Final Report

Divya Patel: 801079381
Namra Desai: 801084948

Python Code

```
#!/usr/bin/env python

# coding: utf-8


# In[90]:



import pandas as pd

import numpy as np

d = pd.read_csv('data.csv')

d.head()



# In[91]:


d


# In[92]:



list(d.loc[d['Name']=='L. Messi']['Dribbling'])[0]



# In[93]:



d.columns
```

```
# In[94]:


import pandas as pd
import numpy as np

from bokeh.io import show, curdoc, output_file
from bokeh.plotting import figure,output_notebook

from bokeh.models import CategoricalColorMapper, HoverTool, ColumnDataSource, Panel ,
FactorRange,Legend, LegendItem ,LabelSet
from bokeh.models.widgets import CheckboxGroup, Slider, RangeSlider, Tabs , Select

from bokeh.layouts import column, row, WidgetBox
from bokeh.palettes import Category20_16

from bokeh.application.handlers import FunctionHandler
from bokeh.application import Application


#output_file("visualization.html")
output_notebook()


df = pd.read_csv('data.csv')

def modify_doc(doc):
    def make_dataset(play1,play2):
```

```
    f1 = np.array([list(df.loc[df['Name']== play1]['Overall'])[0],list(df.loc[df['Name']== play1]['Potential'])[0],
list(df.loc[df['Name']==play1]['Finishing'])[0], list(df.loc[df['Name']==play1]['Dribbling'])[0],
list(df.loc[df['Name']==play1]['Stamina'])[0], list(df.loc[df['Name']== play1]['Strength'])[0],
list(df.loc[df['Name']==play1]['Vision'])[0],list(df.loc[df['Name']==play1]['Composure'])[0]])

    f2 = np.array([list(df.loc[df['Name']== play2]['Overall'])[0],list(df.loc[df['Name']== play2]['Potential'])[0],
list(df.loc[df['Name']==play2]['Finishing'])[0], list(df.loc[df['Name']==play2]['Dribbling'])[0],
list(df.loc[df['Name']==play2]['Stamina'])[0], list(df.loc[df['Name']==play2]['Strength'])[0],
list(df.loc[df['Name']==play2]['Vision'])[0],list(df.loc[df['Name']==play2]['Composure'])[0]])

    for i in range(len(f1)):
        f1[i] = f1[i]/100
        f2[i] = f2[i]/100
    flist = {'player1':f1,'player2':f2}
    return ColumnDataSource(flist)



def make_plot(src):

    num_vars = 8

    centre = 0.5

    theta = np.linspace(0, 2*np.pi, num_vars, endpoint=False)

    theta += np.pi/2

    def unit_poly_verts(theta, centre ):
        x0, y0, r = [centre ] * 3
        verts = [(r*np.cos(t) + x0, r*np.sin(t) + y0) for t in theta]
        return verts

    def radar_patch(r, theta, centre ):
        offset = 0.01
```

```
    yt = (r*centre + offset) * np.sin(theta) + centre
    xt = (r*centre + offset) * np.cos(theta) + centre
    return xt, yt

verts = unit_poly_verts(theta, centre)
x = [v[0] for v in verts]
y = [v[1] for v in verts]

p = figure(title="Baseline - Radar plot")
text = ['Overall', 'Potential', 'Finishing', 'Dribbling', 'Stamina', 'Strength', 'Vision', 'Composure','']
source = ColumnDataSource({'x':x + [centre ],'y':y + [1],'text':text})

p.line(x="x", y="y",source=source)

labels = LabelSet(x="x",y="y",text="text",source=source)



p.add_layout(labels)

print('-------------------------------------------------')
print(src.data)

#xt = np.array(x)
flist = []
play = []
for k,v in src.data.items():
    play.append(k)
    flist.append(v)

colors = ['blue','green']
```

```
    for i in range(len(flist)):
        xt, yt = radar_patch(flist[i], theta, centre)
        p.patch(x=xt, y=yt, fill_alpha=0.15, fill_color=colors[i],legend= play[i])
    return p




# Update the plot based on selections
def update(attr, old, new):
    new_src = make_dataset(select_1.value,select_2.value)
    print('1')
    src.data.update(new_src.data)
    p = make_plot(src)

# Put controls in a single element
    controls = WidgetBox(select_1,select_2)

# Create a row layout
    layout = row(controls, p)

# Make a tab with the layout
    tab = Panel(child=layout, title = 'Project')
    tabs = Tabs(tabs=[tab])


    doc.add_root(tabs)
```

```
select_1 = Select(title="Player 1:", value="L. Messi", options=list(df['Name']))
select_1.on_change('value',update)


select_2 = Select(title="Player 2:", value="De Gea", options=list(df['Name']))
select_2.on_change('value',update)



print(select_1.value,select_2.value)
src = make_dataset(
                select_1.value,
                select_2.value
                   )
print(src.data)

p = make_plot(src)

# Put controls in a single element
controls = WidgetBox(select_1,select_2)

# Create a row layout
layout = row(controls, p)

# Make a tab with the layout
tab = Panel(child=layout, title = 'Project')
tabs = Tabs(tabs=[tab])
```

```
    doc.add_root(tabs)

# Set up an application
handler = FunctionHandler(modify_doc)
app = Application(handler)



# In[95]:



show(app)



# In[96]:



import geopandas as gpd
shapefile = 'ne_110m_admin_0_countries.shp'
#Read shapefile using Geopandas
gdf = gpd.read_file(shapefile)[['ADMIN', 'ADM0_A3', 'geometry']]
#Rename columns.
gdf.columns = ['country', 'country_code', 'geometry']
gdf.head()



# In[97]:



from bokeh.io import curdoc, output_notebook,show,output_file
from bokeh.models import Slider, HoverTool,CheckboxGroup,Tabs,Panel
```

```
from bokeh.layouts import widgetbox, row, column
from bokeh.models import GeoJSONDataSource, LinearColorMapper, ColorBar
from bokeh.palettes import brewer
from bokeh.plotting import figure
import json
from bokeh.application.handlers import FunctionHandler
from bokeh.application import Application
#Define function that returns json_data for year selected by user.
data = pd.read_csv('data.csv')
data = data[~ data['Position'].isna()]
#print(data.head())
country_df = pd.read_csv('contry.csv')
import geopandas as gpd
shapefile = 'ne_110m_admin_0_countries.shp'
#Read shapefile using Geopandas
gdf = gpd.read_file(shapefile)[['ADMIN', 'ADM0_A3', 'geometry']]
#Rename columns.
gdf.columns = ['country', 'country_code', 'geometry']
output_notebook()


def modify_doc(doc):
    def json_data(pos,slider1,slider2):
#         print(pos,slider1,slider2)
        x_df = data[(data['Position'].isin(pos)) & (data['Overall'] > slider1) & (data['Age'] > slider2)]
        x_df = pd.DataFrame((x_df['Nationality'].value_counts()/x_df['Nationality'].count())*100).reset_index()
        x_df.columns = ['Country','Count']
#         print(x_df)
        geo_country = x_df.merge(country_df,left_on = 'Country', right_on = 'COUNTRY')
#         print(geo_country.head())
        merged = gdf.merge(geo_country, left_on = 'country_code', right_on = 'A3 (UN)')
```

```
        #print(merged.shape)
        merged_json = json.loads(merged.to_json())
        json_data = json.dumps(merged_json)
        return json_data


    geosource = GeoJSONDataSource(geojson = json_data(['ST'],80,20))
    #Define a sequential multi-hue color palette.
#     print(geosource)
    palette = brewer['YlGnBu'][8]
    #Reverse color order so that dark blue is highest obesity.
    palette = palette[::-1]
    #Instantiate LinearColorMapper that linearly maps numbers in a range, into a sequence of colors. Input
nan_color.
    color_mapper = LinearColorMapper(palette = palette, low = 0, high = 8, nan_color = '#d9d9d9')
    #Define custom tick labels for color bar.
    tick_labels = {'0': '0%', '1': '1%', '2':'2%', '3':'3%', '4':'4%', '5':'5%', '6':'6%','7':'7%', '8': '>8%'}
    #Add hover tool
    hover = HoverTool(tooltips = [ ('Country','@country'),('count', '@Count')])
    #Create color bar.
    color_bar = ColorBar(color_mapper=color_mapper, label_standoff=8,width = 500, height = 20,
                         border_line_color=None,location = (0,0), orientation = 'horizontal', major_label_overrides =
tick_labels)
    #Create figure object.
    p = figure(title = 'Map', plot_height = 600 , plot_width = 950, toolbar_location = None, tools = [hover])
    p.xgrid.grid_line_color = None
    p.ygrid.grid_line_color = None
    #Add patch renderer to figure.
    p.patches('xs','ys', source = geosource,fill_color = {'field' :'Count', 'transform' : color_mapper},
              line_color = 'black', line_width = 0.25, fill_alpha = 1)
    #Specify layout
    p.add_layout(color_bar, 'below')
```

```
    # Define the callback function: update_plot
    def update_plot(attr, old, new):
        selected_pos = [pos.labels[i] for i in pos.active]
        rating = slider1.value
        age = slider2.value
        new_data = json_data(selected_pos,rating,age)
        geosource.geojson = new_data

    pos = CheckboxGroup(labels=list(set(data['Position'])), active = [0,1])
    pos.on_change('active', update_plot)

    # Make a slider object: slider
    slider1 = Slider(title = 'Ranking',start = 0, end = 100, step = 1, value = 80)
    slider1.on_change('value', update_plot)

    slider2 = Slider(title = 'Age',start = 0, end = 50, step = 1, value = 20)
    slider2.on_change('value', update_plot)
    # Make a column layout of widgetbox(slider) and plot, and add it to the current document
    layout = row(widgetbox(pos,slider1,slider2),p)
    tab = Panel(child=layout, title = 'Project')
    tabs = Tabs(tabs=[tab])

    doc.add_root(tabs)
    #Display plot inline in Jupyter notebook

    #Display plot
handler = FunctionHandler(modify_doc)
d2 = Application(handler)

# In[98]:
```

show(d2)

Abstract:

This website is made in purpose of comparing the stats of two team. It is not feasible to compare any teams without any type of visual interface. This type of the comparison visualization is a key to many problems. Some of them are, finding a good team in a particular attribute only, Finding the weak attribute of the teams and how good they are in that attribute when comparing to other teams etc.

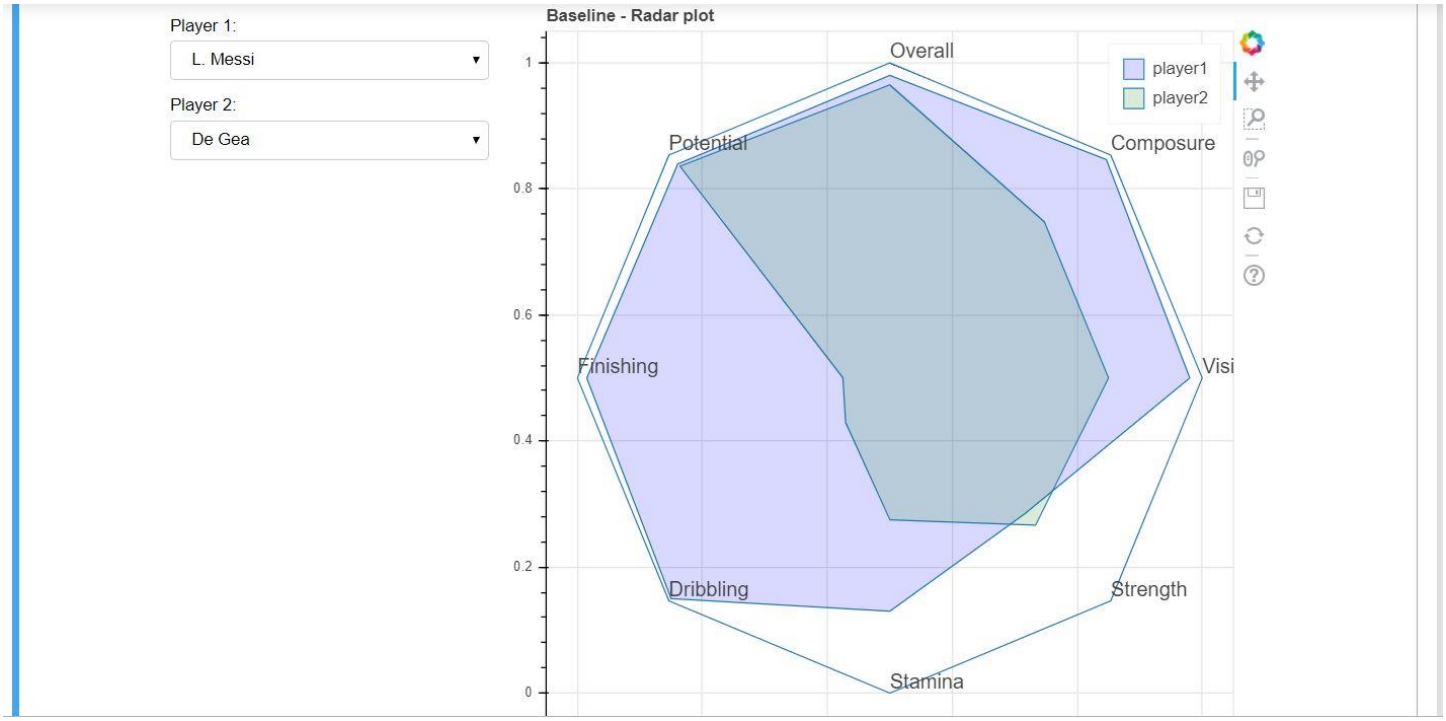
Data:

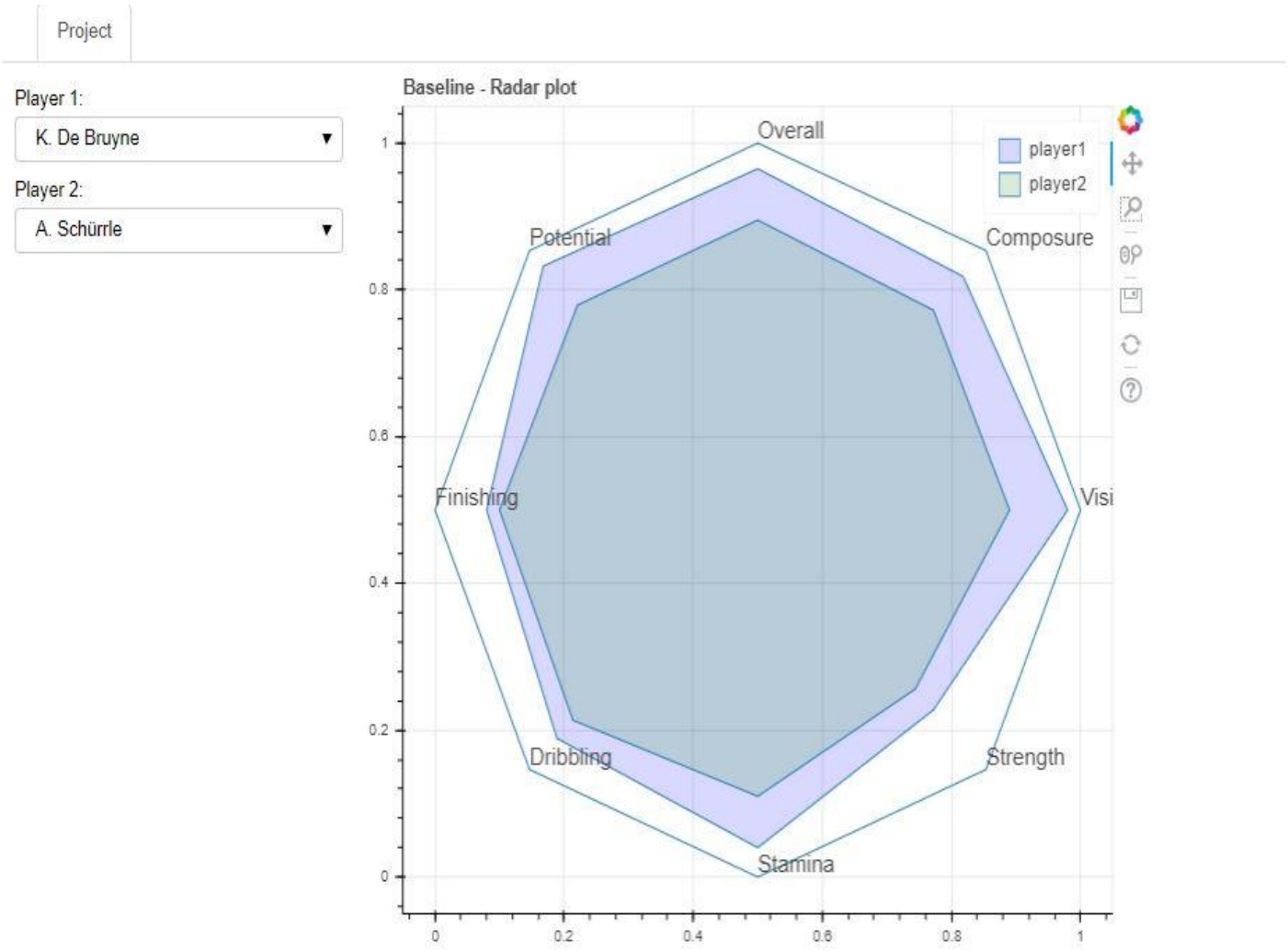
We are going to use https://www.kaggle.com/zaeemnalla/premier-league#stats.csv this data. It contains all 20 teams' stats from 2006-07 season to 2016-2017 season. It has 42 different attributes using which we can compare these teams. Some of the attributes are total tackles, Goals, Clearance, Touches etc. Even though data needs some preprocessing, it is a very well put together data to compare the stats of last 10 season and decide how well every team is doing.

Users:

This interactive visualization is useful for anyone who is interested football (Soccer) and want to grasp the performance of past 10 year in minimum time. Fans of the football team can see their favorite team's performance and progress over the decades. Coaches of the teams can figure out the weak link of the team and can try to improve it in the next season. This visualization can also be answer to many debates going around.

Features:





As you can see in the picture, Changing the players change the radio chart which shows the comparison of the skill level of players.

In the picture below, filtering the playing position, age, ranking change the color steps of the geographical map. This color steps show the strength of players in each country on the given filter.

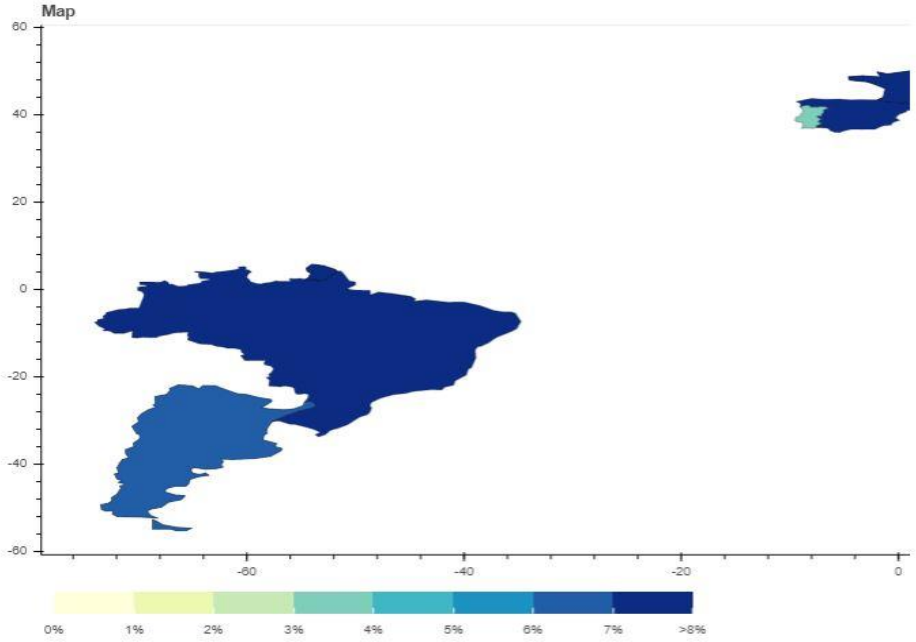
Project

- [x] LAM
- [x] LW
- [ ] RAM
- [ ] RDM
- [ ] LF
- [ ] RW
- [ ] GK
- [ ] RS
- [ ] LCM
- [ ] LWB
- [ ] LCB
- [ ] CM
- [ ] CDM
- [ ] RWB
- [ ] RCB
- [ ] ST
- [ ] CF
- [ ] RB
- [ ] LB
- [ ] RF
- [ ] LS
- [ ] RM
- [ ] CB
- [ ] LDM
- [ ] LM
- [ ] CAM
- [ ] RCM

Ranking: **80**

Age: **20**

Map



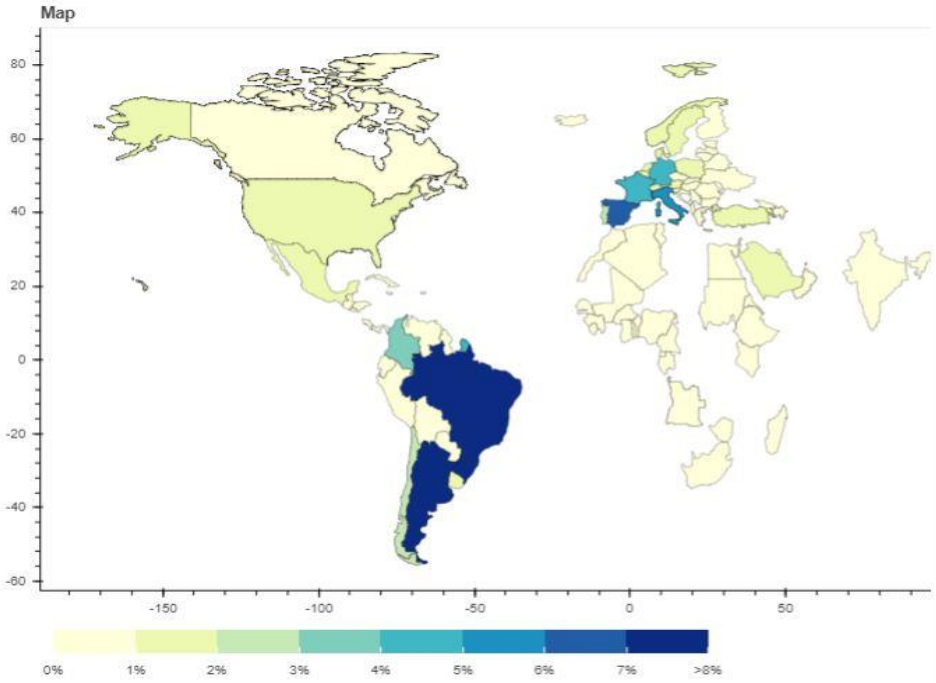
Project

- [x] LAM
- [x] LW
- [x] RAM
- [x] RDM
- [x] LF
- [x] RW
- [x] GK
- [x] RS
- [x] LCM
- [x] LWB
- [x] LCB
- [x] CM
- [x] CDM
- [x] RWB
- [x] RCB
- [x] ST
- [x] CF
- [x] RB
- [x] LB
- [x] RF
- [x] LS
- [x] RM
- [x] CB
- [x] LDM
- [x] LM
- [x] CAM
- [x] RCM

Ranking: **51**

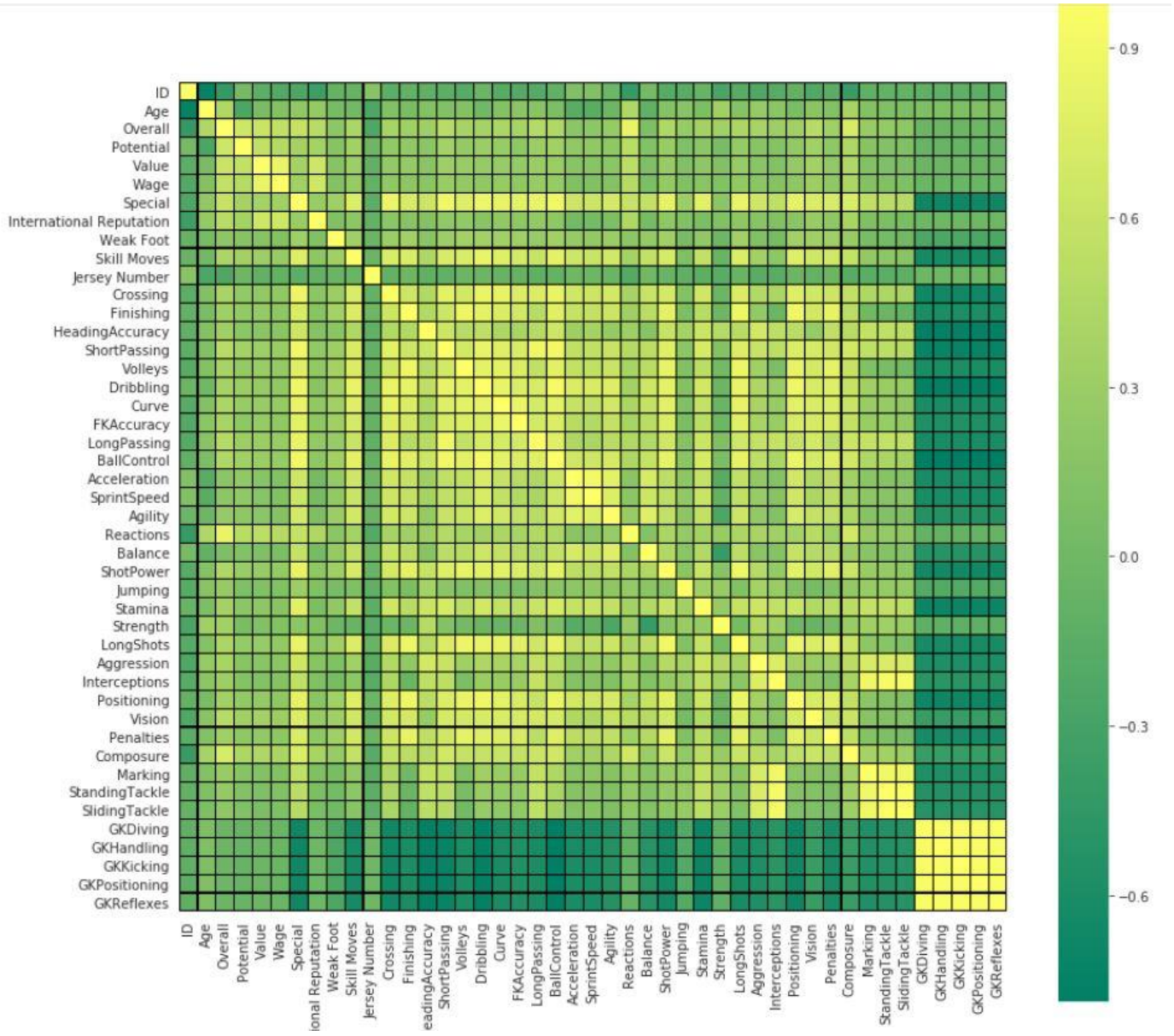
Age: **30**

Map



Future Work and Conclusion:

As you can see in the picture below, dimension of our data is very huge so in future we would try to reduce our data using PCA or combining similar kind of attribute and considering it as a single attribute and can try to make more visualization on it. We can also use machine learning algorithm to make some good predictive visualization. We can use K-mean clustering and try to make a cluster of similar type of players.



So, we learned how to use Bokeh for visualization and how to make choropleth map with user interaction. Moreover, we learned about data reduction and we will try to implement it in future.

Team Contribution:

I worked on data preprocessing and report.

Divya worked on making a visualization using Bokeh and tried to find best interactive visualization for this data.