Simple Linear Regression

```
# Getting the X and Y arrays
X = life_df.drop(['Country','Life_expectancy'], axis=1)
y = life_df['Life_expectancy']

# The x here is the dependent variable
# While y is the independent variable
```

```
print("X=",X.shape,"\ny=",y.shape)
```

```
X= (2938, 20)
y= (2938,)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)
```

```
"""
The values are split to train and test variables
train variables are for machine learning, and where most of the data is found
whereas the test variables are for testing whether the model is accurate
"""
```

```
'\nThe values are split to train and test variables\ntrain variables are for machine learning, and where most of the data is found\nwhereas the test variables are for testing whether the model is accurate\n'
```

```
X_train.shape
```

```
(2056, 20)
```

```
X_test.shape
```

```
(882, 20)
```

```
model = LinearRegression()
```

```
model.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
model.coef_
```

```
array([-6.02084848e-03,  1.09654477e+00, -1.93493879e-02,  1.11831785e-01,
        9.97628369e-02,  3.81053795e-05, -2.54580140e-03, -1.95327687e-05,
        4.55800328e-02, -8.42263377e-02,  2.97401314e-02,  7.13883874e-02,
        3.22091348e-02, -4.49743238e-01,  5.27130583e-05,  1.27457153e-09,
       -1.10310299e-01,  3.54105011e-02,  5.28539213e+00,  6.74403621e-01])
```

```
pd.DataFrame(model.coef_, X.columns, columns=['Coefficients'])
```

	Coefficients
Year	-6.020848e-03
Status	1.096545e+00
Adult_Mortality	-1.934939e-02
infant_deaths	1.118318e-01
Alcohol	9.976284e-02
percentage_expenditure	3.810538e-05
Hepatitis_B	-2.545801e-03
Measles	-1.953277e-05
BMI	4.558003e-02
under-five_deaths	-8.422634e-02
Polio	2.974013e-02
Total_expenditure	7.138839e-02
Diphtheria	3.220913e-02
HIV/AIDS	-4.497432e-01
GDP	5.271306e-05
Population	1.274572e-09
thinness__1-19_years	-1.103103e-01
thinness_5-9_years	3.541050e-02
Income_composition_of_resources	5.285392e+00
Schooling	6.744036e-01

```
y_pred = model.predict(X_test)
```

```
# evaluation metrics, lower is better
MAE = metrics.mean_absolute_error(y_test, y_pred)
MSE = metrics.mean_squared_error(y_test, y_pred)
RMSE = np.sqrt(MSE)
```

```
MAE
```

```
2.972399895070833
```

```
MSE
```

```
15.349007980330086
```

```
RMSE
```

```
3.917781002089076
```

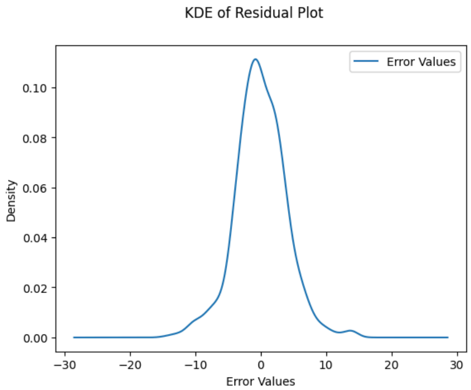
```
"""
It can be noticed that the metrics are evaluated to be
less than 20, which are low
"""
```

```
'\nIt can be noticed that the metrics are evaluated to be\nless than 1x10^-10, 1x10^-18, and 1x10^-9, which are \nvery low\n'
```

```
test_residual = y_test - y_pred
```

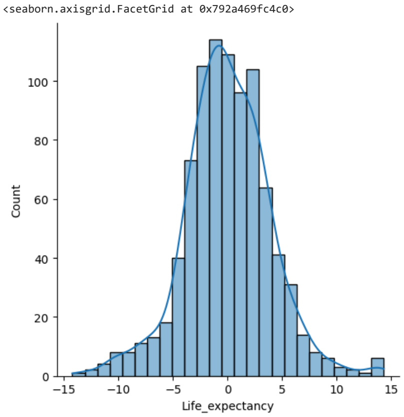
```
# as hvplot is not showing anything, I decided to just use pandas plotting
pd.DataFrame({'Error Values' : (test_residual)}).plot.kde()
plt.xlabel('Error Values')
plt.suptitle('KDE of Residual Plot')
```

```
Text(0.5, 0.98, 'KDE of Residual Plot')
```



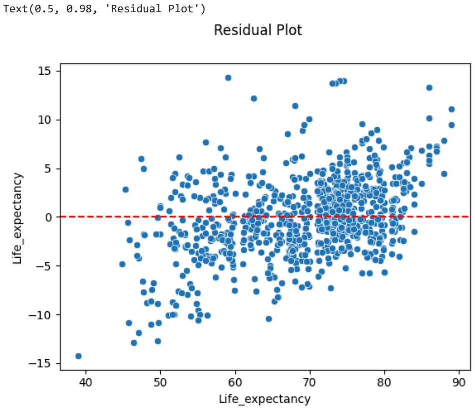

```
sns.displot(test_residual, bins=25, kde=True)
# as both of these plots somewhat follow the normal distribution,
# the linear regression is valid
```

```
<seaborn.axisgrid.FacetGrid at 0x792a469fc4c0>
```




```
sns.scatterplot(x=y_test, y=test_residual)
plt.axhline(y=0, color='r', ls='--')
plt.suptitle('Residual Plot')
# since the residual plot shows that most of the residuals are close to 0,
# and there is no pattern, the linear regression is valid
```

```
Text(0.5, 0.98, 'Residual Plot')
```




```
# comparing the test value with the predicted value
sns.scatterplot(x=y_test, y=y_pred)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.suptitle('Actual vs. Predicted Values')
# the line is somewhat linear, which means that the model is somewhat accurate
```

```
Text(0.5, 0.98, 'Actual vs. Predicted Values')
```

Actual vs. Predicted Values