

# **Suicide Prediction using Machine Learning Algorithm**

Submitted by

**Charan Royston Lobo**

**182006**

**MSc Big Data Analytics  
AIMIT, St. Aloysius college  
Mangalore**

Submitted in Partial Fulfillment of the Requirements for the Award of the Degree of

**Master of Big Data Analytics (MSc)**

Under the guidance of

**Mrs. Laveena C Crasta**

**HOD, Department of Big Data Analytics  
AIMIT, St. Aloysius College  
Mangalore**

Submitted to



**ST ALOYSIUS INSTITUTE OF MANAGEMENT AND INFORMATION  
TECHNOLOGY (AIMIT)  
ST ALOYSIUS COLLEGE (AUTONOMOUS)  
MANGALURU, KARNATAKA**

**2020**



**ST ALOYSIUS INSTITUTE OF MANAGEMENT AND INFORMATION TECHNOLOGY  
ST ALOYSIUS COLLEGE (AUTONOMOUS)  
MANGALORE, KARNATAKA**

**CERTIFICATE**

This is to certify that the project titled

**Suicide Prediction using Machine Learning Algorithm**

Submitted by

**Charan Royston Lobo**  
**182006**  
**MSc Big Data Analytics**  
**AIMIT, St. Aloysius college**  
**Mangalore**

In partial fulfillment of the requirements for award of degree of Master of **Big Data Analytics** of  
Mangalore University, is a bonafide record of the work carried out at

**During the year**  
**2019-2020**

Mrs. Laveena C Crasta  
HOD  
Department of Big Data Analytics  
AIMIT, St. Aloysius College,  
Mangaluru-575 022.

Prof Santhosh Rebello  
Dean, School of IT  
PG Dept of Information Technology,  
AIMIT, St. Aloysius College,  
Mangaluru-575 022.

**Examiners**

Rev. Dr. Melvin Pinto SJ  
Director,  
St. Aloysius College  
Mangaluru-575 022.

1. Mrs. Shubharekha

2. Mrs. Laveena Crasta

### **CERTIFICATE OF AUTHENTICATED WORK**

This is to certify that the domain seminar project report entitled **Suicide Prediction using Machine Learning Algorithm** submitted to Aloysius Institute of Management and Information Technology (AIMIT), St Aloysius College, Mangalore affiliated to Mangalore University in partial fulfillment of the requirement for the award of the degree of MASTER OF SCIENCE (MSC BIG DATA ANALYTICS) is an original work carried out by Mr. **Charan Royston Lobo** Register no: **182006** under my guidance. The matter embodied in this domain seminar project is authentic and is genuine work done by the student and has not been submitted whether to this University, or to any other University / Institute for the fulfillment of the requirement of any course of study.

-----  
Signature of the Student:

Date: 11-June-2020

Charan Royston Lobo  
Upper Nekkare House,  
Pedamale Post,  
Kelrai, Mangalore 575 029  
Register No: **182006**

-----  
Signature of the Guide

Date: .....

Mrs. Laveena C Crasta, HOD  
Department of Big Data Analytics  
AIMIT, St Aloysius College  
Mangalore 575022

## Table of contents

1. INTRODUCTION.....	1
1.1. Aim and Objectives.....	1
1.2. Problem definition.....	1
2. LITERATURE REVIEW.....	1
3. DATA DESCRIPTION.....	2
3.1. Codebook.....	2
3.2. Structure of the data.....	3
3.3. Coding if done (with Screenshots).....	3
3.4. Missing values and treatment.....	4
3.5. Outlier detection and treatment.....	4
4. EXPLORATORY DATA ANALYSIS.....	5
4.1. Need and importance.....	5
4.2. EDA preformed details.....	5
4.3. Univariate analysis.....	5
4.3.1. Categorical data.....	5
4.3.2. Tabular representation (contingency table).....	6
4.3.3. Use of graphical representation.....	8
4.3.4. Numeric data.....	8
4.3.5. Descriptive statistics .....	9
4.3.5.1. Central tendency.....	9
4.3.5.2. Measure of spread.....	9
4.3.5.3. Quartiles .....	10
4.3.6. Tabular representation (frequency table).....	10
4.3.7. Use of graphical representation.....	11
4.4. BIVARIATE/ MULTIVARIATE DATA ANALYSIS.....	11
4.4.1. Categorical data.....	11
4.4.2. Tabular representation (contingency table/ cross tabulation).....	12
4.4.3. Use of graphical representation.....	13
4.4.4. Numeric data.....	15
4.4.5. Descriptive statistics .....	16
4.4.5.1. Central tendency.....	16
4.4.5.2. Measure of spread.....	16
4.4.5.3. Quartiles .....	16
4.4.6. Tabular representation (frequency table).....	16
4.4.7. Use of graphical representation.....	16
5. METHODOLOGY.....	17
5.1. Model 1.....	17
5.1.1. Definition of the model.....	17
5.1.2. Assumptions.....	17
5.1.3. Algorithm.....	17
5.2. Model 2.....	18
5.2.1. Definition of the model.....	18
5.2.2. Assumptions.....	18
5.2.3. Algorithm.....	18
5.3. Model 3.....	19
5.3.1. Definition of the model.....	19
5.3.2. Assumptions.....	19
5.3.3. Algorithm.....	19
6. IMPLEMENTATION OF THE ALGORITHM.....	20
6.1. Screenshots of the code with the necessary tables and diagrams.....	20
6.2. Ensembling .....	27
6.3. User interface.....	29
7. CONCLUSIONS/INTERPRETATIONS.....	31
8. REFERENCES.....	31

## **1. Introduction**

Suicidal ideation refers to thoughts of killing oneself or planning suicide, while suicidal behavior is often defined to include all possible acts of self-harm with the intention of causing death.

After being recognized as a public health priority by the WHO (World Health Organization) various studies have been going out for its prevention. It is one of a serious health problem and it is preventable and can be controlled by proper interventions and study in the field.

### **1.1. Aim and Objectives**

The goal of this project is to build a Machine Learning supervised model using a number of socioeconomic metrics of 101 Countries and try to determine which factors might have a statistically significant correlation with National Suicide Rates. This study determines to find signals correlated to increased suicide rates among different cohorts globally, across the socio-economic spectrum.

### **1.2. Problem definition**

Every suicide is a tragedy that affects families, communities and entire countries and has long-lasting effects on the people left behind. Suicide occurs throughout the lifespan and was the second leading cause of death among 15 to 29- year olds globally in 2016. Knowledge of the most commonly used suicide methods is important to devise prevention strategies. Using this data, we will be able to predict the estimated number of suicides given the various variables that effect these numbers.

## **2. Literature review**

The overall trend globally is toward continued human development improvements, with many countries moving up through the human development categories, out of the 189 countries for which the HDI is calculated, 59 countries are today in the very high human development group and only 38 countries fall in the low HDI group. Just eight years ago in 2010, the figures were 46 and 49 countries respectively. In South Africa, a study carried out by Flisher revealed a suicide rate of 17/100 000 in the year 1990, which is slightly higher than the world average of 16/100 000. The mean annual suicide mortality rate in the age range 15–25 was found to be higher than in other age ranges. From 1993 to 2004 the rate of suicide among people over the age of 14 was 10– 13/100,000 in England and 13/100 000 in the UK and Ireland. Worldwide injury mortality estimates for 2008 as well as trends of the suicide rate from 1950 to 2009 were analysed. Results: Suicides in the world amount to 782 thousand in 2008 according to the WHO estimate, which is 1.4% of total mortality and 15% of injury mortality. The suicide rate for the world as a whole is estimated at 11.6 per 100,000 inhabitants.

### 3. Data description

The dataset consists of with 12 attributes, ranging from the year 1985 to 2016. Out of 12 attributes, there were 6 attributes which were categorical in nature and 6 that were scaled in numerical. The attributes categorical in nature were country, year, sex, age, country\_year and generation and the remaining variables such as suicide number, population, suicides/100k pop, gdp per capita, gdp per year and hdi were numerical.

#### 3.1. Codebook

Codebook is used for mapping between the words. It stores the characters and their meanings into numerical values or symbols as shown below;

<b>Id</b>	<b>Variable Name</b>	<b>Description</b>
1	countries	Total of 101 countries
2	year	year ranging from 1985 - 2016
3	sex	0-male
		1-female
4	age	0- (5-14 years)
		1-(15-24 years)
		2-(25-34 years)
		3-(35-54 years)
		4-(55-74 years)
		5-(75+ years)
5	Suicide number	number of people committing suicide
6	population	total number of people living in that country
7	suicides/100k pop	suicides occurring per 100k population
8	country_year	combination of column year and country
9	gdp_per_capita	Human Development Index per year
10	HDI for year	Human Development Index for year
11	gdp_for_year (\$)	Gross domestic product
12	generation	0-Boomers
		1-Generation X
		2-Generation Z
		3-G.I. Generation
		4-Millennials
		5-Silent

### 3.2. Structure of the data

Structure of the data is very important to understand for any type of analysis. In Python we make use of pandas `str()` class, which defines the data structure of the dataset. Below figure is the data structure of our dataset, where `int64` and `float64` datatype indicates numerical and object datatype indicates likely string values. We can observe that 'HDI for year' has 8364 instances while other columns have 27820 instances, indicating missing values. This will be treated in further analysis. We can notice that columns 'suicides/100k pop' and 'country-year' are redundant columns, which needs to be dropped, also 'gdp\_for\_year' has object data, on inspection it has commas between values that needs to be removed. Proper convention of columns needs to be done on 'gdp\_for\_year(\$)' and 'gdp\_per\_capita(\$)'

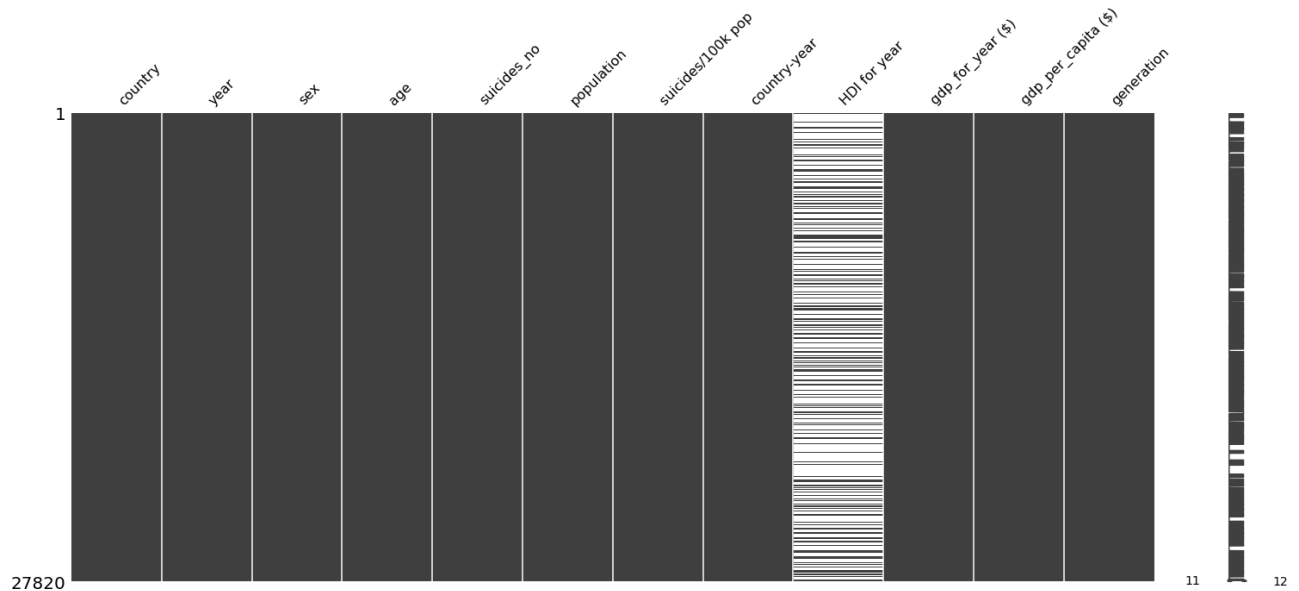
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27820 entries, 0 to 27819
Data columns (total 12 columns):
country                27820 non-null object
year                  27820 non-null int64
sex                   27820 non-null object
age                   27820 non-null object
suicides_no           27820 non-null int64
population            27820 non-null int64
suicides/100k pop     27820 non-null float64
country-year          27820 non-null object
HDI for year          8364 non-null float64
gdp_for_year ($)      27820 non-null object
gdp_per_capita ($)    27820 non-null int64
generation            27820 non-null object
dtypes: float64(2), int64(4), object(6)
memory usage: 2.5+ MB
```

### 3.3. Coding if done (with Screenshots)

```
dataframe.drop(['country-year','suicides/100k pop'],axis =1,inplace = True )
dataframe.rename(columns={"gdp_for_year ($) ":"gdp_for_year_usd","gdp_per_capita
($) ":"gdp_per_capita_usd"},inplace=True)
dataframe["gdp_for_year_usd"] =
dataframe["gdp_for_year_usd"].str.replace(',','').astype('int64')
dataframe['year'] = dataframe['year'].apply(str)
```

### 3.4. Missing values and treatment

We previously noticed that our dataset has missing values in 'HDI for year'. In the below figure we observe that large number of instances are missing from that column, nearly 70% of the data, hence imputing will casue bias in our analysis and final model.



### 3.5. Outlier detection and treatment

Outliers are extreme values that deviate from other observations on data, they may indicate a variability in a measurement, experimental errors or a novelty. In other words, an outlier is an observation that diverges from an overall pattern on a sample. Outliers can have bad impact on dependent variable. Let us look at our dependent variable and its variation in the below figure.

```
count    27820.000000
mean      242.574407
std       902.047917
min        0.000000
25%        3.000000
50%       25.000000
75%      131.000000
max     22338.000000
Name: suicides_no, dtype: float64
```

We observe that suicide rate's range from 0 to 22338, with an average of 25. This implies that our variation is skewed and we have less information about the suicide rates that are above 131. Therefore, the suicide rates above 131 are considered to be outliers and are necessary to be dropped in our analysis.



## 4. Exploratory data analysis

**Exploratory Data Analysis (EDA)** is the process of analysing and visualizing the data to get a better understanding of the data. It is usually performed at the beginning of the data.

### 4.1 Need and importance

It helps in identifying different trends, patterns and relationship of one variable and also with other variables and helps in suggesting variables for modelling strategies. We will observe how suicide numbers are affected due to different factors in the dataset, and also the factors that have least affect on suicide numbers, we shall also analyze the cause of highest suicide numbers and the countries with lowest suicide cases.

### 4.2 EDA preformed details

In Python, we have visualization libraries such as Matplotlib, seaborn, etc. The following variables will be considered into exploratory data Analysis. In this data, out of 12 variables, 6 were categorical and 6 where numerical. We have also two redundant columns that have been dropped, one from each categorical and numerical. We can see that the data from 2016 only has 16 countries and actually does not include an age group (4~15 years). We just have to be sure we don't use these data to mess with the rest of the analysis.

country age		
year		
2016	16	5
1985	48	6

```
dataframe = dataframe[dataframe['year'] != '2016']
```

### 4.3 Univariate analysis

Univariate analysis is the simplest form of analyzing data. It deals with only one variable. It explores each variable in a dataset, separately. It looks at the range of values. It describes the pattern of response to the variable. It describes each variable on its own.

#### 4.3.1 Categorical data

We have dropped 1 categorical variable 'country-year'. Using the python libraries Matplotlib and Seaborn, a descriptive analysis of the categorical data is performed in the following figures.

	country	year	sex	age	generation
count	20708	20708	20708	20708	20708
unique	100	31	2	6	6
top	Singapore	2009	female	5-14 years	Millenials
freq	372	821	11506	4552	5015

### 4.3.2 Tabular representation (contingency table)

Contingency table shows the distribution of variables in rows and columns. It is also used to summarize the relationship between many categorical variables.

```
female    11506
male       9202
Name: sex, dtype: int64

female    55.56%
male      44.44%
Name: sex, dtype: object
Contingency table for Gender - fig (6)
```

```
Millenials    5015
Generation X   4727
Silent         4402
Boomers       3031
G.I. Generation 2081
Generation Z   1452
Name: generation, dtype: int64

Millenials    24.22%
Generation X   22.83%
Silent         21.26%
Boomers       14.64%
G.I. Generation 10.05%
Generation Z    7.01%
Name: generation, dtype: object
```

Contingency table for generation - fig (7)

5-14 years	4552
75+ years	3675
15-24 years	3518
25-34 years	3371
55-74 years	2924
35-54 years	2668

Name: age, dtype: int64

5-14 years	21.98%
75+ years	17.75%
15-24 years	16.99%
25-34 years	16.28%
55-74 years	14.12%
35-54 years	12.88%

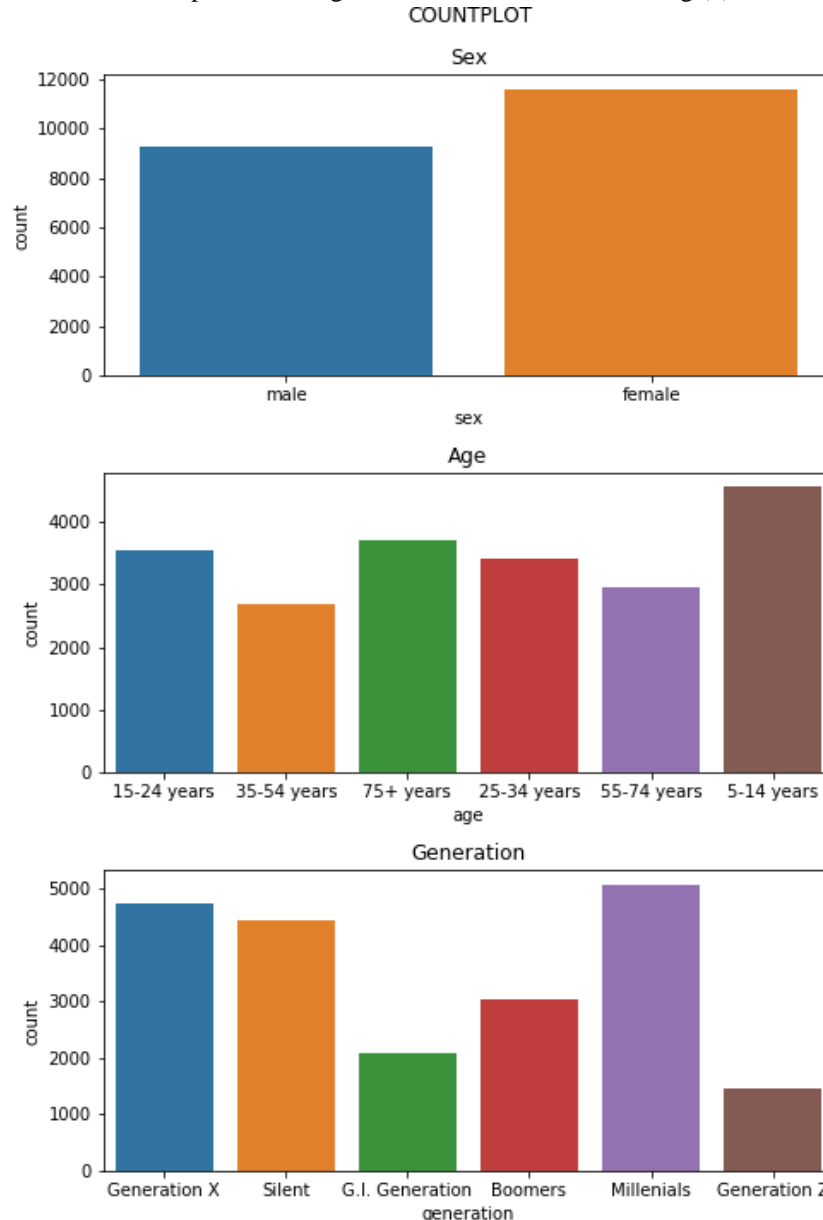
Name: age, dtype: object

**Contingency table for Age- fig (8)**

### 4.3.3 Use of graphical representation

Graphical representation refers to the use of intuitive charts to clearly visualize and simplify data sets.

Bar plots for categorical variables in the dataset-fig (9)



Bar plots for categorical variables in the dataset-fig (9)

### 4.3.4 Numeric data

Numerical data is a data type expressed in numbers, rather than natural language description. Sometimes called quantitative data, numerical data is always collected in number form. Numerical data differentiates itself with other number form data types with its ability to carry out arithmetic operations with these numbers. In our dataset, we have now 4 numerical columns suicide\_no, population, gdp\_for\_year\_usd and gdp\_per\_capita\_usd.

### 4.3.5 Descriptive statistics

A descriptive statistic is a summary statistic that quantitatively describes or summarizes features from a collection of information.

#### 4.3.5.1 Central tendency

In a distribution, measures of central tendency identify where the data is centred. Mean and Median are commonly used measures for numeric central tendency.

```
Mean:
suicides_no      2.505264e+01
population      7.551407e+05
gdp_for_year_usd 1.625539e+11
gdp_per_capita_usd 1.616373e+04
dtype: float64

Median:
suicides_no      9.000000e+00
population      2.790205e+05
gdp_for_year_usd 2.338695e+10
gdp_per_capita_usd 8.257000e+03
dtype: float64
```

#### 4.3.5.2 Measure of spread

	suicides_no	population	gdp_for_year_usd	gdp_per_capita_usd
count	20708.000000	2.070800e+04	2.070800e+04	20708.000000
mean	25.052637	7.551407e+05	1.625539e+11	16163.726096
std	32.917509	1.743205e+06	5.597115e+11	19428.014903
min	0.000000	2.780000e+02	4.691962e+07	251.000000
25%	1.000000	4.847525e+04	4.767303e+09	3342.000000
50%	9.000000	2.790205e+05	2.338695e+10	8257.000000
75%	38.000000	6.130355e+05	1.222107e+11	22857.000000
max	130.000000	2.027392e+07	1.615526e+13	126352.000000

#### 4.3.5.3 Quartiles

A quartile is a type of quantile which divides the number of data points into four more or less equal parts, or quarters. The first quartile (Q1) is defined as the middle number between the smallest number and the median of the data set. It is also known as the lower quartile or the 25th empirical quartile and it marks where 25% of the data is below or to the left of it (if data is ordered on a timeline from smallest to largest). The second quartile (Q2) is the median of a data set and 50% of the data lies below this point. The third quartile (Q3) is the middle value between the median and the highest value of the data set. It is also known as the upper quartile or the 75th empirical quartile and 75% of the data lies below this point. The above table shows the quartiles with central tendency.

#### 4.3.6 Tabular representation (frequency table)

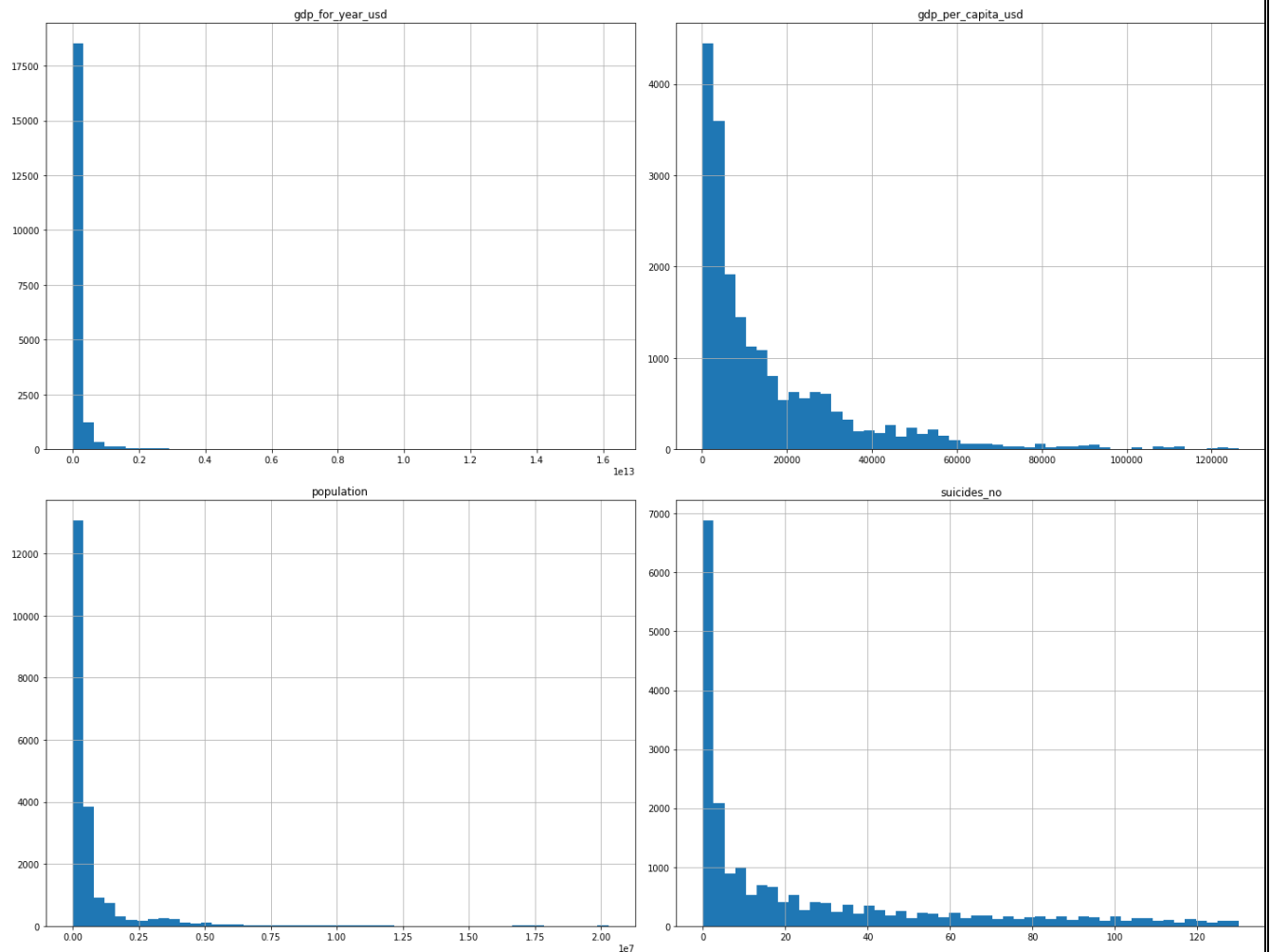
A **frequency table** is a **table** that represents the number of occurrences of every unique value in the variable.

```
female    55.56%
male      44.44%
Name: sex, dtype: object
```

```
Millenials      24.22%
Generation X     22.83%
Silent           21.26%
Boomers          14.64%
G.I. Generation  10.05%
Generation Z      7.01%
Name: generation, dtype: object
Name: age, dtype: object
```

```
5-14 years      21.98%
75+ years       17.75%
15-24 years     16.99%
25-34 years     16.28%
55-74 years     14.12%
35-54 years     12.88%
Name: age, dtype: object
```

### 4.3.7 Use of graphical representation



### 4.4 Bivariate /Multivariate analysis

Bivariate/Multivariate analysis is a set of statistical techniques used for analysis of data that contain more than one variable. This analysis refers to any statistical technique used to analyse more complex sets of data.

#### 4.4.1 Categorical data

The following analysis will be done on categorical columns of our dataset, which include country, year, sex, age and generation.

#### 4.4.2 Tabular representation (contingency table/ cross tabulation)

sex	female	male	All
age			
15-24 years	1959	1559	3518
25-34 years	1951	1420	3371
35-54 years	1590	1078	2668
5-14 years	2302	2250	4552
55-74 years	1682	1242	2924
75+ years	2022	1653	3675
All	11506	9202	20708

Fig shows tabular representation of sex and age

generation	Boomers	G.I. Generation	Generation X	Generation Z	Millenials	Silent	All
sex							
female	1806	1128	2700	732	2653	2487	11506
male	1225	953	2027	720	2362	1915	9202
All	3031	2081	4727	1452	5015	4402	20708

Fig shows tabular representation of sex and generation

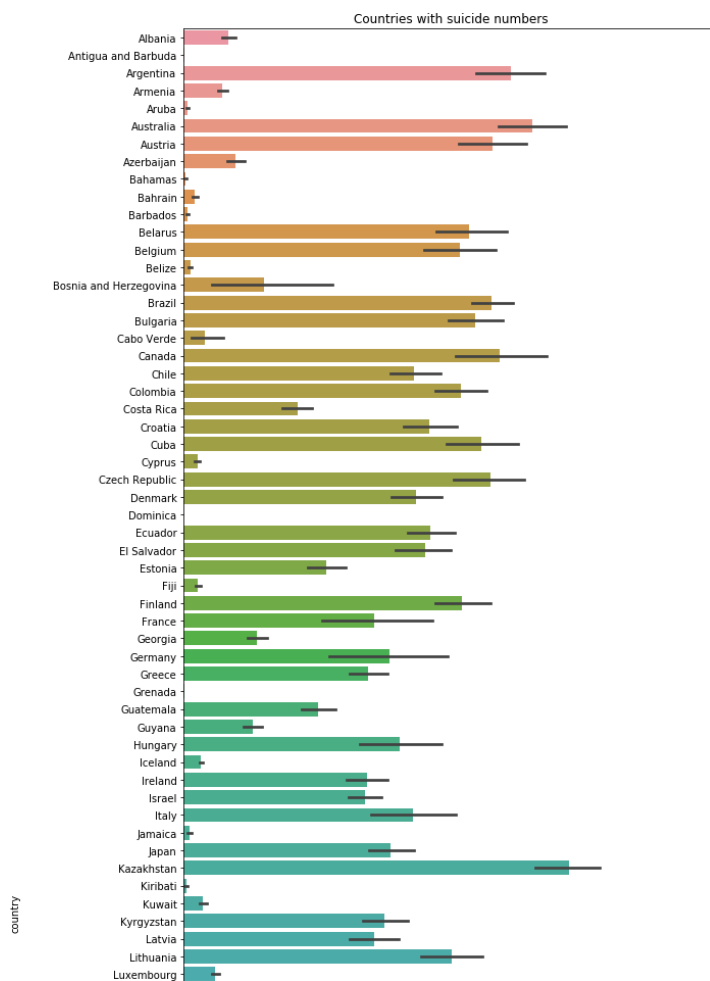
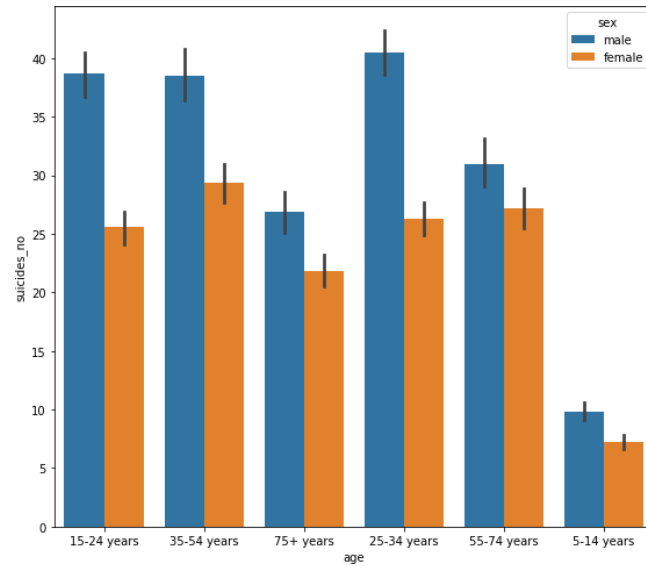
age	15-24 years	25-34 years	35-54 years	5-14 years	55-74 years	75+ years	All
generation							
Boomers	0	797	1741	0	493	0	3031
G.I. Generation	0	0	0	0	402	1679	2081
Generation X	1563	1992	550	622	0	0	4727
Generation Z	0	0	0	1452	0	0	1452
Millenials	1955	582	0	2478	0	0	5015
Silent	0	0	377	0	2029	1996	4402
All	3518	3371	2668	4552	2924	3675	20708

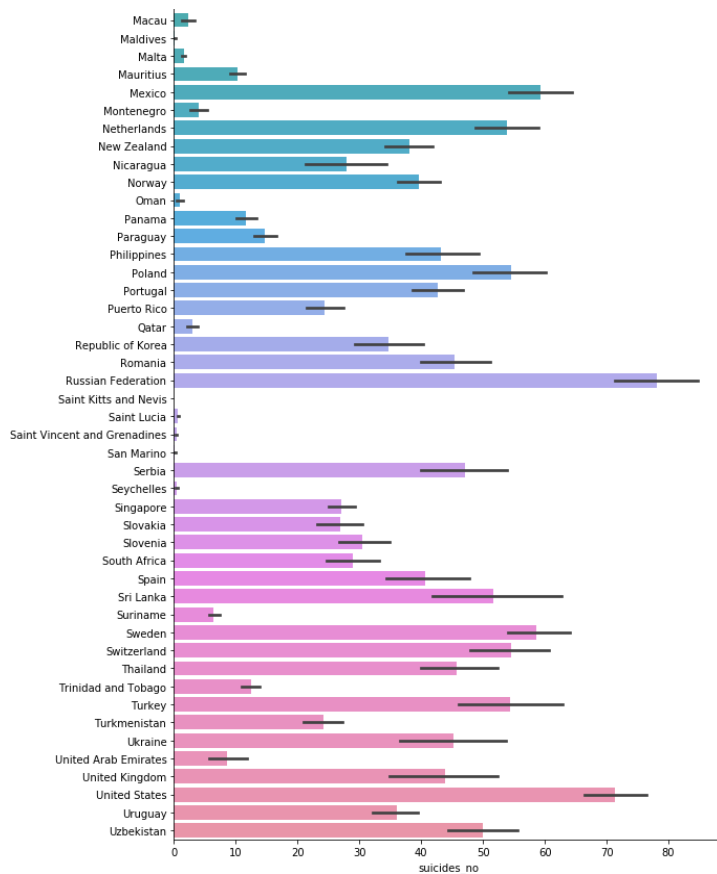
Fig shows tabular representation of age and generation



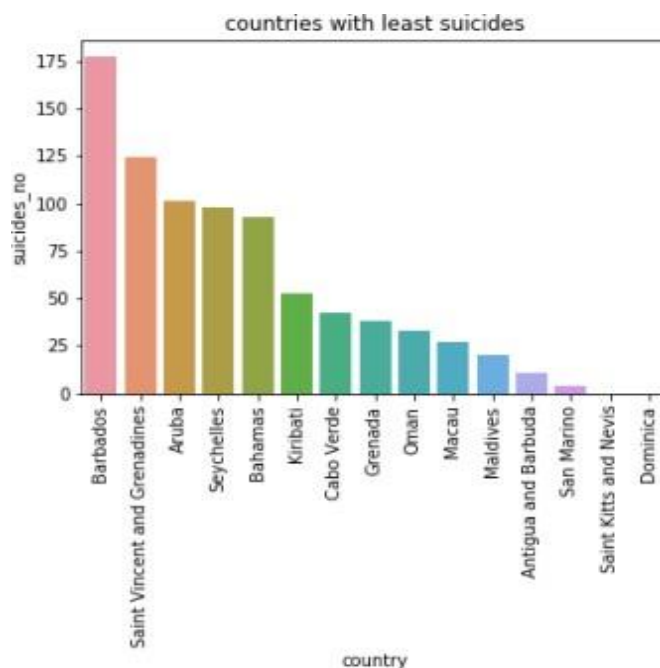
### 4.4.3 Use of graphical representation

Graphical methods for categorical data are performed using matplotlib and Seaborn library



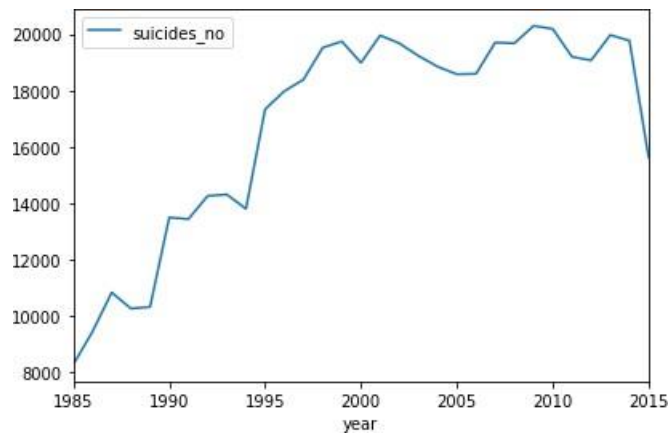


- Suicide rates are higher in Russian Federation, United States and Kazakhstan
- Suicide rates are too low in many countries.
- Suicide rates are moderate in France, Ukraine, Germany, Brazil, Republic of Korea, Poland, Thailand, United Kingdom, Canada, Italy, Mexico, etc.



## Top 15 countries with least suicides

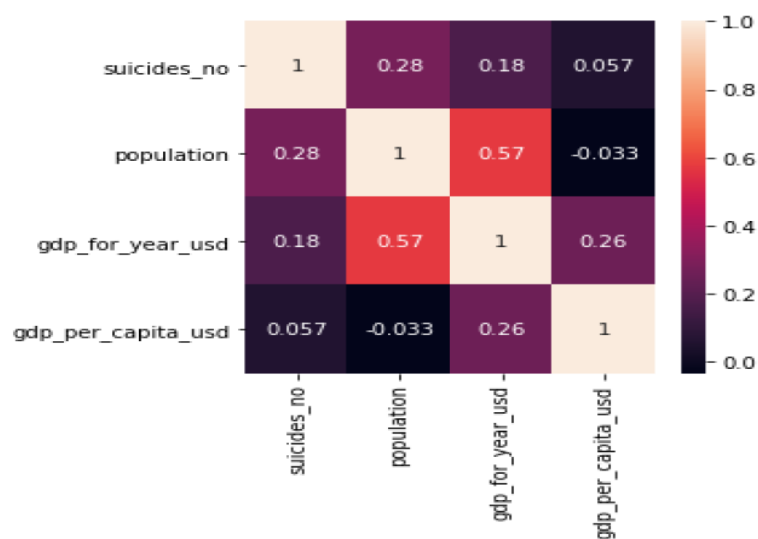
- The suicide count in countries Dominica and Saint Kitts and Nevis is zero.



### 4.4.4 Numeric data

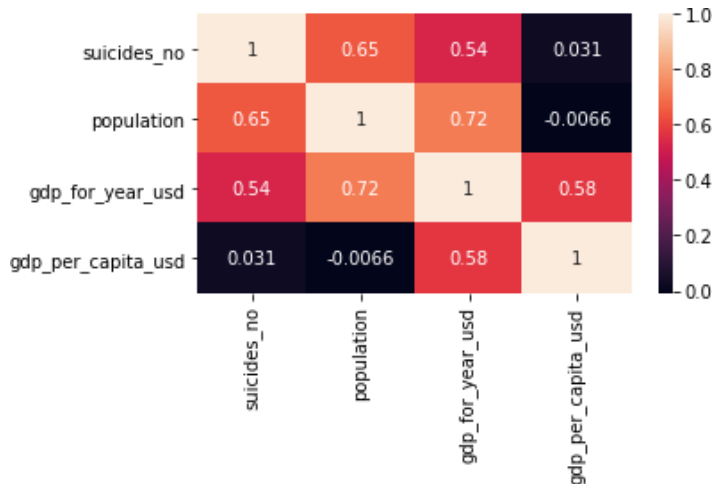
The covariance of two variables  $x$  and  $y$  in a data set measures how the two are linearly related. A positive covariance would indicate a positive linear relationship between the variables, and a negative covariance would indicate the opposite.

## Linear Relationship



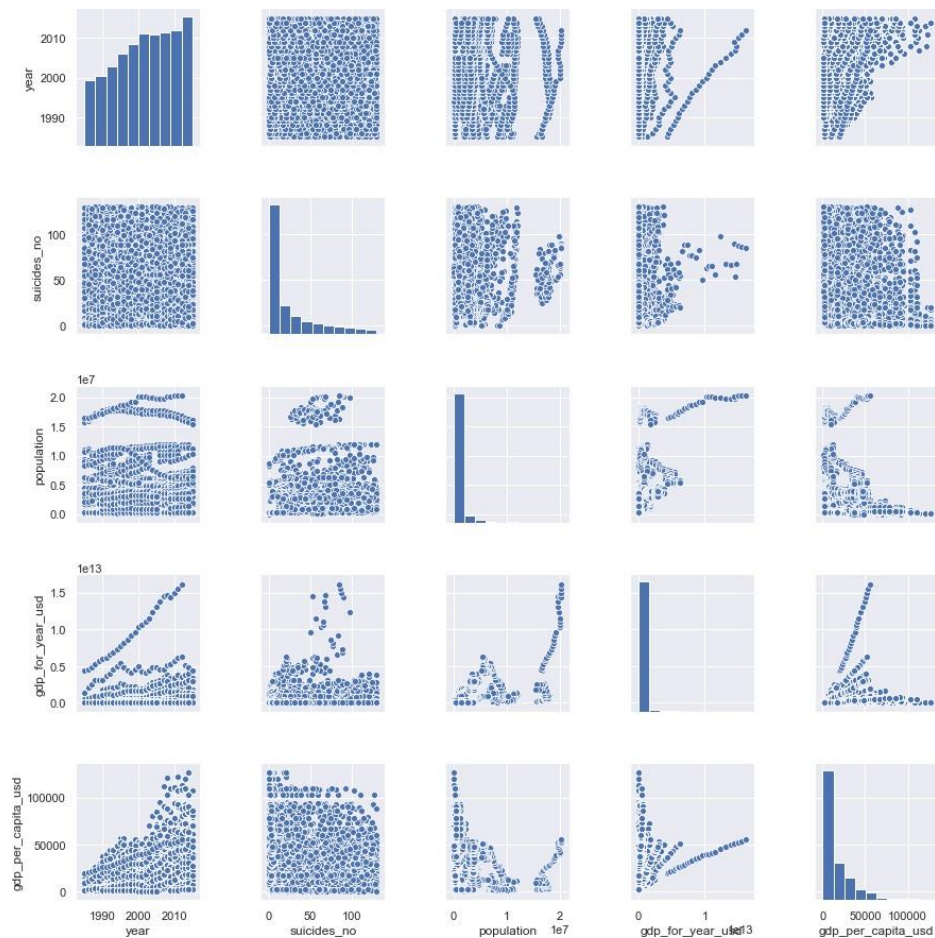
## Non-Linear Relationship

Non-linear relationship is measure using spearman's correlation. This test does not assume gaussian distribution or linear relationship between variables.



### 4.4.5 Use of graphical representation

The graphical representation of scaled variables is normally done with scatter plots. A **scatter plot** is a type of plot or mathematical diagram using Cartesian coordinates to display values for typically two variables for a set of data.



## 5 Methodology

### 5.1 Model 1

Multiple linear regression

#### 5.1.1 Definition of the model

Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. Multiple regression is an extension of linear (OLS) regression that uses just one explanatory variable.

#### 5.1.2 Assumptions

- I. Linear relationship: The model is a roughly linear one. This is slightly different from simple linear regression as we have multiple explanatory variables. This time we want the outcome variable to have a roughly linear relationship with each of the explanatory variables, taking into account the other explanatory variables in the model.
- II. Independent errors: This means that residuals should be uncorrelated.
- III. Outliers/influential cases: As with simple linear regression, it is important to look out for cases which may have a disproportionate influence over your regression model.
- IV. Multicollinearity: Multicollinearity exists when two or more of the explanatory variables are highly correlated. This is a problem as it can be hard to disentangle which of them best explains any shared variance with the outcome. It also suggests that the two variables may actually represent the same underlying factor.
- V. Normally distributed residuals: The residuals should be normally distributed.

#### 5.1.3 Algorithm

Step 1: First Import the data and the required packages, load the required data in excel or csv format and preprocess the data.

Step 2: Create the training and the testing samples from the data. Training set is the one on which we train and fit our model basically to fit the parameters whereas test data is used only to assess performance of model.

Step 3: Create a regression model and fit it with existing data.

Step 4: Check the results of model fitting to know whether the model is satisfactory.

Step 5: Apply the model for predictions.

Step 6: Do model diagnostics.

## 5.2 Model 2

Random forest

### 5.2.1 Definition of the model

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

### 5.2.2 Assumptions

- I. No formal distributional assumptions
- II. Random forests are non-parametric and can thus handle skewed and multi-modal data.
- III. Work's well with both linear and non-linear relationship between dependent and independent variables.
- IV. Do not require scaling of numeric variables.
- V. Do not require monotonic transformations (log) or removal of outliers to a certain extent.

### 5.2.3 Algorithm

Step 1: Import the data and the required packages. First, load the required data in excel or csv format and import the required packages.

Step 2: Create the training and the testing samples from the data. Training set is the one on which we train and fit our model basically to fit the parameters whereas test data is used only to assess performance of model.

Step 4: Build the decision tree.

Step 5: Choose the number Ntree of trees you want to build and repeat step 1 & 2.

Step 6: For a new data point, make each one of your Ntree trees predict the value of Y for the data point, and assign the new data point the average across all of the predicted Y values.

Step 7: Do model diagnostics.

## 5.3 Model 3

Decision Tree regression.

### 5.3.1 Definition of the model

Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, decision tree algorithm can be used for solving regression and classification problems too.

The general motive of using Decision Tree is to create a training model which can use to predict class or value of target variables by learning decision rules inferred from prior data (training data).

The understanding level of Decision Trees algorithm is so easy compared with other classification algorithms. The decision tree algorithm tries to solve the problem, by using tree representation. Each internal node of the tree corresponds to an attribute, and each leaf node corresponds to a class label.

### 5.3.2 Assumptions

- I. At the beginning, the whole training set is considered as the **root**.
- II. Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model.
- III. Records are distributed recursively on the basis of attribute values.
- IV. Order to placing attributes as root or internal node of the tree is done by using some statistical approach.

### 5.3.3 Algorithm

Step 1: Import the data and the required packages. First, load the required data in excel or csv format and import the required packages.

Step 2: Create the training and the testing samples from the data. Training set is the one on which we train and fit our model basically to fit the parameters whereas test data is used only to assess performance of model.

Step 3: Fit decision tree regressor to the dataset.

Step 4: Predicting a new value

Step 5: Visualising the result.

Step 6: Do model diagnostics.

## 6 Implementation of the algorithm

All the 3 algorithms are performed separately and implemented using python.

### 6.1 Screenshots of the code with the necessary tables and diagrams

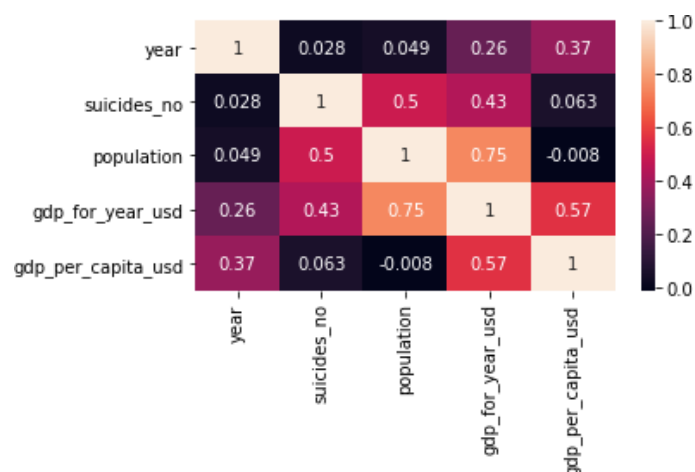
#### Multiple linear regression:

- Importing data and the required libraries.

```
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import pandas as pd
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
dataframe = pd.read_csv("master.csv")
```

- Log Transformation and variable selection

```
dataframe_log = dataframe.copy()
dataframe_log['population'] = np.log(dataframe_log['population'])
dataframe_log['gdp_for_year_usd'] = np.log(dataframe_log['gdp_for_year_usd'])
dataframe_log['gdp_per_capita_usd'] = np.log(dataframe_log['gdp_per_capita_usd'])
```



- Preprocessing and splitting the train and test data



```
X = dataframelog.drop(['suicides_no', 'year', 'gdp_for_year_usd'], axis = 1)
y = dataframelog['suicides_no'].copy()
categorical_features = ['age', 'sex', 'generation', 'country']

full_pipeline = ColumnTransformer([("cat", OneHotEncoder(), categorical_features)], remainder="passthrough")
Xprep = full_pipeline.fit_transform(X)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(Xprep, y, test_size=0.3)
```

- Fitting the model

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression(fit_intercept=True)
regressor.fit(X_train, y_train)

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

- The intercept and coefficient from the above model are as follows

```
regressor.intercept_
```

```
25.073680579510174
```

```
regressor.coef_
```

```
array([ -36.53459126, -33.2167491 , -32.94529348, -69.45414318,
        -28.38310808, -18.12171838, -117.60133827, -101.05426522,
        -36.87148215, -35.72306142, -37.69646804, -32.83166459,
        -37.85057909, -37.68234819, -27.89983206,  32.92900695,
        -6.81483336, -31.06752181,  31.54368893,  7.53871303,
        16.14993316, -41.72591396,  11.77708947,  0.75644263,
         9.98890968,  2.39229837,  9.2342179 ,  14.35829777,
        -17.90460677, -27.40335301,  1.56787913,  4.21964145,
        -1.73529207, -14.14760493, -19.17362653, -16.2919208 ,
         5.09917201,  1.35408685, -6.5293993 ,  6.04293926,
         1.51431138,  33.17415268, -12.69046377, -3.41128866,
         6.18916882, -5.14561653,  11.85090548, -19.62270534,
        -31.64822683, -20.60992216, -22.68642481,  27.59557152,
        -31.81463851,  2.40339242, -0.87424971,  15.25947898,
        -0.64288819, -12.47412495, -20.05908317, -30.51731986,
        -22.8565257 ,  18.19224338,  31.12734843, -13.66847987,
        -8.15935367,  5.55621028,  14.32583216,  11.26655768,
        11.16718111,  11.86791494,  4.78779682, -4.7667651 ,
        -17.61905227, -2.90887989,  5.28533171,  3.61920751,
        -13.64633061,  4.47929851, -24.84270462, -19.68119336,
        -26.93632904, -37.28213974, -6.57269472, -5.20703206,
        -9.0794565 ,  1.04288262, -16.18369277, -8.10692032,
        18.11548141,  42.03885955,  20.66957652,  25.75682251,
        52.37843301,  8.55497752,  32.35396729, -2.7006583 ,
        -9.69467442,  8.28803021, -48.6636351 , -15.56538446,
         1.62343503,  7.42686434,  17.55807413,  17.76445901,
        -18.0913324 , -2.73395253, -24.43836501, -14.46712757,
        -8.13355519, -18.65506787, -16.28445195,  4.91356679,
         1.67568551, -13.59432725,  17.26971263, -2.32766627])
```

- Prediction using test data and train data:

```
y_pred_test = regressor.predict(X_test)
y_pred_train = regressor.predict(X_train)
```

- Evaluate the performance of the algorithm

The Mean Squared Error of the training set is: 20.92887073044551  
The Mean Squared Error of the test set is: 21.08739695839598

The Mean Absolute Error of the training set is: 4.005054426498217  
The Mean Absolute Error of the test set is: 4.029939767512916

Root Mean Squared Error of the training set is: 4.574808272533999  
Root Mean Squared Error of the test set is: 4.5921015840675805

- Predicted value Vs Actual Value

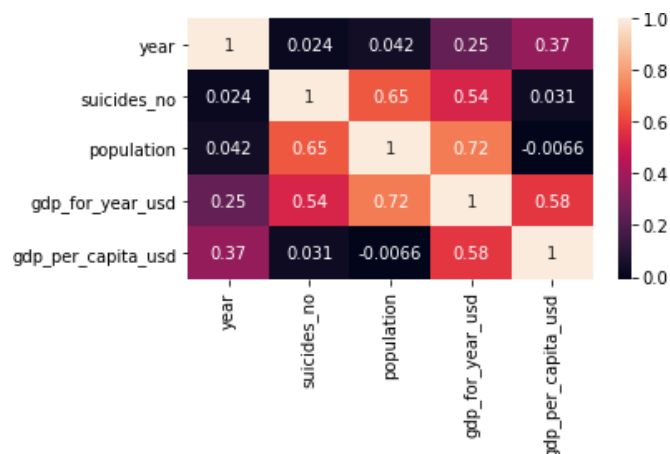
	Actual	Pred
16380	8	24.616322
3547	1	15.403714
16486	11	27.437608
10986	38	26.504378
6506	49	50.334504

## Decision Tree regression:

- Start by importing the data and the necessary packages;

```
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import pandas as pd
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
dataframe = pd.read_csv("master.csv")
```

- Non-Linear Relationship



- Preprocessing and splitting the train and test data

```
X = dataframe.drop(['suicides_no', 'year'], axis = 1)
y = dataframe['suicides_no'].copy()
categorical_features = ['age', 'sex', 'generation', 'country']

full_pipeline = ColumnTransformer([("cat", OrdinalEncoder(), categorical_features)], remainder="passthrough")

Xprep = full_pipeline.fit_transform(X)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(Xprep, y, test_size=0.3)
```

- Fit the model:

```
from sklearn.tree import DecisionTreeRegressor
tree_reg = DecisionTreeRegressor()
tree_reg.fit(X_train, y_train)

DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      presort=False, random_state=None, splitter='best')
```

- Prediction using test data and train data:

```
y_pred_testdt = tree_reg.predict(X_test)
y_pred_traindt = tree_reg.predict(X_train)
```

- Evaluate the performance of the algorithm:

```
The Mean Squared Error of the training set is: 0.0
The Mean Squared Error of the test set is: 11.960871611309823
```

```
The Mean Absolute Error of the training set is: 0.0
The Mean Absolute Error of the test set is: 2.492512528096921
```

```
Root Mean Squared Error of the training set is: 0.0
Root Mean Squared Error of the test set is: 3.458449307321104
```

Training error is 0 comparatively less than test error , this means the model is overfitting our data.

	Actual	Pred
8737	0	0.0
23728	93	93.0
21731	0	0.0
15284	67	67.0
24003	31	31.0

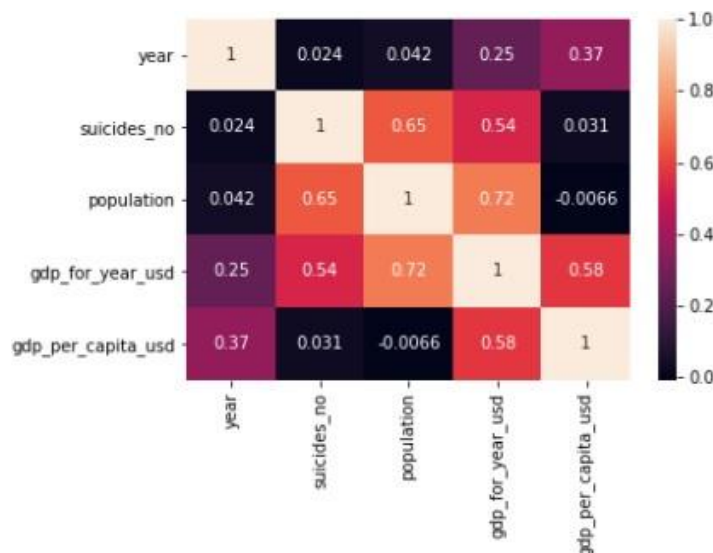
- Compare the Actual and predicted:

## Random forest

- Import the required libraries and the dataset;

```
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import pandas as pd
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
dataframe = pd.read_csv("master.csv")
```

- Non-Linear Relationship



- Preprocessing and splitting the train and test data

```
X = dataframe.drop(['suicides_no', 'year'], axis = 1)
y = dataframe['suicides_no'].copy()
categorical_features = ['age', 'sex', 'generation', 'country']

full_pipeline = ColumnTransformer([("cat", OrdinalEncoder(), categorical_features)], remainder="passthrough")

Xprep = full_pipeline.fit_transform(X)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(Xprep, y, test_size=0.3)
```

- Fit the model:

```
from sklearn.ensemble import RandomForestRegressor
forest_reg = RandomForestRegressor()
forest_reg.fit(X_train,y_train)

RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
                      oob_score=False, random_state=None, verbose=0, warm_start=False)
```

- Prediction using test data and train data:

```
y_pred_test = forest_reg.predict(X_test)
y_pred_train = forest_reg.predict(X_train)
```

- Evaluate the performance of the algorithm:

The Mean Squared Error of the training set is: 4.021627715133604  
 The Mean Squared Error of the test set is: 10.082646327289794

The Mean Absolute Error of the training set is: 1.4642456001803335  
 The Mean Absolute Error of the test set is: 2.3585596872852386

Root Mean Squared Error of the training set is: 2.0053996397560274  
 Root Mean Squared Error of the test set is: 3.175318303302803

- Compare the Actual with the predicted:

	Actual	Pred
8737	0	0.3
23728	93	88.9
21731	0	0.0
15284	67	63.5
24003	31	37.7

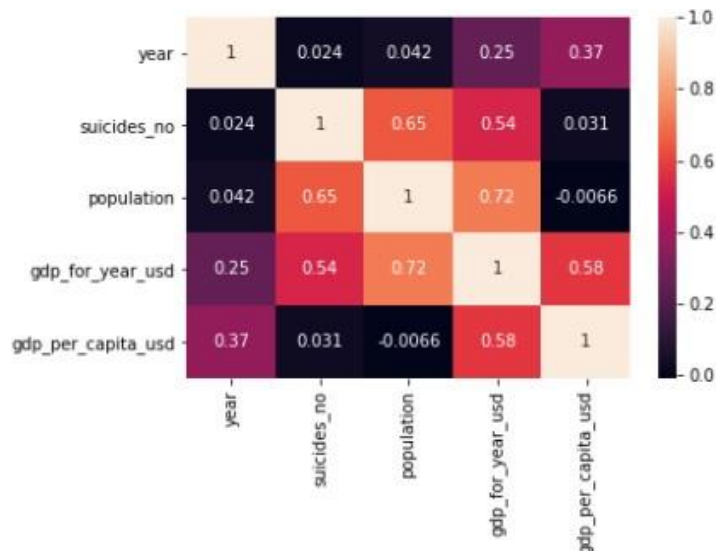
## 6.2 Ensembling

Ensemble methods combine several machine learning techniques into one predictive model in order to decrease variance (bagging), bias (boosting), or improve predictions (stacking). An AdaBoost regressor is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction. As such, subsequent regressors focus more on difficult cases.

### AdaBoost meta-estimator:

- Start by importing the data and the necessary packages;

```
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import pandas as pd
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
dataframe = pd.read_csv("master.csv")
```
- Non-Linear Relationship



- Preprocessing and splitting the train and test data



```

X = dataframe.drop(['suicides_no', 'year'], axis = 1)
y = dataframe['suicides_no'].copy()
categorical_features = ['age', 'sex', 'generation', 'country']

full_pipeline = ColumnTransformer([("cat", OrdinalEncoder(), categorical_features)], remainder="passthrough")

Xprep = full_pipeline.fit_transform(X)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(Xprep, y, test_size=0.3)

```

- Fitting the Decision Tree with Adaboost

```

from sklearn.ensemble import AdaBoostRegressor
adareg = AdaBoostRegressor(DecisionTreeRegressor())

AdaBoostRegressor(base_estimator=DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
max_leaf_nodes=None, min_impurity_decrease=0.0,
min_impurity_split=None, min_samples_leaf=1,
min_samples_split=2, min_weight_fraction_leaf=0.0,
presort=False, random_state=None, splitter='best'),
learning_rate=1.0, loss='linear', n_estimators=50,
random_state=None)

```

- Prediction and evaluation of the ensembling method.

```

y_pred_train = adareg.predict(X_train)
y_pred_test = adareg.predict(X_test)

```

The Mean Squared Error of the training set is: 0.6406382640033903  
The Mean Squared Error of the test set is: 8.91389863788256

The Mean Absolute Error of the training set is: 0.47678062611263616  
The Mean Absolute Error of the test set is: 2.2260763830321535

Root Mean Squared Error of the training set is: 0.8003988155934455  
Root Mean Squared Error of the test set is: 2.9856152863157974

- Compare actual and predicted values

	Actual	Pred
965	10	10.0
24108	9	8.0
11833	4	4.0
8077	1	1.0
3222	30	30.0



### 6.3 User interface

We have used flask Web Framework and Heroku for deployment of our project. Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. Heroku is a cloud platform as a service supporting several programming languages. Heroku is a platform as a service (PaaS) that enables developers to build, run, and operate applications entirely in the cloud.

**Prediction of Suicide case**

Country  
Choose Country

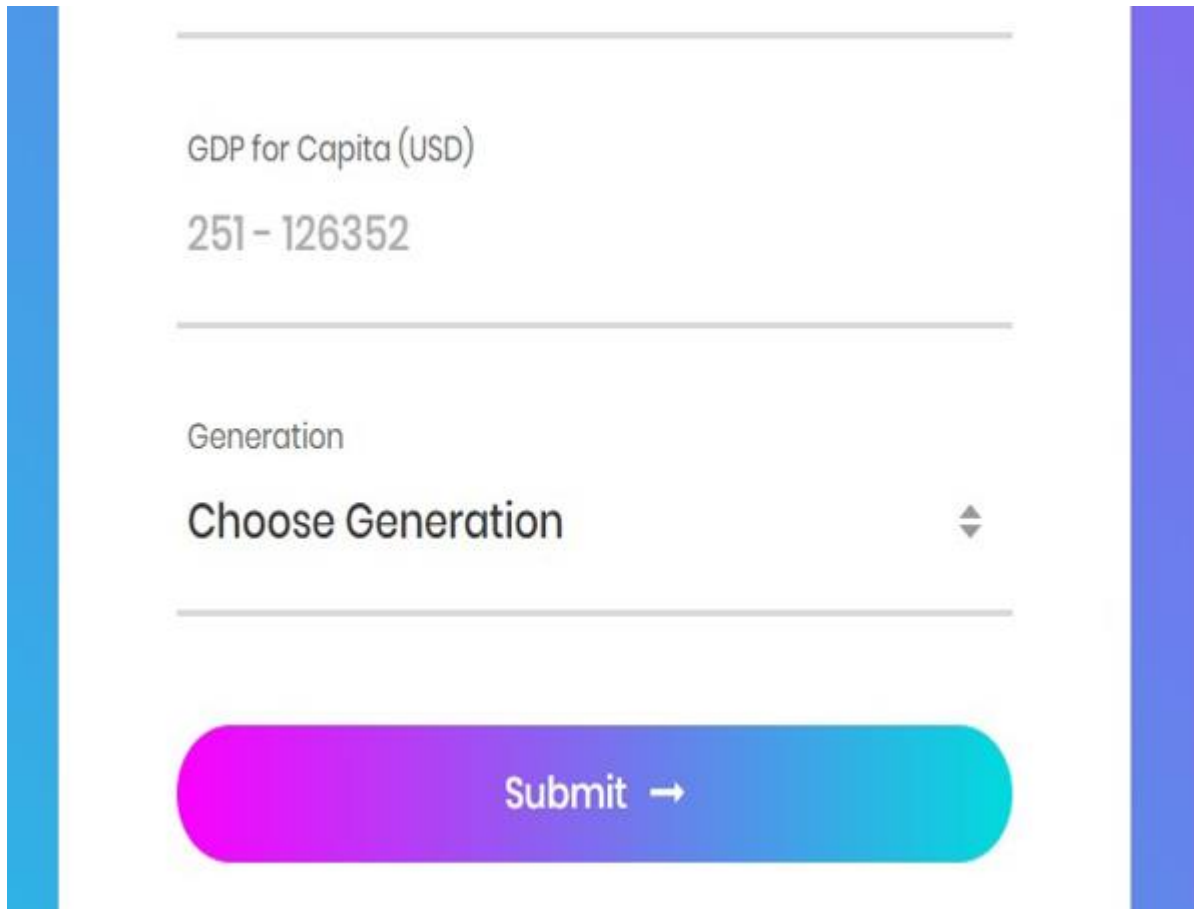
Gender  
Choose Gender

Age  
Choose Age

Population  
278 - 20273920

GDP for Year (USD)  
46919620 - 16155260000000

We fill in the required values, for categorical variables drop down menu is provided and for numerical we can type the values and the end of it, we can click on the submit option.



GDP for Capita (USD)

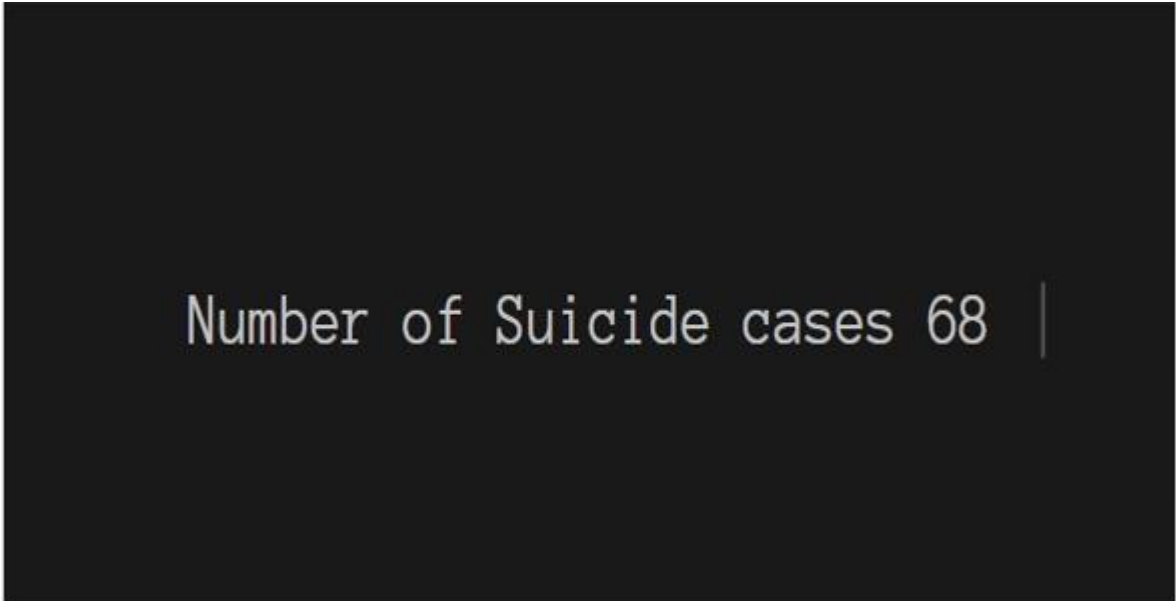
251 - 126352

Generation

Choose Generation

Submit →

Once the values have been submitted, we have supervised trained model stored as .pkl in the backend and the input values are fetched from index.html and passed into the trained model using python flask which then predicts the output and the output will be rendered in the new output page.



Number of Suicide cases 68 |

## **7 Conclusions / interpretation**

The suicide dataset was fit into 3 different supervised classifiers. Compared to all the three classification algorithms Random forest gives the best accuracy. But after introducing an ensemble method called has Adaboost, Decision tree performed well with least test errors. The results of this research demonstrate that the Random forest is a superior algorithm as the evaluation metrics such as mean squared error, mean absolute error and root mean squared error where all better for the random forest algorithm. Also Decision tree performs well boosting technique with more computation period.

## **8 References**

- [1] Studying Suicide Rates from 1985 to 2016 : Prediction using Machine Learning.
- [2] Kanwal Latif. Suicide Rates Overview 1985 to 2016.
- [3] D Syed, Sobia & Amin, Imran. (2017). Prediction of Suicide Causes in India using Machine Learning. Journal of Independent Studies and Research.



# **ST.ALOYSIUS COLLEGE (AUTONOMOUS)**

**MANGALURU**

**&**

**SCHOOL OF INFORMATION TECHNOLOGY - AIMIT**

***PRESENTS***

**INTERNATIONAL CONFERENCE ON ADVANCED IT,  
ENGINEERING AND MANAGEMENT  
[SACAIM 2019]**

---

## **Certificate of Presentation**

---

**This is to certify that**

**Mr. Charan Royston Lobo**

has attended the International Conference on Advanced IT, Engineering and Management  
[SACAIM 2019].and presented a paper titled

**A statistical Analysis on Road Accidents Using Machine Learning  
Algorithm**

organized by the IT Department of AIMIT, St Aloysius College (Autonomous), Mangaluru in association with Computer Society of India (CSI)-AIMIT Branch, Indian Society for Technical Education (ISTE).Iconic IT Professional Society (IPS). ICT Academy & Wadhwani Foundation International on 21 & 22nd November 2019 at AIMIT Campus Beerli, Mangalore - 575022.

---

Signature of Student