Project Name : **PREDICTING LIFE EXPECTANCY USING MACHINE LEARNING**
Kickoff Date : 15-5-2020

INTRODUCTION

The project tries to create a model based on data provided by the World Health Organization (WHO) to evaluate the life expectancy for different countries in years. The data offers a timeframe from 2000 to 2015. The data originates from here: https://www.kaggle.com/kumarajarshi/life-expectancy-who/data The output algorithms have been used to test if they can maintain their accuracy in predicting the life expectancy for data they haven't been trained. Four algorithms have been used:

Linear Regression
Ridge Regression
Lasso Regression
ElasticNet Regression
Linear Regression with Polynomic features
Decision Tree Regression
Random Forest Regression

Project Scope

The project tries to create a model based on data provided by varios means to evaluate the life expectancy for different countries in years. Life expectancy can be evaluated on various data sets such as medical infrastructure, illness or disease , GDP , Sex differences, Food habbits , education etc.

This project will have two benifits, Firstly it will predict the life expectancy of the country or region and secondly we can take predictive measures or precautions for as to increse the life span of the people.

This model will make use of different regression techniques for prediction.

LITERATURE SURVEY

Although there have been lot of studies undertaken in the past on factors affecting life expectancy considering demographic variables, income composition and mortality rates. It was found that affect of immunization and human development index was not taken into account in the past. Also, some of the past research was done considering multiple linear regression based on data set of one year for all the countries. Hence, this gives motivation to resolve both the factors stated previously by formulating a regression model based on mixed effects model and multiple linear regression while considering data from a period of 2000 to 2015 for all the countries. Important immunization like Hepatitis B, Polio and Diphtheria will also be considered. In a nutshell, this study will focus on immunization factors, mortality factors, economic factors, social factors and other health related factors as well. Since the observations this dataset are based on different countries, it will be easier for a country to determine the predicting factor which is contributing to lower value of life expectancy. This will help in suggesting a country which area should be given importance in order to efficiently improve the life expectancy of its population.

SOURCE CODE

In [97]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
%matplotlib inline
```

Check out the Data

In [98]:

```
import types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
client_b26d70c0e00f4950a6461edd5e8d8074 = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='xUfWi3H7ozyf1G8E1c-I8CgTVeR8C_EKLebnGjsIvYXi',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.eu-geo.objectstorage.service.networklayer.com')

body =
client_b26d70c0e00f4950a6461edd5e8d8074.get_object(Bucket='lifeexpectancy-donotdelete-pr-hcdjuwhwpr2vis',Key='lifeExpectancy.csv')['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

# If you are reading an Excel file into a pandas DataFrame, replace `read_csv` by `read_excel` in the
next statement.

df_data_1 = pd.read_csv(body)
df_data_1.head()
```

Out[98]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	...	Polio	Total expenditure	Diphtheria	HIV/AIDS	GDP	Population	thinness 1-19 years	thinness 5-9 years	Income composition of resources	Schooling
0	Afg	2	Dev	65.0	26	6	0.	71.2	65	11	..	6	8.16	65	0.1	584.	337	17.	17.	0.47	10.

	hanistan	015	eloping		3	2	01	79624		54	.					259210	36494.0	2	3	9	1
1	Afghanistan	2014	Developing	59.9	271	64	0.01	73.523582	62	492	...	58	8.18	62	0.1	612.696514	327582.0	17.5	17.5	0.476	10.0
2	Afghanistan	2013	Developing	59.9	268	66	0.01	73.219243	64	430	...	62	8.13	64	0.1	631.744976	31731688.0	17.7	17.7	0.470	9.9
3	Afghanistan	2012	Developing	59.5	272	69	0.01	78.184215	67	2787	...	67	8.52	67	0.1	669.959000	3696958.0	17.9	18.0	0.463	9.8
4	Afghanistan	2011	Developing	59.2	275	71	0.01	7.097109	68	3013	...	68	7.87	68	0.1	63.537231	2978599.0	18.2	18.2	0.454	9.5

5 rows × 22 columns

In [ ]:

In [ ]:

In [99]:

```
df_data_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2938 entries, 0 to 2937
Data columns (total 22 columns):
Country                            2938 non-null object
Year                               2938 non-null int64
Status                             2938 non-null object
Life expectancy                    2938 non-null float64
Adult Mortality                    2938 non-null int64
infant deaths                      2938 non-null int64
Alcohol                            2938 non-null float64
percentage expenditure             2938 non-null float64
Hepatitis B                        2938 non-null int64
Measles                            2938 non-null int64
 BMI                               2938 non-null float64
under-five deaths                  2938 non-null int64
Polio                              2938 non-null int64
```

```
Total expenditure                2938 non-null float64
Diphtheria                       2938 non-null int64
 HIV/AIDS                        2938 non-null float64
GDP                              2938 non-null float64
Population                       2938 non-null float64
 thinness  1-19 years            2938 non-null float64
 thinness 5-9 years              2938 non-null float64
Income composition of resources  2938 non-null float64
Schooling                        2938 non-null float64
dtypes: float64(12), int64(8), object(2)
memory usage: 505.0+ KB
```

In [100]:

```
df_data_1.describe()
```

Out[100]:

	Year	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	BMI	under-five deaths	Polio	Total expenditure	Diphtheria	HIV/AIDS	GDP	Population	thinness 1-19 years	thinness 5-9 years	Income composition of resources	Schooling
count	2938.000000	2938.000000	2938.000000	2938.000000	2938.000000	2938.000000	2938.000000	2938.000000	2938.000000	2938.000000	2938.000000	2938.000000	2938.000000	2938.000000	2938.000000	2.938000e+03	2938.000000	2938.000000	2938.000000	2938.000000
mean	2007.518720	69.007318	164.816542	30.303948	4.606062	738.251295	75.683799	2419.592240	38.020150	42.035739	82.307692	5.905211	82.075221	1.742103	7475.593613	1.316870e+07	4.955956	5.002553	0.631418	11.935671
std	4.613841	10.252962	124.433748	117.926501	4.045598	1987.914858	28.851806	11467.272489	20.175077	160.445548	23.636677	2.484620	23.917022	5.077785	13728.461983	5.629984e+07	4.541403	4.670154	0.210920	3.340202
mi	2000.	0.000	1.000	0.000	0.010	0.0000	1.000	0.00000	1.000	0.000	3.000	0.370	2.000	0.100	1.68135	3.4000	0.100	0.100	0.000	0.000

n	000000	000	000	000	000	00	000	0	000	000	000	000	000	000	0	00e+01	000	000	000	000
25%	2004.000000	63.000000	74.000000	0.000000	0.880000	4.685343	66.000000	0.000000	19.000000	0.000000	77.000000	4.260000	78.000000	0.100000	456.766527	1.364255e+05	1.600000	1.600000	0.494000	10.100000
50%	2008.000000	72.000000	144.000000	3.000000	3.765000	64.912906	89.000000	17.000000	43.000000	4.000000	93.000000	5.710000	93.000000	0.100000	1680.834893	1.289898e+06	3.400000	3.400000	0.684000	12.300000
75%	2012.000000	75.600000	228.000000	22.000000	7.650000	441.534144	96.000000	360.250000	56.100000	28.000000	97.000000	7.400000	97.000000	0.800000	6454.061607	7.394106e+06	7.300000	7.300000	0.791000	14.100000
max	2015.000000	89.000000	723.000000	1800.000000	17.870000	19479.911610	99.000000	212183.000000	87.300000	2500.000000	99.000000	17.600000	99.000000	50.600000	119172.741800	1.293859e+09	27.700000	28.600000	0.948000	20.700000

In [101]:

```
df_data_1.columns
```

Out[101]:

```
Index(['Country', 'Year', 'Status', 'Life expectancy ', 'Adult Mortality',
       'infant deaths', 'Alcohol', 'percentage expenditure', 'Hepatitis B',
       'Measles ', ' BMI ', 'under-five deaths ', 'Polio', 'Total expenditure',
       'Diphtheria ', ' HIV/AIDS', 'GDP', 'Population',
       ' thinness  1-19 years', ' thinness 5-9 years',
       'Income composition of resources', 'Schooling'],
      dtype='object')
```

EDA

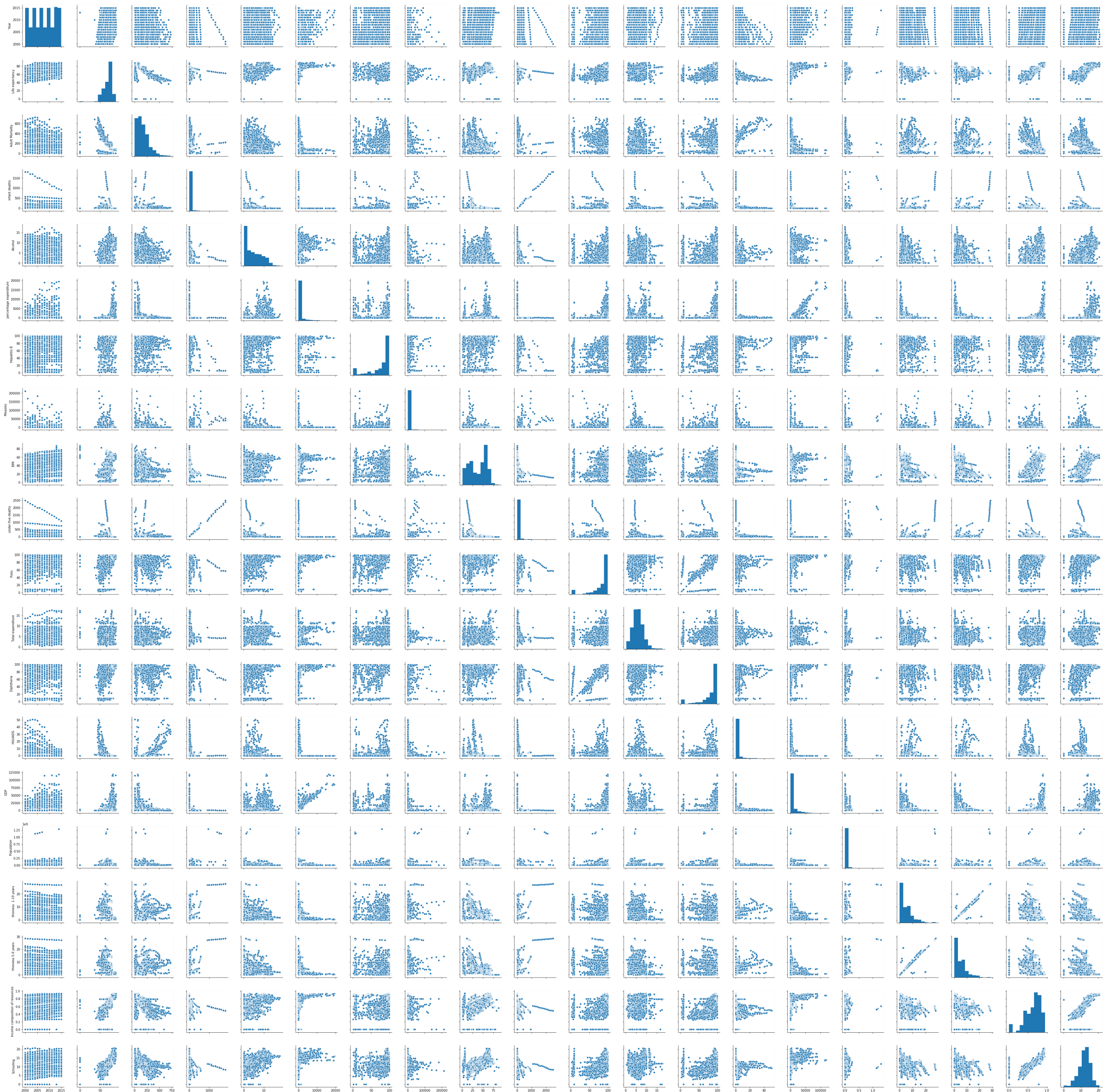
Let's create some simple plots to check out the data!

In [102]:

```
sns.pairplot(df_data_1)
```

Out[102]:

```
<seaborn.axisgrid.PairGrid at 0x7f3c4b3564e0>
```

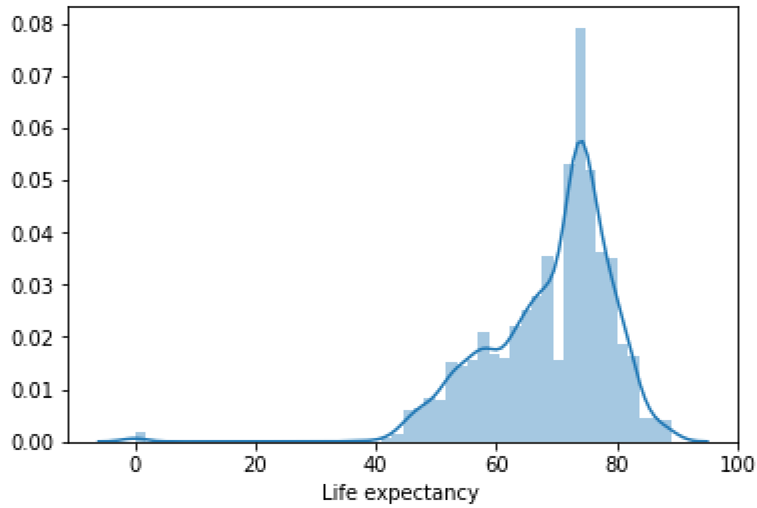


In [103]:

```
sns.distplot(df_data_1['Life expectancy '])
```

Out[103]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f3c3cc7dcf8>

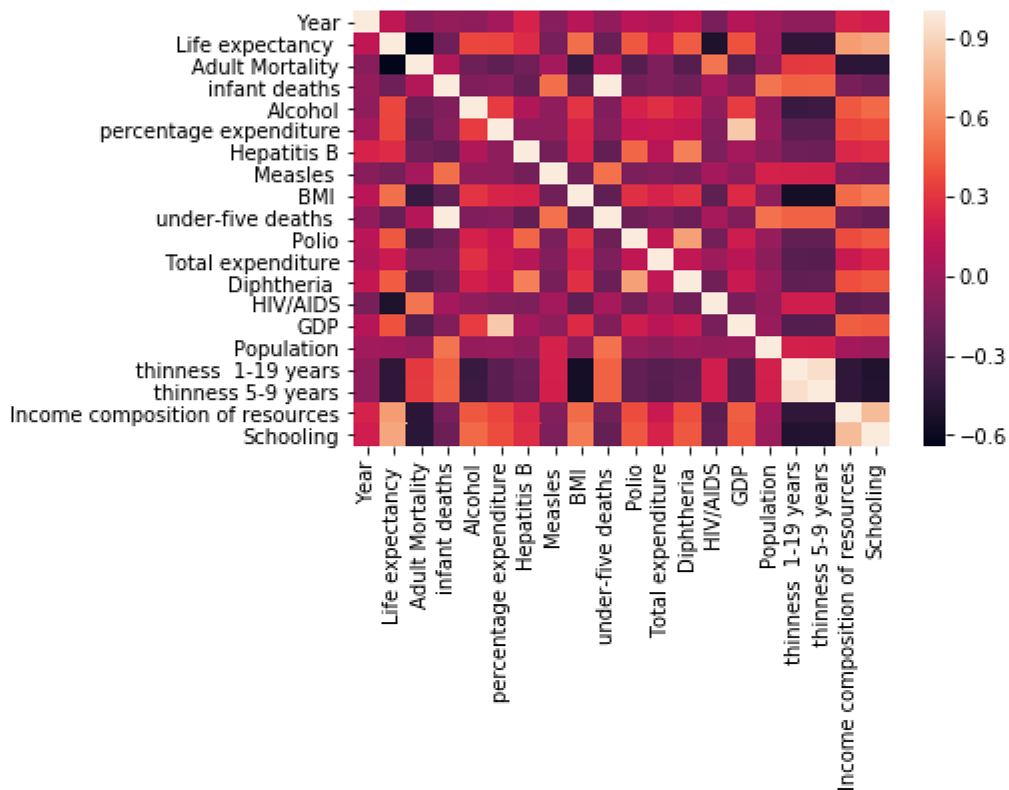


In [104]:

```
sns.heatmap(df_data_1.corr())
```

Out[104]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f3c3ca80fd0>



Training a Linear Regression Model

Let's now begin to train out regression model! We will need to first split up our data into an X array that contains the features to train on, and a y array with the target variable, in this case the Price column. We will toss out the Address column because it only has text info that the linear regression model can't use.

X and y arrays

In [111]:

```
X = df_data_1[['Year','Adult Mortality',
       'infant deaths', 'Alcohol', 'percentage expenditure', 'Hepatitis B',
       'Measles ', ' BMI ', 'under-five deaths ', 'Polio', 'Total expenditure',
       'Diphtheria ', ' HIV/AIDS', 'GDP', 'Population',
       ' thinness  1-19 years', ' thinness 5-9 years',
       'Income composition of resources', 'Schooling']]
y = df_data_1['Life expectancy ']
```

Train Test Split

Now let's split the data into a training set and a testing set. We will train out model on the training set and then use the test set to evaluate the model.

In [112]:

```
from sklearn.model_selection import train_test_split
print(df_data_1.shape)
```

```
(2938, 22)
```

In [113]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=101)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(1762, 19)
(1762,)
(1176, 19)
(1176,)
```

Creating and Training the Model

In [114]:

```
from sklearn.linear_model import LinearRegression
```

In [115]:

```
lm = LinearRegression()
```

In [116]:

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

Out[116]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

Model Evaluation

Let's evaluate the model by checking out it's coefficients and how we can interpret them.

In [117]:

```
# print the intercept
print(lm.intercept_)
```

218.95038401492707

In [118]:

```
coeff_df = pd.DataFrame(lm.coef_,X.columns,columns=['Coefficient'])
coeff_df
```

Out[118]:

	Coefficient
Year	-8.358939e-02
Adult Mortality	-1.737777e-02
infant deaths	1.105582e-01
Alcohol	1.286120e-01
percentage expenditure	1.281630e-04
Hepatitis B	-5.736524e-03
Measles	-3.824968e-05
BMI	1.299507e-02
under-five deaths	-8.483045e-02
Polio	2.137096e-02
Total expenditure	1.015781e-01
Diphtheria	3.010966e-02
HIV/AIDS	-4.673176e-01
GDP	4.133842e-05
Population	1.997925e-09
thinness 1-19 years	-5.550394e-02
thinness 5-9 years	7.192970e-02

Income composition of resources	7.864402e+00
Schooling	9.083642e-01

Interpreting the coefficients:

- Holding all other features fixed, a 1 unit increase in **Avg. Area Income** is associated with an **increase of $21.52** .
- Holding all other features fixed, a 1 unit increase in **Avg. Area House Age** is associated with an **increase of $164883.28** .
- Holding all other features fixed, a 1 unit increase in **Avg. Area Number of Rooms** is associated with an **increase of $122368.67** .
- Holding all other features fixed, a 1 unit increase in **Avg. Area Number of Bedrooms** is associated with an **increase of $2233.80** .
- Holding all other features fixed, a 1 unit increase in **Area Population** is associated with an **increase of $15.15** .

Does this make sense? Probably not because I made up this data. If you want real data to repeat this sort of analysis, check out the boston dataset:

```
from sklearn.datasets import load_boston
boston = load_boston()
print(boston.DESCR)
boston_df = boston.data
```

Predictions from our Model

Let's grab predictions off our test set and see how well it did!

In [119]:

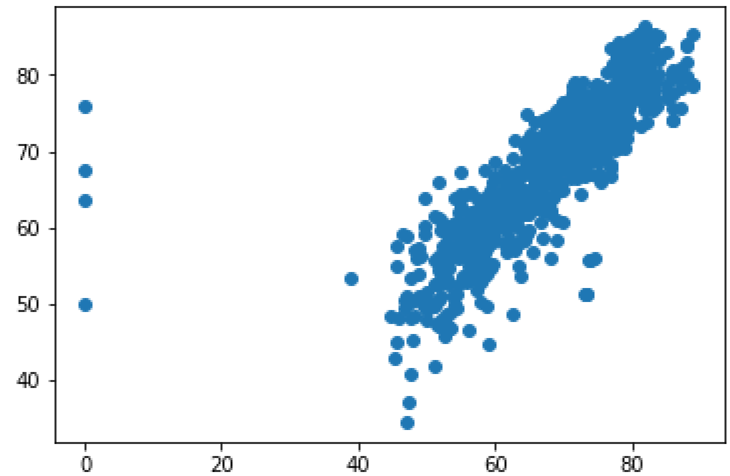
```
predictions = lm.predict(X_test)
```

In [120]:

```
plt.scatter(y_test,predictions)
```

Out[120]:

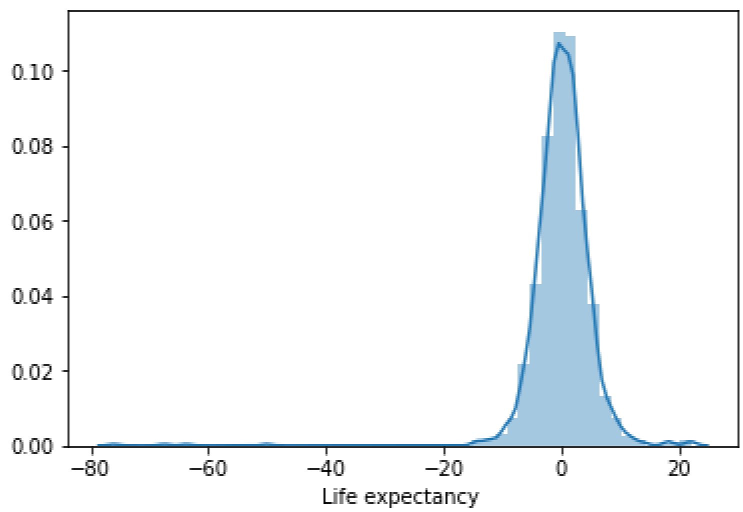
```
<matplotlib.collections.PathCollection at 0x7f3c38b2fb00>
```



Residual Histogram

In [121]:

```
sns.distplot((y_test-predictions),bins=50);
```



Regression Evaluation Metrics

Here are three common evaluation metrics for regression problems:

Mean Absolute Error (MAE) is the mean of the absolute value of the errors:

$$\frac{1}{n}\sum_{i=1}^{n}|y_i-\hat{y}_i|$$

Mean Squared Error (MSE) is the mean of the squared errors:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2$$

Root Mean Squared Error (RMSE) is the square root of the mean of the squared errors:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}$$

Comparing these metrics:

- **MAE** is the easiest to understand, because it's the average error.
- **MSE** is more popular than MAE, because MSE "punishes" larger errors, which tends to be useful in

the real world.
- **RMSE** is even more popular than MSE, because RMSE is interpretable in the "y" units.

All of these are **loss functions**, because we want to minimize them.

In [122]:

```
from sklearn import metrics
```

In [123]:

```
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
#print(metrics.confusion_matrix(y_test,predictions))
```

```
MAE: 3.2783474897107308
MSE: 31.30027608427658
RMSE: 5.594664966222426
```

In [ ]:

Write the following code as is

In [124]:

```
!pip install watson-machine-learning-client
```

```
Requirement already satisfied: watson-machine-learning-client in
/opt/conda/envs/Python36/lib/python3.6/site-packages (1.0.376)
Requirement already satisfied: requests in /opt/conda/envs/Python36/lib/python3.6/site-packages
(from watson-machine-learning-client) (2.21.0)
Requirement already satisfied: lomond in /opt/conda/envs/Python36/lib/python3.6/site-packages
(from watson-machine-learning-client) (0.3.3)
Requirement already satisfied: pandas in /opt/conda/envs/Python36/lib/python3.6/site-packages
(from watson-machine-learning-client) (0.24.1)
Requirement already satisfied: tqdm in /opt/conda/envs/Python36/lib/python3.6/site-packages
(from watson-machine-learning-client) (4.31.1)
Requirement already satisfied: ibm-cos-sdk in
/opt/conda/envs/Python36/lib/python3.6/site-packages (from watson-machine-learning-client)
(2.4.3)
Requirement already satisfied: urllib3 in /opt/conda/envs/Python36/lib/python3.6/site-packages
(from watson-machine-learning-client) (1.24.1)
Requirement already satisfied: certifi in /opt/conda/envs/Python36/lib/python3.6/site-packages
(from watson-machine-learning-client) (2020.4.5.1)
Requirement already satisfied: tabulate in /opt/conda/envs/Python36/lib/python3.6/site-packages
(from watson-machine-learning-client) (0.8.2)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in
/opt/conda/envs/Python36/lib/python3.6/site-packages (from
requests->watson-machine-learning-client) (3.0.4)
Requirement already satisfied: idna<2.9,>=2.5 in
```

```
/opt/conda/envs/Python36/lib/python3.6/site-packages (from
requests->watson-machine-learning-client) (2.8)
Requirement already satisfied: six>=1.10.0 in
/opt/conda/envs/Python36/lib/python3.6/site-packages (from
lomond->watson-machine-learning-client) (1.12.0)
Requirement already satisfied: pytz>=2011k in
/opt/conda/envs/Python36/lib/python3.6/site-packages (from
pandas->watson-machine-learning-client) (2018.9)
Requirement already satisfied: python-dateutil>=2.5.0 in
/opt/conda/envs/Python36/lib/python3.6/site-packages (from
pandas->watson-machine-learning-client) (2.7.5)
Requirement already satisfied: numpy>=1.12.0 in
/opt/conda/envs/Python36/lib/python3.6/site-packages (from
pandas->watson-machine-learning-client) (1.15.4)
Requirement already satisfied: ibm-cos-sdk-s3transfer==2.*,>=2.0.0 in
/opt/conda/envs/Python36/lib/python3.6/site-packages (from
ibm-cos-sdk->watson-machine-learning-client) (2.4.3)
Requirement already satisfied: ibm-cos-sdk-core==2.*,>=2.0.0 in
/opt/conda/envs/Python36/lib/python3.6/site-packages (from
ibm-cos-sdk->watson-machine-learning-client) (2.4.3)
Requirement already satisfied: docutils>=0.10 in
/opt/conda/envs/Python36/lib/python3.6/site-packages (from
ibm-cos-sdk-core==2.*,>=2.0.0->ibm-cos-sdk->watson-machine-learning-client) (0.14)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in
/opt/conda/envs/Python36/lib/python3.6/site-packages (from
ibm-cos-sdk-core==2.*,>=2.0.0->ibm-cos-sdk->watson-machine-learning-client) (0.9.3)
```

In [125]:

```
from watson_machine_learning_client import WatsonMachineLearningAPIClient
```

Change the following cell to match your credentials

In [126]:

```
wml_credentials={
  "apikey": "tjPDYet_pYtfXiEGTa_dNbWxdAizYsCCAET3ixzeYFON",
  "instance_id": "98f61133-01b0-4ebe-81b5-717f1979099d",
  "url": "https://eu-gb.ml.cloud.ibm.com"
}
```

In [ ]:

```

```

In [127]:

```
client = WatsonMachineLearningAPIClient( wml_credentials )
```

In [128]:

```
model_props = {client.repository.ModelMetaNames.AUTHOR_NAME: "Saikat",
```

```
        client.repository.ModelMetaNames.AUTHOR_EMAIL: "saikat.duttaroy@somaiya.edu",
        client.repository.ModelMetaNames.NAME: "Life Expectancy"}
```

In the following cell, change 'lm' to whatever your regression varialbe is.

In [129]:

```
model_artifact =client.repository.store_model(lm, meta_props=model_props)
```

copy paste the following cells as - is

In [130]:

```
published_model_uid = client.repository.get_model_uid(model_artifact)
```

In [131]:

```
published_model_uid
```

Out[131]:

'b47c4340-6303-424e-8810-3dbb5352e5be'

In [ ]:

```

```

In [ ]:

```

```

In [132]:

```
deployment = client.deployments.create(published_model_uid, name="Life Expectancy")
```

```
#######################################################################################

Synchronous deployment creation for uid: 'b47c4340-6303-424e-8810-3dbb5352e5be' started

#######################################################################################


INITIALIZING
DEPLOY_SUCCESS


------------------------------------------------------------------------------------------------
Successfully finished deployment creation, deployment_uid='127ff3ad-8337-4e77-acde-36244ccd2512'
------------------------------------------------------------------------------------------------
```

In [133]:

In [134]:

```
scoring_endpoint
```

Out[134]:

```
'https://eu-gb.ml.cloud.ibm.com/v3/wml_instances/98f61133-01b0-4ebe-81b5-717f1979099d/deplo
yments/127ff3ad-8337-4e77-acde-36244ccd2512/online'
```

In [ ]:

```

```

In [ ]:

```

```

CONCLUSION

After comparing all the algorithms we can conclude the Lasso and the Elastic Net Regression offer which are the same:

1. Best Parameters: {‘alpha’: 0, ‘max_iter’: 10}
2. R square on the test data of 92%
3. MAE of 1.83
4. MSE of 6.05