Review 2

CSE3020 – Data Visualization

Slot: C1/C2

Faculty: Lydia Jane G

**Visualization of Player Data in FIFA-19 game using Python and R**

# Group Members

1. Aditi Verma – 20BCE0813
2. Amul Thatai – 20BCE0837
3. Rishabh Manish Agrawal – 20BCE802

# Dataset Details

URL: <https://www.kaggle.com/karangadiya/fifa19>

Number of attributes: 88

Number of Rows: 18,207

## Attributes Information

Detailed attributes for every player registered in the latest edition of FIFA 19 database.

Our dataset consists of 88 attributes. The categorical attributes used to index the players are ‘name’ and ‘player ID’. Player ID is unique for each player.

25 attributes are performance ratings of the player at a specific position. For example, the attribute ‘GK’ numerically denotes the ability of the player to perform as a Goalkeeper. Other such attributes are RM (right mid), LW(left wing), RF(right forward), and many more.

Data regarding age, nationality, monetary value, current club, lease period, joining date, etc. is also provided in the data set. Skill wise scores for all players are also provided in the dataset, for example, Crossing, Finishing, Dribbling etc.

All these attributes provide us with a detailed and extensive data set to analyze and finally find out what would be the ‘perfect’ FIFA-19 team.

# Data Abstraction

|  |  |
| --- | --- |
| Attribute Name | Attribute Type |
| ID | Categorical |
| Name | Categorical |
| Age | Numerical (Quantitative ordered) |
| Overall | Numerical (Quantitative ordered) |
| Potential | Numerical (Quantitative ordered) |
| International Reputation | Numerical (Quantitative ordered) |
| Height | Numerical (Quantitative ordered) |
| Weight | Numerical (Quantitative ordered) |
| Position Data (LS, GK, ST, RM, LW, LT) | Numerical (Quantitative ordered) |
| Skill Data (Dribbling, Passing, Volleying, Shooting, Agility) | Numerical (Quantitative ordered) |
| Club | Ordinally ordered |
| Nationality | Categorical |
| Image Data (Real Face, Photo, Flag, Club Logo) | Categorical Graphics |

# Task Abstraction

1. From which country are the majority of the players from in FIFA-19?
   1. Home country data of all players needed.
2. What is the distribution of age of players in FIFA-19?
   1. Age data of all players needed.
3. Which is the preferred foot of a greater number of players, left or right?
   1. Preferred foot data of all players needed.
4. How are the players distributed in terms of positions?
   1. Position data of all players needed.
5. Who are the top 10 players for each position?
   1. Positional score data of all players needed.
6. What is the financial distribution of release values among Real Madrid players?
   1. Net value data of Real Madrid players needed.
   2. Average Value of all players in the world?
7. How is the overall rating distributed among all the players?
   1. Overall rating data of all players needed.
8. What is correlation between weight and height of players
   1. Weight and height data of all players needed
9. Obtain the attribute wise graphic of any player
   1. All details of a player needed
10. What is the best possible team line-up that can be formed with some specific resources?
    1. All attributes need to be collectively analysed to determine this

# Encoding

# Import Modules

import warnings

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

import plotly.graph\_objects as go

import scipy as sp

import seaborn as sns

from PIL import Image

from plotly.subplots import make\_subplots

import plotly.express as px

warnings.filterwarnings("ignore")

# Read FIFA 19 dataset and show info and describe columns from it.

df = pd.read\_csv(r"Datasets\data.csv", index\_col="Unnamed: 0")

## Q1

# Calculate top 10 countries sorted by most players in the game

# Group data by Nationality and sort it by number of players to get most countries having players.

national\_players = (

    df[["Nationality", "ID"]]

    .groupby(by=["Nationality"], as\_index=False)

    .count()

    .sort\_values("ID", ascending=False)

)

national\_players.rename(

    columns={"Nationality": "country", "ID": "player\_count"}, inplace=True

)

national\_players = national\_players.reset\_index()

national\_players = national\_players.drop(["index"], axis=1)

# Slicing first 10 rows from country player\_count dataset

player\_count = national\_players.iloc[0:10, 1]

nation = national\_players.iloc[0:10, 0]

# Select seaborn style of chart to make display easy on the eyes.

temp\_df = pd.DataFrame(

    list(zip(list(player\_count), list(nation))), columns=["PlayerCount", "Nation"]

)

temp\_df.to\_csv(r"Outputs\MostPlayersNation.csv")

fig = px.bar(

    temp\_df,

    x="Nation",

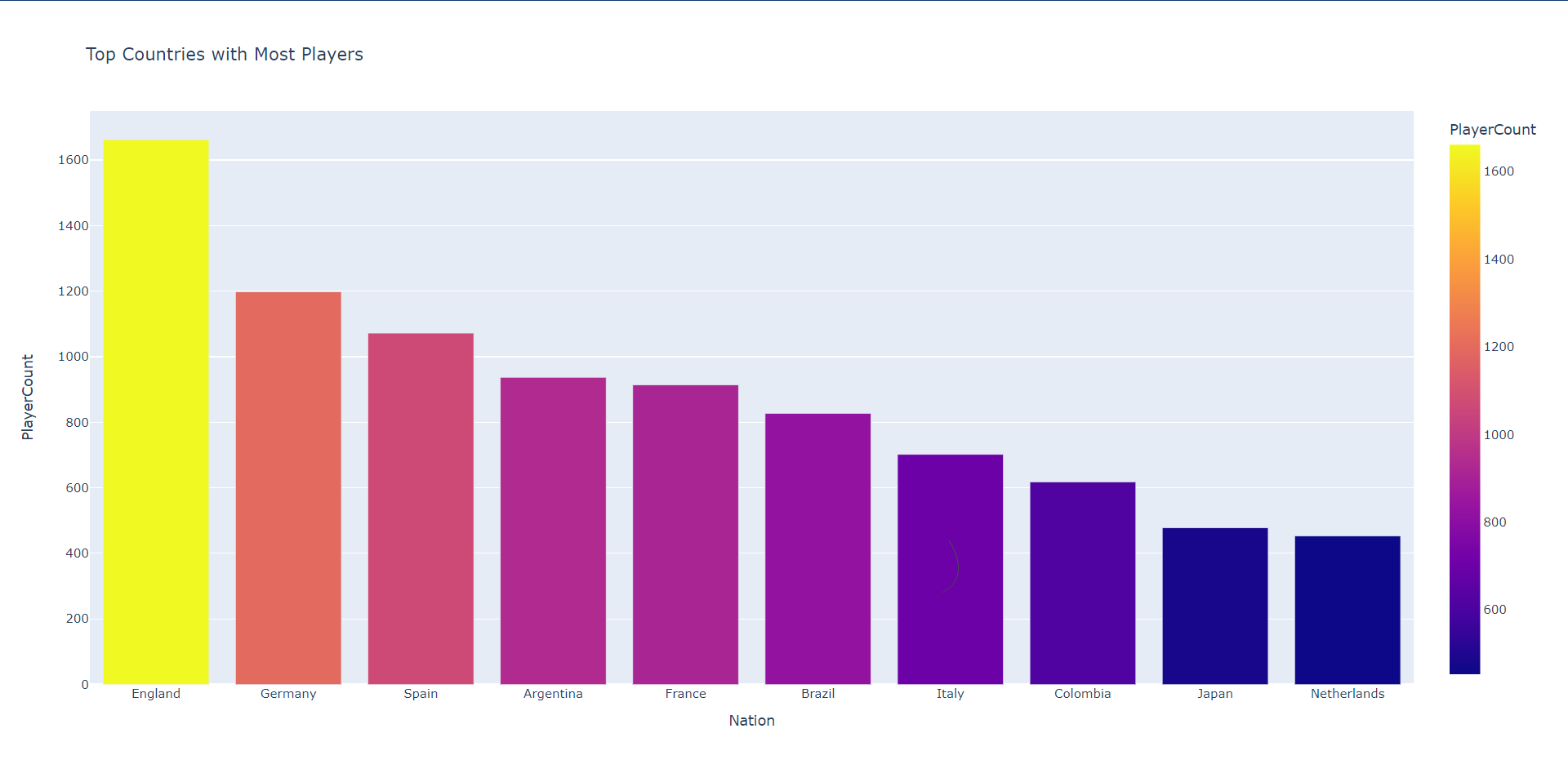
    y="PlayerCount",

    color="PlayerCount",

    title="Top Countries with Most Players",

)

fig.show()



It can hence be seen that England has the greatest number of players

## Q2

# Show Distribution of Age for all players

# Slicing Age column and group it and count no. of players that have same age for all ages.

player\_ages = (

    df[["Age", "ID"]]

    .groupby(by=["Age"], as\_index=False)

    .count()

    .sort\_values("ID", ascending=False)

)

player\_ages.rename(columns={"ID": "count"}, inplace=True)

player\_ages = player\_ages.reset\_index().drop(["index"], axis=1)

# Display histogram of age for all players and fit a normal distribution line for it.

\_, bins, \_ = plt.hist(

    df.Age, bins=df.Age.max() - df.Age.min(), label="Age with no. of player"

)

mu, sigma = sp.stats.norm.fit(df.Age)

best\_fit\_line = sp.stats.norm.pdf(bins, mu, sigma)

plt.plot(bins, df.shape[0] \* best\_fit\_line, label="fit\_line", color="red")

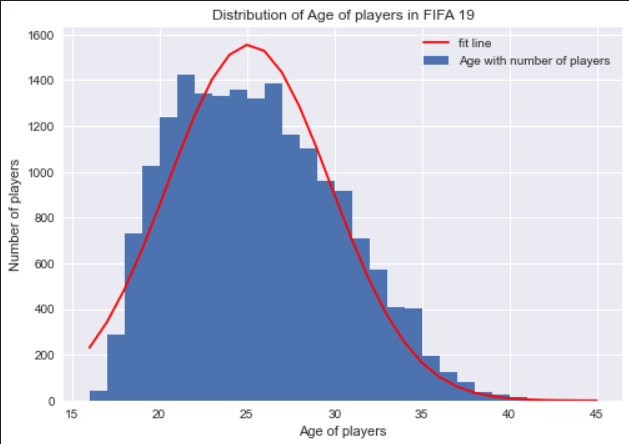
plt.title("Distribution of Age of players in FIFA 19")

plt.ylabel("Number of players")

plt.xlabel("Age of players")

plt.legend()

plt.show()



It can hence be concluded that the maximum number of players are middle aged, i.e., in the range of 25-30 years

## Q3

# Preferred foot analysis.

# Count number of left and right foot preferred players

preferred\_foot = df.groupby("Preferred Foot")[

    "Preferred Foot"].count().to\_list()

temp\_df = pd.DataFrame(

    list(zip(["Left", "Right"], preferred\_foot)), columns=["Foot", "Count"]

)

temp\_df.to\_csv(r"Outputs\PreferredFootAnalysis.csv")

# Plot pie chart to display the percentage for the preferred foot

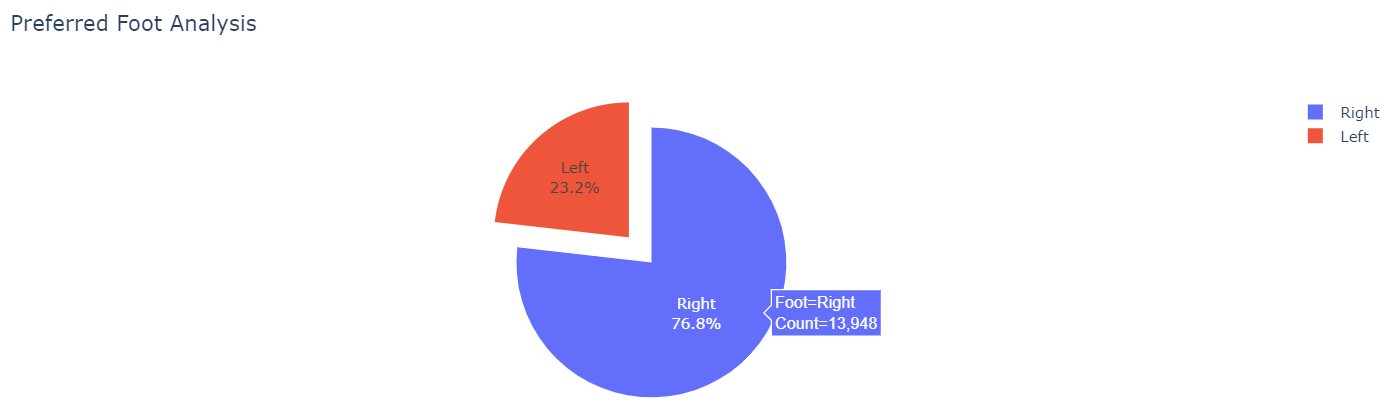
fig = px.pie(temp\_df, values="Count", names="Foot",

             title="Preferred Foot Analysis")

fig.update\_traces(textposition="inside",

                  textinfo="percent+label", pull=[0.25, 0])

fig.show()

It can be clearly seen that most of the players prefer their right foot (76.81%). The count for the respective foot is shown with a mouse hover.

Q4 plot bar chart to display the number of players for every position.

# Show positions with the greatest number of players

# Count number of players for every position in playground that have players and sort it.

player\_position = (

    df[["Position", "ID"]]

    .groupby(by=["Position"], as\_index=False)

    .count()

    .sort\_values("ID", ascending=False)

)

player\_position.rename(columns={"ID": "count"}, inplace=True)

player\_position = player\_position.reset\_index().drop(["index"], axis=1)

# Plot bar chart to display the number of players for every position.

temp\_df = pd.DataFrame(

    list(zip(list(player\_position["Position"]),

         list(player\_position["count"]))),

    columns=["Position", "Count"],

)

temp\_df.to\_csv(r"Outputs\PlayerPositionAnalysis.csv")

fig = px.bar(

    temp\_df,

    x="Position",

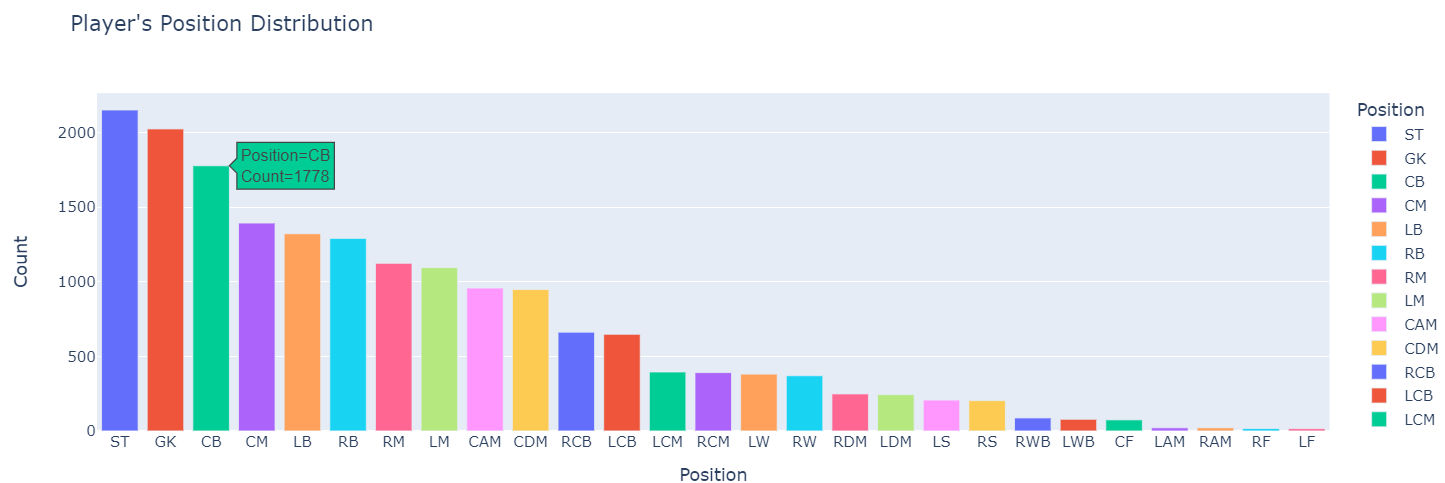
    y="Count",

    color="Position",

    title="Player's Position Distribution",

)

fig.show()



The plot shows the number of players at each position, showing the respective position and player count with a mouse hover over the bar plot.

## Q5

# Top 10 players for ST, GK, LW, RF Position

ST\_position = df[df["Position"] == "ST"].sort\_values("Overall", ascending=False)[

    ["Name", "Overall"]

]

ST\_position = ST\_position.iloc[:10, :]

GK\_position = df[df["Position"] == "GK"].sort\_values("Overall", ascending=False)[

    ["Name", "Overall"]

]

GK\_position = GK\_position.iloc[:10, :]

LW\_position = df[df["Position"] == "LW"].sort\_values("Overall", ascending=False)[

    ["Name", "Overall"]

]

LW\_position = LW\_position.iloc[:10, :]

RF\_position = df[df["Position"] == "RF"].sort\_values("Overall", ascending=False)[

    ["Name", "Overall"]

]

RF\_position = RF\_position.iloc[:10, :]

# Function to plot bar chart for top 10 players in selected positions.

def draw(df, color, position, ax):

    plt.style.use("tableau-colorblind10")

    sns.barplot(df["Name"], df["Overall"], color=color, ax=ax).set\_title(

        "Top 10 " + position + " players", fontsize=8

    )

    ax.set\_xticklabels(ax.get\_xticklabels(), rotation=25,

                       fontdict={"fontsize": 6})

# Plot 4 figures that display Top 10 players in ST, GK, LW, RF positions.

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=[30, 30])

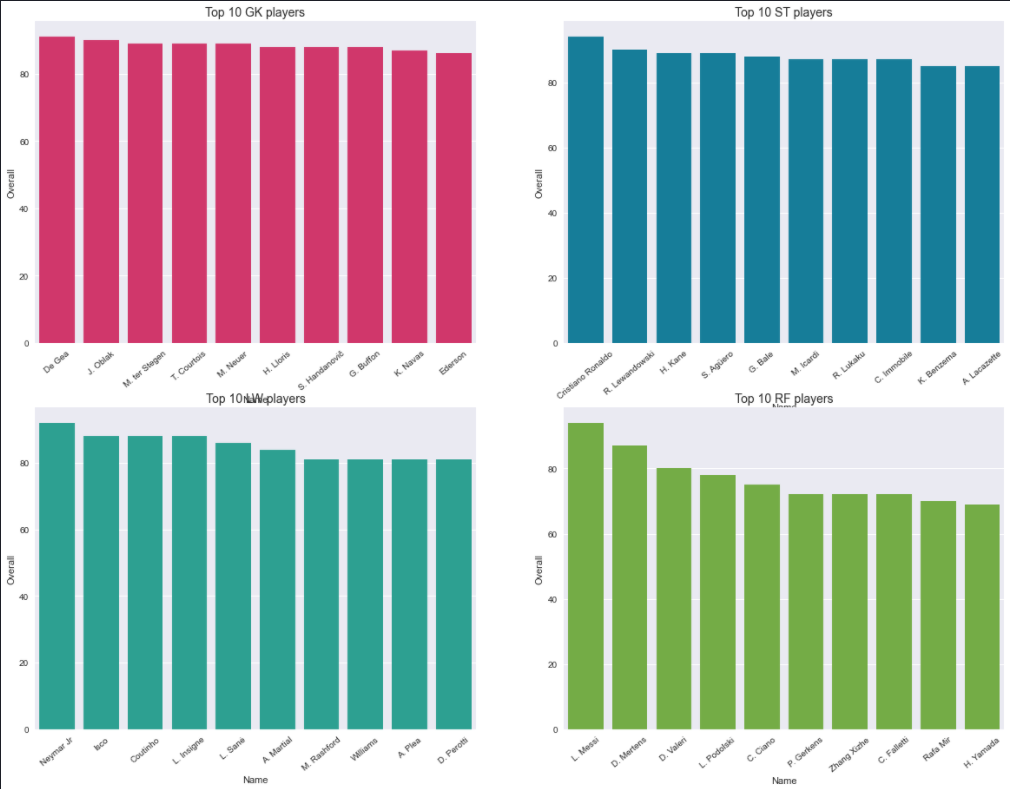
draw(GK\_position, "#e91e63", "GK", axes[0, 0])

draw(ST\_position, "#0089af", "ST", axes[0, 1])

draw(LW\_position, "#1ab39f", "LW", axes[1, 0])

draw(RF\_position, "#72bd35", "RF", axes[1, 1])

plt.show()

The plot shows the top players for each position, along with their overall scores.

## Q6

# Distribution of all player's value and calculate the average value of players.

# Function that converts value column of players to numeric.

def getValue(df):

    new = []

    for i in df:

        i = i.strip("€")

        if "K" in i:

            i = i.strip("K")

            new.append(float(i) \* 1000)

        elif "M" in i:

            i = i.strip("M")

            new.append(float(i) \* (10 \*\* 6))

        else:

            new.append(0.0)

    return new

# Convert value columns to numeric and calculate the average value.

lis = getValue(df.Value.values)

with open(r"Outputs\Report.txt", "w+") as f:

    data = (

        "The average value of players in the world = "

        + str(round(np.average(np.array(lis)) / 10 \*\* 6, 2))

        + "M"

    )

    f.write(data)

    f.write("\n")

The average value of players in the world = 2.41M.

# Make Analysis for Real Madrid Club.

# Select Real Madrid player from data

real\_Madrid\_players = df[df.Club == "Real Madrid"]

real\_Madrid\_players = real\_Madrid\_players[

    ["Name", "Age", "Nationality", "Value", "Release Clause", "Overall"]

]

real\_Madrid\_players.rename(columns={"Release Clause": "Release"}, inplace=True)

# Convert value column to numeric.

real\_Madrid\_players.Release = getValue(real\_Madrid\_players.Release)

# Sort them by Release column.

real\_Madrid\_players.sort\_values("Release", inplace=True, ascending=False)

real\_Madrid\_players = real\_Madrid\_players.reset\_index()

real\_Madrid\_players.drop("index", axis=1, inplace=True)

real\_Madrid\_players.to\_csv(r"Outputs\RealMadridPlayers.csv")

with open(r"Outputs\Report.txt", "a+") as f:

    data = (

        "Overall mean value for Real Madrid Team = "

        + str(round(real\_Madrid\_players.Overall.mean()))

        + "%"

    )

    f.write(data)

    f.write("\n")

    data2 = "Release Corr. Overall= " + str(

        round(real\_Madrid\_players.Release.corr(real\_Madrid\_players.Overall), 2)

    )

    f.write(data2)

    f.write("\n")

# Plot chart for Release Values.

fig = px.bar(

    real\_Madrid\_players,

    x="Name",

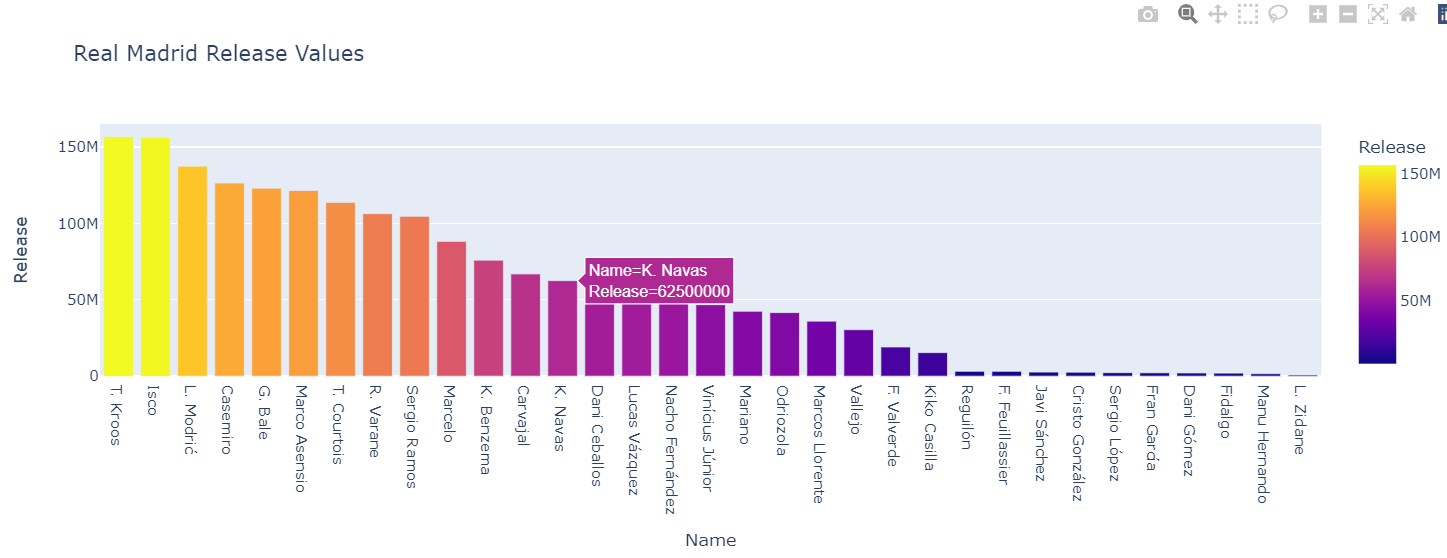
    y="Release",

    color="Release",

    title="Real Madrid Release Values",

)

fig.show()



The plots shows the distribution of release values of all players of Real Madrid.

Overall mean value for Real Madrid Team = 78%

Release Corr. Overall= 0.89

## Q7

# Overall rating distribution and most fit line for it.

# Plot the distribution of overall rating.

fig = px.histogram(

    df,

    x="Overall",

    color="Overall",

    title="Overall rating distribution for all players",

)

fig.show()

# Plot the distribution of overall rating and get most fit line for it.

\_, bins, \_ = plt.hist(

    df.Overall,

    bins=(df.Overall.max() - df.Overall.min()),

    label="Overall",

    color="#0093bc",

)

mu, sigma = sp.stats.norm.fit(df.Overall)

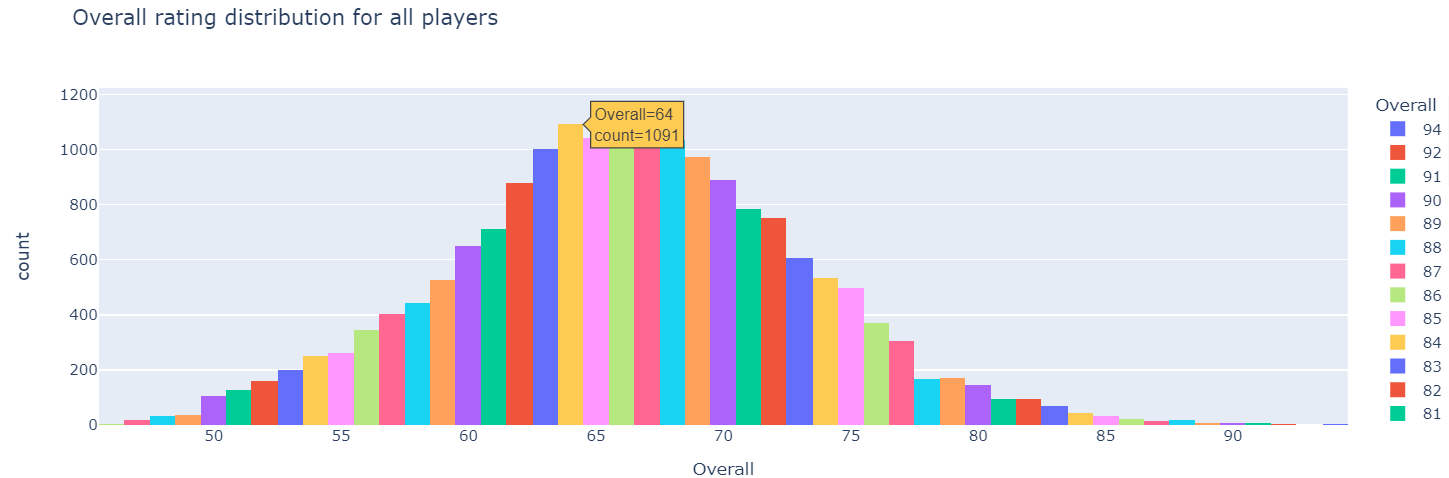
best\_fit\_line = sp.stats.norm.pdf(bins, mu, sigma)

plt.plot(bins, df.shape[0] \* best\_fit\_line, label="fit\_line", color="red")

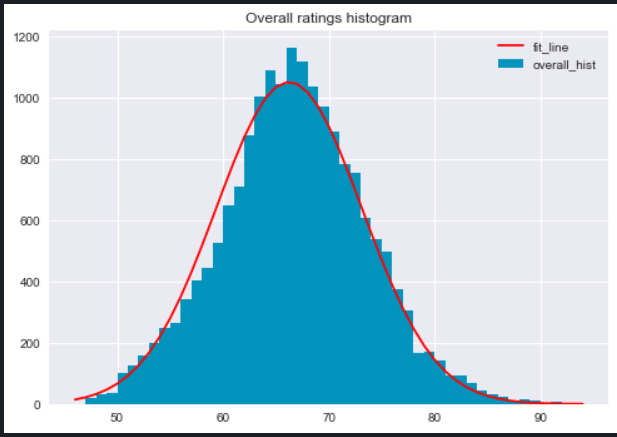
plt.title("Overall ratings histogram")

plt.legend()

plt.show()



The plot shows the overall rating distribution of players and respective count with a mouse hover.



The plot shows the best fit line for the overall ratings which are distributed among the players

## Q8

# Calculate the relation between weight and height for all players(corr. and distribution)

# Function that convert weight to numeric.

def get\_weight(weight):

    new = []

    for i in weight:

        if "lbs" in str(i):

            new.append(i.strip("lbs"))

    return new

# Function that convert height to numeric.

def get\_height(height):

    new = []

    for i in height:

        if "'" in str(i):

            new.append(i.replace("'", "."))

    return new

# Drop nan values form weight and height columns and convert it too numeric.

weight\_height = df[["Weight", "Height"]].dropna(how="any")

weight\_height.Weight = get\_weight(list(weight\_height.Weight.values))

weight\_height.Height = get\_height(list(weight\_height.Height.values))

# Calculate the correlation.

weight\_height.sort\_values("Weight", ascending=True, inplace=True)

weight\_height.Weight = weight\_height.Weight.astype("float64")

weight\_height.Height = weight\_height.Height.astype("float64")

with open(r"Outputs\Report.txt", "a+") as f:

    data = "Correlation between Weight and Height of players = " + str(

        round(weight\_height.Weight.corr(weight\_height.Height), 2)

    )

    f.write(data)

    f.write("\n")

Correlation between Weight and Height of players = 0.45

## Q9

# Calculate ATTRIBUTE DETAILS for any player you want.

# Attribute dictionary key attribute and values skills columns for every attribute.

attribute\_dict = {

    "shooting": [

        "Positioning",

        "Finishing",

        "ShotPower",

        "LongShots",

        "Volleys",

        "Penalties",

    ],

    "passing": [

        "Vision",

        "Crossing",

        "FKAccuracy",

        "ShortPassing",

        "LongPassing",

        "Curve",

    ],

    "dribbling": [

        "Agility",

        "Balance",

        "Reactions",

        "BallControl",

        "Dribbling",

        "Composure",

    ],

    "defending": [

        "Interceptions",

        "HeadingAccuracy",

        "Marking",

        "StandingTackle",

        "SlidingTackle",

    ],

    "physical": ["Jumping", "Stamina", "Strength", "Aggression"],

}

# Function that calculate attribute for any player: need using player index

def calculate\_attribute(dataframe, player\_index):

    allcols = []

    for i in attribute\_dict.values():

        allcols.extend(i)

    player\_observation = dataframe.loc[player\_index, allcols].astype("int64")

    player\_skills = []

    for i in attribute\_dict.keys():

        lis = attribute\_dict.get(i)

        player\_skills.append(

            int(sum(player\_observation[lis]) / len(player\_observation[lis]))

        )

    return {

        i.upper() + ": " + str(j) + "%": j

        for i, j in zip(attribute\_dict.keys(), player\_skills)

    }

# Function gets skills values for any attribute.

def get\_attributes\_values(attribute, observation):

    return observation.loc[attribute\_dict.get(attribute)].astype("int64")

# Function that plot player attribute: need index of player skills.

def plot\_player\_attribute(player\_index, observation, skills):

    colors = ["#03a309", "#a3037e", "#fd3689", "#ded118", "#474bc9"]

    go.Figure()

    fig = make\_subplots(rows=1, cols=5)

    # Create skills bar chart

    for key, skill\_name, color\_i, column in zip(

        attribute\_dict.keys(), skills, colors, range(1, 6)

    ):

        values = get\_attributes\_values(key, observation)

        fig.add\_trace(

            go.Bar(

                x=values,

                y=attribute\_dict.get(key),

                name=skill\_name,

                marker=go.bar.Marker(

                    color=color\_i, line=dict(color="#454545", width=1)

                ),

                orientation="h",

                width=0.5,

                text=values,

                textposition="auto",

            ),

            row=1,

            col=column,

        )

    # Read image

    img = Image.open("faces/" + str(player\_index) + ".png")

    # Add image

    fig.add\_layout\_image(

        dict(

            source=img,

            xref="paper",

            yref="paper",

            x=1,

            y=1.5,

            sizex=0.5,

            sizey=0.5,

            xanchor="right",

            yanchor="top",

        )

    )

    # Update layout properties

    fig.update\_layout(

        autosize=False,

        height=300,

        width=2300,

        bargap=0.5,

        bargroupgap=0.3,

        barmode="overlay",

        hovermode="x",

        margin=dict(r=0, l=0, b=0, t=100),

        title=(

            {

                "text": observation["Name"] + " Attribute Details",

                "y": 0.9,

                "x": 0.5,

                "xanchor": "right",

                "yanchor": "top",

            }

        ),

    )

    fig.update\_xaxes(range=[0, 100])

    fig.show()

# Player Attributes select player index form dataset

# Draw attribute details for MESSI.

player\_index = 0

player\_skills = calculate\_attribute(df, player\_index)

plot\_player\_attribute(

    player\_index, df.iloc[player\_index], list(player\_skills.keys()))

# Draw attribute details for RONALDO.

player\_index = 1

player\_skills = calculate\_attribute(df, player\_index)

plot\_player\_attribute(

    player\_index, df.iloc[player\_index], list(player\_skills.keys()))

# Draw attribute details for Neymar Jr

player\_index = 2

player\_skills = calculate\_attribute(df, player\_index)

plot\_player\_attribute(

    player\_index, df.iloc[player\_index], list(player\_skills.keys()))

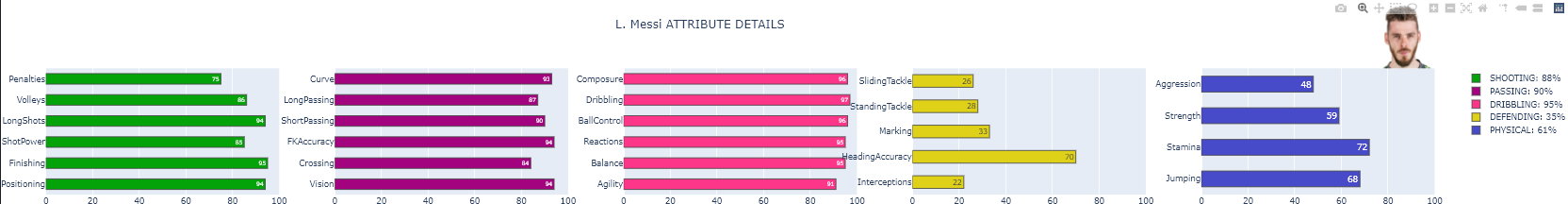
# Draw attribute details for M.SALAH.

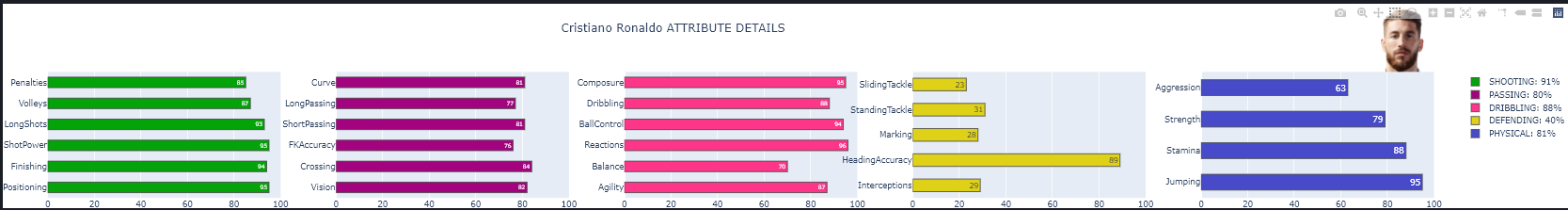
player\_index = 26

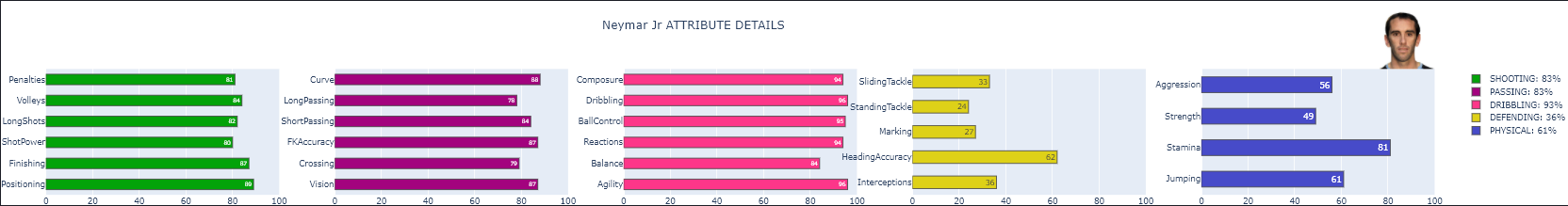
player\_skills = calculate\_attribute(df, player\_index)

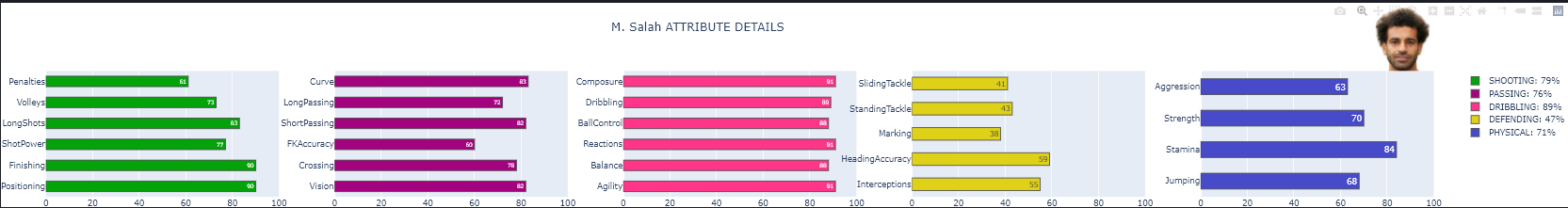
plot\_player\_attribute(

    player\_index, df.iloc[player\_index], list(player\_skills.keys()))









## Q10

# Plot the best squad on playground based on Lineup [3,4,3].

# Location of player on chart.

location\_3\_4\_3 = {

    0: [150, 80],

    1: [150, 145],

    2: [220, 145],

    3: [80, 145],

    4: [60, 300],

    5: [150, 230],

    6: [240, 300],

    7: [150, 320],

    8: [60, 400],

    9: [150, 450],

    10: [240, 400],

}

# Create figure

fig = go.Figure()

# Constants

img\_width = 900  # 900

img\_height = 1200  # 1200

scale\_factor = 0.4

# Add plot

# Noinspection PyTypeChecker

fig.add\_trace(

    go.Scatter(

        x=[0, img\_width \* scale\_factor],

        y=[0, img\_height \* scale\_factor],

        mode="markers",

        marker\_opacity=0,

    )

)

# Configure axes

fig.update\_xaxes(visible=False, range=[0, img\_width \* scale\_factor])

# Disable y-axis visible

fig.update\_yaxes(visible=False, range=[

                 0, img\_height \* scale\_factor], scaleanchor="x")

# Add player image

for i in range(11):

    img = Image.open("faces/" + str(i) + ".png")

    fig.add\_layout\_image(

        dict(

            x=location\_3\_4\_3[i][0],

            y=location\_3\_4\_3[i][1],

            sizex=60,

            sizey=60,

            xref="x",

            yref="y",

            opacity=1.0,

            layer="above",

            source=img,

        )

    )

# Add background image

img = Image.open(r"field.jpg")

fig.add\_layout\_image(

    dict(

        x=0,

        sizex=img\_width \* scale\_factor,

        y=img\_height \* scale\_factor,

        sizey=img\_height \* scale\_factor,

        xref="x",

        yref="y",

        opacity=1.0,

        layer="below",

        sizing="stretch",

        source=img,

    )

)

# Configure another layout

fig.update\_layout(

    width=img\_width \* scale\_factor,

    height=img\_height \* scale\_factor,

    margin={"l": 0, "r": 0, "t": 50, "b": 0},

    title\_font\_size=15,

    title\_font\_family="Dosis",

    title=(

        {

            "text": "---Best Squad in The World for Lineup[3,4,3]---",

            "y": 0.95,

            "x": 0.5,

            "xanchor": "center",

            "yanchor": "top",

        }

    ),

)

fig.show()

