Day-10

2015 Dataset

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
d=pd.read_csv(r"C:\Users\user\Downloads\2015.csv")
d
```

Out[2]:

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

158 rows × 12 columns

In [3]:

```
d.columns
```

Out[3]:

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

In [4]:

```
d.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 [5]:

```
x=d[['Happiness Rank', 'Happiness Score',
       'Standard Error', 'Economy (GDP per Capita)', 'Family',
       'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
       'Generosity']]
y=d['Dystopia Residual']
```

In [6]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [7]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[7]:

```
LinearRegression()
```

In [8]:

```
print(lr.intercept_)
```

```
-0.0001623774325247851
```

In [9]:

```
print(lr.score(x_test,y_test))
```

```
0.9999997281342263
```

In [10]:

```
print(lr.score(x_train,y_train))
```

```
0.9999997500715344
```

Ridge Regression

In [11]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [12]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

Out[12]:

```
0.6621369628958466
```

Lasso Regression

In [13]:

```
la=Lasso(alpha=10)
```

In [14]:

```
la.fit(x_train,y_train)
```

Out[14]:

```
Lasso(alpha=10)
```

In [15]:

```
la.score(x_test,y_test)
```

Out[15]:

```
0.02759259052125862
```

In [16]:

```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[16]:

```
ElasticNet()
```

In [17]:

```
predict=(en.predict(x_test))
print(predict)
```

```
[1.83377735 2.16594831 1.77287934 2.09397793 2.07736938 2.23791868
 2.37078707 2.30988906 2.42614889 2.55348109 2.49258308 2.3264976
 1.77841552 1.89467536 1.73412606 1.97218191 2.11612266 2.44275744
 2.17702067 1.99986283 2.03861611 1.76180697 1.96110955 2.55901728
 2.3597147  2.38739561 1.75073461 1.72305369 2.14380358 1.8171688
 1.9112839  2.39846798 2.38185943 2.44829362 1.9777181  1.70644515
 1.87253063 2.54794491 2.15487594 1.88913917 1.82270498 1.79502407
 2.0718332  2.02754374 1.7839517  2.2323825  2.06076084 2.13273121]
```

In [18]:

```
print(en.score(x_test,y_test))
```

```
0.2783333925043625
```

Evaluation Method

In [19]:

```
from sklearn import metrics
```

In [20]:

```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,predict))
```

```
Mean Absolute Error: 0.386992644510085
```

In [21]:

```
print("Mean Square Error:",metrics.mean_squared_error(y_test,predict))
```

```
Mean Square Error: 0.2612470404014113
```

In [22]:

```
print("Root Mean Square Error:",np.sqrt(metrics.mean_squared_error(y_test,predict)))
```

```
Root Mean Square Error: 0.5111233123243464
```

In [ ]:

```
```