obviously the model where we down-sampled our data performed the poorest. While the first model with cost\_weights attached and the model where we up-sampled our data perform quite similarly and both performed excellently.
- The model trained on the up-sampled data, which artificially increased the number of instances in the minority class, performed similarly to the model with class weights attached. However, the reliance on up-sampling to address class imbalance may introduce biases into the model and may not be the most robust approach for predicting NBA Champions for the current season.
- In conclusion, based on our SVM machine learning experiments, we believe that the model trained on the original dataset with cost weights attached is the most suitable for predicting NBA Champions. This model effectively balances predictive performance with considerations for class imbalance, resulting in reliable predictions for the outcome of the NBA season.



# Machine Learning Conclusion, Data Driven Insights & Recommendations

After working with 3 different types of Machine Learning models (Multivariate Classification Tree, Random Forest & Support Vector Machine), we come to the conclusion that our Cost-Sensitive Support Vector Machine is arguably the best model in predicting NBA Champion.

What we would recommend is to use our SVM model to predict NBA Champion.



Additionally, we can see that the variable WIN% is constantly rank top 2 in the importance of variable to the models for ALL 3 MODELS, this probably means that WIN% is the most important factor in determining which team becomes/is an NBA Champion.

Hence what we would recommend is for teams to recruit more players with higher WIN%. To go further, teams can look into WIN SHARES as well, to see how much said player contributes to the WINS of their teams.

