WORLD HAPPINESS REPORT

The World Happiness Report is a landmark survey of the state of global happiness. The first report was published in 2012, the second in 2013, the third in 2015, and the fourth in the 2016 Update. The World Happiness 2017, which ranks 155 countries by their happiness levels, was released at the United Nations at an event celebrating International Day of Happiness on March 20th. The report continues to gain global recognition as governments, organizations and civil society increasingly use happiness indicators to inform their policy-making decisions. Leading experts across fields – economics, psychology, survey analysis, national statistics, health, public policy and more – describe how measurements of well-being can be used effectively to assess the progress of nations. The reports review the state of happiness in the world today and show how the new science of happiness explains personal and national variations in happiness.

Dystopia

Dystopia is an imaginary country that has the world's least-happy people. The purpose in establishing Dystopia is to have a benchmark against which all countries can be favorably compared (no country performs more poorly than Dystopia) in terms of each of the six key variables, thus allowing each sub-bar to be of positive width. The lowest scores observed for the six key variables, therefore, characterize Dystopia. Since life would be very unpleasant in a country with the world's lowest incomes, lowest life expectancy, lowest generosity, most corruption, least freedom and least social support, it is referred to as "Dystopia," in contrast to Utopia

Residuals

The residuals, or unexplained components, differ for each country, reflecting the extent to which the six variables either over- or under-explain average life evaluations. These residuals have an average value of approximately zero over the whole set of countries.

Columns Contributing to the Happiness score

The following columns: GDP per Capita, Family, Life Expectancy, Freedom, Generosity, Trust Government Corruption describe the extent to which these factors contribute in evaluating the happiness in each country. The Dystopia Residual metric actually is the Dystopia Happiness Score(1.85) + the Residual value or the unexplained value for each country

IMPORT LIBRARIES

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import zscore
import sklearn
```

LOAD DATASET

In [3]:
```
sv('Happiness_score.csv')
```

```
df
```

Out[3]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Tr (Governm Corrupti
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557	0.419
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62877	0.141
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64938	0.483
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.66973	0.365
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.63297	0.329
...	...	...	...	...	...	...	...	...	...	...
153	Rwanda	Sub-Saharan Africa	154	3.465	0.03464	0.22208	0.77370	0.42864	0.59201	0.551
154	Benin	Sub-Saharan Africa	155	3.340	0.03656	0.28665	0.35386	0.31910	0.48450	0.080
155	Syria	Middle East and Northern Africa	156	3.006	0.05015	0.66320	0.47489	0.72193	0.15684	0.189
156	Burundi	Sub-Saharan Africa	157	2.905	0.08658	0.01530	0.41587	0.22396	0.11850	0.100
157	Togo	Sub-Saharan Africa	158	2.839	0.06727	0.20868	0.13995	0.28443	0.36453	0.107

158 rows × 12 columns

In [4]:
```
df.columns
```

Out[4]:
```
Index(['Country', 'Region', 'Happiness Rank', 'Happiness Score',
       'Standard Error', 'Economy (GDP per Capita)', 'Family',
       'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
       'Generosity', 'Dystopia Residual'],
      dtype='object')
```

About THe Columns

1. Country:

It represents the country in which most people are happy depending on the key factors.

1. Region:

It shows the region to which the country belongs to

1. Happiness Rank:

It shows the rank of each countrywith respect to happiness score in ascending order

1. Happiness Score

It represents the sum of values of all the factors contributing to the happiness of a country.

1. Standard Error:

It shows the minimum error value each country can have. It differs for each country depending on key factors values.

The Key Factors Contributing to Happiness

1. Economy (GDP per Capita)

It shows the country's economic range per person

1. Family

This column shows howmuch family contributes to the happiness score

1. Health (Life Expectancy)

It shows the average life with good health and required medical facilities

1. Freedom

It shows the freedom factor which the people of different countries enjoy it to different levels

1. Trust (Government Corruption)

It represents the corruption level of a country's Government

1. Generosity

It shows the kindness level enjoyed by people of different countries

Dystopia Residual

Dystopia happiness score (1.85) i.e. the score of a hypothetical country that has a lower rank than the lowest ranking country

Exploratory Data Analysis

Checking for null values and removing it

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
Country                          0
Region                           0
Happiness Rank                   0
Happiness Score                  0
Standard Error                   0
Economy (GDP per Capita)         0
Family                           0
Health (Life Expectancy)         0
Freedom                          0
Trust (Government Corruption)    0
                                 0
```

```
Dystopia Residual                0
dtype: int64
```

The Dataset doesn't have any null values.

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 158 entries, 0 to 157
Data columns (total 12 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   Country                        158 non-null    object 
 1   Region                         158 non-null    object 
 2   Happiness Rank                 158 non-null    int64  
 3   Happiness Score                158 non-null    float64
 4   Standard Error                 158 non-null    float64
 5   Economy (GDP per Capita)       158 non-null    float64
 6   Family                         158 non-null    float64
 7   Health (Life Expectancy)       158 non-null    float64
 8   Freedom                        158 non-null    float64
 9   Trust (Government Corruption)  158 non-null    float64
 10  Generosity                     158 non-null    float64
 11  Dystopia Residual              158 non-null    float64
dtypes: float64(9), int64(1), object(2)
memory usage: 14.9+ KB
```

In [7]:
```
df.shape
```

Out[7]:
```
(158, 12)
```

Since there are 158 unique countries, we are checking for uniqueness of the region to which the countries belong to

In [8]:
```
df['Region'].unique()
```

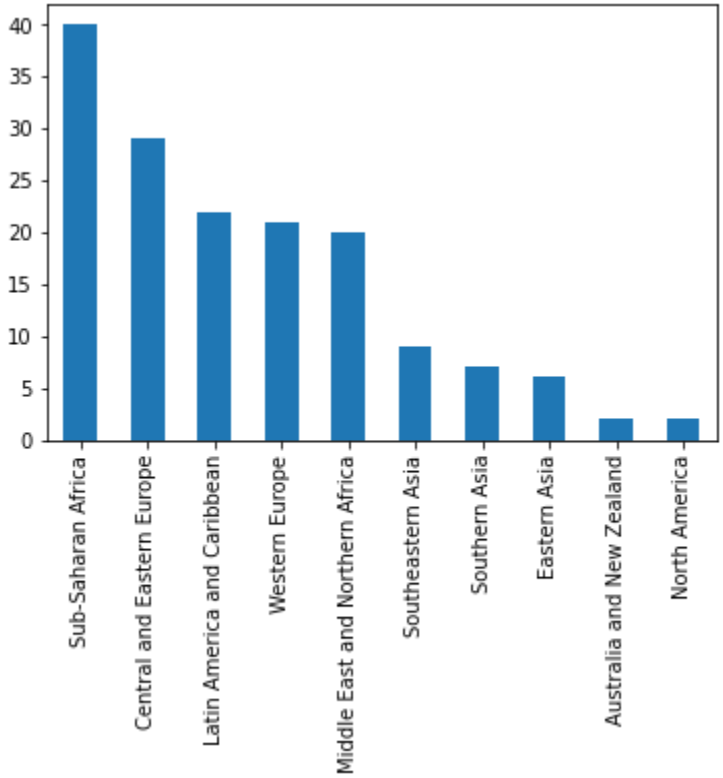
Out[8]:
```
array(['Western Europe', 'North America', 'Australia and New Zealand',
       'Middle East and Northern Africa', 'Latin America and Caribbean',
       'Southeastern Asia', 'Central and Eastern Europe', 'Eastern Asia',
       'Sub-Saharan Africa', 'Southern Asia'], dtype=object)
```

In [9]:
```
x=df['Region'].value_counts()
x
```

Out[9]:
```
Sub-Saharan Africa                 40
Central and Eastern Europe         29
Latin America and Caribbean        22
Western Europe                     21
Middle East and Northern Africa    20
Southeastern Asia                   9
Southern Asia                       7
Eastern Asia                        6
Australia and New Zealand           2
North America                       2
Name: Region, dtype: int64
```

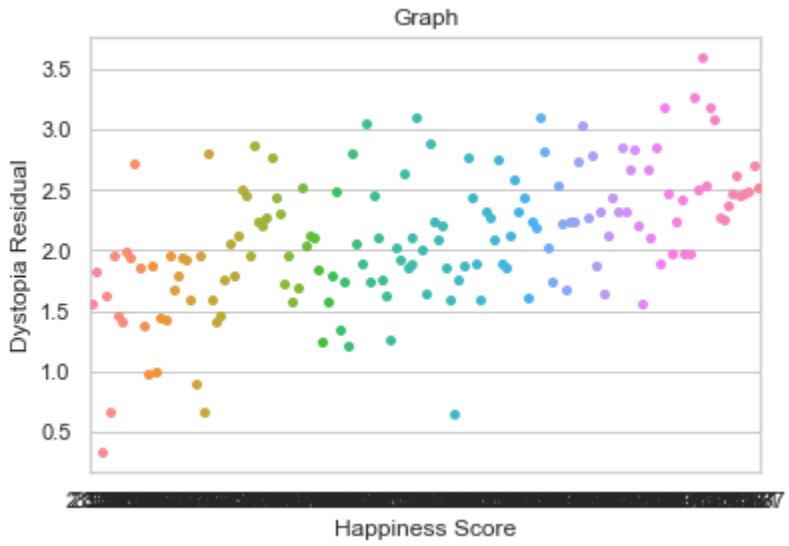
In [10]:
```
x.plot(kind='bar')
```

Out[10]:
```
<AxesSubplot:>
```



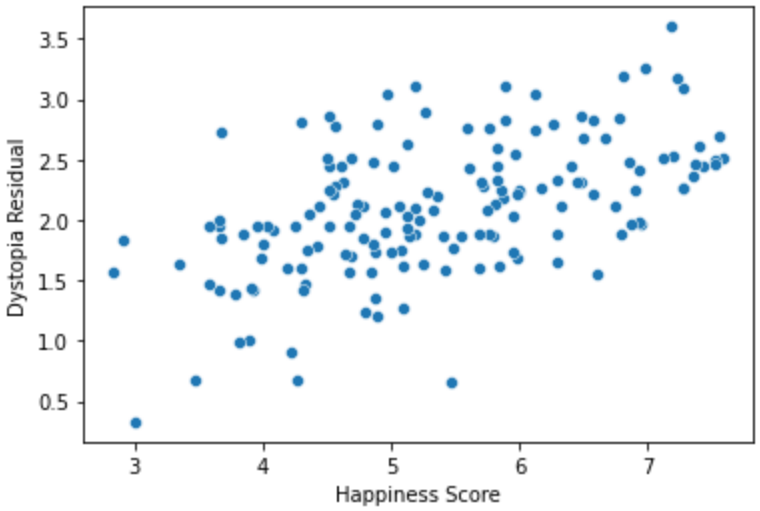
From the above plot, its clear that the SUB-SAHARAN AFRICA region has the most happiest countries and North American region has less number of happiest countries.

In [10]:
```
sns.set(style='whitegrid')
b=sns.stripplot(x='Happiness Score', y='Dystopia Residual', data=df)
plt.title('Graph')
plt.show()
```



In [11]:
```
sns.scatterplot(x='Happiness Score', y='Dystopia Residual', data = df)
```

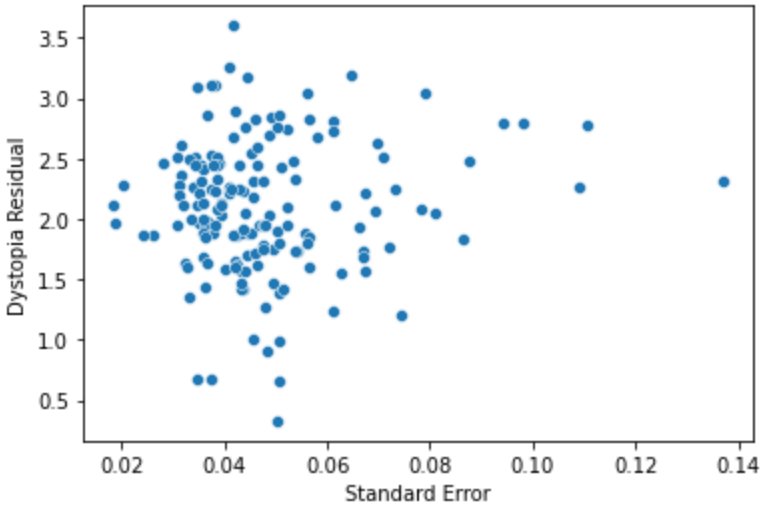
Out[11]:
```
<AxesSubplot:xlabel='Happiness Score', ylabel='Dystopia Residual'>
```



Happiness Score is mostly distributed in the range of 4 to 5.5

In [12]:
```
sns.scatterplot(x='Standard Error', y='Dystopia Residual', data = df)
```

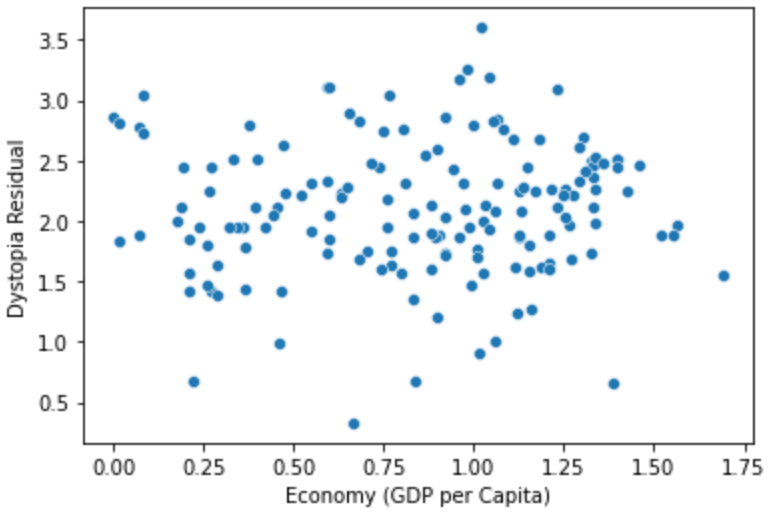
Out[12]:
```
<AxesSubplot:xlabel='Standard Error', ylabel='Dystopia Residual'>
```



Standard Error is mostly distributed in the range of 0.03 to 0.05

In [13]:
```
sns.scatterplot(x='Economy (GDP per Capita)', y='Dystopia Residual', data = df)
```

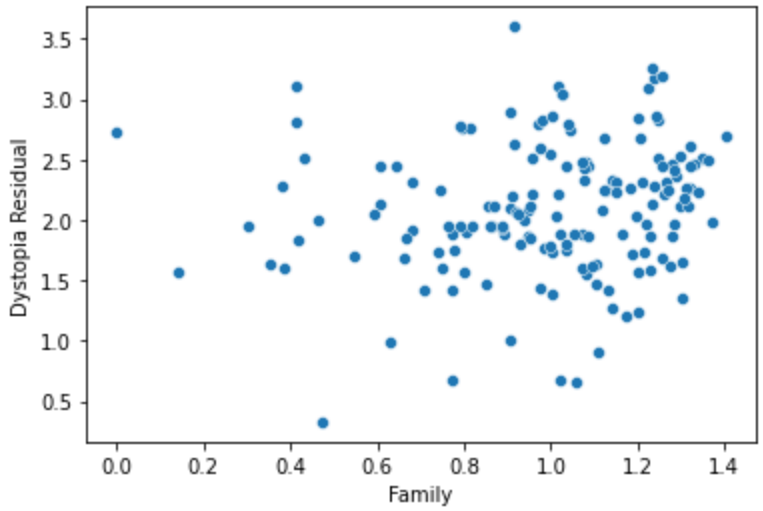
Out[13]:
```
<AxesSubplot:xlabel='Economy (GDP per Capita)', ylabel='Dystopia Residual'>
```



Economy (GDP per Capita)is mostly distributed in the range of 0.25 to 1.25

In [14]:
```
sns.scatterplot(x='Family', y='Dystopia Residual', data = df)
```

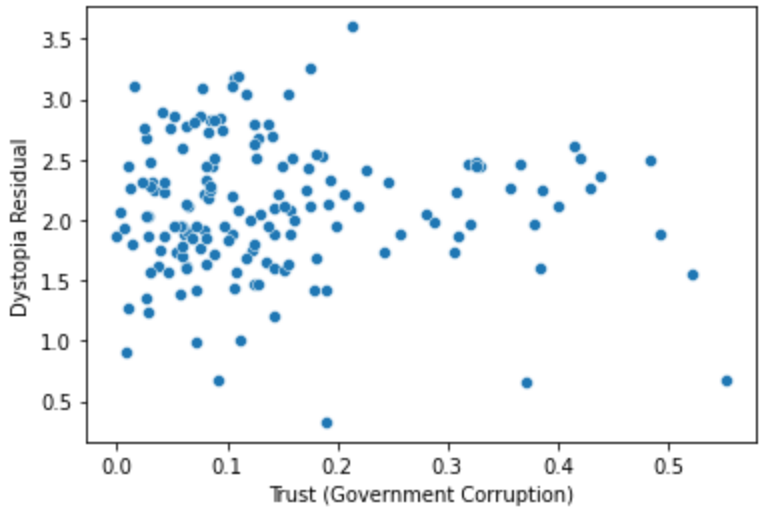
Out[14]:
```
<AxesSubplot:xlabel='Family', ylabel='Dystopia Residual'>
```



Family is mostly distributed in the range of 0.8 to 1.3

In [15]:
```
sns.scatterplot(x='Trust (Government Corruption)', y='Dystopia Residual', data = df)
```

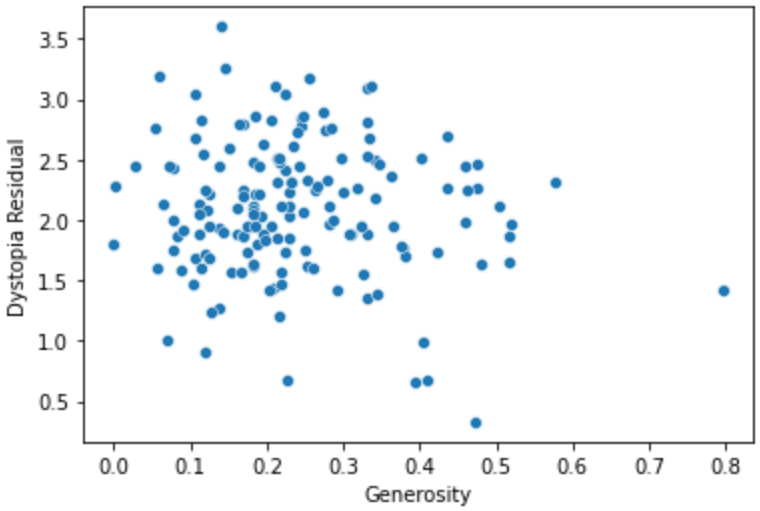
Out[15]:
```
<AxesSubplot:xlabel='Trust (Government Corruption)', ylabel='Dystopia Residual'>
```



Government corruption is mostly distributed in the range of 0 to 0.15

In [16]:
```
sns.scatterplot(x='Generosity', y='Dystopia Residual', data = df)
```

Out[16]:
```
<AxesSubplot:xlabel='Generosity', ylabel='Dystopia Residual'>
```



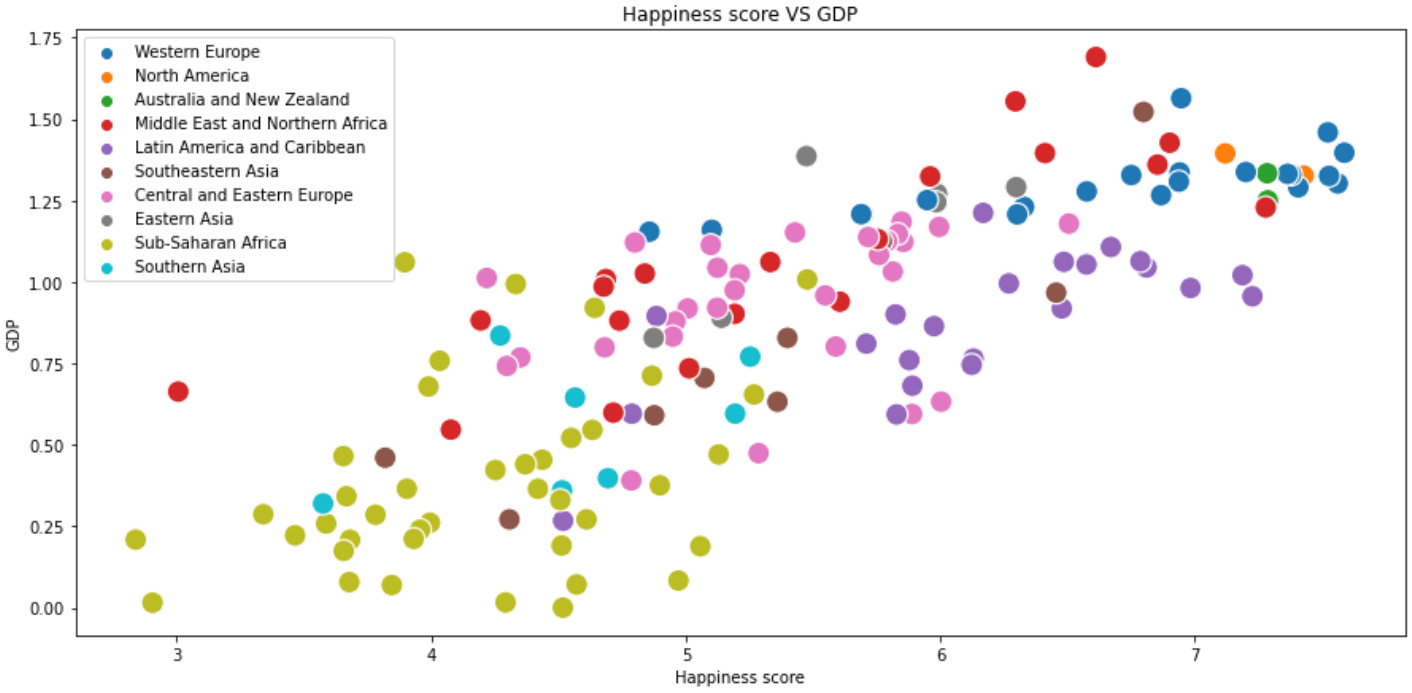
Generosity is mostly distributed in the range of 0.1 to 0.35

ECONOMY AND HAPPINESS SCORE RELATION

In [17]:

```
plt.figure(figsize=(15,7))
sns.scatterplot(x=df['Happiness Score'] , y= df['Economy (GDP per Capita)'], hue= df['Regi
plt.title("Happiness score VS GDP")

plt.legend(loc= 'upper left', fontsize='10')
plt.xlabel('Happiness score')
plt.ylabel('GDP')
plt.show()
```



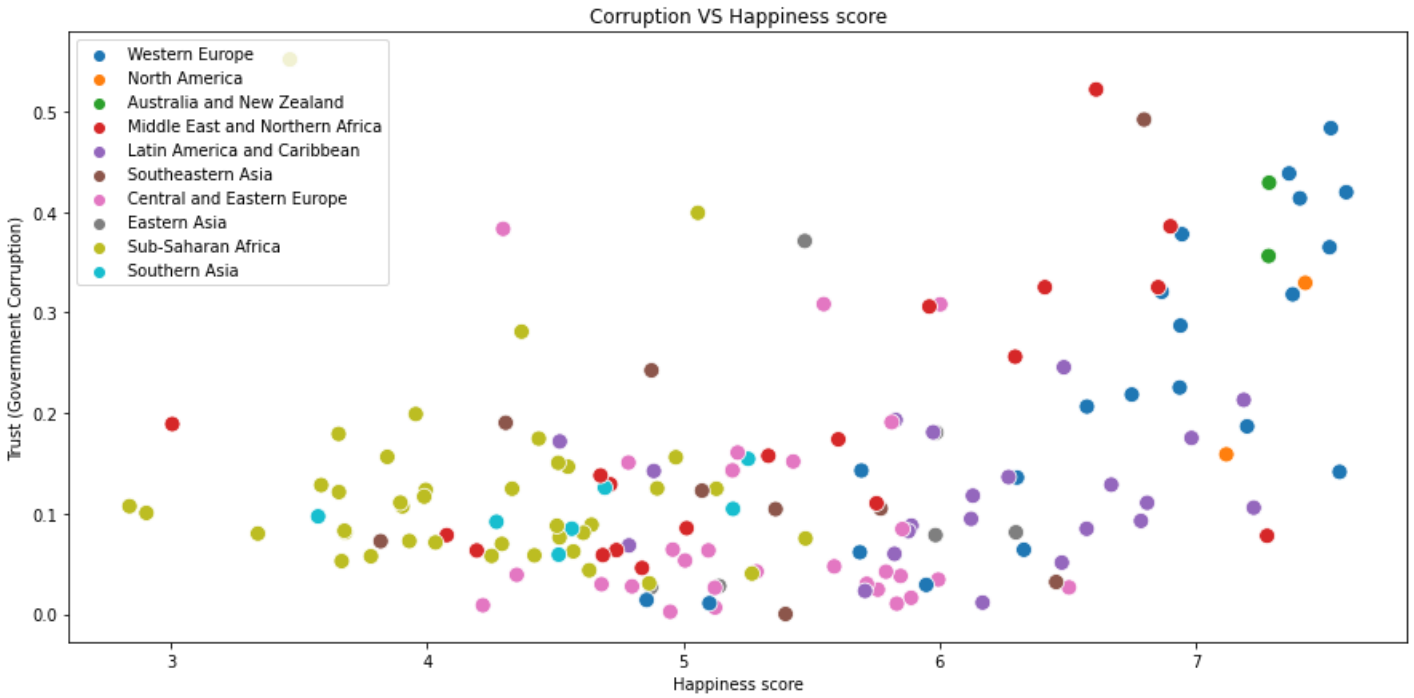
The Above plot shows that Happiness score is high in Western Europian Countries and North American with the increase in GDP.

Sub-Saharan African Countries have low happiness score corresponding to GDP

In [18]:

```
plt.figure(figsize=(15,7))
sns.scatterplot(y=df['Trust (Government Corruption)'] , x= df['Happiness Score'], hue= df[
plt.title("Corruption VS Happiness score")
```

```
plt.legend(loc= 'upper left', fontsize='10')
plt.ylabel('Trust (Government Corruption)')
plt.xlabel('Happiness score')
plt.show()
```



THe corruptions and Happiness score have less correlation.

Latin American and caribean countries and some Central and Eastern Europian countries have registered high happiness score with less corruptions.

Sub-Saharan african Countries have low happiness score.

In [ ]:

In [ ]:

In [ ]:

CORELLATION

In [19]:
```
df.corr()
```

Out[19]:

	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Gener
Happiness Rank	1.000000	-0.992105	0.158516	-0.785267	-0.733644	-0.735613	-0.556886	-0.372315	-0.16
Happiness Score	-0.992105	1.000000	-0.177254	0.780966	0.740605	0.724200	0.568211	0.395199	0.18
Standard Error	0.158516	-0.177254	1.000000	-0.217651	-0.120728	-0.310287	-0.129773	-0.178325	-0.08

	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Gener
Economy (GDP per Capita)	-0.785267	0.780966	-0.217651	1.000000	0.645299	0.816478	0.370300	0.307885	-0.01
Family	-0.733644	0.740605	-0.120728	0.645299	1.000000	0.531104	0.441518	0.205605	0.08
Health (Life Expectancy)	-0.735613	0.724200	-0.310287	0.816478	0.531104	1.000000	0.360477	0.248335	0.10
Freedom	-0.556886	0.568211	-0.129773	0.370300	0.441518	0.360477	1.000000	0.493524	0.37
Trust (Government Corruption)	-0.372315	0.395199	-0.178325	0.307885	0.205605	0.248335	0.493524	1.000000	0.27
Generosity	-0.160142	0.180319	-0.088439	-0.010465	0.087513	0.108335	0.373916	0.276123	1.00
Dystopia Residual	-0.521999	0.530474	0.083981	0.040059	0.148117	0.018979	0.062783	-0.033105	-0.10

We see that Happiness Rank doesn't contribute much to the dystopia residual. So we will drop this column

In [20]:
```
df.drop('Happiness Rank', axis=1, inplace=True)
```

In [21]:
```
df
```

Out[21]:

	Country	Region	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Genero
0	Switzerland	Western Europe	7.587	0.03411	1.39651	1.34951	0.94143	0.66557	0.41978	0.29
1	Iceland	Western Europe	7.561	0.04884	1.30232	1.40223	0.94784	0.62877	0.14145	0.43
2	Denmark	Western Europe	7.527	0.03328	1.32548	1.36058	0.87464	0.64938	0.48357	0.34
3	Norway	Western Europe	7.522	0.03880	1.45900	1.33095	0.88521	0.66973	0.36503	0.34
4	Canada	North America	7.427	0.03553	1.32629	1.32261	0.90563	0.63297	0.32957	0.45
...	...	...	...	...	...	...	...	...	...	...
153	Rwanda	Sub-Saharan Africa	3.465	0.03464	0.22208	0.77370	0.42864	0.59201	0.55191	0.22
154	Benin	Sub-Saharan Africa	3.340	0.03656	0.28665	0.35386	0.31910	0.48450	0.08010	0.18
155	Syria	Middle East and Northern Africa	3.006	0.05015	0.66320	0.47489	0.72193	0.15684	0.18906	0.47
156	Burundi	Sub-Saharan Africa	2.905	0.08658	0.01530	0.41587	0.22396	0.11850	0.10062	0.19
157	Togo	Sub-Saharan Africa	2.839	0.06727	0.20868	0.13995	0.28443	0.36453	0.10731	0.16

158 rows × 11 columns

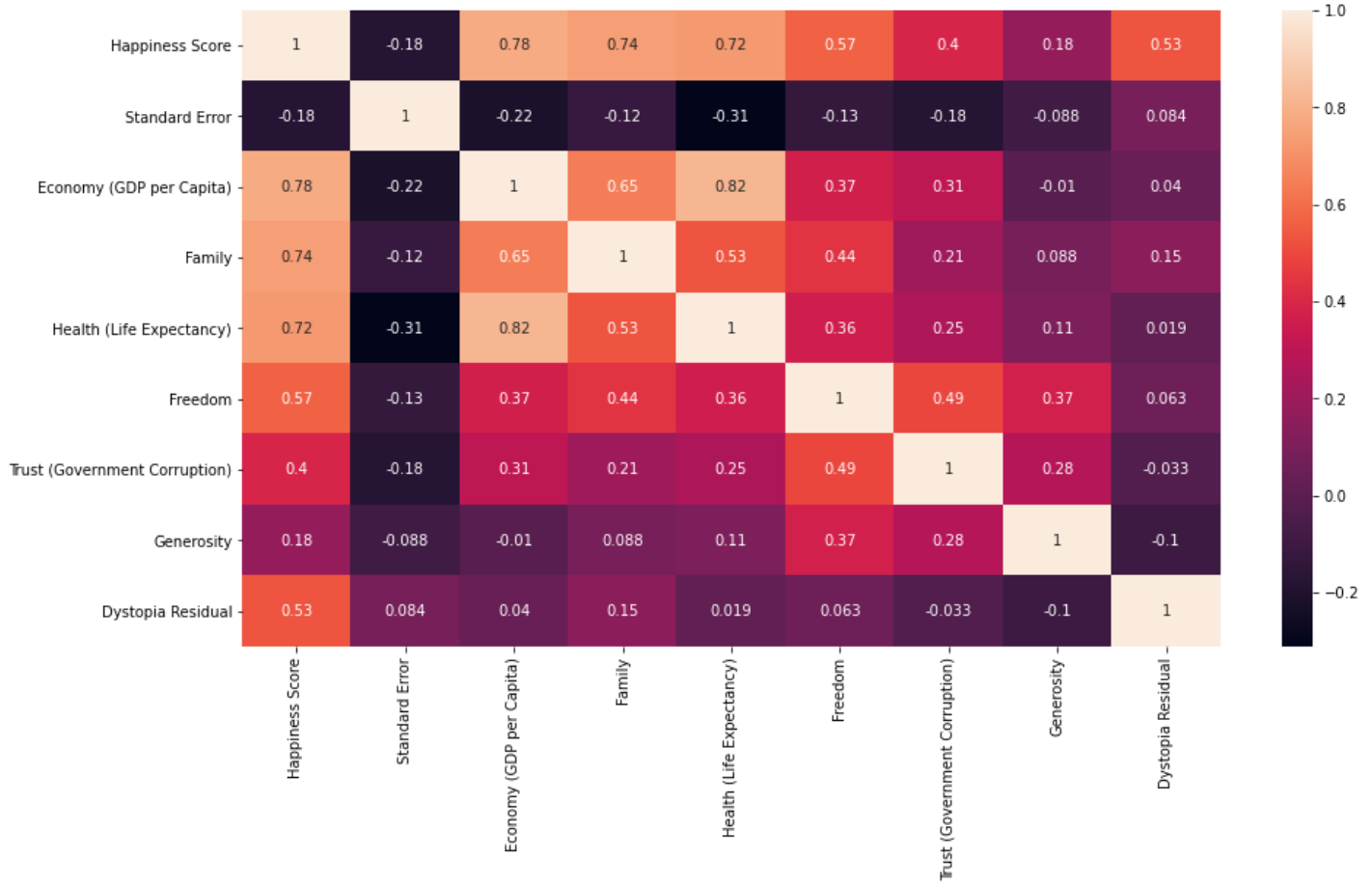
In [22]:
```
df.corr()
```

Out[22]:

	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dyst Resi
Happiness Score	1.000000	-0.177254	0.780966	0.740605	0.724200	0.568211	0.395199	0.180319	0.530
Standard Error	-0.177254	1.000000	-0.217651	-0.120728	-0.310287	-0.129773	-0.178325	-0.088439	0.083
Economy (GDP per Capita)	0.780966	-0.217651	1.000000	0.645299	0.816478	0.370300	0.307885	-0.010465	0.040
Family	0.740605	-0.120728	0.645299	1.000000	0.531104	0.441518	0.205605	0.087513	0.148
Health (Life Expectancy)	0.724200	-0.310287	0.816478	0.531104	1.000000	0.360477	0.248335	0.108335	0.018
Freedom	0.568211	-0.129773	0.370300	0.441518	0.360477	1.000000	0.493524	0.373916	0.062
Trust (Government Corruption)	0.395199	-0.178325	0.307885	0.205605	0.248335	0.493524	1.000000	0.276123	-0.033
Generosity	0.180319	-0.088439	-0.010465	0.087513	0.108335	0.373916	0.276123	1.000000	-0.101
Dystopia Residual	0.530474	0.083981	0.040059	0.148117	0.018979	0.062783	-0.033105	-0.101301	1.000

In [23]:
```
plt.figure(figsize=(15,8))
sns.heatmap(df.corr(), annot=True)
```

Out[23]:
```
<AxesSubplot:>
```



OutCome Of Correlation

Happiness score is highly correlated to Dystopia residual.

Happiness score is also highly correlated to Economy, Family, Health.

Economy has a good correlation with Health and Family

Corruption and Generosity is negatively correlated to Dystropia

In [54]:
```
#
```

#

Describe Dataset

In [24]:
```
df.describe()
```

Out[24]:

	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dys Res
count	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.0
mean	5.375734	0.047885	0.846137	0.991046	0.630259	0.428615	0.143422	0.237296	2.09
std	1.145010	0.017146	0.403121	0.272369	0.247078	0.150693	0.120034	0.126685	0.5
min	2.839000	0.018480	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.3
25%	4.526000	0.037268	0.545808	0.856823	0.439185	0.328330	0.061675	0.150553	1.7
50%	5.232500	0.043940	0.910245	1.029510	0.696705	0.435515	0.107220	0.216130	2.0

	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dys Res
75%	6.243750	0.052300	1.158448	1.214405	0.811013	0.549092	0.180255	0.309883	2.46
max	7.587000	0.136930	1.690420	1.402230	1.025250	0.669730	0.551910	0.795880	3.60

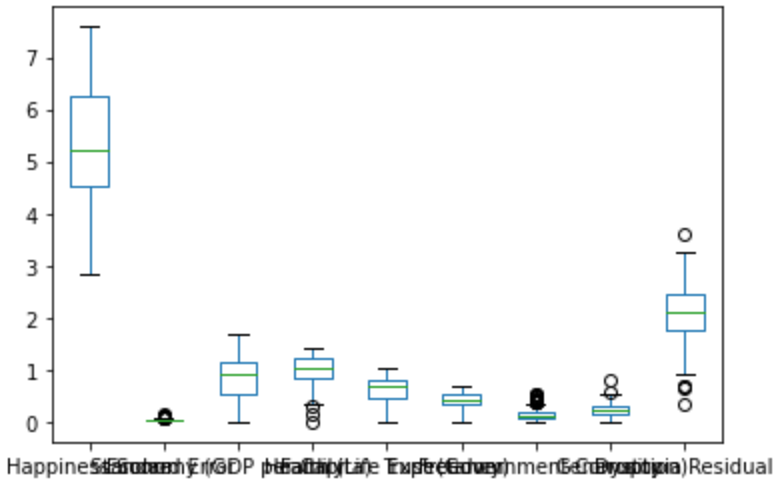
All the columns have mean to be nearly equal to the median. So the data is not much skewed.

There may be chances of outliers in Happiness Score, Economy,Generosity.

Checking For Outliers

In [25]:
```
plt.figure(figsize=(20,20))
df.plot.box()
```

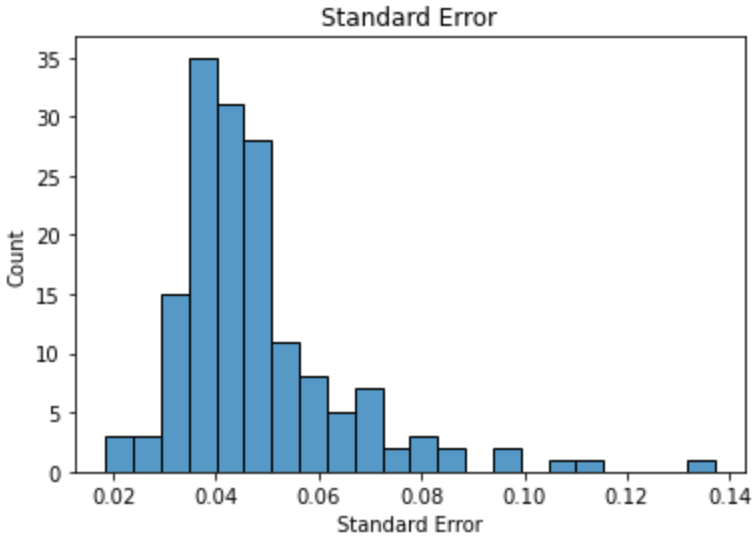
Out[25]:
```
<AxesSubplot:>
<Figure size 1440x1440 with 0 Axes>
```

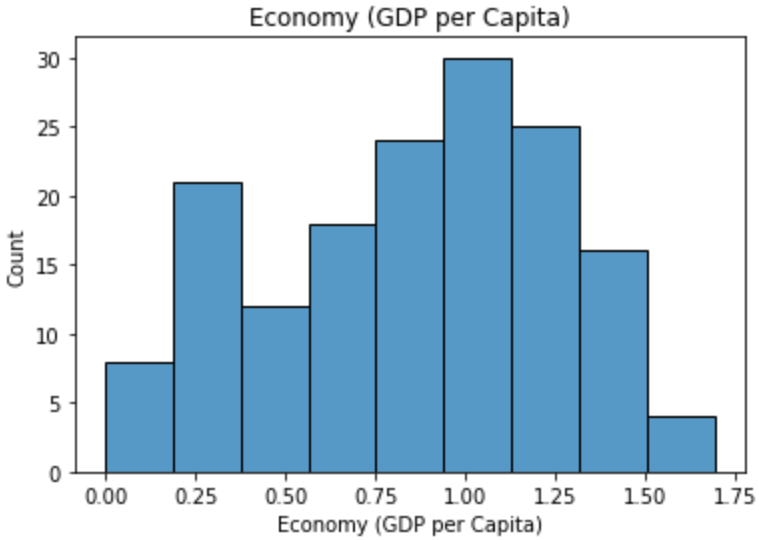


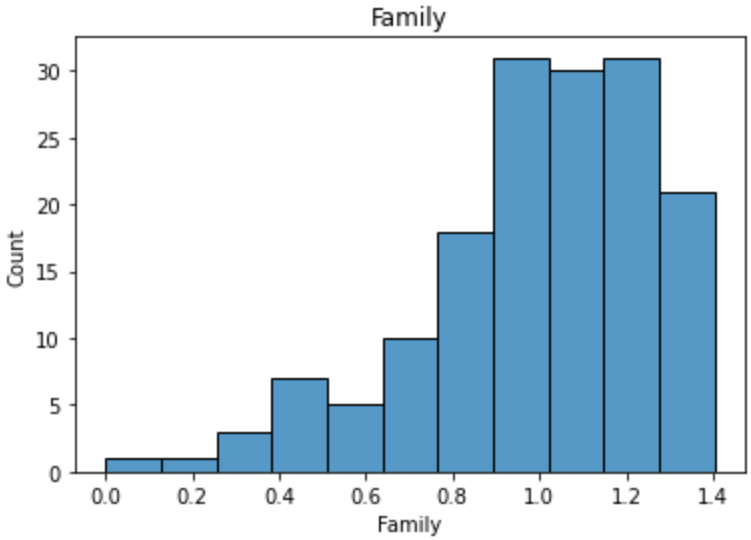
This shows that most columns have little outliers

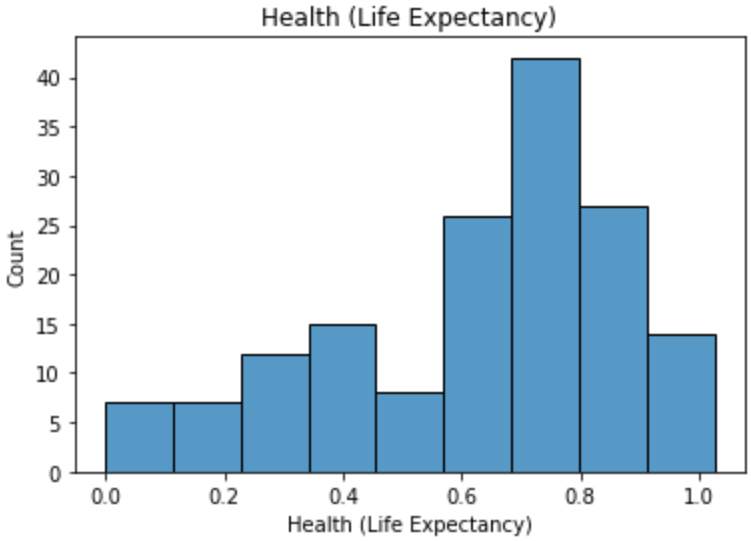
In [26]:
```
m= df.columns[3:]
```

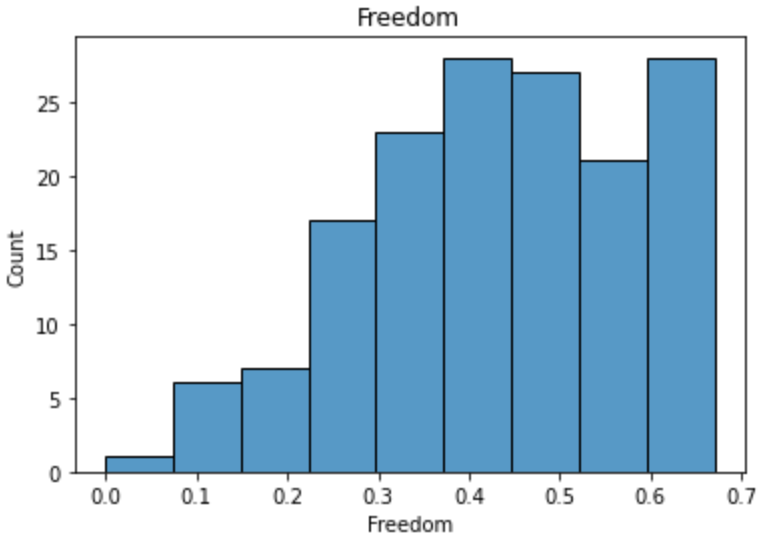
In [27]:
```
for i in m:
    sns.histplot(df[i])
    plt.title(i)
    plt.show()
    plt.tight_layout()
```

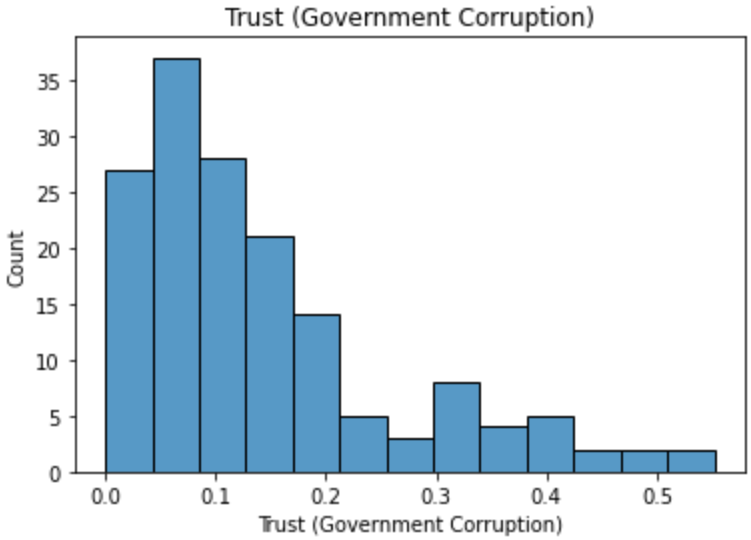


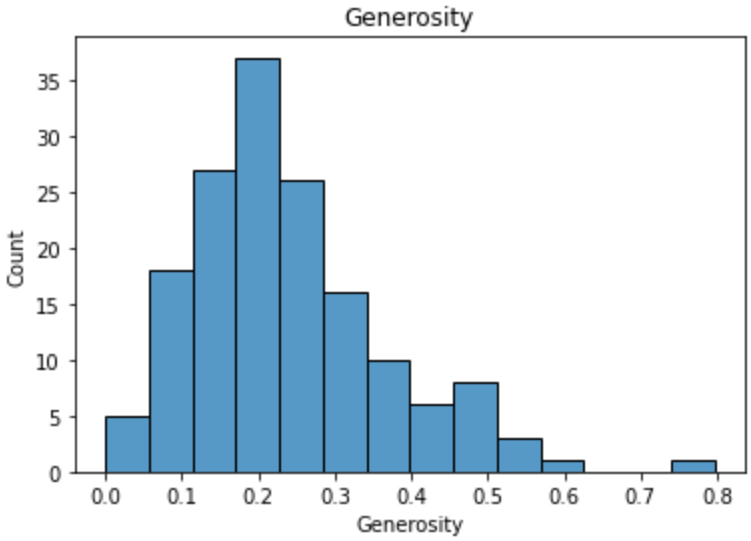


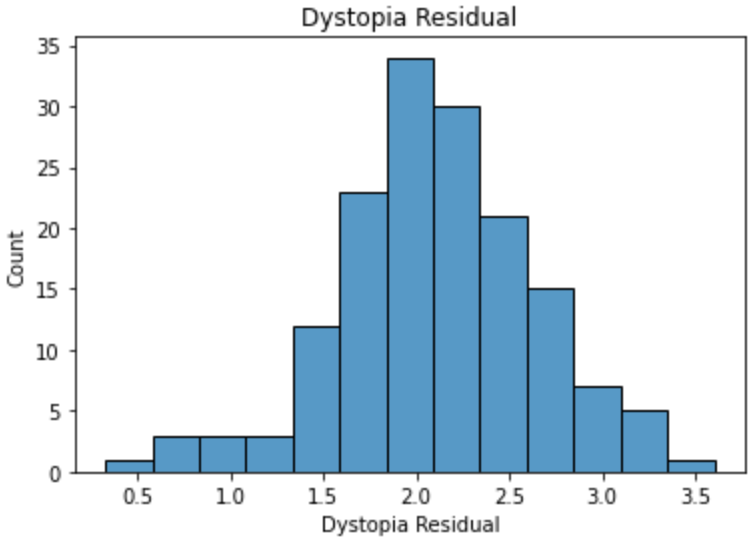













```
<Figure size 432x288 with 0 Axes>
```

The above graphs shows that there are some skewness present in Generosity, Standard error and life expectancy

Removing Outliers

In [28]:
```
df1= df.iloc[:,3:12]
```

In [29]:
```
from scipy.stats import zscore
z=np.abs(zscore(df1))
z
```

Out[29]:
```
array([[0.80592569, 1.36962124, 1.32028142, ..., 2.30965159, 0.47103971,
        0.75825809],
       [0.05588945, 1.13522625, 1.51445776, ..., 0.01647953, 1.57585637,
        1.09285682],
       [0.8544869 , 1.19286069, 1.36105403, ..., 2.8427738 , 0.8242928 ,
        0.71233526],
       ...,
       [0.13253425, 0.45524543, 1.90108634, ..., 0.38141902, 1.85689094,
        3.20843049],
       [2.26396166, 2.06756644, 2.1184666 , ..., 0.35771452, 0.31694987,
        0.48198451],
       [1.13418227, 1.58633379, 3.13472485, ..., 0.30180313, 0.5581534 ,
        0.96361241]])
```

In [30]:
```
# threshold for zscore is 3........., zscore greater than 3 is outliers
threshold =3
print(np.where(z>3))
```

```
(array([ 27,  40,  64, 115, 128, 147, 153, 155, 157], dtype=int64), array([5, 0, 0, 0, 6,
2, 5, 7, 2], dtype=int64))
```

In [31]:
```
z[40][0]
```

Out[31]:
```
3.5727739331415806
```

In [32]:
```
df_new= df1[(z<3).all(axis=1)]
```

In [33]:
```
df.shape
```

Out[33]:
```
(158, 11)
```

In [34]:
```
df_new.shape
```

Out[34]:
```
(149, 8)
```

Percentage loss

In [35]:
```
loss_percent= (158-149)/ 158*100
loss_percent
```

Out[35]:
```
5.69620253164557
```

In [52]:
```
x= df.iloc[:,4:]
y= df.iloc[:,3]
```

Transforming data to remove skewness

In [53]:
```
from sklearn.preprocessing import power_transform
x= power_transform(x,method='yeo-johnson')
```

In [54]:
```
x
```

Out[54]:
```
array([[ 1.44606101,  1.66920633,  1.4917645 , ...,  1.77399061,
         0.62239051,  0.75099154],
       [ 1.17332111,  2.01213244,  1.53234847, ...,  0.31599326,
         1.48099498,  1.11001108],
       [ 1.23983557,  1.73958573,  1.08522306, ...,  1.90679207,
         0.92797276,  0.70227525],
       ...,
       [-0.5134688 , -1.69066357,  0.26293312, ...,  0.73891461,
         1.65933595, -2.86621557],
       [-1.89495386, -1.79680304, -1.52569971, ..., -0.15194624,
        -0.19482942, -0.51480136],
       [-1.52122584, -2.16039658, -1.37181091, ..., -0.06732623,
        -0.49041465, -0.97664547]])
```

In [55]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
```

```
x= sc.fit_transform(x)
x
```

Out[55]:
```
array([[ 1.44606101,  1.66920633,  1.4917645 , ...,  1.77399061,
         0.62239051,  0.75099154],
       [ 1.17332111,  2.01213244,  1.53234847, ...,  0.31599326,
         1.48099498,  1.11001108],
       [ 1.23983557,  1.73958573,  1.08522306, ...,  1.90679207,
         0.92797276,  0.70227525],
       ...,
       [-0.5134688 , -1.69066357,  0.26293312, ...,  0.73891461,
         1.65933595, -2.86621557],
       [-1.89495386, -1.79680304, -1.52569971, ..., -0.15194624,
        -0.19482942, -0.51480136],
       [-1.52122584, -2.16039658, -1.37181091, ..., -0.06732623,
        -0.49041465, -0.97664547]])
```

In [56]:
```
x.shape
```

Out[56]:
```
(158, 7)
```

In [57]:
```
#
```

In [58]:
```
y.shape
```

Out[58]:
```
(158,)
```

Target variable Happiness score is a continous type variable, therefore regression models are used

Model Selection

In [60]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
lr= LinearRegression()
from sklearn.metrics import mean_squared_error, mean_absolute_error,r2_score

import warnings
warnings.filterwarnings('ignore')
```

In [61]:
```
for i in range(0,100):
    xtrain,xtest,ytrain,ytest= train_test_split(x,y,test_size=0.20, random_state=i)
    lr.fit(xtrain,ytrain)
    pred_tr= lr.predict(xtrain)
    pred_ts= lr.predict(xtest)
    print(f"At random state {i}, the training accuracy is:- {r2_score(ytrain,pred_tr)}" )
    print(f"At randon state {i}, the testing accuracy is:- {r2_score(ytest,pred_ts)}")
    print("\n")
```

```
At random state 0, the training accuracy is:- 0.1394728866098277
At randon state 0, the testing accuracy is:- 0.004542962948915319


At random state 1, the training accuracy is:- 0.11003527473439989
At randon state 1, the testing accuracy is:- 0.13896092554199047


At random state 2, the training accuracy is:- 0.11882646254147189
```

```
At randon state 2, the testing accuracy is:- 0.1553279384886408


At random state 3, the training accuracy is:- 0.1032257067568586
At randon state 3, the testing accuracy is:- 0.06733394722556063


At random state 4, the training accuracy is:- 0.13355462950557517
At randon state 4, the testing accuracy is:- 0.08269500232467242


At random state 5, the training accuracy is:- 0.15963955280584852
At randon state 5, the testing accuracy is:- -0.012586185428631325


At random state 6, the training accuracy is:- 0.10836957218107701
At randon state 6, the testing accuracy is:- 0.14222366159032407


At random state 7, the training accuracy is:- 0.12975789869673937
At randon state 7, the testing accuracy is:- 0.06523664530156437


At random state 8, the training accuracy is:- 0.12292047652409821
At randon state 8, the testing accuracy is:- 0.023566499890244952


At random state 9, the training accuracy is:- 0.1592836420557947
At randon state 9, the testing accuracy is:- -0.031025927755205185


At random state 10, the training accuracy is:- 0.15805363165814412
At randon state 10, the testing accuracy is:- -0.13454090972560007


At random state 11, the training accuracy is:- 0.17016952126835216
At randon state 11, the testing accuracy is:- -0.39010790367050463


At random state 12, the training accuracy is:- 0.1292474130060698
At randon state 12, the testing accuracy is:- 0.02410191464415079


At random state 13, the training accuracy is:- 0.1489965846650806
At randon state 13, the testing accuracy is:- -0.0816682272204079


At random state 14, the training accuracy is:- 0.13155181519338488
At randon state 14, the testing accuracy is:- 0.038031889080968884


At random state 15, the training accuracy is:- 0.13678521620930895
At randon state 15, the testing accuracy is:- 0.06112052110244548


At random state 16, the training accuracy is:- 0.18491626913593118
At randon state 16, the testing accuracy is:- -0.5979736619887905


At random state 17, the training accuracy is:- 0.17520982601887825
At randon state 17, the testing accuracy is:- -0.4317628145381551


At random state 18, the training accuracy is:- 0.10399248209113288
At randon state 18, the testing accuracy is:- 0.09081801538782253


At random state 19, the training accuracy is:- 0.13658974956487258
At randon state 19, the testing accuracy is:- 0.07798616068616893
```

```
At random state 20, the training accuracy is:- 0.13667316509914718
At randon state 20, the testing accuracy is:- -0.03607602566601176


At random state 21, the training accuracy is:- 0.1379432380868051
At randon state 21, the testing accuracy is:- -0.005872746219661051


At random state 22, the training accuracy is:- 0.16980484356063585
At randon state 22, the testing accuracy is:- -0.0016424181388245973


At random state 23, the training accuracy is:- 0.14774349215990012
At randon state 23, the testing accuracy is:- 0.03315488046430659


At random state 24, the training accuracy is:- 0.18943047659244272
At randon state 24, the testing accuracy is:- -1.6349305133786185


At random state 25, the training accuracy is:- 0.12668475892660547
At randon state 25, the testing accuracy is:- 0.06117201498378666


At random state 26, the training accuracy is:- 0.15853680923632085
At randon state 26, the testing accuracy is:- -0.2622711887948559


At random state 27, the training accuracy is:- 0.13643793843771335
At randon state 27, the testing accuracy is:- 0.0723183022198498


At random state 28, the training accuracy is:- 0.15221985112638137
At randon state 28, the testing accuracy is:- -0.15340730317826723


At random state 29, the training accuracy is:- 0.13453952732166818
At randon state 29, the testing accuracy is:- 0.0698239912498606


At random state 30, the training accuracy is:- 0.11809350614125824
At randon state 30, the testing accuracy is:- 0.05785828132172954


At random state 31, the training accuracy is:- 0.15650466939171004
At randon state 31, the testing accuracy is:- -0.018955882604114738


At random state 32, the training accuracy is:- 0.16665970877557168
At randon state 32, the testing accuracy is:- -0.17916528797520126


At random state 33, the training accuracy is:- 0.15761795348680852
At randon state 33, the testing accuracy is:- -0.16691740961565715


At random state 34, the training accuracy is:- 0.11698794525152889
At randon state 34, the testing accuracy is:- 0.10527680011057494


At random state 35, the training accuracy is:- 0.12532314680997914
At randon state 35, the testing accuracy is:- 0.00496813282923847


At random state 36, the training accuracy is:- 0.15047173619916332
At randon state 36, the testing accuracy is:- -0.2247193639177263


e 37, the training accuracy is:- 0.11122267677351783
```

```
At randon state 37, the testing accuracy is:- 0.10237635183639826


At random state 38, the training accuracy is:- 0.1432460570487747
At randon state 38, the testing accuracy is:- -0.02514866601778465


At random state 39, the training accuracy is:- 0.15869217388534407
At randon state 39, the testing accuracy is:- -0.3302913535888474


At random state 40, the training accuracy is:- 0.10859617008607414
At randon state 40, the testing accuracy is:- 0.09247436810360976


At random state 41, the training accuracy is:- 0.15838514873817622
At randon state 41, the testing accuracy is:- -0.09269799345605456


At random state 42, the training accuracy is:- 0.12231437147345436
At randon state 42, the testing accuracy is:- -0.03610744635909202


At random state 43, the training accuracy is:- 0.16063238820046333
At randon state 43, the testing accuracy is:- -0.18573945451381157


At random state 44, the training accuracy is:- 0.17169157492064302
At randon state 44, the testing accuracy is:- 0.012239439008815745


At random state 45, the training accuracy is:- 0.15854494516265583
At randon state 45, the testing accuracy is:- 0.013688839353579563


At random state 46, the training accuracy is:- 0.15817599047635755
At randon state 46, the testing accuracy is:- -0.27425152055570146


At random state 47, the training accuracy is:- 0.12322918576579589
At randon state 47, the testing accuracy is:- 0.03266635566016207


At random state 48, the training accuracy is:- 0.1464976818990592
At randon state 48, the testing accuracy is:- -0.0780594360254836


At random state 49, the training accuracy is:- 0.1385920730757686
At randon state 49, the testing accuracy is:- -0.0041122540341700375


At random state 50, the training accuracy is:- 0.13305957479100583
At randon state 50, the testing accuracy is:- -0.025632234241745433


At random state 51, the training accuracy is:- 0.15754556269601883
At randon state 51, the testing accuracy is:- -0.15472829306811797


At random state 52, the training accuracy is:- 0.10321169023715804
At randon state 52, the testing accuracy is:- 0.11661529545650506


At random state 53, the training accuracy is:- 0.15800276873154073
At randon state 53, the testing accuracy is:- 0.03944384212626639


At random state 54, the training accuracy is:- 0.13309988178313248
At randon state 54, the testing accuracy is:- 0.01618344092508195
```

```
At random state 55, the training accuracy is:- 0.12421875987777109
At randon state 55, the testing accuracy is:- 0.1371659193905561


At random state 56, the training accuracy is:- 0.13745341317634374
At randon state 56, the testing accuracy is:- -0.18592128178561684


At random state 57, the training accuracy is:- 0.19362449225181733
At randon state 57, the testing accuracy is:- -0.588789153459083


At random state 58, the training accuracy is:- 0.14003517102712848
At randon state 58, the testing accuracy is:- 0.040771282194072556


At random state 59, the training accuracy is:- 0.13060590463779376
At randon state 59, the testing accuracy is:- 0.028906231261758508


At random state 60, the training accuracy is:- 0.17771978993908732
At randon state 60, the testing accuracy is:- -0.49441923069900384


At random state 61, the training accuracy is:- 0.11304999430188423
At randon state 61, the testing accuracy is:- 0.15417779554798583


At random state 62, the training accuracy is:- 0.16399199399084563
At randon state 62, the testing accuracy is:- -0.27151701669866224


At random state 63, the training accuracy is:- 0.1623575385184587
At randon state 63, the testing accuracy is:- -0.05745322816457121


At random state 64, the training accuracy is:- 0.11708044030577314
At randon state 64, the testing accuracy is:- 0.14380233129267095


At random state 65, the training accuracy is:- 0.1347773462250339
At randon state 65, the testing accuracy is:- 0.054538787768055985


At random state 66, the training accuracy is:- 0.1598058963471498
At randon state 66, the testing accuracy is:- -0.44475707954375965


At random state 67, the training accuracy is:- 0.11954021095597056
At randon state 67, the testing accuracy is:- 0.06176039556507351


At random state 68, the training accuracy is:- 0.11922132390412499
At randon state 68, the testing accuracy is:- 0.09835717349438011


At random state 69, the training accuracy is:- 0.15941192752399902
At randon state 69, the testing accuracy is:- -0.04361562768522198


At random state 70, the training accuracy is:- 0.08874838985251476
At randon state 70, the testing accuracy is:- 0.2738058117997463


At random state 71, the training accuracy is:- 0.13796310806879586
At randon state 71, the testing accuracy is:- -0.10269778340944402


e 72, the training accuracy is:- 0.13233948254660377
```

```
At randon state 72, the testing accuracy is:- 0.049731585325446526


At random state 73, the training accuracy is:- 0.13219835123650125
At randon state 73, the testing accuracy is:- 0.04877544408601886


At random state 74, the training accuracy is:- 0.13453680881731078
At randon state 74, the testing accuracy is:- -0.07756587581083463


At random state 75, the training accuracy is:- 0.13895093190559638
At randon state 75, the testing accuracy is:- -0.012762891562899448


At random state 76, the training accuracy is:- 0.15728931770794174
At randon state 76, the testing accuracy is:- -0.26825510208013514


At random state 77, the training accuracy is:- 0.1640866585854195
At randon state 77, the testing accuracy is:- -0.36687332125759653


At random state 78, the training accuracy is:- 0.14639844968857962
At randon state 78, the testing accuracy is:- -0.18250012567734708


At random state 79, the training accuracy is:- 0.1676590968409859
At randon state 79, the testing accuracy is:- -0.18156369947919493


At random state 80, the training accuracy is:- 0.12398094850624786
At randon state 80, the testing accuracy is:- 0.11853892376006336


At random state 81, the training accuracy is:- 0.08597098401064507
At randon state 81, the testing accuracy is:- 0.1587039781504932


At random state 82, the training accuracy is:- 0.13161992297009972
At randon state 82, the testing accuracy is:- -0.0018710607272960367


At random state 83, the training accuracy is:- 0.11025788270213144
At randon state 83, the testing accuracy is:- 0.1815311086375393


At random state 84, the training accuracy is:- 0.11130896095793175
At randon state 84, the testing accuracy is:- 0.03465245025031183


At random state 85, the training accuracy is:- 0.16052815958563216
At randon state 85, the testing accuracy is:- -0.17966611752993877


At random state 86, the training accuracy is:- 0.09826640589329261
At randon state 86, the testing accuracy is:- 0.1014620425529249


At random state 87, the training accuracy is:- 0.12127854299145147
At randon state 87, the testing accuracy is:- -0.10342223498434855


At random state 88, the training accuracy is:- 0.1330038871307011
At randon state 88, the testing accuracy is:- 0.0627456226895623


At random state 89, the training accuracy is:- 0.07537929501414975
At randon state 89, the testing accuracy is:- 0.2166271910850398
```

```
At random state 90, the training accuracy is:- 0.15197600801632594
At randon state 90, the testing accuracy is:- -0.08310112985036766


At random state 91, the training accuracy is:- 0.13684795250086168
At randon state 91, the testing accuracy is:- -0.0029465959264189934


At random state 92, the training accuracy is:- 0.14316261449970236
At randon state 92, the testing accuracy is:- 0.04786843248417583


At random state 93, the training accuracy is:- 0.15907381493197525
At randon state 93, the testing accuracy is:- -0.006249013454435071


At random state 94, the training accuracy is:- 0.1677664065047969
At randon state 94, the testing accuracy is:- -0.0015046122080455415


At random state 95, the training accuracy is:- 0.15344900280092888
At randon state 95, the testing accuracy is:- -0.09624038634324328


At random state 96, the training accuracy is:- 0.137153955219654
At randon state 96, the testing accuracy is:- 0.053304106690357234


At random state 97, the training accuracy is:- 0.11680033766212417
At randon state 97, the testing accuracy is:- 0.16062645966570843


At random state 98, the training accuracy is:- 0.18713233828607367
At randon state 98, the testing accuracy is:- -0.8206119498500091


At random state 99, the training accuracy is:- 0.15615152141580058
At randon state 99, the testing accuracy is:- -0.6573532730249874
```

In [109…
```
xtrain,xtest,ytrain,ytest= train_test_split(x,y,test_size=.20, random_state=62)
```

In [110…
```
lr.fit(xtrain,ytrain)
```

Out[110…
```
LinearRegression()
```

In [111…
```
lr.score(xtrain,ytrain)
```

Out[111…
```
0.16399199399084563
```

This shows that only 16 percent of the model works well.

In [112…
```
pred_lr= lr.predict(xtest)
print('mean_squared_error', mean_squared_error(ytest,pred_lr))
```

```
mean_squared_error 0.00041352509648559494
```

In [113…
```
print('mean_absolute_error:', mean_absolute_error(pred_lr,ytest))
```

```
mean_absolute_error: 0.014322502449364238
```

In [114…
```
print(r2_score(pred_lr,ytest))
```

```
-10.360514482485673
```

In [115…
```
print('Predicted values:', pred_lr)
```

```
Predicted values: [0.04720608 0.04132952 0.05283639 0.06210127 0.0454988  0.04792681
 0.0454189  0.03653997 0.0408559  0.05491734 0.04507189 0.04592365
 0.04344758 0.04049654 0.04682119 0.04848066 0.04784863 0.05403897
 0.05108236 0.03346498 0.04883321 0.05319862 0.05681095 0.04448141
 0.05284819 0.05005066 0.04301661 0.03906992 0.04683793 0.03952366
 0.04313792 0.04951297]
```

In [116…
```
print('Actual Values:', ytest)
```

```
Actual Values: 29     0.04612
131    0.03751
123    0.03947
119    0.03680
82     0.05235
107    0.04394
62     0.07832
65     0.05635
54     0.04251
117    0.06740
133    0.04828
92     0.05376
70     0.07197
48     0.05412
39     0.09811
125    0.08096
134    0.03260
112    0.04585
76     0.03823
27     0.06257
52     0.04563
152    0.03084
130    0.06130
56     0.05371
118    0.07331
32     0.05051
47     0.04528
46     0.04098
53     0.04114
38     0.04456
60     0.04330
40     0.10895
Name: Standard Error, dtype: float64
```

Cross Validation Score

In [75]:
```
#
```

In [117…
```
train_accuracy= r2_score(ytrain,pred_tr)
test_accuracy= r2_score(ytest,pred_ts)


from sklearn.model_selection import cross_val_score
for j in range(2,10):
    cv_score= cross_val_score(lr,x,y,cv=j)
    cv_mean=cv_score.mean()
    print(f"At cross fold {j} the cv score is {cv_mean} and accuracy score for training is
")
```

```
At cross fold 2 the cv score is -0.38395380855218897 and accuracy score for training is -
0.22711219250310744 and the accuracy for testing is -0.0002501172604392199


At cross fold 3 the cv score is -0.3952534372169725 and accuracy score for training is -0.
22711219250310744 and the accuracy for testing is -0.0002501172604392199


At cross fold 4 the cv score is -0.5127854622652036 and accuracy score for training is -0.
22711219250310744 and the accuracy for testing is -0.0002501172604392199


At cross fold 5 the cv score is -0.44007357240339146 and accuracy score for training is -
0.22711219250310744 and the accuracy for testing is -0.0002501172604392199


At cross fold 6 the cv score is -0.4032947769710284 and accuracy score for training is -0.
22711219250310744 and the accuracy for testing is -0.0002501172604392199


At cross fold 7 the cv score is -0.34131952729226306 and accuracy score for training is -
0.22711219250310744 and the accuracy for testing is -0.0002501172604392199


At cross fold 8 the cv score is -0.7934627963087527 and accuracy score for training is -0.
22711219250310744 and the accuracy for testing is -0.0002501172604392199


At cross fold 9 the cv score is -0.6012746338228675 and accuracy score for training is -0.
22711219250310744 and the accuracy for testing is -0.0002501172604392199
```

Here cv_fold=9 is selected which is nearly close to r2 score .

In [129…
```
#
```

Decision Tree Regressor

In [118…
```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

In [119…
```
dtc= DecisionTreeRegressor()
dtc.fit(xtrain,ytrain)
```

Out[119…
```
DecisionTreeRegressor()
```

In [120…
```
dtc.score(xtrain,ytrain)
```

Out[120…
```
1.0
```

In [121…
```
preddtc = dtc.predict(xtest)
print('mean_squared_error:',mean_squared_error(preddtc, ytest))
```

```
mean_squared_error: 0.000853003646875
```

In [122…
```
print('mean_absolute_error:', mean_absolute_error(preddtc,ytest))
```

```
mean_absolute_error: 0.0198671875
```

In [123…
```
print(r2_score(preddtc,ytest))
```

```
-1.3926663066382243
```

Model has more error in Decision tree.

In [132…
```
from sklearn.ensemble import RandomForestRegressor
```

In [133…
```
rf=RandomForestRegressor(criterion='mse',max_features= 'auto')
rf.fit(xtrain,ytrain)
rf.score(xtrain,ytrain)
```

Out[133…
```
0.8676090005959268
```

In [134…
```
pred_rf= rf.predict(xtest)
```

In [135…
```
rfs= r2_score(ytest,pred_rf)
print('r2_score:', rfs*100)
```

```
r2_score: -26.664515222559302
```

In [137…
```
#Linear Regression model shows better performance than other models
```

MODEL SAVING

In [126…
```
import pickle
filename= 'World Happiness Report.pkl'
pickle.dump(lr,open(filename, 'wb'))
```

In [127…
```
x=np.array(ytest)
predicted= np.array(lr.predict(xtest))
df_con= pd.DataFrame({'original': x, 'Predicted': predicted}, index= range(len(x)))
df_con
```

Out[127…

	original	Predicted
0	0.04612	0.047206
1	0.03751	0.041330
2	0.03947	0.052836
3	0.03680	0.062101
4	0.05235	0.045499
5	0.04394	0.047927
6	0.07832	0.045419
7	0.05635	0.036540
8	0.04251	0.040856
9	0.06740	0.054917
10	0.04828	0.045072

	original	Predicted
11	0.05376	0.045924
12	0.07197	0.043448
13	0.05412	0.040497
14	0.09811	0.046821
15	0.08096	0.048481
16	0.03260	0.047849
17	0.04585	0.054039
18	0.03823	0.051082
19	0.06257	0.033465
20	0.04563	0.048833
21	0.03084	0.053199
22	0.06130	0.056811
23	0.05371	0.044481
24	0.07331	0.052848
25	0.05051	0.050051
26	0.04528	0.043017
27	0.04098	0.039070
28	0.04114	0.046838
29	0.04456	0.039524
30	0.04330	0.043138
31	0.10895	0.049513

In [ ]:

```
The model works well with the dataset. The actual values are near to the predicted values
```