Soccer Game Predictor

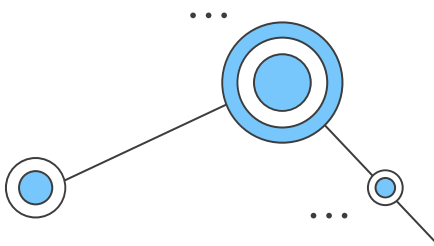
By Brandon Zau, Sam Camargo, Ray Soda

Brandon

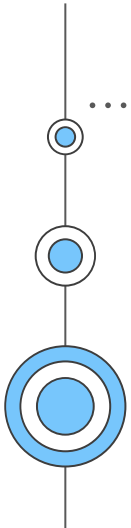
Computer/Data Science

Sam

Mechatronic Engineering
and Finance or Marketing



Ray



Finance

What is soccer? How do I play?-Ray

- Originated in China B.C
- Spread in Europe
- Simple rule that anyone can understand
- You can play it whenever you have a ball
- There are 250 million players in the world
- Team Sports

Champions League

Group Stage

Tournament

- A conference to determine the top European teams
- Annual world-class competition
- A lot of money moves every game.

Teams and Players-Ray

- These three people are in the spotlight now.(same team)
- Paris Saint-Germain is the best team in the world (I think)

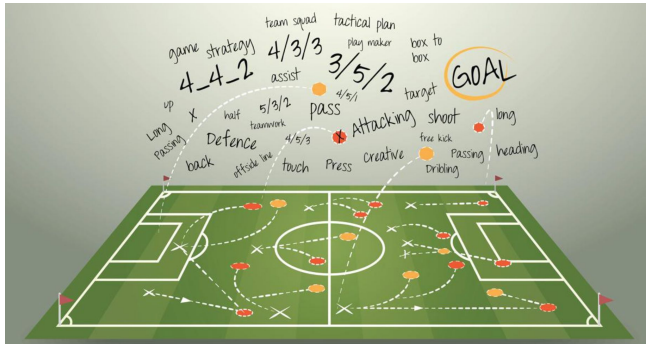
FIFA Stats for Players

- Every player has a FIFA card that represents them and their measured values
- This data can be used to make a very detailed prediction of a team's performance
- This data only affects the performance of a player and the team, not the results
- We chose to use team-level statistics because these are related directly to the number of goals

Research Question and Project Overview

Can we predict the # of goals of a futbol team using multivariable regression? (50%)

- Identity and analyze soccer statistics to improve game strategy



- Tools used: Tableau for Data Visualization

Our Dataset

We used 4 datasets: Campions_League_2021-2022.csv (from kaggle)

Attacking.csv (from kaggle)
Defending.csv (from kaggle)
goals.csv (from kaggle)

FIFA World Cup.csv (from kaggle)

Categories

Team Name	Wins	Draws	Loses	Points	Qualified	Goals	Goals conc.	Possession	Passing acc (%)	Balls rec	Tackles won	Clean sheet	Saves	Distance

Yellow card	Red card	Assists	Attacks	Clear chances	Penalties	Corners	Offsides	Runs into	Runs into	Passes comp	Crosses comp

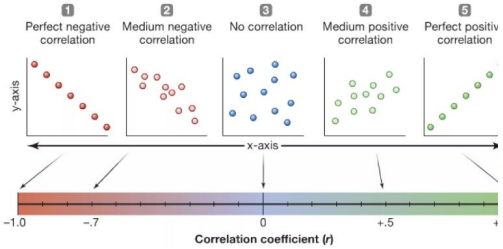
Crosses comp	Free-kicks	Blocks	Clearance	Attempts conc	High claim	Low claim	Punches	Fouls	Fouls suff

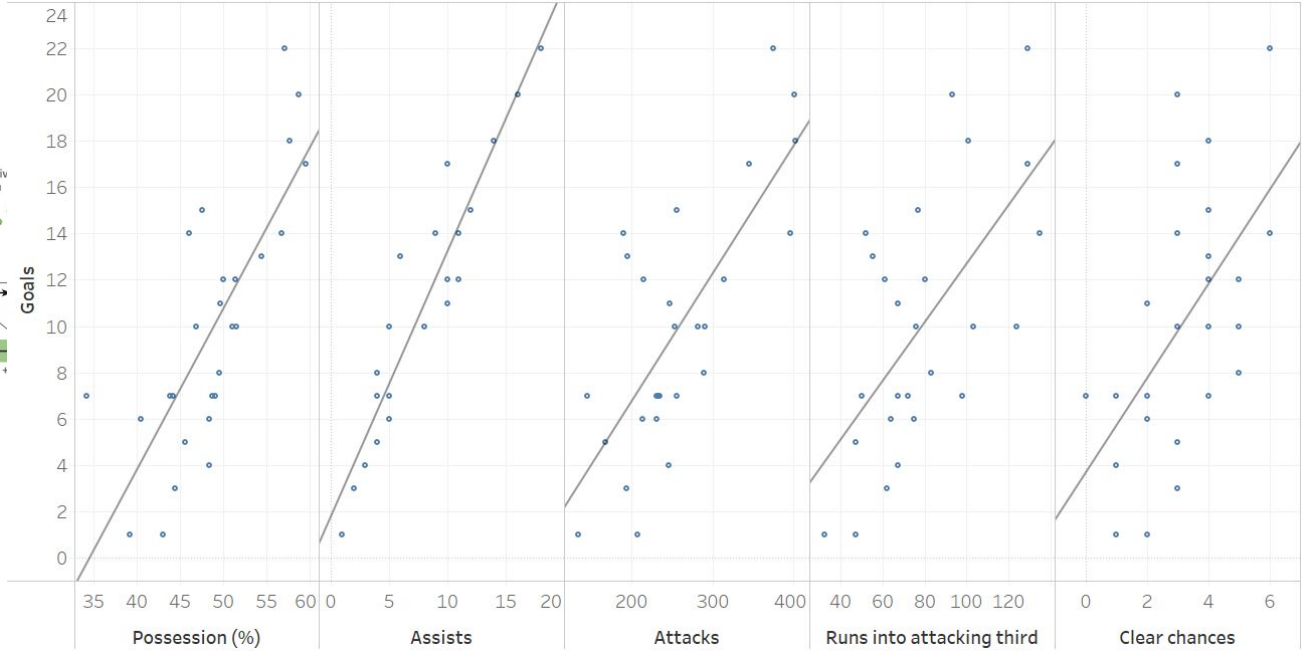
The Data-Sam

- Goals Vs Goals conceded
- Assists
- Attacks
- Clear chances
- Possession (%)
- Accuracy
- Clean Sheets
- Runs into key play area

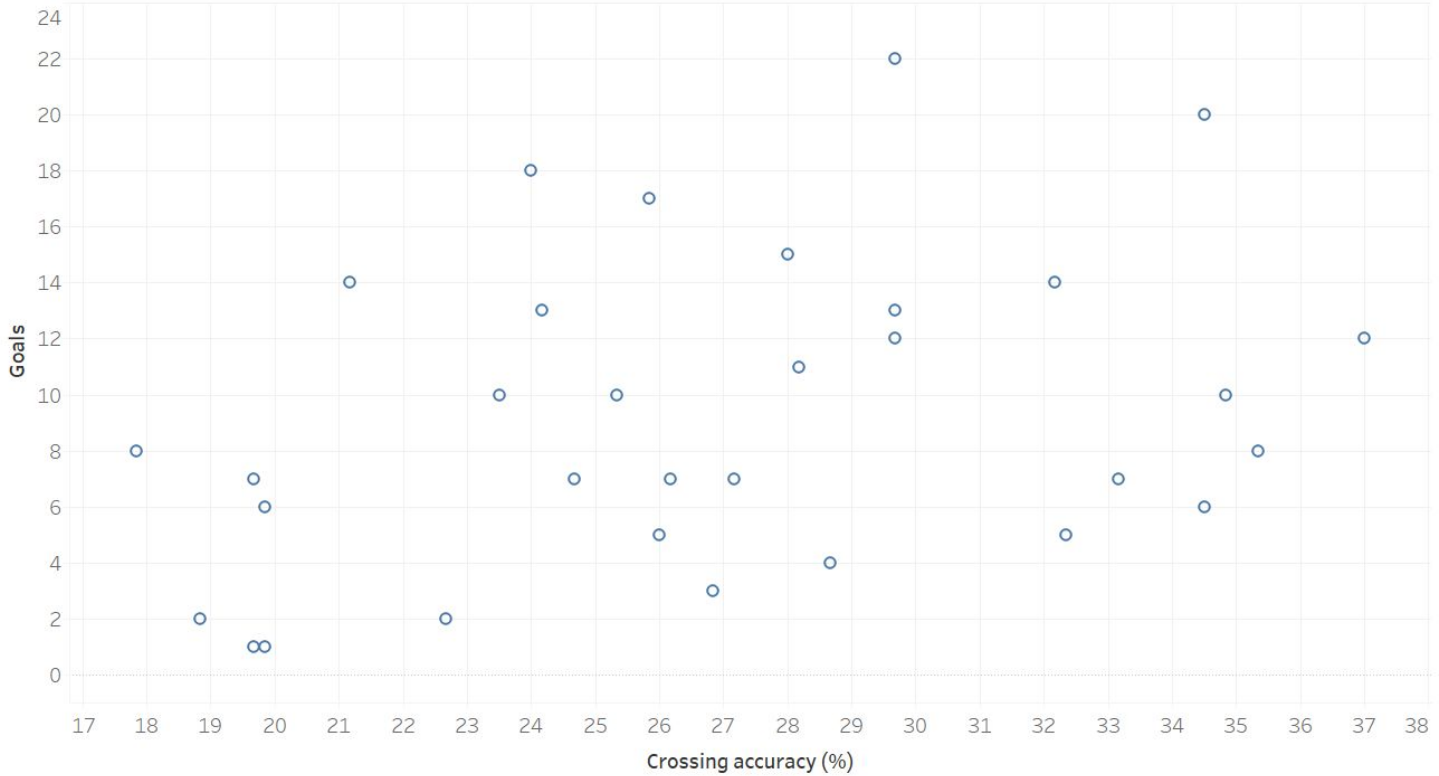
The Data (cont.)-correlation

Most defined correlations between values to the number of goals scored

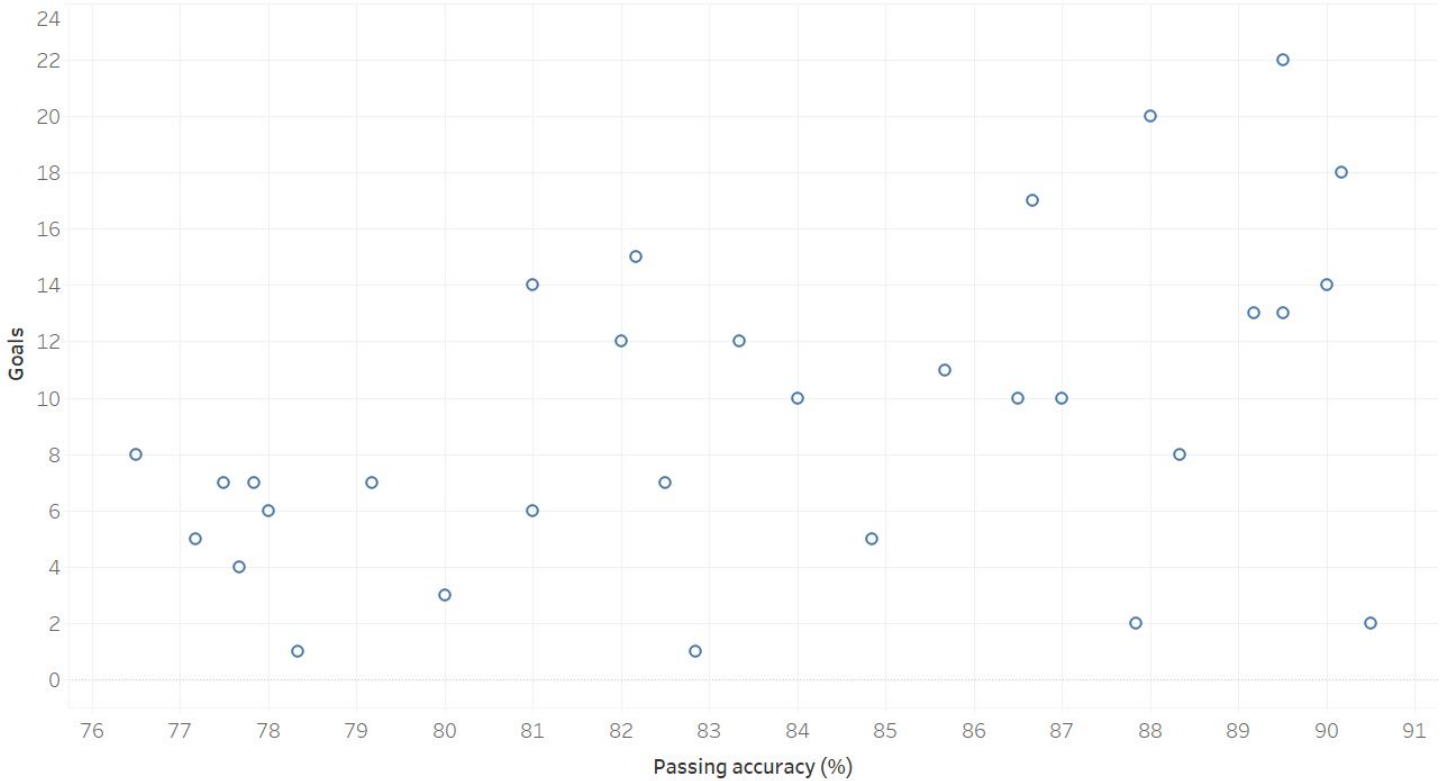




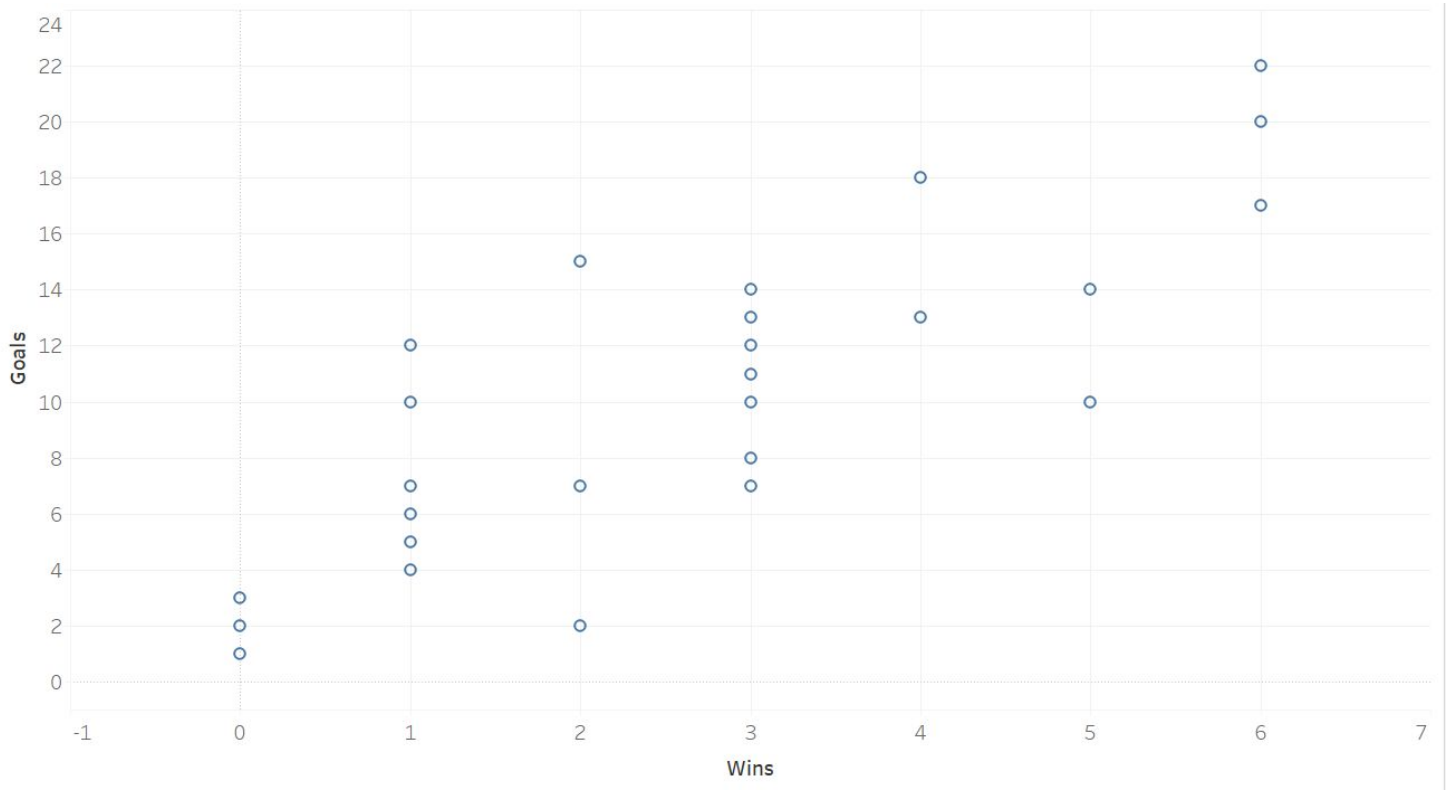
Goals Vs Crossing Accuracy(%)



Passing Accuracy(%) vs. Goals



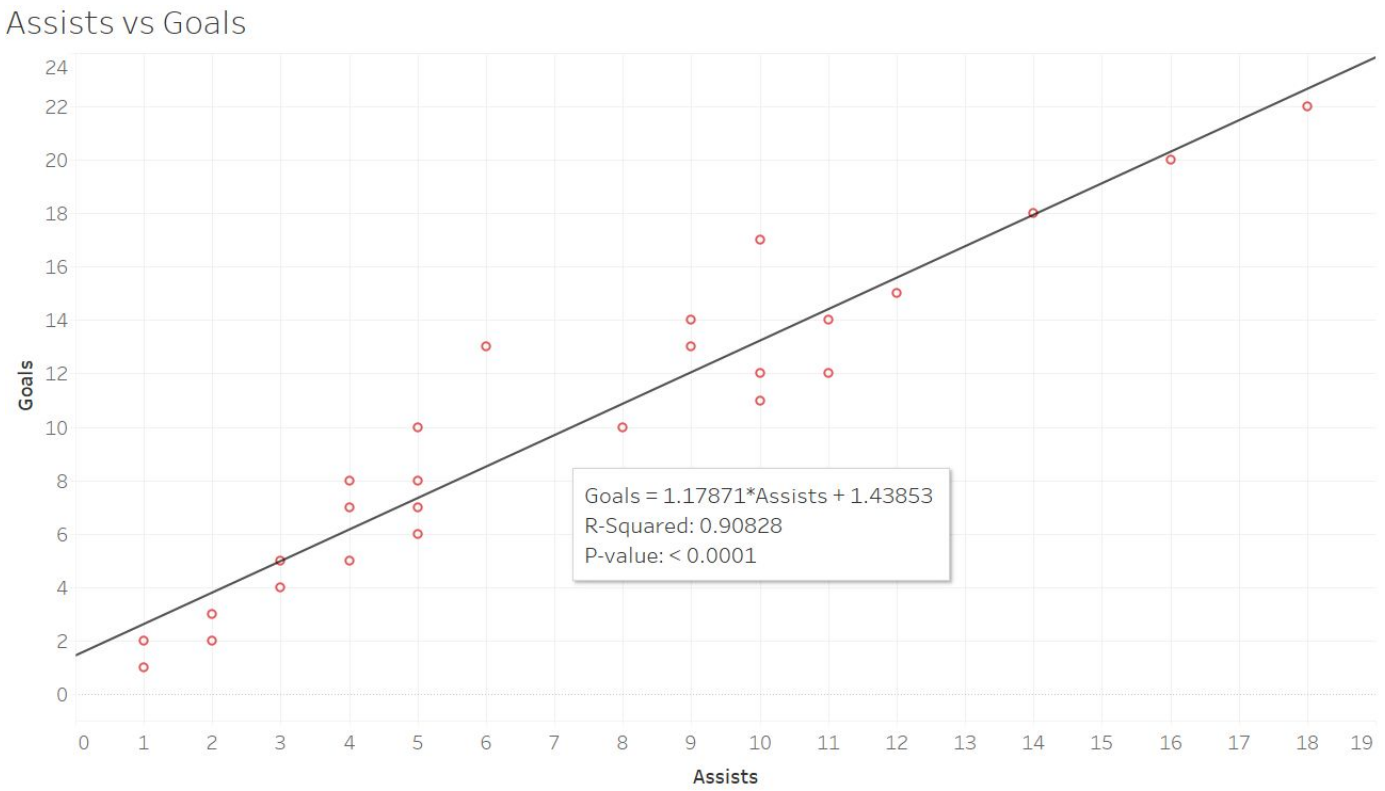
Wins vs Goals



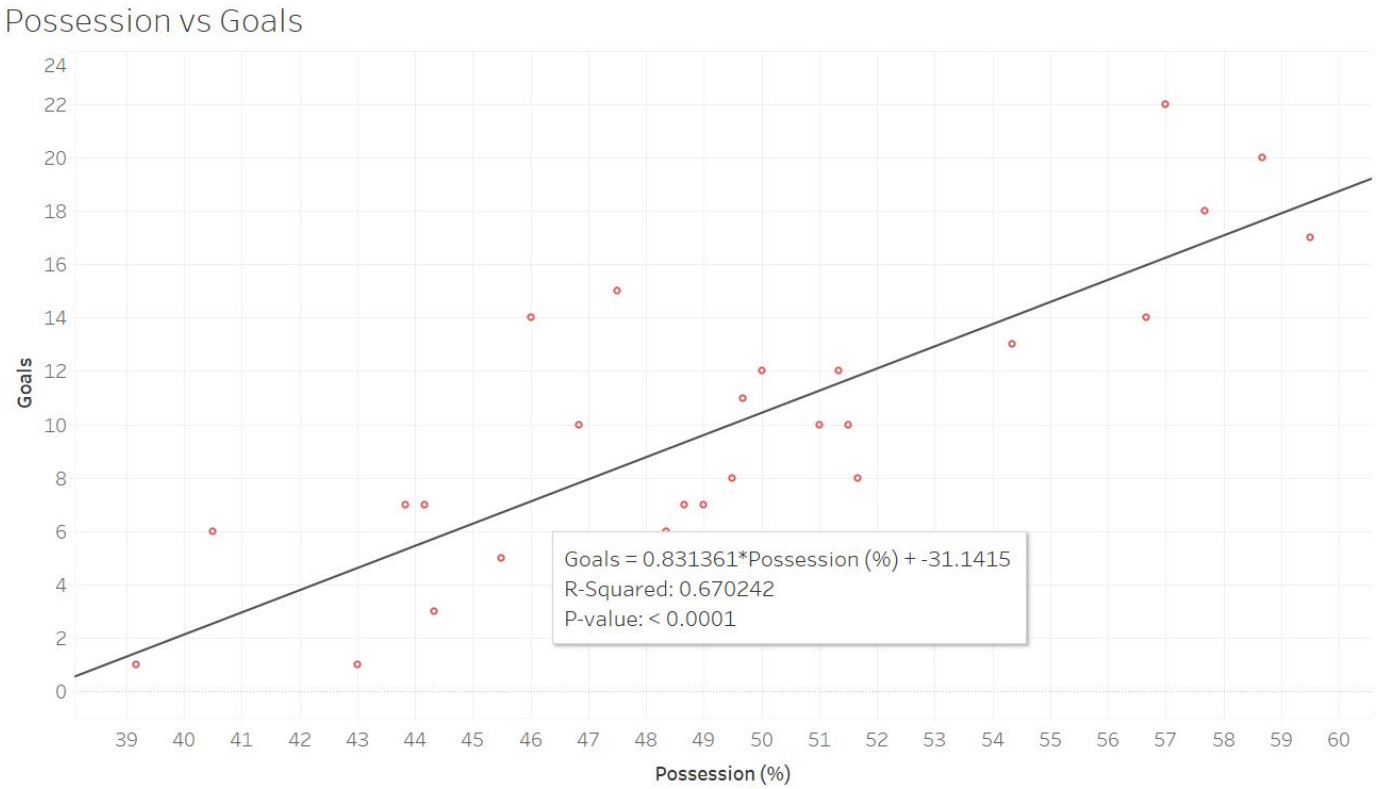
Choosing a value

- The number of goals is equal to the score of a team
- Many graphs have little to no relation between the value and the number of goals
- Other values have a clear relation between the number of goals and itself
- These values can be used to predict the score of a game

Assists vs Goals Graph



Possession vs Goals Graph



Purpose of this data

- We use the Champions league to understand correlations between measured values and the number of total goals scored in a team

- We will then apply the information from these datasets to other matches such as the World Cup 2022

  For regression, our Champions League Dataset did not include the right values, so we had to use a FIFA World Cup Dataset instead.

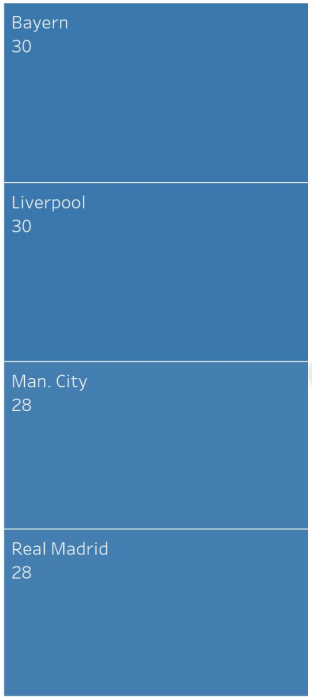
- Thi will also allow us to remove any biases that reside in the Champions League and purify the correlations of major league soccer matches statistics

The Data: From a Coach's Eye

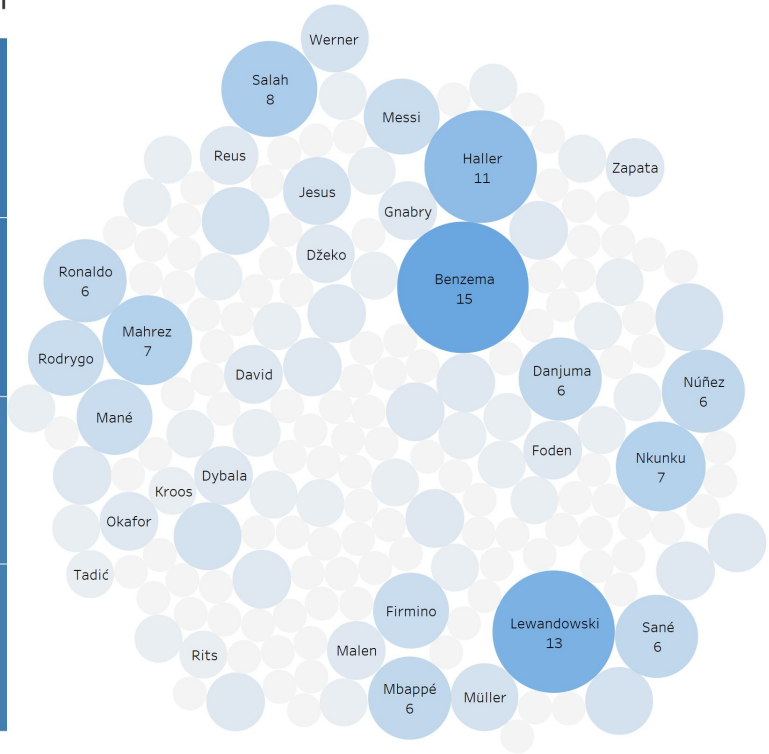
Used a total of 3 datasets:

- attacking.csv
- defending.csv
- goals.csv

Clubs and tl

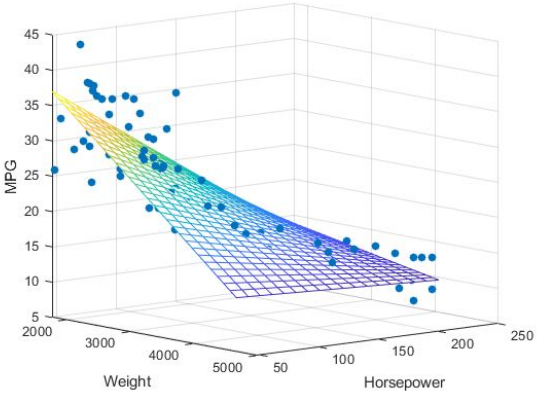


Players and their Goals



Multivariable Regression

- Multivariable regression-models the relationship between a dependent variable (i.e. an outcome of interest) and more than 1 independent variable.
- We modeled goals (dependent variable) to 'ashotsOnTarget','hsaves','ashots','aPossession','asaves' (indep.)



Importing libraries, creating our dataframe

- Imported libraries
- Assigned our dataset

```
import pandas as pd
import numpy as np
from sklearn import preprocessing
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
import seaborn as sb

football=pd.read_csv("FIFAallMatchBoxData.csv")
football.head(3)
print(football.shape)
print(list(football.columns))
football.head()
#we can create a multi-linear regression with indepedent variables of aPossesion/hPossesion, hshots/ashots
```

```
(1482, 19)
['year', 'hname', 'aname', 'hgoals', 'agoals', 'hPossesion', 'aPossesion', 'hshotsOnTarget', 'ashotsOnTarget', 'hshots', 'as
hots', 'hyellowCards', 'ayellowCards', 'hredCards', 'aredCards', 'hfouls', 'afouls', 'hsaves', 'asaves']
```

	year	hname	aname	hgoals	agoals	hPossesion	aPossesion	hshotsOnTarget	ashotsOnTarget	hshots	ashots	hyellowCards	ayellowCards	hre
0	2012	Mexico	Costa Rica	1	0	0	0	0	0	0	0	1	2	
1	2012	Antigua and Barbuda	United States	1	2	24	76	2	4	14	17	1	1	
2	2012	United States	Guatemala	3	1	80	20	3	1	5	2	0	0	
3	2013	Honduras	United States	2	1	57	43	4	2	12	11	0	0	
4	2013	Panama	Costa Rica	2	2	47	53	5	3	13	11	1	1	

Checking for null values

- Check for null values
- If there are null values, replace with value with mean

```
football.isna().sum()
```

```
year              0
hname             0
aname             0
hgoals            0
agoals            0
hPossesion        0
aPossesion        0
hshotsOnTarget    0
ashotsOnTarget    0
hshots            0
ashots            0
hyellowCards      0
ayellowCards      0
hredCards         0
aredCards         0
hfouls            0
afouls            0
hsaves            0
asaves            0
dtype: int64
```

looking/understanding the dataset

```
football.describe()
```

	year	hgoals	agoals	hPossesion	aPossesion	hshotsOnTarget	ashotsOnTarget	hshots	ashots	hyellowCards	ayellov
count	1482.000000	1482.000000	1482.000000	1482.000000	1482.000000	1482.000000	1482.000000	1482.000000	1482.000000	1482.000000	1482.
mean	2013.495277	1.540486	1.185560	44.704453	41.122807	4.908232	3.877193	12.438596	10.016194	1.759784	1.
std	4.597541	1.474077	1.402219	21.849874	20.656505	3.320544	2.964755	6.675103	5.889003	1.333737	1.
min	2002.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
25%	2010.000000	0.000000	0.000000	37.000000	32.000000	3.000000	2.000000	8.000000	6.000000	1.000000	1.
50%	2013.000000	1.000000	1.000000	50.000000	45.500000	4.000000	3.000000	12.000000	9.000000	2.000000	2.
75%	2017.000000	2.000000	2.000000	59.000000	55.000000	7.000000	5.000000	16.000000	13.000000	3.000000	3.
max	2021.000000	10.000000	13.000000	86.000000	85.000000	26.000000	19.000000	40.000000	44.000000	8.000000	7.

Finding Correlation in Data

- Used the function corr()
- Correlation coefficient ranges from -1 to 1
- Answers the question: What columns of data should we use for the multivariable regression?

```
corr=football.corr()
corr
```

	year	hgoals	agoals	hPossesion	aPossesion	hshotsOnTarget	ashotsOnTarget	hshots	ashots	hyellowCards	ayellowCard
year	1.000000	0.037590	0.087959	0.295950	0.288497	-0.115246	-0.103228	0.046162	0.010751	-0.069228	-0.10325
hgoals	0.037590	1.000000	-0.244882	0.179468	-0.222248	0.526617	-0.270477	0.382480	-0.291527	-0.205237	0.06327
agoals	0.087959	-0.244882	1.000000	-0.155056	0.236338	-0.245915	0.539361	-0.260827	0.413058	0.144078	-0.16969
hPossesion	0.295950	0.179468	-0.155056	1.000000	0.346589	0.341612	-0.122774	0.524708	-0.061668	-0.149127	-0.00331
aPossesion	0.288497	-0.222248	0.236338	0.346589	1.000000	-0.162038	0.369227	-0.107449	0.552782	0.107855	-0.15731
hshotsOnTarget	-0.115246	0.526617	-0.245915	0.341612	-0.162038	1.000000	-0.129062	0.777171	-0.187110	-0.159579	0.08819
ashotsOnTarget	-0.103228	-0.270477	0.539361	-0.122774	0.369227	-0.129062	1.000000	-0.206086	0.782249	0.162953	-0.13400
hshots	0.046162	0.382480	-0.260827	0.524708	-0.107449	0.777171	-0.206086	1.000000	-0.205359	-0.158122	0.08571
ashots	0.010751	-0.291527	0.413058	-0.061668	0.552782	-0.187110	0.782249	-0.205359	1.000000	0.194180	-0.13959
hyellowCards	-0.069228	-0.205237	0.144078	-0.149127	0.107855	-0.159579	0.162953	-0.158122	0.194180	1.000000	0.16771
ayellowCards	-0.103255	0.063270	-0.169695	-0.003316	-0.157316	0.088191	-0.134005	0.085712	-0.139596	0.167710	1.00000
hredCards	-0.035472	-0.130300	0.119584	-0.086105	0.064699	-0.096693	0.144645	-0.102778	0.148298	0.127756	0.04356
aredCards	-0.113087	0.075325	-0.100743	-0.045027	-0.095945	0.114794	-0.049290	0.093167	-0.061868	0.156990	0.11591
hfouls	-0.110428	-0.137257	0.013217	0.091529	0.263279	0.024614	0.198101	0.038549	0.267881	0.364339	0.10813
afouls	-0.137083	-0.022653	-0.109305	0.209454	0.110077	0.160964	0.070760	0.189131	0.099847	0.134052	0.34705
hsaves	-0.229639	-0.232503	0.210595	-0.092559	0.316909	-0.127242	0.812502	-0.206361	0.649522	0.165746	-0.07799
asaves	-0.242701	0.212607	-0.234592	0.312947	-0.130955	0.810094	-0.138301	0.663652	-0.183229	-0.093001	0.09660

```
corr=football.corr()
corr
```

esion	hshotsOnTarget	shotsOnTarget	hshots	ashots	hyellowCards	ayellowCards	hredCards	aredCards	hfouls	afouls	hsaves	asaves
88497	-0.115246	-0.103228	0.046162	0.010751	-0.069228	-0.103255	-0.035472	-0.113087	-0.110428	-0.137083	-0.229639	-0.242701
22248	0.526617	-0.270477	0.382480	-0.291527	-0.205237	0.063270	-0.130300	0.075325	-0.137257	-0.022653	-0.232503	0.212607
36338	-0.245915	0.539361	-0.260827	0.413058	0.144078	-0.169695	0.119584	-0.100743	0.013217	-0.109305	0.210595	-0.234592
46589	0.341612	-0.122774	0.524708	-0.061668	-0.149127	-0.003316	-0.086105	-0.045027	0.091529	0.209454	-0.092559	0.312947
00000	-0.162038	0.369227	-0.107449	0.552782	0.107855	-0.157316	0.064699	-0.095945	0.263279	0.110077	0.316909	-0.130955
62038	1.000000	-0.129062	0.777171	-0.187110	-0.159579	0.088191	-0.096693	0.114794	0.024614	0.160964	-0.127242	0.810094
69227	-0.129062	1.000000	-0.206086	0.782249	0.162953	-0.134005	0.144645	-0.049290	0.198101	0.070760	0.812502	-0.138301
07449	0.777171	-0.206086	1.000000	-0.205359	-0.158122	0.085712	-0.102778	0.093167	0.038549	0.189131	-0.206361	0.663652
52782	-0.187110	0.782249	-0.205359	1.000000	0.194180	-0.139596	0.148298	-0.061868	0.267881	0.099847	0.649522	-0.183229
07855	-0.159579	0.162953	-0.158122	0.194180	1.000000	0.167710	0.127756	0.156990	0.364339	0.134052	0.165746	-0.093001
57316	0.088191	-0.134005	0.085712	-0.139596	0.167710	1.000000	0.043560	0.115910	0.108134	0.347059	-0.077995	0.096608
64699	-0.096693	0.144645	-0.102778	0.148298	0.127756	0.043560	1.000000	0.157497	0.100574	0.055845	0.092903	-0.084449
95945	0.114794	-0.049290	0.093167	-0.061868	0.156990	0.115910	0.157497	1.000000	0.077490	0.126437	-0.008682	0.109837
63279	0.024614	0.198101	0.038549	0.267881	0.364339	0.108134	0.100574	0.077490	1.000000	0.443330	0.171628	0.008679
10077	0.160964	0.070760	0.189131	0.099847	0.134052	0.347059	0.055845	0.126437	0.443330	1.000000	0.071003	0.150262
16909	-0.127242	0.812502	-0.206361	0.649522	0.165746	-0.077995	0.092903	-0.008682	0.171628	0.071003	1.000000	-0.023577
30955	0.810094	-0.138301	0.663652	-0.183229	-0.093001	0.096608	-0.084449	0.109837	0.008679	0.150262	-0.023577	1.000000

Creating the model

```
X=pd.DataFrame(football,columns = ['ashotsOnTarget','hsaves','ashots','aPossesion','asaves'])
y=pd.DataFrame(football, columns = ['agoals'])

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=3)
```

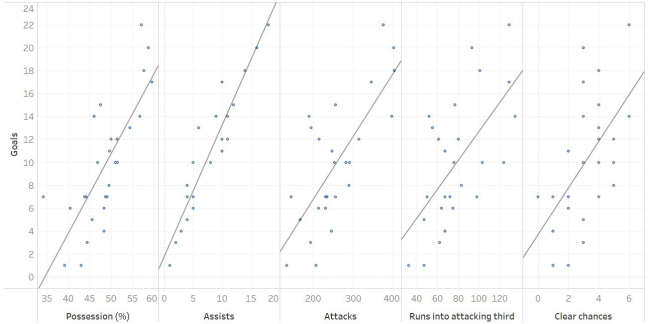
```
from sklearn.linear_model import LinearRegression
# Create linear regression model
lin_reg_mod = LinearRegression()
# Fit linear regression
lin_reg_mod.fit(X_train, y_train)
# Make prediction on the testing data
pred = lin_reg_mod.predict(X_test)
print(lin_reg_mod.intercept_)
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
print(lin_reg_mod.coef_)
test_set_r2 = r2_score(y_test, pred)
print(test_set_r2)
```

```
[0.50362727]
[[ 0.51775325 -0.40672024 -0.01709499  0.00400776 -0.04894926]]
0.408068290578366
```

Reflection/Conclusion

- Model only had a 41% R-square. Why?
  - Possible ways to improve for the future:
    - Improve the dataset
    - Use Logistic Regression

Identified important stats for determining goals



References/Citations

Dataset download links: https://www.kaggle.com/datasets/kaito510/fifa-world-cup-match-stats
https://www.kaggle.com/datasets/azminetoushikwasi/ucl-202122-uefa-champions-league

https://medium.com/swlh/predicting-nfl-scores-in-python-3560ccd58cb1

Python and tablaeu