Artificial Intelligence
Lab 4

Isabella Quicaño
Raúl López

E.D.A

1.0.a) Presence of null variables in:

```
In [7]:
for i in range(0,len(life_expectancy_df.isnull().sum())):
    if life_expectancy_df.isnull().sum()[i] == 0:
        continue
    else :
        print(life_expectancy_df.columns[i],life_expectancy_df.isnull().sum()[i])
```

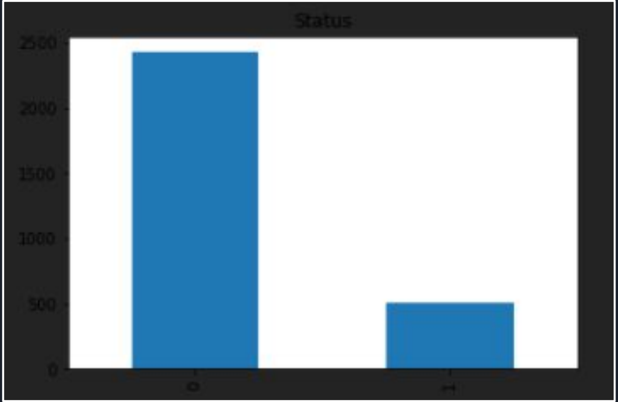
```
life_expectancy 10
adult_mortality 10
alcohol 194
hepatitis_b 553
bmi 34
polio 19
total_expenditure 226
diphtheria 19
gdp 448
population 652
thinness_10-19_years 34
thinness_5-9_years 34
income_composition_of_resources 167
schooling 163
```

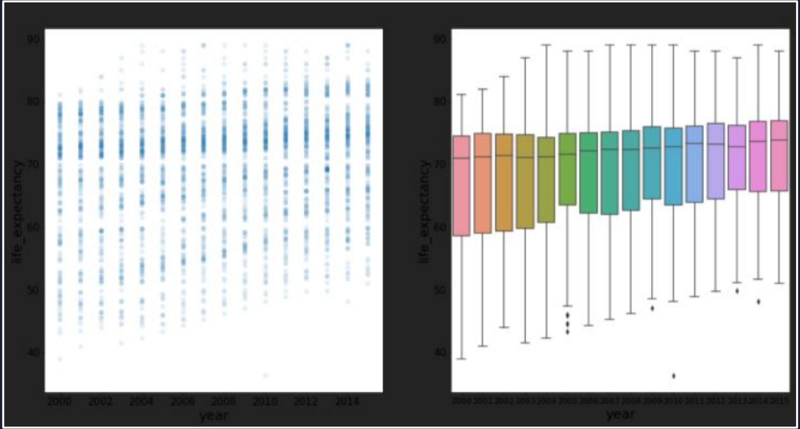
1.0.b) Impute replacing with mean

```
In [12]:
# basic operations: you can do better here
life_expectancy_df = life_expectancy_df.fillna(life_expectancy_df.mean())
```

```
In [13]:
for i in range(0,len(life_expectancy_df.isnull().sum())):
    if life_expectancy_df.isnull().sum()[i] == 0:
        continue
    else :
        print(life_expectancy_df.columns[i],life_expectancy_df.isnull().sum()[i])

print('No null values')
```

```
No null values
```





E.D.A

In [21]:
```python
life_expectancy_df['country'].unique()
```

```
array(['Afghanistan', 'Albania', 'Algeria', 'Angola',
       'Antigua and Barbuda', 'Argentina', 'Armenia', 'Australia',
       'Austria', 'Azerbaijan', 'Bahamas', 'Bahrain', 'Bangladesh',
       'Barbados', 'Belarus', 'Belgium', 'Belize', 'Benin', 'Bhutan',
       'Bolivia (Plurinational State of)', 'Bosnia and Herzegovina',
       'Botswana', 'Brazil', 'Brunei Darussalam', 'Bulgaria',
       'Burkina Faso', 'Burundi', "Côte d'Ivoire", 'Cabo Verde',
       'Cambodia', 'Cameroon', 'Canada', 'Central African Republic',
       'Chad', 'Chile', 'China', 'Colombia', 'Comoros', 'Congo',
       'Cook Islands', 'Costa Rica', 'Croatia', 'Cuba', 'Cyprus',
       'Czechia', "Democratic People's Republic of Korea",
       'Democratic Republic of the Congo', 'Denmark', 'Djibouti',
       'Dominica', 'Dominican Republic', 'Ecuador', 'Egypt',
       'El Salvador', 'Equatorial Guinea', 'Eritrea', 'Estonia',
       'Ethiopia', 'Fiji', 'Finland', 'France', 'Gabon', 'Gambia',
       'Georgia', 'Germany', 'Ghana', 'Greece', 'Grenada', 'Guatemala',
       'Guinea', 'Guinea-Bissau', 'Guyana', 'Haiti', 'Honduras',
       'Hungary', 'Iceland', 'India', 'Indonesia',
       'Iran (Islamic Republic of)', 'Iraq', 'Ireland', 'Israel', 'Italy',
       'Jamaica', 'Japan', 'Jordan', 'Kazakhstan', 'Kenya', 'Kiribati',
       'Kuwait', 'Kyrgyzstan', "Lao People's Democratic Republic",
       'Latvia', 'Lebanon', 'Lesotho', 'Liberia', 'Libya', 'Lithuania',
       'Luxembourg', 'Madagascar', 'Malawi', 'Malaysia', 'Maldives',
       'Mali', 'Malta', 'Marshall Islands', 'Mauritania', 'Mauritius',
```

In [25]:
```python
zip_country = zip(countries, [i for i in range(193)])
for i in zip_country:
    print(i[0], ' = ', i[1])
```

```
Afghanistan  =  0
Albania  =  1
Algeria  =  2
Angola  =  3
Antigua and Barbuda  =  4
Argentina  =  5
Armenia  =  6
Australia  =  7
Austria  =  8
Azerbaijan  =  9
Bahamas  =  10
Bahrain  =  11
Bangladesh  =  12
Barbados  =  13
Belarus  =  14
Belgium  =  15
Belize  =  16
Benin  =  17
Bhutan  =  18
Bolivia (Plurinational State of)  =  19
Bosnia and Herzegovina  =  20
Botswana  =  21
Brazil  =  22
Brunei Darussalam  =  23
```

In [33]:
```python
# Examples
Afghanistan_rows = life_expectancy_df.country.value_counts()[0]
print(Afghanistan_rows)

France_rows = life_expectancy_df.country.value_counts()[60]
print(France_rows)
```

```
16
16
```

4.1

Decision trees

```
Best hyper parameters of tree regressor are
max_depth = 6 and min_samples_leaf = 10
```

```
R2 square of train =  0.9674561010438811
R2 square of test =  0.9212856111117357
Some kind of over-fitting between both; min samples = 5 and depth 8
MSE = 7.023853692583779

R2 square of train =  0.9435765567480785
R2 square of test =  0.9111058419427298
Some kind of over-fitting between both but less; min samples = 4 and depth 6
MSE = 7.932216322050023

R2 square of train =  0.9407137927572228
R2 square of test =  0.9112935726105833
Some kind of over-fitting between both but less; min samples = 10 and depth 6
MSE = 7.915464712042782
```

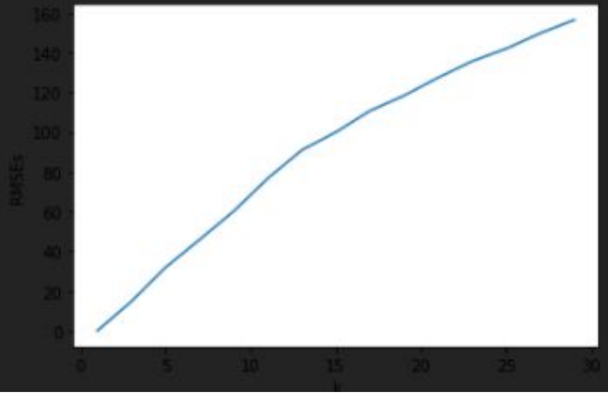
k-NN

```
R2 square of train =  1.0
R2 square of test =  0.884919847873046
Some kind of over-fitting between both; n_ngh = 1 and weights from distance formula
MSE = 10.268848718464996

R2 square of train =  1.0
R2 square of test =  0.9021812959699682
Some kind of over-fitting between both; n_ngh = 5 and weights from distance formula
MSE = 8.728572694382358

R2 square of train =  1.0
R2 square of test =  0.884919847873046
Some kind of over-fitting between both; n_ngh = 1 and weights from uniform formula
MSE = 10.268848718464996

R2 square of train =  0.9462595948551477
R2 square of test =  0.8922809050658804
Some kind of over-fitting between both; n_ngh = 4 and weights from uniform formula
MSE = 9.612005802253034
```

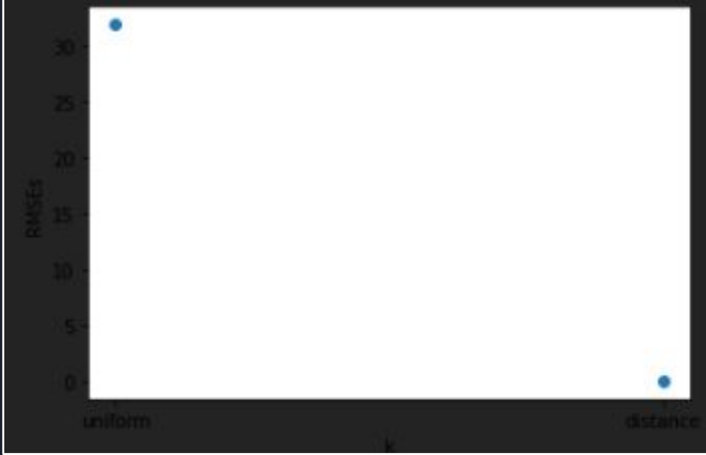
```
Best hyper parameters of kNN regressor are
n_neighbors = 4 and weights = uniform
```

4.1

```
K value =  1 RMSE =  0.0
K value =  3 RMSE =  14.816741954827396
K value =  5 RMSE =  31.893593277774347
K value =  7 RMSE =  45.88777394076914
K value =  9 RMSE =  60.16353338646903
K value =  11 RMSE =  76.7267776899998
K value =  13 RMSE =  90.98818800598046
K value =  15 RMSE =  100.01293674547173
K value =  17 RMSE =  110.75072284928193
K value =  19 RMSE =  118.42414757022938
K value =  21 RMSE =  127.46122999083266
K value =  23 RMSE =  135.74696095319828
K value =  25 RMSE =  142.06802545425452
K value =  27 RMSE =  149.86487360020683
K value =  29 RMSE =  156.56247032863945
```




```
K value =  uniform RMSE =  31.893593277774347
K value =  distance RMSE =  0.0
```



4.2 Best performance between MLP, K-NN and Decision Trees?

```python
mlp = MLPRegressor(max_iter = 1000, random_state=0).fit(X_train_normalized, Y_train)
kNN = KNeighborsRegressor(n_neighbors = 4 , weights = 'uniform').fit(X_train_normalized, Y_train)
tree = DecisionTreeRegressor(random_state = 0, max_depth=6 ,
                             min_samples_leaf = 10).fit(X_train_normalized, Y_train)
```

```python
y_predict_mlp = mlp.predict(X_test_normalized)
y_predict_kNN = kNN.predict(X_test_normalized)
y_predict_tree = tree.predict(X_test_normalized)
```

```python
print('MLP')
print('MSE = ', mean_squared_error(Y_test,  y_predict_mlp) )
print('R2_score = ', r2_score(Y_test , y_predict_mlp) )
print('New metric measure . . .')
print('MAE = ', mean_absolute_error(Y_test, y_predict_mlp) )
print('')
print('Tree')
print('MSE = ', mean_squared_error(Y_test,  y_predict_tree) )
print('R2_score = ', r2_score(Y_test , y_predict_tree) )
print('New metric measure . . .')
print('MAE = ', mean_absolute_error(Y_test, y_predict_tree) )
print('')
print('kNN')
print('MSE = ', mean_squared_error(Y_test,  y_predict_kNN) )
print('R2_score = ', r2_score(Y_test , y_predict_kNN) )
print('New metric measure . . .')
print('MAE = ', mean_absolute_error(Y_test, y_predict_kNN) )
```

```
MLP
MSE =  6.603746768711202
R2_score =  0.9259936334065728
New metric measure . . .
MAE =  1.85425363200399

Tree
MSE =  7.915464712042782
R2_score =  0.9112935726105833
New metric measure . . .
MAE =  1.985046544764065

kNN
MSE =  9.612005802253034
R2_score =  0.8922809050658804
New metric measure . . .
MAE =  2.0533944486264453
```

Performance metrics	MAE	MSE	R^2 Score
MLP Model	1.8542	6.6037	0.9259
Decision Tree Model	1.9850	7.9155	0.9113
K-NN Model	2.0534	9.6120	0.8923

4.3 Comparing performance with Linear Regression Model

```
LR = LinearRegression().fit(X_train_normalized, Y_train)
y_predict_lr = LR.predict(X_test_normalized)

print('LR')
print('MAE = ', mean_absolute_error(Y_test, y_predict_lr) )
print('MSE = ', mean_squared_error(Y_test,  y_predict_lr) )
print('R2_score = ', r2_score(Y_test , y_predict_lr) )
print('New metric measure . . .')
print('Max error =',max_error(Y_test,y_predict_lr))

print('')
print('Max error of Tree =', max_error(Y_test,y_predict_tree) )
print('Max error of kNN =', max_error(Y_test,y_predict_kNN) )
print('Max error of MLP =', max_error(Y_test,y_predict_mlp) )
```

```
LR
MAE =  3.1006779575076706
MSE =  17.05144323514485
R2_score =  0.8089091839520337
New metric measure . . .
Max error = 21.68422095120461

Max error of Tree = 13.994117647058829
Max error of kNN = 21.975
Max error of MLP = 10.99887749677201
```

Performance metrics	MAE	MSE	R^2 Score	Max error
MLP Model	1.8542	6.6037	0.9259	10.9988
Decision Tree Model	1.9850	7.9155	0.9113	13.9941
K-NN Model	2.0534	9.6120	0.8923	21.975
Linear Model	3.1006	17.0514	0.8089	21.6842