Which Factor Has a Greater Impact on Happiness Scores?

Introduction

With the progress of the times, every country in the world is making corresponding efforts to achieve progress. To assess the progress of the country, some experts use people's happiness scores to analyze and evaluate, and some governments will make targeted adjustments to their politics based on happiness scores. According to the happiness level of citizens, scholars who compiled the World Happiness Report 2019 found that the living environment (including social environment and political system) of each country is an important source of happiness. The huge international differences in people's life evaluation are caused by the differences in people's contact ways between them and their common institutions and social norms. Moreover, people can further explore how changes in technology, social norms, conflicts and government policies affect people's happiness through happiness scores. The purpose of this work is to analyze those factors that have a greater impact on happiness scores to give corresponding adjustment programs.

We import `download_plotlyjs` , `init_notebook_mode` , `plot` , and `iplot` from **plotly.offline**, to visualize the happiness scores and the independent variables of various countries on the world map.

In [1]:
```
import time
start_time = time.time()
```

In [2]:
```
import pandas as pd
pd.set_option('display.max_rows', 200)
from sklearn.cluster import DBSCAN
from matplotlib import pyplot
import matplotlib.pyplot as plt
import numpy as np
from plotly.offline import download_plotlyjs,init_notebook_mode,plot,iplot
init_notebook_mode(connected=True)
import plotly.graph_objs as go
from sklearn import metrics

from sklearn.linear_model import LinearRegression
pd.options.display.float_format = '{:40,.4f}'.format # specifies default number form
plt.style.use('ggplot') # specifies that graphs should use ggplot styling
%matplotlib inline
```

Literature Review

Julie (2012) used a multiple linear regression model to simulate and analyze what is the most important factor affecting national happiness. So we also use multiple linear regression models

Presentation of Data and Data preprocessing

In this work, we use the World Happiness Report data in kaggle, and mainly analyze the 2019 data. In this data, the happiness scores of each country are recorded, and they are ranked according to their happiness scores. Not only that, but also recorded six factors that lead to happiness, these six factors are: GDP per capita, Social support, Healthy life expectancy, Freedom to make life choices, Generosity and Perceptions of corruption. Therefore, we choose the

happiness score (Score) as the dependent variable, and the six factors are six independent variables.

In [3]:
```
data = pd.read_csv('2019.csv')
data2=pd.read_csv('2015.csv')
data.head()
```

Out[3]:

	Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
0	1	Finland	7.7690	1.3400	1.5870	0.9860	0.5960	0.1530	0.3930
1	2	Denmark	7.6000	1.3830	1.5730	0.9960	0.5920	0.2520	0.4100
2	3	Norway	7.5540	1.4880	1.5820	1.0280	0.6030	0.2710	0.3410
3	4	Iceland	7.4940	1.3800	1.6240	1.0260	0.5910	0.3540	0.1180
4	5	Netherlands	7.4880	1.3960	1.5220	0.9990	0.5570	0.3220	0.2980

For convenience, we use the `.rename()` function to simplify complex column names.

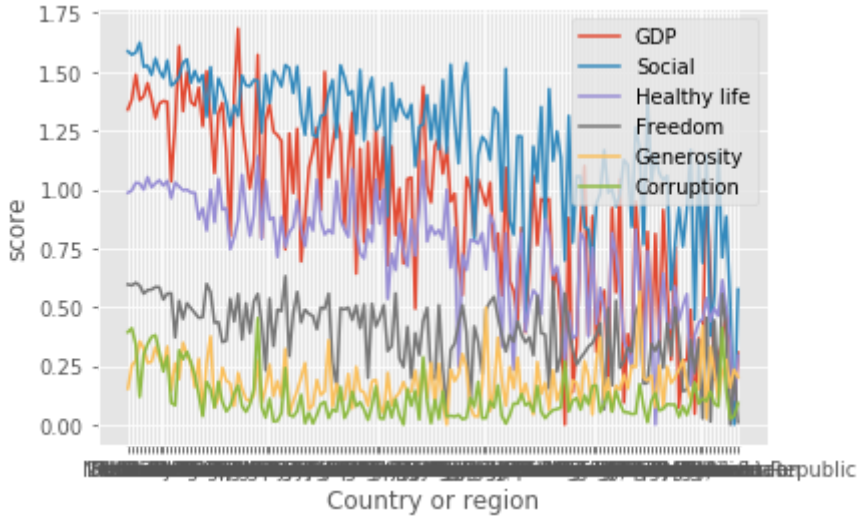
In [4]:
```
data = data.rename(columns={'GDP per capita': 'GDP','Social support':'Social','Healt
```

Then we use a line chart to visualize all variables.

In [5]:
```
x=data['Country or region']
y1=data['GDP']
y2=data['Social']
y3=data['Life']
y4=data['Freedom']
y5=data['Generosity']
y6=data['Corruption']
Y=data['Score']
```

In [6]:
```
plt.plot(x, y1,  label='GDP')
plt.plot(x, y2,label='Social')
plt.plot(x, y3,label='Healthy life')
plt.plot(x, y4,label='Freedom')
plt.plot(x, y5,label='Generosity')
plt.plot(x, y6,label='Corruption')
plt.legend()
plt.xlabel('Country or region')
plt.ylabel("score")
```

Out[6]:
```
Text(0, 0.5, 'score')
```



The effect of this picture is not very good, especially chaotic.

We use `drop()` to delete the **"Overall rank"** column in the data, because this variable is useless.

In [7]:
```
data = data.drop(['Overall rank'], axis=1)
data.head()
```

Out[7]:

	Country or region	Score	GDP	Social	Life	Freedom	Generosity	Corruption
0	Finland	7.7690	1.3400	1.5870	0.9860	0.5960	0.1530	0.3930
1	Denmark	7.6000	1.3830	1.5730	0.9960	0.5920	0.2520	0.4100
2	Norway	7.5540	1.4880	1.5820	1.0280	0.6030	0.2710	0.3410
3	Iceland	7.4940	1.3800	1.6240	1.0260	0.5910	0.3540	0.1180
4	Netherlands	7.4880	1.3960	1.5220	0.9990	0.5570	0.3220	0.2980

Then we use the `info()` function to check the missing value and dtpye of each column of the data.

In [8]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 156 entries, 0 to 155
Data columns (total 8 columns):
Country or region    156 non-null object
Score                156 non-null float64
GDP                  156 non-null float64
Social               156 non-null float64
Life                 156 non-null float64
Freedom              156 non-null float64
Generosity           156 non-null float64
Corruption           156 non-null float64
dtypes: float64(7), object(1)
memory usage: 9.9+ KB
```

Using the `.count()` function to calculate the count of **non-null** values for each column.

In [9]:
```
data.count()
```

Out[9]:
```
Country or region    156
Score                156
GDP                  156
Social               156
Life                 156
```

```
Freedom              156
Generosity           156
Corruption           156
dtype: int64
```

Using the . `isnull().sum()` function to count the number of empty columns.

In [10]:
```
data.isnull().sum()
```

Out[10]:
```
Country or region    0
Score                0
GDP                  0
Social               0
Life                 0
Freedom              0
Generosity           0
Corruption           0
dtype: int64
```

We use the `.describe()` function to check to understand some basic information about these data.

In [11]:
```
data['Country or region'].describe()
```

Out[11]:
```
count         156
unique        156
top       Estonia
freq            1
Name: Country or region, dtype: object
```

In [12]:
```
data['Score'].describe()
```

Out[12]:
```
count                                   156.0000
mean                                      5.4071
std                                       1.1131
min                                       2.8530
25%                                       4.5445
50%                                       5.3795
75%                                       6.1845
max                                       7.7690
Name: Score, dtype: float64
```

In [13]:
```
data['GDP'].describe()
```

Out[13]:
```
count                                   156.0000
mean                                      0.9051
std                                       0.3984
min                                       0.0000
25%                                       0.6028
50%                                       0.9600
75%                                       1.2325
max                                       1.6840
Name: GDP, dtype: float64
```

In [14]:
```
data['Social'].describe()
```

Out[14]:
```
count                                   156.0000
mean                                      1.2088
std                                       0.2992
min                                       0.0000
25%                                       1.0557
50%                                       1.2715
75%                                       1.4525
max                                       1.6240
Name: Social, dtype: float64
```

In [15]:
```
data['Life'].describe()
```

Out[15]:
```
count                                156.0000
mean                                   0.7252
std                                    0.2421
min                                    0.0000
25%                                    0.5477
50%                                    0.7890
75%                                    0.8817
max                                    1.1410
Name: Life, dtype: float64
```

In [16]:
```
data['Freedom'].describe()
```

Out[16]:
```
count                                156.0000
mean                                   0.3926
std                                    0.1433
min                                    0.0000
25%                                    0.3080
50%                                    0.4170
75%                                    0.5072
max                                    0.6310
Name: Freedom, dtype: float64
```

In [17]:
```
data['Generosity'].describe()
```

Out[17]:
```
count                                156.0000
mean                                   0.1848
std                                    0.0953
min                                    0.0000
25%                                    0.1088
50%                                    0.1775
75%                                    0.2482
max                                    0.5660
Name: Generosity, dtype: float64
```

In [18]:
```
data['Corruption'].describe()
```

Out[18]:
```
count                                156.0000
mean                                   0.1106
std                                    0.0945
min                                    0.0000
25%                                    0.0470
50%                                    0.0855
75%                                    0.1412
max                                    0.4530
Name: Corruption, dtype: float64
```

We use the `.dtypes` function to confirm the data type of each column of data

In [19]:
```
data.dtypes
```

Out[19]:
```
Country or region     object
Score                float64
GDP                  float64
Social               float64
Life                 float64
Freedom              float64
Generosity           float64
Corruption           float64
dtype: object
```

We can see that the format of the **"Country or region"** column is an object, and the data format of the remaining columns are float. This shows that we do not need to change the data format of the independent and dependent variables.

So we introduce the variable of the continent from the **'2015.csv'** table, name this column **'Continent'**, and then combine it with the 2019 data by using `.merge()` function.

In [20]:
```
df=data2[{'Region','Country'}]
df
```

Out[20]:

	Country	Region
0	Switzerland	Western Europe
1	Iceland	Western Europe
2	Denmark	Western Europe
3	Norway	Western Europe
4	Canada	North America
5	Finland	Western Europe
6	Netherlands	Western Europe
7	Sweden	Western Europe
8	New Zealand	Australia and New Zealand
9	Australia	Australia and New Zealand
10	Israel	Middle East and Northern Africa
11	Costa Rica	Latin America and Caribbean
12	Austria	Western Europe
13	Mexico	Latin America and Caribbean
14	United States	North America
15	Brazil	Latin America and Caribbean
16	Luxembourg	Western Europe
17	Ireland	Western Europe
18	Belgium	Western Europe
19	United Arab Emirates	Middle East and Northern Africa
20	United Kingdom	Western Europe
21	Oman	Middle East and Northern Africa
22	Venezuela	Latin America and Caribbean
23	Singapore	Southeastern Asia
24	Panama	Latin America and Caribbean
25	Germany	Western Europe
26	Chile	Latin America and Caribbean
27	Qatar	Middle East and Northern Africa
28	France	Western Europe
29	Argentina	Latin America and Caribbean
30	Czech Republic	Central and Eastern Europe
31	Uruguay	Latin America and Caribbean
32	Colombia	Latin America and Caribbean
33	Thailand	Southeastern Asia

	Country	Region
34	Saudi Arabia	Middle East and Northern Africa
35	Spain	Western Europe
36	Malta	Western Europe
37	Taiwan	Eastern Asia
38	Kuwait	Middle East and Northern Africa
39	Suriname	Latin America and Caribbean
40	Trinidad and Tobago	Latin America and Caribbean
41	El Salvador	Latin America and Caribbean
42	Guatemala	Latin America and Caribbean
43	Uzbekistan	Central and Eastern Europe
44	Slovakia	Central and Eastern Europe
45	Japan	Eastern Asia
46	South Korea	Eastern Asia
47	Ecuador	Latin America and Caribbean
48	Bahrain	Middle East and Northern Africa
49	Italy	Western Europe
50	Bolivia	Latin America and Caribbean
51	Moldova	Central and Eastern Europe
52	Paraguay	Latin America and Caribbean
53	Kazakhstan	Central and Eastern Europe
54	Slovenia	Central and Eastern Europe
55	Lithuania	Central and Eastern Europe
56	Nicaragua	Latin America and Caribbean
57	Peru	Latin America and Caribbean
58	Belarus	Central and Eastern Europe
59	Poland	Central and Eastern Europe
60	Malaysia	Southeastern Asia
61	Croatia	Central and Eastern Europe
62	Libya	Middle East and Northern Africa
63	Russia	Central and Eastern Europe
64	Jamaica	Latin America and Caribbean
65	North Cyprus	Western Europe
66	Cyprus	Western Europe
67	Algeria	Middle East and Northern Africa
68	Kosovo	Central and Eastern Europe
69	Turkmenistan	Central and Eastern Europe

	Country	Region
70	Mauritius	Sub-Saharan Africa
71	Hong Kong	Eastern Asia
72	Estonia	Central and Eastern Europe
73	Indonesia	Southeastern Asia
74	Vietnam	Southeastern Asia
75	Turkey	Middle East and Northern Africa
76	Kyrgyzstan	Central and Eastern Europe
77	Nigeria	Sub-Saharan Africa
78	Bhutan	Southern Asia
79	Azerbaijan	Central and Eastern Europe
80	Pakistan	Southern Asia
81	Jordan	Middle East and Northern Africa
82	Montenegro	Central and Eastern Europe
83	China	Eastern Asia
84	Zambia	Sub-Saharan Africa
85	Romania	Central and Eastern Europe
86	Serbia	Central and Eastern Europe
87	Portugal	Western Europe
88	Latvia	Central and Eastern Europe
89	Philippines	Southeastern Asia
90	Somaliland region	Sub-Saharan Africa
91	Morocco	Middle East and Northern Africa
92	Macedonia	Central and Eastern Europe
93	Mozambique	Sub-Saharan Africa
94	Albania	Central and Eastern Europe
95	Bosnia and Herzegovina	Central and Eastern Europe
96	Lesotho	Sub-Saharan Africa
97	Dominican Republic	Latin America and Caribbean
98	Laos	Southeastern Asia
99	Mongolia	Eastern Asia
100	Swaziland	Sub-Saharan Africa
101	Greece	Western Europe
102	Lebanon	Middle East and Northern Africa
103	Hungary	Central and Eastern Europe
104	Honduras	Latin America and Caribbean
105	Tajikistan	Central and Eastern Europe

	Country	Region
106	Tunisia	Middle East and Northern Africa
107	Palestinian Territories	Middle East and Northern Africa
108	Bangladesh	Southern Asia
109	Iran	Middle East and Northern Africa
110	Ukraine	Central and Eastern Europe
111	Iraq	Middle East and Northern Africa
112	South Africa	Sub-Saharan Africa
113	Ghana	Sub-Saharan Africa
114	Zimbabwe	Sub-Saharan Africa
115	Liberia	Sub-Saharan Africa
116	India	Southern Asia
117	Sudan	Sub-Saharan Africa
118	Haiti	Latin America and Caribbean
119	Congo (Kinshasa)	Sub-Saharan Africa
120	Nepal	Southern Asia
121	Ethiopia	Sub-Saharan Africa
122	Sierra Leone	Sub-Saharan Africa
123	Mauritania	Sub-Saharan Africa
124	Kenya	Sub-Saharan Africa
125	Djibouti	Sub-Saharan Africa
126	Armenia	Central and Eastern Europe
127	Botswana	Sub-Saharan Africa
128	Myanmar	Southeastern Asia
129	Georgia	Central and Eastern Europe
130	Malawi	Sub-Saharan Africa
131	Sri Lanka	Southern Asia
132	Cameroon	Sub-Saharan Africa
133	Bulgaria	Central and Eastern Europe
134	Egypt	Middle East and Northern Africa
135	Yemen	Middle East and Northern Africa
136	Angola	Sub-Saharan Africa
137	Mali	Sub-Saharan Africa
138	Congo (Brazzaville)	Sub-Saharan Africa
139	Comoros	Sub-Saharan Africa
140	Uganda	Sub-Saharan Africa
141	Senegal	Sub-Saharan Africa

	Country	Region
142	Gabon	Sub-Saharan Africa
143	Niger	Sub-Saharan Africa
144	Cambodia	Southeastern Asia
145	Tanzania	Sub-Saharan Africa
146	Madagascar	Sub-Saharan Africa
147	Central African Republic	Sub-Saharan Africa
148	Chad	Sub-Saharan Africa
149	Guinea	Sub-Saharan Africa
150	Ivory Coast	Sub-Saharan Africa
151	Burkina Faso	Sub-Saharan Africa
152	Afghanistan	Southern Asia
153	Rwanda	Sub-Saharan Africa
154	Benin	Sub-Saharan Africa
155	Syria	Middle East and Northern Africa
156	Burundi	Sub-Saharan Africa
157	Togo	Sub-Saharan Africa

In [21]:
```
df=df.rename(columns = {'Region':'Continent','Country':'Country or region'})
df
```

Out[21]:

	Country or region	Continent
0	Switzerland	Western Europe
1	Iceland	Western Europe
2	Denmark	Western Europe
3	Norway	Western Europe
4	Canada	North America
5	Finland	Western Europe
6	Netherlands	Western Europe
7	Sweden	Western Europe
8	New Zealand	Australia and New Zealand
9	Australia	Australia and New Zealand
10	Israel	Middle East and Northern Africa
11	Costa Rica	Latin America and Caribbean
12	Austria	Western Europe
13	Mexico	Latin America and Caribbean
14	United States	North America
15	Brazil	Latin America and Caribbean

	Country or region	Continent
16	Luxembourg	Western Europe
17	Ireland	Western Europe
18	Belgium	Western Europe
19	United Arab Emirates	Middle East and Northern Africa
20	United Kingdom	Western Europe
21	Oman	Middle East and Northern Africa
22	Venezuela	Latin America and Caribbean
23	Singapore	Southeastern Asia
24	Panama	Latin America and Caribbean
25	Germany	Western Europe
26	Chile	Latin America and Caribbean
27	Qatar	Middle East and Northern Africa
28	France	Western Europe
29	Argentina	Latin America and Caribbean
30	Czech Republic	Central and Eastern Europe
31	Uruguay	Latin America and Caribbean
32	Colombia	Latin America and Caribbean
33	Thailand	Southeastern Asia
34	Saudi Arabia	Middle East and Northern Africa
35	Spain	Western Europe
36	Malta	Western Europe
37	Taiwan	Eastern Asia
38	Kuwait	Middle East and Northern Africa
39	Suriname	Latin America and Caribbean
40	Trinidad and Tobago	Latin America and Caribbean
41	El Salvador	Latin America and Caribbean
42	Guatemala	Latin America and Caribbean
43	Uzbekistan	Central and Eastern Europe
44	Slovakia	Central and Eastern Europe
45	Japan	Eastern Asia
46	South Korea	Eastern Asia
47	Ecuador	Latin America and Caribbean
48	Bahrain	Middle East and Northern Africa
49	Italy	Western Europe
50	Bolivia	Latin America and Caribbean
51	Moldova	Central and Eastern Europe

	Country or region	Continent
52	Paraguay	Latin America and Caribbean
53	Kazakhstan	Central and Eastern Europe
54	Slovenia	Central and Eastern Europe
55	Lithuania	Central and Eastern Europe
56	Nicaragua	Latin America and Caribbean
57	Peru	Latin America and Caribbean
58	Belarus	Central and Eastern Europe
59	Poland	Central and Eastern Europe
60	Malaysia	Southeastern Asia
61	Croatia	Central and Eastern Europe
62	Libya	Middle East and Northern Africa
63	Russia	Central and Eastern Europe
64	Jamaica	Latin America and Caribbean
65	North Cyprus	Western Europe
66	Cyprus	Western Europe
67	Algeria	Middle East and Northern Africa
68	Kosovo	Central and Eastern Europe
69	Turkmenistan	Central and Eastern Europe
70	Mauritius	Sub-Saharan Africa
71	Hong Kong	Eastern Asia
72	Estonia	Central and Eastern Europe
73	Indonesia	Southeastern Asia
74	Vietnam	Southeastern Asia
75	Turkey	Middle East and Northern Africa
76	Kyrgyzstan	Central and Eastern Europe
77	Nigeria	Sub-Saharan Africa
78	Bhutan	Southern Asia
79	Azerbaijan	Central and Eastern Europe
80	Pakistan	Southern Asia
81	Jordan	Middle East and Northern Africa
82	Montenegro	Central and Eastern Europe
83	China	Eastern Asia
84	Zambia	Sub-Saharan Africa
85	Romania	Central and Eastern Europe
86	Serbia	Central and Eastern Europe
87	Portugal	Western Europe

	Country or region	Continent
88	Latvia	Central and Eastern Europe
89	Philippines	Southeastern Asia
90	Somaliland region	Sub-Saharan Africa
91	Morocco	Middle East and Northern Africa
92	Macedonia	Central and Eastern Europe
93	Mozambique	Sub-Saharan Africa
94	Albania	Central and Eastern Europe
95	Bosnia and Herzegovina	Central and Eastern Europe
96	Lesotho	Sub-Saharan Africa
97	Dominican Republic	Latin America and Caribbean
98	Laos	Southeastern Asia
99	Mongolia	Eastern Asia
100	Swaziland	Sub-Saharan Africa
101	Greece	Western Europe
102	Lebanon	Middle East and Northern Africa
103	Hungary	Central and Eastern Europe
104	Honduras	Latin America and Caribbean
105	Tajikistan	Central and Eastern Europe
106	Tunisia	Middle East and Northern Africa
107	Palestinian Territories	Middle East and Northern Africa
108	Bangladesh	Southern Asia
109	Iran	Middle East and Northern Africa
110	Ukraine	Central and Eastern Europe
111	Iraq	Middle East and Northern Africa
112	South Africa	Sub-Saharan Africa
113	Ghana	Sub-Saharan Africa
114	Zimbabwe	Sub-Saharan Africa
115	Liberia	Sub-Saharan Africa
116	India	Southern Asia
117	Sudan	Sub-Saharan Africa
118	Haiti	Latin America and Caribbean
119	Congo (Kinshasa)	Sub-Saharan Africa
120	Nepal	Southern Asia
121	Ethiopia	Sub-Saharan Africa
122	Sierra Leone	Sub-Saharan Africa
123	Mauritania	Sub-Saharan Africa

	Country or region	Continent
124	Kenya	Sub-Saharan Africa
125	Djibouti	Sub-Saharan Africa
126	Armenia	Central and Eastern Europe
127	Botswana	Sub-Saharan Africa
128	Myanmar	Southeastern Asia
129	Georgia	Central and Eastern Europe
130	Malawi	Sub-Saharan Africa
131	Sri Lanka	Southern Asia
132	Cameroon	Sub-Saharan Africa
133	Bulgaria	Central and Eastern Europe
134	Egypt	Middle East and Northern Africa
135	Yemen	Middle East and Northern Africa
136	Angola	Sub-Saharan Africa
137	Mali	Sub-Saharan Africa
138	Congo (Brazzaville)	Sub-Saharan Africa
139	Comoros	Sub-Saharan Africa
140	Uganda	Sub-Saharan Africa
141	Senegal	Sub-Saharan Africa
142	Gabon	Sub-Saharan Africa
143	Niger	Sub-Saharan Africa
144	Cambodia	Southeastern Asia
145	Tanzania	Sub-Saharan Africa
146	Madagascar	Sub-Saharan Africa
147	Central African Republic	Sub-Saharan Africa
148	Chad	Sub-Saharan Africa
149	Guinea	Sub-Saharan Africa
150	Ivory Coast	Sub-Saharan Africa
151	Burkina Faso	Sub-Saharan Africa
152	Afghanistan	Southern Asia
153	Rwanda	Sub-Saharan Africa
154	Benin	Sub-Saharan Africa
155	Syria	Middle East and Northern Africa
156	Burundi	Sub-Saharan Africa
157	Togo	Sub-Saharan Africa

In [22]:
```
df2 = pd.merge(data, df, how='left', on=['Country or region'])
df2
```

Out[22]:

	Country or region	Score	GDP	Social	Life	Freedom	Generosity	Corruption	Continent
0	Finland	7.7690	1.3400	1.5870	0.9860	0.5960	0.1530	0.3930	Western Europe
1	Denmark	7.6000	1.3830	1.5730	0.9960	0.5920	0.2520	0.4100	Western Europe
2	Norway	7.5540	1.4880	1.5820	1.0280	0.6030	0.2710	0.3410	Western Europe
3	Iceland	7.4940	1.3800	1.6240	1.0260	0.5910	0.3540	0.1180	Western Europe
4	Netherlands	7.4880	1.3960	1.5220	0.9990	0.5570	0.3220	0.2980	Western Europe
5	Switzerland	7.4800	1.4520	1.5260	1.0520	0.5720	0.2630	0.3430	Western Europe
6	Sweden	7.3430	1.3870	1.4870	1.0090	0.5740	0.2670	0.3730	Western Europe
7	New Zealand	7.3070	1.3030	1.5570	1.0260	0.5850	0.3300	0.3800	Australia and New Zealand
8	Canada	7.2780	1.3650	1.5050	1.0390	0.5840	0.2850	0.3080	North America
9	Austria	7.2460	1.3760	1.4750	1.0160	0.5320	0.2440	0.2260	Western Europe
10	Australia	7.2280	1.3720	1.5480	1.0360	0.5570	0.3320	0.2900	Australia and New Zealand
11	Costa Rica	7.1670	1.0340	1.4410	0.9630	0.5580	0.1440	0.0930	Latin America and Caribbean
12	Israel	7.1390	1.2760	1.4550	1.0290	0.3710	0.2610	0.0820	Middle East and Northern Africa
13	Luxembourg	7.0900	1.6090	1.4790	1.0120	0.5260	0.1940	0.3160	Western Europe
14	United Kingdom	7.0540	1.3330	1.5380	0.9960	0.4500	0.3480	0.2780	Western Europe
15	Ireland	7.0210	1.4990	1.5530	0.9990	0.5160	0.2980	0.3100	Western Europe
16	Germany	6.9850	1.3730	1.4540	0.9870	0.4950	0.2610	0.2650	Western Europe
17	Belgium	6.9230	1.3560	1.5040	0.9860	0.4730	0.1600	0.2100	Western Europe
18	United States	6.8920	1.4330	1.4570	0.8740	0.4540	0.2800	0.1280	North America
19	Czech Republic	6.8520	1.2690	1.4870	0.9200	0.4570	0.0460	0.0360	Central and Eastern Europe

	Country or region	Score	GDP	Social	Life	Freedom	Generosity	Corruption	Continent
20	United Arab Emirates	6.8250	1.5030	1.3100	0.8250	0.5980	0.2620	0.1820	Middle East and Northern Africa
21	Malta	6.7260	1.3000	1.5200	0.9990	0.5640	0.3750	0.1510	Western Europe
22	Mexico	6.5950	1.0700	1.3230	0.8610	0.4330	0.0740	0.0730	Latin America and Caribbean
23	France	6.5920	1.3240	1.4720	1.0450	0.4360	0.1110	0.1830	Western Europe
24	Taiwan	6.4460	1.3680	1.4300	0.9140	0.3510	0.2420	0.0970	Eastern Asia
25	Chile	6.4440	1.1590	1.3690	0.9200	0.3570	0.1870	0.0560	Latin America and Caribbean
26	Guatemala	6.4360	0.8000	1.2690	0.7460	0.5350	0.1750	0.0780	Latin America and Caribbean
27	Saudi Arabia	6.3750	1.4030	1.3570	0.7950	0.4390	0.0800	0.1320	Middle East and Northern Africa
28	Qatar	6.3740	1.6840	1.3130	0.8710	0.5550	0.2200	0.1670	Middle East and Northern Africa
29	Spain	6.3540	1.2860	1.4840	1.0620	0.3620	0.1530	0.0790	Western Europe
30	Panama	6.3210	1.1490	1.4420	0.9100	0.5160	0.1090	0.0540	Latin America and Caribbean
31	Brazil	6.3000	1.0040	1.4390	0.8020	0.3900	0.0990	0.0860	Latin America and Caribbean
32	Uruguay	6.2930	1.1240	1.4650	0.8910	0.5230	0.1270	0.1500	Latin America and Caribbean
33	Singapore	6.2620	1.5720	1.4630	1.1410	0.5560	0.2710	0.4530	Southeastern Asia
34	El Salvador	6.2530	0.7940	1.2420	0.7890	0.4300	0.0930	0.0740	Latin America and Caribbean
35	Italy	6.2230	1.2940	1.4880	1.0390	0.2310	0.1580	0.0300	Western Europe
36	Bahrain	6.1990	1.3620	1.3680	0.8710	0.5360	0.2550	0.1100	Middle East and Northern Africa

	Country or region	Score	GDP	Social	Life	Freedom	Generosity	Corruption	Continent
37	Slovakia	6.1980	1.2460	1.5040	0.8810	0.3340	0.1210	0.0140	Central and Eastern Europe
38	Trinidad & Tobago	6.1920	1.2310	1.4770	0.7130	0.4890	0.1850	0.0160	NaN
39	Poland	6.1820	1.2060	1.4380	0.8840	0.4830	0.1170	0.0500	Central and Eastern Europe
40	Uzbekistan	6.1740	0.7450	1.5290	0.7560	0.6310	0.3220	0.2400	Central and Eastern Europe
41	Lithuania	6.1490	1.2380	1.5150	0.8180	0.2910	0.0430	0.0420	Central and Eastern Europe
42	Colombia	6.1250	0.9850	1.4100	0.8410	0.4700	0.0990	0.0340	Latin America and Caribbean
43	Slovenia	6.1180	1.2580	1.5230	0.9530	0.5640	0.1440	0.0570	Central and Eastern Europe
44	Nicaragua	6.1050	0.6940	1.3250	0.8350	0.4350	0.2000	0.1270	Latin America and Caribbean
45	Kosovo	6.1000	0.8820	1.2320	0.7580	0.4890	0.2620	0.0060	Central and Eastern Europe
46	Argentina	6.0860	1.0920	1.4320	0.8810	0.4710	0.0660	0.0500	Latin America and Caribbean
47	Romania	6.0700	1.1620	1.2320	0.8250	0.4620	0.0830	0.0050	Central and Eastern Europe
48	Cyprus	6.0460	1.2630	1.2230	1.0420	0.4060	0.1900	0.0410	Western Europe
49	Ecuador	6.0280	0.9120	1.3120	0.8680	0.4980	0.1260	0.0870	Latin America and Caribbean
50	Kuwait	6.0210	1.5000	1.3190	0.8080	0.4930	0.1420	0.0970	Middle East and Northern Africa
51	Thailand	6.0080	1.0500	1.4090	0.8280	0.5570	0.3590	0.0280	Southeastern Asia
52	Latvia	5.9400	1.1870	1.4650	0.8120	0.2640	0.0750	0.0640	Central and Eastern Europe
53	South Korea	5.8950	1.3010	1.2190	1.0360	0.1590	0.1750	0.0560	Eastern Asia

	Country or region	Score	GDP	Social	Life	Freedom	Generosity	Corruption	Continent
54	Estonia	5.8930	1.2370	1.5280	0.8740	0.4950	0.1030	0.1610	Central and Eastern Europe
55	Jamaica	5.8900	0.8310	1.4780	0.8310	0.4900	0.1070	0.0280	Latin America and Caribbean
56	Mauritius	5.8880	1.1200	1.4020	0.7980	0.4980	0.2150	0.0600	Sub-Saharan Africa
57	Japan	5.8860	1.3270	1.4190	1.0880	0.4450	0.0690	0.1400	Eastern Asia
58	Honduras	5.8600	0.6420	1.2360	0.8280	0.5070	0.2460	0.0780	Latin America and Caribbean
59	Kazakhstan	5.8090	1.1730	1.5080	0.7290	0.4100	0.1460	0.0960	Central and Eastern Europe
60	Bolivia	5.7790	0.7760	1.2090	0.7060	0.5110	0.1370	0.0640	Latin America and Caribbean
61	Hungary	5.7580	1.2010	1.4100	0.8280	0.1990	0.0810	0.0200	Central and Eastern Europe
62	Paraguay	5.7430	0.8550	1.4750	0.7770	0.5140	0.1840	0.0800	Latin America and Caribbean
63	Northern Cyprus	5.7180	1.2630	1.2520	1.0420	0.4170	0.1910	0.1620	NaN
64	Peru	5.6970	0.9600	1.2740	0.8540	0.4550	0.0830	0.0270	Latin America and Caribbean
65	Portugal	5.6930	1.2210	1.4310	0.9990	0.5080	0.0470	0.0250	Western Europe
66	Pakistan	5.6530	0.6770	0.8860	0.5350	0.3130	0.2200	0.0980	Southern Asia
67	Russia	5.6480	1.1830	1.4520	0.7260	0.3340	0.0820	0.0310	Central and Eastern Europe
68	Philippines	5.6310	0.8070	1.2930	0.6570	0.5580	0.1170	0.1070	Southeastern Asia
69	Serbia	5.6030	1.0040	1.3830	0.8540	0.2820	0.1370	0.0390	Central and Eastern Europe
70	Moldova	5.5290	0.6850	1.3280	0.7390	0.2450	0.1810	0.0000	Central and Eastern Europe
71	Libya	5.5250	1.0440	1.3030	0.6730	0.4160	0.1330	0.1520	Middle East and Northern Africa

	Country or region	Score	GDP	Social	Life	Freedom	Generosity	Corruption	Continent
72	Montenegro	5.5230	1.0510	1.3610	0.8710	0.1970	0.1420	0.0800	Central and Eastern Europe
73	Tajikistan	5.4670	0.4930	1.0980	0.7180	0.3890	0.2300	0.1440	Central and Eastern Europe
74	Croatia	5.4320	1.1550	1.2660	0.9140	0.2960	0.1190	0.0220	Central and Eastern Europe
75	Hong Kong	5.4300	1.4380	1.2770	1.1220	0.4400	0.2580	0.2870	Eastern Asia
76	Dominican Republic	5.4250	1.0150	1.4010	0.7790	0.4970	0.1130	0.1010	Latin America and Caribbean
77	Bosnia and Herzegovina	5.3860	0.9450	1.2120	0.8450	0.2120	0.2630	0.0060	Central and Eastern Europe
78	Turkey	5.3730	1.1830	1.3600	0.8080	0.1950	0.0830	0.1060	Middle East and Northern Africa
79	Malaysia	5.3390	1.2210	1.1710	0.8280	0.5080	0.2600	0.0240	Southeastern Asia
80	Belarus	5.3230	1.0670	1.4650	0.7890	0.2350	0.0940	0.1420	Central and Eastern Europe
81	Greece	5.2870	1.1810	1.1560	0.9990	0.0670	0.0000	0.0340	Western Europe
82	Mongolia	5.2850	0.9480	1.5310	0.6670	0.3170	0.2350	0.0380	Eastern Asia
83	North Macedonia	5.2740	0.9830	1.2940	0.8380	0.3450	0.1850	0.0340	NaN
84	Nigeria	5.2650	0.6960	1.1110	0.2450	0.4260	0.2150	0.0410	Sub-Saharan Africa
85	Kyrgyzstan	5.2610	0.5510	1.4380	0.7230	0.5080	0.3000	0.0230	Central and Eastern Europe
86	Turkmenistan	5.2470	1.0520	1.5380	0.6570	0.3940	0.2440	0.0280	Central and Eastern Europe
87	Algeria	5.2110	1.0020	1.1600	0.7850	0.0860	0.0730	0.1140	Middle East and Northern Africa
88	Morocco	5.2080	0.8010	0.7820	0.7820	0.4180	0.0360	0.0760	Middle East and Northern Africa
89	Azerbaijan	5.2080	1.0430	1.1470	0.7690	0.3510	0.0350	0.1820	Central and Eastern Europe

	Country or region	Score	GDP	Social	Life	Freedom	Generosity	Corruption	Continent
90	Lebanon	5.1970	0.9870	1.2240	0.8150	0.2160	0.1660	0.0270	Middle East and Northern Africa
91	Indonesia	5.1920	0.9310	1.2030	0.6600	0.4910	0.4980	0.0280	Southeastern Asia
92	China	5.1910	1.0290	1.1250	0.8930	0.5210	0.0580	0.1000	Eastern Asia
93	Vietnam	5.1750	0.7410	1.3460	0.8510	0.5430	0.1470	0.0730	Southeastern Asia
94	Bhutan	5.0820	0.8130	1.3210	0.6040	0.4570	0.3700	0.1670	Southern Asia
95	Cameroon	5.0440	0.5490	0.9100	0.3310	0.3810	0.1870	0.0370	Sub-Saharan Africa
96	Bulgaria	5.0110	1.0920	1.5130	0.8150	0.3110	0.0810	0.0040	Central and Eastern Europe
97	Ghana	4.9960	0.6110	0.8680	0.4860	0.3810	0.2450	0.0400	Sub-Saharan Africa
98	Ivory Coast	4.9440	0.5690	0.8080	0.2320	0.3520	0.1540	0.0900	Sub-Saharan Africa
99	Nepal	4.9130	0.4460	1.2260	0.6770	0.4390	0.2850	0.0890	Southern Asia
100	Jordan	4.9060	0.8370	1.2250	0.8150	0.3830	0.1100	0.1300	Middle East and Northern Africa
101	Benin	4.8830	0.3930	0.4370	0.3970	0.3490	0.1750	0.0820	Sub-Saharan Africa
102	Congo (Brazzaville)	4.8120	0.6730	0.7990	0.5080	0.3720	0.1050	0.0930	Sub-Saharan Africa
103	Gabon	4.7990	1.0570	1.1830	0.5710	0.2950	0.0430	0.0550	Sub-Saharan Africa
104	Laos	4.7960	0.7640	1.0300	0.5510	0.5470	0.2660	0.1640	Southeastern Asia
105	South Africa	4.7220	0.9600	1.3510	0.4690	0.3890	0.1300	0.0550	Sub-Saharan Africa
106	Albania	4.7190	0.9470	0.8480	0.8740	0.3830	0.1780	0.0270	Central and Eastern Europe
107	Venezuela	4.7070	0.9600	1.4270	0.8050	0.1540	0.0640	0.0470	Latin America and Caribbean
108	Cambodia	4.7000	0.5740	1.1220	0.6370	0.6090	0.2320	0.0620	Southeastern Asia

	Country or region	Score	GDP	Social	Life	Freedom	Generosity	Corruption	Continent
109	Palestinian Territories	4.6960	0.6570	1.2470	0.6720	0.2250	0.1030	0.0660	Middle East and Northern Africa
110	Senegal	4.6810	0.4500	1.1340	0.5710	0.2920	0.1530	0.0720	Sub-Saharan Africa
111	Somalia	4.6680	0.0000	0.6980	0.2680	0.5590	0.2430	0.2700	NaN
112	Namibia	4.6390	0.8790	1.3130	0.4770	0.4010	0.0700	0.0560	NaN
113	Niger	4.6280	0.1380	0.7740	0.3660	0.3180	0.1880	0.1020	Sub-Saharan Africa
114	Burkina Faso	4.5870	0.3310	1.0560	0.3800	0.2550	0.1770	0.1130	Sub-Saharan Africa
115	Armenia	4.5590	0.8500	1.0550	0.8150	0.2830	0.0950	0.0640	Central and Eastern Europe
116	Iran	4.5480	1.1000	0.8420	0.7850	0.3050	0.2700	0.1250	Middle East and Northern Africa
117	Guinea	4.5340	0.3800	0.8290	0.3750	0.3320	0.2070	0.0860	Sub-Saharan Africa
118	Georgia	4.5190	0.8860	0.6660	0.7520	0.3460	0.0430	0.1640	Central and Eastern Europe
119	Gambia	4.5160	0.3080	0.9390	0.4280	0.3820	0.2690	0.1670	NaN
120	Kenya	4.5090	0.5120	0.9830	0.5810	0.4310	0.3720	0.0530	Sub-Saharan Africa
121	Mauritania	4.4900	0.5700	1.1670	0.4890	0.0660	0.1060	0.0880	Sub-Saharan Africa
122	Mozambique	4.4660	0.2040	0.9860	0.3900	0.4940	0.1970	0.1380	Sub-Saharan Africa
123	Tunisia	4.4610	0.9210	1.0000	0.8150	0.1670	0.0590	0.0550	Middle East and Northern Africa
124	Bangladesh	4.4560	0.5620	0.9280	0.7230	0.5270	0.1660	0.1430	Southern Asia
125	Iraq	4.4370	1.0430	0.9800	0.5740	0.2410	0.1480	0.0890	Middle East and Northern Africa
126	Congo (Kinshasa)	4.4180	0.0940	1.1250	0.3570	0.2690	0.2120	0.0530	Sub-Saharan Africa
127	Mali	4.3900	0.3850	1.1050	0.3080	0.3270	0.1530	0.0520	Sub-Saharan Africa
128	Sierra Leone	4.3740	0.2680	0.8410	0.2420	0.3090	0.2520	0.0450	Sub-Saharan Africa

	Country or region	Score	GDP	Social	Life	Freedom	Generosity	Corruption	Continent
129	Sri Lanka	4.3660	0.9490	1.2650	0.8310	0.4700	0.2440	0.0470	Southern Asia
130	Myanmar	4.3600	0.7100	1.1810	0.5550	0.5250	0.5660	0.1720	Southeastern Asia
131	Chad	4.3500	0.3500	0.7660	0.1920	0.1740	0.1980	0.0780	Sub-Saharan Africa
132	Ukraine	4.3320	0.8200	1.3900	0.7390	0.1780	0.1870	0.0100	Central and Eastern Europe
133	Ethiopia	4.2860	0.3360	1.0330	0.5320	0.3440	0.2090	0.1000	Sub-Saharan Africa
134	Swaziland	4.2120	0.8110	1.1490	0.0000	0.3130	0.0740	0.1350	Sub-Saharan Africa
135	Uganda	4.1890	0.3320	1.0690	0.4430	0.3560	0.2520	0.0600	Sub-Saharan Africa
136	Egypt	4.1660	0.9130	1.0390	0.6440	0.2410	0.0760	0.0670	Middle East and Northern Africa
137	Zambia	4.1070	0.5780	1.0580	0.4260	0.4310	0.2470	0.0870	Sub-Saharan Africa
138	Togo	4.0850	0.2750	0.5720	0.4100	0.2930	0.1770	0.0850	Sub-Saharan Africa
139	India	4.0150	0.7550	0.7650	0.5880	0.4980	0.2000	0.0850	Southern Asia
140	Liberia	3.9750	0.0730	0.9220	0.4430	0.3700	0.2330	0.0330	Sub-Saharan Africa
141	Comoros	3.9730	0.2740	0.7570	0.5050	0.1420	0.2750	0.0780	Sub-Saharan Africa
142	Madagascar	3.9330	0.2740	0.9160	0.5550	0.1480	0.1690	0.0410	Sub-Saharan Africa
143	Lesotho	3.8020	0.4890	1.1690	0.1680	0.3590	0.1070	0.0930	Sub-Saharan Africa
144	Burundi	3.7750	0.0460	0.4470	0.3800	0.2200	0.1760	0.1800	Sub-Saharan Africa
145	Zimbabwe	3.6630	0.3660	1.1140	0.4330	0.3610	0.1510	0.0890	Sub-Saharan Africa
146	Haiti	3.5970	0.3230	0.6880	0.4490	0.0260	0.4190	0.1100	Latin America and Caribbean
147	Botswana	3.4880	1.0410	1.1450	0.5380	0.4550	0.0250	0.1000	Sub-Saharan Africa
148	Syria	3.4620	0.6190	0.3780	0.4400	0.0130	0.3310	0.1410	Middle East and Northern Africa

	Country or region	Score	GDP	Social	Life	Freedom	Generosity	Corruption	Continent
149	Malawi	3.4100	0.1910	0.5600	0.4950	0.4430	0.2180	0.0890	Sub-Saharan Africa
150	Yemen	3.3800	0.2870	1.1630	0.4630	0.1430	0.1080	0.0770	Middle East and Northern Africa
151	Rwanda	3.3340	0.3590	0.7110	0.6140	0.5550	0.2170	0.4110	Sub-Saharan Africa
152	Tanzania	3.2310	0.4760	0.8850	0.4990	0.4170	0.2760	0.1470	Sub-Saharan Africa
153	Afghanistan	3.2030	0.3500	0.5170	0.3610	0.0000	0.1580	0.0250	Southern Asia
154	Central African Republic	3.0830	0.0260	0.0000	0.1050	0.2250	0.2350	0.0350	Sub-Saharan Africa
155	South Sudan	2.8530	0.3060	0.5750	0.2950	0.0100	0.2020	0.0910	NaN

Let us take **Western Europe** as an example, extract the countries where **'Continent'** is **'Western Europe'** from the table, and draw a line chart of all variables in these countries (the same is true for other continents).

In [23]:
```
West_Europe=df2[df2['Continent']=='Western Europe']
West_Europe
```

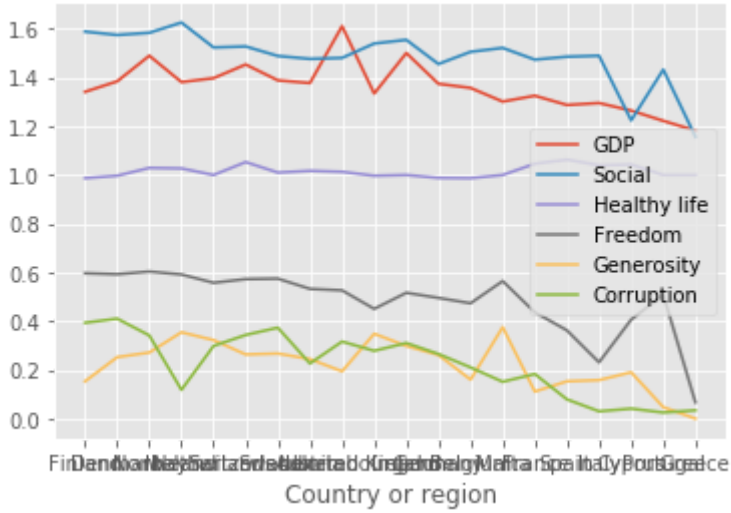
Out[23]:

	Country or region	Score	GDP	Social	Life	Freedom	Generosity	Corruption	Continent
0	Finland	7.7690	1.3400	1.5870	0.9860	0.5960	0.1530	0.3930	Western Europe
1	Denmark	7.6000	1.3830	1.5730	0.9960	0.5920	0.2520	0.4100	Western Europe
2	Norway	7.5540	1.4880	1.5820	1.0280	0.6030	0.2710	0.3410	Western Europe
3	Iceland	7.4940	1.3800	1.6240	1.0260	0.5910	0.3540	0.1180	Western Europe
4	Netherlands	7.4880	1.3960	1.5220	0.9990	0.5570	0.3220	0.2980	Western Europe
5	Switzerland	7.4800	1.4520	1.5260	1.0520	0.5720	0.2630	0.3430	Western Europe
6	Sweden	7.3430	1.3870	1.4870	1.0090	0.5740	0.2670	0.3730	Western Europe
9	Austria	7.2460	1.3760	1.4750	1.0160	0.5320	0.2440	0.2260	Western Europe
13	Luxembourg	7.0900	1.6090	1.4790	1.0120	0.5260	0.1940	0.3160	Western Europe
14	United Kingdom	7.0540	1.3330	1.5380	0.9960	0.4500	0.3480	0.2780	Western Europe

	Country or region	Score	GDP	Social	Life	Freedom	Generosity	Corruption	Continent
15	Ireland	7.0210	1.4990	1.5530	0.9990	0.5160	0.2980	0.3100	Western Europe
16	Germany	6.9850	1.3730	1.4540	0.9870	0.4950	0.2610	0.2650	Western Europe
17	Belgium	6.9230	1.3560	1.5040	0.9860	0.4730	0.1600	0.2100	Western Europe
21	Malta	6.7260	1.3000	1.5200	0.9990	0.5640	0.3750	0.1510	Western Europe
23	France	6.5920	1.3240	1.4720	1.0450	0.4360	0.1110	0.1830	Western Europe
29	Spain	6.3540	1.2860	1.4840	1.0620	0.3620	0.1530	0.0790	Western Europe
35	Italy	6.2230	1.2940	1.4880	1.0390	0.2310	0.1580	0.0300	Western Europe
48	Cyprus	6.0460	1.2630	1.2230	1.0420	0.4060	0.1900	0.0410	Western Europe
65	Portugal	5.6930	1.2210	1.4310	0.9990	0.5080	0.0470	0.0250	Western Europe
81	Greece	5.2870	1.1810	1.1560	0.9990	0.0670	0.0000	0.0340	Western Europe

In [24]:
```
X=West_Europe['Country or region']
Y1=West_Europe['GDP']
Y2=West_Europe['Social']
Y3=West_Europe['Life']
Y4=West_Europe['Freedom']
Y5=West_Europe['Generosity']
Y6=West_Europe['Corruption']
plt.plot(X, Y1,  label='GDP')
plt.plot(X, Y2,label='Social')
plt.plot(X, Y3,label='Healthy life')
plt.plot(X, Y4,label='Freedom')
plt.plot(X, Y5,label='Generosity')
plt.plot(X, Y6,label='Corruption')
plt.legend()
plt.xlabel('Country or region')
```

Out[24]:
```
Text(0.5, 0, 'Country or region')
```

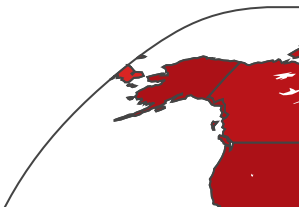


We plot the distribution of each variable on the world map separately for easy identification. (These codes are from Jordanka, M, could be found from here.)

In [25]:
```
Score=dict(type='choropleth',
           locations=data['Country or region'],
           locationmode='country names',
           colorscale="Jet",
           text=data['Country or region'],
           z=data['Score'],
           colorbar={'title':'Happiness score'})

layout=dict(title='2019 World Happiness Score',geo=dict(showframe=True,projection={'
choromap=go.Figure(data=[Score],layout=layout)
iplot(choromap)
```

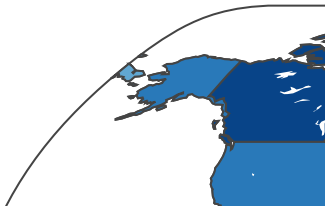
In [26]:
```
GDP=dict(type='choropleth',
          locations=data['Country or region'],
          locationmode='country names',
          colorscale="reds",
          text=data['Country or region'],
          z=data['GDP'],
          colorbar={'title':'GDP score'})

layout=dict(title='2019 World GDP Score',geo=dict(showframe=True,projection={'type':
choromap=go.Figure(data=[GDP],layout=layout)
iplot(choromap)
```

2019 World GDP Score



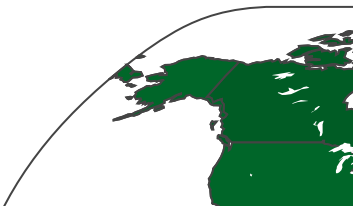
In [27]:
```
Freedom=dict(type='choropleth',
          locations=data['Country or region'],
          locationmode='country names',
          colorscale="blues",
          text=data['Country or region'],
          z=data['Freedom'],
          colorbar={'title':'Freedom score'})

layout=dict(title='2019 World Freedom Score',geo=dict(showframe=True,projection={'ty
choromap=go.Figure(data=[Freedom],layout=layout)
iplot(choromap)
```

2019 World Freedom Score



In [28]:
```
Social=dict(type='choropleth',
           locations=data['Country or region'],
           locationmode='country names',
           colorscale="greens",
           text=data['Country or region'],
           z=data['Social'],
           colorbar={'title':'Social Support score'})

layout=dict(title='2019 World Social Support Score',geo=dict(showframe=True,projecti
choromap=go.Figure(data=[Social],layout=layout)
iplot(choromap)
```

2019 World Social Support Score

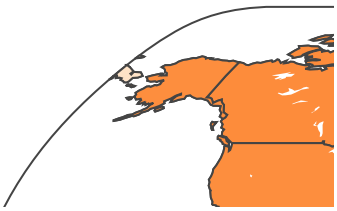


In [29]:
```
Life=dict(type='choropleth',
          locations=data['Country or region'],
          locationmode='country names',
          colorscale="purples",
          text=data['Country or region'],
          z=data['Life'],
          colorbar={'title':'Healthy life expectancy score'})

layout=dict(title='2019 World Healthy life expectancy Score',geo=dict(showframe=True
choromap=go.Figure(data=[Life],layout=layout)
iplot(choromap)
```

In [30]:
```
Generosity=dict(type='choropleth',
          locations=data['Country or region'],
          locationmode='country names',
          colorscale="oranges",
          text=data['Country or region'],
          z=data['Generosity'],
          colorbar={'title':'Generosity score'})

layout=dict(title='2019 World Generosity Score',geo=dict(showframe=True,projection={
```

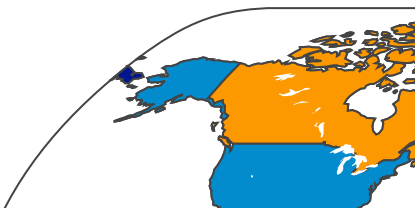
```
choromap=go.Figure(data=[Generosity],layout=layout)
iplot(choromap)
```

2019 World Generosity Score



In [31]:
```
Corruption=dict(type='choropleth',
           locations=data['Country or region'],
           locationmode='country names',
           colorscale="Jet",
           text=data['Country or region'],
           z=data['Corruption'],
           colorbar={'title':'Perceptions of corruption score'})

layout=dict(title='2019 World Perceptions of corruption Score',geo=dict(showframe=Tr
choromap=go.Figure(data=[Corruption],layout=layout)
iplot(choromap)
```

2019 World Perceptions of corruption Score



Methods

Since we selected six independent variables and one dependent variable, we can choose multiple linear regression to analyze which factors have a greater impact on happiness scores.

Before we perform multiple linear regression analysis, we should first consider the **multicollinearity** between independent variables. Multicollinearity means that there is a high degree of correlation between two or more independent variables. Multicollinearity may cause errors in data analysis results, so models with multicollinearity are not reliable. The causes of multicollinearity may be: insufficient data, incorrect use of dummy variables, an independent variable may be a combination of two or more other independent variables, and the two independent variables are the same (Stephanie, 2015). Therefore, we want to ensure that there is no multicollinearity between variables. We can use **"stepwise regression"** to check the **Variance Inflation Factor (VIF)** value of the independent variable and **LASSO** to filter the independent variable and then perform multiple linear regression.

We can use the correlation matrix to check the multicollinearity between variables.

We delete the variable **'Country or region'** that is not a numeric type and then check the result again with the `.info()` function.

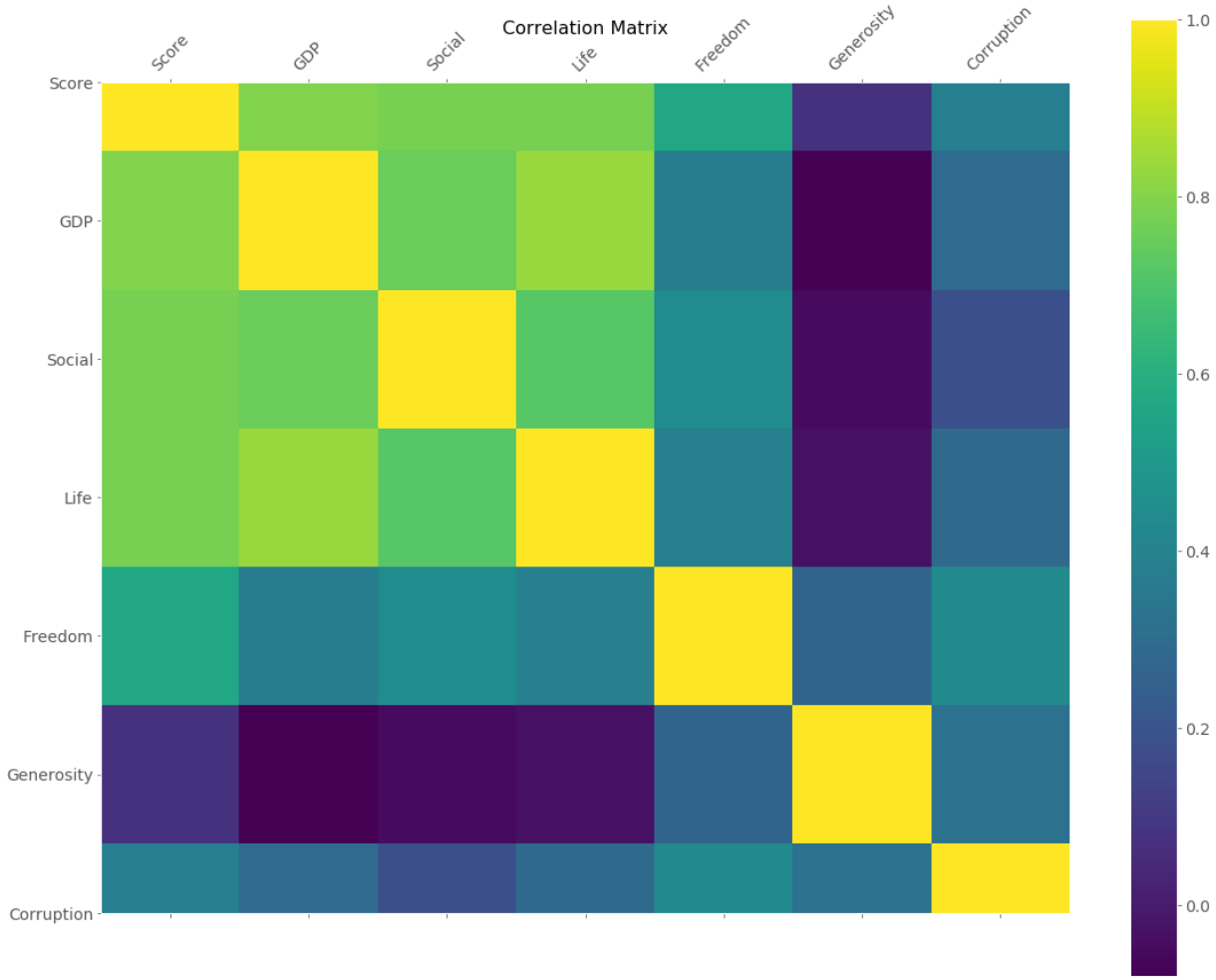
In [32]:
```
data_new=data.drop(['Country or region'], axis=1)
data_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 156 entries, 0 to 155
Data columns (total 7 columns):
Score         156 non-null float64
GDP           156 non-null float64
Social        156 non-null float64
Life          156 non-null float64
Freedom       156 non-null float64
Generosity    156 non-null float64
Corruption    156 non-null float64
dtypes: float64(7)
memory usage: 8.7 KB
```

In [33]:
```
df = data_new
plt.rcParams["axes.grid"] = False
f = plt.figure(figsize=(19, 15))
plt.matshow(df.corr(), fignum=f.number)
plt.xticks(range(df.shape[1]), df.columns, fontsize=14, rotation=45)
plt.yticks(range(df.shape[1]), df.columns, fontsize=14)
```

```
cb = plt.colorbar()
cb.ax.tick_params(labelsize=14)
plt.title('Correlation Matrix', fontsize=16)
```

Out[33]: Text(0.5, 1.05, 'Correlation Matrix')



Stepwise regression

The following code stepwise regression specific process.

In [34]:
```
def stepwise_selection(X, y,
                       initial_list=[],
                       threshold_in=0.01,
                       threshold_out = 0.05,
                       verbose=True):
    """ Perform a forward-backward feature selection
    based on p-value from statsmodels.api.OLS
    Arguments:
        X - pandas.DataFrame with candidate features
        y - list-like with the target
        initial_list - list of features to start with (column names of X)
        threshold_in - include a feature if its p-value < threshold_in
        threshold_out - exclude a feature if its p-value > threshold_out
        verbose - whether to print the sequence of inclusions and exclusions
    Returns: list of selected features
    Always set threshold_in < threshold_out to avoid infinite looping.
    See https://en.wikipedia.org/wiki/Stepwise_regression for the details
    """
    included = list(initial_list)
    while True:
        changed=False
        # forward step
```

```
        excluded = list(set(X.columns)-set(included))
        new_pval = pd.Series(index=excluded)
        for new_column in excluded:
            model = sm.OLS(y, sm.add_constant(pd.DataFrame(X[included+[new_column]]))
            new_pval[new_column] = model.pvalues[new_column]
        best_pval = new_pval.min()
        if best_pval < threshold_in:
            best_feature = new_pval.idxmin()
            included.append(best_feature)
            changed=True
            if verbose:
                print('Add  {:30} with p-value {:.6}'.format(best_feature, best_pval

        # backward step
        model = sm.OLS(y, sm.add_constant(pd.DataFrame(X[included]))).fit()
        # use all coefs except intercept
        pvalues = model.pvalues.iloc[1:]
        worst_pval = pvalues.max() # null if pvalues is empty
        if worst_pval > threshold_out:
            changed=True
            worst_feature = pvalues.argmax()
            included.remove(worst_feature)
            if verbose:
                print('Drop {:30} with p-value {:.6}'.format(worst_feature, worst_pv
        if not changed:
            break
    return included
```

We divide the data frame into independent variables(predictors_data) and dependent variables(response_data).

In [35]:
```
predictors_data = data_new.drop(columns=['Score'], axis=1)
response_data = data[['Score']]
```

Perform stepwise regression on our data.

In [36]:
```
import statsmodels.api as sm
import statsmodels.formula.api as smf
```

In [37]:
```
list_predictor_select_stepwise = stepwise_selection(X=predictors_data, y=response_da
print("The predictors that are selected by stepwise regression are:")
print(list_predictor_select_stepwise)
```

```
Add  GDP                            with p-value 4.31548e-35
Add  Freedom                        with p-value 5.45461e-10
Add  Social                         with p-value 1.44921e-06
Add  Life                           with p-value 0.000910305
The predictors that are selected by stepwise regression are:
['GDP', 'Freedom', 'Social', 'Life']
/Users/wangshengming/opt/anaconda3/lib/python3.7/site-packages/numpy/core/fromnumeri
c.py:2542: FutureWarning:

Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp ins
tead.
```

From the above results, we can see that stepwise regression has selected four independent variables for us, namely: **GDP per capita**, **Freedom to make life choices**, **Social support**, **Healthy life expectancy**.

Now we add these four independent variables to the model for fitting and summarize the model through the `.summary()` function.

In [38]:
```
multi_regression_model_1 = smf.ols(formula='Score ~ GDP+Freedom+Social+Life ',data=d

print(multi_regression_model_1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Score   R-squared:                       0.771
Model:                            OLS   Adj. R-squared:                  0.765
Method:                 Least Squares   F-statistic:                     127.0
Date:                Thu, 07 May 2020   Prob (F-statistic):           2.82e-47
Time:                        14:42:51   Log-Likelihood:                -122.62
No. Observations:                 156   AIC:                             255.2
Df Residuals:                     151   BIC:                             270.5
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      1.8921      0.199      9.491      0.000       1.498       2.286
GDP            0.8105      0.216      3.745      0.000       0.383       1.238
Freedom        1.8458      0.340      5.423      0.000       1.173       2.518
Social         1.0166      0.235      4.331      0.000       0.553       1.480
Life           1.1414      0.337      3.384      0.001       0.475       1.808
==============================================================================
Omnibus:                        5.077   Durbin-Watson:                   1.641
Prob(Omnibus):                  0.079   Jarque-Bera (JB):                4.685
Skew:                          -0.413   Prob(JB):                       0.0961
Kurtosis:                       3.198   Cond. No.                         17.8
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
```

The regression results show that our model **R2** score is 0.771, which can explain 77.1% of the variability of the happiness score. The **p-values** of these four independent variables are all less than 0.05, that is to say, all four independent variables are significant

Variance Inflation Factor (VIF)

Variance Inflation Factor is a measure of multicollinearity in multiple linear regression models. VIF is used to estimate the degree of expansion of regression coefficients in the model due to the presence of multicollinearity (Jim, 2018). The higher the value of VIF, the more serious the multicollinearity between the independent variables; and if the value of VIF is less than 10, it means that the multicollinearity between the independent variables is not serious, do not need to worry.

We use the following code to calculate the VIF value of each variable.

In [39]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant

def calculate_vif_(df, thresh=5):
    '''
    Calculates VIF each feature in a pandas dataframe
    A constant must be added to variance_inflation_factor or the results will be inc

    :param df: the pandas dataframe containing only the predictor features, not the
    :param thresh: the max VIF value before the feature is removed from the datafram
    :return: dataframe with features removed
    '''
    const = add_constant(df)
    cols = const.columns
```

```
    variables = np.arange(const.shape[1])
    vif_df = pd.Series([variance_inflation_factor(const.values, i)
               for i in range(const.shape[1])],
              index=const.columns).to_frame()

    vif_df = vif_df.sort_values(by=0, ascending=False).rename(columns={0: 'VIF'})
    vif_df = vif_df.drop('const')
    vif_df = vif_df[vif_df['VIF'] > thresh]

    print('Features above VIF threshold:\n')
    print(vif_df[vif_df['VIF'] > thresh])

    col_to_drop = list(vif_df.index)

    for i in col_to_drop:
        print('Dropping: {}'.format(i))
        df = df.drop(columns=i)

    return df
```

In [40]:
```
df_predictors_select_VIF = calculate_vif_(predictors_data)
print("The columns remaining after VIF selection are:")
print(df_predictors_select_VIF.columns)
```

```
Features above VIF threshold:

Empty DataFrame
Columns: [VIF]
Index: []
The columns remaining after VIF selection are:
Index(['GDP', 'Social', 'Life', 'Freedom', 'Generosity', 'Corruption'], dtype='objec
t')
```

We can see that after calculating the VIF, all the independent variables are selected.

After processing multicollinearity, we can perform multiple linear regression fitting. We will use the `.fit()` function in the `sklearn` package for fitting.

In [41]:
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X=df_predictors_select_VIF, y=response_data)
```

Out[41]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

After creating the fit, we add the variables selected by the "VIF" method to the model and hope to get the model's constraints and the coefficients of each independent variable. We can achieve this process through the following code.

In [42]:
```
print(lr.intercept_)
df_coef_lr_VIF = pd.DataFrame({"var": df_predictors_select_VIF.columns.values, "coef
print(df_coef_lr_VIF)
```

```
[1.79522023]
          var                                   coef
0         GDP                                 0.7754
1      Social                                 1.1242
2        Life                                 1.0781
3     Freedom                                 1.4548
4  Generosity                                 0.4898
5  Corruption                                 0.9723
```

From the above results, we get the intercept of the model and the coefficients of each independent variable. Moreover, we also want to get the **R2** score of this model so that we can

understand how well the model fits. We can get the **R2** score of this model by running the following code.

In [43]:
```
lr.score(X=df_predictors_select_VIF, y=response_data)
```

Out[43]:
```
0.7791638079594221
```

To summarize this model, we will use the `summary()` function in the `statsmodel` package to perform this operation.

In [44]:
```
regressor_OLS = sm.OLS(endog=response_data, exog=sm.add_constant(df_predictors_selec
regressor_OLS.summary()
```

```
/Users/wangshengming/opt/anaconda3/lib/python3.7/site-packages/numpy/core/fromnumeri
c.py:2542: FutureWarning:

Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp ins
tead.
```

Out[44]:

OLS Regression Results

Dep. Variable:	Score	**R-squared:**	0.779
Model:	OLS	**Adj. R-squared:**	0.770
Method:	Least Squares	**F-statistic:**	87.62
Date:	Thu, 07 May 2020	**Prob (F-statistic):**	2.40e-46
Time:	14:42:52	**Log-Likelihood:**	-119.76
No. Observations:	156	**AIC:**	253.5
Df Residuals:	149	**BIC:**	274.9
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1.7952	0.211	8.505	0.000	1.378	2.212
GDP	0.7754	0.218	3.553	0.001	0.344	1.207
Social	1.1242	0.237	4.745	0.000	0.656	1.592
Life	1.0781	0.335	3.223	0.002	0.417	1.739
Freedom	1.4548	0.375	3.876	0.000	0.713	2.197
Generosity	0.4898	0.498	0.984	0.327	-0.494	1.473
Corruption	0.9723	0.542	1.793	0.075	-0.099	2.044

Omnibus:	8.188	**Durbin-Watson:**	1.648
Prob(Omnibus):	0.017	**Jarque-Bera (JB):**	7.971
Skew:	-0.498	**Prob(JB):**	0.0186
Kurtosis:	3.483	**Cond. No.**	28.7

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

This summary shows that the **R2** score of our model is 0.77916, which means that 77.9% of the variance in the dependent variable (happiness score) can be explained by the independent variables used. Therefore, the fitting degree of our model is relatively good.

The **P>|t|** column represents the **p-value** of each independent variable, of which only the **Generosity** and **Perceptions of corruption** have **p-values** greater than 0.05 (0.327 and 0.075). This indicates that these two independent variables are non-significant, and the other independent variables are significant.

Lasso

To improve the prediction accuracy and interpretability of the regression model, we can use the **Lasso** model to select variables and perform regularized regression methods. We run the following codes to fit the **Lasso** model.

In [45]:
```
from sklearn import linear_model
lasso_model = linear_model.Lasso(max_iter=10e7, normalize=True)
lasso_model.fit(X=predictors_data, y=response_data)
```

Out[45]:
```
Lasso(alpha=1.0, copy_X=True, fit_intercept=True, max_iter=100000000.0,
      normalize=True, positive=False, precompute=False, random_state=None,
      selection='cyclic', tol=0.0001, warm_start=False)
```

In [46]:
```
lasso_model.score(X=predictors_data, y=response_data)
```

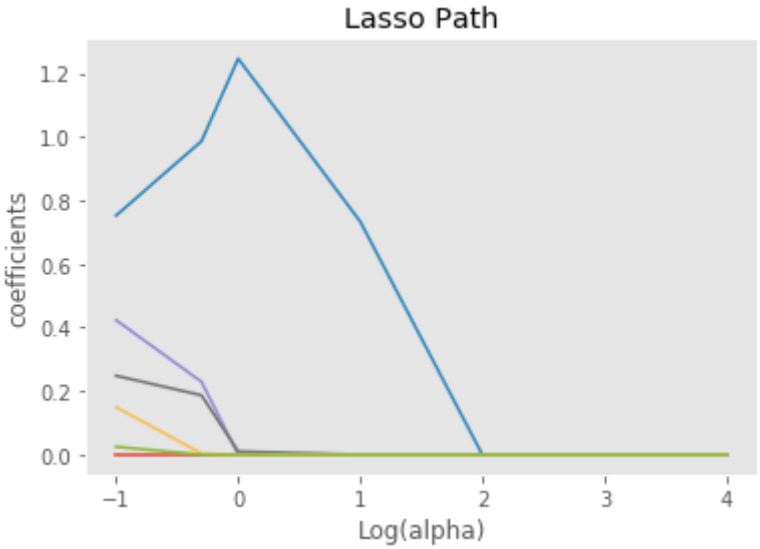
Out[46]:
```
0.0
```

In [47]:
```
df_coef_lasso = pd.DataFrame({"var": predictors_data.columns.values, "coef":lasso_mo
print(df_coef_lasso)
```

```
          var                                       coef
0         GDP                                     0.0000
1      Social                                     0.0000
2        Life                                     0.0000
3     Freedom                                     0.0000
4  Generosity                                     0.0000
5  Corruption                                     0.0000
```

In [48]:
```
from itertools import cycle
from sklearn.linear_model import lasso_path

predictors_data_standardised = predictors_data/predictors_data.std(axis=0)

alphas_lasso, coefs_lasso, _ = lasso_path(predictors_data_standardised, response_dat


log_alphas_lasso = np.log10(alphas_lasso)
for coef_l in coefs_lasso:
    l1 = plt.plot(log_alphas_lasso, coef_l)

plt.xlabel('Log(alpha)')
plt.ylabel('coefficients')
plt.title('Lasso Path')
plt.axis('tight')
```

Out[48]:
```
(-1.25, 4.25, -0.06233520502743616, 1.3090393055761593)
```



Since our **lasso_model.score** is 0, it means that the Lasso model does not do well in fitting the relationship between the independent and dependent variables. Moreover, the plot of Lasso Path is not very good.

Regression Tree

Then we use the **Regression Tree** for this data set and check its R2 score with the `.score()` function.

In [49]:
```
from sklearn.tree import DecisionTreeRegressor
reg_tree = DecisionTreeRegressor(random_state=0)
reg_tree.fit(predictors_data, response_data)
```

Out[49]:
```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      presort=False, random_state=0, splitter='best')
```

In [50]:
```
reg_tree.score(X=predictors_data, y=response_data)
```

Out[50]:
```
1.0
```

The **R2** score of our regression model is 1, it seems that the fitting effect of this model is perfect. However, there may be overfitting.

We use the `.describe(include = [np.number])` function to check the numeric values.

In [51]:
```
data.describe(include=[np.number])
```

Out[51]:

	Score	GDP	Social	Life	Freedom	Generosity	Corruption
count	156.0000	156.0000	156.0000	156.0000	156.0000	156.0000	156.0000
mean	5.4071	0.9051	1.2088	0.7252	0.3926	0.1848	0.1106
std	1.1131	0.3984	0.2992	0.2421	0.1433	0.0953	0.0945
min	2.8530	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
25%	4.5445	0.6028	1.0557	0.5477	0.3080	0.1088	0.0470
50%	5.3795	0.9600	1.2715	0.7890	0.4170	0.1775	0.0855
75%	6.1845	1.2325	1.4525	0.8817	0.5072	0.2482	0.1412

	Score	GDP	Social	Life	Freedom	Generosity	Corruption
max	7.7690	1.6840	1.6240	1.1410	0.6310	0.5660	0.4530

Then we continue to calculate the **average** of the happiness score (Score).

In [52]:
```
ave=np.mean(data['Score'])
ave
```

Out[52]: 5.407096153846153

We take whether the average value of the happiness score of each country exceeds the average value as an indicator. Our goal is to establish the relationship between the six factors that affect the happiness score and whether the happiness score is above the average value (described by **Boolean**).

In [53]:
```
a=pd.DataFrame({'over_mean':data['Score']>=ave})
a
```

Out[53]:

	over_mean
0	True
1	True
2	True
3	True
4	True
5	True
6	True
7	True
8	True
9	True
10	True
11	True
12	True
13	True
14	True
15	True
16	True
17	True
18	True
19	True
20	True
21	True
22	True
23	True

	over_mean
24	True
25	True
26	True
27	True
28	True
29	True
30	True
31	True
32	True
33	True
34	True
35	True
36	True
37	True
38	True
39	True
40	True
41	True
42	True
43	True
44	True
45	True
46	True
47	True
48	True
49	True
50	True
51	True
52	True
53	True
54	True
55	True
56	True
57	True
58	True
59	True

	over_mean
60	True
61	True
62	True
63	True
64	True
65	True
66	True
67	True
68	True
69	True
70	True
71	True
72	True
73	True
74	True
75	True
76	True
77	False
78	False
79	False
80	False
81	False
82	False
83	False
84	False
85	False
86	False
87	False
88	False
89	False
90	False
91	False
92	False
93	False
94	False
95	False

	over_mean
96	False
97	False
98	False
99	False
100	False
101	False
102	False
103	False
104	False
105	False
106	False
107	False
108	False
109	False
110	False
111	False
112	False
113	False
114	False
115	False
116	False
117	False
118	False
119	False
120	False
121	False
122	False
123	False
124	False
125	False
126	False
127	False
128	False
129	False
130	False
131	False

	over_mean
132	False
133	False
134	False
135	False
136	False
137	False
138	False
139	False
140	False
141	False
142	False
143	False
144	False
145	False
146	False
147	False
148	False
149	False
150	False
151	False
152	False
153	False
154	False
155	False

We use the `.concat()` function to merge the Boolean value of whether the obtained happiness score is above the average with our data table.

In [54]:
```
b = pd.concat( [data , a], axis=1 )
b
```

Out[54]:

	Country or region	Score	GDP	Social	Life	Freedom	Generosity	Corruption	over_mean
0	Finland	7.7690	1.3400	1.5870	0.9860	0.5960	0.1530	0.3930	True
1	Denmark	7.6000	1.3830	1.5730	0.9960	0.5920	0.2520	0.4100	True
2	Norway	7.5540	1.4880	1.5820	1.0280	0.6030	0.2710	0.3410	True
3	Iceland	7.4940	1.3800	1.6240	1.0260	0.5910	0.3540	0.1180	True
4	Netherlands	7.4880	1.3960	1.5220	0.9990	0.5570	0.3220	0.2980	True
5	Switzerland	7.4800	1.4520	1.5260	1.0520	0.5720	0.2630	0.3430	True

	Country or region	Score	GDP	Social	Life	Freedom	Generosity	Corruption	over_mean
6	Sweden	7.3430	1.3870	1.4870	1.0090	0.5740	0.2670	0.3730	True
7	New Zealand	7.3070	1.3030	1.5570	1.0260	0.5850	0.3300	0.3800	True
8	Canada	7.2780	1.3650	1.5050	1.0390	0.5840	0.2850	0.3080	True
9	Austria	7.2460	1.3760	1.4750	1.0160	0.5320	0.2440	0.2260	True
10	Australia	7.2280	1.3720	1.5480	1.0360	0.5570	0.3320	0.2900	True
11	Costa Rica	7.1670	1.0340	1.4410	0.9630	0.5580	0.1440	0.0930	True
12	Israel	7.1390	1.2760	1.4550	1.0290	0.3710	0.2610	0.0820	True
13	Luxembourg	7.0900	1.6090	1.4790	1.0120	0.5260	0.1940	0.3160	True
14	United Kingdom	7.0540	1.3330	1.5380	0.9960	0.4500	0.3480	0.2780	True
15	Ireland	7.0210	1.4990	1.5530	0.9990	0.5160	0.2980	0.3100	True
16	Germany	6.9850	1.3730	1.4540	0.9870	0.4950	0.2610	0.2650	True
17	Belgium	6.9230	1.3560	1.5040	0.9860	0.4730	0.1600	0.2100	True
18	United States	6.8920	1.4330	1.4570	0.8740	0.4540	0.2800	0.1280	True
19	Czech Republic	6.8520	1.2690	1.4870	0.9200	0.4570	0.0460	0.0360	True
20	United Arab Emirates	6.8250	1.5030	1.3100	0.8250	0.5980	0.2620	0.1820	True
21	Malta	6.7260	1.3000	1.5200	0.9990	0.5640	0.3750	0.1510	True
22	Mexico	6.5950	1.0700	1.3230	0.8610	0.4330	0.0740	0.0730	True
23	France	6.5920	1.3240	1.4720	1.0450	0.4360	0.1110	0.1830	True
24	Taiwan	6.4460	1.3680	1.4300	0.9140	0.3510	0.2420	0.0970	True
25	Chile	6.4440	1.1590	1.3690	0.9200	0.3570	0.1870	0.0560	True
26	Guatemala	6.4360	0.8000	1.2690	0.7460	0.5350	0.1750	0.0780	True
27	Saudi Arabia	6.3750	1.4030	1.3570	0.7950	0.4390	0.0800	0.1320	True
28	Qatar	6.3740	1.6840	1.3130	0.8710	0.5550	0.2200	0.1670	True
29	Spain	6.3540	1.2860	1.4840	1.0620	0.3620	0.1530	0.0790	True
30	Panama	6.3210	1.1490	1.4420	0.9100	0.5160	0.1090	0.0540	True
31	Brazil	6.3000	1.0040	1.4390	0.8020	0.3900	0.0990	0.0860	True
32	Uruguay	6.2930	1.1240	1.4650	0.8910	0.5230	0.1270	0.1500	True
33	Singapore	6.2620	1.5720	1.4630	1.1410	0.5560	0.2710	0.4530	True
34	El Salvador	6.2530	0.7940	1.2420	0.7890	0.4300	0.0930	0.0740	True
35	Italy	6.2230	1.2940	1.4880	1.0390	0.2310	0.1580	0.0300	True
36	Bahrain	6.1990	1.3620	1.3680	0.8710	0.5360	0.2550	0.1100	True
37	Slovakia	6.1980	1.2460	1.5040	0.8810	0.3340	0.1210	0.0140	True
38	Trinidad & Tobago	6.1920	1.2310	1.4770	0.7130	0.4890	0.1850	0.0160	True

	Country or region	Score	GDP	Social	Life	Freedom	Generosity	Corruption	over_mean
39	Poland	6.1820	1.2060	1.4380	0.8840	0.4830	0.1170	0.0500	True
40	Uzbekistan	6.1740	0.7450	1.5290	0.7560	0.6310	0.3220	0.2400	True
41	Lithuania	6.1490	1.2380	1.5150	0.8180	0.2910	0.0430	0.0420	True
42	Colombia	6.1250	0.9850	1.4100	0.8410	0.4700	0.0990	0.0340	True
43	Slovenia	6.1180	1.2580	1.5230	0.9530	0.5640	0.1440	0.0570	True
44	Nicaragua	6.1050	0.6940	1.3250	0.8350	0.4350	0.2000	0.1270	True
45	Kosovo	6.1000	0.8820	1.2320	0.7580	0.4890	0.2620	0.0060	True
46	Argentina	6.0860	1.0920	1.4320	0.8810	0.4710	0.0660	0.0500	True
47	Romania	6.0700	1.1620	1.2320	0.8250	0.4620	0.0830	0.0050	True
48	Cyprus	6.0460	1.2630	1.2230	1.0420	0.4060	0.1900	0.0410	True
49	Ecuador	6.0280	0.9120	1.3120	0.8680	0.4980	0.1260	0.0870	True
50	Kuwait	6.0210	1.5000	1.3190	0.8080	0.4930	0.1420	0.0970	True
51	Thailand	6.0080	1.0500	1.4090	0.8280	0.5570	0.3590	0.0280	True
52	Latvia	5.9400	1.1870	1.4650	0.8120	0.2640	0.0750	0.0640	True
53	South Korea	5.8950	1.3010	1.2190	1.0360	0.1590	0.1750	0.0560	True
54	Estonia	5.8930	1.2370	1.5280	0.8740	0.4950	0.1030	0.1610	True
55	Jamaica	5.8900	0.8310	1.4780	0.8310	0.4900	0.1070	0.0280	True
56	Mauritius	5.8880	1.1200	1.4020	0.7980	0.4980	0.2150	0.0600	True
57	Japan	5.8860	1.3270	1.4190	1.0880	0.4450	0.0690	0.1400	True
58	Honduras	5.8600	0.6420	1.2360	0.8280	0.5070	0.2460	0.0780	True
59	Kazakhstan	5.8090	1.1730	1.5080	0.7290	0.4100	0.1460	0.0960	True
60	Bolivia	5.7790	0.7760	1.2090	0.7060	0.5110	0.1370	0.0640	True
61	Hungary	5.7580	1.2010	1.4100	0.8280	0.1990	0.0810	0.0200	True
62	Paraguay	5.7430	0.8550	1.4750	0.7770	0.5140	0.1840	0.0800	True
63	Northern Cyprus	5.7180	1.2630	1.2520	1.0420	0.4170	0.1910	0.1620	True
64	Peru	5.6970	0.9600	1.2740	0.8540	0.4550	0.0830	0.0270	True
65	Portugal	5.6930	1.2210	1.4310	0.9990	0.5080	0.0470	0.0250	True
66	Pakistan	5.6530	0.6770	0.8860	0.5350	0.3130	0.2200	0.0980	True
67	Russia	5.6480	1.1830	1.4520	0.7260	0.3340	0.0820	0.0310	True
68	Philippines	5.6310	0.8070	1.2930	0.6570	0.5580	0.1170	0.1070	True
69	Serbia	5.6030	1.0040	1.3830	0.8540	0.2820	0.1370	0.0390	True
70	Moldova	5.5290	0.6850	1.3280	0.7390	0.2450	0.1810	0.0000	True
71	Libya	5.5250	1.0440	1.3030	0.6730	0.4160	0.1330	0.1520	True
72	Montenegro	5.5230	1.0510	1.3610	0.8710	0.1970	0.1420	0.0800	True
73	Tajikistan	5.4670	0.4930	1.0980	0.7180	0.3890	0.2300	0.1440	True

	Country or region	Score	GDP	Social	Life	Freedom	Generosity	Corruption	over_mean
74	Croatia	5.4320	1.1550	1.2660	0.9140	0.2960	0.1190	0.0220	True
75	Hong Kong	5.4300	1.4380	1.2770	1.1220	0.4400	0.2580	0.2870	True
76	Dominican Republic	5.4250	1.0150	1.4010	0.7790	0.4970	0.1130	0.1010	True
77	Bosnia and Herzegovina	5.3860	0.9450	1.2120	0.8450	0.2120	0.2630	0.0060	False
78	Turkey	5.3730	1.1830	1.3600	0.8080	0.1950	0.0830	0.1060	False
79	Malaysia	5.3390	1.2210	1.1710	0.8280	0.5080	0.2600	0.0240	False
80	Belarus	5.3230	1.0670	1.4650	0.7890	0.2350	0.0940	0.1420	False
81	Greece	5.2870	1.1810	1.1560	0.9990	0.0670	0.0000	0.0340	False
82	Mongolia	5.2850	0.9480	1.5310	0.6670	0.3170	0.2350	0.0380	False
83	North Macedonia	5.2740	0.9830	1.2940	0.8380	0.3450	0.1850	0.0340	False
84	Nigeria	5.2650	0.6960	1.1110	0.2450	0.4260	0.2150	0.0410	False
85	Kyrgyzstan	5.2610	0.5510	1.4380	0.7230	0.5080	0.3000	0.0230	False
86	Turkmenistan	5.2470	1.0520	1.5380	0.6570	0.3940	0.2440	0.0280	False
87	Algeria	5.2110	1.0020	1.1600	0.7850	0.0860	0.0730	0.1140	False
88	Morocco	5.2080	0.8010	0.7820	0.7820	0.4180	0.0360	0.0760	False
89	Azerbaijan	5.2080	1.0430	1.1470	0.7690	0.3510	0.0350	0.1820	False
90	Lebanon	5.1970	0.9870	1.2240	0.8150	0.2160	0.1660	0.0270	False
91	Indonesia	5.1920	0.9310	1.2030	0.6600	0.4910	0.4980	0.0280	False
92	China	5.1910	1.0290	1.1250	0.8930	0.5210	0.0580	0.1000	False
93	Vietnam	5.1750	0.7410	1.3460	0.8510	0.5430	0.1470	0.0730	False
94	Bhutan	5.0820	0.8130	1.3210	0.6040	0.4570	0.3700	0.1670	False
95	Cameroon	5.0440	0.5490	0.9100	0.3310	0.3810	0.1870	0.0370	False
96	Bulgaria	5.0110	1.0920	1.5130	0.8150	0.3110	0.0810	0.0040	False
97	Ghana	4.9960	0.6110	0.8680	0.4860	0.3810	0.2450	0.0400	False
98	Ivory Coast	4.9440	0.5690	0.8080	0.2320	0.3520	0.1540	0.0900	False
99	Nepal	4.9130	0.4460	1.2260	0.6770	0.4390	0.2850	0.0890	False
100	Jordan	4.9060	0.8370	1.2250	0.8150	0.3830	0.1100	0.1300	False
101	Benin	4.8830	0.3930	0.4370	0.3970	0.3490	0.1750	0.0820	False
102	Congo (Brazzaville)	4.8120	0.6730	0.7990	0.5080	0.3720	0.1050	0.0930	False
103	Gabon	4.7990	1.0570	1.1830	0.5710	0.2950	0.0430	0.0550	False
104	Laos	4.7960	0.7640	1.0300	0.5510	0.5470	0.2660	0.1640	False
105	South Africa	4.7220	0.9600	1.3510	0.4690	0.3890	0.1300	0.0550	False
106	Albania	4.7190	0.9470	0.8480	0.8740	0.3830	0.1780	0.0270	False

	Country or region	Score	GDP	Social	Life	Freedom	Generosity	Corruption	over_mean
107	Venezuela	4.7070	0.9600	1.4270	0.8050	0.1540	0.0640	0.0470	False
108	Cambodia	4.7000	0.5740	1.1220	0.6370	0.6090	0.2320	0.0620	False
109	Palestinian Territories	4.6960	0.6570	1.2470	0.6720	0.2250	0.1030	0.0660	False
110	Senegal	4.6810	0.4500	1.1340	0.5710	0.2920	0.1530	0.0720	False
111	Somalia	4.6680	0.0000	0.6980	0.2680	0.5590	0.2430	0.2700	False
112	Namibia	4.6390	0.8790	1.3130	0.4770	0.4010	0.0700	0.0560	False
113	Niger	4.6280	0.1380	0.7740	0.3660	0.3180	0.1880	0.1020	False
114	Burkina Faso	4.5870	0.3310	1.0560	0.3800	0.2550	0.1770	0.1130	False
115	Armenia	4.5590	0.8500	1.0550	0.8150	0.2830	0.0950	0.0640	False
116	Iran	4.5480	1.1000	0.8420	0.7850	0.3050	0.2700	0.1250	False
117	Guinea	4.5340	0.3800	0.8290	0.3750	0.3320	0.2070	0.0860	False
118	Georgia	4.5190	0.8860	0.6660	0.7520	0.3460	0.0430	0.1640	False
119	Gambia	4.5160	0.3080	0.9390	0.4280	0.3820	0.2690	0.1670	False
120	Kenya	4.5090	0.5120	0.9830	0.5810	0.4310	0.3720	0.0530	False
121	Mauritania	4.4900	0.5700	1.1670	0.4890	0.0660	0.1060	0.0880	False
122	Mozambique	4.4660	0.2040	0.9860	0.3900	0.4940	0.1970	0.1380	False
123	Tunisia	4.4610	0.9210	1.0000	0.8150	0.1670	0.0590	0.0550	False
124	Bangladesh	4.4560	0.5620	0.9280	0.7230	0.5270	0.1660	0.1430	False
125	Iraq	4.4370	1.0430	0.9800	0.5740	0.2410	0.1480	0.0890	False
126	Congo (Kinshasa)	4.4180	0.0940	1.1250	0.3570	0.2690	0.2120	0.0530	False
127	Mali	4.3900	0.3850	1.1050	0.3080	0.3270	0.1530	0.0520	False
128	Sierra Leone	4.3740	0.2680	0.8410	0.2420	0.3090	0.2520	0.0450	False
129	Sri Lanka	4.3660	0.9490	1.2650	0.8310	0.4700	0.2440	0.0470	False
130	Myanmar	4.3600	0.7100	1.1810	0.5550	0.5250	0.5660	0.1720	False
131	Chad	4.3500	0.3500	0.7660	0.1920	0.1740	0.1980	0.0780	False
132	Ukraine	4.3320	0.8200	1.3900	0.7390	0.1780	0.1870	0.0100	False
133	Ethiopia	4.2860	0.3360	1.0330	0.5320	0.3440	0.2090	0.1000	False
134	Swaziland	4.2120	0.8110	1.1490	0.0000	0.3130	0.0740	0.1350	False
135	Uganda	4.1890	0.3320	1.0690	0.4430	0.3560	0.2520	0.0600	False
136	Egypt	4.1660	0.9130	1.0390	0.6440	0.2410	0.0760	0.0670	False
137	Zambia	4.1070	0.5780	1.0580	0.4260	0.4310	0.2470	0.0870	False
138	Togo	4.0850	0.2750	0.5720	0.4100	0.2930	0.1770	0.0850	False
139	India	4.0150	0.7550	0.7650	0.5880	0.4980	0.2000	0.0850	False
140	Liberia	3.9750	0.0730	0.9220	0.4430	0.3700	0.2330	0.0330	False

	Country or region	Score	GDP	Social	Life	Freedom	Generosity	Corruption	over_mean
141	Comoros	3.9730	0.2740	0.7570	0.5050	0.1420	0.2750	0.0780	False
142	Madagascar	3.9330	0.2740	0.9160	0.5550	0.1480	0.1690	0.0410	False
143	Lesotho	3.8020	0.4890	1.1690	0.1680	0.3590	0.1070	0.0930	False
144	Burundi	3.7750	0.0460	0.4470	0.3800	0.2200	0.1760	0.1800	False
145	Zimbabwe	3.6630	0.3660	1.1140	0.4330	0.3610	0.1510	0.0890	False
146	Haiti	3.5970	0.3230	0.6880	0.4490	0.0260	0.4190	0.1100	False
147	Botswana	3.4880	1.0410	1.1450	0.5380	0.4550	0.0250	0.1000	False
148	Syria	3.4620	0.6190	0.3780	0.4400	0.0130	0.3310	0.1410	False
149	Malawi	3.4100	0.1910	0.5600	0.4950	0.4430	0.2180	0.0890	False
150	Yemen	3.3800	0.2870	1.1630	0.4630	0.1430	0.1080	0.0770	False
151	Rwanda	3.3340	0.3590	0.7110	0.6140	0.5550	0.2170	0.4110	False
152	Tanzania	3.2310	0.4760	0.8850	0.4990	0.4170	0.2760	0.1470	False
153	Afghanistan	3.2030	0.3500	0.5170	0.3610	0.0000	0.1580	0.0250	False
154	Central African Republic	3.0830	0.0260	0.0000	0.1050	0.2250	0.2350	0.0350	False
155	South Sudan	2.8530	0.3060	0.5750	0.2950	0.0100	0.2020	0.0910	False

We will decompose our data set, one of which contains attribute data and the other contains classification data set. Then we pass these two datasets to the classifier to enable it to recognize the classification relationship.

In [55]:
```
A = b.drop(['over_mean','Score','Country or region'], axis=1)
y = b['over_mean']
```

Check for variables whose happiness score exceeds the average.

In [56]:
```
b['over_mean'].value_counts()
```

Out[56]:
```
False    79
True     77
Name: over_mean, dtype: int64
```

Create a label dataset from the "over_mean" column.

In [57]:
```
from sklearn.preprocessing import LabelEncoder
```

In [58]:
```
le = LabelEncoder()
le.fit([True,False])
labels_fea = le.transform(b['over_mean'])
```

In [59]:
```
pd.Series(labels_fea).value_counts()
```

Out[59]:
```
0    79
1    77
dtype: int64
```

We use the `to_dict()` function to convert the attribute dataset to a dictionary array, then create a dictionary and check the first entry to verify the data.

In [60]:
```
from sklearn.feature_extraction import DictVectorizer
```

In [61]:
```
A_dict = A.to_dict('record')
print(A_dict[1])
```

```
{'GDP': 1.383, 'Social': 1.5730000000000002, 'Life': 0.996, 'Freedom': 0.5920000000000001, 'Generosity': 0.252, 'Corruption': 0.41}
```

Extract the matrix. The matrix `A_mat` does not contain variable names.

In [62]:
```
vec = DictVectorizer()
A_mat = vec.fit_transform(A_dict)
```

In [63]:
```
print(vec.feature_names_[0:5])
print(vec.vocabulary_)
```

```
['Corruption', 'Freedom', 'GDP', 'Generosity', 'Life']
{'GDP': 2, 'Social': 5, 'Life': 4, 'Freedom': 1, 'Generosity': 3, 'Corruption': 0}
```

We check the extended range of the data set by checking the `A_mat` matrix, and then convert it into an array and load it into the classifier.

In [64]:
```
print(A_mat.shape)
print(type(A_mat))
```

```
(156, 6)
<class 'scipy.sparse.csr.csr_matrix'>
```

In [65]:
```
A_array = A_mat.toarray()
print(A_array.shape)
print(type(A_array))
```

```
(156, 6)
<class 'numpy.ndarray'>
```

We use the `train_test_split` method and divide the data into a training set and a test set according to a **75:25** division. And check the length of the two data sets.

In [66]:
```
from sklearn.model_selection import train_test_split
```

In [67]:
```
train_d, test_d, train_lab, test_lab = train_test_split(A_array, labels_fea)
```

In [68]:
```
print(train_d.shape)
print(test_d.shape)
```

```
(117, 6)
(39, 6)
```

We use the following code to create a decision tree classifier, and then use `tree.plot_tree()` to output the structure of the Decision Tree.

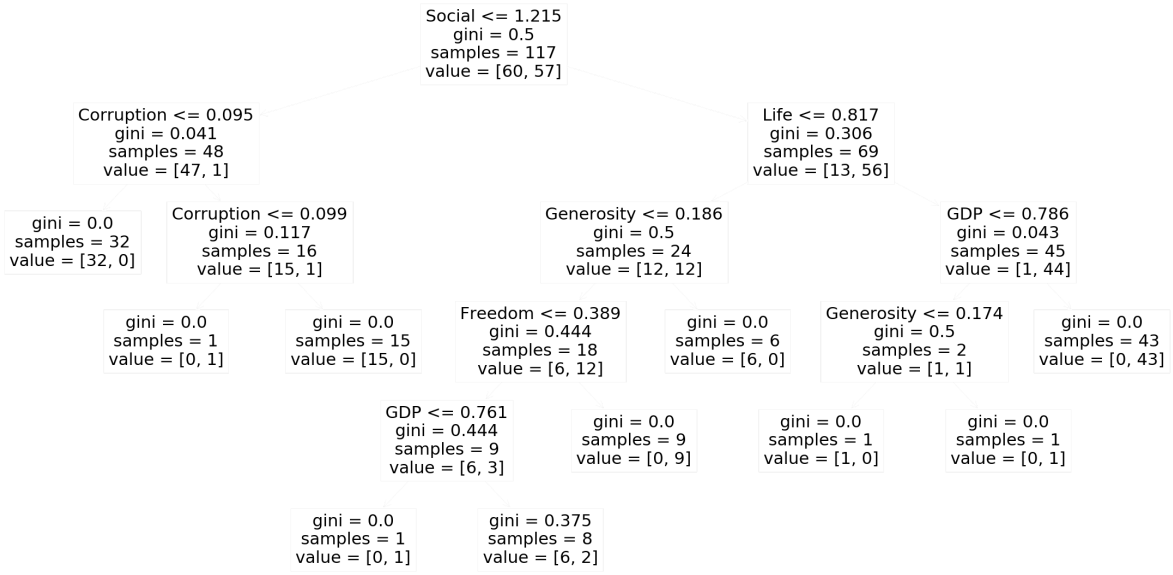
In [69]:
```
from sklearn.tree import DecisionTreeClassifier
clf_decision_tree = DecisionTreeClassifier()
clf_decision_tree.fit(train_d, train_lab)
test_pred_decision_tree = clf_decision_tree.predict(test_d)
```

In [70]:
```
print (metrics.classification_report(test_lab, test_pred_decision_tree))
```

```
              precision    recall  f1-score   support

           0       0.76      0.84      0.80        19
           1       0.83      0.75      0.79        20

    accuracy                           0.79        39
```

macro avg	0.80	0.80	0.79	39
weighted avg	0.80	0.79	0.79	39

In [71]:
```
decision_tree_depth_5 = DecisionTreeClassifier(max_depth=5)
decision_tree_depth_5.fit(train_d, train_lab)

plt.figure(figsize=(40,20))
from sklearn import tree


_ = tree.plot_tree(decision_tree_depth_5, feature_names = vec.feature_names_)

plt.show()
```



For easy viewing, the following is the text format of the decision tree.

In [72]:
```
from sklearn.tree import export_text
tree_rules = export_text(decision_tree_depth_5, feature_names=vec.feature_names_)
print(tree_rules)
```

```
|--- Social <= 1.22
|   |--- Corruption <= 0.10
|   |   |--- class: 0
|   |--- Corruption >  0.10
|   |   |--- Corruption <= 0.10
|   |   |   |--- class: 1
|   |   |--- Corruption >  0.10
|   |   |   |--- class: 0
|--- Social >  1.22
|   |--- Life <= 0.82
|   |   |--- Generosity <= 0.19
|   |   |   |--- Freedom <= 0.39
|   |   |   |   |--- GDP <= 0.76
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- GDP >  0.76
|   |   |   |   |   |--- class: 0
|   |   |   |--- Freedom >  0.39
|   |   |   |   |--- class: 1
|   |   |--- Generosity >  0.19
|   |   |   |--- class: 0
|   |--- Life >  0.82
|   |   |--- GDP <= 0.79
|   |   |   |--- Generosity <= 0.17
|   |   |   |   |--- class: 0
|   |   |   |--- Generosity >  0.17
|   |   |   |   |--- class: 1
|   |   |--- GDP >  0.79
```

```
|   |   |   |--- class: 1
```

Results

We compare the multiple linear regression results obtained by the two methods, we can see that the two methods have the same degree of fit. Although the "VIF" method has more independent variables of Generosity and Perceptions of corruption, these two independent variables are non-significant and have no significant effect on the dependent variable. However, the order of the independent variable coefficients under the two methods is indeed different. In the stepwise regression method, the independent variables are arranged in order from the largest to the smallest: Freedom to make life choices, Healthy life expectancy, Social support, GDP per capita; and the order in the "VIF" method is: Freedom to make life choices, Social support, Healthy life expectancy, GDP per capita.

This result indicates that Freedom to make life choices has the greatest impact on happiness scores, while GDP per capita has the least impact on happiness scores. Taking the stepwise regression method as an example, when the other variables remain unchanged, the happiness score increases by 1.8458 for each additional unit of "Freedom to make life choices"; the happiness score increases by 1.1414 for each additional unit of "Healthy life expectancy"; "Social support" for each additional unit, the happiness score will increase by 1.0166; "GDP per capita" for each additional unit, the happiness score will increase by 0.8105. Meanwhile, the intercept of this model is 1.8921. When all the independent variables are 0, the happiness score of this world is still positive, which shows that people can still feel happiness in this world without any external interference.

Limitations to the model

We did not check whether each variable has outliers before analyzing this set of data. Due to the existence of outliers, it may lead to the inaccurate fitting of the multiple linear regression model and errors. We can check whether each variable has abnormal values by drawing box plots, and then go directly to delete the abnormal values and other methods to process the data. Moreover, our regression model must also meet a series of important assumptions (we have considered multicollinearity between variables):

1. There are linear relationships between the dependent and independent variables.
2. The residuals in the model should be normally distributed.
3. Homoscedasticity.
4. Independence of errors.

If our model handles outliers and satisfies this series of important assumptions, our model may be improved.

Conclusions

Our analysis shows that each significant independent variable is directly proportional to the happiness score. When the other independent variables remain unchanged, the score of each independent variable increase, the happiness score will increase; the score of each independent variable decrease, the happiness score will also decrease. Among them, the independent variable "Freedom to make life choices" has the greatest influence on the happiness score.

Moreover, we can see that people pay more attention to their lives, and a healthy and free lifestyle is very important to them. Therefore, while developing science and technology and the economy, the government should pay more attention to people's livelihood. If the government wants to improve the happiness score of their own country, they should encourage people to choose life freely, and give some targeted policies in this regard.

In [73]:
```
end_time = time.time()
run_time = end_time - start_time
print("Total run time: ", run_time, "seconds")
```

```
Total run time:  7.042267799377441 seconds
```

(2125 words)

References

World Happiness Report 2019, online available at: here, retrieved on May 3, 2020

Julie, L., (2012). The Most Influential Factors in Determining the Happiness of Nations, online available at: here, retrieved on May 3, 2020

Stephanie, (2015). Multicollinearity: Definition, Causes, Examples, online available at: here, retrieved on May 3, 2020

Jim, C., (2018). Variance Inflation Factor, online available at: here, retrieved on May 3, 2020