In [4]:

```
import pandas as pd
import numpy as np
whr=pd.read_csv('World Happiness Report.csv')
```

In [5]:

```
whr
```

Out[5]:

	Country	Happiness Rank	Happiness Score	Economy	Family	Health	Freedom	Generosity	Corruption	Dystopia	Job Satisfaction	Regi
0	Norway	1	7.537	1.616463	1.533524	0.796667	0.635423	0.362012	0.315964	2.277027	94.6	Weste Euro
1	Denmark	2	7.522	1.482383	1.551122	0.792566	0.626007	0.355280	0.400770	2.313707	93.5	Weste Euro
2	Iceland	3	7.504	1.480633	1.610574	0.833552	0.627163	0.475540	0.153527	2.322715	94.5	Weste Euro
3	Switzerland	4	7.494	1.564980	1.516912	0.858131	0.620071	0.290549	0.367007	2.276716	93.7	Weste Euro
4	Finland	5	7.469	1.443572	1.540247	0.809158	0.617951	0.245483	0.382612	2.430182	91.2	Weste Euro
...	...	...	...	...	...	...	...	...	...	...	...	
148	Rwanda	151	3.471	0.368746	0.945707	0.326425	0.581844	0.252756	0.455220	0.540061	51.7	Afr
149	Syria	152	3.462	0.777153	0.396103	0.500533	0.081539	0.493664	0.151347	1.061574	62.7	As Pac
150	Tanzania	153	3.349	0.511136	1.041990	0.364509	0.390018	0.354256	0.066035	0.621130	57.8	Afr
151	Burundi	154	2.905	0.091623	0.629794	0.151611	0.059901	0.204435	0.084148	1.683024	54.3	Afr
152	Central African Republic	155	2.693	0.000000	0.000000	0.018773	0.270842	0.280876	0.056565	2.066005	70.4	Afr

153 rows × 12 columns

In [6]:

```
whr.dtypes
```

Out[6]:

```
Country              object
Happiness Rank        int64
Happiness Score     float64
Economy             float64
Family              float64
Health              float64
Freedom             float64
Generosity          float64
Corruption          float64
Dystopia            float64
Job Satisfaction    float64
Region               object
dtype: object
```

In [7]:

```
whr.columns
```

Out[7]:

```
Index(['Country', 'Happiness Rank', 'Happiness Score', 'Economy', 'Family',
       'Health', 'Freedom', 'Generosity', 'Corruption', 'Dystopia',
       'Job Satisfaction', 'Region'],
```

dtype='object')

In [8]:

```
whr.describe()
```

Out[8]:

	Happiness Rank	Happiness Score	Economy	Family	Health	Freedom	Generosity	Corruption	Dystopia	Job Satisfaction
count	153.000000	153.000000	153.000000	153.000000	153.000000	153.000000	153.000000	153.000000	153.000000	151.000000
mean	78.169935	5.349281	0.982433	1.186630	0.550117	0.408489	0.245324	0.123179	1.853072	75.209934
std	45.008741	1.134997	0.421901	0.288441	0.237769	0.150744	0.134395	0.102133	0.499490	12.962365
min	1.000000	2.693000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.377914	44.400000
25%	40.000000	4.497000	0.659517	1.041990	0.364509	0.300741	0.153075	0.057070	1.597970	68.950000
50%	78.000000	5.279000	1.064578	1.251826	0.606042	0.437454	0.231503	0.089848	1.832910	78.100000
75%	117.000000	6.098000	1.315175	1.416404	0.719217	0.518631	0.322228	0.153066	2.150801	85.100000
max	155.000000	7.537000	1.870766	1.610574	0.949492	0.658249	0.838075	0.464308	3.117485	95.100000

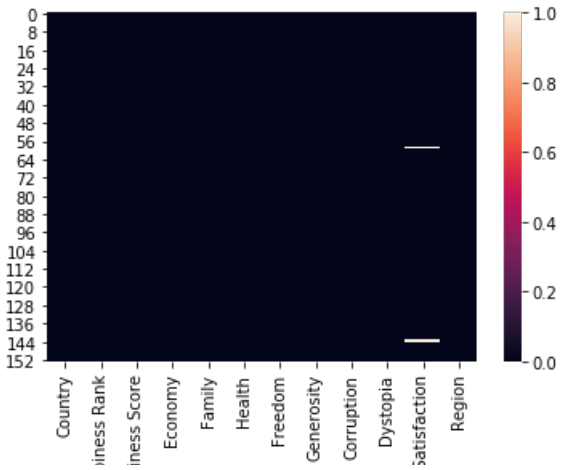
In [9]:

```
whr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 153 entries, 0 to 152
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Country           153 non-null    object 
 1   Happiness Rank    153 non-null    int64  
 2   Happiness Score   153 non-null    float64
 3   Economy           153 non-null    float64
 4   Family            153 non-null    float64
 5   Health            153 non-null    float64
 6   Freedom           153 non-null    float64
 7   Generosity        153 non-null    float64
 8   Corruption        153 non-null    float64
 9   Dystopia          153 non-null    float64
 10  Job Satisfaction  151 non-null    float64
 11  Region            153 non-null    object 
dtypes: float64(9), int64(1), object(2)
memory usage: 14.5+ KB
```

In [10]:

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(whr.isnull())
plt.show()
```



Happ Happi Job s

In [11]:

```
whr.isnull().sum()
```

Out[11]:

```
Country              0
Happiness Rank       0
Happiness Score      0
Economy              0
Family               0
Health               0
Freedom              0
Generosity           0
Corruption           0
Dystopia             0
Job Satisfaction     2
Region               0
dtype: int64
```

In [12]:

```
whr.dropna(inplace=True)
```

In [13]:

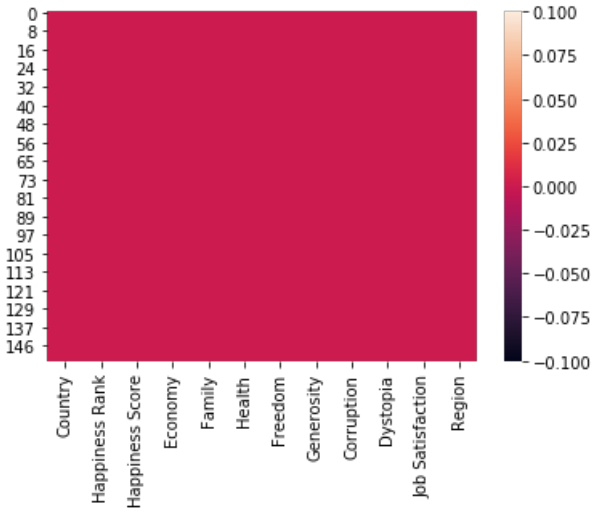
```
whr.isnull().sum()
```

Out[13]:

```
Country              0
Happiness Rank       0
Happiness Score      0
Economy              0
Family               0
Health               0
Freedom              0
Generosity           0
Corruption           0
Dystopia             0
Job Satisfaction     0
Region               0
dtype: int64
```

In [14]:

```
sns.heatmap(whr.isnull())
plt.show()
```



In [15]:

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
list1=['Country','Region']
for val in list1:
    whr[val]=le.fit_transform(whr[val].astype(str))
```

In [16]:

```
whr.head()
```

Out[16]:

	Country	Happiness Rank	Happiness Score	Economy	Family	Health	Freedom	Generosity	Corruption	Dystopia	Job Satisfaction	Region
0	102	1	7.537	1.616463	1.533524	0.796667	0.635423	0.362012	0.315964	2.277027	94.6	6
1	36	2	7.522	1.482383	1.551122	0.792566	0.626007	0.355280	0.400770	2.313707	93.5	6
2	56	3	7.504	1.480633	1.610574	0.833552	0.627163	0.475540	0.153527	2.322715	94.5	6
3	129	4	7.494	1.564980	1.516912	0.858131	0.620071	0.290549	0.367007	2.276716	93.7	6
4	43	5	7.469	1.443572	1.540247	0.809158	0.617951	0.245483	0.382612	2.430182	91.2	6

In [17]:

```
#conutry and region are drop and happpiness rank is droped because rank can be formed after knowin
g happiness score
whr1=whr.drop(['Country','Region','Happiness Rank'],axis=1)
whr1
```

Out[17]:

	Happiness Score	Economy	Family	Health	Freedom	Generosity	Corruption	Dystopia	Job Satisfaction
0	7.537	1.616463	1.533524	0.796667	0.635423	0.362012	0.315964	2.277027	94.6
1	7.522	1.482383	1.551122	0.792566	0.626007	0.355280	0.400770	2.313707	93.5
2	7.504	1.480633	1.610574	0.833552	0.627163	0.475540	0.153527	2.322715	94.5
3	7.494	1.564980	1.516912	0.858131	0.620071	0.290549	0.367007	2.276716	93.7
4	7.469	1.443572	1.540247	0.809158	0.617951	0.245483	0.382612	2.430182	91.2
...	...	...	...	...	...	...	...	...	...
148	3.471	0.368746	0.945707	0.326425	0.581844	0.252756	0.455220	0.540061	51.7
149	3.462	0.777153	0.396103	0.500533	0.081539	0.493664	0.151347	1.061574	62.7
150	3.349	0.511136	1.041990	0.364509	0.390018	0.354256	0.066035	0.621130	57.8
151	2.905	0.091623	0.629794	0.151611	0.059901	0.204435	0.084148	1.683024	54.3
152	2.693	0.000000	0.000000	0.018773	0.270842	0.280876	0.056565	2.066005	70.4

151 rows × 9 columns

In [18]:

```
whr1.skew()
```

Out[18]:

```
Happiness Score     0.018978
Economy            -0.388885
Family             -1.186278
Health             -0.582749
Freedom            -0.617270
Generosity          0.925782
Corruption          1.471012
Dystopia           -0.251490
Job Satisfaction   -0.613859
```

```
dtype: float64
```

In [19]:

```
whr1.corr()
```

Out[19]:

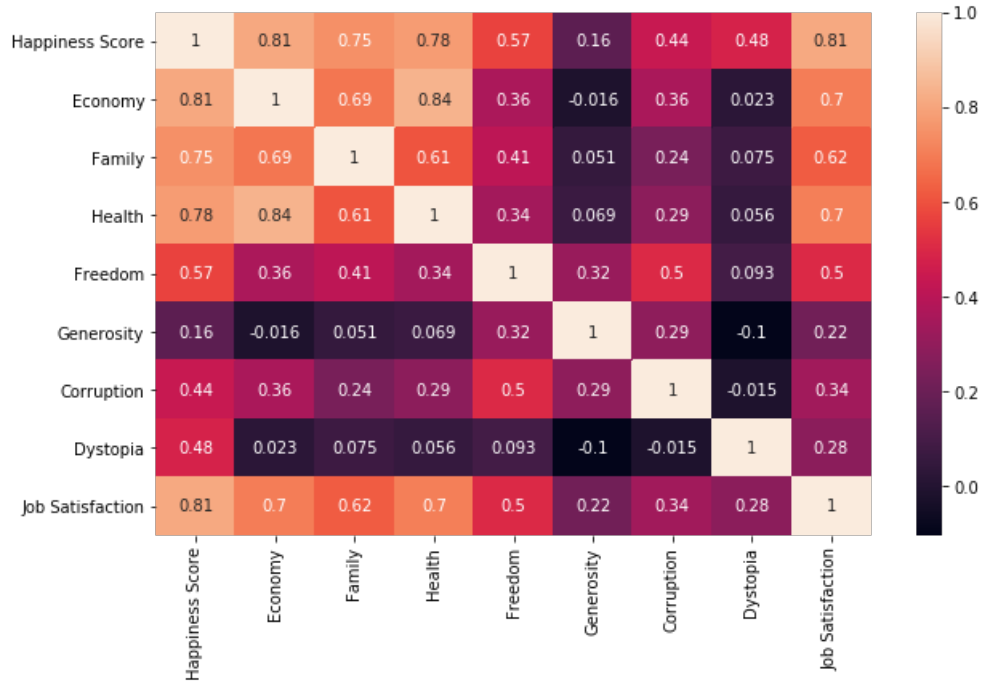
	Happiness Score	Economy	Family	Health	Freedom	Generosity	Corruption	Dystopia	Job Satisfaction
Happiness Score	1.000000	0.808678	0.749612	0.777731	0.567948	0.164123	0.438262	0.481117	0.812873
Economy	0.808678	1.000000	0.685524	0.838884	0.363843	-0.015614	0.358750	0.022620	0.700662
Family	0.749612	0.685524	1.000000	0.606674	0.412633	0.050771	0.236262	0.075480	0.623266
Health	0.777731	0.838884	0.606674	1.000000	0.340986	0.068895	0.286777	0.055886	0.704795
Freedom	0.567948	0.363843	0.412633	0.340986	1.000000	0.319387	0.501632	0.092923	0.500655
Generosity	0.164123	-0.015614	0.050771	0.068895	0.319387	1.000000	0.292363	-0.102683	0.220032
Corruption	0.438262	0.358750	0.236262	0.286777	0.501632	0.292363	1.000000	-0.014995	0.337131
Dystopia	0.481117	0.022620	0.075480	0.055886	0.092923	-0.102683	-0.014995	1.000000	0.281655
Job Satisfaction	0.812873	0.700662	0.623266	0.704795	0.500655	0.220032	0.337131	0.281655	1.000000

In [20]:

```
plt.figure(figsize=(10,6))
sns.heatmap(whr1.corr(),annot=True)
```

Out[20]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x211d83d6308>
```



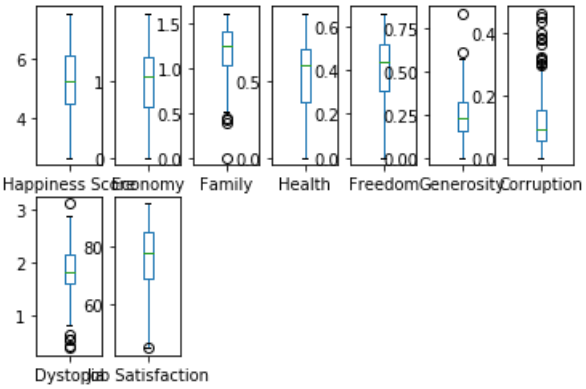
In [21]:

```
whr1.plot(kind='box',subplots=True,layout=(2,7))
```

Out[21]:

```
Happiness Score        AxesSubplot(0.125,0.536818;0.0945122x0.343182)
Economy             AxesSubplot(0.238415,0.536818;0.0945122x0.343182)
Family              AxesSubplot(0.351829,0.536818;0.0945122x0.343182)
Health              AxesSubplot(0.465244,0.536818;0.0945122x0.343182)
```

```
Freedom              AxesSubplot(0.578659,0.536818;0.0945122x0.343182)
Generosity           AxesSubplot(0.692073,0.536818;0.0945122x0.343182)
Corruption           AxesSubplot(0.805488,0.536818;0.0945122x0.343182)
Dystopia                  AxesSubplot(0.125,0.125;0.0945122x0.343182)
Job Satisfaction       AxesSubplot(0.238415,0.125;0.0945122x0.343182)
dtype: object
```



In [22]:

```
#Removing outliers
from scipy.stats import zscore
z_score=abs(zscore(whr1))
print(whr1.shape)
whr_df=whr1.loc[(z_score<3).all(axis=1)]
print(whr_df.shape)
```

```
(151, 9)
(146, 9)
```

In [23]:

```
whr_df
```

Out[23]:

	Happiness Score	Economy	Family	Health	Freedom	Generosity	Corruption	Dystopia	Job Satisfaction
0	7.537	1.616463	1.533524	0.796667	0.635423	0.362012	0.315964	2.277027	94.6
1	7.522	1.482383	1.551122	0.792566	0.626007	0.355280	0.400770	2.313707	93.5
2	7.504	1.480633	1.610574	0.833552	0.627163	0.475540	0.153527	2.322715	94.5
3	7.494	1.564980	1.516912	0.858131	0.620071	0.290549	0.367007	2.276716	93.7
4	7.469	1.443572	1.540247	0.809158	0.617951	0.245483	0.382612	2.430182	91.2
...	...	...	...	...	...	...	...	...	...
146	3.507	0.244550	0.791245	0.194129	0.348588	0.264815	0.110938	1.552312	55.1
147	3.495	0.305445	0.431883	0.247106	0.380426	0.196896	0.095665	1.837229	44.8
149	3.462	0.777153	0.396103	0.500533	0.081539	0.493664	0.151347	1.061574	62.7
150	3.349	0.511136	1.041990	0.364509	0.390018	0.354256	0.066035	0.621130	57.8
151	2.905	0.091623	0.629794	0.151611	0.059901	0.204435	0.084148	1.683024	54.3

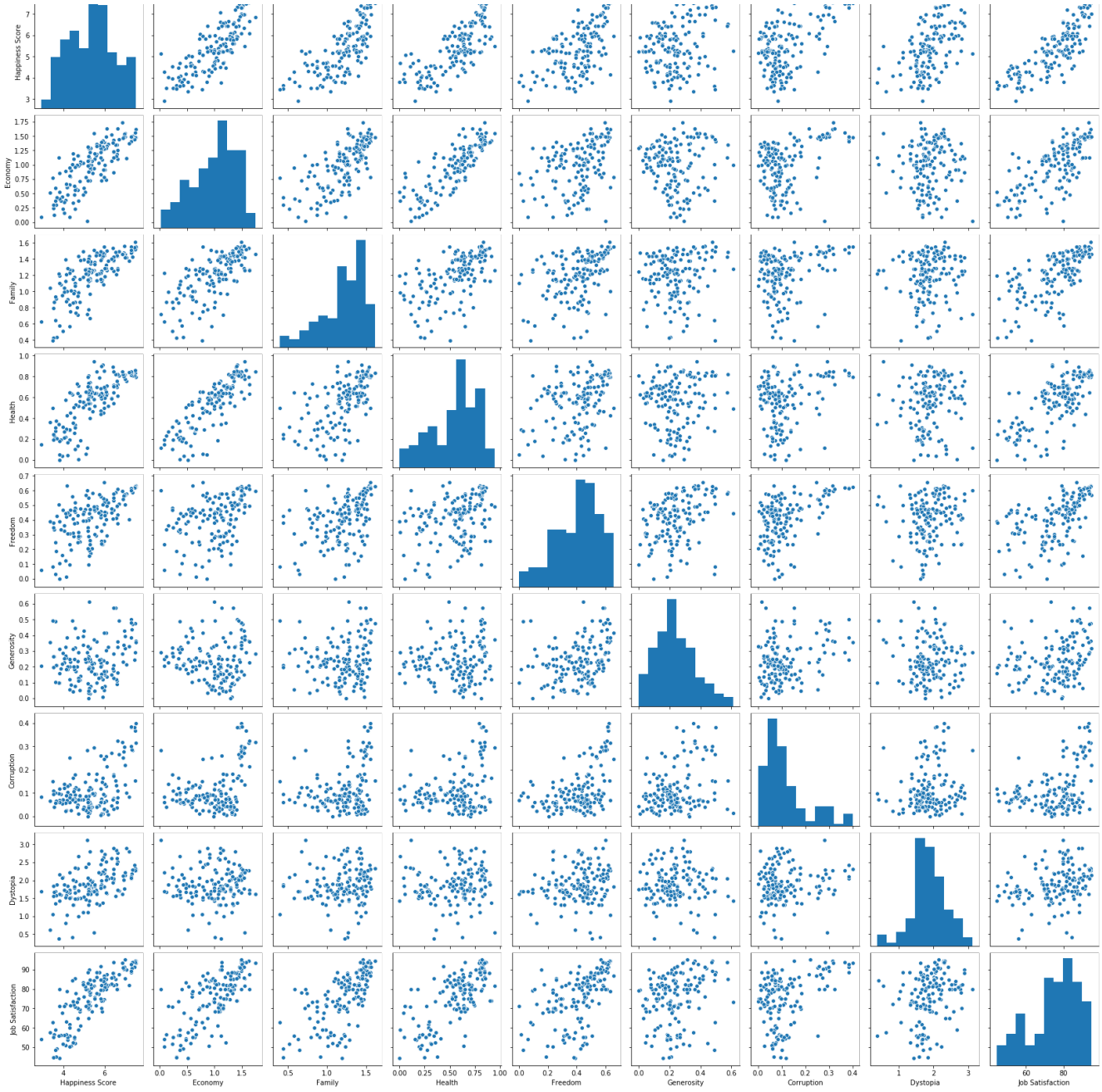
146 rows × 9 columns

In [24]:

```
sns.pairplot(whr_df)
```

Out[24]:

```
<seaborn.axisgrid.PairGrid at 0x211d8124a48>
```



In [25]:

```
x1=whr_df.iloc[:,1:-1]
x1.head()
```

Out[25]:

	Economy	Family	Health	Freedom	Generosity	Corruption	Dystopia
0	1.616463	1.533524	0.796667	0.635423	0.362012	0.315964	2.277027
1	1.482383	1.551122	0.792566	0.626007	0.355280	0.400770	2.313707
2	1.480633	1.610574	0.833552	0.627163	0.475540	0.153527	2.322715
3	1.564980	1.516912	0.858131	0.620071	0.290549	0.367007	2.276716
4	1.443572	1.540247	0.809158	0.617951	0.245483	0.382612	2.430182

In [26]:

```
y=whr_df.iloc[:,0]
y.head()
```

Out[26]:

```
0    7.537
```

```
1    7.522
2    7.504
3    7.494
4    7.469
Name: Happiness Score, dtype: float64
```

In [27]:

```
x1.shape
```

Out[27]:

```
(146, 7)
```

In [28]:

```
y.shape
```

Out[28]:

```
(146,)
```

In [29]:

```
x1.skew()
```

Out[29]:

```
Economy      -0.451234
Family       -0.933474
Health       -0.610641
Freedom      -0.603302
Generosity    0.564921
Corruption    1.354733
Dystopia     -0.222308
dtype: float64
```

In [30]:

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x=sc.fit_transform(x1)
x=pd.DataFrame(x,columns=x1.columns)
```

In [31]:

```
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression
scr=cross_val_score(LinearRegression(),x,y,cv=5,scoring="r2")
scr
```

Out[31]:

```
array([0.99999932, 0.99999745, 0.99999476, 0.99999846, 0.99999908])
```

In [32]:

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
max_r_score=0
for r_state in range (40,140):
    x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=r_state,test_size=.20)
    regr=linear_model.LinearRegression()
    regr.fit(x_train,y_train)
    y_pred=regr.predict(x_test)
    r_scr=r2_score(y_test,y_pred)
```

```
    if r_scr>max_r_score:
        max_r_score=r_scr
        final_r_state=r_state
print("max r2 score corresponding to ",final_r_state,"is",max_r_score)
```

```
max r2 score corresponding to  72 is 0.9999999597150357
```

In [33]:

```
from sklearn.linear_model import LinearRegression,Lasso,Ridge,ElasticNet
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import cross_val_score
import warnings
warnings.filterwarnings('ignore')
```

In [34]:

```
model=[LinearRegression(),DecisionTreeRegressor(),KNeighborsRegressor(),SVR(),Lasso(),Ridge(),Elast
icNet()]
for m in model:
    x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=72,test_size=.20)
    m.fit(x_train,y_train)
    print('Score of',m,'is:',m.score(x_train,y_train))
    predm=m.predict(x_test)
    print('Error:')
    print('Mean Absolute Error :',mean_absolute_error(y_test,predm))
    print('Mean Squared Error :',mean_squared_error(y_test,predm))
    print('r2_score',r2_score(y_test,predm))

print('**********************************************************************************************
*******')
    print('\n')
```

```
Score of LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False) is: 0.999
9999928368205
Error:
Mean Absolute Error : 0.00021959141033075133
Mean Squared Error : 7.086539738680373e-08
r2_score 0.9999999597150357
**********************************************************************************************



Score of DecisionTreeRegressor(ccp_alpha=0.0, 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='deprecated',
                      random_state=None, splitter='best') is: 1.0
Error:
Mean Absolute Error : 0.47063338760000023
Mean Squared Error : 0.3562189754441714
r2_score 0.7974996371403085
**********************************************************************************************



Score of KNeighborsRegressor(algorithm='auto', leaf_size=30, metric='minkowski',
                    metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                    weights='uniform') is: 0.9405887466231456
Error:
Mean Absolute Error : 0.27515335242666694
Mean Squared Error : 0.11123523076531455
r2_score 0.9367659329077824
**********************************************************************************************



Score of SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='scale',
```

```
    kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False) is: 0.9923499886002044
Error:
Mean Absolute Error : 0.1936537190140878
Mean Squared Error : 0.09968882511481529
r2_score 0.9433297363408698
**********************************************************************************************



Score of Lasso(alpha=1.0, copy_X=True, fit_intercept=True, max_iter=1000,
      normalize=False, positive=False, precompute=False, random_state=None,
      selection='cyclic', tol=0.0001, warm_start=False) is: 0.0
Error:
Mean Absolute Error : 1.157019536845977
Mean Squared Error : 1.776346063540471
r2_score -0.009802248694935178
**********************************************************************************************



Score of Ridge(alpha=1.0, copy_X=True, fit_intercept=True, max_iter=None,
      normalize=False, random_state=None, solver='auto', tol=0.001) is: 0.9999640837339236
Error:
Mean Absolute Error : 0.006592445196220931
Mean Squared Error : 5.3723550891431e-05
r2_score 0.9999694596882829
**********************************************************************************************



Score of ElasticNet(alpha=1.0, copy_X=True, fit_intercept=True, l1_ratio=0.5,
           max_iter=1000, normalize=False, positive=False, precompute=False,
           random_state=None, selection='cyclic', tol=0.0001, warm_start=False) is: 0.3339069944387
446
Error:
Mean Absolute Error : 0.9544786811555931
Mean Squared Error : 1.1831572139138233
r2_score 0.3274087523303183
**********************************************************************************************
```

In [35]:

```
from sklearn.model_selection import cross_val_score
for m in model:
    score=cross_val_score(m,x,y,cv=5,scoring='r2')
    print('Score of',m,'is:',score)
    print('Mean score:',score.mean())
    print('Standard deviation:',score.std())

print('**********************************************************************************************
*******')
    print('\n')
```

```
Score of LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False) is: [0.99
999932 0.99999745 0.99999476 0.99999846 0.99999908]
Mean score: 0.999997813968106
Standard deviation: 1.658043667759746e-06
**********************************************************************************************



Score of DecisionTreeRegressor(ccp_alpha=0.0, 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='deprecated',
                      random_state=None, splitter='best') is: [ -4.39130893 -10.89415722 -
32.68135669  -5.12440354  -3.94220177]
Mean score: -11.406685628269326
Standard deviation: 10.929550877393511
**********************************************************************************************
```

```
Score of KNeighborsRegressor(algorithm='auto', leaf_size=30, metric='minkowski',
                    metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                    weights='uniform') is: [-5.89088001 -2.25108545 -6.49121131 -3.12479166 -5.6726
9042]
Mean score: -4.68613176987301
Standard deviation: 1.6763210882366675
**********************************************************************************************



Score of SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='scale',
    kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False) is: [-10.07675395   0.3933
0088   0.66781179   0.6879335   -7.79390585]
Mean score: -3.2243227272973116
Standard deviation: 4.719715331414879
**********************************************************************************************



Score of Lasso(alpha=1.0, copy_X=True, fit_intercept=True, max_iter=1000,
      normalize=False, positive=False, precompute=False, random_state=None,
      selection='cyclic', tol=0.0001, warm_start=False) is: [-28.28635306 -16.71472884  -
0.12327462 -13.10567121 -34.03136464]
Mean score: -18.45227847237902
Standard deviation: 11.890959750263681
**********************************************************************************************



Score of Ridge(alpha=1.0, copy_X=True, fit_intercept=True, max_iter=None,
      normalize=False, random_state=None, solver='auto', tol=0.001) is: [0.99726087 0.99876076
0.99910396 0.99908454 0.99808084]
Mean score: 0.9984581963237954
Standard deviation: 0.0007038254527990534
**********************************************************************************************



Score of ElasticNet(alpha=1.0, copy_X=True, fit_intercept=True, l1_ratio=0.5,
           max_iter=1000, normalize=False, positive=False, precompute=False,
           random_state=None, selection='cyclic', tol=0.0001, warm_start=False) is: [-26.20007724
-6.7556489   -1.39469345  -6.07295837 -33.69876688]
Mean score: -14.824428971012065
Standard deviation: 12.709817075431094
**********************************************************************************************
```

In [37]:

```
import joblib
joblib.dump(LinearRegression,'World Happiness Report.pkl')
```

Out[37]:

```
['World Happiness Report.pkl']
```

In [ ]:

```

```