Project “Suicide Analysis”

Course: BDOS - 20/21

Student:
Loretucci Lorenzo (1903794)

1.Introduction

The aim of the project is conduct EDA (Exploratory Data Analysis) on World Health Organization[1] suicide dataset. It also did data engineering on a dataset and two different machine learning tasks: Regression and Classification. As a bonus part it has been used Pycaret.

2. Dataset

The dataset coming from a kaggle competition (link) hosted two years ago with differents suicide information around the world. There are 12 columns where 9 are coming from WHO dataset (*country, year, sex, age, suidice_no, population, suicides/100k_pop, country-year and generation*) and the others from Work Bank Dataset[2] (*HDI_for_year, gdp_for_year* and *gdp_per_capita*) as specified on Kaggle description. All data travels years 1985-2016.

3. EDA

For EDA I decided to split the analysis in sub-section divided by themes.

3.1 Gender analysis

In this section I conduct an analysis by sex and year. The total number of suicide throw 30 years for female is 1.559.510 and for male is 5.188.910.
I decide to plot the suicide number over all years. (fig.1)

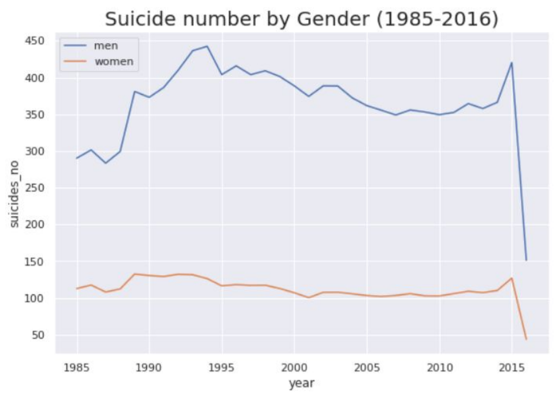

fig.1

Data speaks cleary, men commit suicide more common than woman. There is a negative peak at the end because, how i will show in point 3.3, for year 2016 there are few information.

3.2 Age and Generation

Similar to the above analysis I plot different age suicide number through years (fig.2).

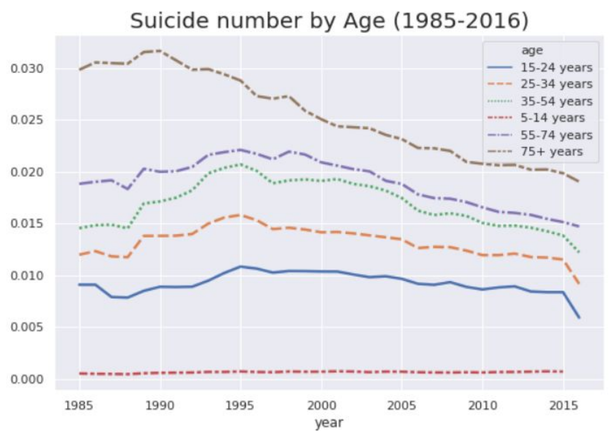

fig.2

In general the trend to commit suicide is in decline for each group. But from another plot we can see a strange situation for ages 5-14, first picture in second row (fig.3).

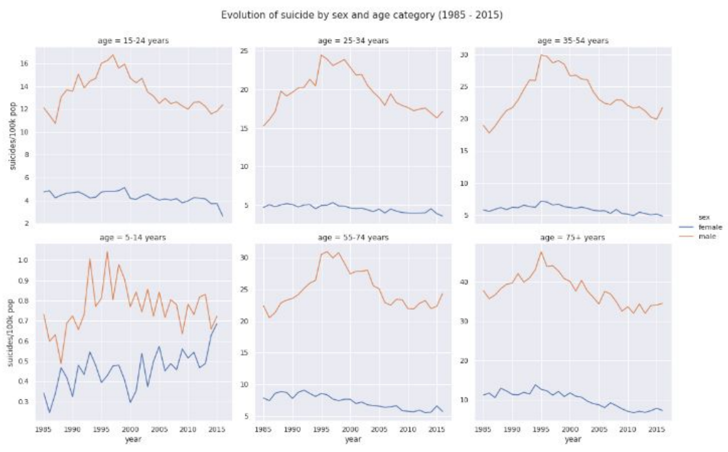

fig.3

I also sum all number of suicide for each generation in the last 30 years

Generation	Number of Suicide
Boomers	2.284.498
Silent	1.781.744
Generation X	1.532.804
Millenials	623.459
G.I Generation	510.009
Generation Z	15.906

Of course the Generation Z have the lower number of suicide because is the last generation.

3.3 World analysis

In this section, I have compared the suicide rates of all countries in the world. Some plots are very complex so I recommend that you see the notebook for more details. On the plot below (fig.4) it's possible to see the suicide trend. As written in section 3.1 for 2016 the dataset is poor.

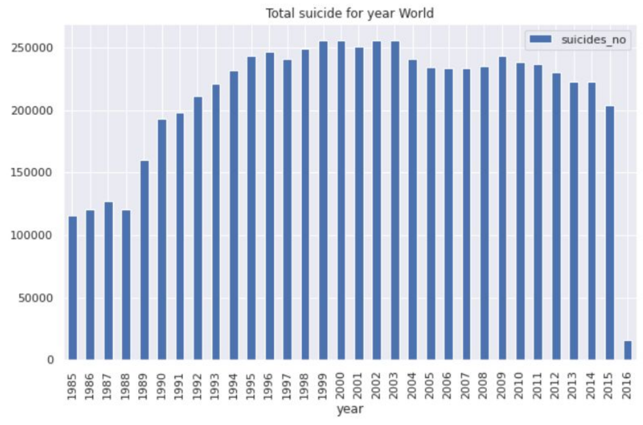

fig.4

In this analysis it was possible to show how some countries, 3 in particular, have a high number of suicides. For this reason I decided to conduct another analysis only with the focus on these specific countries.

3.4 Top 5 country with highest suicide

From the analysis of suicide trends it emerged that there are countries with a high number of suicides. These countries outweigh the others quite significantly (fig.5).

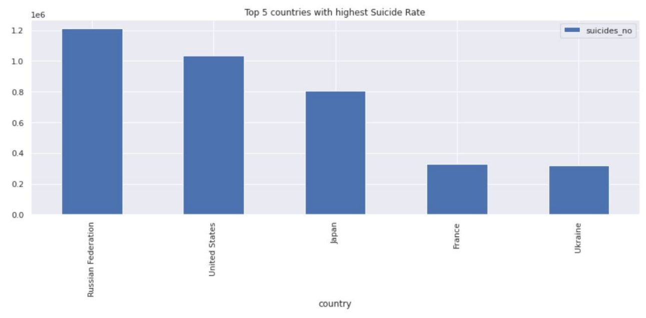

fig.5

Russian Federation, United States and Japan are bigger than the other but of course we have to relate them with the population . In fact in relation to the population we have: Lithuania, Sri Lanka and Russian Federation. Other countries are in the middle of the plot that you can find on the notebook (the plot unfortunately it's too large to be put into the report).
Another interesting plot that I did compare the relation between suicide and GDP per capita (fig.6). In other I plot the relationship for Russin Federation, Japan and USA:

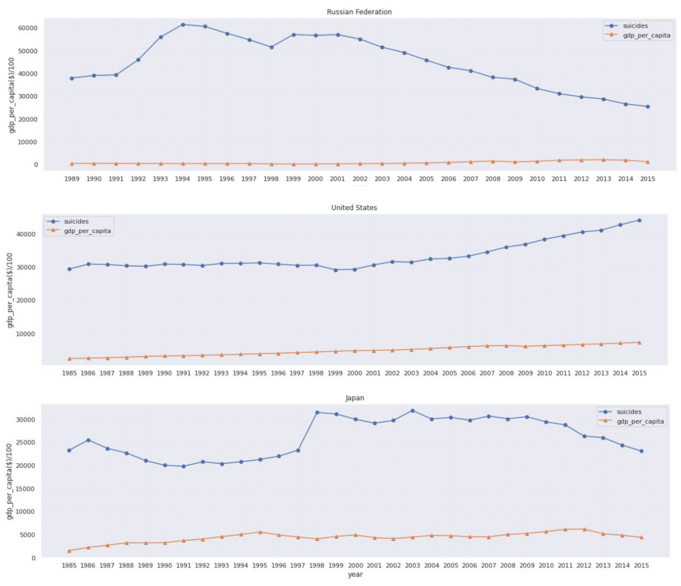

Fig.6

As the plots for Japan and Russian Federation show, there seems to be no relationship between the two variables. On the other hand, for the United States it seems that by increasing, albeit slightly, the GDP increases suicides.

4. Data Engineering

Thanks to the official origin of this dataset, making data engineering and feature engineering it was easy. First of all during the EDA part I saw that the columns HDI for year (Human Development Index) are full of missing values. In support of my reason I have made a plot that demonstrates this visually (fig.7).

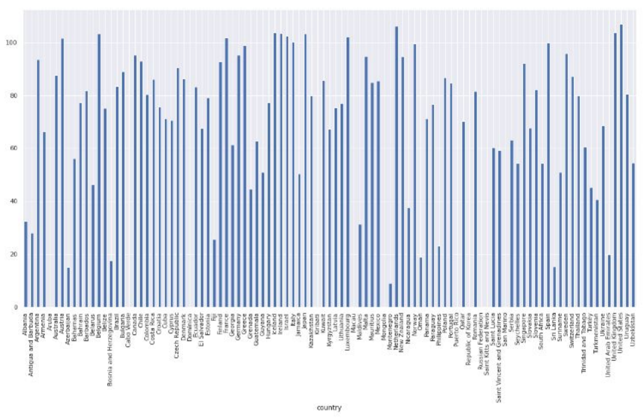

fig.7

For this reason I decided to write an algorithm that, set a threshold of missing values on the total (40% in my case), eliminates all the countries below this value. Eventually 78 countries remained, and 23 were eliminated. Of those that remained, the missing values were set = 0. The *country-year* column has been deleted because it is not relevant.
Another important point was that of categorical variables. For this part I wrote a function "**return_values**" that takes as input the column to be transformed and applies a manual label encoding (no library used). The transformed variables are: age, sex, country, generation.
A difficult column to deal with was gdp_per_year. This was in fact saved as an object as the values were extremely high. My idea was to convert it first to int64 and then convert it to logarithmic scale.
The same procedure was also applied to the column *population* and *suicide_no* now *log_pop* and *log_suicide*.

5. Machine Learning

In order to cover the whole first part of the course program I decided to create two different tasks to solve respectively a **regression** problem and a **classification** problem. All models that I use come from scikit-learn[3] library.

5.1 Regression Problem

To solve a regression problem I decided to set the *log_suicide* column as the target variable.
The information concerning the split between train and test are: *X_train* (17266, 11), *X_test* (7400, 11), *y_train* (17266,) and *y_test* (7400,).
The first model used for this task is a simple linear regression with the module .**LinearRegression()**.

The results obtained on *X_test* using different metrics are:

Accuracy	0.84
MSE	0.82
r-squared	0.16
RMSE	0.90

As we can see from the table above the results are good but not great. I did a plot to show it (fig.8).

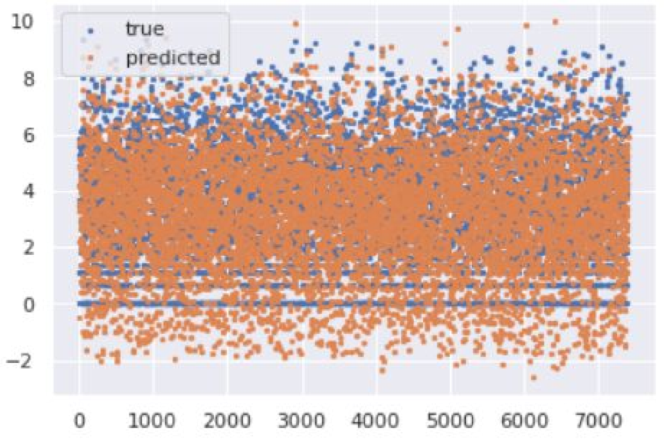

fig.8

The second model that I used to have better results is a random forest with module **.RandomForest.Regressor()**.
The results obtained with a random forest are more strong on the *X_test*:

Accuracy	0.99
MSE	0.00
r-squared	0.00
RMSE	0.02

Perhaps more than strong we can define them almost perfectly. With results like this I think it is better to have more data available as the model could actually be overfitting.

5.2 Classification Problem

For this task my idea was to create a new column, *fatality_rate*. This column was obtained with the **np.where()** function by comparing the *suicides/100k pop* column with its average. If the average is higher, I

insert in line 0 otherwise 1. At the end I put the fatality_rate column as my target. The information concerning the split between train and test are: *X_train* (18499, 11), *X_test* (6167, 11), *y_train* (18499,) and *y_test* (6167,).
The first model used for this task is a simple logistic regression with the module .**LogisticRegression()**.
The results obtained on *X_test* using accuracy is 0.81.
It was possible make a classification report:

	precision	**recall**	**f1-score**	**support**
0	0.69	0.73	0.71	1872
1	0.88	0.85	0.87	4295
accuracy			0.82	6167
macro avg	0.78	0.79	0.79	6167
weighted avg	0.82	0.82	0.82	6167

The results are excellent, but as for the linear model I decided to apply another algorithm to try to get better results.
The second model that I used to have better results is a random forest classifier with the module **.RandomForest.Classifier()**.
The results obtained with a random forest are more strong on the *X_test* with an accuracy of 0.99.
The results obtained with the random forest algorithms always give perfect results. Upstream of this we can say that the data in possession are not very many, certainly with more data the conclusions could be better.

6. Bonus part - Pycaret

As a bonus part of the project I decided to use the Pycaret library. Installing it was not immediate but once used it turned out to be easy and immediate.
The convenience of the library lies in being able to have an overview of all the models in a single code string.
Here too I have applied both the regression and classification tasks. The results obtained are more are less similar.

6.1 Compare Results: Linear Problem

Results for **linear regression algorithm**:

	Pycaret	**MyModel**
R2	0.89	0.16
MSE	0.51	0.82
RMSE	0.71	0.90

Results for **random forest regressor**:

	Pycaret	**MyModel**
R2	0.99	0.99
MSE	0.00	0.00
RMSE	0.03	0.02

6.2 Compare Results: Classification Problem

Results for **logistic regression algorithm**:

	Pycaret	**MyModel**
accuracy	0.99	0.81

Results for **random forest classification**:

	Pycaret	**MyModel**
accuracy	0.99	0.99

References

[1] World Health Organization [link]
[2] Work Bank Dataset [link]
[3] scikit-learn [link]