UCD Intro to Data Analytics & Intro to Python

Project report

Prepared by: Conor Cahalane

Tutor: Shubham Jain

2022

Contents

[GitHub URL 2](#_Toc109234930)

[Abstract 2](#_Toc109234931)

[Introduction 2](#_Toc109234932)

[Dataset 2](#_Toc109234933)

[Justification: 2](#_Toc109234934)

[Source: 2](#_Toc109234935)

[Implementation Process 3](#_Toc109234936)

[Results 7](#_Toc109234937)

[Insights 9](#_Toc109234938)

[References 12](#_Toc109234939)

# GitHub URL

https://github.com/conortcahalane/UCDPythonProject

# Abstract

For this project, I chose to do Python analysis on Premier League statistics spanning the last 22 years.

The purpose of the application was to develop a Python project to analyse a real-world scenario and generate valuable insights by visualizing information. The project aims to analyse data from different data sources, manipulate information and visualise to generate insights.

Using Jupyter notebook as my platform and a host of useful resources such as packages NumPy and Pandas, an API from AlphaVantage and Visualization tools from Plotly, I was able to develop and implement the functionality required for the project.

# Introduction

As described in paper information provided by University College Dublin, the goal of the assignment is to demonstrate how you are thinking about putting course concepts and learning into practice to demonstrate the course learning outcomes.

Prior to this course, I had no previous experience with the programming language Python or Data Analysis tools such as data visualization. Due to this I knew a substantial portion of time during this project would be dedicated to investigating, understanding, and testing smaller elements of functionality, as well as looking at different implementations to try better understanding how the code was operating and what I thought would be the best solution to each bullet point set out by UCD.

Due to this for my use case I chose Premier league table statistics from 2000-2022. I chose this dataset as I knew there would be so much new concepts and functionality to deal with that my dataset may as well be from a topic in which I am familiar with. Using this dataset allowed me to focus on the coding and functionality aspects of the project more and did not impair me in having to wrap my head around aspects of the dataset.

# Dataset

## Justification:

I used Kaggle to find a dataset which appealed to me. It is always beneficial to work on topics you are interested in. Due to understanding the topic in detail, more insightful analysis can be conducted on it, thus leading to more applications for the data. This results in a greater number of outcomes that can be gained from the analysis. If I was to choose Cricket instead of Football, then many interesting data points might have gone over my head as I would not have had the background knowledge as to why it was interesting.

The EPL Standings 200-2022 is comprised of 12 columns with the headers:

Season, Pos, Team, Pld, W, L, D, GF, GA, GD, Pts, Qualification or relegation

The dataset is comprised of 440 rows comprising all the competing teams and their statistics for each season of the Premier League.

## Source:

https://www.kaggle.com/datasets/quadeer15sh/premier-league-standings-11-seasons-20102021?resource=download

# Implementation Process

## 2. Importing

Firstly, to get our dataset into a Python DataFrame we use Pandas. We import this into our Python code and use it’s .read\_csv() function to create a DataFrame out of our csv file downloaded from Kaggle. This can be seen below:

PL = pd.read\_csv("EPL Standings 2000-2022.csv",index\_col = 0)

PL

Another example of importing in the project is the use of the TIME\_SERIES\_INTRADAY API courtesy of AlphaVantage. Given the Premier League theme of the project I thought it would be interesting to use this API which retrieves stock data from an online API to pull information on one of the only football clubs in the world that has a ticker symbol on the New York Stock Exchange, Manchester United (Ticker Symbol: MANU) which also happens to be a Premier league club.

This was achieved by using my unique API Key given to me by AlphaVantage and inputting it into the below code. The url was also edited to only retrieve MANU data.

import requests

url = 'https://www.alphavantage.co/query?function=TIME\_SERIES\_INTRADAY&symbol=MANU&interval=5min&apikey=XGA4O6I3IUUXDA1P'

r = requests.get(url)

data = r.json()

print(data)

## 3. Preparation

### Sorting

Utilized Pythons .sort\_values() function in order to group together all the individual teams row data to be in sequence, having that data then be sorted by descending years and finally showing which position they came in said year. A code snippet of this can be seen below:

PL\_Seasons.sort\_values(

by=["Team", "Season","Pos"],

ascending=False

)[["Team", "Season","Pos"]]

Output:

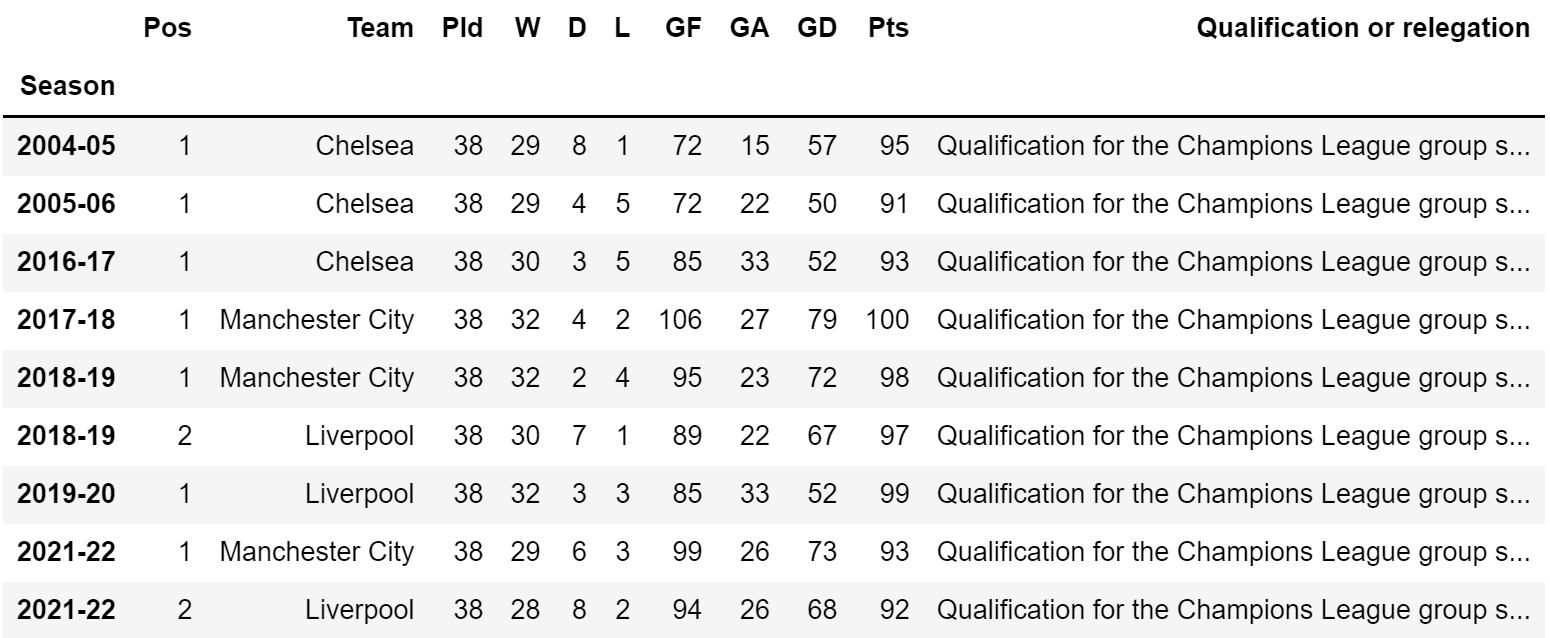


### Indexing

A quick example of some Boolean Indexing involved in the project was my wish to have a view of all the teams that scored over 90 points in a single season. To achieve this I simply indexed the PL\_Seasons DataFrame on the column ‘Pts’ and added a condition to only return the rows that had scored over 90. Below you can find this snippet and output:

PL\_Seasons[PL\_Seasons['Pts']>90]

Output:



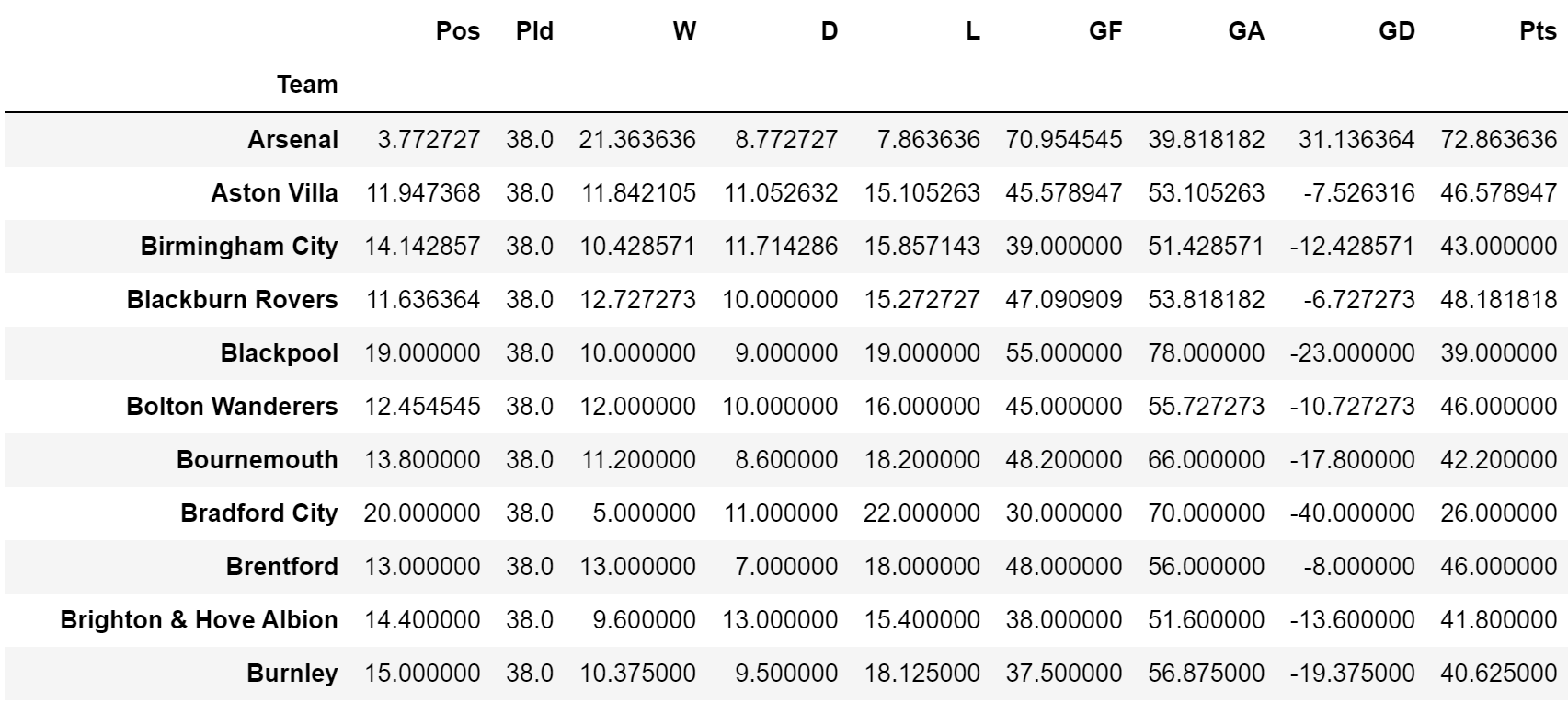
### Grouping

Grouping is an excellent way of providing a lot of data quickly and efficiently. Here I used the .groupby() function in conjunction with the .mean() function on all the teams in the Premier League dataframe to give a quick overview of their average league position, average points, average wins and many more.

Teams\_Grouped = PL\_Seasons.groupby(["Team"]).mean()

Teams\_Grouped

Output (Not the full output):



### Drop duplicates, replace missing values

Thankfully my dataset imported with no duplicates or missing values, but these would’ve been remedied using the below coding functions.

This would have dropped any rows that came back as duplicates:

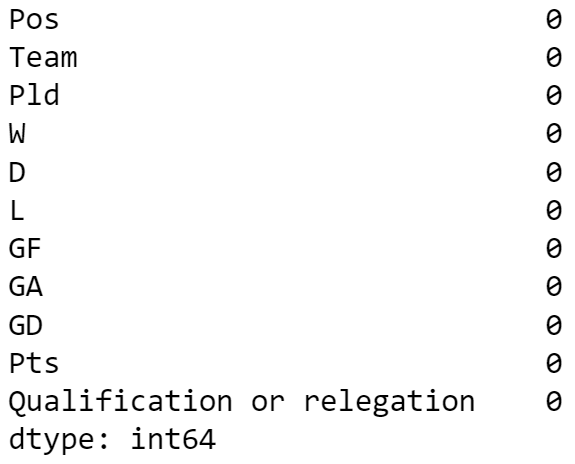
drop\_duplicates= PL.drop\_duplicates()

This is a check to see if there are any data gaps in the code, checking this is important in data analysis to ensure you aren’t surprised in the future if you code breaks due to significant data gaps:

missing\_values\_count = PL.isnull().sum()

print(missing\_values\_count[0:11])

Output:



### Merge DataFrames & Lists

For this ask I manually created a DataFrame which contained the Manchester United starting eleven squad and their ages. I initially created lists and added them to an array. I then called the DataFrame function in order to convert the lists. This was needed in order to merge this data to my main PL\_Seasons DataFrame. This can be seen below:

MU\_Player\_data = {'Name': ['David De Gea', 'Harry Maguire', 'Victor Lindelof', 'Luke Shaw',

'Diogo Dalot', 'Fred', 'Scott McTominay', 'Bruno Fernandes',

'Cristiano Ronaldo', 'Marcus Rashford', 'Jadon Sancho'],

'Age': [31, 29, 28, 27,

23, 29, 25, 27,

37, 24, 22],

'Team': ['Manchester United','Manchester United','Manchester United','Manchester United',

'Manchester United','Manchester United','Manchester United','Manchester United',

'Manchester United','Manchester United','Manchester United']}

# Create DataFrame

MU\_Player = pd.DataFrame(MU\_Player\_data)

Following, In the below code we can see an outer merge being used with the how parameter, which results in a full outer join. In an outer join all rows from both DataFrames will be present in the new DataFrame. This results in showing the full team for each season Manchester United played and all the other teams, with their player information showing NaN (Not a Number). No rows are lost in an outer join, even when they do not have a match in the other DataFrame.

MU\_outer\_merged = pd.merge(

PL\_Seasons, MU\_Player, how="outer", on=["Team"]

)

MU\_outer\_merged

Output:



## 4.Analysis

### Conditional statements & Looping & NumPy & Classes

Below we introduce a for loop which will help minimize the amount of code needed in the analysis. This for loop will loop continuously through the number of values in the 'Wins' column until it has passed through them all. For each row of the DataFrame the for loop will check using the condition statement if and elif whether the teams amount of wins that year were above the 22-year PL average. This PL win average was defined earlier in the project as an integer and is now being called back here as it is relevant. If the team had above average wins that year then the loop will return a sentence stating the year, team name and that they have won more than the PL average, else it will return that they won less games than the average. The colour class is also used in the return to emphasise the result and help the readability of the sentence. The NumPy function .any() is used here to test whether any array element along a given axis evaluates to True. Without this function the loop would return an error saying a series cannot be used.

for i in range(len(PL\_Seasons['W'])):

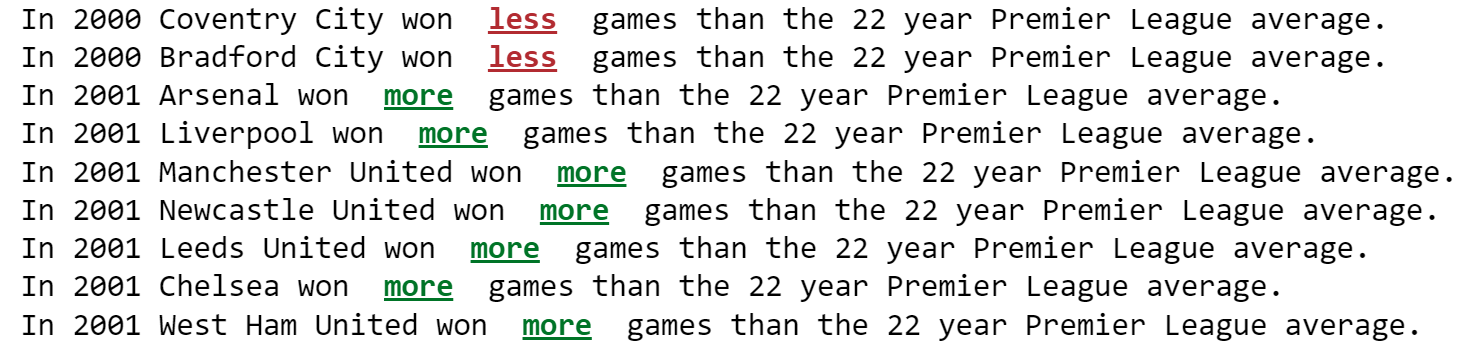
if (PL\_Seasons['W'][i] > PL\_Average\_W).any():

print("In",PL\_Seasons['Season'][i],PL\_Seasons['Team'][i],"won ",colour.GREEN + color.BOLD + colour.UNDERLINE +"more"+ colour.END," games than the 22 year Premier League average.")

elif (PL\_Seasons['W'][i] < PL\_Average\_W).any():

print("In",PL\_Seasons['Season'][i],PL\_Seasons['Team'][i],"won ",colour.RED + color.BOLD+ colour.UNDERLINE +"less"+ colour.END," games than the 22 year Premier League average.")

Output (Snippet):



### Define a custom function to create reusable code

Below can be found a user defined function that checks to see if the team typed into the function played in the premier league. This function uses the conditional statement if to check the DataFrame for the function’s argument.

User defined function:

def PremierLeague\_Team\_Check(i):

if (i == PL\_Seasons['Team']).any():

print("This team", colour.GREEN + color.BOLD + "did" + colour.END, "play in the premier league.")

else:

print("This team", colour.RED + color.BOLD + "didn't" + colour.END, "play in the premier league.")

Calling the User defined function:

#An example of a team that is in the premier league

PremierLeague\_Team\_Check('Chelsea')

Result:



Calling the User defined function:

#An example of a team that is in the premier league

PremierLeague\_Team\_Check(‘Barcelona’)

Result:



# Results

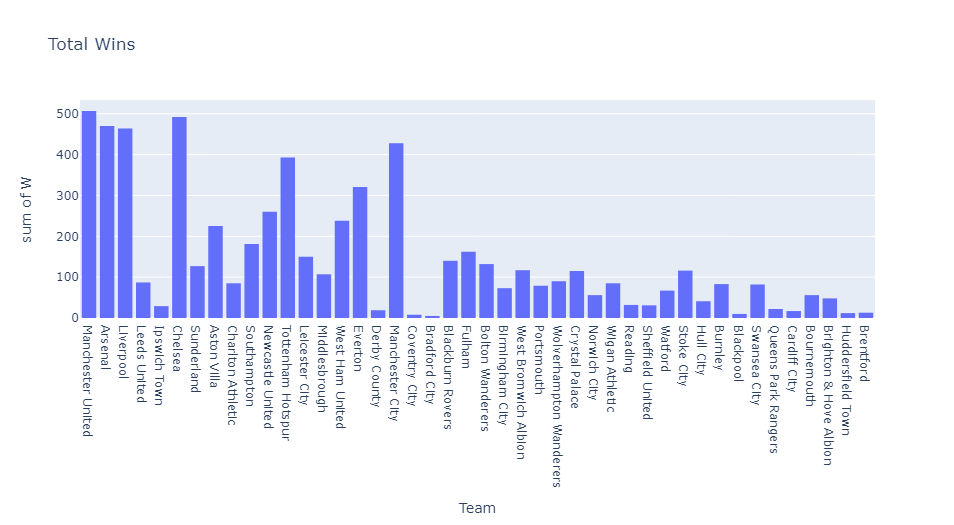
## 5. Visualization

Below we imported Plotly after installing it into our Jupyter through pip in the terminal. Using Plotly we can visualise the DataFrame in a multitude of diverse ways. Firstly, below is a histogram chart of the total wins by each team that participated in the premier league. By simply hovering over each bin the user can get a view of how many wins each team exactly had.

import plotly.express as px

df = PL

fig = px.histogram(df, x="Team",y="W", title="Total Wins")

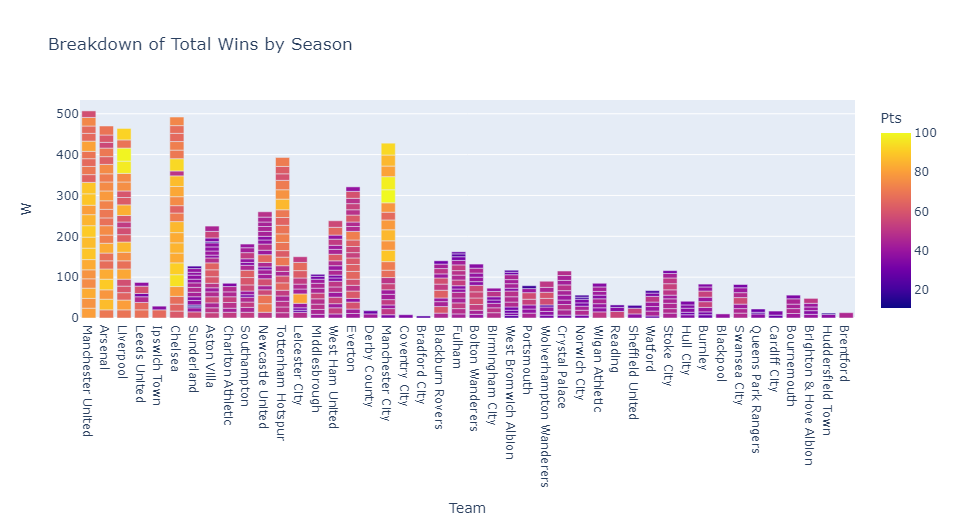
fig.show()

Secondly, below is a bar chart of the total wins by each team that’s participated in the premier league. By simply hovering over each bin the user can get a view of how many wins each team exactly per season, starting with the earliest season they participated in at the bottom. Using the color attribute on points, we can also visually see how each teams points tally was in each season. The legend to the right of the graph shows that the higher the points tally the brighter the season will display. By defining Hover\_data we can expand on the information providing giving users the full W/L/D/Points data for each season.

fig = px.bar(PL\_Seasons, x='Team', y='W', hover\_data=['Team','Season','Pos', 'W', 'D', 'L', 'Pts'],

title="Breakdown of Total Wins by Season", color = "Pts")

fig.show()

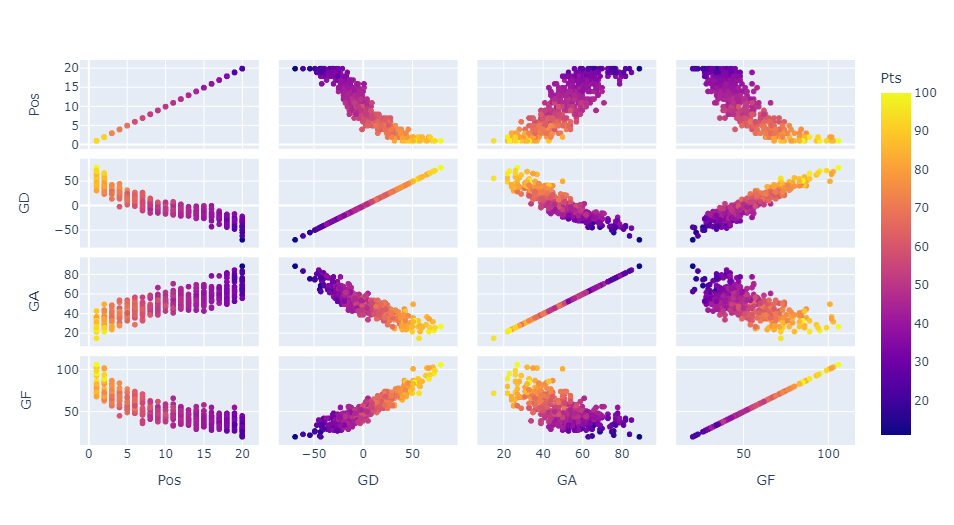


Lastly, here we see the visualization of scatter graphs being used with Goal Difference, Goals Against and Goals For. The colors correspond with the points gained with the more points resulting in brighter datapoints. By defining Hover\_data we can expand on the information providing giving users the full Pos/GD/GA/GF data for each season.

fig = px.scatter\_matrix(PL\_Seasons,dimensions=["Pos", "GD", "GA", "GF"],

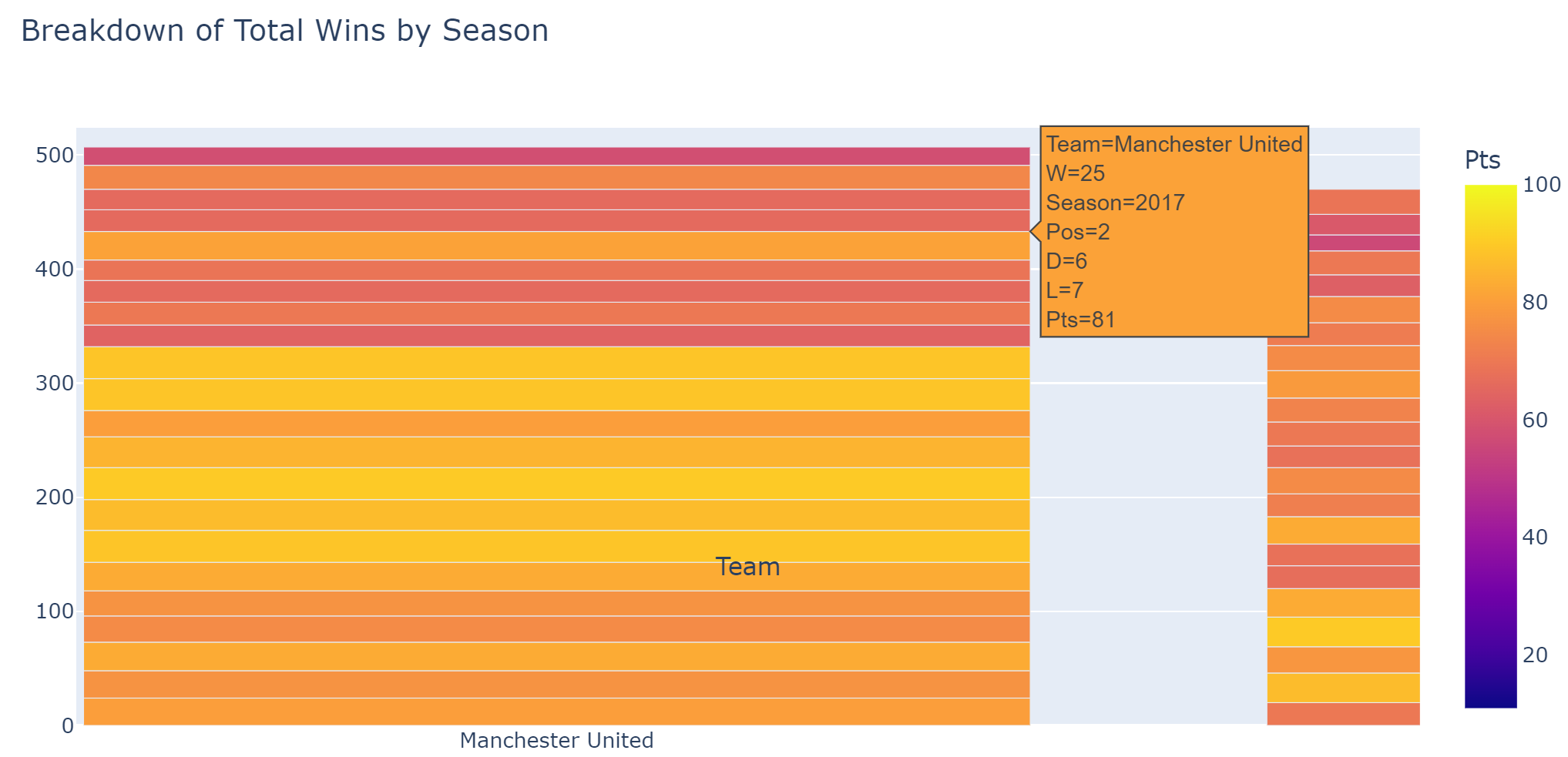
hover\_data=['Team','Season', 'GA', 'GF', 'Pts','Pos'],color="Pts")

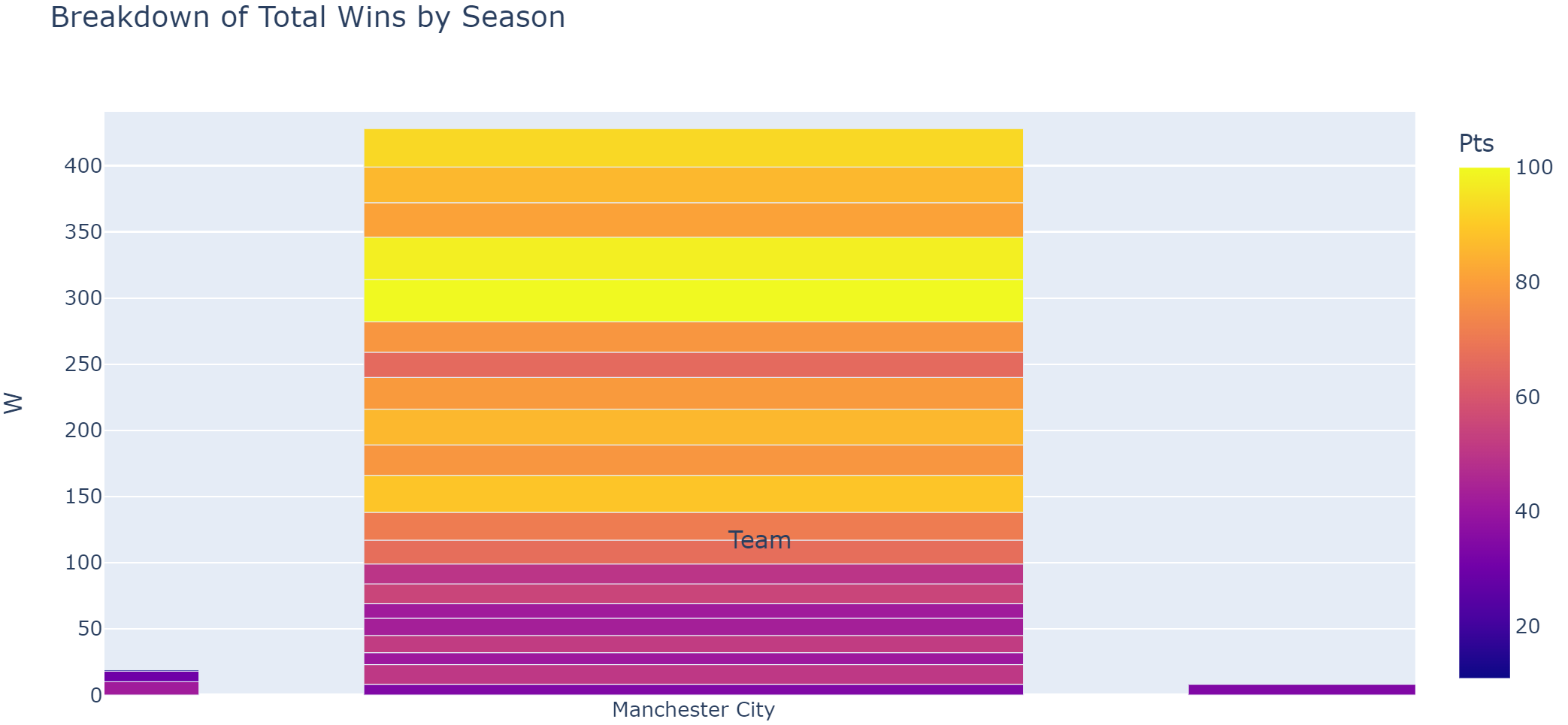
fig.show()



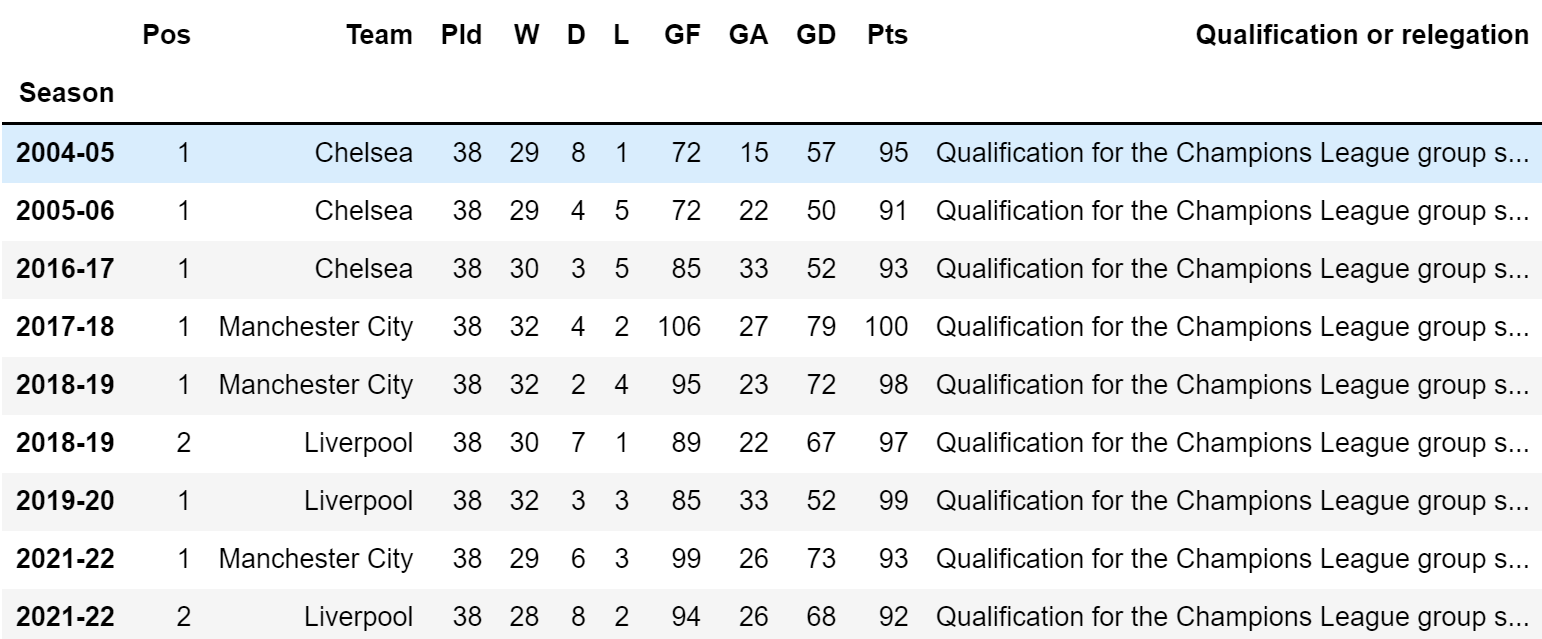
# Insights

One of the most striking insights that can be found from the bar chart visualization is the pure hand-over in dominance from Manchester United to Manchester City in the last decade. This can be strikingly seen in the bar chart colour tones. Man United going from bright coloured data points (Indicating very high league positions) to darker tones (indicating lower league positions). This was the opposite case for Man City as can be seen vividly in the two screenshots below:

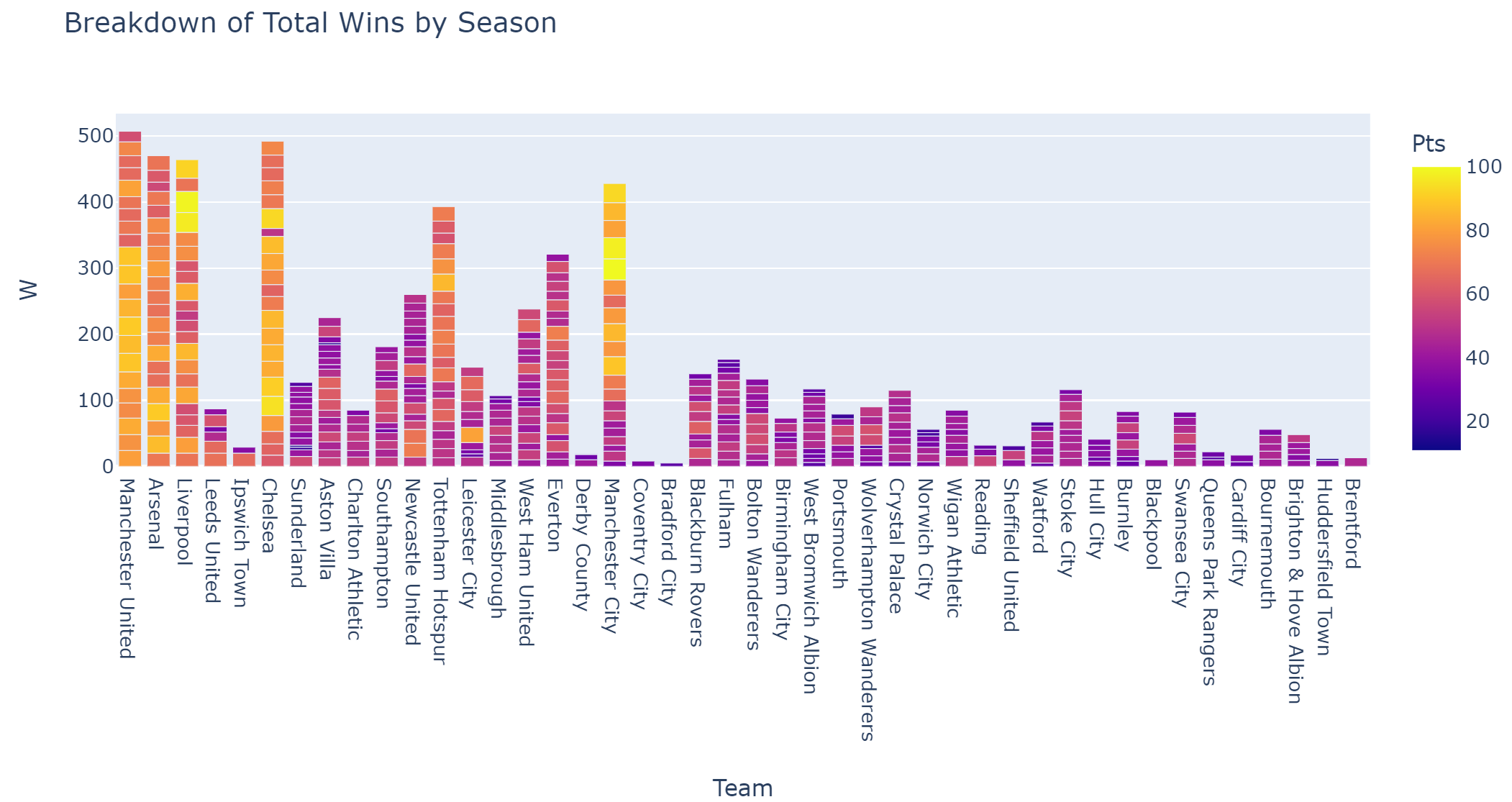




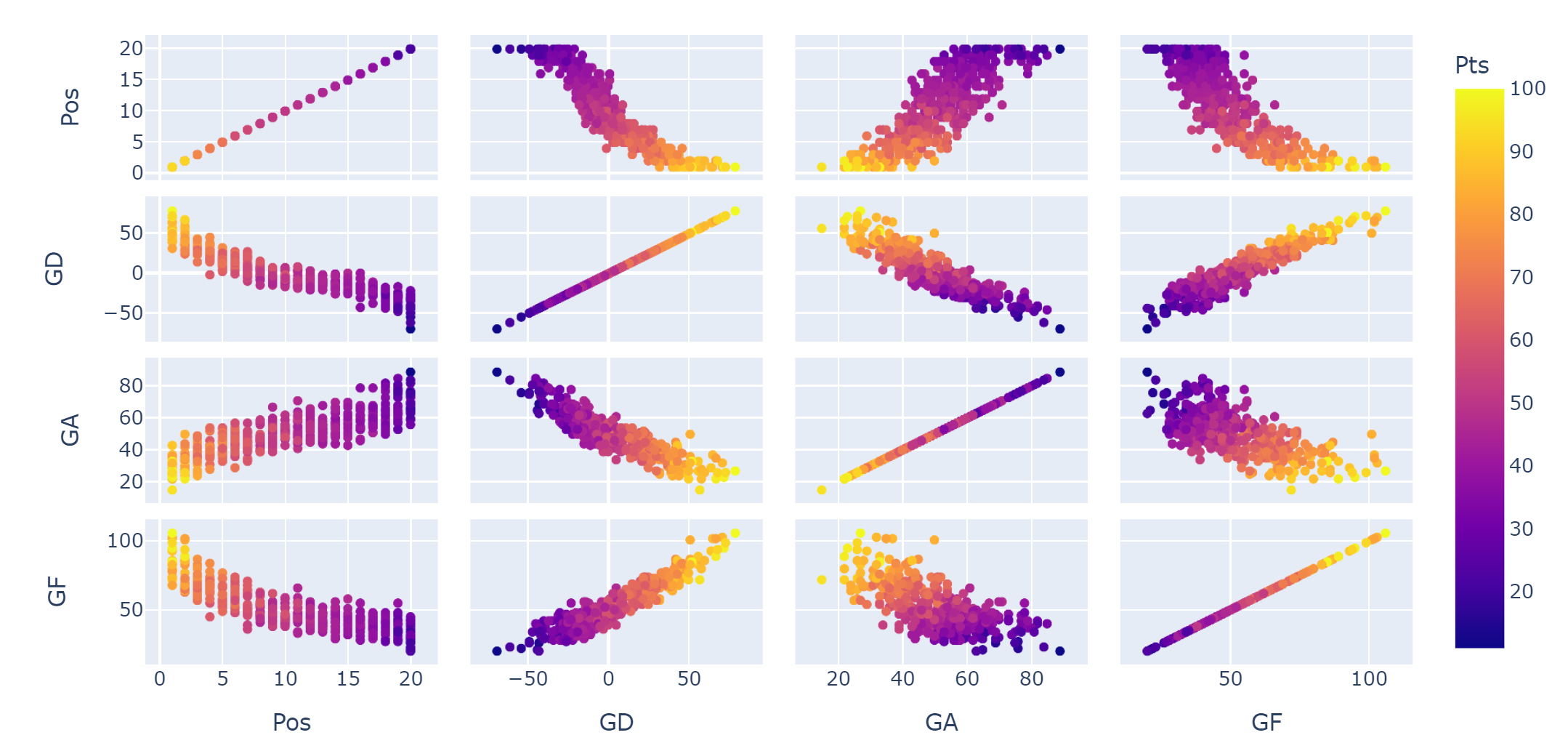
Following, another interesting insight is that in only 7 of the 22 Premier League seasons analyzed ninety points or more were required to win the league outright, with 2 of these seasons having 2 contenders scoring over 90 points.



Furthermore, through our bar chart we can instantly see that of all the teams involved in the Premier League throughout the last 22 year that only 3 of them have been promoted to only instantly be relegated and never to reappear, these teams being Coventry City, Bradford City and Blackpool.



Moving on, we can see using our scatter graph that Goals For, Goals Against and Goal Differences are all heavily linked to league Positions. There being very few outliers to this trend, one of the being Liverpool’s 2013 season where they had a very high goals for of 101 and a high goals against of 51 despite still coming 2nd in the league with a goal difference of 51. These stats made them stand out against the rest for still placing high in the leaderboard but conceding an awful amount of goals.



The final insight being a group view of how all the teams ranked when it came to two key attributes, Position and Points throughout the 22 years. This showed that even with the recent failures this decade Manchester United still retain the best Premier League position average and points average.



# Machine learning

Using machine learning in the future any object, picture, video or soundbite will be able to solely be reproduced by machine learning. Once thought only possible in a sci-fi novel is now ever increasingly being proven to be true.

Evidence of this prediction is the recent release of Dall-e 2. DALL·E 2 is a new AI system that can create realistic images and art from a description in natural language. Its results are crisp, accurate and return in seconds. With this open-source machine learning AI users are prompted to type anything with the aim that the AI will output an image of exactly what you asked for. Horse on the moon, Dalle-2 in its current state will output exactly that in under a minute. This AI not only returns what you ask for but completely pixel by pixel creates it for you ensuring that this image has never been seen or presented before. This in the future will be applied to video and paired with audio to have AI create whole videos are completely of the AI’s making.

# Regression Vs Classification

The biggest difference in the argument of regression versus classification is that regression can aid in the prediction of a continuous quantity, classification labels the classes it predicts and gives a result based on those labels.

In a school setting for example using classification it would return a pass or fail based on the data it was given, yet with regression it would return an accurate percentage. There are appropriate use cases for both models given the circumstances they are being asked of.

# References

<https://github.com/conortcahalane/UCDPythonProject>

https://www.kaggle.com/datasets/quadeer15sh/premier-league-standings-11-seasons-20102021?resource=download

<https://jupyter.org/>

<https://pandas.pydata.org/>

<https://numpy.org/>

https://plotly.com/