Data processing (Life expectancy example)

Table of Contents

[List of Figures 3](#_Toc119728015)

[List of Tables 4](#_Toc119728016)

[Introduction 5](#_Toc119728017)

[1.1 Background 5](#_Toc119728018)

[1.2 Aim and objectives 5](#_Toc119728019)

[Methodology 6](#_Toc119728020)

[2.1 Data understanding 6](#_Toc119728021)

[2.2 Data pre-processing 8](#_Toc119728022)

[2.2.1 Data importing 8](#_Toc119728023)

[2.2.2 Check the problems in the data 10](#_Toc119728024)

[2.2.3 Deal with data problem 14](#_Toc119728025)

[Data processing 27](#_Toc119728026)

[References 32](#_Toc119728027)

[Appendix 34](#_Toc119728028)

# List of Figures

[Figure 1: The life expectancy globally in 2019 (Roser, et al., 2013) 5](#_Toc119727944)

[Figure 2: The steps of data analysis that were followed in the project 6](#_Toc119727945)

[Figure 3: Data importing steps-2 9](#_Toc119727946)

[Figure 4: Data importing steps-3 9](#_Toc119727947)

[Figure 5: Data importing steps-4 10](#_Toc119727948)

[Figure 6: Data importing steps-5 10](#_Toc119727949)

[Figure 7: The results of using info command 11](#_Toc119727950)

[Figure 8: The results of checking the data type 12](#_Toc119727951)

[Figure 9: Using duplicated function to check duplication 12](#_Toc119727952)

[Figure 10: The exact number of missing value in each column 13](#_Toc119727953)

[Figure 11: The differences between mean values in each single feature 14](#_Toc119727954)

[Figure 12: Simple imputer with the missing data problem 14](#_Toc119727955)

[Figure 13: Using the groupby command to check the status and the results of it 15](#_Toc119727956)

[Figure 14: The flow chart of problem solving 15](#_Toc119727957)

[Figure 15: The strategy of solving missing data problem 16](#_Toc119727958)

[Figure 16: The method of solving the initial spaces in the dataframe 16](#_Toc119727959)

[Figure 17: The results of checking the trailing spaces 17](#_Toc119727960)

[Figure 18: The results of strip function 17](#_Toc119727961)

[Figure 19: First method of splitting the data 18](#_Toc119727962)

[Figure 20: Second method of splitting the data 18](#_Toc119727963)

[Figure 21: The required data type to be converted 19](#_Toc119727964)

[Figure 22: Categorical encoding method cheat sheet 23](#_Toc119727965)

[Figure 23: The methodology and the result of extracting the unique categorical data 24](#_Toc119727966)

[Figure 24: The methodology of dealing with status column using label encoding 24](#_Toc119727967)

[Figure 25: The result of label encoding 24](#_Toc119727968)

[Figure 26: Target encoding of the country column 25](#_Toc119727969)

[Figure 27: A comparison between both methods 26](#_Toc119727970)

[Figure 28: Understand the correlation (robinhood, 2022) 29](#_Toc119727971)

# List of Tables

[Table 1: Features and their impact on life expectancy 6](#_Toc119727999)

[Table 2: The differences between both types data problem 10](#_Toc119728000)

[Table 3: The differences between the dependent and independent variables 18](#_Toc119728001)

[Table 4: The popular methods of dealing with the categorical data in python 19](#_Toc119728002)

# Introduction

## 1.1 Background

As mentioned by (Roser, et al., 2013), life expectancy is one of the most important tools that are is used to test the population health, it gives indication regarding the expected average age of the death in population. When comparing the life expectancy globally, although it has developed widely, it can be noticed that it is differ between countries, because many countries are suffering from issues (war, lake of resources, etc.) that led to bad health situation. In addition to this, the average life expectancy is affected by number of factors such as disease, the location of that country, alcohol, etc. Figure 1 shows the life expectancy globally in 2019.

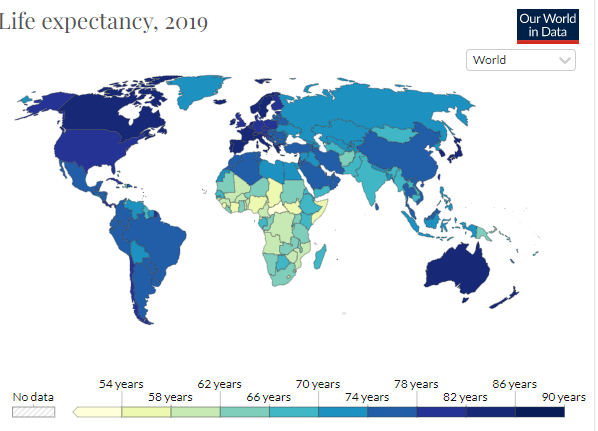


Figure 1: The life expectancy globally in 2019 (Roser, et al., 2013)

## 1.2 Aim and objectives

This work aim to study the feasibility of predicting the factors that play major role in life expectancy globally based backward elimination algorithm. This aim will be achieved by the below objectives:

1. Understand the factors that might lead to increase or decrease life expectancy.
2. Use algorithm of backward elimination to build the prediction model.
3. Provide recommendations for further improvements.

# Methodology

In order to achieve the aim of this project, this section provide summary about the flow of the steps that were followed starting from understanding the data to extracting the results. Figure 2 illustrates the steps of data analysis that were followed in the project.

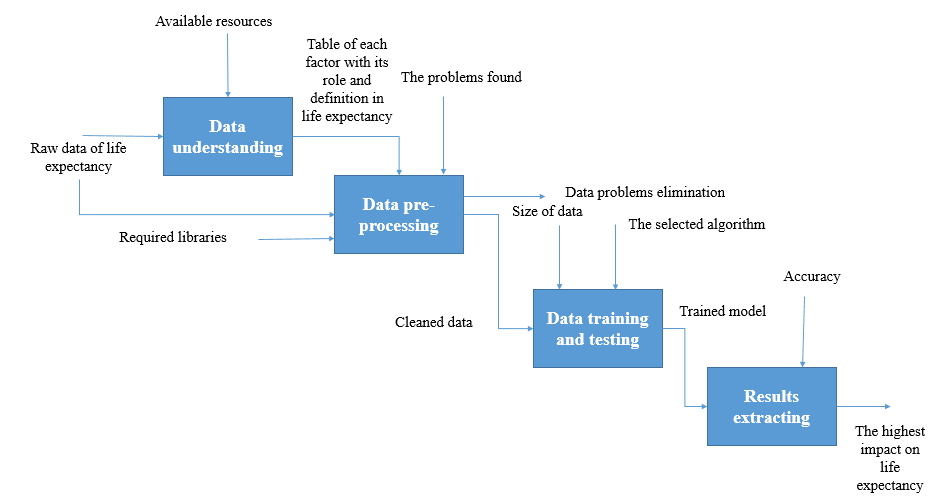


Figure 2: The steps of data analysis that were followed in the project

## 2.1 Data understanding

The step of data understanding is collected data and giving information of each single feature in the raw data. This can be done based on previous literatures review, the data was collected from 193 different countries, and there were 21 factors that might affect the life expectancy, Table 1 presents each feature and its impact on life expectancy.

Table 1: Features and their impact on life expectancy

|  |  |  |
| --- | --- | --- |
| **Feature** | **Definition** | **References** |
| Country | Wealthier countries has higher average life expectancy than poorer country due to high living standards, their access to resources, healthier system. Furthermore, the political and commercial status play major roles in controlling the life expectancy. | (Freeman, et al., 2020) |
| Status | The socioeconomic factor is crucial in controlling life expectancy, this because the increasing in economic and living standard led to elevate levels of health sector and life expectancy as a result. | (Miladinov, 2020) |
| Year | Life expectancy increased over years as a result of developing in health sector. | (Roser, et al., 2013) |
| Adult mortality | It gives an indication of the health and life style of the society regardless the expenditure on health sector by the government- Albanian.  The adult mortality means that the probability that those how reached age 15 will die before age 60 (shown per 1000 person) | (Sanja Musić Milanović, et al., 2006) |
| Infant deaths | The lower infant deaths the higher life expectancy. | (OECD, 2017) |
| Healthcare expenditure | The health spending and income have typically had a stronger impact on reducing  avoidable mortality or infant mortality than on increasing life expectancy. 1% of increasing in healthcare expenditure will increase the life expectancy by 2.625. | (OECD, 2017) (Thomas, 2020) |
| Alcohol | The consumption of different types of alcoholic beverages leads to different health risks (liver disease ; higher risk of high blood pressure, heart failure, and dementia). | (Gavurová, et al., 2020) |
| Disease (Measles; Hepatitis B; Polio; Diphtheria; HIV/AIDS) death cases | The more dissease the less life expectancy | (Miladinov, 2020) |
| Income composition of resources | ICOR is the measure of how good a country is at utilizing its resources . ICOR is graded between 0 to 1 , and higher ICOR indicates optimal utilization of available resources . ICOR has a considerably high correlation with Life expectancy . Let's visualize the impact of ICOR on Life expectancy continent - wise . |  |
| BMI | longer life expectancy is accompanied by good health among ageing populations has important implications for health and long-term care systems.  It was found that an individual's risk of death continued to increase at the high end of that BMI range. Specifically, we found that BMIs from 40 to 44 were associated with 6.5 years of life lost | (Jia & Lubetkin, 2022) |
| GPD | Gross domestic product (GDP) is the standard measure of the value added created through the production of goods and services in a country during a certain period. As such, it also measures the income earned from that production, or the total amount spent on final goods and services. | (oecd, 2022) |

## 2.2 Data pre-processing

The step of data pre-processing is important in order to prepare the data for analysis, the below steps were followed in order to clean data:

### 2.2.1 Data importing

The first step is import the raw data from excel to python, and import the required modules. It should be noted that the python and raw data should be at the same path, the below steps were followed when opening the csv file:

1. Create new folder at copy the required file that contains the raw data to this folder.

2. Copy the path of this folder to python.

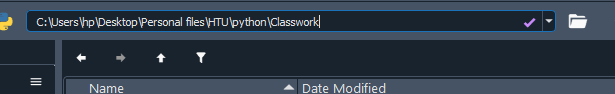


Figure 3: Data importing steps-2

3. Set the console directory to the current directory.

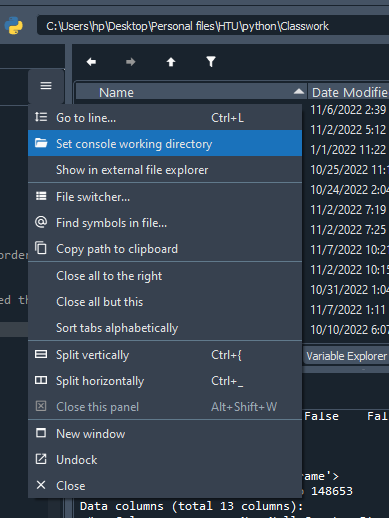


Figure 4: Data importing steps-3

4. Create new python file and ensure to save it at the same path of excel path, to ensure that open it from the console file.

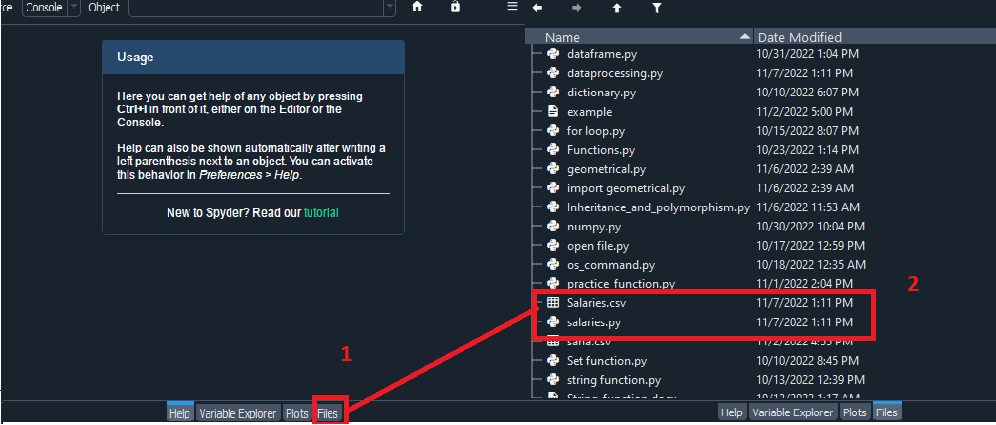


Figure 5: Data importing steps-4

5. Now in the directory you can open the file by importing the csv file by using read\_csv (‘File Name’).

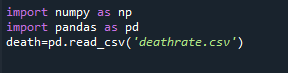


Figure 6: Data importing steps-5

### 2.2.2 Check the problems in the data

After importing the data, the following step is to clean the data from any unrequired data, this include missing data, duplication, cartographical data, unscaled numeric data, zero impact row and column data, etc. To clean the data, data problems should be recognized first, data problem can be noted using two methods, these are by necked eye and by using python code. Table 1 illustrates the differences between both types data problem.

Table 2: The differences between both types data problem

|  |  |  |
| --- | --- | --- |
|  | **By necked eye** | **By python code** |
| **Degree of assurance** | High: in this case, the data problem is clear and easy to be recognized from excel sheet. | Low: In this case and due to the amount of data, it is clear if the common problems is existed in this data set or not, accordingly it was checked using python commands. |
| **Example** | Null, zero impact data (not provided string), zero impact features (have zero impact on the output such as name and ID). | Duplicated data (Duplicated rows). |

**1. Use info function to provide general information**

By using info (info=death.info()) you can get general information regarding the type of the data, the and missing values. Figure 7 shows the results of using info command.

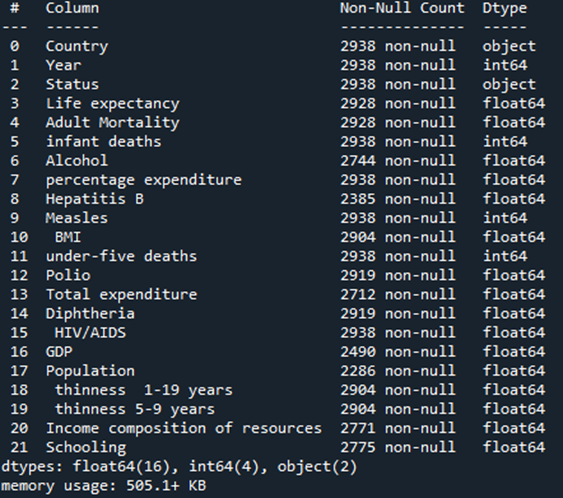
****

Figure 7: The results of using info command

The results shows that are 2 data object type (string) which means that there is no mixed data type.

**2. Use types.infer\_dtype for check the data type**

You can double check your data type by using pandas.api.types.infer\_dtype (), it will return exactly if the object is string or mixed data type.

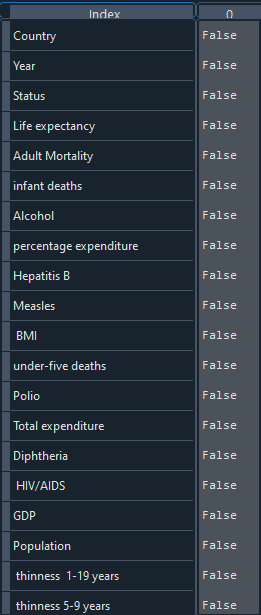


Figure 8: The results of checking the data type

It is clear from Figure 8 that there is no mixed data type, although this, it is still important to deal with categorical data string type.

**3. Check for duplicated values**

It is important to check the duplicated values between rows, in order to avoid any inaccurate results, this can be done by using the duplicated function.



Figure 9: Using duplicated function to check duplication

The results show that there is zero duplication between rows.

**4. Check for null values**

In order to count the null values the death.isnull().sum() can be used to return the exact number of missing values in ech column.

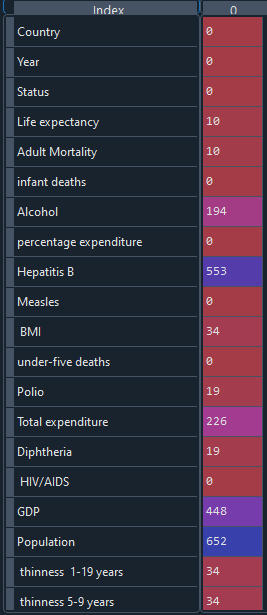


Figure 10: The exact number of missing value in each column

After identifying the problems now you need to solve and clean the data from these problems.

### 2.2.3 Deal with data problem

Based on the previous section, there are two main problems, these are the missing data problem and the categorical data.

**1. Deal with missing data:**

When check the data you can notice that there are 193 different countries, but the main problem is each country has mean value of each feature. Figure 11 shows the differences between mean values in each single feature.

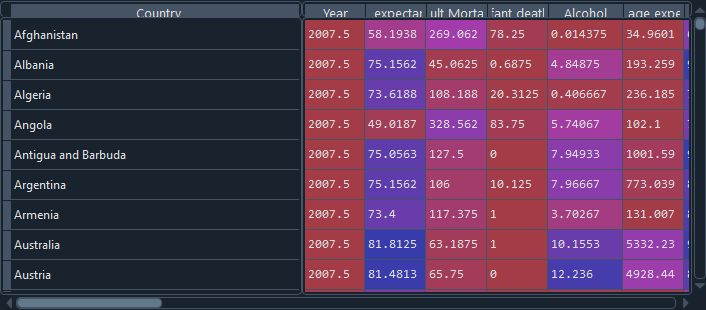


Figure 11: The differences between mean values in each single feature

For example, in alcohol feature, Afghanistan mean value is 0.014 but in Albania 4.848 which means there is evidence support the statement that said both the mean value can be the mean of all countries in the data set. Furthermore, countries like Dominica, was mentioned once, which means that the average can’t be taken when missing data exist.

Missing data problem usually solved by using imputer module from sklearn, but this method will not work with groups.

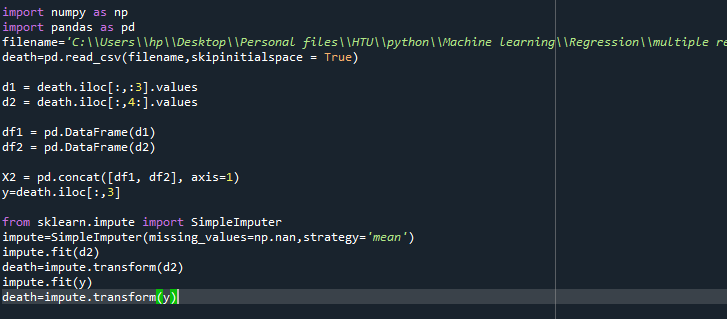


Figure 12: Simple imputer with the missing data problem

As mentioned previously, the countries should be grouped in order to deal with missing data in each country separately, then to deal with the missing data in the countries that were mentioned only once, the data should be grouped based on their status. To check if there is a difference between the developing and developed countries, groupby status command was used.

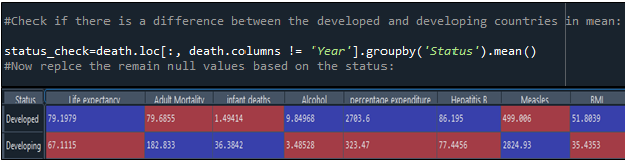


Figure 13: Using the groupby command to check the status and the results of it

The results show dramatic difference between the developed and developing countries. Figure 14 shows the flow chart of problem solving.

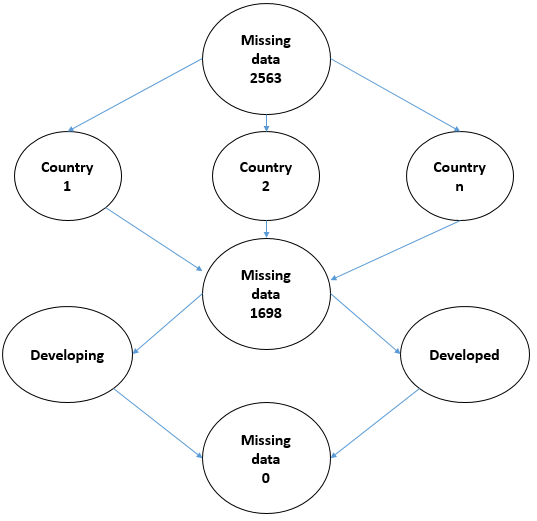


Figure 14: The flow chart of problem solving

To reflect this flow chart on the code, Figure 15 present the strategy of solving missing data problem.

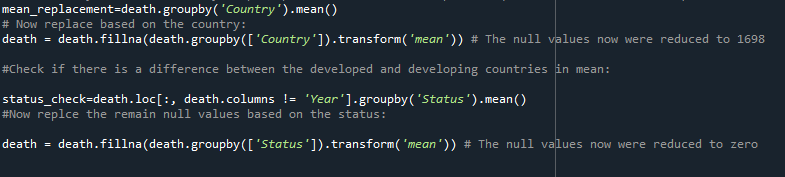


Figure 15: The strategy of solving missing data problem

**2. Deal with white spaces**

One important problem that appear when dealing with the categorical data is the white space problem. Unless the white space problem solved, you won’t be able to encode the categorical data, because you will call the columns name incorrectly. However, the problem initially solved by adding the skipinitialspace () to the read\_csv file. Figure 16 shows the method of solving the initial spaces in the dataframe.

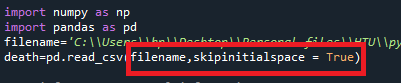


Figure 16: The method of solving the initial spaces in the dataframe

It should be noted that the skipinitialspace - It controls how the space following the delimiter will be interpreted, which means that it will remove the leading spaces only, while the trailing spaces problem were not solved. To double check this problem, the .tolist() function was used. Figure 17 presents the results of checking the trailing spaces. As it clear in Figure 17, the problem of trailing white spaces should be solved.

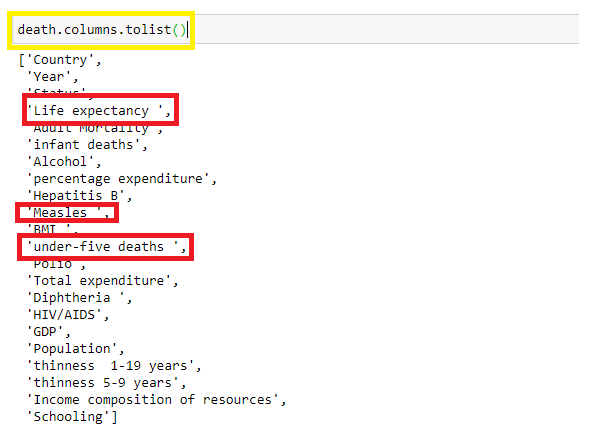


Figure 17: The results of checking the trailing spaces

To solve this problem was solved by using the strip function and apply it on the dataframe, when do so, all the problem regarding the white spaces were solved. Figure 18 presents the results of strip function.

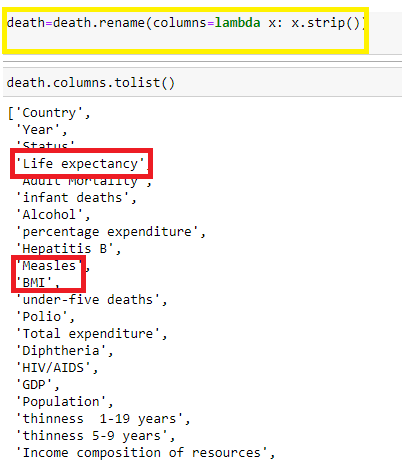


Figure 18: The results of strip function

**3. Split the data**

Now to be able to deal with the data, the data should be split into dependent and independent variables. The differences between the dependent and independent variables are provided in Table 3.

Table 3: The differences between the dependent and independent variables

|  |  |  |
| --- | --- | --- |
|  | **Dependent data** | **Independent data** |
| Definition | The data that is mainly affected by other data, or the probability of increasing or decreasing affected by the other data. | The data that is not affected by other data, or its probability of increasing or decreasing is not affected by the other data. |
| Example from the data set | life expectancy | All the features except life expectancy |

In the given data all features are independent except the life expectancy, two methods can be used, and the methods differ from each other with the execution time.

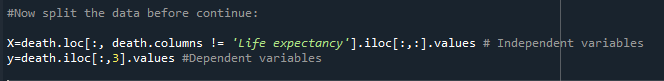


Figure 19: First method of splitting the data

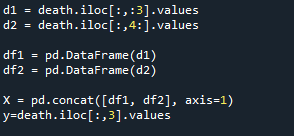


Figure 20: Second method of splitting the data

**4. Deal with the categorical and unique data**

The categorical data means the data types that can stored in groups, sex, age education level, countries, and cities are an example of categorical data types. High cardinality (unique values) data also considered as categorical data. In the given dataset, the year, country, and status are an example of categorical data type (formpl, 2022). Categorical data can take on numerical values like the year in the data set. The categorical data actually lead to two main problems in the machine learning model, these are: most of the machine learning model don’t accept those data type,

memory consuming and dimensionality (towardsdatascience, 2022). Figure 21 shows the required data type to be converted.

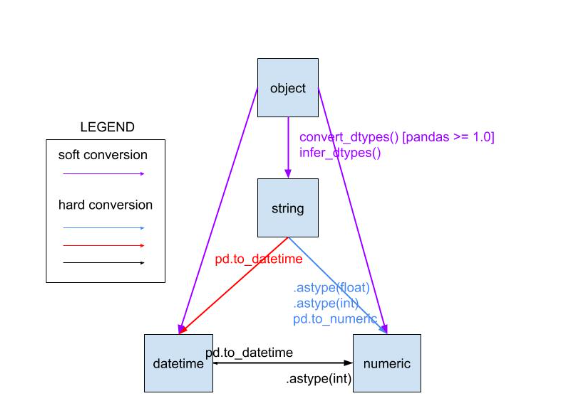


Figure 21: The required data type to be converted

There are wide range of techniques that are used to handle the categorical data, but each method has its limitations and case of use, furthermore, there is no optimal solution it is all related to the dataset itself. Table exhibits the popular methods of dealing with the categorical data in python.

Table 4: The popular methods of dealing with the categorical data in python

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **The technique** | **The definition** | **The limitations** | **Required libraries\tools** | **Use case** | **Reference** |
| Label encoding | In label encoding, each category is assigned a value from 1 through N where N is the number of categories for the feature. There is no relation or order between these assignments. | In the above scenario, the Country names do not have an order or rank. But, when label encoding is performed, the country names are ranked based on the alphabets. Due to this, there is a very high probability that the model captures the relationship between countries such as India < Japan < the US. | **from** sklearn.preprocessing **import** LabelEncoder | 1. The categorical feature is ordinal (like Jr. kg, Sr. kg, Primary school, high school) 2. The number of categories is quite large as one-hot encoding can lead to high memory consumption | (analyticsvidhya, 2022) (maxhalford, 2022) |
| One Hot Encoding | In one hot encoding, for each level of a categorical feature, we create a new variable. Each category is mapped with a binary variable containing either 0 or 1. Here, 0 represents the absence, and 1 represents the presence of that category. These newly created binary features are known as**Dummy variables.** The number of dummy variables depends on the levels present in the categorical variable. This might sound complicated. | One-Hot Encoding results in a Dummy Variable Trap as the outcome of one variable can easily be predicted with the help of the remaining variables. | from sklearn from sklearn.preprocessing import OneHotEncoder  onehotencoder = OneHotEncoder() | 1. The categorical feature is**not ordinal** (like the countries above) 2. The number of categorical features is less so one-hot encoding can be effectively applied | (analyticsvidhya, 2022) (maxhalford, 2022) |
| Dummy encoding | Dummy coding scheme is similar to one-hot encoding. This categorical data encoding method transforms the categorical variable into a set of binary variables (also known as dummy variables, Dummy encoding uses N-1 features to represent N labels/categories. | 1. A large number of levels are present in data. If there are multiple categories in a feature variable in such a case we need a similar number of dummy variables to encode the data. For example, a column with 30 different values will require 30 new variables for coding. 2. If we have multiple categorical features in the dataset similar situation will occur and again we will end to have several binary features each representing the categorical feature and their multiple categories e.g a dataset having 10 or more categorical columns. | data\_encoded=  pd.get\_dummies  (data=data,drop\_first=True) | 1. The ***get\_dummies*** can’t handle the unknown category during the transformation natively. You have to apply some techniques to handle it. But it is not efficient. On the other hand, ***OneHotEncoder***will natively handle unknown categories. All you need to do is set the parameter handle\_unknown='ignore' to OneHotEncoder.  2. For example, in the tips dataset you have seen above, the day column contains four unique values — Thur, Fri, Sat, and Sun. If the test dataset contains a new category, say Mon or Tue, then the get dummies will create a new column day\_Mon or day\_Tue which will be inconsistent with train data and will eventually fail during the model building process.  3. Though **get\_dummies** can’t handle unknown categories natively, you could get around this inconsistency by applying the below technique. You will have to save the columns of the train set and load it during prediction on test set. Then you need to apply **reindex**and after filling the missing values will get you the same features as the train set. Refer below.  **If you are building the machine learning models then go for OneHotEncoder and for data analysis tasks you can consider either OneHotEncoder or get\_dummies**. | (pythonsimplified, 2022) |
| Target encoding | Target-based encoding is numerization of categorical variables via target. In this method, we replace the categorical variable with just one new numerical variable and replace each category of the categorical variable with its corresponding probability of the target (if categorical) or average of the target (if numerical). The main drawbacks of this method are its dependency to the distribution of the target, and its lower predictability power compare to the binary encoding method. | It is very easy to over fit with target encoding.  The problem of target encoding has a name: over-fitting. Indeed, relying on an average value isn’t always a good idea when the number of values used in the average is low. You’ve got to keep in mind that the dataset you’re training on is a sample of a larger set. This means that whatever artifacts you may find in the training set might not hold true when applied to another dataset (i.e. the test set). (if 0.5 0.5 )  Also data leakage. | from category\_encoders import TargetEncoder  targetencoder = TargetEncoder()  X[:,0] = encoder.fit\_transform(X[:,0], y) #You must pip install  #category\_encoders library  OR  death.groupby('Country')['Life expectancy'].mean()  X[:,0] = death['Country'].map(mean) | **Use Cases for Target Encoding** Target encoding is great for:   * **High-cardinality features**: A feature with a large number of categories can be troublesome to encode: a one-hot encoding would generate too many features and alternatives, like a label encoding, might not be appropriate for that feature. A target encoding derives numbers for the categories using the feature's most important property: its relationship with the target. * Domain-motivated features: From prior experience, you might suspect that a categorical feature should be important even if it scored poorly with a feature metric. A target encoding can help reveal a feature's true in formativeness. | (saedsayad, 2022)  (kaggle, 2022) |

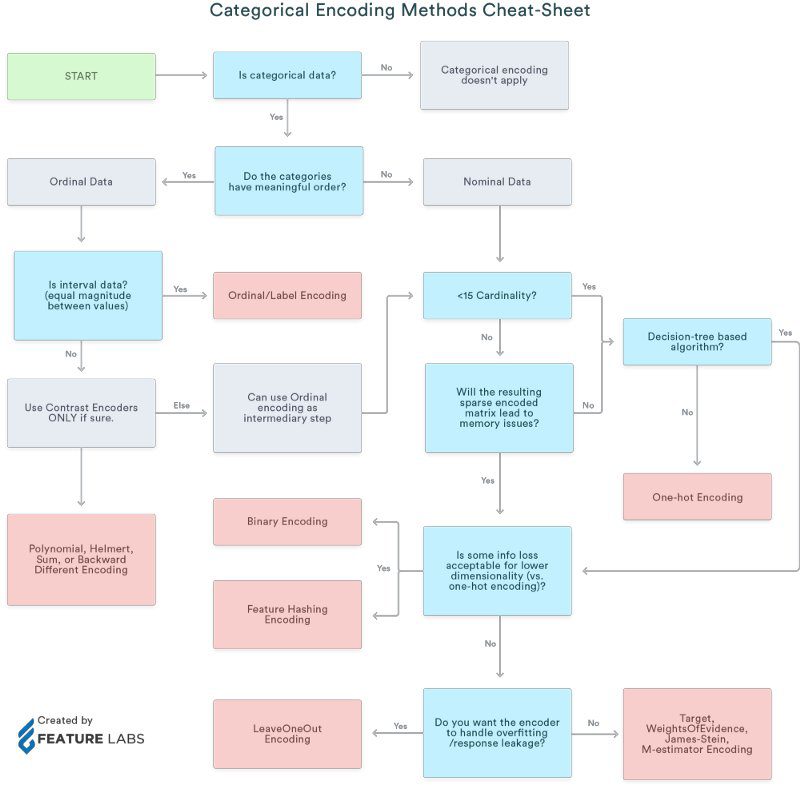


Figure 22: Categorical encoding method cheat sheet

Now, after read about the each case, now it’s time to select the encoding of each categorical data type. Before that, it is important to check the unique categorical data in each column, Figure 23 illustrate the methodology and the result of extracting the unique categorical data.

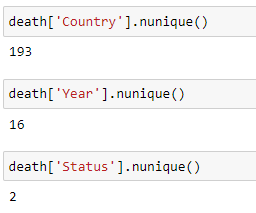


Figure 23: The methodology and the result of extracting the unique categorical data

**1. Status column:**

The status column contain only two category, these are developed and developing, to handle these two data, the label encoding was used. Figure 23 present the methodology of dealing with status column using label encoding.

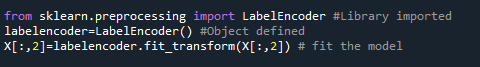


Figure 24: The methodology of dealing with status column using label encoding

Each cell will be replaced with either zero or one value, Figure 24 shows the result of label encoding.

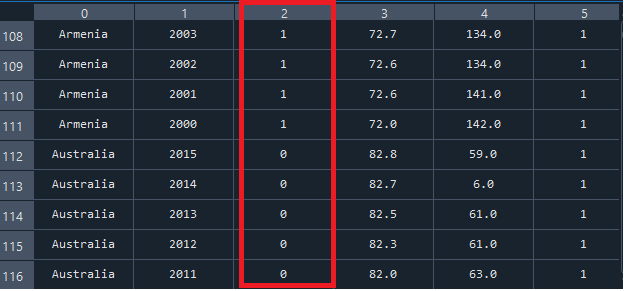


Figure 25: The result of label encoding

**2. The country column:**

Hence there are around 193 unique country the optimal solution here is to use the target encoder method. The mathematical equation is represented below:

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

In which:

* is the mean we’re trying to compute (the one that’s going to replace our categorical values)
* n is the number of values you have
* is your estimated mean
* m is the “weight” you want to assign to the overall mean
* w is the overall mean

This can be done by either use the category\_encoders library or by using the mean directly, but the library of category encoder need to be downloaded, the second method was selected.

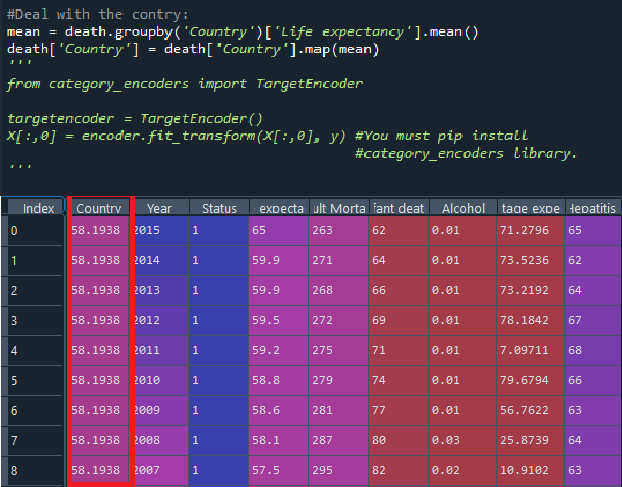


Figure 26: Target encoding of the country column

**3. The year column:**

The year column has 16 year that need to be encoded, to do so, one hot encoding (ordinal data) and label encoding both were used to test the optimal encoding method. Then the accuracy was tested, the label encoding shows a little bit better performance when it compared to the one hot encoder. Figure 27 present a comparison between both methods.

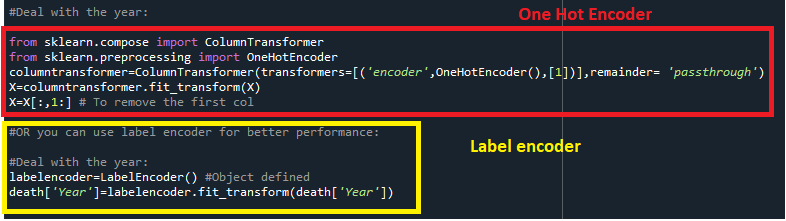


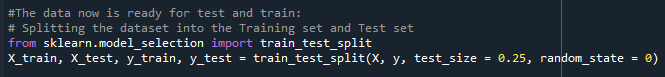
Figure 27: A comparison between both methods

Now the data is ready for test and train.

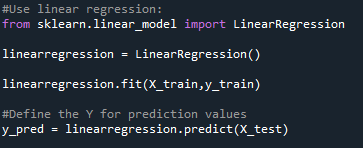
# Data processing

After cleaning the data, the following step is to train and test the data, this can be done in 4 stages, the below sections provide further explanation.

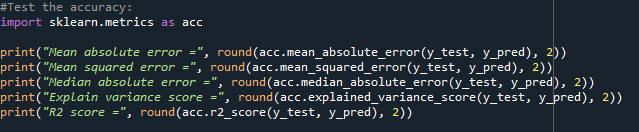
1. Split the data for test and train:



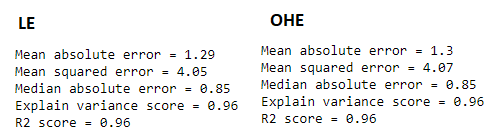
2. Apply linear regression model:



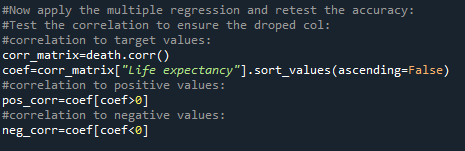
3. Test the accuracy:



* The results of the OHE and LE:

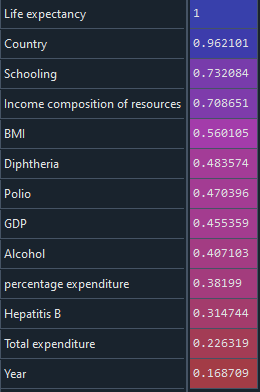


4. Test the correlation before apply MLR to be aware which col will be dropped:

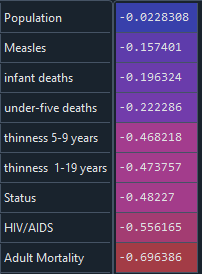


**The correlation results:**

**1. Positive correlation:**



* **Negative correlation:**



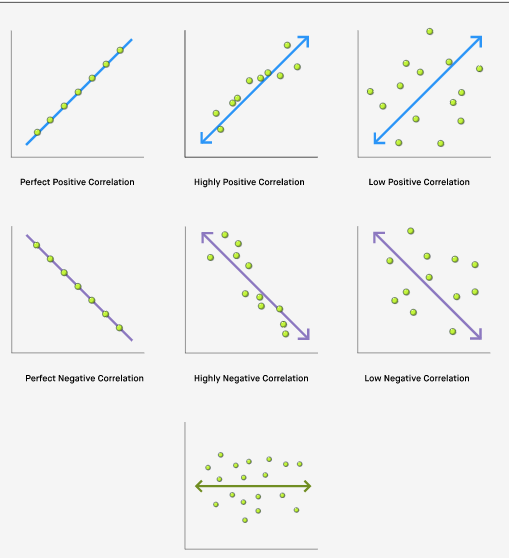
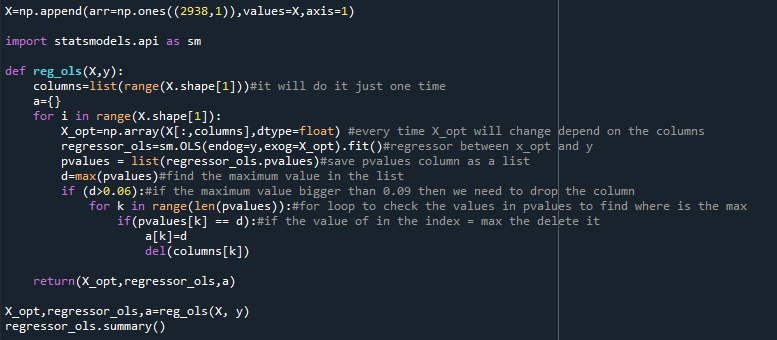
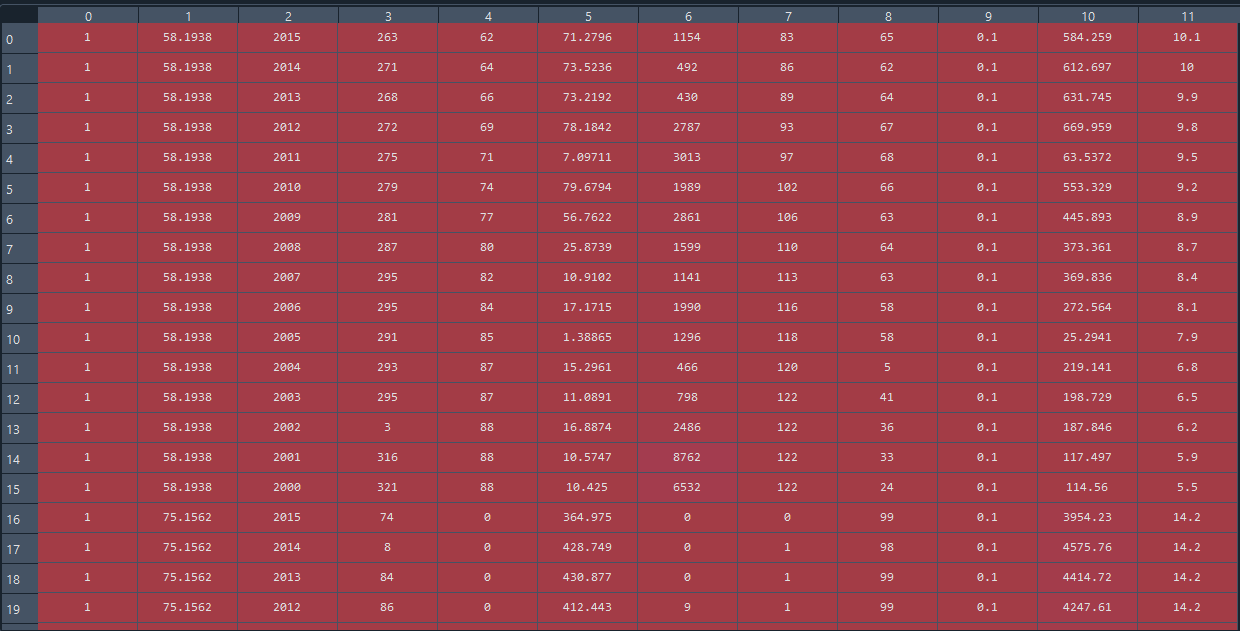


Figure 28: Understand the correlation (robinhood, 2022)

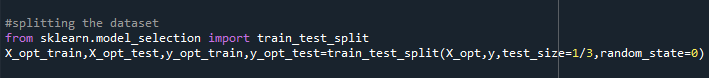
5. Apply the backward elimination algorithm in MLR:



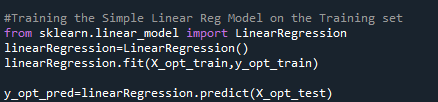
The X\_opt value:



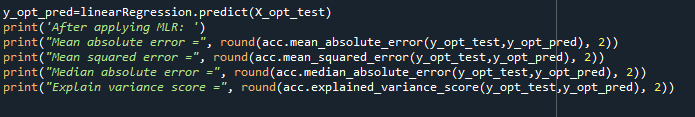
6. Split the train and test again:



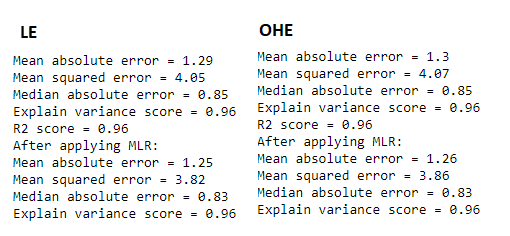
7. Apply the linear regression again:



8. Test the accuracy:



* **The results of accuracy:**



# References

analyticsvidhya, 2022. *analyticsvidhya.* [Online]   
Available at: https://www.analyticsvidhya.com/blog/2020/03/one-hot-encoding-vs-label-encoding-using-scikit-learn/  
[Accessed 18 11 2022].

formpl, 2022. *formpl.* [Online]   
Available at: https://www.formpl.us/blog/categorical-data  
[Accessed 18 11 2022].

Freeman, T. et al., 2020. Why do some countries do better or worse in life expectancy relative to income? An analysis of Brazil, Ethiopia, and the United States of America. *International Journal for Equity in Health volume ,* 19(202).

Gavurová, B., Ivankova, V. & Rigelský, M., 2020. Does Alcohol Consumption Afect Life Expectancy in OECD Countries?. *Adiktologie,* 20(2), pp. 115-123.

Jia, H. & Lubetkin, E. I., 2022. Association between self-reported body mass index and active life expectancy in a large community-dwelling sample of older U.S. adults. *BMC Geriatrics.*

kaggle, 2022. *kaggle.* [Online]   
Available at: https://www.kaggle.com/code/ryanholbrook/target-encoding  
[Accessed 18 11 2022].

maxhalford, 2022. *maxhalford.* [Online]   
Available at: https://maxhalford.github.io/blog/target-encoding/  
[Accessed 18 11 2022].

Miladinov, G., 2020. Socioeconomic development and life expectancy relationship: evidence from the EU accession candidate countries. *Genus volume ,* 72(2).

OECD, 2017. Health at a Glance . In: *What has driven life expectancy gains in recent decades? A cross-country analysis of OECD member states.* Paris: OECD Publishing, Paris, pp. 31-46.

oecd, 2022. *oecd.* [Online]   
Available at: https://data.oecd.org/gdp/gross-domestic-product-gdp.htm  
[Accessed 16 11 2022].

pythonsimplified, 2022. *pythonsimplified.* [Online]   
Available at: https://pythonsimplified.com/difference-between-onehotencoder-and-get\_dummies/  
[Accessed 18 11 2022].

robinhood, 2022. *robinhood.* [Online]   
Available at: https://learn.robinhood.com/articles/dhlLGCRuKy6YZY6k2bLPM/what-is-negative-correlation/  
[Accessed 18 11 2022].

Roser, M., Ortiz-Ospina, E. & Ritchie, H., 2013. *Life Expectancy.* [Online]   
Available at: https://ourworldindata.org/life-expectancy  
[Accessed 16 11 2022].

saedsayad, 2022. *saedsayad.* [Online]   
Available at: http://www.saedsayad.com/encoding.htm  
[Accessed 18 11 2022].

Sanja Musić Milanović, 1. A. I. U. et al., 2006. Life Expectancy and Mortality Differences between Populations on Croatian Islands and the Mainland. *Croat Med J,* 47(4), p. 611–618.

Thomas, A., 2020. *The Impact of Health Expenditure on Life Expectancy and Mortality Rates,* s.l.: The University of Akron-Senior Project.

towardsdatascience, 2022. *towardsdatascience.* [Online]   
Available at: https://towardsdatascience.com/dealing-with-features-that-have-high-cardinality-1c9212d7ff1b  
[Accessed 18 11 2022].

# Appendix

# -\*- coding: utf-8 -\*-

"""

Created on Wed Nov 16 12:21:32 2022

Leen Alnajjar

Life expectancy

"""

import numpy as np

import pandas as pd

filename='C:\\Users\\hp\\Desktop\\Personal files\\HTU\\python\\Machine learning\\Regression\\multiple reg\\life expectancy\\deathrate.csv'

death=pd.read\_csv(filename,skipinitialspace = True)

#use info to get general information:

info=death.info()

#check mixed data type:`

from pandas.api.types import infer\_dtype

mixed=death.apply(lambda x: 'mixed' in infer\_dtype(x))

#check for duplicted values:

duplicated=death.duplicated().sum() #no duplicated rows

#Check for null values:

null=death.isnull().sum() # The total number of null values is 2563

#Deal with data problems:

#Group by year:

# applying groupby() function to

# group the data on team value in order to replace the missing value:

#for missing values:

#ensure that each country has its own mean:

#I'm afraid you cannot use only SimpleImputer for this kind of problem (at least as far as I know).

mean\_replacement=death.groupby('Country').mean()

# Now replace based on the country:

death = death.fillna(death.groupby(['Country']).transform('mean')) # The null values now were reduced to 1698

#Check if there is a difference between the developed and developing countries in mean:

status\_check=death.loc[:, death.columns != 'Year'].groupby('Status').mean()

#Now replce the remain null values based on the status:

death = death.fillna(death.groupby(['Status']).transform('mean')) # The null values now were reduced to zero

#The following step is to deal with the caterogical daata:

#The selected method to dael with the catero

#When trying to use the labeling method an error appear regarding the header name, to check that use tolist col:

whitespaces=death.columns.tolist() #Check from jupyter it will be clearer

#Now remove all white spaces:

death=death.rename(columns=lambda x: x.strip())

#You can now deal with the categorical data:

#Deal with status:

Unique=death.iloc[:,:].nunique() #this important for label encoder

from sklearn.preprocessing import LabelEncoder #Library imported

labelencoder=LabelEncoder() #Object defined

death['Status']=labelencoder.fit\_transform(death['Status'])

#Deal with the contry:

mean = death.groupby('Country')['Life expectancy'].mean()

death['Country'] = death['Country'].map(mean)

'''

from category\_encoders import TargetEncoder

targetencoder = TargetEncoder()

X[:,0] = encoder.fit\_transform(X[:,0], y) #You must pip install

#category\_encoders library.

'''

#Now split the data before continue:

X=death.loc[:, death.columns != 'Life expectancy'].iloc[:,:].values # Independent variables

y=death.iloc[:,3].values #Dependent variables

#You can also split the data using the below method:

'''

d1 = death.iloc[:,:3].values

d2 = death.iloc[:,4:].values

df1 = pd.DataFrame(d1)

df2 = pd.DataFrame(d2)

X = pd.concat([df1, df2], axis=1)

y=death.iloc[:,3].values

'''

#Deal with the year:

'''

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder

columntransformer=ColumnTransformer(transformers=[('encoder',OneHotEncoder(),[1])],remainder= 'passthrough')

X=columntransformer.fit\_transform(X)

X=X[:,1:] # To remove the first col

'''

#OR you can use label encoder for better performance:

''

#Deal with the year:

labelencoder=LabelEncoder() #Object defined

death['Year']=labelencoder.fit\_transform(death['Year'])

#The data now is ready for test and train:

# Splitting the dataset into the Training set and Test set

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0)

#Use linear regression:

from sklearn.linear\_model import LinearRegression

linearregression = LinearRegression()

linearregression.fit(X\_train,y\_train)

#Define the Y for prediction values

y\_pred = linearregression.predict(X\_test)

#Test the accuracy:

import sklearn.metrics as acc

print("Mean absolute error =", round(acc.mean\_absolute\_error(y\_test, y\_pred), 2))

print("Mean squared error =", round(acc.mean\_squared\_error(y\_test, y\_pred), 2))

print("Median absolute error =", round(acc.median\_absolute\_error(y\_test, y\_pred), 2))

print("Explain variance score =", round(acc.explained\_variance\_score(y\_test, y\_pred), 2))

print("R2 score =", round(acc.r2\_score(y\_test, y\_pred), 2))

#Now apply the multiple regression and retest the accuracy:

#Test the correlation to ensure the droped col:

#correlation to target values:

corr\_matrix=death.corr()

coef=corr\_matrix["Life expectancy"].sort\_values(ascending=False)

#correlation to positive values:

pos\_corr=coef[coef>0]

#correlation to negative values:

neg\_corr=coef[coef<0]

X=np.append(arr=np.ones((2938,1)),values=X,axis=1)

import statsmodels.api as sm

def reg\_ols(X,y):

columns=list(range(X.shape[1]))#it will do it just one time

a={}

for i in range(X.shape[1]):

X\_opt=np.array(X[:,columns],dtype=float) #every time X\_opt will change depend on the columns

regressor\_ols=sm.OLS(endog=y,exog=X\_opt).fit()#regressor between x\_opt and y

pvalues = list(regressor\_ols.pvalues)#save pvalues column as a list

d=max(pvalues)#find the maximum value in the list

if (d>0.06):#if the maximum value bigger than 0.09 then we need to drop the column

for k in range(len(pvalues)):#for loop to check the values in pvalues to find where is the max

if(pvalues[k] == d):#if the value of in the index = max the delete it

a[k]=d

del(columns[k])

return(X\_opt,regressor\_ols,a)

X\_opt,regressor\_ols,a=reg\_ols(X, y)

regressor\_ols.summary()

#splitting the dataset

from sklearn.model\_selection import train\_test\_split

X\_opt\_train,X\_opt\_test,y\_opt\_train,y\_opt\_test=train\_test\_split(X\_opt,y,test\_size=1/3,random\_state=0)

#Training the Simple Linear Reg Model on the Training set

from sklearn.linear\_model import LinearRegression

linearRegression=LinearRegression()

linearRegression.fit(X\_opt\_train,y\_opt\_train)

y\_opt\_pred=linearRegression.predict(X\_opt\_test)

print('After applying MLR: ')

print("Mean absolute error =", round(acc.mean\_absolute\_error(y\_opt\_test,y\_opt\_pred), 2))

print("Mean squared error =", round(acc.mean\_squared\_error(y\_opt\_test,y\_opt\_pred), 2))

print("Median absolute error =", round(acc.median\_absolute\_error(y\_opt\_test,y\_opt\_pred), 2))

print("Explain variance score =", round(acc.explained\_variance\_score(y\_opt\_test,y\_opt\_pred), 2))