27-07-2023

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

```
In [2]: a=pd.read_csv(r"C:\Users\user\Downloads\2_2015.csv")
        a
```

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.
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.
...	...	...	...	...	...	...	...	...	
153	Rwanda	Sub-Saharan Africa	154	3.465	0.03464	0.22208	0.77370	0.42864	0.
154	Benin	Sub-Saharan Africa	155	3.340	0.03656	0.28665	0.35386	0.31910	0.
155	Syria	Middle East and Northern Africa	156	3.006	0.05015	0.66320	0.47489	0.72193	0.
156	Burundi	Sub-Saharan Africa	157	2.905	0.08658	0.01530	0.41587	0.22396	0.
157	Togo	Sub-Saharan Africa	158	2.839	0.06727	0.20868	0.13995	0.28443	0.

158 rows × 12 columns

In [3]: `a.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 [4]: `a.columns`

Out[4]:
```
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 [5]: `a.head()`

Out[5]:

|	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	0.665
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.628
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.649
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.669
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.632

In [6]: `a.describe()`

Out[6]:

	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 [7]: `sns.pairplot(a)`

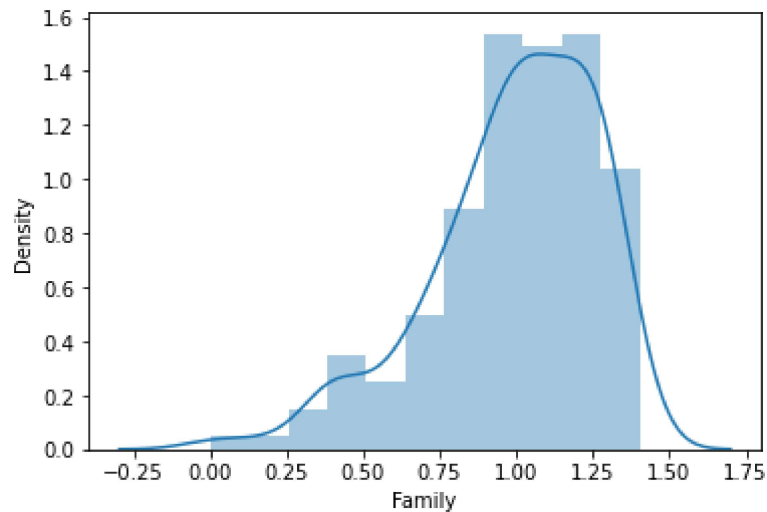Out[7]: `<seaborn.axisgrid.PairGrid at 0x25f7ba3d790>`
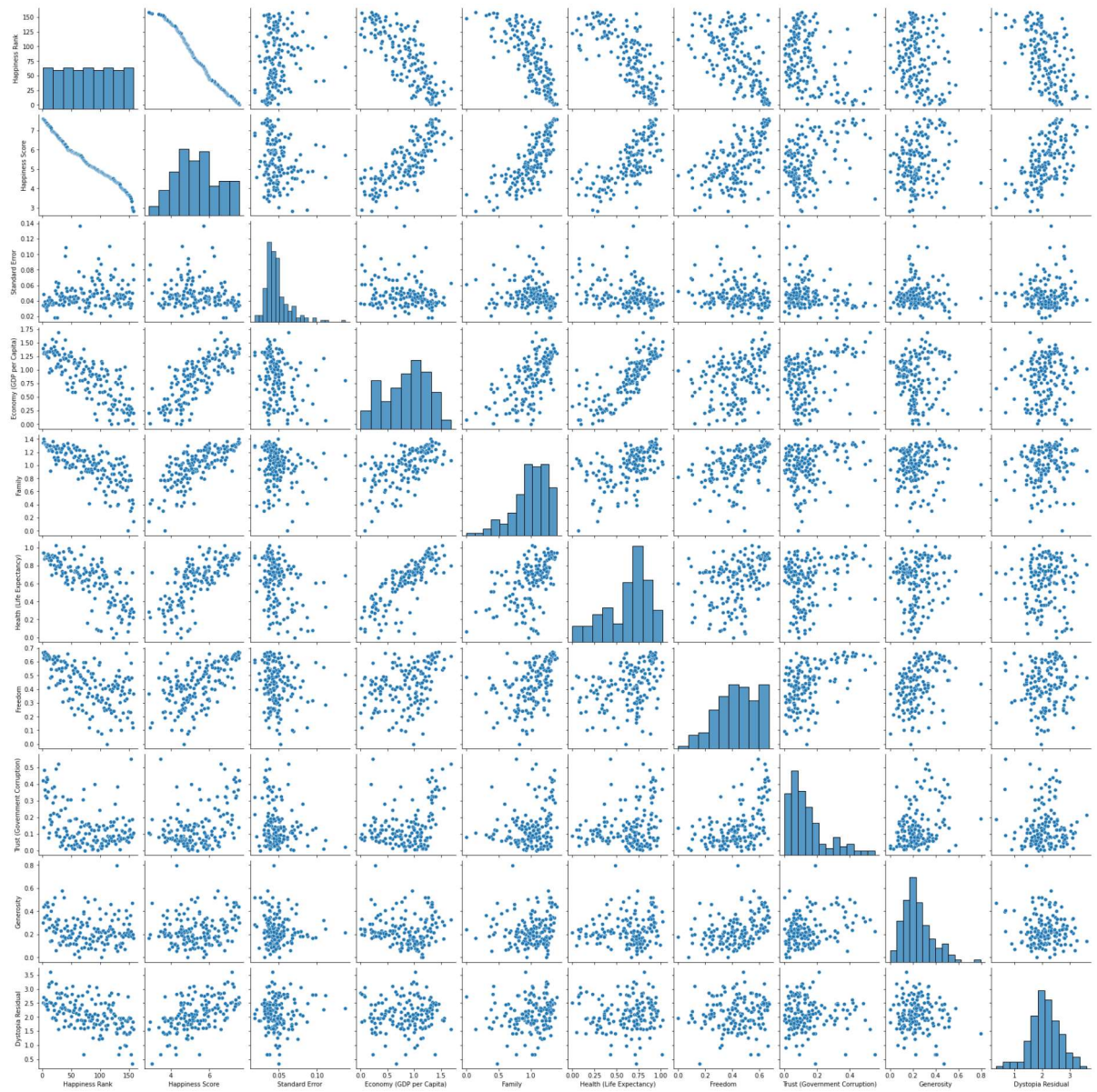
In [8]:
```python
sns.distplot(a['Family'])
```

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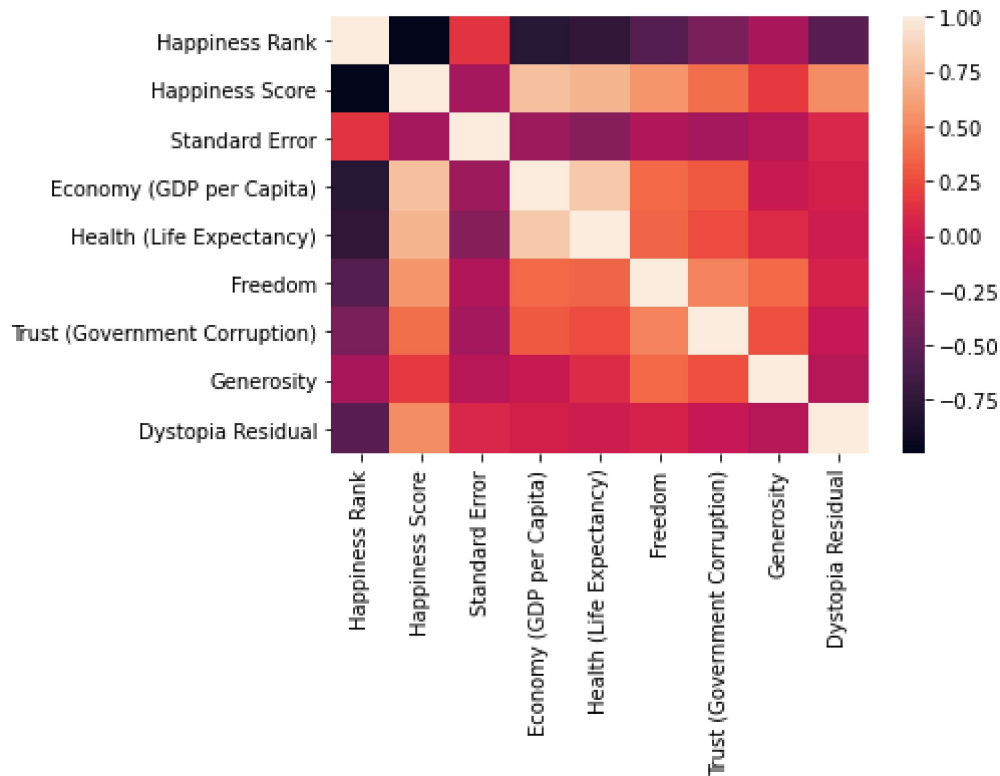
Out[8]: <AxesSubplot:xlabel='Family', ylabel='Density'>

In [28]:
```python
x1=a[['Happiness Rank', 'Happiness Score',
      'Standard Error', 'Economy (GDP per Capita)',
      'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
      'Generosity', 'Dystopia Residual']]
```

In [29]:
```python
sns.heatmap(x1.corr())
```

Out[29]:
```
<AxesSubplot:>
```


In [34]:
```python
x=a[[ 'Happiness Rank', 'Happiness Score',
      'Standard Error', 'Economy (GDP per Capita)',
      'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
      'Generosity','Dystopia Residual']]
y=a[ 'Family']
```

In [35]:
```python
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 [36]:
```python
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[36]:
```
LinearRegression()
```

In [37]:
```python
print(lr.intercept_)
```

```
-0.0015805488823738
```

In [38]:
```python
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[38]:

	Co-efficient
Happiness Rank	0.000004
Happiness Score	1.000223
Standard Error	0.000785
Economy (GDP per Capita)	-1.000113
Health (Life Expectancy)	-1.000012
Freedom	-0.999781
Trust (Government Corruption)	-0.999858
Generosity	-1.000017
Dystopia Residual	-1.000002

In [39]:
```python
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[39]: <matplotlib.collections.PathCollection at 0x25f078d45b0>
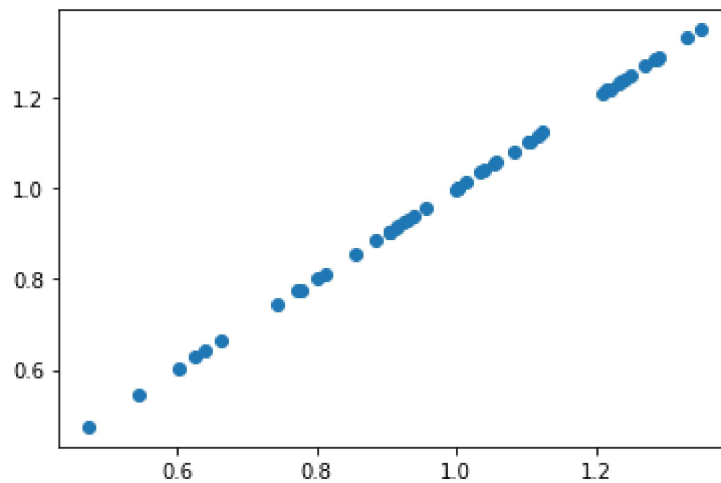
In [40]:
```python
print(lr.score(x_test,y_test))
```

0.9999982527928005