DATA COLLECTION

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

```
In [2]:  # To Import Dataset
         sd=pd.read_csv(r"c:\Users\user\Downloads\\VE.csv")
         sd
```

Out[2]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Fre
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.6
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.6
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.6
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.6
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.6
...	...	...	...	...	...	...	...	...	
153	Rwanda	Sub-Saharan Africa	154	3.465	0.03464	0.22208	0.77370	0.42864	0.5
154	Benin	Sub-Saharan Africa	155	3.340	0.03656	0.28665	0.35386	0.31910	0.4
155	Syria	Middle East and Northern Africa	156	3.006	0.05015	0.66320	0.47489	0.72193	0.1
156	Burundi	Sub-Saharan Africa	157	2.905	0.08658	0.01530	0.41587	0.22396	0.1
157	Togo	Sub-Saharan Africa	158	2.839	0.06727	0.20868	0.13995	0.28443	0.3

158 rows × 12 columns

```
In [3]:  # to display top 10 rows
         sd.head(10)
```

Out[3]:

	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
5	Finland	Western Europe	6	7.406	0.03140	1.29025	1.31826	0.88911	0.64
6	Netherlands	Western Europe	7	7.378	0.02799	1.32944	1.28017	0.89284	0.61
7	Sweden	Western Europe	8	7.364	0.03157	1.33171	1.28907	0.91087	0.65
8	New Zealand	Australia and New Zealand	9	7.286	0.03371	1.25018	1.31967	0.90837	0.63
9	Australia	Australia and New Zealand	10	7.284	0.04083	1.33358	1.30923	0.93156	0.65

DATA CLEANING AND PRE_PROCESSING

```
In [4]:  sd.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]:  # to display summary of statistics
         sd.describe()
```

Out[5]:

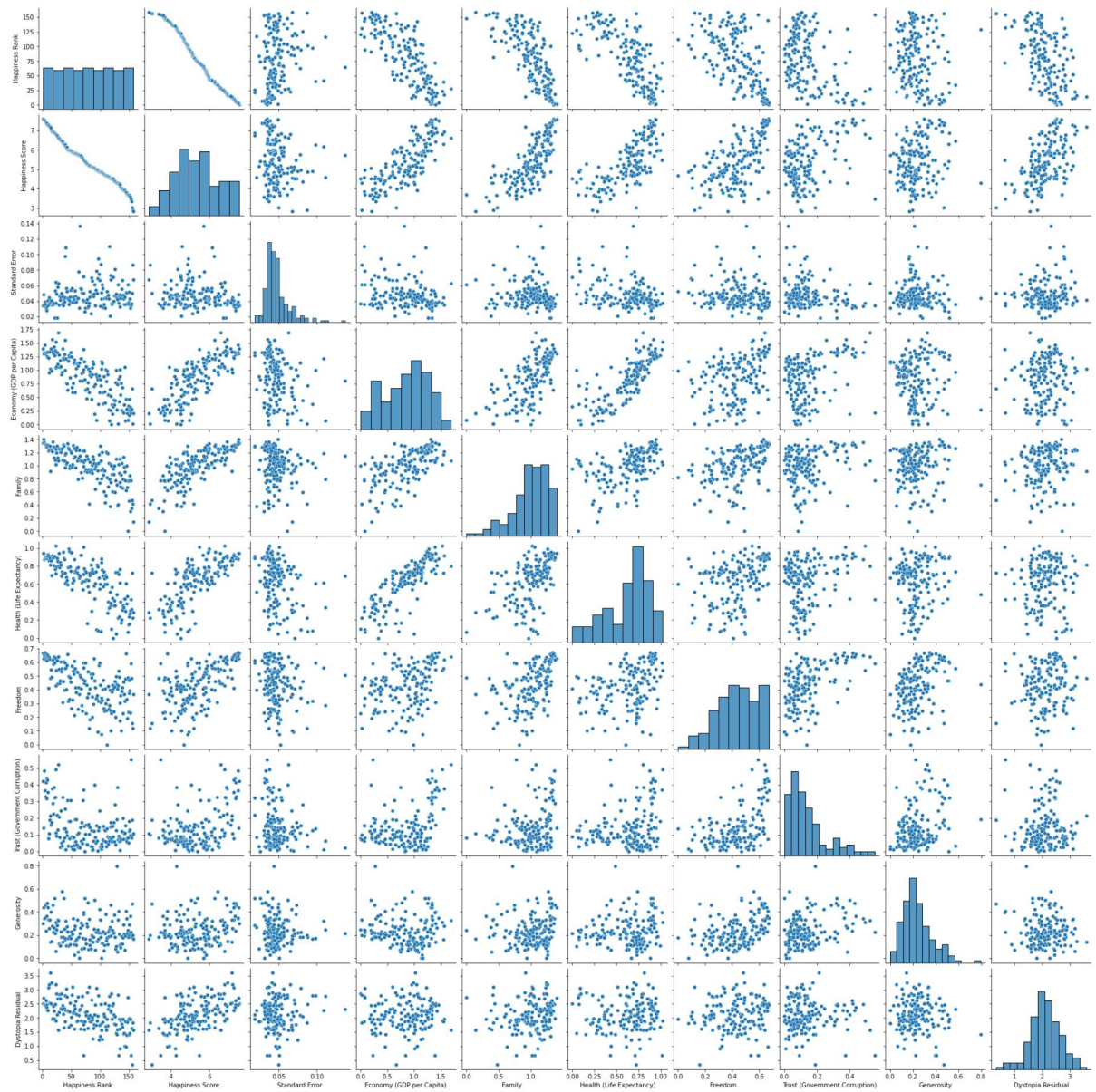|	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	(Go C(
count	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	1:
mean	79.493671	5.375734	0.047885	0.846137	0.991046	0.630259	0.428615	
std	45.754363	1.145010	0.017146	0.403121	0.272369	0.247078	0.150693	
min	1.000000	2.839000	0.018480	0.000000	0.000000	0.000000	0.000000	
25%	40.250000	4.526000	0.037268	0.545808	0.856823	0.439185	0.328330	
50%	79.500000	5.232500	0.043940	0.910245	1.029510	0.696705	0.435515	
75%	118.750000	6.243750	0.052300	1.158448	1.214405	0.811013	0.549092	
max	158.000000	7.587000	0.136930	1.690420	1.402230	1.025250	0.669730	

```
In [6]:  #to display colums heading
         sd.columns
```

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

EDA and visualization

```
In [7]: sns.pairplot(sd)
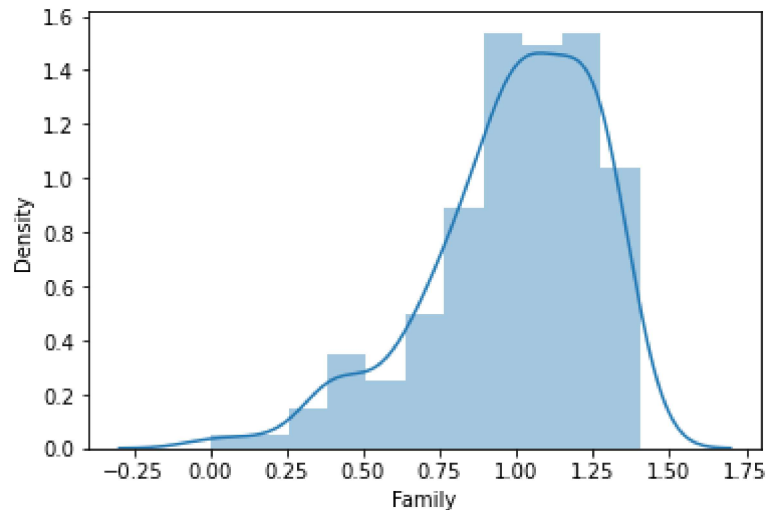```

Out[7]: <seaborn.axisgrid.PairGrid at 0x1ec4ff25790>

```
In [8]: sns.distplot(sd['Family'])
```

C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  warnings.warn(msg, FutureWarning)

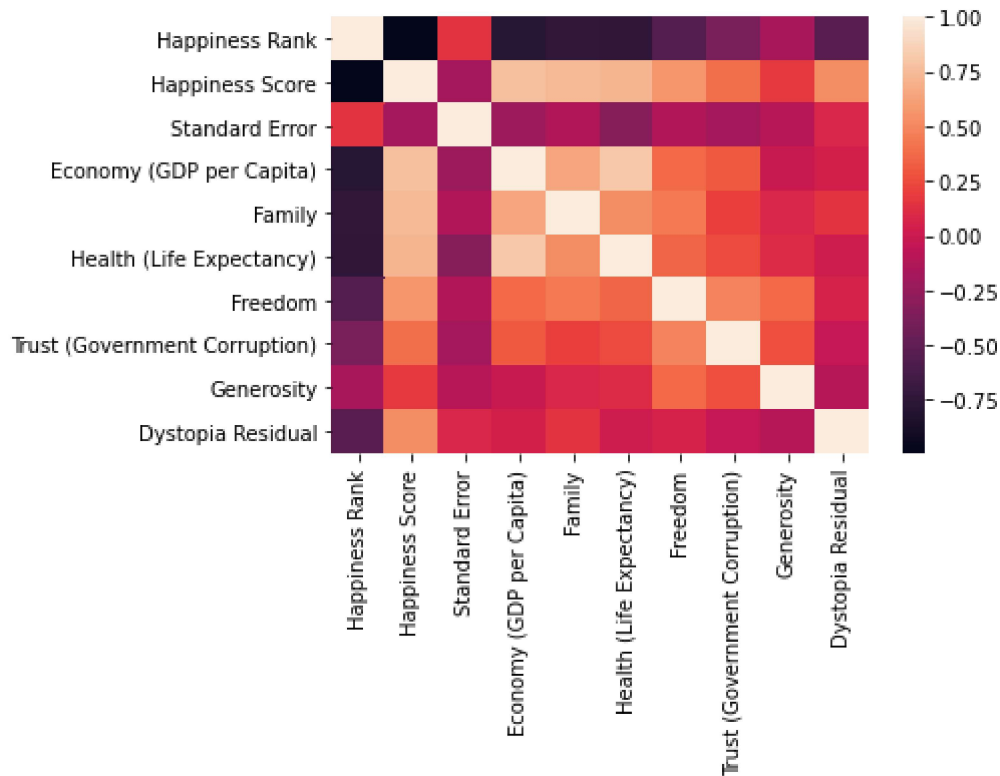Out[8]: <AxesSubplot:xlabel='Family', ylabel='Density'>


```
In [9]: sd1=sd[['Country', 'Region', 'Happiness Rank', 'Happiness Score',
            'Standard Error', 'Economy (GDP per Capita)', 'Family',
            'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
            'Generosity', 'Dystopia Residual']]
```

```
In [10]: sns.heatmap(sd1.corr())
```

Out[10]: <AxesSubplot:>

TO TRAIN THE MODEL _MODEL BUILDING

we are goint train Liner Regression model; we need to split out the data into two varibles x and y where x is independent on x (output) and y is dependent on x(output) adress coloumn as it is not required our model

```
In [11]: x= sd1[['Happiness Rank', 'Happiness Score',
         'Standard Error', 'Economy (GDP per Capita)',
         'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
         'Generosity', 'Dystopia Residual']]
         y=sd1[ 'Family']
```

```
In [12]: # To split my dataset  into training data and test data
         from sklearn .model_selection import train_test_split
         x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.4)
```

```
In [13]: from sklearn.linear_model import LinearRegression

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

Out[13]: LinearRegression()

```
In [14]: print(lr.intercept_)

         -0.001876774443453022
```

```
In [15]: coeff= pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
         coeff
```

Out[15]:

	Co-efficient
Happiness Rank	0.000006
Happiness Score	1.000219
Standard Error	0.002202
Economy (GDP per Capita)	-1.000139
Health (Life Expectancy)	-0.999665
Freedom	-0.999919
Trust (Government Corruption)	-0.999509
Generosity	-1.000194
Dystopia Residual	-0.999999

```
In [16]: prediction = lr.predict(x_test)
         plt.scatter(y_test,prediction)
```

Out[16]: <matplotlib.collections.PathCollection at 0x1ec56966e80>
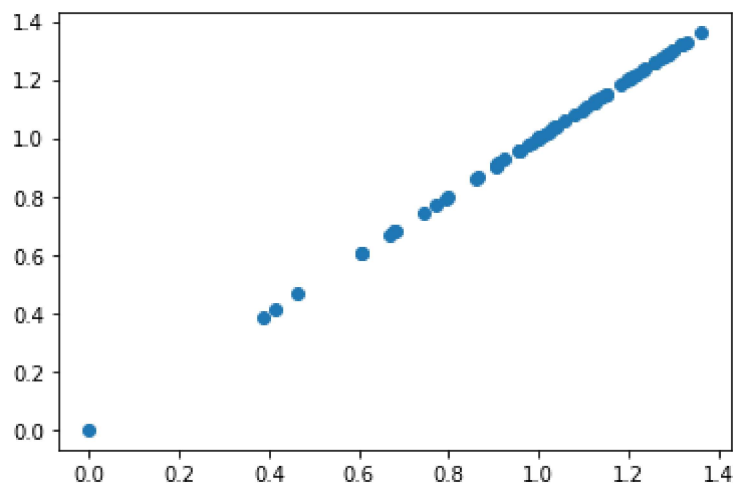

```
In [17]: print(lr.score(x_test,y_test))

         0.9999987706254002
```

```
In [18]: lr.score(x_train,y_train)
```

Out[18]: 0.9999990326559676

```
In [19]: from sklearn.linear_model import Ridge,Lasso
```

```
In [20]: dr=Ridge(alpha=10)
         dr.fit(x_train,y_train)
```

Out[20]: Ridge(alpha=10)

```
In [21]: dr.score(x_test,y_test)
```

Out[21]: 0.446512215885698

```
In [22]: dr.score(x_train,y_train)
```

Out[22]: 0.7057085250476514

```
In [23]: la=Lasso(alpha=10)
         la.fit(x_train,y_train)
```

Out[23]: Lasso(alpha=10)

```
In [24]: la.score(x_test,y_test)
```

Out[24]: -0.012842519254037787

```
In [25]: la.score(x_train,y_train)
```

Out[25]: 0.0

```
In [ ]:
```