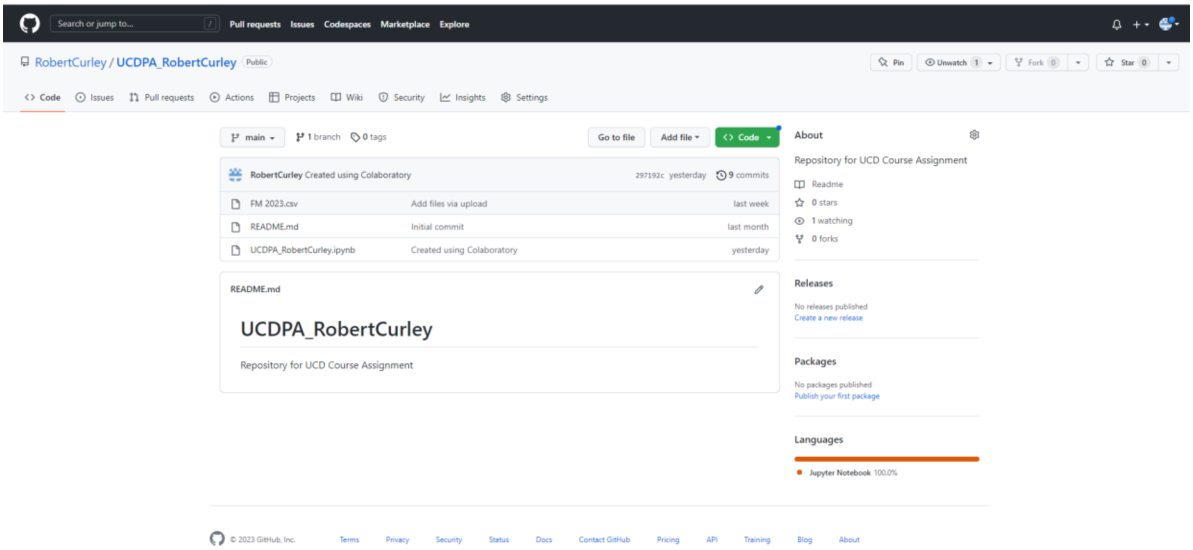
Contents

Project Report

GitHub URL

https://github.com/RobertCurley/UCDPA_RobertCurley



Abstract

The following project will show Python data analytics in action through Google Colab. Touching on the key actions of a data scientist, such as;

- Data extraction
- Data cleaning
- Data manipulation
- Data visualization

The project will use the Football Manager 2023 Dataset. Which can be viewed on Kaggle (https://www.kaggle.com/datasets/platinum22/foot-ball-manager-2023-dataset). This dataset was used due to my passion for gaming and soccer. This game uses real-world data which gives stats to players across the globe, in-fact in 2014 there was an article in the Guardian titled “Why clubs are using Football Manager as a real-life scouting tool” (Stuart, 2014).

I wanted to use the knowledge I have learned to analysis this dataset myself to see if there is any correlation between different statistics and to find who the is the player with the highest potential ability.

Introduction

In 2014, the guardian wrote an article titled "Why clubs are using Football Manager as a real-life scouting tool" (Stuart, 2014). This led me to think about the data that is being presented in Football Manager and if I could use the data to gather insights into players. Football Manager 2023 is a computer game which allows you to manage a real-world team.

With Soccer being more and more statistical in recent years with the introduction of expected goals, I wanted to use the Football Manager dataset to delve deeper into the specifics of the data. I have been playing Football Manager for the last few years and have been drawn to the real-world data that is present in the game which sets the game apart from the competition.

As I have an interest in the sport of Football and gaming, I have decided to look at the "Football Manager 2023 Dataset" to gain insights around the players and positions that are in Football Manager 2023.

Dataset

The Dataset that I used is “Football Manager 2023 Dataset” which was taken from Kaggle. I had a few different options to choose from when looking at football player data but decided to stick with the “Football Manager 2023 Dataset”, due to the number of data points that are present with it covering 8,452 players.

Although the Usability score on Kaggle gave only a 6.18, I felt it the dataset had enough information present and usable that I could work with the data.

The dataset consists of 8,452 players covering a number of key statistics such as Finishing, Heading, Salary, Height etc. Which when I viewed the CSV file felt I could analysis and use to come up with specific findings.

Implementation Process

(Describe your entire process in detail)

Once I had decided on using the "Football Manager 2023 Dataset", I had to download the data.

The system which I decided to use to conduct my analysis was Google Colab. This was due to the simply set up involved in starting a new notebook and the user-friendly interface. The direct link to GitHub I also saw as a big benefit to using Google Colab.

(Data Extraction)

Download the Data

I downloaded the dataset in CSV format onto my local C: Drive, from here I viewed the dataset in CSV format to ensure that I had the columns and initial data that I wanted.

From this initial view of the CSV dataset, I concluded that the dataset could be used in my project and had the enough columns and rows to conduct an analysis.

Python Upload

The very first thing which I did in the notebook was to import NumPy and Pandas package. I done this as it allowed me to recall the packages properties going forward in the project without having to constantly import the packages.

```
[90] # Import NumPy Package
     import numpy as np

     # Import Pandas Package
     import pandas as pd
```

Initially, I struggled to find an easy way to upload the dataset to Google Colab that allowed the code to run seamlessly what ever device I was using. At first I used Google Colab's built it upload function, to upload from my local drive.

```
#Initial Upload of CSV file of Football Manager 2023 from local drive

from google.colab import files
uploaded = files.upload()

Choose files FM 2023.csv
• FM 2023.csv(text/csv) - 2879701 bytes, last modified: 02/03/2023 - 100% done
Saving FM 2023.csv to FM 2023.csv
```

I found this was very time consuming as I had to select the data from my local drive every time I was changing or adding to the code.

After a bit of research, I found that you could upload the CSV file directly to your GitHub repository. This allowed Google Colab to read a URL which contained the raw CSV file, it also meant that when I ran all cells I didn't have to select the dataset constantly. To do this I named the GitHub URL which contained the raw dataset to *'url'*. I then used the *'.read_csv'* function from pandas to call the URL and called it fm_2023_df.

```
# Upload CSV file of Football Manager 2023 from github repository

url = 'https://raw.githubusercontent.com/RobertCurley/UCDPA_RobertCurley/main/FM%202023.csv'
```

```
# View FM 2023.csv file in python

fm_2023_df = pd.read_csv(url) # Changed Name to FM_2023_df
fm_2023_df
```

	Name	Position	Age	ca	pa	Nationality	Club	Corners	Crossing	Dribbling	...	World reputation	Race	RCA
0	Kevin De Bruyne	M/AM RLC	31	189	189	Belgium	Manchester City	14	19	15	...	9400	Northern_European	181
1	Kylian Mbappé	AM/S RL	23	188	197	France	Paris Saint-Germain	13	13	18	...	9248	African_Caribbean	172
2	Robert Lewandowski	S	33	186	190	Poland	Barcelona	3	8	13	...	9250	Northern_European	183

Understanding the data present

When I had the dataset uploaded onto Google Colab I wanted to understand the data. To do this I used functions and properties that are present in python.

```
#Shows the type of data that fm_2023_df is
type(fm_2023_df)
```

```
pandas.core.frame.DataFrame
```

```
# Prints out full DataFrame
print(fm_2023_df)
```

```
                    Name  Position  Age   ca   pa     Nationality  \
0        Kevin De Bruyne  M/AM RLC   31  189  189         Belgium
1          Kylian Mbappé   AM/S RL   23  188  197          France
2     Robert Lewandowski         S   33  186  190          Poland
3         Erling Haaland         S   22  185  195  Norway,England
4          Mohamed Salah   AM/S RL   30  185  187           Egypt
...                  ...       ...  ...  ...  ...             ...
8447          Joe Ashton       D L   16   45  135         England
8448          River Ries         S   17   45  135         Germany
8449       Halilcan Doğan       D C   23   45  -75          Turkey
8450        Adijat Sefer         S   17   45  135         Germany
8451         Linus Urban         S   17   45  135         Germany

                     Club  Corners  Crossing  Dribbling  ...  \
0         Manchester City       14        19         15  ...
1     Paris Saint-Germain       13        13         18  ...
2               Barcelona        3         8         13  ...
3         Manchester City        7        10         14  ...
4               Liverpool       12        14         17  ...
...                   ...      ...       ...        ...  ...
8447              Burnley        3         4          5
```

*More Data is present

```
[96] # Returns the shape of the dataframe
     fm_2023_df.shape
```

```
(8452, 98)
```

```
# Returns the information present in the dataframe
fm_2023_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8452 entries, 0 to 8451
Data columns (total 98 columns):
 #    Column                              Non-Null Count  Dtype
---   ------                              --------------  -----
 0    Name                                8452 non-null   object
 1    Position                            8452 non-null   object
 2    Age                                 8452 non-null   int64
 3    ca                                  8452 non-null   int64
 4    pa                                  8452 non-null   int64
 5    Nationality                         8452 non-null   object
 6    Club                                8345 non-null   object
 7    Corners                             8452 non-null   int64
 8    Crossing                            8452 non-null   int64
 9    Dribbling                           8452 non-null   int64
 10   Finishing                           8452 non-null   int64
 11   First Touch                         8452 non-null   int64
 12   Free Kick Taking                    8452 non-null   int64
 13   Heading                             8452 non-null   int64
 14   Long Shots                          8452 non-null   int64
 15   Long Throws                         8452 non-null   int64
```

*More Data is present on notebook

```
[125] #Returns the number of Null values in each column in the DataFrame
fm_2023_df.isnull().sum()
```

```
Name                                 0
Position                             0
Age                                  0
ca                                   0
pa                                   0
                                  ...
Number of national team appearances  0
Goals scored for the national team   0
Salary                             107
Rental club                       7457
UID                                  0
Length: 98, dtype: int64
```

```
# Returns List of Title of all Columns in DataFrame
fm_2023_df.columns
```

```
Index(['Name', 'Position', 'Age', 'ca', 'pa', 'Nationality', 'Club', 'Corners',
       'Crossing', 'Dribbling', 'Finishing', 'First Touch', 'Free Kick Taking',
       'Heading', 'Long Shots', 'Long Throws', 'Marking', 'Passing',
       'Penalty Taking', 'Tackling', 'Technique', 'Aggressiion',
       'Anticipation', 'Bravery', 'Composure', 'Concentration', 'Vision',
       'Decision', 'Determination', 'Flair', 'Leadership', 'Off The Ball',
       'Position.1', 'Teamwork', 'Work Rate', 'Acceleration', 'Agility',
       'Balance', 'Jumping Reach', 'Natural Fitness', 'Pace', 'Stamina',
       'Strength', 'Stability', 'Foul', 'Contest performance', 'Injury',
       'diversity', 'Aerial Reach', 'Command Of Area', 'Communication',
       'Eccentricity', 'Handling', 'Kicking', 'One On Ones', 'Reflexes',
       'Rushing Out', 'Punching', 'Throwing', 'Adaptation', 'Ambition',
       'Argue', 'Loyal', 'Resistant to stress', 'Professional',
       'Sportsmanship', 'Emotional control', 'GK', 'DL', 'DC', 'DR', 'WBL',
       'WBR', 'DM', 'ML', 'MC', 'MR', 'AML', 'AMC', 'AMR', 'ST', 'Height',
       'Weight', 'Left Foot', 'Right Foot', 'Values', 'Current reputation',
       'Domestic reputation', 'World reputation', 'Race', 'RCA',
       'Colour of skin', 'Date of birth',
       'Number of national team appearances',
       'Goals scored for the national team', 'Salary', 'Rental club', 'UID'],
      dtype='object')
```

Cleaning the data frame (Data Cleaning)

With 98 Columns present in the data frame and Null values present in the data frame I had to clean the data so that I only had specific values which I wanted to analysis.

Creating specific data frames

I sliced the data frame *'fm_2023_df'* into two data frames. The first one looking at some of the key details for a player such as nationality, club etc. named *'KeyDetails_fm_2023_df'* and the second looking at some stats which surround scoring named *'scoring_fm_2023_df'.*

- Key Details_fm_2023_df

```
# Slice Data in DataFrame (fm_2023_df) and name KeyDetails_fm_2023_df
# Key Details : Name, Position, Nationality, Club, Age, Height, Weight, salary ca & pa

KeyDetails_fm_2023_df = fm_2023_df.loc[:,['Name', 'Position', 'Nationality', 'Club',
                                          'Age', 'Height', 'Weight', 'Salary', 'ca', 'pa']].set_index('Name')

print(KeyDetails_fm_2023_df)
```

```
                    Position     Nationality                     Club  Age  \
Name
Kevin De Bruyne     M/AM RLC         Belgium          Manchester City   31
Kylian Mbappé        AM/S RL          France      Paris Saint-Germain   23
Robert Lewandowski         S          Poland                Barcelona   33
Erling Haaland             S  Norway,England          Manchester City   22
Mohamed Salah        AM/S RL           Egypt                Liverpool   30
...                      ...             ...                      ...  ...
Joe Ashton               D L         England                  Burnley   16
River Ries                 S         Germany             Karlsruher SC   17
Halilcan Doğan           D C          Turkey  Osmaniyespor Futbol Kulübü   23
Adijat Sefer               S         Germany           TSG Hoffenheim   17
Linus Urban                S         Germany         SV Werder Bremen   17

                    Height  Weight     Salary   ca   pa
```

- scoring_fm_2023_df

```
# Slice Data in DataFrame (fm_2023_df) based on key stats for scoring and name it scoring_fm_2023_df
# Key Stats : Name, Dribbling, Finishing, First Touch, Free Kick Taking, Heading, Long Shots, Penalty Taking

scoring_fm_2023_df = fm_2023_df.loc[:, ['Name', 'Dribbling', 'Finishing', 'First Touch', 'Free Kick Taking', 'Heading',
                                        'Long Shots', 'Penalty Taking']].set_index('Name')
print(scoring_fm_2023_df)
```

```
                    Dribbling  Finishing  First Touch  Free Kick Taking  \
Name
Kevin De Bruyne            15         16           16                17
Kylian Mbappé              18         17           18                12
Robert Lewandowski         13         19           18                15
Erling Haaland             14         18           16                13
Mohamed Salah              17         17           17                12
...                       ...        ...          ...               ...
Joe Ashton                  5          4            4                 1
River Ries                 11         11           11                 2
Halilcan Doğan              1          2            4                 1
Adijat Sefer               12          7            8                 3
Linus Urban                10         13            8                 4

                    Heading  Long Shots  Penalty Taking
Name
Kevin De Bruyne           6          17              16
```

Merging data frames

When I had these data frames created, I decided to merge them together creating the *'Summary_fm_2023_df'*. As the 2 data frames had the Name coloumn in common I decided to Merge the data on this column using the .merge function.

```
# Create Summary List based of 2 created DataFrames (KeyDetails_fm_2023_df & scoring_fm_2023_df) and name Summary_fm_2023_df

Summary_fm_2023_df = KeyDetails_fm_2023_df.merge(scoring_fm_2023_df, on = 'Name')
print(Summary_fm_2023_df)
```

```
                    Position     Nationality                     Club  Age  \
Name
Kevin De Bruyne     M/AM RLC         Belgium          Manchester City   31
Kylian Mbappé        AM/S RL          France      Paris Saint-Germain   23
Robert Lewandowski         S          Poland                Barcelona   33
Erling Haaland             S  Norway,England          Manchester City   22
Mohamed Salah        AM/S RL           Egypt                Liverpool   30
...                      ...             ...                      ...  ...
Joe Ashton               D L         England                  Burnley   16
River Ries                 S         Germany             Karlsruher SC   17
Halilcan Doğan           D C          Turkey  Osmaniyespor Futbol Kulübü   23
Adijat Sefer               S         Germany           TSG Hoffenheim   17
Linus Urban                S         Germany         SV Werder Bremen   17

                    Height  Weight     Salary   ca   pa  Dribbling  Finishing  \
Name
Kevin De Bruyne        181      68   394372.0  189  189         15         16
Kylian Mbappé          178      73  1035616.0  188  197         18         17
Robert Lewandowski     185      81   345204.0  186  190         13         19
Erling Haaland         195      88   394372.0  185  195         14         18
```

Dropping Duplicates in Data frame

As there was Null values present under the Salary and Club columns I had to tidy up the data. As the data frame had 214 null values rows which contained null values out of a total 8,452 rows I decided to drop those rows altogether using .dropna() function. I used this function instead of replacing duplicates as I felt that there was more than enough data present to analysis after dropping the rows with duplicates as it left the data frame with 8345 rows. There is an example of replacing null values within salary with the mean salary by age using a dictionary and .fillna() function.

```
[36] #Drop Null values within Summary_fm_2023_df

     print("The Summary_fm_2023_df originally had a total of :", Summary_fm_2023_df.isnull().sum().sum(), " Null Values")

     Summary_fm_2023_df = Summary_fm_2023_df.dropna()

     print("After using dropna() the dataframe now has :", Summary_fm_2023_df.isnull().sum().sum()," Null Values")

     The Summary_fm_2023_df originally had a total of : 0  Null Values
     After using dropna() the dataframe now has : 0  Null Values
```

Data Manipulation

There was a number of different ways I wanted to view the data.

Sorting

The first was to sort the data to show the Top players by different aspects. This was done through utilizing the sort_values function alongside the iloc function.

The first table was to show the top 5 players who were the best at Finishing in the game alongside their Salary, ca (Current ability), pa (Potential ability) and dribbling.

```
# Sort Summary_fm_2023_df by Finishing

Summary_fm_2023_df.sort_values(by='Finishing', ascending=False).iloc[:5,6:11]
```

Name	Salary	ca	pa	Dribbling	Finishing
Cristiano Ronaldo	556760.0	158	196	11	19
Robert Lewandowski	345204.0	186	190	13	19
Harry Kane	231983.0	183	185	14	19
Ciro Immobile	142499.0	160	165	12	18
Mauro Icardi	229999.0	150	174	12	18

The second table was to show the Top 10 players ranked on pa (Potential Ability) alongside their Salary and ca (Current Ability).

```python
# Sort Summary_fm_2023_df by pa

Summary_fm_2023_df.sort_values(by="pa", ascending=False).iloc[:10, 6:9]
```

Name	Salary	ca	pa
Lionel Messi	776712.0	180	200
Kylian Mbappé	1035616.0	188	197
Cristiano Ronaldo	556760.0	158	196
Erling Haaland	394372.0	185	195
Gianluigi Buffon	30239.0	125	194
Manuel Neuer	345204.0	175	193
Robert Lewandowski	345204.0	186	190
Neymar	939648.0	177	190
Kevin De Bruyne	394372.0	189	189
Luis Suárez	8121.0	147	188

Looping and defining a function

To tidy up the position present in the database into a clearer view I wanted to create a formula which converted them in 4 key positions that are present on a football pitch (Goalkeeper, Defender, Midfielder and Forward). As there was 147 unique positions present in the dataset to begin, I felt having 4 would make the data more viewable going forward. I then printed out the list and began to build out a looping formula which will create a new column based on the position.

The looping formula will take the first letter/letters from the position column and convert into either Goalkeeper, Defender, Midfielder or Forward. To input the starting letters I printed out the Positions list, scanned over the list to ensure my theory of the first letter was correct and then manually went through the first few that I could see converting them in the formula. To ensure that I had all positions covered I used the isnull().sum() function to show how many null values were present in the new column. If this was not zero I revisited the list looking for the position that was missing and then added it to the loop.

```
#Create loop to return a new column based on the Position
def f(x):

  if x.startswith('M') :
    return 'Midfielder'
  elif x.startswith('AM') :
    return 'Midfielder'
  elif x.startswith('S') :
    return 'Forward'
  elif x.startswith('GK') :
    return 'Goalkeeper'
  elif x.startswith('D') :
    return 'Defender'
  elif x.startswith('DM') :
    return 'Midfielder'
  elif x.startswith('WB') :
    return 'Defender'

Summary_fm_2023_df['Pos_2'] = Summary_fm_2023_df.Position.str[:2].apply(f)
#Check if any Nulls present after formula
print(Summary_fm_2023_df['Pos_2'].isnull().sum())
print(Summary_fm_2023_df['Pos_2'])
```

Grouping

I wanted to view the data based on the mean data by Club. I did this to get a view of the game at a Club level, sorting the data by Salary to see what club pays the highest mean salary.

```
[24] # Group Summary_fm_2023_df by Club

Club_df = Summary_fm_2023_df.groupby(by="Club").mean()
Club_df.sort_values(by="Salary", ascending=False).iloc[:10,0:5]
```

Club	Age	Height	Weight	Salary	ca
Paris Saint-Germain	24.439024	180.463415	56.609756	154867.682927	135.000000
FC Bayern München	22.045455	183.977273	71.318182	108386.931818	127.295455
Manchester UFC	22.375000	182.321429	66.267857	98694.982143	118.928571
Al-Rayyan Sports Club	30.666667	186.333333	78.333333	98606.333333	127.666667
Liverpool	23.044444	181.888889	48.355556	84998.777778	127.688889
A. Madrid	24.794872	181.000000	65.025641	84920.487179	134.948718
Barcelona	22.076923	179.653846	56.634615	82630.711538	128.346154
Chelsea	21.650794	182.174603	56.714286	80974.777778	119.111111
Tottenham Hotspur	23.736842	182.736842	58.394737	78315.789474	131.763158

Conditional Statements

Using conditional statements I was able to decipher that amount of players with a salary greater than 50,000. I then found out what did this collate to in terms of the percentage of the total players in the database.

```
#Create conditional statement to return the amount of players with Salary greater than 50000

Summary_fm_2023_df.loc[Summary_fm_2023_df['Salary'] > 50000, 'Salary > 50,000'] = 'True'

print("The total amount of players in the database is :", Summary_fm_2023_df["Club"].count())
print("The amount of players with a salary over 50,000 is :", Summary_fm_2023_df['Salary > 50,000'].notnull().sum())

Percentage_Salary_Insight = round(((Summary_fm_2023_df['Salary > 50,000'].notnull().sum() / Summary_fm_2023_df["Club"].count()) * 100), 1)

print("The percentage of players in football manager 2023 that earn greater than 50,000 is :", Percentage_Salary_Insight,"%")
```

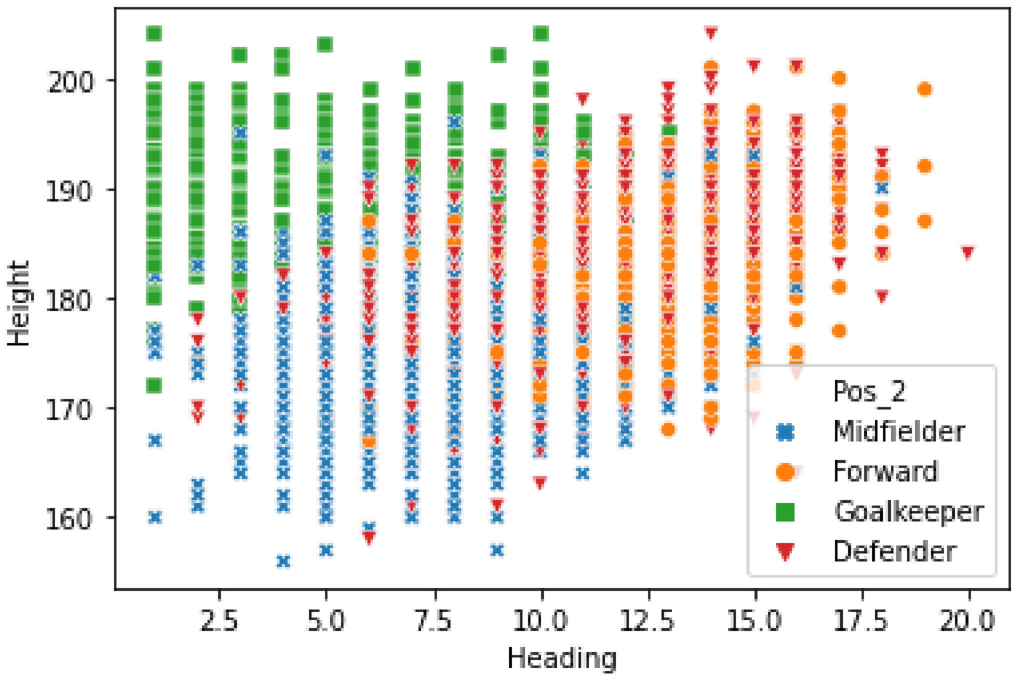
```
The total amount of players in the database is : 8345
The amount of players with a salary over 50,000 is : 814
The percentage of players in football manager 2023 that earn greater than 50,000 is : 9.8 %
```

Results

I wanted to create three charts which looked at the data, first one I wanted to view if there was a correlation between Height, Heading ability and Position. The second to view the Salary split by position. The final chart was to view the ca (current ability) split to see where the majority of the players sit in terms of current ability.

Chart 1 – Seaborn Scatter Plot

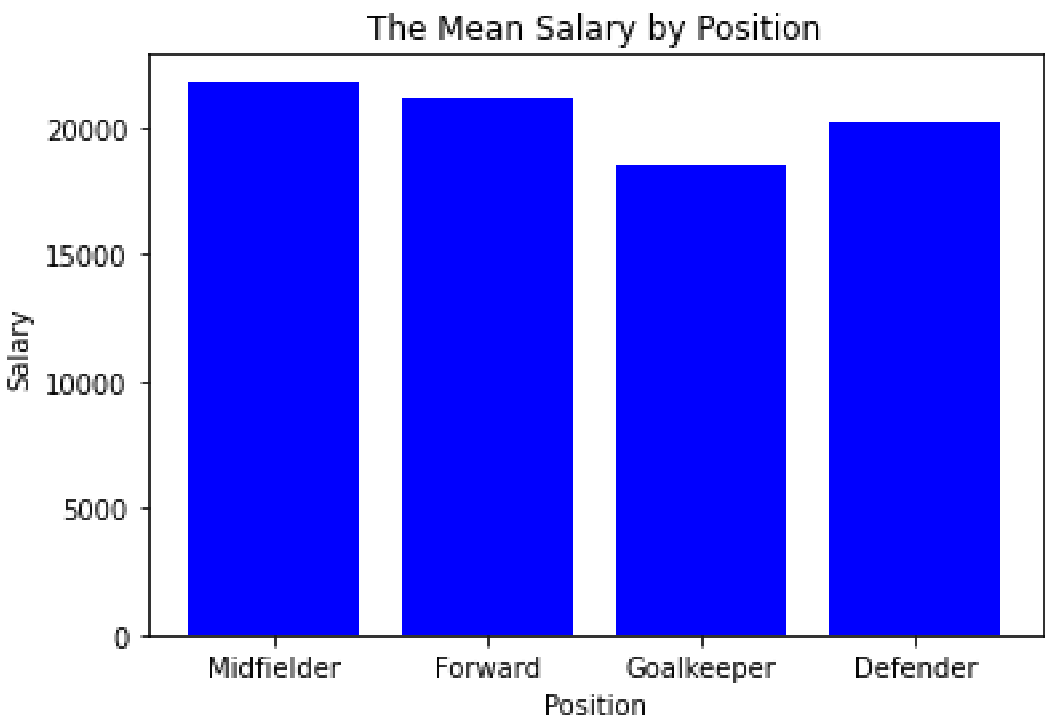
```
#Import seaborn
import seaborn as sns

#Plot Height vs Heading
markers = {"Goalkeeper": "s", "Defender" : "v", "Midfielder" : "X", "Forward" : "o"}
sns.scatterplot(x='Heading', y='Height', data=Summary_fm_2023_df,
                style="Pos_2", hue = "Pos_2", markers=markers)
```



From this chart you can see that in general the taller the player the better the heading ability. You can also see that Goalkeepers are generally tall with a Heading stat of less than 10 and Midfielders are generally in small and have a Heading stat of less than 10.

Chart 2 – Bar Chart

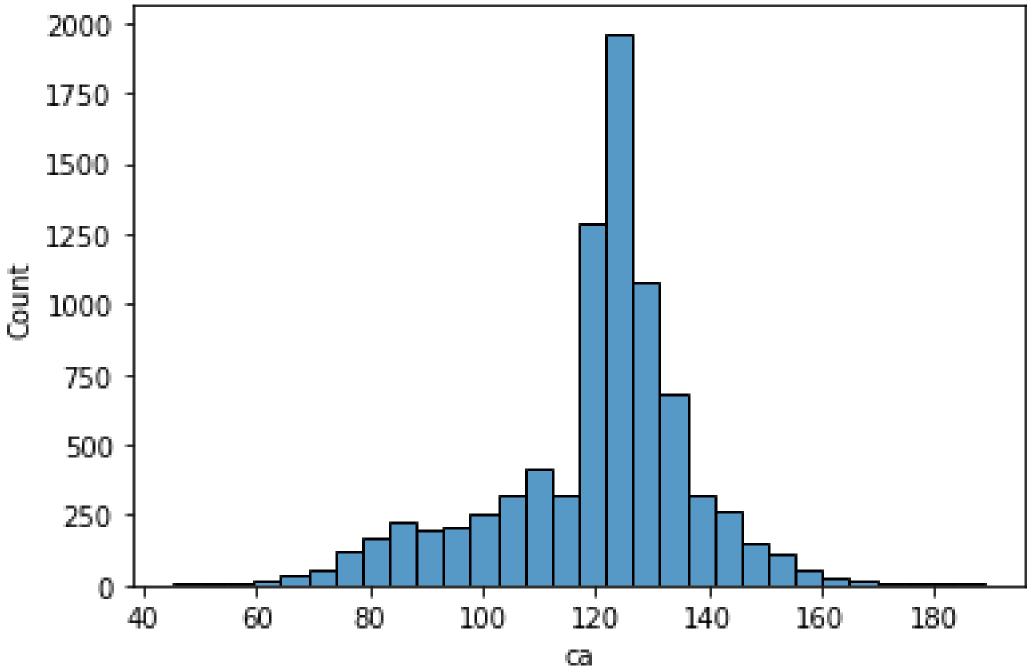
```
import matplotlib.pyplot as plt

Position = list(Summary_fm_2023_df['Pos_2'].drop_duplicates().reset_index(drop=True))
Salary = list(round(Summary_fm_2023_df.groupby('Pos_2').mean()['Salary']))

plt.bar(Position, Salary, color= 'blue')

plt.xlabel('Position')
plt.ylabel('Salary')
plt.title('The Mean Salary by Position')
plt.show()

print(round(Summary_fm_2023_df.groupby('Pos_2').mean()['Salary']))
```



From this chart you can se that based on the Mean salary in this dataset, Goalkeepers earn the less where Midfielders earn the most.

Chart 3 – Histogram

```
sns.histplot(Summary_fm_2023_df['ca'], bins=30)
```



From this chart you can see that majority of the players ca (Current Ability) sits in the 120-140 range with it tailing off significantly after 140.

Insights

1. Only 9.8% of players in Football Manager 2023 earn greater than or equal to 50,000 (814 out of a total 8,345)

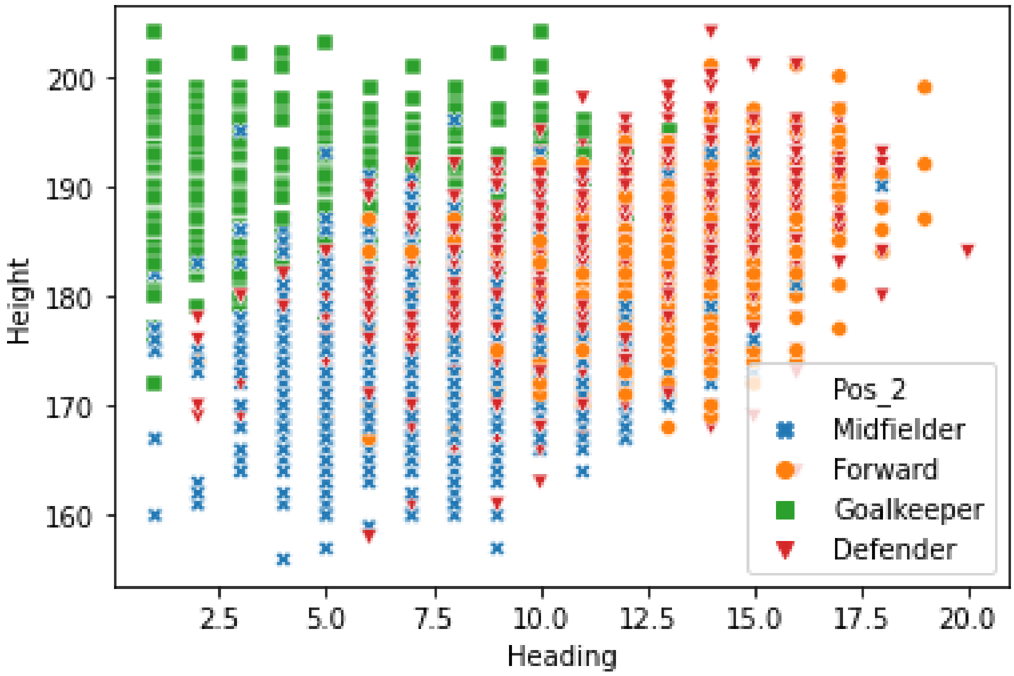
   This can be seen from my conditional statement analysis which I performed.

2. Paris Saint-Germain pays the most salary to the players on average.

   This can be seen from my grouping/sorting analysis by Club.

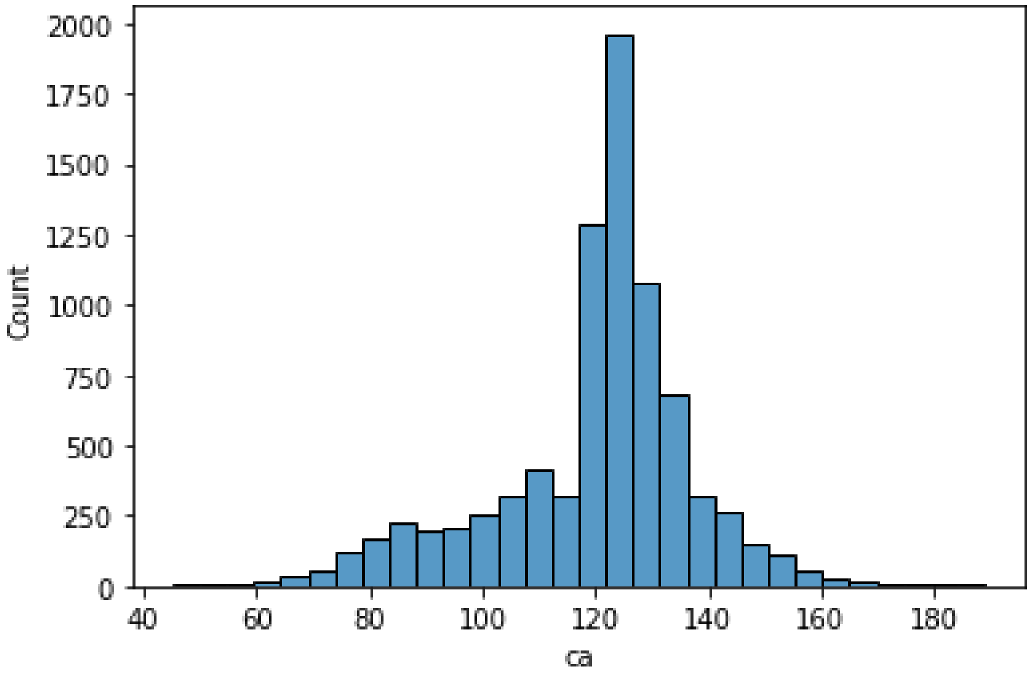
3. The taller a player is the higher the heading rating.

   This can be seen through the scatter plot which was created.



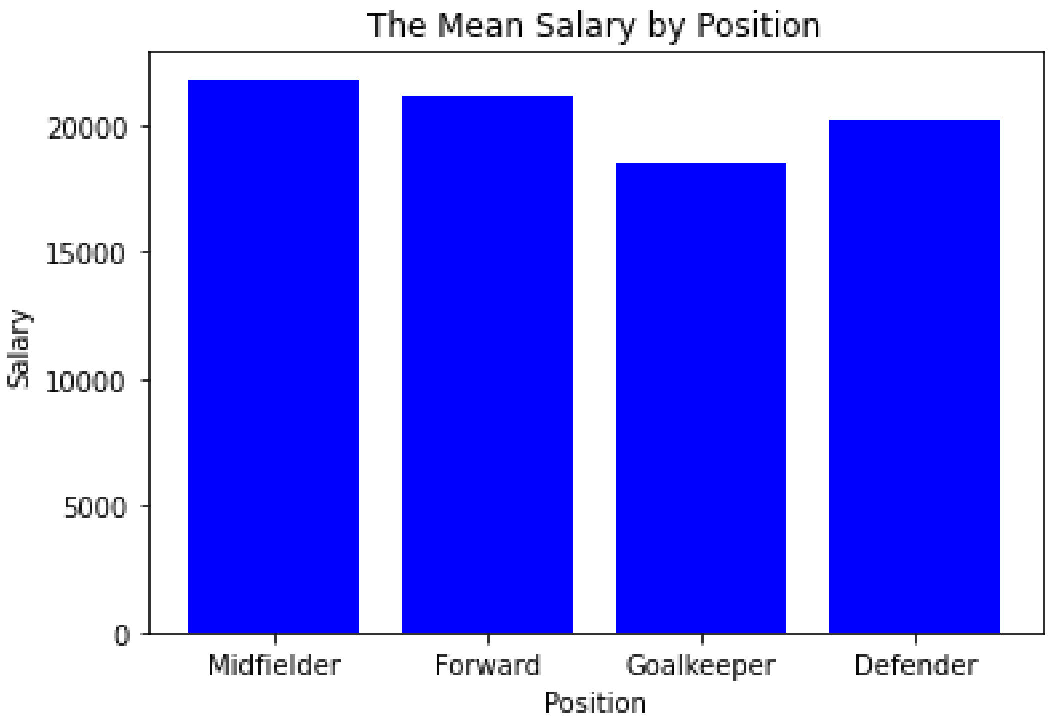
4. The average ca (current Ability) sits around 120-140. With more players ability being less than 120 than greater than 140.

   This can be seen through the Histogram created.



5. Goalkeeper is the position with the lowest mean Salary with Midfielder being the highest.

   This can be seen through the bar chart created.



Machine Learning

The power of Machine Learning has several benefits. The main benefit which I see is in the predictive side of machine learning. One key part of machine learning which I could use is Regression, this is used to predict numerical values. With the high number of numerical data points that are present in this dataset I feel that using the regression method is the best course of action.

An example of the type of prediction that I would perform in the future on this dataset would the salary of a player with certain stats.

The dependent variable in this case would be Salary with the independent variables being the likes of Height, Weight, Nationality, Position and Age. Using machine learning I would be able to predict the salary of a new player into the database with the knowledge of their Height, Weight, Nationality, Position and Age. This will allow me going forward to ensure that the player in my game is getting fair pay.

References

(Include any references if required)

- Stuart, K. (2014) Why clubs are using football manager as a real-life scouting tool, The Guardian. Guardian News and Media. Available at: https://www.theguardian.com/technology/2014/aug/12/why-clubs-football-manager-scouting-tool (Accessed: March 20, 2023). Stuart, K. (2014) *Why clubs are using football manager as a real-life scouting tool*, *The Guardian*. Guardian News and Media. Available at: https://www.theguardian.com/technology/2014/aug/12/why-clubs-football-manager-scouting-tool (Accessed: March 20, 2023).
- Bai, J. (2023) "Football Manager 2023 Dataset." Available at: https://www.kaggle.com/datasets/platinum22/foot-ball-manager-2023-dataset (Accessed: March 20, 2023).