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 [41]:
```
#to import dataset
data=pd.read_csv(r"C:\Users\user\Downloads\2015 - 2015.csv")
data
```

Out[41]:

	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
...	...	...	...	...	...	...	...	...	
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

DATA PREPROCESSING AND CLEANING

In [3]:
```
data.head()
```

Out[3]:

Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual
7.587	0.03411	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678	2.51738
7.561	0.04884	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630	2.70201
7.527	0.03328	1.32548	1.36058	0.87464	0.64938	0.48357	0.34139	2.49204
7.522	0.03880	1.45900	1.33095	0.88521	0.66973	0.36503	0.34699	2.46531
7.427	0.03553	1.32629	1.32261	0.90563	0.63297	0.32957	0.45811	2.45176

In [4]:
```
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 [5]:
```
data.describe()
```

Out[5]:

	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual
158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	
0.047885	0.846137	0.991046	0.630259	0.428615	0.143422	0.237296	2.098977	
0.017146	0.403121	0.272369	0.247078	0.150693	0.120034	0.126685	0.553550	
0.018480	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.328580	
0.037268	0.545808	0.856823	0.439185	0.328330	0.061675	0.150553	1.759410	
0.043940	0.910245	1.029510	0.696705	0.435515	0.107220	0.216130	2.095415	
0.052300	1.158448	1.214405	0.811013	0.549092	0.180255	0.309883	2.462415	
0.136930	1.690420	1.402230	1.025250	0.669730	0.551910	0.795880	3.602140	

In [6]:
```
data.columns
```

Out[6]:
```
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')
```

In [7]:
```
data.isnull()
```

Out[7]:

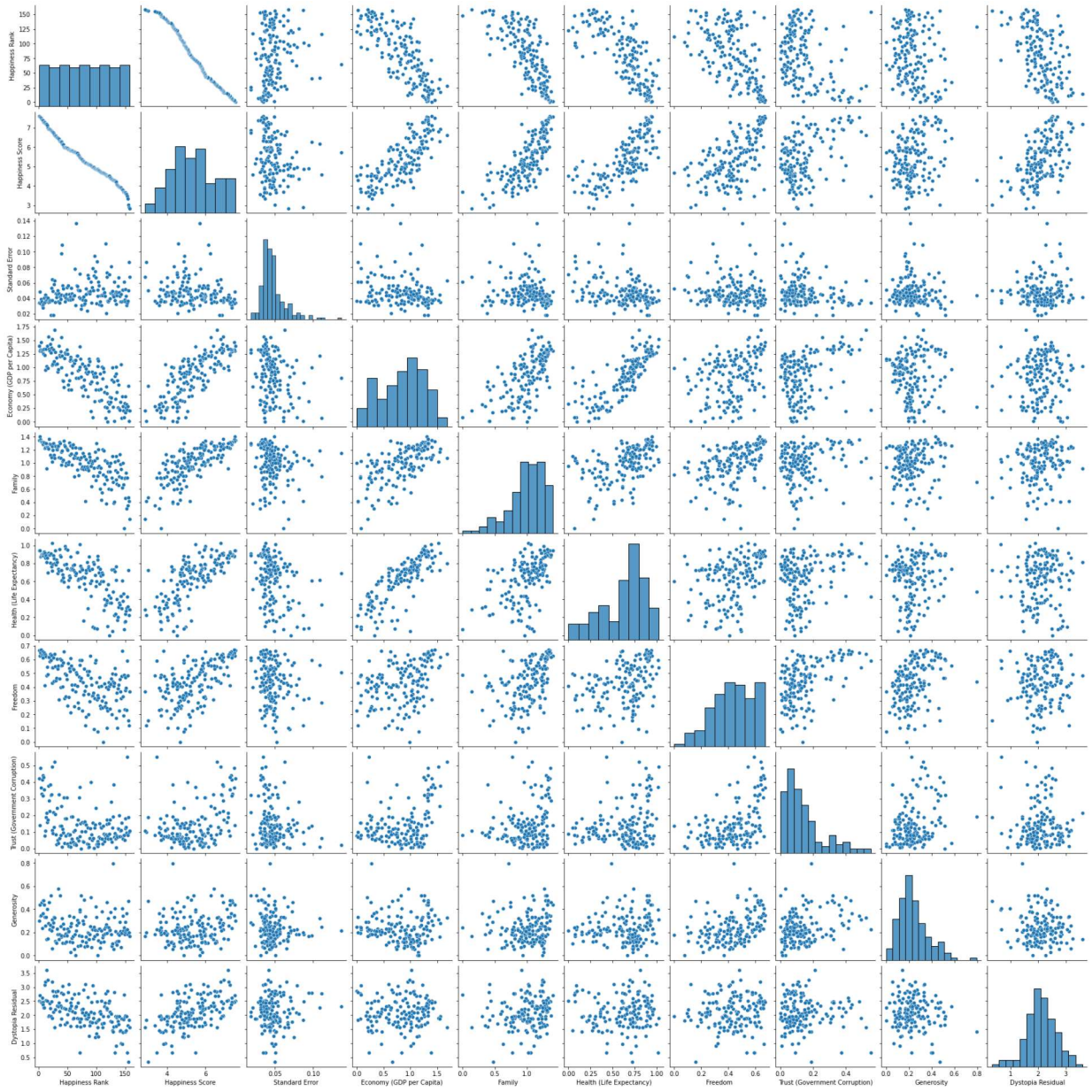
	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
153	False	False	False	False	False	False	False	False	False
154	False	False	False	False	False	False	False	False	False
155	False	False	False	False	False	False	False	False	False
156	False	False	False	False	False	False	False	False	False
157	False	False	False	False	False	False	False	False	False

158 rows × 12 columns

EDA and DATA VISUALIZATION

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

Out[8]: <seaborn.axisgrid.PairGrid at 0x18f5fb7e460>



In [39]:
```
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[39]: <AxesSubplot:xlabel='Happiness Score', ylabel='Density'>

In [40]:
```
df=data[['Happiness Rank','Standard Error',  'Family','Health (Life Expectancy)',
       'Generosity', 'Dystopia Residual']]
```

In [11]:
```
sns.heatmap(df.corr())
```

Out[11]: <AxesSubplot:>

MODEL TRAINING

In [32]:
```
x=df[['Happiness Rank','Standard Error', 'Family','Health (Life Expectancy)', 'Fr
       'Generosity', 'Dystopia Residual']]
y=df['Happiness Score']
```

In [33]:
```
#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 [34]:
```
from sklearn.linear_model import LinearRegression

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

Out[34]:
```
LinearRegression()
```

In [35]:
```
#to find intercept
print(lr.intercept_)
```

```
5.151006969019501
```

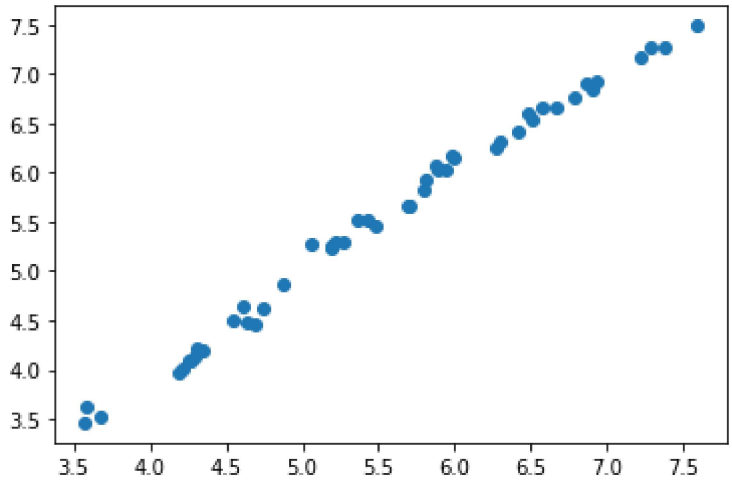
In [36]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[36]:

	Co-efficient
Happiness Rank	-0.017907
Standard Error	-0.946385
Family	0.526754
Health (Life Expectancy)	0.466867
Freedom	0.240797
Trust (Government Corruption)	0.609948
Generosity	0.210178
Dystopia Residual	0.301595

In [37]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[37]: <matplotlib.collections.PathCollection at 0x18f66adbfa0>



In [38]:
```
print(lr.score(x_test,y_test))
```

```
0.9888285108093425
```