kaviyadevi 20106064

In [1]:
```
#to import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
#to import dataset
data=pd.read_csv(r"C:\Users\user\Downloads\2015 - 2015.csv")
data
```

Out[3]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freec
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.66
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.63
...	...	...	...	...	...	...	...	...	
153	Rwanda	Sub-Saharan Africa	154	3.465	0.03464	0.22208	0.77370	0.42864	0.59
154	Benin	Sub-Saharan Africa	155	3.340	0.03656	0.28665	0.35386	0.31910	0.48
155	Syria	Middle East and Northern Africa	156	3.006	0.05015	0.66320	0.47489	0.72193	0.15
156	Burundi	Sub-Saharan Africa	157	2.905	0.08658	0.01530	0.41587	0.22396	0.11
157	Togo	Sub-Saharan Africa	158	2.839	0.06727	0.20868	0.13995	0.28443	0.36

158 rows × 12 columns

In [4]:
```
#to display top 5 rows
data.head()
```

Out[4]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freed
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.66
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.63

DATA CLEANING AND PREPROCESSING

In [5]:
```
data.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 [6]:
```
#to display summary of statistics
data.describe()
```

Out[6]:

	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	(Gov Cor
count	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158
mean	79.493671	5.375734	0.047885	0.846137	0.991046	0.630259	0.428615	0
std	45.754363	1.145010	0.017146	0.403121	0.272369	0.247078	0.150693	0
min	1.000000	2.839000	0.018480	0.000000	0.000000	0.000000	0.000000	0
25%	40.250000	4.526000	0.037268	0.545808	0.856823	0.439185	0.328330	0
50%	79.500000	5.232500	0.043940	0.910245	1.029510	0.696705	0.435515	0
75%	118.750000	6.243750	0.052300	1.158448	1.214405	0.811013	0.549092	0
max	158.000000	7.587000	0.136930	1.690420	1.402230	1.025250	0.669730	0

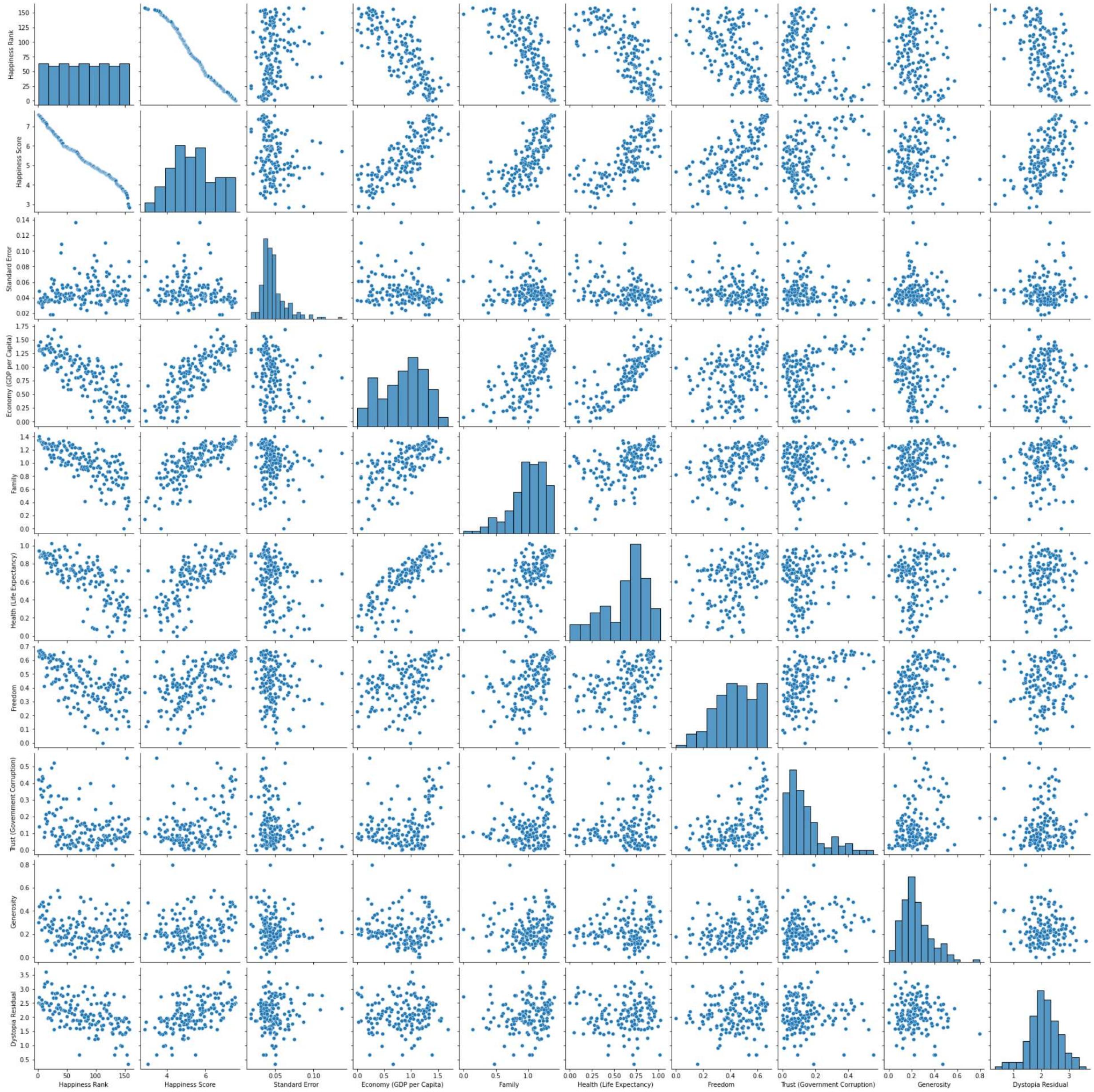
In [7]:
```
#to display the column heading
data.columns
```

Out[7]:
```
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')
```

EDA and DATA VISUALIZATION

In [8]:
```
sns.pairplot(data)
```

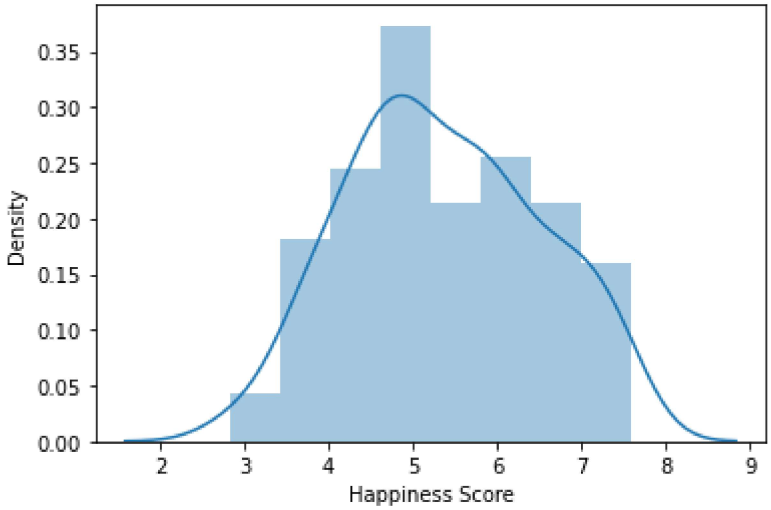
Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x2925b193970>
```



In [9]:
```
sns.distplot(data['Happiness Score'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```

Out[9]:
```
<AxesSubplot:xlabel='Happiness Score', ylabel='Density'>
```



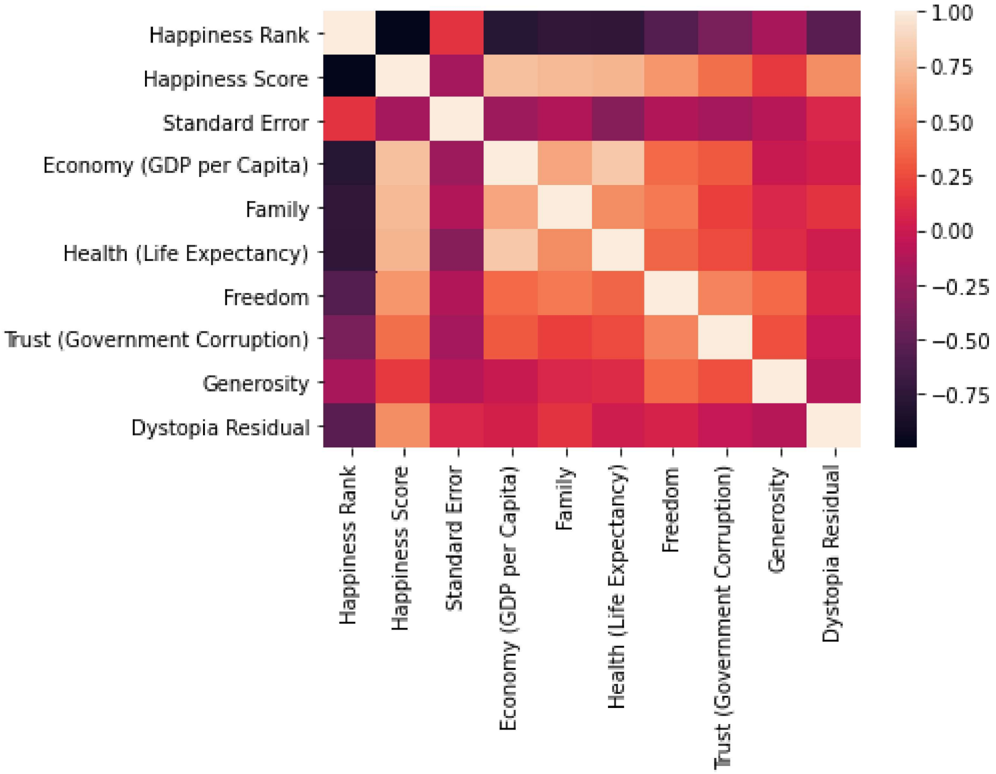
In [10]:
```
df=data[['Country', 'Region', 'Happiness Rank', 'Happiness Score',
       'Standard Error', 'Economy (GDP per Capita)', 'Family',
       'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
       'Generosity', 'Dystopia Residual']]
```

In [11]:

```
sns.heatmap(df.corr())
```

Out[11]:

```
<AxesSubplot:>
```



TRAINING MODEL

In [26]:
```
x=df[['Happiness Rank','Standard Error', 'Economy (GDP per Capita)', 'Family','He
y=df['Happiness Score']
```

In [27]:
```
#to split my dataset into trainning and test
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 [28]:
```
from sklearn.linear_model import LinearRegression

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

Out[28]:
```
LinearRegression()
```

In [29]:
```
#to find intercept
print(lr.intercept_)
```

```
0.0022510866673615126
```

In [30]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[30]:

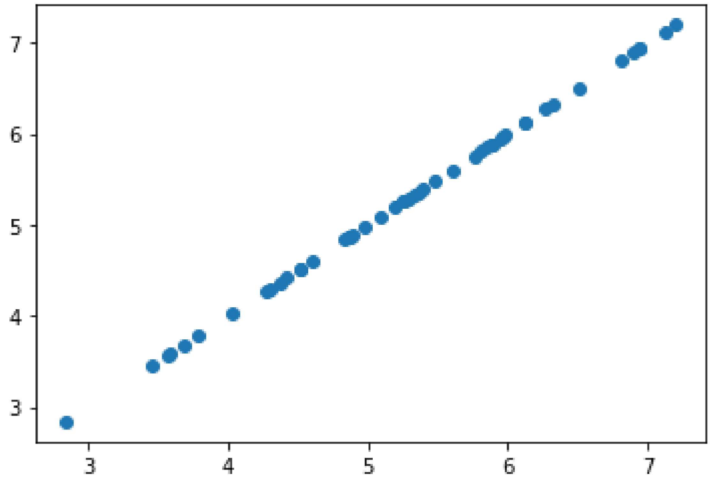
	Co-efficient
Happiness Rank	-0.000008
Standard Error	-0.000863
Economy (GDP per Capita)	0.999855
Family	0.999737
Health (Life Expectancy)	0.999368
Freedom	0.999382
Trust (Government Corruption)	0.999388
Generosity	0.999901
Dystopia Residual	0.999800

In [31]:

```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[31]:

```
<matplotlib.collections.PathCollection at 0x29261d6f5e0>
```



In [32]:

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

```
0.9999999124031285
```

RIDGE AND LASSO REGRESSION

In [33]:

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

In [34]:

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

Out[34]:

```
Ridge(alpha=10)
```

In [35]:

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

Out[35]:

```
0.9804967099777797
```

In [36]:

```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[36]:

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

```
In [37]: la.score(x_test,y_test)

Out[37]: 0.953355166182402
```