

# औद्योगिक प्रशिक्षण के लिए राष्ट्रीय संस्थान

## National Institute for Industrial Training

One Premier Organization with Non Profit Status | Registered Under Govt. of WB

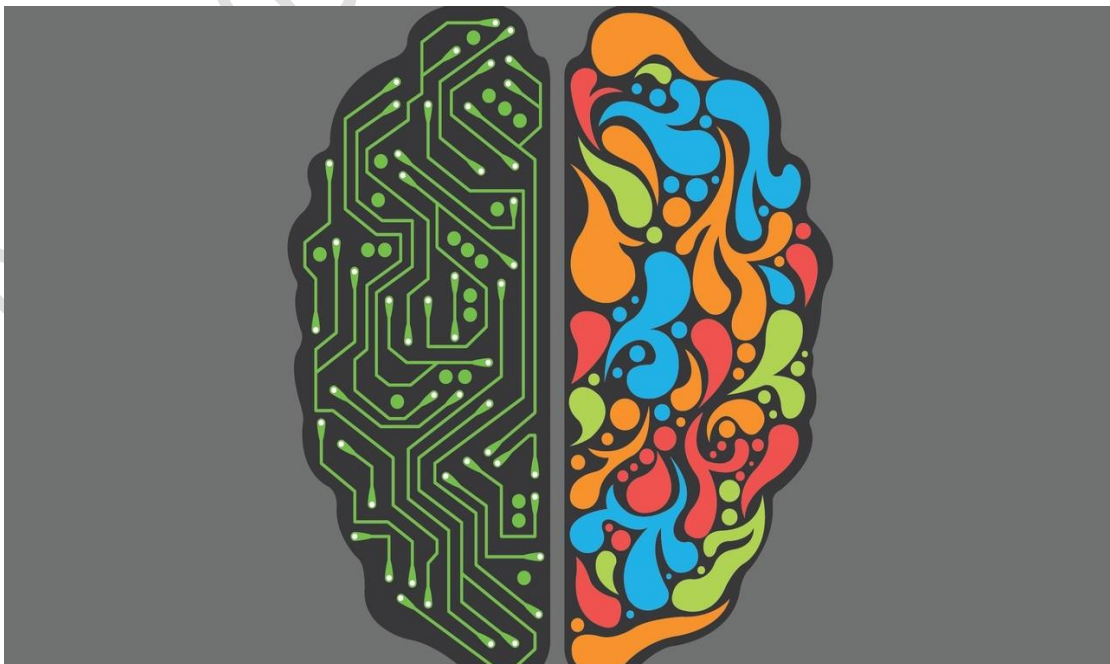
Empanelled Under Planning Commission Govt. of India

Inspired By: National Task Force on IT & SD Government of India

National Institute for Industrial Training- One Premier Organization with Non Profit Status Registered Under Govt. of West Bengal, Empanelled Under Planning Commission Govt. of India, Empanelled Under Central Social Welfare Board Govt. of India, Registered with National Career Services, Registered with National Employment Services.

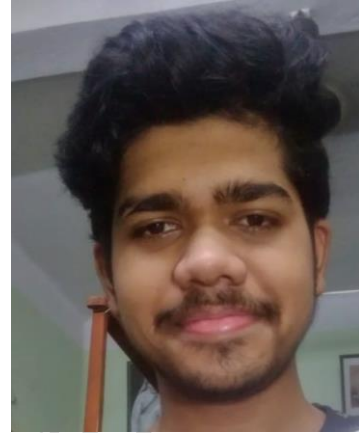


## Python With Artificial Intelligence



# ANALYSIS OF SUICIDE CASES IN DIFFERENT COUNTRIES

(Using Python with AI)



## **STUDENT'S PROFILE:**

Submitted by :- Subhrajyoti Manna

Stream: E.C.E

College: University Of Engineering & Management, Kolkata

Year : 2<sup>nd</sup> year 3<sup>rd</sup> semester

Email: [manna095subhra@gmail.com](mailto:manna095subhra@gmail.com)

LinkedIn: <https://www.linkedin.com/in/subhrajyoti-mann58a2541a6>

## **CONTENTS**

- 1.Acknowledgement
- 2.Introduction
- 3.Objectives
- 4.Hardware and software requirements
- 5.Snapshots
- 6.Coding
- 7.Advantages
- 8.Future Scope
- 9.Conclusion
10. Bibliography

# Acknowledgement

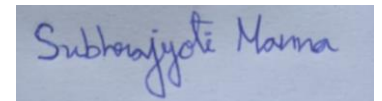
The success and final shape of this project required a lot of effort, guidance and support from many people. I feel privileged to have got this all along the completion of my Analysis of Suicides all over the country.

I take this opportunity to express my profound gratitude and deep regards to my faculty ( *Mr. Soumyadip Chowdhury*) for their exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by them time to time shall carry me a long way in the journey of life on which I am about to embark.

I am obliged to my other friends, teachers for their valuable information provided by them in their respective fields. I would like to express my gratitude towards them for their cooperation during the period of our project.

Last but not the least, I thank my friends who shared necessary information and useful web links for preparing our project. It would not be possible to complete the project without the support of our friends , teachers .

Finally I apologize all other unnamed who helped me in various ways for a well training.

A rectangular box containing a handwritten signature in blue ink. The signature reads "Subhajyoti Manna".

## Introduction

### *Motivations:*

Being extremely interested in everything having a relation with the Artificial Intelligence, the independent project was a great occasion to give me the time to learn and confirm my interest for this field. The fact that we can make estimations, predictions and give the ability for machines to learn by themselves is both powerful and limitless in term of application possibilities. We can use Artificial Intelligence in Finance, Medicine, almost everywhere. That's why I decided to conduct my project around the Artificial Intelligence.

### *Idea:*

Now a days, suicide becomes a serious social issue. In the era of machines and modern technologies though we become faster and faster, our minds become more emotionless. We all know that machines can give us speed but can not provide the happiness of heart, we are now connected virtual mediums but bonds between hearts become weaker. In the modern generation the mental health of the people is in a questionable position critically in youths. At the end of the day after sharing so many posts and spending times in virtual relationship when a person feels that he is after all alone the depression grasp his or her mind and yes, depression is the main reason of most of the suicides. It does not mean that depression only comes from technologies but also from the socio-economic conditions. We see the suicide rate is high in poor countries as we see the marginalisation of rich and poor increasing day by day, for example the farmer suicide in India. So considering all these issues I become interested on analysis of suicide statistics of many countries and make a conclusion on this. This the moto of my project and i want to work on it.

## Objective

- **Data mining:** In simple words, data mining is defined as a process used to extract usable data from a larger set of any raw data. It implies analyzing data patterns in large batches of data using one or more software.
- **Data cleaning and manipulating:** Data cleansing or data cleaning is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data.
- **Data preprocessing:** Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues. Data preprocessing prepares raw data for further processing.
- **Splitting dataset:** Data splitting is the act of partitioning available data into two portions; usually for cross-validatory purposes. One portion of the data is used to develop a predictive model and the other to evaluate the model's performance
- **Training the dataset with different models:** there are various machine learning models that can be tested. Various models such as linear regression, Logistic regression, linear discriminant analysis, Decision trees. Learning vector quantization, Support vector machines can be applied on the dataset and can be observed which models gives more precision.
- **Obtaining result and model evaluation:** Model Evaluation is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work in the future. Evaluating model performance with the data used for training is not acceptable in data science since it generates overoptimistic and over fitted models.

# **-:Hardware and software requirements:-**

## **Hardware Requirements:**

- Intel core i3 5<sup>th</sup> generation is used as a processor because it is faster and provide reliable and stable working environment clocked at 2.4GHz.
- A ram size of 4 gb is used as it will provide fast reading and writing capabilities.
- Minimum required Hard Disk space should be 256gb.

## **Software Requirements:**

- Python required version is 3.5 or 3.7.
- Anaconda platform .
- Operating system can be of windows /Linux/ Mac .

# Snapshots

## Importing the libraries:-

```
[1]: import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from math import *
from pylab import *
```

## Reading the CSV File Data:-

```
[2]: Suicide=pd.read_csv('suicide_ai.csv')
```

```
[3]: Suicide.head(10)
```

```
[3]:
```

	country	year	sex	age	suicides_no	population	suicides/100k pop	country-year	HDI for year	gdp_for_year (\$)	gdp_per_capita (\$)	generation
0	Albania	1987	male	15-24 years	21	312900	6.71	Albania1987	NaN	2,156,624,900	796	Generation X
1	Albania	1987	male	35-54 years	16	308000	5.19	Albania1987	NaN	2,156,624,900	796	Silent
2	Albania	1987	female	15-24 years	14	289700	4.83	Albania1987	NaN	2,156,624,900	796	Generation X
3	Albania	1987	male	75+ years	1	21800	4.59	Albania1987	NaN	2,156,624,900	796	G.I. Generation
4	Albania	1987	male	25-34 years	9	274300	3.28	Albania1987	NaN	2,156,624,900	796	Boomers
5	Albania	1987	female	75+ years	1	35600	2.81	Albania1987	NaN	2,156,624,900	796	G.I. Generation
6	Albania	1987	female	35-54 years	6	278800	2.15	Albania1987	NaN	2,156,624,900	796	Silent
7	Albania	1987	female	25-34 years	4	257200	1.56	Albania1987	NaN	2,156,624,900	796	Boomers
8	Albania	1987	male	55-74 years	1	137500	0.73	Albania1987	NaN	2,156,624,900	796	G.I. Generation
9	Albania	1987	female	5-14 years	0	311000	0.00	Albania1987	NaN	2,156,624,900	796	Generation X

## Filling up the NaN values:-

```
24]: print("Filling null values with mean of that particular column")
Suicide=Suicide.fillna(np.mean(Suicide))
```

Filling null values with mean of that particular column

## Taking out the Mean:-

```
[9]: print("Mean of Suicide:")
print(np.mean(Suicide))
```

```
Mean of Suicide:
year                2.001258e+03
suicides_no         2.425744e+02
population          1.844794e+06
suicides/100k pop   1.281610e+01
HDI for year        7.766011e-01
gdp_per_capita ($)  1.686646e+04
dtype: float64
```



```
[10]: print("Group by:")
      Suicide.groupby("age").size()
```

Group by:

```
[10]: age
      15-24 years    4642
      25-34 years    4642
      35-54 years    4642
      5-14 years     4610
      55-74 years    4642
      75+ years      4642
      dtype: int64
```

## Statistical information of the dataset:-

```
[11]: Suicide.describe()
```

```
[11]:
```

	year	suicides_no	population	suicides/100k pop	HDI for year	gdp_per_capita (\$)
<b>count</b>	27820.000000	27820.000000	2.782000e+04	27820.000000	8364.000000	27820.000000
<b>mean</b>	2001.258375	242.574407	1.844794e+06	12.816097	0.776601	16866.464414
<b>std</b>	8.469055	902.047917	3.911779e+06	18.961511	0.093367	18887.576472
<b>min</b>	1985.000000	0.000000	2.780000e+02	0.000000	0.483000	251.000000
<b>25%</b>	1995.000000	3.000000	9.749850e+04	0.920000	0.713000	3447.000000
<b>50%</b>	2002.000000	25.000000	4.301500e+05	5.990000	0.779000	9372.000000
<b>75%</b>	2008.000000	131.000000	1.486143e+06	16.620000	0.855000	24874.000000
<b>max</b>	2016.000000	22338.000000	4.380521e+07	224.970000	0.944000	126352.000000

```
[5]: Suicide.columns
```

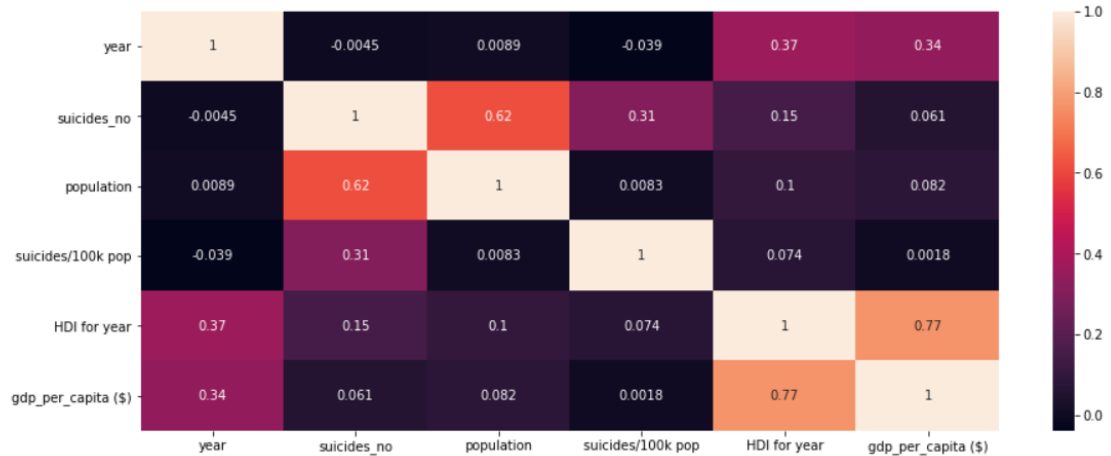
```
[5]: Index(['country', 'year', 'sex', 'age', 'suicides_no', 'population',
        'suicides/100k pop', 'country-year', 'HDI for year',
        'gdp_for_year ($)', 'gdp_per_capita ($)', 'generation'],
        dtype='object')
```

```
[6]: sns.pairplot(Suicide)
```

Now lets have a look of this statistical information in pictorial form(EDA)

```
[8]: plt.subplots(figsize=(15,6))
     sns.heatmap(Suicide.corr(),annot=True)
```

```
[8]: <matplotlib.axes._subplots.AxesSubplot at 0x183c6d83430>
```

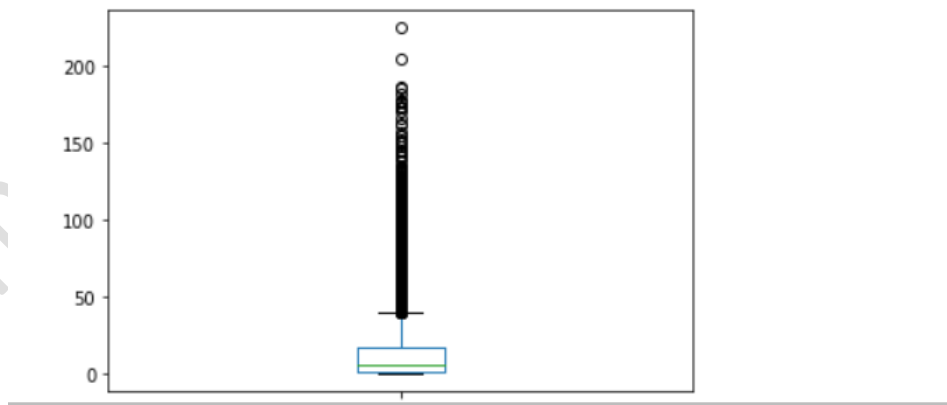


Checking out the Heatmap:-

Box plot of suicides/100k pop:-

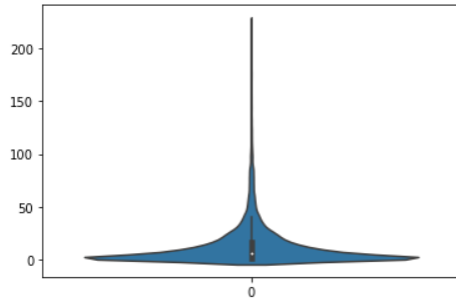
```
}]: Suicide['suicides/100k pop'].plot(kind='box',sharex=True,sharey=False)
```

```
}]: <matplotlib.axes._subplots.AxesSubplot at 0x183c71e4cd0>
```



Violin Plot of suicides/100K population:-

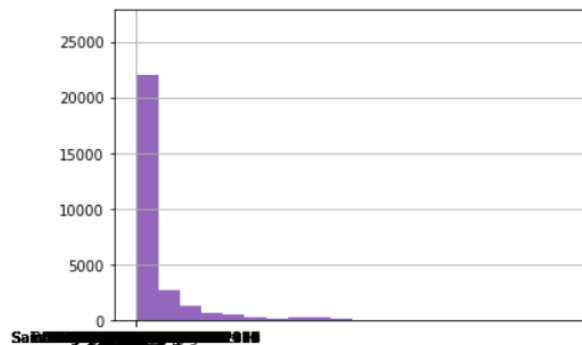
```
] sns.violinplot(data=Suicide['suicides/100k pop'])
]: <matplotlib.axes._subplots.AxesSubplot at 0x183c723cdf0>
```



**These Histograms showing the data from different attributes from the Dataset:-**

```
[15]: Suicide['country'].hist(bins=20)
      Suicide['year'].hist(bins=20)
      Suicide['age'].hist(bins=20)
      Suicide['suicides_no'].hist(bins=20)
      Suicide['population'].hist(bins=20)
      Suicide['suicides/100k pop'].hist(bins=20)
      Suicide['country-year'].hist(bins=20)
      Suicide['HDI for year'].hist(bins=20)
      Suicide['gdp_per_capita ($)'].hist(bins=20)
      Suicide['generation'].hist(bins=20)

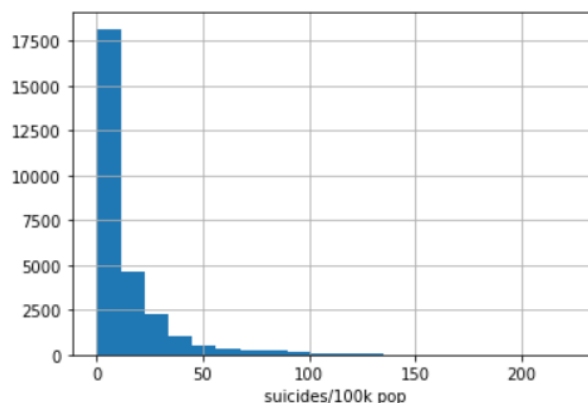
[15]: <matplotlib.axes._subplots.AxesSubplot at 0x183c78734f0>
```



**Histogram showing the suicides/100k population of all countries:-**

```
[16]: Suicide['suicides/100k pop'].hist(bins=20)
      plt.xlabel('suicides/100k pop')

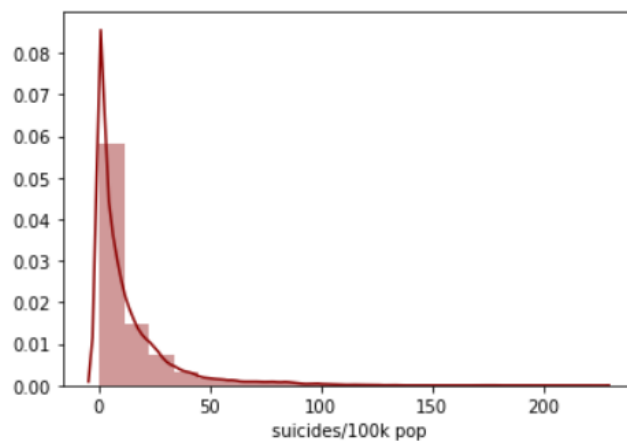
[16]: Text(0.5, 0, 'suicides/100k pop')
```



Displot for suicides/100k population and checking the probability density function of continuous variable country:-

```
[17]: sns.distplot(Suicide['suicides/100k pop'],kde=True,color='darkred',bins=20)
```

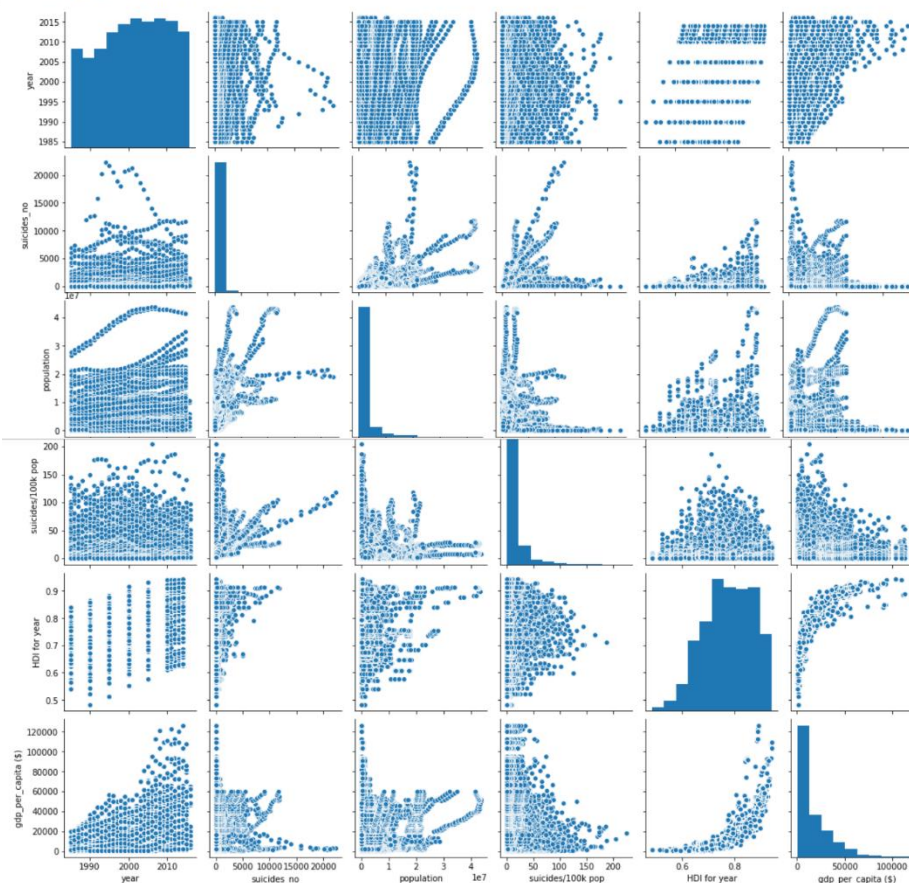
```
[17]: <matplotlib.axes._subplots.AxesSubplot at 0x183ca945580>
```



Pairplot the whole data:-

```
[6]: sns.pairplot(Suicide)
```

```
[6]: <seaborn.axisgrid.PairGrid at 0x183bf7bb940>
```



## Linear Regression Of The Model:-

Linear Regression Model

```
[115]: X= Suicide[['suicides_no', 'suicides/100k pop', 'HDI for year', 'gdp_per_capita ($)']  
y= Suicide['population']
```

```
[116]: from sklearn.model_selection import train_test_split
```

```
[117]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=101)
```

```
[118]: lm=LinearRegression()
```

```
[119]: lm.fit(X_train, y_train)
```

```
[119]: LinearRegression()
```

```
[120]: print(lm.intercept_)
```

1793213.2194713857

```
[121]: lm.coef_
```

```
[121]: array([ 2.95052506e+03, -4.13589285e+04, -3.59440064e+05,  9.42275095e+00])
```

```
[122]: coeff=pd.DataFrame(lm.coef_,X.columns, columns=['Coefficient'])  
print(coeff)
```

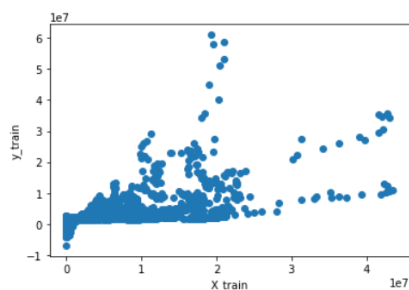
	Coefficient
suicides_no	2950.525057
suicides/100k pop	-41358.928511
HDI for year	-359440.064125
gdp_per_capita (\$)	9.422751

## Prediction of the Model:-

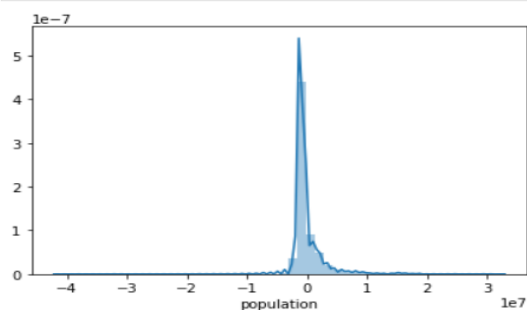
```
[123]: pred=lm.predict(X_test)  
print(pred)  
  
[1582675.43722182 2109766.43642379 1434821.99668113 ... 1440417.09921942  
666589.2154409 1238958.50379645]
```

```
[124]: plt.scatter(y_test,pred)  
plt.xlabel("X_train")  
plt.ylabel("y_train")
```

```
[124]: Text(0, 0.5, 'y_train')
```



```
[125]: sns.distplot((y_test-pred),bins=50);  
#sns.set_style('whitegrid')
```



## Checking Errors:-

```
[126]: from sklearn import metrics

[127]: metrics.mean_absolute_error(y_test,pred)

[127]: 1612163.0279865386

[128]: print(metrics.mean_absolute_error(y_test,pred))
print(metrics.mean_squared_error(y_test,pred))
print(np.sqrt(metrics.mean_squared_error(y_test,pred)))

1612163.0279865386
8641442758375.547
2939633.099278811

[129]: metrics.mean_squared_error(y_test,pred)

[129]: 8641442758375.547

[130]: np.sqrt(metrics.mean_squared_error(y_test,pred))

[130]: 2939633.099278811
```

## Clustering of an iris model:-

```
[198]: from sklearn.datasets import load_iris
iris=load_iris()
print(iris)
```

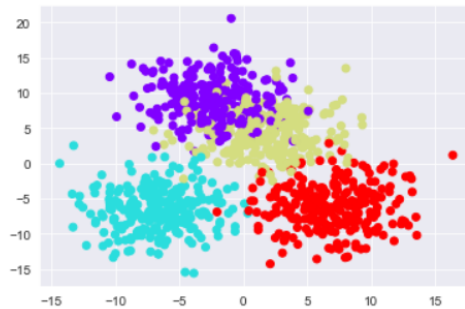
```
{'data': array([[5.1, 3.5, 1.4, 0.2],
                [4.9, 3. , 1.4, 0.2],
                [4.7, 3.2, 1.3, 0.2],
                [4.6, 3.1, 1.5, 0.2],
                [5. , 3.6, 1.4, 0.2],
                [5.4, 3.9, 1.7, 0.4],
                [4.6, 3.4, 1.4, 0.3],
                [5. , 3.4, 1.5, 0.2],
                [4.4, 2.9, 1.4, 0.2],
                [4.9, 3.1, 1.5, 0.1],
                [5.4, 3.7, 1.5, 0.2],
                [4.8, 3.4, 1.6, 0.2],
                [4.8, 3. , 1.4, 0.1],
                [4.3, 3. , 1.1, 0.1],
                [5.8, 4. , 1.2, 0.2],
                [5.7, 4.4, 1.5, 0.4],
                [5.4, 3.9, 1.3, 0.4],
                [5.1, 3.5, 1.4, 0.3],
                [5.7, 3.8, 1.7, 0.3],
                [5.1, 3.8, 1.5, 0.3],
                [5.4, 3.4, 1.7, 0.2],
                [5.1, 3.7, 1.5, 0.4],
                [4.6, 3.6, 1. , 0.2],
                [5.1, 3.3, 1.7, 0.5],
                [4.8, 3.4, 1.9, 0.2],
                [5. , 3. , 1.6, 0.2],
                [5. , 3.4, 1.6, 0.4],
                [5.2, 3.5, 1.5, 0.2],
                [5.2, 3.4, 1.4, 0.2],
                [4.7, 3.2, 1.6, 0.2],
                [4.8, 3.1, 1.6, 0.2],
                [5.4, 3.4, 1.5, 0.4],
                [5.2, 4.1, 1.5, 0.1]]),
```

```
[197]: z=iris.data
print(z)

[[ 0.03807591  0.05068012  0.06169621 ... -0.00259226  0.01990842
```

```
[212]: # c = data[1] - is from the centers, it tells which datapoint belong
# to which blob or cluster, let's replot the data
plt.scatter(data[0][:,0],
            data[0][:,1],
            c=data[1],cmap='rainbow')
# Set1, Set2, Set3, magma
```

```
[212]: <matplotlib.collections.PathCollection at 0x21b4c1c0d60>
```



```
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
```

```
[201]: print(iris.target.shape)

(150,)
```

```
[208]: data = make_blobs(n_samples=1000,
                        n_features=4,centers=4,cluster_std=3.0,random_state=42)
```

```
[209]: data[0], len(data[0])
```

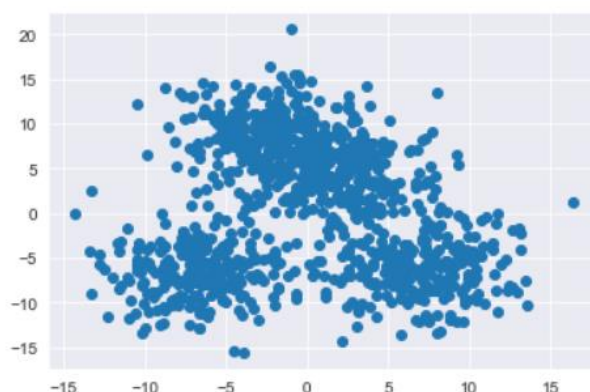
```
[209]: (array([[ -10.92100606,  -9.79495171,  -5.23708603,   5.35284008],
               [  -3.50728903,  -4.03753021, -11.15696076,   8.54467978],
               [  -3.86611658,   1.74264815,  -0.11182963,   4.25441365],
               ...,
               [  -4.6132855 ,   8.76175266, -13.86018157,   8.59824006],
               [   7.61509975,  -5.7864938 ,  -8.80254377,  -3.86207006],
               [  -4.12847666,   6.67937195,   5.2274146 ,  -0.96194865]]),
        1000)
```

```
[210]: data[0].shape
```

```
[210]: (1000, 4)
```

```
[211]: # x = data[0][:,0] - all the rows in the first col of data[0]
# y = data[0][:,1] - all the rows in the second col of data[0]
plt.scatter(data[0][:,0],
            data[0][:,1])
```

```
[211]: <matplotlib.collections.PathCollection at 0x21b4c178b50>
```



## Kmeans Clustering of Iris Dataset:-

```
[213]: from sklearn.cluster import KMeans
```

```
[214]: # Let's create an instance for KMeans
km = KMeans(n_clusters=7)
```

```
[215]: # Lets fit the model to our features (data[0])
km.fit(data[0])
```

```
[215]: KMeans(n_clusters=7)
```

```
[216]: #We can check the cluster centers
centers = km.cluster_centers_
centers
```

```
[216]: array([[ 6.72995612, -7.62606468, -4.89347984, -5.49720212],
 [ 2.39699548, 3.64305374, -10.55436187, 7.79466456],
 [ -6.98396894, -4.87601615, -10.17993597, 5.54908895],
 [ -2.50523614, 9.0640247, 4.70387791, 2.12149447],
 [ 7.11028104, -4.28460988, -8.77519234, -6.98546626],
 [ -6.55527155, -8.0040731, -7.55596498, 9.01778703],
 [ 1.23449976, 5.3728186, -7.90776175, 11.62536728]])
```

```
[217]: km.labels_
```

```
[217]: array([5, 2, 3, 3, 5, 3, 0, 5, 6, 6, 3, 3, 3, 0, 6, 4, 2, 3, 0, 3, 3, 5,
3, 3, 5, 1, 5, 1, 4, 1, 1, 0, 1, 0, 4, 6, 1, 3, 6, 1, 3, 4, 6, 0,
5, 2, 5, 4, 5, 1, 4, 1, 6, 6, 5, 0, 2, 5, 3, 1, 1, 1, 5, 3, 6, 1,
1, 2, 4, 2, 3, 3, 0, 1, 3, 3, 1, 0, 5, 6, 0, 2, 6, 3, 3, 6, 2,
2, 5, 5, 1, 4, 0, 3, 2, 3, 4, 2, 3, 4, 0, 2, 1, 3, 0, 3, 4, 5, 4,
5, 6, 4, 0, 2, 1, 5, 6, 1, 6, 1, 3, 5, 3, 5, 3, 6, 1, 3, 3, 0, 2,
4, 6, 2, 1, 5, 3, 1, 0, 0, 6, 2, 3, 1, 4, 2, 2, 3, 4, 0, 6, 4, 0,
3, 6, 0, 3, 4, 1, 1, 3, 5, 6, 0, 5, 3, 1, 1, 5, 1, 5, 3, 6, 1, 2,
0, 6, 3, 1, 0, 6, 4, 3, 1, 1, 2, 3, 3, 3, 6, 5, 2, 1, 3, 5, 3,
4, 6, 3, 4, 4, 1, 2, 0, 5, 2, 6, 4, 3, 1, 4, 0, 6, 0, 2, 4, 3, 3])
```



```
[218]: # c = data[1] - is from the centers, it tells which datapoint belong
# to which blob or cluster, let's replot the data

f, (ax1, ax2) = plt.subplots(nrows=1,
                             ncols=2,
                             sharey=True,
                             figsize=(10,6))

# For the fitted one, c = kmeans.labels_
#ax1.set_title('K Means (K = 8)')
# x = data[0][:,0] - all the rows in the first col of data[0]
# y = data[0][:,1] - all the rows in the second col of data[0]

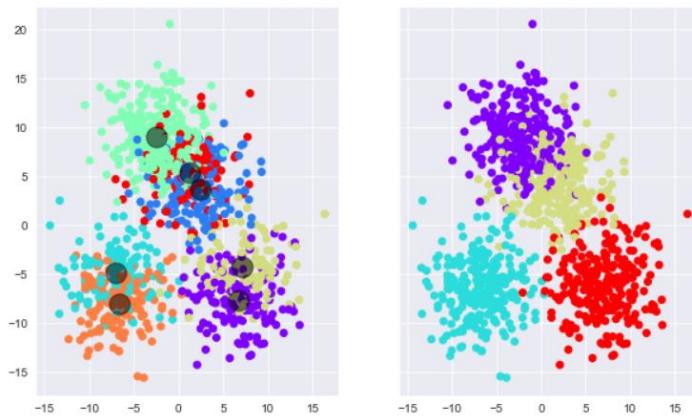
ax1.scatter(data[0][:,0],
            data[0][:,1],
            c=km.labels_,
            cmap='rainbow')

# For the original one, c = data[1]
#ax2.set_title("Original")

ax2.scatter(data[0][:,0],
            data[0][:,1],
            c=data[1],
            cmap='rainbow')

#sns.set_style('darkgrid') for grid purpose
# Let's put the cluster centers on the plot as well
#centers=kmeans.cluster_centers_
#centers
#FOR CENTROID PLOTTING OR CENTRE VALUES OF THE CLUSTERS

ax1.scatter(x=centers[:, 0],
            y=centers[:, 1],
            c='black',
            s=300,
            alpha=0.5);
```



# **-: Coding :-**

## **#Importing The Essential Libraries for Analyzing the Data**

```
import numpy as np
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

```
sns.set_style("whitegrid")
```

## **#Reading the csv file data this involves pandas dataframing**

```
Suicide = pd.read_csv("suicide_ai.csv")
```

```
Suicide = suicide.head(10)
```

## **#Now check out the NaN values**

```
print("Filling null values with mean of that  
particullar column")
```

```
Suicide=Suicide.fillna(np.mean(Suicide))
```

## **#Taking out the mean**

```
print("Mean of Suicide:")
```

```
print(np.mean(Suicide))
```

#Gathering some information about the data using info

```
print("Group by:")
```

```
Suicide.groupby("age").size()
```

**Statistical information of the dataset**

```
Suicide.describe()
```

```
Suicide.columns
```

```
Suicide.head(10)
```

Now lets have a look of this statistical information in pictorial form(EDA)

**#Checking out the Heatmap**

```
plt.subplots(figsize=(15,6))
```

```
sns.heatmap(Suicide.corr(),annot=True)
```

**#Checking out the Pairplot**

```
sns.pairplot(Suicide)
```

**Scatter plot of suicides/100k and age attributes**

```
sns.scatterplot(x="age",y="suicides/10k
```

```
pop", data=Suicide)
```

**Box Plot of suicides/100k population :-**

```
Suicide['suicides/100k
```

```
pop'].plot(kind='box',sharex=True,sharey=Fa
```

lse)

**Violin Plot of suicides of 100k/population :-**

```
sns.violinplot(data=Suicide['suicides/100k  
pop'])
```

**These Histograms showing the data  
from attributes of different  
information in the Dataset:-**

```
Suicide['country'].hist(bins=20)
```

```
Suicide['year'].hist(bins=20)
```

```
Suicide['age'].hist(bins=20)
```

```
Suicide['suicides_no'].hist(bins=20)
```

```
Suicide['population'].hist(bins=20)
```

```
Suicide['suicides/100k pop'].hist(bins=20)
```

```
Suicide['country-year'].hist(bins=20)
```

```
Suicide['HDI for year'].hist(bins=20)
```

```
Suicide['gdp_per_capita ($)'].hist(bins=20)
```

```
Suicide['generation'].hist(bins=20)
```

**Histogram showing suicides/100k population of the all Countries:**

```
Suicide['suicides/100k pop'].hist(bins=20)
```

```
plt.xlabel('suicides/100k pop')
```

### **Distplot for suicides/100k population and checking the probability density function of continuous variable**

```
sns.distplot(Suicide['suicides/100k pop'],kde=True,color='darkred',bins=20)
```

### **Now for the training of the dataset:-**

```
X= Suicide[['suicides_no','suicides/100k pop', 'HDI for year', 'gdp_per_capita ($)']]
```

```
y= Suicide['population']
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=101)
```

```
lm.fit(X_train, y_train)
```

### **Linear Regression of this model:-**

```
from sklearn.linear_model import LinearRegression
```

```
lm=LinearRegression()
```

```
print(lm.intercept_)
```

```
lm.coef_
```

```
coeff=pd.DataFrame(lm.coef_,X.columns, columns=['Coefficient'])
```

```
print(coeff)
```

### **Prediction of this model:-**

```
pred=lm.predict(X_test)
```

```
print(pred)
```

```
plt.scatter(y_test,pred)
```

```
plt.xlabel("X_train")
plt.ylabel("y_train")
sns.distplot((y_test-pred),bins=50);
```

Logistic Regression of this model to calculate error using import of matrices :-

```
from sklearn import metrics
metrics.mean_absolute_error(y_test,pred)
print(metrics.mean_absolute_error(y_test,pred))
print(metrics.mean_squared_error(y_test,pred))
print(np.sqrt(metrics.mean_squared_error(y_test,pred)))
metrics.mean_squared_error(y_test,pred)
np.sqrt(metrics.mean_squared_error(y_test,pred))
```

### **:-Unsupervised Learning using an iris Dataset :-**

Import of iris dataset:-

```
from sklearn.datasets import load_iris
```

```
iris=load_iris()
print(iris)
```

### Managing the data :-

```
z=iris.data
print(z)
print(iris.feature_names)
print(iris.target.shape)
data = make_blobs(n_samples=1000,
                  n_features=4,centers=4,cluster_std=3.0,random_state=42)
data[0], len(data[0])
data[0].shape
```

### Scattering of the data:-

```
plt.scatter(data[0][:,0],
            data[0][:,1])
```

### Scattering of Data in rainbow:-

```
plt.scatter(data[0][:,0],
            data[0][:,1],
            c=data[1],cmap='rainbow')
```

### KMeans clustering of iris Dataset:-

```
from sklearn.cluster import KMeans
```

```
km = KMeans(n_clusters=7)
km.fit(data[0])
centers = km.cluster_centers_
centers
km.labels_
```

clustering with centers:-

```
f, (ax1, ax2) = plt.subplots(nrows=1,
ncols=2,
sharey=True,
figsize=(10,6))
```

```
ax1.scatter(data[0][:,0],
data[0][:,1],
c=km.labels_,
cmap='rainbow')
```

```
ax2.scatter(data[0][:,0],
data[0][:,1],
c=data[1],
cmap='rainbow')
```

```
ax1.scatter(x=centers[:, 0],
y=centers[:, 1],
c='black',
s=300,
alpha=0.5);
```



## -:Advantages:-

### Python preferred over machine learning tools

Python becomes Pythonic when the code is written naturally. It has many other features that attract the machine learning community. Being a data science tool, Python helps to explore the concepts of machine learning in the best way possible. Machine Learning is all about probability, mathematical optimization, and statistics, which are all made easy by Python.

### Easy to learn

What drives developers to Python is that it is easy to learn and code. It promotes an easy-to-understand syntax especially when compared to other languages, such as R and thereby leads to a shorter learning curve.

Today, Jupyter is a tool used for writing code and text within a web page's context. It helps data scientists and engineers work in a collaborative manner. The code works on a server and you get results in HTML and integrated into your writing page. If you want to run it, you need to follow three simple steps.

0. install python3
1. install pip3
2. install Jupyter
3. generate the config: jupyter notebook --generate-config

### Extremely Scalable

Python has emerged as a scalable language compared to R and is faster to use than Matlab and Stata. Even YouTube has migrated to Python due to its scalability that lies in its flexibility during problem-solving situations. Skilled data scientists in various industries use this language to develop various types of applications successfully.

### Libraries:-

Many libraries are available to perform data analysis, here's an important one to

start with:

**NumPy** is important to perform scientific computing with Python. It encompasses an assortment of high-level mathematical functions to operate on multi-dimensional arrays and matrices.

**Pandas**, also developed on top of NumPy, delivers data structures and operations to change numerical tables and time series.

**Matplotlib** is a 2D plotting library. It offers data visualizations in the form of histograms, power spectra, bar charts, and scatterplots with minimal coding lines.

Developed on NumPy and Matplotlib acts as a machine learning & artificial intelligence library that leads to classification, regression, and clustering algorithms that involve support vector machines, logistic regression, gradient boosting.

## Python community

The growth of Python is due to its ecosystem. Today, an increased number of volunteers are developing Python libraries as Python has extended its reach to the data science community. This helps develop advanced tools and processes in Python.

The community helps Python aspirants look for relevant solutions to their coding problems. Moreover, code mentor and stack flow are available to find the right answers to questions.

## Graphics

Python offers many visualization options. Matplotlib is the base for the development of libraries like Seaborn, pandas plotting. Developers can understand data, develop charts, graphical plot and develop web-ready plots with the help of data visualization packages.

## -:Future Scope:-

### *For Python*

**The future is bright.** Artificial intelligence & Machine learning have been around for a long time. It just wasn't explicitly called that. Fields such as **statistical data analysis, data warehousing and high performance computing** have been in existence for a long time.

We will need only one, who could do the above end-to-end.

Most of the Data analyst, Machine Learning and NLP companies in india using Python because of the following reasons:

- **Python is easy to learn:** Python's main advantage is that anyone can learn it quickly and easily. The language was designed to be **simple**.
- **Visualization / Graphics:** Python is not as good as R (yet), but we'll see more and more cool APIs (e.g., [Plotly](#)) and Data Visualization libraries that make the partial advantage of R insignificant compared to Python. You can do really cool stuff with Python..
- **PyBrain** ([PyBrain](#)) and **Tensorflow** for some neural network.
- **iPython** notebook for interactive programming as in R.

**Large community = Documentation = Brainpower:** with Python, you can find a large (and still growing!) active community. At the end of the day, if you get lost you can rely on a this large community of experts to help you find a proper solution for coding problems (even niche ones) and answers to questions related to the topic.

### *For Analysis of Suicides:-*

Analysis of suicide is extremely complex and challenging. It has very large opportunities and social implications. Advances in artificial intelligence techniques are occurring all the time. One of the most promising, which is finding greater and greater commercial use is Deep Learning. Deep Learning differs from Artificial intelligence in that neural network architectures and algorithms used are of a more complex nature. Deep Learning could be applied to this suicidal analysis problem, one possible future directions could involve the use of Recurrent Neural Networks. Recurrent Neural Networks have the ability to be able to model changes across time. Coupled with the fact that Recurrent Neural Networks have the ability to include a regression output layer which would give a hard number as an output predication . The work carried out in this thesis paves the way to train a future Recurrent Neural Network as the initial tasks feature engineering has already been undertaken.

## -:Conclusion:-

### For python

python is easy, simple, powerful, and innovative due to its broader usage in a variety of contexts, some of which are not associated with artificial intelligence. R is an optimized environment for data analysis, but it is difficult to learn.

It's only one way to shape the debate: considering it as a zero-sum game. The fact is that understanding both tools and utilizing them as per their respective strengths can refine you as a data scientist. Every data scientist should be versatile and should stay at the top of their game.

### For Suicidal Rate Analysis :-

- Predicting suicide rates depending on some list of datas and some calculations of algebraic functions appears to be difficult
- Neural Networks gives a modest improvement over linear regression results .
- The developed system the Suicides Data has been predicted using feature column those can be given like country, year, suicides no, population, suicides/100k population, country-year, gdp for year, gdp per capita etc. Though the rainfall also depends & effects on some extra things. But for future prediction those data will be unknown. This is a big problem for future data prediction.

## **-:Bibliography:-**

- **Artificial Intelligence: A Modern Approach** by Stuart Russell and Peter Norvig
- Introduction to **Artificial Intelligence** by Philip C Jackson.
- The Emotion Machine: Commonsense Thinking, **Artificial Intelligence**, and the Future of the Human Mind by Marvin Minsky.
- **Suicide** ([French](#): *Le suicide*) written by French sociologist **Émile Durkheim**.

manna095subhira@gmail.com

