Conor McCarthy

conormccarthy7895@gmail.com

Final Project Report:

GitHub URL:

https://github.com/cesignon/UCDPA_conormccarthy.git

Abstract:

For my final project I have decided to analyse FIFA player data. The data includes information about the player (nationality, reputation, value, etc.) along with player ratings. Straight away, this gave me some good ideas on what insights I wanted to find. Through analysis and trial and error, I feel I have gained valuable insights into FIFA player data. Some possible improvements to my project include:

- Including more FIFA years (15, 16, 17) for more data points
- Applying the '_20' suffix correctly for the FIFA 20 data
- Including the teams and leagues data to possibly compare players in different leagues.
- Modifying the pace vs. value scatter plot to help show a positive correlation – possibly change to a log graph.

Introduction:

I chose to analyse FIFA data because I have a strong interest in sport and was interested in analysing player data to gain insights. As an avid FIFA gamer, I often wondered what variables affect a player's rating or value. More specifically, I wondered what nations have the best player ratings. Also, at what age is a player at their peak rating? Does a player's pace affect their value? All these questions will be answered through graph analysis.

Dataset:

The FIFA dataset I used was obtained from the Kaggle website from the user stefanoleone992. The dataset itself comprises of a zip file which contains 7 csv files. The files include a teams and leagues file which we won't use. Along with this there are 6 csv files containing player data from FIFA 15 all the way up to FIFA 20. I have chosen to analyse the FIFA 18, 19 and 20 files as I feel 3 years of data is sufficient for my analysis.

Implementation Process:

Step 1: Import the Datasets

Firstly, I installed the Kaggle module along with other modules I will need further down the line (Numpy, Pandas, Matplotlib, Seaborn, etc.)

```
In [1]: !pip install kaggle
```

```
In [29]: import pandas as pd
         import matplotlib.pyplot as plt
         import numpy as np
         import seaborn as sns
         from kaggle.api.kaggle_api_extended import KaggleApi

         pd.set_option('display.max_columns', None)
```

I have also set an option using Pandas to display all columns, the reasoning will be evident later. Now to import the datasets from Kaggle. This was achieved using Kaggle API commands. Before coding, I authenticated myself on Kaggle by signing up and creating a username. This allowed me to generate an API key which granted me access to the Kaggle FIFA datasets. I created an instance of the 'KaggleApi' class which allows me to interact with the Kaggle platform programmatically. I then called the 'authenticate()' method of the 'KaggleApi' instance. This allowed me to access Kaggle resources after prompting me to enter my username and API key.

```
In [3]: api = KaggleApi()
        api.authenticate()
```

Now for the dataset download. I used the 'dataset_download_files()' method of the 'KaggleApi' instance to download files from a specific dataset. In my case the specific dataset

identifier was 'stefanoleone992/fifa-20-complete-player-dataset' created by the user ‘stefanoleone992’. I then specified the directory path where I wanted the files downloaded to (path=’./data’). The files are provided in a zipped folder, so I used the simple stipulation: unzip=True to unzip the files. I then used the ‘read_csv()’ function of the pandas library to read the csv and store its contents in a pandas DataFrame (df20, df19, df18).

```
In [4]: api.dataset_download_files('stefanoleone992/fifa-20-complete-player-dataset', path='./data', unzip=True)
In [5]: df20 = pd.read_csv('./data/players_20.csv')
In [6]: df19 = pd.read_csv('./data/players_19.csv')
In [7]: df18 = pd.read_csv('./data/players_18.csv')
```

Step 2: Modify the DataFrames

Firstly, I printed the top few rows of each DataFrame using the ‘head()’ method to see what we are working with. I then looked at the shape of the DataFrame and printed out the individual columns (and their data types) to see what (if any) columns I could think about excluding.

```
In [8]: print(df20.head())
In [10]: df20.shape
In [52]: for column, dtype in df20.dtypes.items():
             print(column, dtype)
```

Upon analysing the individual columns, I figured I can remove several columns that I don’t need in my analysis. On a side note, analysing the data type of each column has given me a head start on thinking about comparisons to make. However, more on this later.

Now to modify the DataFrames. I decided the main columns I wanted to keep are the player stats, player names and their IDs, player value and what country they are from, among others. I used the ‘drop()’ method to remove columns based on the column name utilising the ‘loc()’ method. I needed to include a union to remove a subset of columns along with individual columns. I assigned the modified DataFrames to variables called df**_mod. I then printed out the head and shape of the modified DataFrames to see firstly if the modification worked and secondly to give me a better understanding of what I’m now working with.

```
In [12]: df20_mod = df20.drop(columns=df20.loc[:, ["long_name", "player_url", "player_tags", 'release_clause_eur', 'team_position', 'team_
```

```
In [12]: tract_valid_until', 'nation_position', 'nation_jersey_number']].columns.union(df20.loc[:, "player_traits":"rb"].columns))
```

Step 3: Merge the DataFrames

Now to merge our modified DataFrames. I achieved this through creating a reusable function called 'merge_df'. The following five parameters were set:

- df1: The first DataFrame to be merged.
- df2: The second DataFrame to be merged.
- column_name: The common column name used for the merge operation.
- join_type: The type of join to be performed.
- suff1 and suff2: Suffixes to be added to column names that exist in both DataFrames to differentiate.

I then used the 'merge()' function, passing through the corresponding five parameters to perform the merge operation. This function is reusable if I need to perform additional DataFrame merges, which is needed later.

```
In [20]: def merge_df(df1, df2, column_name, join_type, suff1, suff2):
             merged_df = df1.merge(df2, on=column_name, how=join_type, suffixes=(suff1, suff2))
             return merged_df
```

I then passed through the FIFA 18 and 19 modified DataFrames through an inner join. The reason for this was because I only wanted to retrieve the matching records between the two datasets based on a common column – 'sofifa_id'. I then merged this resulting DataFrame with the FIFA 20 modified DataFrame. I printed the head and the individual columns to check that the join worked and to get an idea of the resulting DataFrame.

```
In [21]: df_18_and_19 = merge_df(df18_mod, df19_mod, "sofifa_id", "inner", "_18", "_19")
```

```
In [23]: df_18_19_20 = merge_df(df_18_and_19, df20_mod, "sofifa_id", "inner", "", "_20")
```

For some reason the suffixes were not applied on the second join. Not to worry, as I can still differentiate the columns as the FIFA 20 columns will not have a suffix applied.

Step 4: Check for Null Values and Populate

I used the 'isnull()' method to create a Boolean table that shoes what columns contain null values. Additionally, I summed all null values and showed the results on a DataFrame to give

me a better visual. This gives me a scope of columns where null values are present. It is important to replace null values so they do not interfere with statistical calculations, data visualization and machine learning algorithms. Again, the importance of this is prevalent later. Upon analysing the columns where the null values are present, I determined that all the columns with null values relate to player stats. Therefore, it is reasonable to replace the null values with a zero integer. I then reproduced the summed null value DataFrame to check that no null values remain.

```
In [30]: # Check for null values

         df_18_19_20.isnull()
```

```
In [31]: # Sum all null values and show this using a DataFrame (Transpose to show columns)

         null_counts = df_18_19_20.isnull().sum()
         null_counts_df = pd.DataFrame(null_counts).T
```

```
In [32]: null_counts_df
```

```
In [33]: # Since the only null values in the dataset relate to stats, it's a good idea to replace the null value with value 0

         df_18_19_20_filled = df_18_19_20.fillna(0)
```

I am now happy with this finalized DataFrame. Let's replace the DataFrame name with df_fifa and get analysing!


```
In [33]: # Since the only null values in the dataset relate to stats, it's a good idea to replace the null value with value 0

         df_18_19_20_filled = df_18_19_20.fillna(0)
```


Step 5: Data Analysis & Plots

Firstly, I ran some simple numpy functions to get a feel for the age & overall columns. I took the FIFA 20 columns as an example.

In [38]:
```
# Lets run numpy functions to get a better feel for the age & overall columns (Take Fifa 20 values as an example)


# Calculate mean values
mean_age_value = np.mean(df_fifa["age"], axis=0)
print("Mean age value in Fifa 20 is:", mean_age_value)

mean_overall_value = np.mean(df_fifa["overall"], axis=0)
print("Mean overall value in Fifa 20 is:", mean_overall_value)


# Calculate the median values
median_age_value = np.median(df_fifa["age"], axis=0)
print("Median age value in Fifa 20 is:", median_age_value)

median_overall_value = np.median(df_fifa["overall"], axis=0)
print("Median overall value in Fifa 20 is:", median_overall_value)


# Calculate the min/max values
min_age_value = np.min(df_fifa["age"], axis=0)
print("Minimum age value in Fifa 20 is:", min_age_value)

max_age_value = np.max(df_fifa["age"], axis=0)
print("Maximum age value in Fifa 20 is:", max_age_value)

min_overall_value = np.min(df_fifa["overall"], axis=0)
print("Minimum overall value in Fifa 20 is:", min_overall_value)

max_overall_value = np.max(df_fifa["overall"], axis=0)
print("Maximum overall value in Fifa 20 is:", max_overall_value)
```

Some interesting stats here are that the youngest player is 18 and the oldest player is 41. Also, the maximum overall value is 94. That's pretty cool! Let's dive further into the analysis.

I used the 'describe()' function to get a DataFrame showing summary statistics for each column. Upon analysing this and the column data types, I decided the first plot I wanted to make was mean overall rating vs. age. I felt that a line plot was the best option because this would show a clear trend analysis of overall rating over time (age). I grouped the 'df_fifa' DataFrame by the age column and calculated the mean value of the 'overall' column for each age group. I then used the 'plot()' function with kind="line" and marker="o" to create the line plot. The markers give a better visual for each individual data point. I modified the axes, title, y ticks and x ticks (using lists) appropriately. I copied this process to create line plots for the corresponding _18 and _19 columns for a comparison.

In [41]:
```
grouped_data_18 = df_fifa.groupby(df_fifa["age_18"])["overall_18"].mean()  # Calculate mean overall rating for each interval


# Line Plot - will give a good visual on max/min values
grouped_data_18.plot(kind="line", marker="o", label="18")
plt.xlabel("Age in Years")
plt.ylabel("Mean Overall Rating")
plt.title("Mean Overall Rating vs. Age (Fifa 18)")
plt.yticks([60, 62, 64, 66, 68, 70, 72])
plt.xticks([16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39])

plt.xticks(rotation=90)

plt.show()
```

I noticed an outlier for the oldest players for each line plot - the data point seemed to be unusually high. Through the 'value_counts()' method, I could conclude that there were simply not enough data points for the oldest players to get a true value for the overall mean.

In [44]:
```
# There seems to be an outlier for the oldest players, let's see why:

value_counts_18 = df_fifa['age_18'].value_counts()
print(value_counts_18)

value_counts_19 = df_fifa['age_19'].value_counts()
print(value_counts_19)

value_counts_20 = df_fifa['age'].value_counts()
print(value_counts_20)

# We can see there are only 4 readings for the oldest players on each fifa, this is not enough data points to accurately create
```

I then decided to create a line plot to show if the mean age of players differs from FIFA 18 data to FIFA 20 data. Firstly, I got the mean of each age column then plotted that against the column index.

In [45]:
```
# Create a line plot to show if the mean age of players differs from Fifa 18 to Fifa 20

mean_age = df_fifa[["age_18", "age_19", "age"]].mean()

plt.plot(mean_age.index, mean_age.values)

plt.xlabel('Fifa Data')
plt.ylabel('Mean Age')
plt.title('Mean Age From Fifa 18 to Fifa 20')

plt.show()

# mean age increases nearly exactly by a year each game
```

I used another line plot (this time using the seaborn module) to plot player value vs. international reputation. In this instance, a line plot shows the trend analysis for player value over the sequential data of international reputation.

In [60]:
```
# Create the line plot with customized style
sns.set_style("whitegrid")
sns.lineplot(x="international_reputation", y="value_eur", data=df_fifa)

plt.xlabel("International Reputation", fontsize=12)
plt.ylabel("Player Value Per 10 Million", fontsize=12)
plt.title("Player Value vs. International Reputation (Fifa 20)", fontsize=14)

plt.show()
```

Now for a bar plot. I chose to compare mean overall rating per nationality. Here, the bar plot compares different categories (nationalities). By sorting these in order, I can get a good visual on what nationality has the highest and lowest ratings. I used the seaborn module and messed around with the parameters to create a nice visual graph.

In [56]:
```
# Create a bar plot to show if the mean rating differs by nationality

selected_nationalities = ["England", "Germany", "France", "Brazil", "Argentina", "Spain", "Republic of Ireland"]


# Filter the DataFrame to include only the selected nationalities
filtered_df = df_fifa[df_fifa["nationality"].isin(selected_nationalities)]

# Calculate the average rating for each nationality
avg_rating = filtered_df.groupby("nationality")["overall"].mean().reset_index()

# Sort the DataFrame by the overall mean rating in ascending order
sorted_avg_rating = avg_rating.sort_values("overall", ascending=True)

# Create the bar plot using Seaborn
sns.barplot(x="nationality", y="overall", data=sorted_avg_rating, palette="deep")

plt.xlabel("Nationality", fontsize=12)
plt.ylabel("Overall Mean Rating", fontsize=12)
plt.title("Bar Plot of Overall Mean Rating vs. Nationality", fontsize=14)


plt.xticks(rotation=90)

plt.show()
```

For my final plot of pace vs. player value, I decided to go with a scatter plot. I chose this because a scatter plot allows me to visualize the pattern or trend between the variables and assess the correlation (if any) between them. I decided to remove any player value points below 20 million. The reason being there was just too many data points to analyse. I also removed the goalkeepers (pace=0). I once again used the seaborn module and altered the parameters to show a regression line.

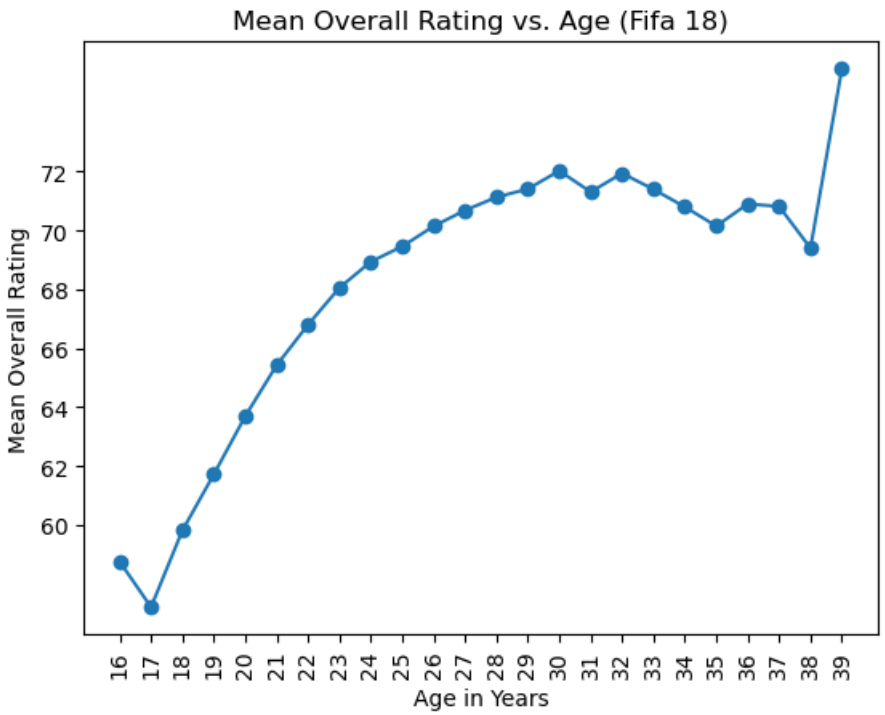
In [49]:
```
# Filter to remove all goalkeepers (pace=0) and players with value under 20 mil (too many data points)

filtered_df = df_fifa[df_fifa["pace"] != 0]
filtered_df = filtered_df[filtered_df["value_eur"] >= 20000000]

sns.regplot(x="pace", y="value_eur", data=filtered_df, scatter_kws={'color': 'blue'}, line_kws={'color': 'red'})

plt.xlabel("Pace", fontsize=12)
plt.ylabel("Player Value in 100 Million Euro", fontsize=12)
plt.title("Scatter Plot of Pace vs. Value (Fifa 20)", fontsize=14)

plt.show()
```

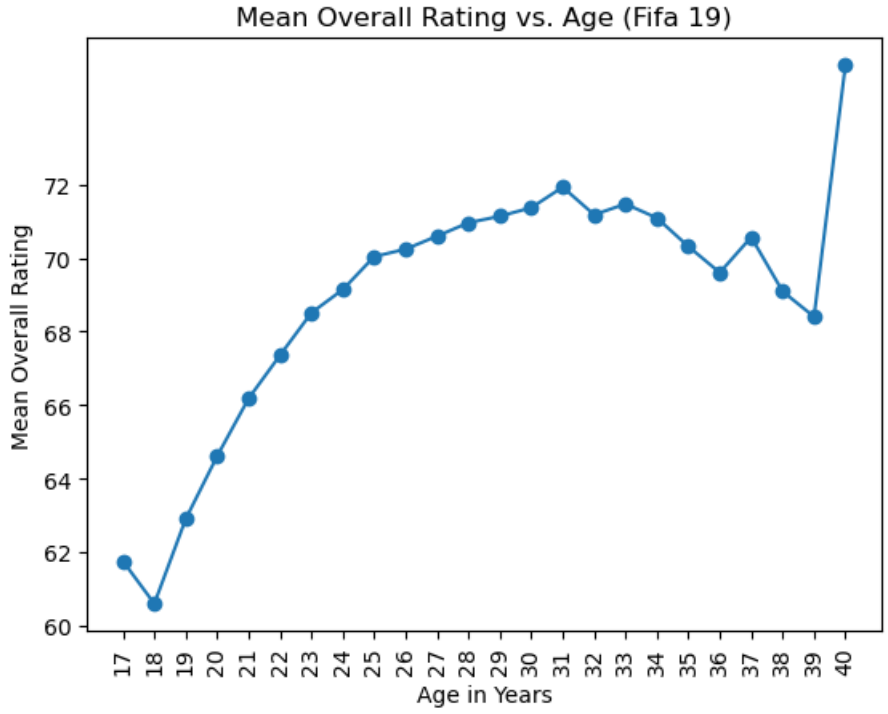
Results:

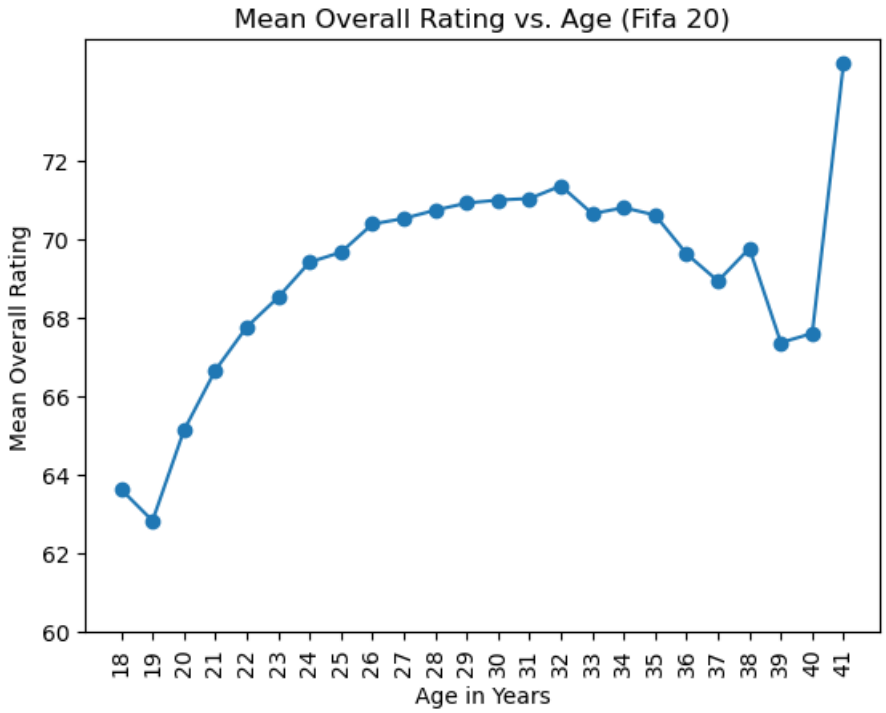
Graph 1 – Mean Overall Rating vs. Age:

The result of the 3 graphs were very similar. Mean overall rating seemed to increase sharply from ages 17 – 24 and then steadily increases until a player reaches their early 30s, then it starts to decline. The highest mean ratings occurred in the below ages:

- Fifa 18 – 30 years
- Fifa 19 – 31 years
- Fifa 20 – 32 years

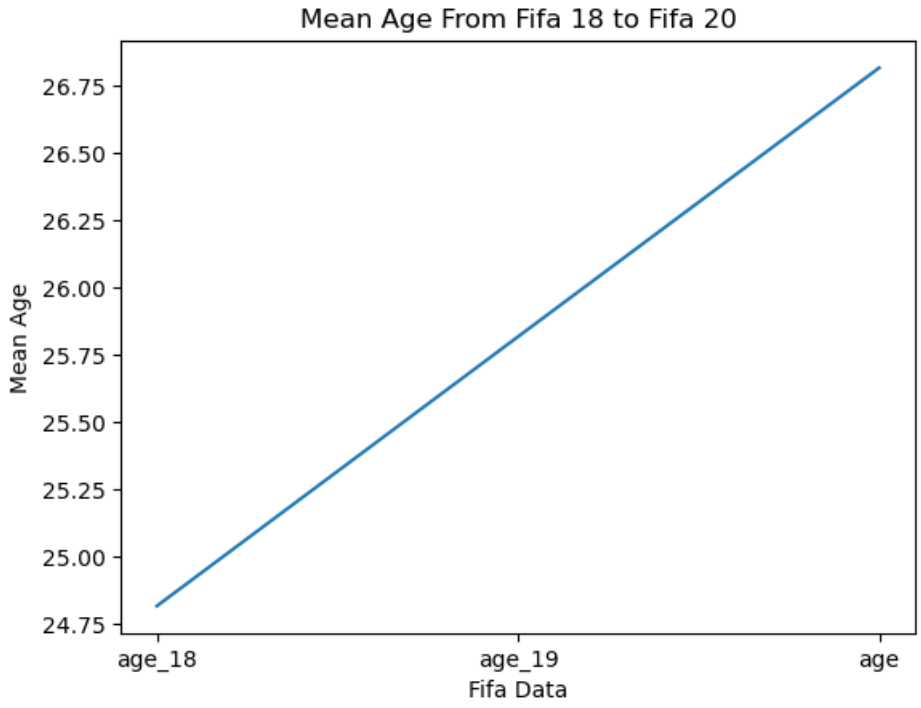

Mean Overall Rating vs. Age (Fifa 18)
Mean Overall Rating
60
62
64
66
68
70
72
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
Age in Years


Mean Overall Rating vs. Age (Fifa 19)
Mean Overall Rating
60
62
64
66
68
70
72
17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
Age in Years



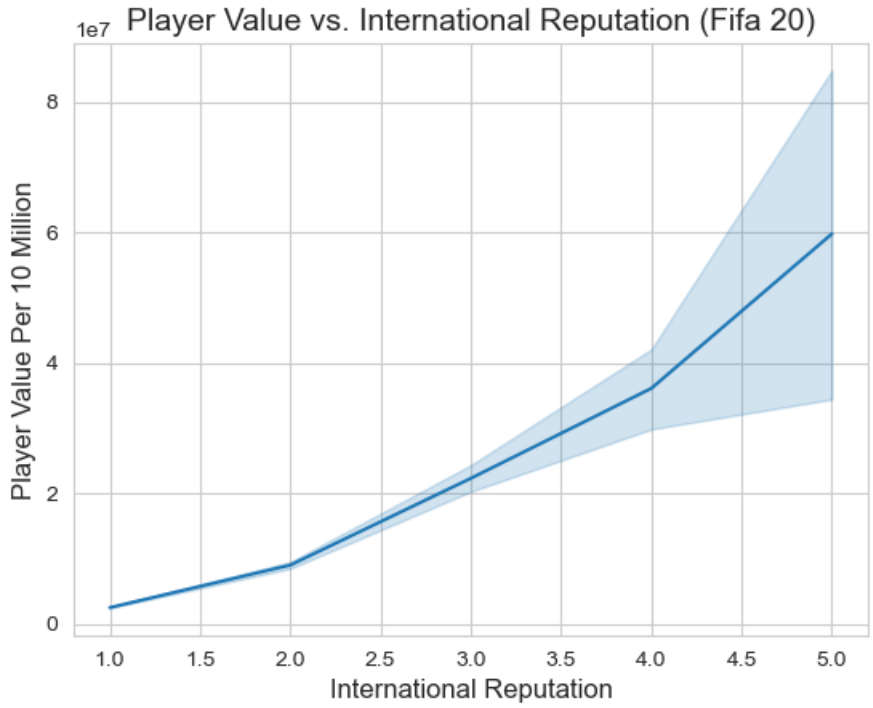
Graph 2 – Mean Age from FIFA 18 to FIFA 20:

The resulting graph here produced a straight increasing line from FIFA 18 to FIFA 20. The mean age increased by almost exactly one year from the previous game.



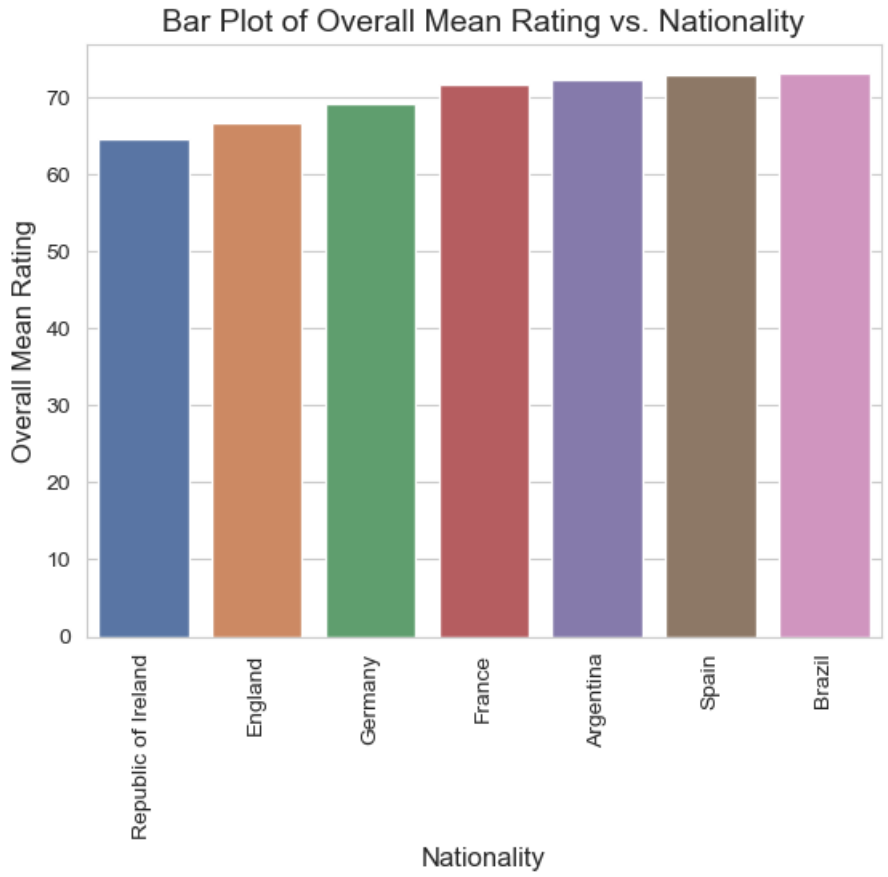
Graph 3 – Player Value vs. International Reputation:

The resulting graph here produced an increasing curved line. This indicates a positive relationship between player value and international reputation. The faded area towards the upper end of the graph suggests a big spread of player value for higher international reputation. This could be due to having a lot of data points for reputations of 4 and 5.



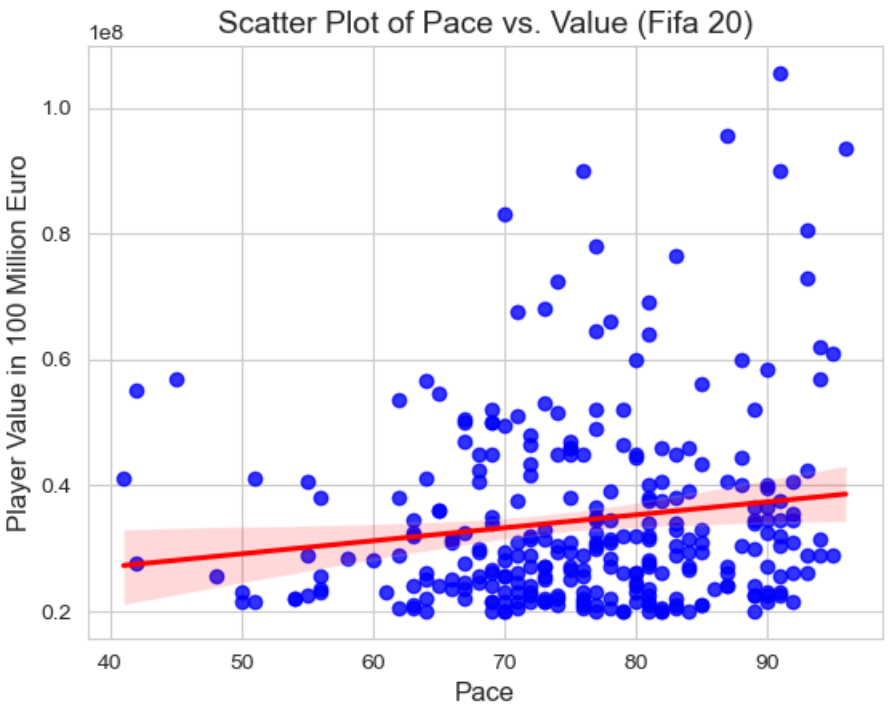
Graph 4 – Mean Overall Rating vs. Nationality:

I ordered this bar chart to see what nations had the best players and who had the worst (in terms of overall rating). The results showed that from the nations I chose to analyse, Brazil had the highest overall mean player ratings, while Republic of Ireland had the lowest (no surprise there). The order from lowest to highest mean ratings was Republic of Ireland, England, Germany, France, Argentina, Spain, Brazil.



Graph 5 – Pace vs. Value:

The result of this scatter plot was that the regression line showed a slight positive correlation. Is this conclusive enough to suggest a positive relationship between the two variables?



Insights:

1. Based on FIFA stats, players improve rapidly from ages 17 – 24, improve steadily until their early 30s at which point they hit their peak. This is because player development is at its most prevalent between the ages of 17-24 when their bodies are developing rapidly. The decline after the age of 31/32 suggests that a player's body is deteriorating.
2. Based on FIFA stats, the mean player age is increasing by approximately 1 year per season. This suggests that career longevity is improving. There could be a number of reasons for this i.e. improvements in sports science, better training facilities, etc.
3. Based on FIFA stats, when a player's international reputation increases, so does their value. Media could be a factor here, as the more reputation a player has internationally, they attract more media coverage which could drive their value upwards.
4. Based of FIFA stats, players from Brazil are the highest rated. This could be due to the quality of their youth systems/academies compared to the rest of the world.
5. Based on FIFA stats, there is no clear relationship between a player's pace and their value. The assumption here was that there would be a positive relationship between the two variables, however we can not say that conclusively based on the regression line.

References:

- *Research Gate – Data Analysis Methods for Qualitative Research*
- *Dartmouth – Introduction to Data Analysis with Python*
- *Tutorialspoint – Machine Learning with Python*
- *University of Glasgow – data Programming in Python Plotting in Python*