LIFE EXPECTANCY PROJECT

- **Main idea:**

In this project, we will estimate the average life expectancy of people on each country in 15 years (from 2000 to 2015) based on a set of features, and this prediction will be automated so that we will train the machine on a set of data to predict the life expectancy with high accuracy.

And since we are dealing with numerical values, we decided to use the (regression) method, and we will use (python) as a programming language, which contains a special library for these operations, we will specifically use (multi linear regression).

- **Recognise the dataset:**

we will use a dataset issued by the United states, containing data for 197 countries from 2000 to 2015 (year), and it also contains 22 feature(attributes/columns) detailed as follows:

- *Country:*
  This column contains a set of 197 countries.
- *Year:*
  This column contains the list of years from 2000 to 2015 and repeats them for every country.
- *Status:*
  this column describes the status of each country so that it categorizes it either into (developing) or (developed).
- *Adult Mortality:*
  This column contains percentage of adult mortality (deaths) on each country on a specific year.

- *infant deaths:*
  This column contains percentage of infant deaths on each country on a specific year.
- *Alcohol:*
  This column contains percentage of alcohol consumption on each country on a specific year.
- *percentage expenditure:*
  This column contains percentage of expenditure (funds spending) on each country on a specific year.
- *Hepatitis B:*
  This column contains numbers describes the number of people infected with hepatitis B on each country on a specific year.
- *Measles:*
  This column contains numbers describes the number of people infected with measles on each country on a specific year.
- *BMI:*
  This column contains numbers describes the average of body mass index (BMI) for people on each country on a specific year.
- *under-five deaths:*
  This column contains numbers describes the number of deaths of people under the age of 5 on each country on a specific year.
- *Polio:*
  This column contains numbers describes the number of people infected with polio on each country on a specific year.
- *Total expenditure:*
  This column contains the total number of expenditure (funds spending) on each country on a specific year.

- *Diphtheria:*
  This column contains numbers describes the number of people infected with diphtheria (an acute, highly contagious bacterial disease) on each country on a specific year.
- *HIV/AIDS:*
  This column contains numbers describes the number of people infected with HIV/AIDS (human immunodeficiency virus) on each country on a specific year.
- *GDP:*
  This column contains numbers describes the number of Gross domestic product (GDP) (the total monetary or market value of all the finished goods and services produced within a country's borders in a specific time period) on each country on a specific year.
- *Population:*
  This column contains numbers describes the number of population on each country on a specific year.
- *thinness 1-19 years:*
  This column contains numbers describes the number of people infected with thinness between the age of 1 to 19 on each country on a specific year.
- *thinness 5-9 years:*
  This column contains numbers describes the number of people infected with thinness between the age of 5 to 9 on each country on a specific year.
- *Income composition of resources:*
  This column contains numbers describes the number of income composition of resources on each country on a specific year.

- *Schooling:*

  This column contains numbers describes the number of schooling (number of students on schools divided by the population of the country) on each country on a specific year.
- *Life expectancy:*

  This column contains numbers that presents the estimated average of life expectancy for people on each country on a specific year.

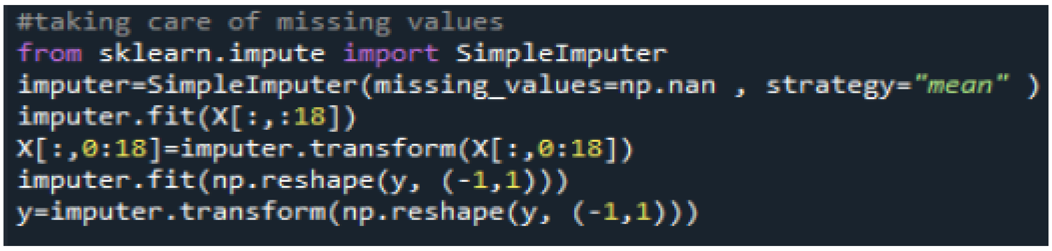
- **Pre-processing steps:**

  After recognising the dataset and defining the attributes on it, we concluded that the dependent value is the (LIFE EXPECTANCY column) and the independent values are the rest of the dataset, and we noticed a number of problems in the dataset, we will review what they are and how we dealt with:

  - Missing values:

  We noticed that 14 attributes (features) of the 22 attributes have missing values and all of them are numerical values, therefore we decided to apply the mean to these values, by taking the mean for each column and substituting the result for the empty values in that column.

  # On the code we used the simple imputer for this step.



```
#taking care of missing values
from sklearn.impute import SimpleImputer
imputer=SimpleImputer(missing_values=np.nan , strategy="mean" )
imputer.fit(X[:,:18])
X[:,0:18]=imputer.transform(X[:,0:18])
imputer.fit(np.reshape(y, (-1,1)))
y=imputer.transform(np.reshape(y, (-1,1)))
```

➢ Categorical data on numerical dataset:

At the dataset there were two categorical columns ("country", "status"), so we apply some python code for them to invert them to numerical, and that's by applying the column transformer using one hot encoder on them, but, unfortunately, another problem appears, and we will discuss it on the next step.

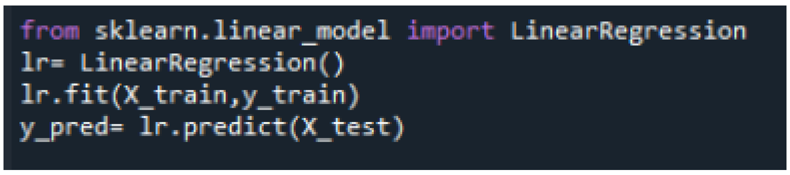
➢ High dimensionality:

After applying the second step on pre-processing stage, we noticed that the dimensionality of the dataset increased, so reconsidering the effect of these two columns and the all dataset, we concluded that the first three columns on the dataset have a weak effect on this project, so we decided to eliminate them for the next steps.

➢ Deferent scale of values on the deferent columns:

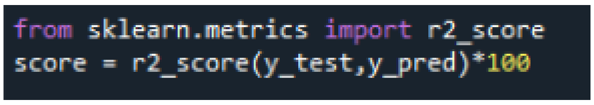
It was obvious from the first glance on the dataset that it contained data from deferent scales, so to solve this problem we need to use (feature scaling) and that's by either use the python function (stander scaler) or let the regression function scale the feature by itself, and we decided to go with the second option, because we noticed that its results are more effective and accurate.

- **<u>Applying regression:</u>**
  Now after dividing the dataset into independent and dependent values, and finished from the pre-processing stage, we are going to divide the dataset into train set and test set then apply the regression method using python, and that's by calling the linear regression function.



```
from sklearn.linear_model import LinearRegression
lr= LinearRegression()
lr.fit(X_train,y_train)
y_pred= lr.predict(X_test)
```

- **Evaluation:**
  To ensure that the operation is successful and satisfactory, we must evaluate it, by applying a set of methods, the first here was: the (r2_score function) which calculate the accuracy of the y_pred compared to y_test (the original output).



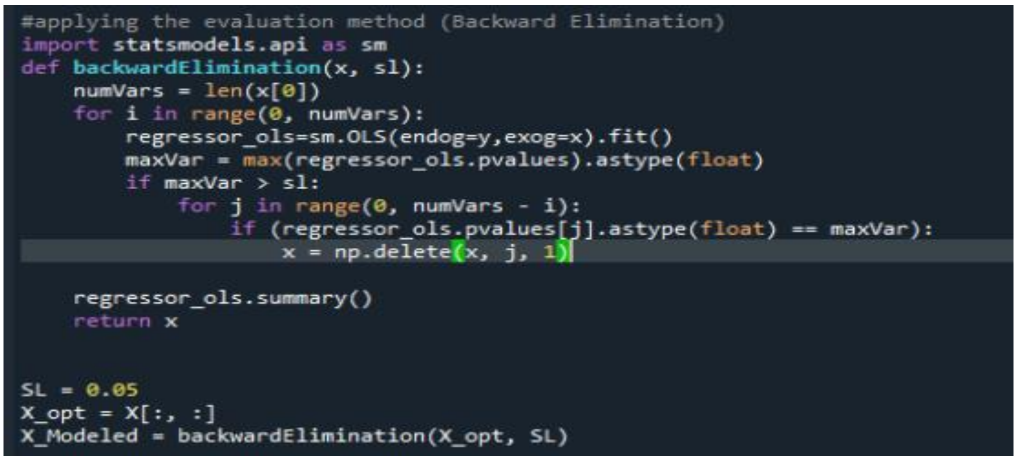
```
from sklearn.metrics import r2_score
score = r2_score(y_test,y_pred)*100
```

  And for the next step we will use the (backward elimination), We need to add a constant column to the data set as the first step then applying these steps as shown:

- Step1: Select a significance level to stay in the model (e.g. SL=0.05).
- Step2: Fit the full model with all possible predictors.
- Step3: Consider the predictor with the highest P-Value. If P>SL, go to step 4, otherwise go to FIN.
- Step4: Remove the predictor.
- Step5: Fit model without this variable.

But because of this large number of columns, it will be difficult to apply it manually, so I searched for how to apply these steps on a loop to facilitate the matter and save time, and I found the solution in the link below:

"https://stackoverflow.com/questions/59704085/backward-elimination-in-python"

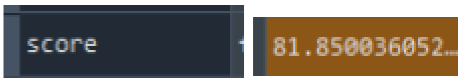


```
#applying the evaluation method (Backward Elimination)
import statsmodels.api as sm
def backwardElimination(x, sl):
    numVars = len(x[0])
    for i in range(0, numVars):
        regressor_ols=sm.OLS(endog=y,exog=x).fit()
        maxVar = max(regressor_ols.pvalues).astype(float)
        if maxVar > sl:
            for j in range(0, numVars - i):
                if (regressor_ols.pvalues[j].astype(float) == maxVar):
                    x = np.delete(x, j, 1)

    regressor_ols.summary()
    return x


SL = 0.05
X_opt = X[:, :]
X_Modeled = backwardElimination(X_opt, SL)
```

And the result of this function was to eliminate 3 more columns from the dataset for more accurate outputs, and these columns were:

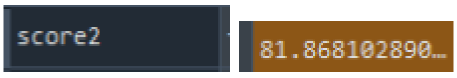
[“percentage expenditure”, “population”, “thinness 5-9 years”]

Then reapplying the regression and the r2_score we can have noticed that this elimination improved the output accuracy.



```
score    81.850036052…
```

vs



```
score2    81.868102890…
```

As a result, we can see here that “score” which represents the accuracy of the regression before the backward elimination is less than “score2” which represent the accuracy of the regression after.

Done by:

Hashem shehadeh

supervised by:

Ruba al khusheiny