

Problem Statement:

A real estate agent want to help to predict the house price for regions in USA. He gave us the dataset to work on to use Linear Regression model. Create a Model that helps him to estimate of what the house would sell for.

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

```
In [2]: df=pd.read_csv("2015.csv")
df
```

Out[2]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62877
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64938
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.66973
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.63297
...
153	Rwanda	Sub-Saharan Africa	154	3.465	0.03464	0.22208	0.77370	0.42864	0.59201
154	Benin	Sub-Saharan Africa	155	3.340	0.03656	0.28665	0.35386	0.31910	0.48450
155	Syria	Middle East and Northern Africa	156	3.006	0.05015	0.66320	0.47489	0.72193	0.15684
156	Burundi	Sub-Saharan Africa	157	2.905	0.08658	0.01530	0.41587	0.22396	0.11850
157	Togo	Sub-Saharan Africa	158	2.839	0.06727	0.20868	0.13995	0.28443	0.36453

158 rows × 12 columns

In [3]:

`df.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]:

`df.head()`

Out[4]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	(Go C)
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557	
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62877	
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64938	
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.66973	
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.63297	

Data cleaning and Pre-Processing

In [5]:

`df.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 
```

Linear Regression-2015

```

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]:

```
df.describe()
```

Out[6]:

	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)
count	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000
mean	79.493671	5.375734	0.047885	0.846137	0.991046	0.630259	0.428615	0.14342
std	45.754363	1.145010	0.017146	0.403121	0.272369	0.247078	0.150693	0.12003
min	1.000000	2.839000	0.018480	0.000000	0.000000	0.000000	0.000000	0.000000
25%	40.250000	4.526000	0.037268	0.545808	0.856823	0.439185	0.328330	0.06167
50%	79.500000	5.232500	0.043940	0.910245	1.029510	0.696705	0.435515	0.10722
75%	118.750000	6.243750	0.052300	1.158448	1.214405	0.811013	0.549092	0.18025
max	158.000000	7.587000	0.136930	1.690420	1.402230	1.025250	0.669730	0.55191

In [7]:

```
a = df.dropna(axis='columns')
a
```

Out[7]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62877
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64938
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.66973
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.63297
...
153	Rwanda	Sub-Saharan	154	3.465	0.03464	0.22208	0.77370	0.42864	0.59201

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom
Africa									
154	Benin	Sub-Saharan Africa	155	3.340	0.03656	0.28665	0.35386	0.31910	0.48450
155	Syria	Middle East and Northern Africa	156	3.006	0.05015	0.66320	0.47489	0.72193	0.15684
156	Burundi	Sub-Saharan Africa	157	2.905	0.08658	0.01530	0.41587	0.22396	0.11850
157	Togo	Sub-Saharan Africa	158	2.839	0.06727	0.20868	0.13995	0.28443	0.36453

158 rows × 12 columns

In [8]:

`a.columns`

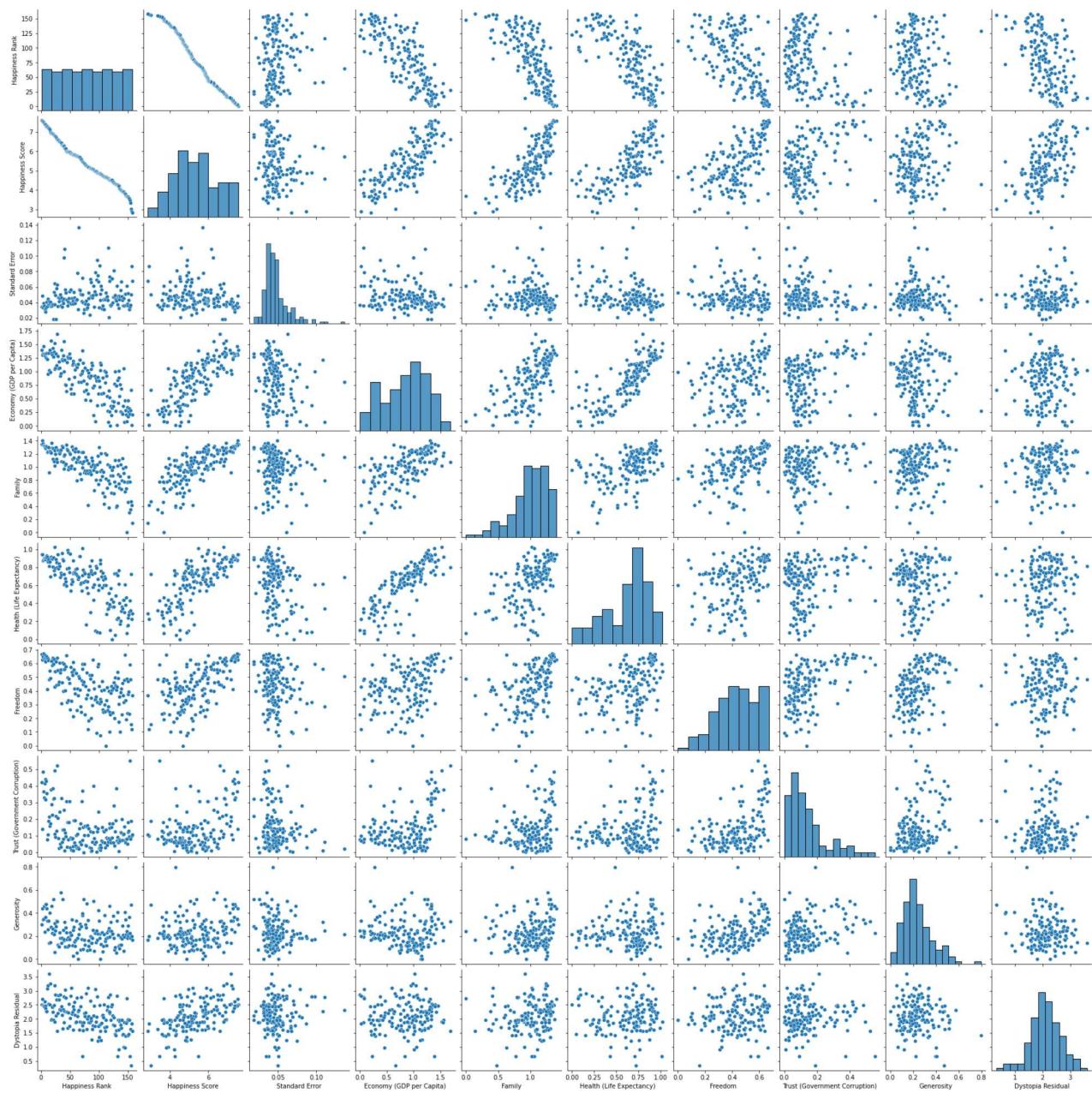
```
Out[8]: Index(['Country', 'Region', 'Happiness Rank', 'Happiness Score',
       'Standard Error', 'Economy (GDP per Capita)', 'Family',
       'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
       'Generosity', 'Dystopia Residual'],
      dtype='object')
```

EDA and VISUALIZATION

In [9]:

`sns.pairplot(df)`

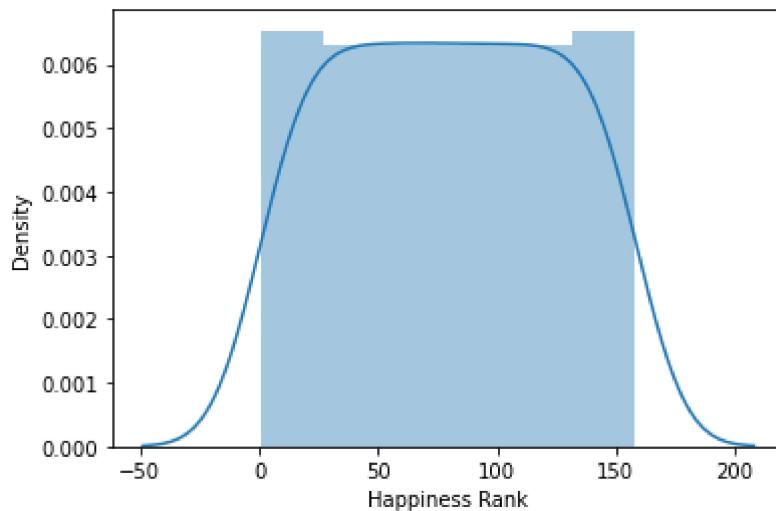
```
Out[9]: <seaborn.axisgrid.PairGrid at 0x282c70ebd90>
```



In [10]: `sns.distplot(df['Happiness Rank'])`

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

Out[10]: <AxesSubplot:xlabel='Happiness Rank', ylabel='Density'>

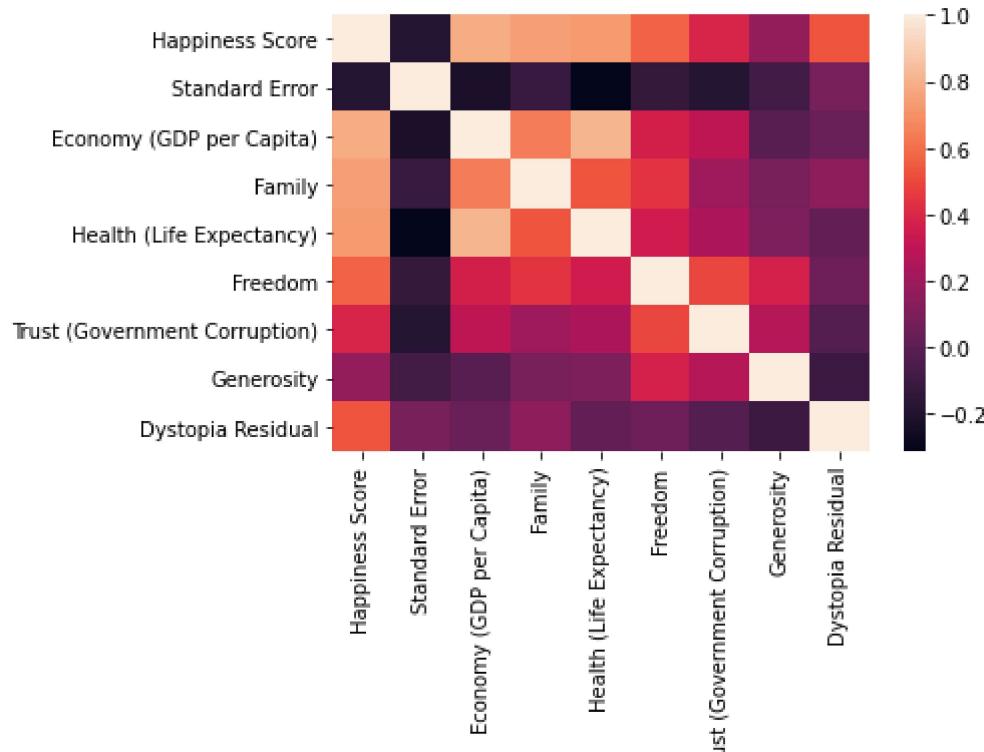


```
In [11]: df1=df[['Happiness Score',
   'Standard Error', 'Economy (GDP per Capita)', 'Family',
   'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
   'Generosity', 'Dystopia Residual']]
```

Plot Using Heat Map

```
In [12]: sns.heatmap(df1.corr())
```

```
Out[12]: <AxesSubplot:>
```



To Train The Model-Model Building

we are going to train Linear Regression Model; We need to split our data into two variables x and y where x is independent variable(input) and y is dependent on x(output) we could ignore address column as it required for our model

```
In [13]: x=df1[['Happiness Score',
           'Standard Error', 'Economy (GDP per Capita)', 'Family',
           'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
           'Generosity']]
y=df1['Dystopia Residual']
```

To Split my dataset into training and test data

```
In [14]: 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 [15]: from sklearn.linear_model import LinearRegression
lr= LinearRegression()
lr.fit(x_train,y_train)
```

Out[15]: LinearRegression()

```
In [16]: lr.intercept_
```

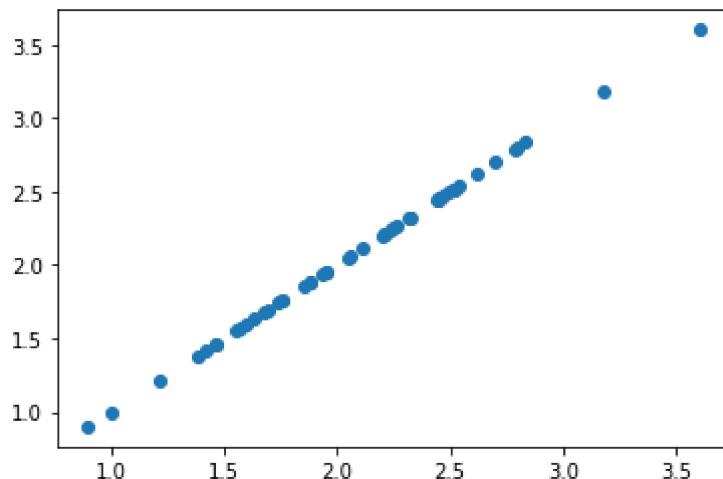
Out[16]: -0.00015856242462497505

```
In [17]: coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

	Co-efficient
Happiness Score	0.999977
Standard Error	0.000609
Economy (GDP per Capita)	-1.000146
Family	-0.999915
Health (Life Expectancy)	-0.999815
Freedom	-0.999669
Trust (Government Corruption)	-0.999948
Generosity	-0.999875

```
In [18]: prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[18]: <matplotlib.collections.PathCollection at 0x282cdb53070>



```
In [19]: lr.score(x_test,y_test)
```

```
Out[19]: 0.9999997436073631
```

```
In [20]: from sklearn.linear_model import Ridge,Lasso
```

```
In [21]: rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
rr.score(x_train,y_train)
```

```
Out[21]: 0.6146063669494978
```

```
In [22]: rr.score(x_test,y_test)
```

```
Out[22]: 0.7314300661439926
```

```
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]: -0.0021578243440882883
```