

Problem Statement

A real estate agent want 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

Import libraries

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

```
In [2]: # To import dataset
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)	Fre
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.1
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.1
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.1
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.1
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.1
...
153	Rwanda	Sub-Saharan Africa	154	3.465	0.03464	0.22208	0.77370	0.42864	0.1
154	Benin	Sub-Saharan Africa	155	3.340	0.03656	0.28665	0.35386	0.31910	0.1
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	0.1

158 rows × 12 columns



```
In [3]: # To display top 10 rows
df.head(10)
```

Out[3]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Free
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]: 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 [5]: df.describe()

Out[5]:

	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	(G 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]: df.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')

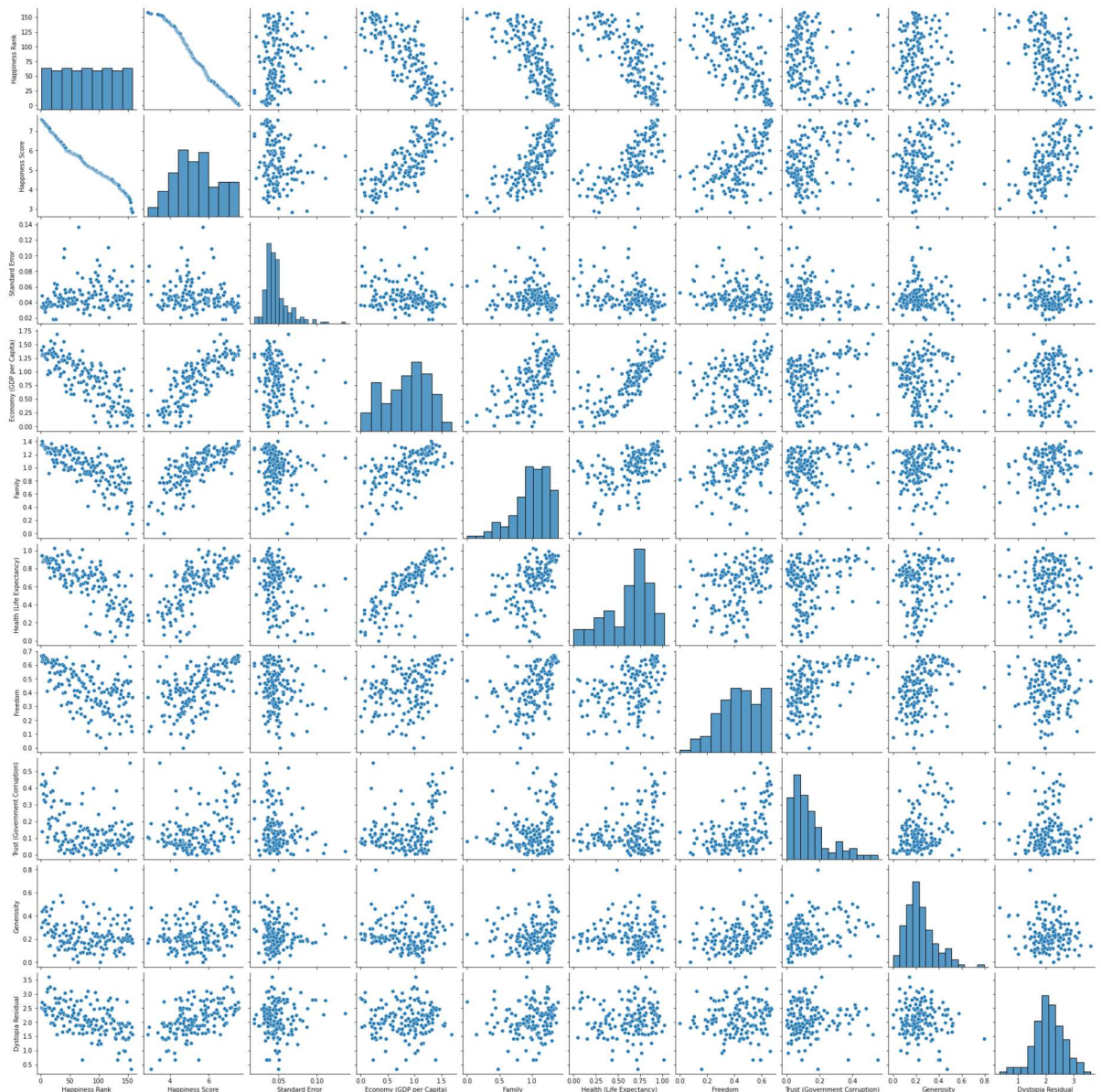
```
In [7]: a = df.dropna(axis='columns')  
a.columns
```

```
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 Visualization

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

```
Out[8]: <seaborn.axisgrid.PairGrid at 0x146d7ec3970>
```

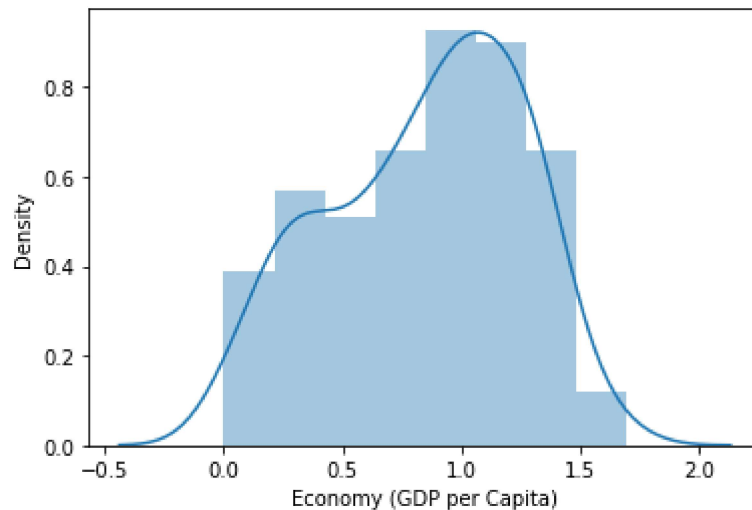


```
In [9]: sns.distplot(a['Economy (GDP per Capita)'])
```

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[9]: <AxesSubplot:xlabel='Economy (GDP per Capita)', ylabel='Density'>
```



```
In [10]: a1=a[['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(a1.corr())
```

```
Out[11]: <AxesSubplot:>
```



To Train the Model - Model Building

We are going to train Linear Regression model; We need to split out 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 is not required for our model.

```
In [12]: x=a1[['Happiness Rank', 'Happiness Score',
               'Standard Error', 'Economy (GDP per Capita)', 'Family',
               'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
               'Generosity', 'Dystopia Residual']]
y=a1['Economy (GDP per Capita)']
```

To split my dataset into training and test data

```
In [13]: 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 [14]: from sklearn.linear_model import LinearRegression
```

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

Out[14]: LinearRegression()

```
In [15]: print(lr.intercept_)
```

-5.229150445984487e-14

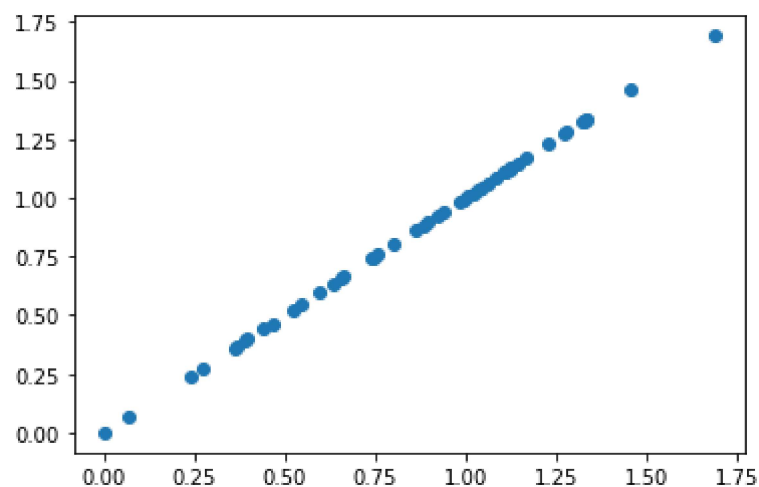
```
In [16]: coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[16]:

	Co-efficient
Happiness Rank	6.786886e-16
Happiness Score	-1.822860e-14
Standard Error	2.270041e-15
Economy (GDP per Capita)	1.000000e+00
Family	1.767644e-14
Health (Life Expectancy)	1.707975e-14
Freedom	1.792876e-14
Trust (Government Corruption)	1.871497e-14
Generosity	1.817082e-14
Dystopia Residual	1.807245e-14

```
In [17]: prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[17]: <matplotlib.collections.PathCollection at 0x146de8f71f0>



In [18]: `print(lr.score(x_test,y_test))`

1.0

In []: