Messi vs Ronaldo

Lakhi Charan Mahato

Dataset(s)

- Soccer Dataset

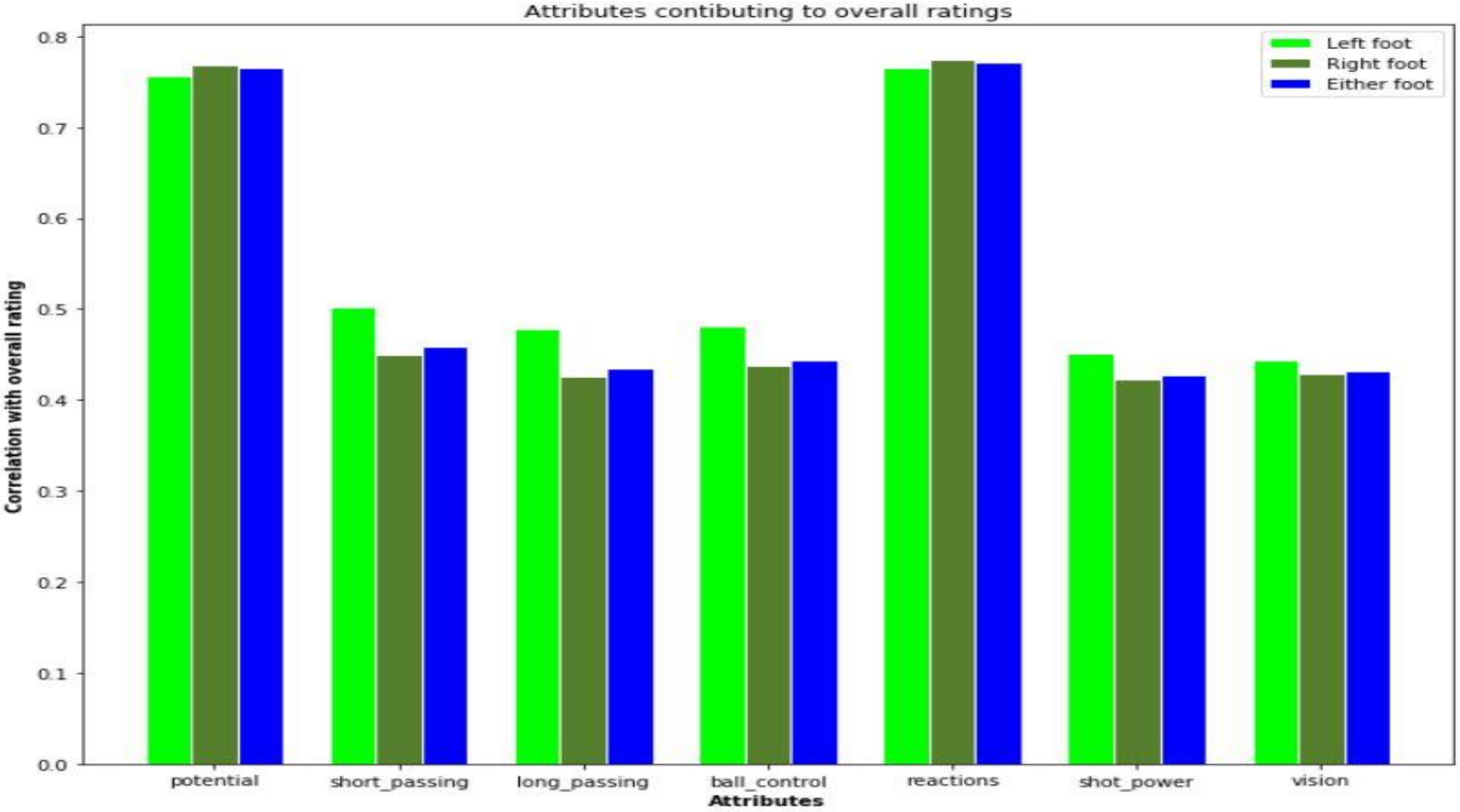
Motivation

Lionel Messi uses left foot, whereas Christiano Ronaldo uses his right foot to play football. Both of them are equally great. However Messi seems to be an effortless player of the ball. Does Messi have an upper hand because he uses his left foot? Through the stats of left footed and right footed players we can see what factors actually govern the overall ratings of left footed players or right footed players. This might break the enigma of whether the left footed players actually have an upper hand in the play because the stats never lie.

Research Question(s)

What factors govern the overall ratings of left footed football players and right footed football players and by how much?

Findings


Attributes contibuting to overall ratings
Left foot
Right foot
Either foot
Correlation with overall rating
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
potential
short_passing
long_passing
ball_control
reactions
shot_power
vision
Attributes

Findings

✓ Attributes which are highly related(having a correlation>0.4) to the overall rating of a football player:

▪ Potential(**Strong** correlation)

▪ Short Passing(Moderate correlation)

▪ Long Passing(Moderate correlation)

▪ Ball control(Moderate correlation)

▪ Reactions(**Strong** correlation)

▪ Shot power(Moderate correlation)

▪ Vision(Moderate correlation)

Findings

✓ Attributes which are more effective in right footed players:

- Potential(**Strong** correlation)
- Reactions(**Strong** correlation)

✓ Attributes which are more effective for left footed players:

- Ball control(Moderate correlation)
- Shot power(Moderate correlation)
- Vision(Moderate correlation)
- Short Passing(Moderate correlation)
- Long Passing(Moderate correlation)

Findings

- ✓ Although overall ratings of right footed players have a greater correlation with potential and reactions, left footed player also have a higher correlation with them but a little less than right footed players. The correlation here is strong(0.60-0.79).
- ✓ Overall Ratings of left footed players have a greater correlation with short passing, long passing, ball control and short power as compared to right footed players. Another notable thing is that the correlation is moderate(0.4-0.59).
- ✓ However vision although greater correlation with overall rating of left footed players but right footed player also close to this correlation. This is also a moderate correlation.

Acknowledgements

I would like to thank the instructors- Ilkay Altintas and Leo Porter for their teachings throughout the course. That was a great help for me in my project. Moreover the jupyter notebooks served as a great reference for my project. I Googled the things which I did not know or whenever I got stuck.

References

I took the help of certain websites whenever I got stuck or wanted to learn something I did not know:

- https://www.researchgate.net/post/What_is_the_minimum_value_of_correlation_coefficient_to_prove_the_existence_of_the_accepted_relationship_between_scores_of_two_of_more_tests
- https://stackoverflow.com/questions/1207406/how-to-remove-items-from-a-list-while-iterating
- https://python-graph-gallery.com/11-grouped-barplot/

In [1]:

```
import sqlite3
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:

```
cnx = sqlite3.connect('database.sqlite')
df = pd.read_sql_query("SELECT * FROM Player_Attributes", cnx)
```

In [3]:

```
df.columns
```

Out[3]:

```
Index(['id', 'player_fifa_api_id', 'player_api_id', 'date', 'overall_rating',
       'potential', 'preferred_foot', 'attacking_work_rate',
       'defensive_work_rate', 'crossing', 'finishing', 'heading_accuracy',
       'short_passing', 'volleys', 'dribbling', 'curve', 'free_kick_accuracy',
       'long_passing', 'ball_control', 'acceleration', 'sprint_speed',
       'agility', 'reactions', 'balance', 'shot_power', 'jumping', 'stamina',
       'strength', 'long_shots', 'aggression', 'interceptions', 'positioning',
       'vision', 'penalties', 'marking', 'standing_tackle', 'sliding_tackle',
       'gk_diving', 'gk_handling', 'gk_kicking', 'gk_positioning',
       'gk_reflexes'],
      dtype='object')
```

In [4]:

```
df.head()
```

Out[4]:

	id	player_fifa_api_id	player_api_id	date	overall_rating	potential	preferred_foot	attacking_work_rate	defensive_work_rate	cro
0	1	218353	505942	2016-02-18 00:00:00	67.0	71.0	right	medium	medium	
1	2	218353	505942	2015-11-19 00:00:00	67.0	71.0	right	medium	medium	
2	3	218353	505942	2015-09-21 00:00:00	62.0	66.0	right	medium	medium	
3	4	218353	505942	2015-03-20 00:00:00	61.0	65.0	right	medium	medium	
4	5	218353	505942	2007-02-22 00:00:00	61.0	65.0	right	medium	medium	

5 rows × 42 columns

In [5]:

```
#Finding the values of denfsive work rate to convert into number
df['defensive_work_rate'].unique()
```

Out[5]:

```
array(['medium', 'high', 'low', '_0', None, '5', 'ean', 'o', '1', 'ormal',
       '7', '2', '8', '4', 'tocky', '0', '3', '6', '9', 'es'],
      dtype=object)
```

In [6]:

```
#Finding the values of attacking work rate to convert into number
df['attacking_work_rate'].unique()
```

Out[6]:

```
array(['medium', 'high', None, 'low', 'None', 'le', 'norm', 'stoc', 'y'],
      dtype=object)
```

Cannnot be converted into numbers to serve much research purpose

In [7]:

```
df.describe().transpose()
```

Out[7]:

	count	mean	std	min	25%	50%	75%	max
id	183978.0	91989.500000	53110.018250	1.0	45995.25	91989.5	137983.75	183978.0
player_fifa_api_id	183978.0	165671.524291	53851.094769	2.0	155798.00	183488.0	199848.00	234141.0
player_api_id	183978.0	135900.617324	136927.840510	2625.0	34763.00	77741.0	191080.00	750584.0
overall_rating	183142.0	68.600015	7.041139	33.0	64.00	69.0	73.00	94.0
potential	183142.0	73.460353	6.592271	39.0	69.00	74.0	78.00	97.0
crossing	183142.0	55.086883	17.242135	1.0	45.00	59.0	68.00	95.0
finishing	183142.0	49.921078	19.038705	1.0	34.00	53.0	65.00	97.0
heading_accuracy	183142.0	57.266023	16.488905	1.0	49.00	60.0	68.00	98.0
short_passing	183142.0	62.429672	14.194068	3.0	57.00	65.0	72.00	97.0
volleys	181265.0	49.468436	18.256618	1.0	35.00	52.0	64.00	93.0
dribbling	183142.0	59.175154	17.744688	1.0	52.00	64.0	72.00	97.0
curve	181265.0	52.965675	18.255788	2.0	41.00	56.0	67.00	94.0
free_kick_accuracy	183142.0	49.380950	17.831746	1.0	36.00	50.0	63.00	97.0
long_passing	183142.0	57.069880	14.394464	3.0	49.00	59.0	67.00	97.0
ball_control	183142.0	63.388879	15.196671	5.0	58.00	67.0	73.00	97.0
acceleration	183142.0	67.659357	12.983326	10.0	61.00	69.0	77.00	97.0
sprint_speed	183142.0	68.051244	12.569721	12.0	62.00	69.0	77.00	97.0
agility	181265.0	65.970910	12.954585	11.0	58.00	68.0	75.00	96.0
reactions	183142.0	66.103706	9.155408	17.0	61.00	67.0	72.00	96.0
balance	181265.0	65.189496	13.063188	12.0	58.00	67.0	74.00	96.0
shot_power	183142.0	61.808427	16.135143	2.0	54.00	65.0	73.00	97.0
jumping	181265.0	66.969045	11.006734	14.0	60.00	68.0	74.00	96.0
stamina	183142.0	67.038544	13.165262	10.0	61.00	69.0	76.00	96.0
strength	183142.0	67.424529	12.072280	10.0	60.00	69.0	76.00	96.0
long_shots	183142.0	53.339431	18.367025	1.0	41.00	58.0	67.00	96.0
aggression	183142.0	60.948046	16.089521	6.0	51.00	64.0	73.00	97.0
interceptions	183142.0	52.009271	19.450133	1.0	34.00	57.0	68.00	96.0
positioning	183142.0	55.786504	18.448292	2.0	45.00	60.0	69.00	96.0
vision	181265.0	57.873550	15.144086	1.0	49.00	60.0	69.00	97.0
penalties	183142.0	55.003986	15.546519	2.0	45.00	57.0	67.00	96.0
marking	183142.0	46.772242	21.227667	1.0	25.00	50.0	66.00	96.0
standing_tackle	183142.0	50.351257	21.483706	1.0	29.00	56.0	69.00	95.0
sliding_tackle	181265.0	48.001462	21.598778	2.0	25.00	53.0	67.00	95.0
gk_diving	183142.0	14.704393	16.865467	1.0	7.00	10.0	13.00	94.0
gk_handling	183142.0	16.063612	15.867382	1.0	8.00	11.0	15.00	93.0

	count	mean	std	min	25%	50%	75%	max
gk_kicking	183142.0	20.998362	21.452980	1.0	8.00	12.0	15.00	97.0
gk_positioning	183142.0	16.132154	16.099175	1.0	8.00	11.0	15.00	96.0
gk_reflexes	183142.0	16.441439	17.198155	1.0	8.00	11.0	15.00	96.0

In [8]:

```
#Deleting rows with null values
rows = df.shape[0]
df = df.dropna()
print(rows - df.shape[0])
```

```
3624
```

In [9]:

```
#Storing left foot data in one table and right foot in another
dfl=df[df['preferred_foot']=='left']
dfr=df[df['preferred_foot']=='right']
```

In [10]:

```
#Storing all column headings into a list for use in graphs and loops
list=['id', 'player_fifa_api_id', 'player_api_id', 'date', 'overall_rating',
       'potential', 'preferred_foot', 'attacking_work_rate',
       'defensive_work_rate', 'crossing', 'finishing', 'heading_accuracy',
       'short_passing', 'volleys', 'dribbling', 'curve', 'free_kick_accuracy',
       'long_passing', 'ball_control', 'acceleration', 'sprint_speed',
       'agility', 'reactions', 'balance', 'shot_power', 'jumping', 'stamina',
       'strength', 'long_shots', 'aggression', 'interceptions', 'positioning',
       'vision', 'penalties', 'marking', 'standing_tackle', 'sliding_tackle',
       'gk_diving', 'gk_handling', 'gk_kicking', 'gk_positioning',
       'gk_reflexes']
```

In [11]:

```
for i in list:
    print(i,"=",type(df[i][0]))
```

```
id = <class 'numpy.int64'>
player_fifa_api_id = <class 'numpy.int64'>
player_api_id = <class 'numpy.int64'>
date = <class 'str'>
overall_rating = <class 'numpy.float64'>
potential = <class 'numpy.float64'>
preferred_foot = <class 'str'>
attacking_work_rate = <class 'str'>
defensive_work_rate = <class 'str'>
crossing = <class 'numpy.float64'>
finishing = <class 'numpy.float64'>
heading_accuracy = <class 'numpy.float64'>
short_passing = <class 'numpy.float64'>
volleys = <class 'numpy.float64'>
dribbling = <class 'numpy.float64'>
curve = <class 'numpy.float64'>
free_kick_accuracy = <class 'numpy.float64'>
long_passing = <class 'numpy.float64'>
ball_control = <class 'numpy.float64'>
acceleration = <class 'numpy.float64'>
sprint_speed = <class 'numpy.float64'>
agility = <class 'numpy.float64'>
reactions = <class 'numpy.float64'>
balance = <class 'numpy.float64'>
shot_power = <class 'numpy.float64'>
jumping = <class 'numpy.float64'>
stamina = <class 'numpy.float64'>
strength = <class 'numpy.float64'>
long_shots = <class 'numpy.float64'>
aggression = <class 'numpy.float64'>
interceptions = <class 'numpy.float64'>
positioning = <class 'numpy.float64'>
```

```
vision = <class 'numpy.float64'>
penalties = <class 'numpy.float64'>
marking = <class 'numpy.float64'>
standing_tackle = <class 'numpy.float64'>
sliding_tackle = <class 'numpy.float64'>
gk_diving = <class 'numpy.float64'>
gk_handling = <class 'numpy.float64'>
gk_kicking = <class 'numpy.float64'>
gk_positioning = <class 'numpy.float64'>
gk_reflexes = <class 'numpy.float64'>
```

In [12]:

```
#Removing ids because it is of no use in further research. Moreover overall_rating is the target v
aribale.
list.remove('id')
list.remove('player_fifa_api_id')
list.remove('player_api_id')
list.remove('overall_rating')

#The Strings present do not serve much help because they are a large varieties of them
#and cannot be converted into numbers to serve much research purpose
for i in list[:]:
    if(type(df[i][0]) is str):
        list.remove(i)

for i in list:
    print(i,"=",type(df[i][0]))
```

```
potential = <class 'numpy.float64'>
crossing = <class 'numpy.float64'>
finishing = <class 'numpy.float64'>
heading_accuracy = <class 'numpy.float64'>
short_passing = <class 'numpy.float64'>
volleys = <class 'numpy.float64'>
dribbling = <class 'numpy.float64'>
curve = <class 'numpy.float64'>
free_kick_accuracy = <class 'numpy.float64'>
long_passing = <class 'numpy.float64'>
ball_control = <class 'numpy.float64'>
acceleration = <class 'numpy.float64'>
sprint_speed = <class 'numpy.float64'>
agility = <class 'numpy.float64'>
reactions = <class 'numpy.float64'>
balance = <class 'numpy.float64'>
shot_power = <class 'numpy.float64'>
jumping = <class 'numpy.float64'>
stamina = <class 'numpy.float64'>
strength = <class 'numpy.float64'>
long_shots = <class 'numpy.float64'>
aggression = <class 'numpy.float64'>
interceptions = <class 'numpy.float64'>
positioning = <class 'numpy.float64'>
vision = <class 'numpy.float64'>
penalties = <class 'numpy.float64'>
marking = <class 'numpy.float64'>
standing_tackle = <class 'numpy.float64'>
sliding_tackle = <class 'numpy.float64'>
gk_diving = <class 'numpy.float64'>
gk_handling = <class 'numpy.float64'>
gk_kicking = <class 'numpy.float64'>
gk_positioning = <class 'numpy.float64'>
gk_reflexes = <class 'numpy.float64'>
```

In [13]:

```
#Correlation below 0.4 is said to be a weak correlation
#
https://www.researchgate.net/post/What_is_the_minimum_value_of_correlation_coefficient_to_prove_the
stence_of_the_accepted_relationship_between_scores_of_two_of_more_tests
for i in list[:]:
    if(df['overall_rating'].corr(df[i])<0.4):
        list.remove(i)
```

In [14]:

```
for i in list:
    print(i,"=",type(df[i][0]))
```

```
potential = <class 'numpy.float64'>
short_passing = <class 'numpy.float64'>
long_passing = <class 'numpy.float64'>
ball_control = <class 'numpy.float64'>
reactions = <class 'numpy.float64'>
shot_power = <class 'numpy.float64'>
vision = <class 'numpy.float64'>
```

In [15]:

```
#Storing the correlation values in a list m=either_foot ml=left_foot mr=right_foot
m=[]
ml=[]
mr=[]
```

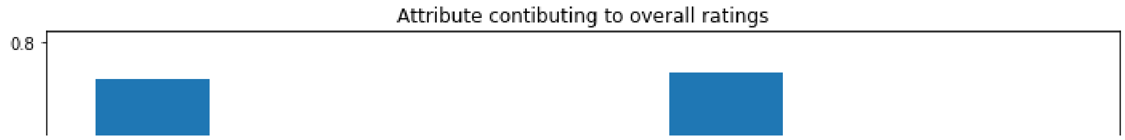
In [16]:

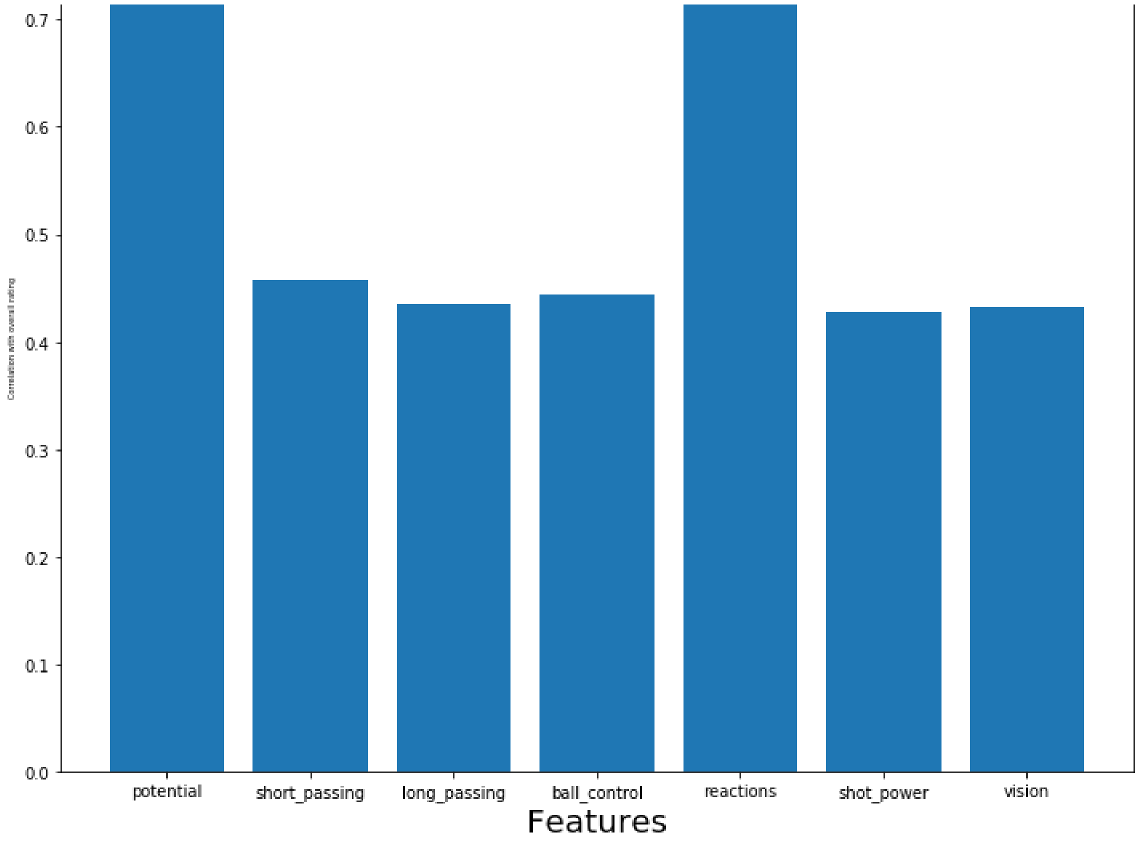
```
for i in list:
    m.append(df['overall_rating'].corr(df[i]))
    ml.append(dfl['overall_rating'].corr(dfl[i]))
    mr.append(dfr['overall_rating'].corr(dfr[i]))
    print("overall rating vs ",i,"=",df['overall_rating'].corr(df[i]))
    print("overall rating vs ",i,"=",dfl['overall_rating'].corr(dfl[i]))
    print("overall rating vs ",i,"=",dfr['overall_rating'].corr(dfr[i]))
```

```
overall rating vs  potential = 0.7654346716757741
overall rating vs  potential = 0.7560145627826399
overall rating vs  potential = 0.7683723065747465
overall rating vs  short_passing = 0.458242655815442
overall rating vs  short_passing = 0.5015978313512415
overall rating vs  short_passing = 0.45034469804285243
overall rating vs  long_passing = 0.43452499155719804
overall rating vs  long_passing = 0.4786613912999902
overall rating vs  long_passing = 0.425847701148177
overall rating vs  ball_control = 0.443990762826299
overall rating vs  ball_control = 0.48181675284093783
overall rating vs  ball_control = 0.43719059432459473
overall rating vs  reactions = 0.7718560966627348
overall rating vs  reactions = 0.7648099783573223
overall rating vs  reactions = 0.7739027487795825
overall rating vs  shot_power = 0.4280531322219387
overall rating vs  shot_power = 0.4510054228439738
overall rating vs  shot_power = 0.4226062991564441
overall rating vs  vision = 0.43149329504794093
overall rating vs  vision = 0.44404485344262706
overall rating vs  vision = 0.4286229000307069
```

In [17]:

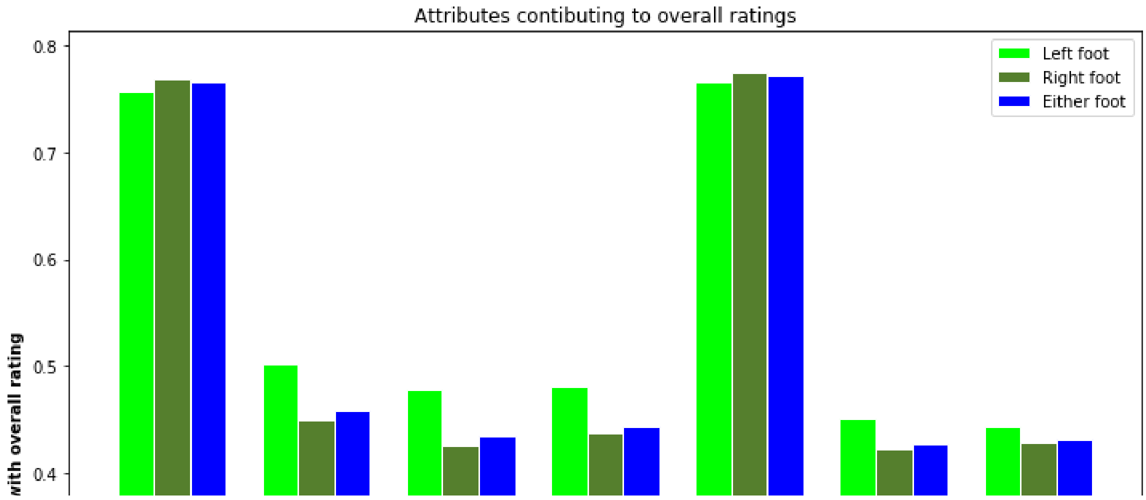
```
#https://stackoverflow.com/questions/332289
#/how-do-you-change-the-size-of-figures-drawn-with-matplotlib
from pylab import rcParams
rcParams['figure.figsize'] = 12, 10
index = np.arange(len(list))
plt.bar(index, m)
plt.xlabel('Features', fontsize=20)
plt.ylabel('Correlation with overall rating', fontsize=5)
plt.xticks(index, list, fontsize=10)
plt.title('Attribute contibuting to overall ratings')
plt.show()
```

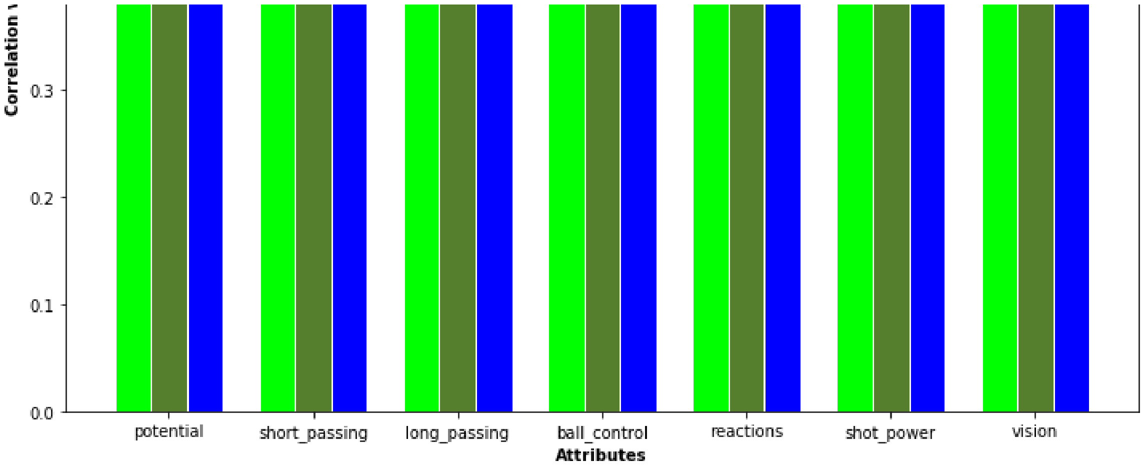




In [18]:

```
# https://python-graph-gallery.com/11-grouped-barplot/
# set width of bar
barWidth = 0.25

# Set position of bar on X axis
r1 = np.arange(len(ml))
r2 = [x + barWidth for x in r1]
r3 = [x + barWidth for x in r2]

# Make the plot
plt.bar(r1, ml, color='#00FF00', width=barWidth, edgecolor='white', label='Left foot')
plt.bar(r2, mr, color='#557f2d', width=barWidth, edgecolor='white', label='Right foot')
plt.bar(r3, m, color='#0000ff', width=barWidth, edgecolor='white', label='Either foot')

# Add xticks on the middle of the group bars
plt.xlabel('Attributes', fontweight='bold')
plt.ylabel('Correlation with overall rating', fontweight='bold')
plt.xticks([r + barWidth for r in range(len(m))], list)
plt.title('Attributes contibuting to overall ratings')

# Create legend & Show graphic
plt.legend()
plt.show()
```




Correlation
0.3
0.2
0.1
0.0
potential
short_passing
long_passing
ball_control
reactions
shot_power
vision
Attributes