Suicide Rates Data Analysis

Gedion Alemayehu

COSC 315: Data Analytics

Dr. Dan Steinwand

December 4, 2020

1. Introduction

If for instance, a country wanted to lower their suicide rates it would have to take into consideration some variables that contribute to the rates and perhaps analyze other countries in the world that face the same problem and analyze some commonalities and differences to help them approach the problem. In this process, it would imperative to set a goal and finding the variables with high correlation to predict the rate in the future and compare it with their solutions. My project is to help find the best correlation between a variable and the suicide rate to help better predict the future rates, simultaneously identifying the most conducive problem contributing to suicide, consequently narrowing a possible target for a solution.

After exploring multiple data-set sources, I ended up using Kaggle as my data source. The data set that I chose is the Suicide Rates Data Set. There are multiple reasons why I chose this data set. Evidently, suicide is a pertinent and perennial problem on a global basis. Especially in the western world. The contents of this data set are pretty bulky and detailed. The data set contains the recorded suicide number in 101 countries from 1985 to 2016. It presents us with multiple variables as columns to add more detail about the recorded suicides; accordingly, it entails 12 factors: the country where it was recorded, the population of each country, the gross domestic product of each country, the human development index of each country, the age group of each record, the generation to which each record belongs to, and the sex for each record. It presents alternative representations for some of these factors for further clarification, for instance, it comprises suicide number per 100 thousand in addition to the total population to represent each country in a fair manner by countering their differences in the total population.

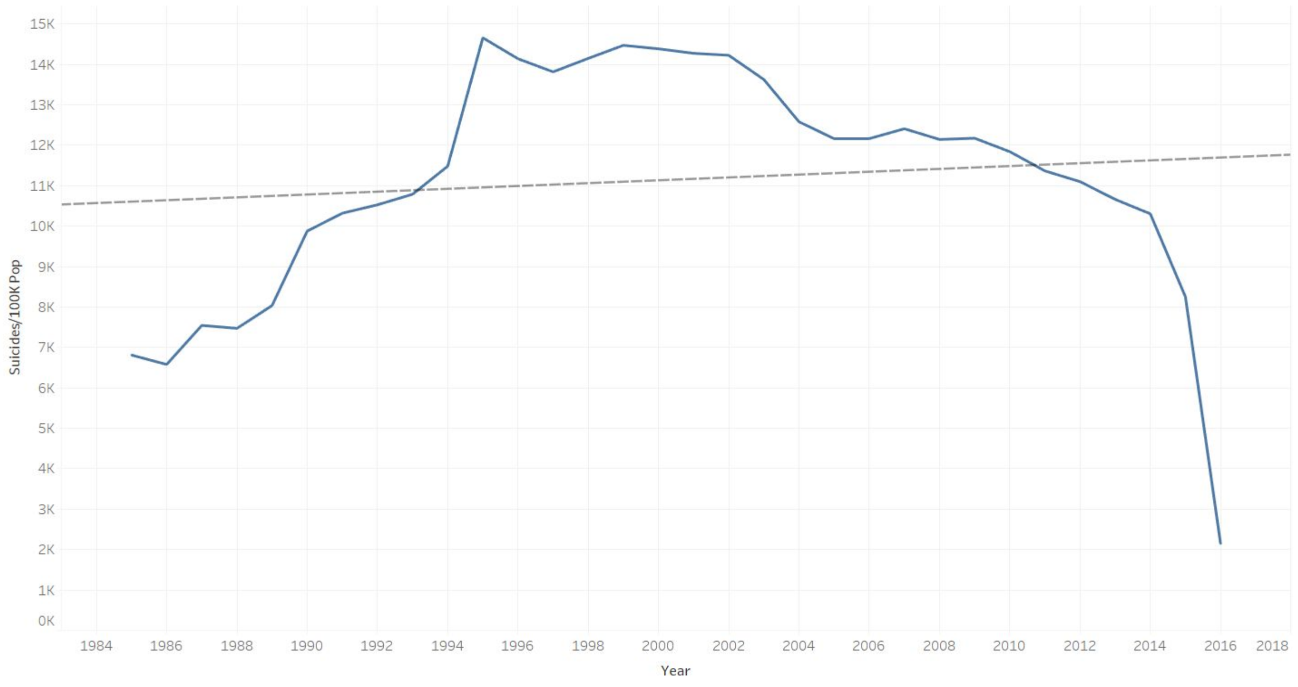
2. Data Exploration

Exploration with visualization of the data is a crucial step as it makes it easier to comprehend the data and detect any patterns, trends, and/or outliers. The majority of the data used in this data set was obtained from raw recordings in the World Health Organisation database. Data Exploration also plays a huge role in the later sections of modeling and predicting as it dictates the instinctive selections of variables to compare when attempting to predict an outcome. The first step was research on some of the variables. For example, the data set contained a column named "HDI" which, assumingly, not everyone would be aware of prior to this project; thus, some more research was done to discover what it meant and what possible contributions it might have in predicting suicide. HDI, the human development index, is a standard that evaluates the development of an individual by using average annual income and educational experiences. Accordingly, it can be speculated GDP and HDI go hand to hand in predicting suicide. The next step was to dive into visual and statistical comparisons and correlations of multiple variables to further explore and understand the data set. As a result, both Python and Tableau were utilized in this project to visualize

Most of the exploration of this project was to depict and understand the relationship between variables and how they collectively affect the number of suicide. The following section will describe and portray some of the key insights discovered in the exploration stage of the data set.

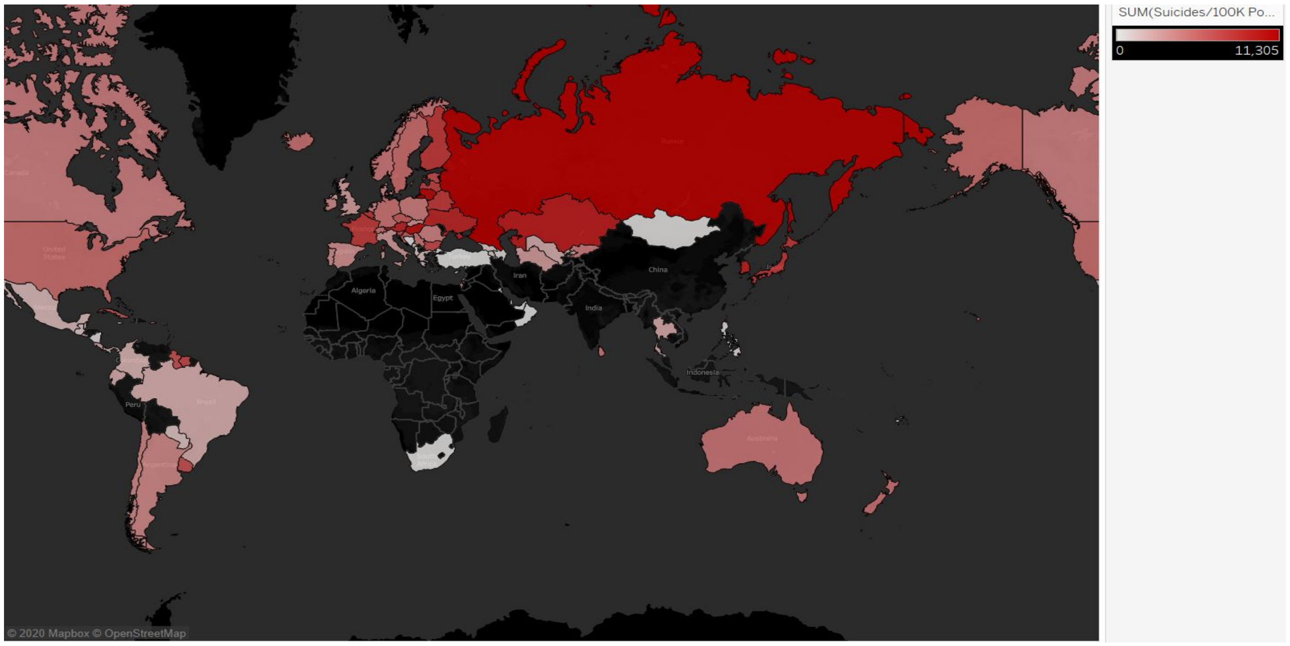
2.1 By Year

The evolution of suicide numbers, worldwide, by year is depicted below. It includes the data from 1985 to 2016. Some key takeaways from the curve are the surge in the late 90s and the dramatic decline from 2014-2016. The peak suicide rate per 100 thousand was in 1995 which was around fifteen thousand deaths. Subsequently, it was decreased by over 25 percent in the next decade (till 2015) to about eleven thousand deaths.



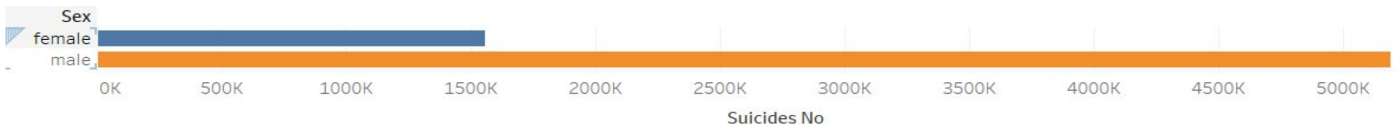
The trend line is still tilted upwards as the surge in the 90s significantly outnumbers both the initial rates and the declined rates in the 2000s. However, it is significant to note that recordings were highly limited in the 1980s, and so there is a possibility that the set was not truly representative of the true data during that period.

2.2 By Country

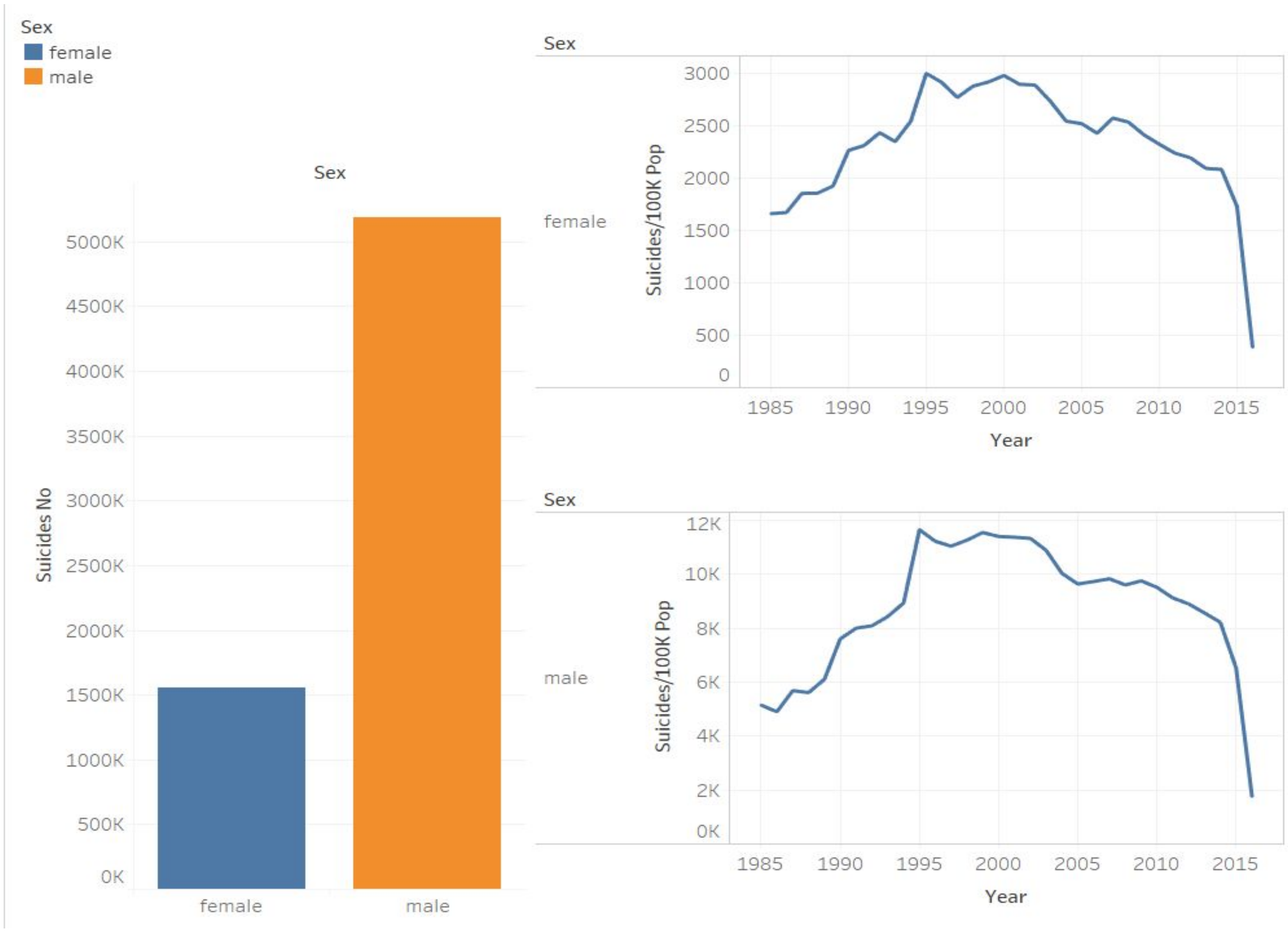


As this dataset is geographical by nature, the above graph shows where in the world map suicide is more pertinent than others. It can be inferred from the graph that the Russian Federation leads the world by the number of suicides. After a deeper look, it is important to notice that some countries are not included in the list of this data set. Out of about 195 countries that exist, only 101 of them are listed in the data set, some of which, like China and India, are very populous. Consequently, the predictions and models will only apply to portray the listed countries' trends and not the entire population of the world.

2.3 By Gender

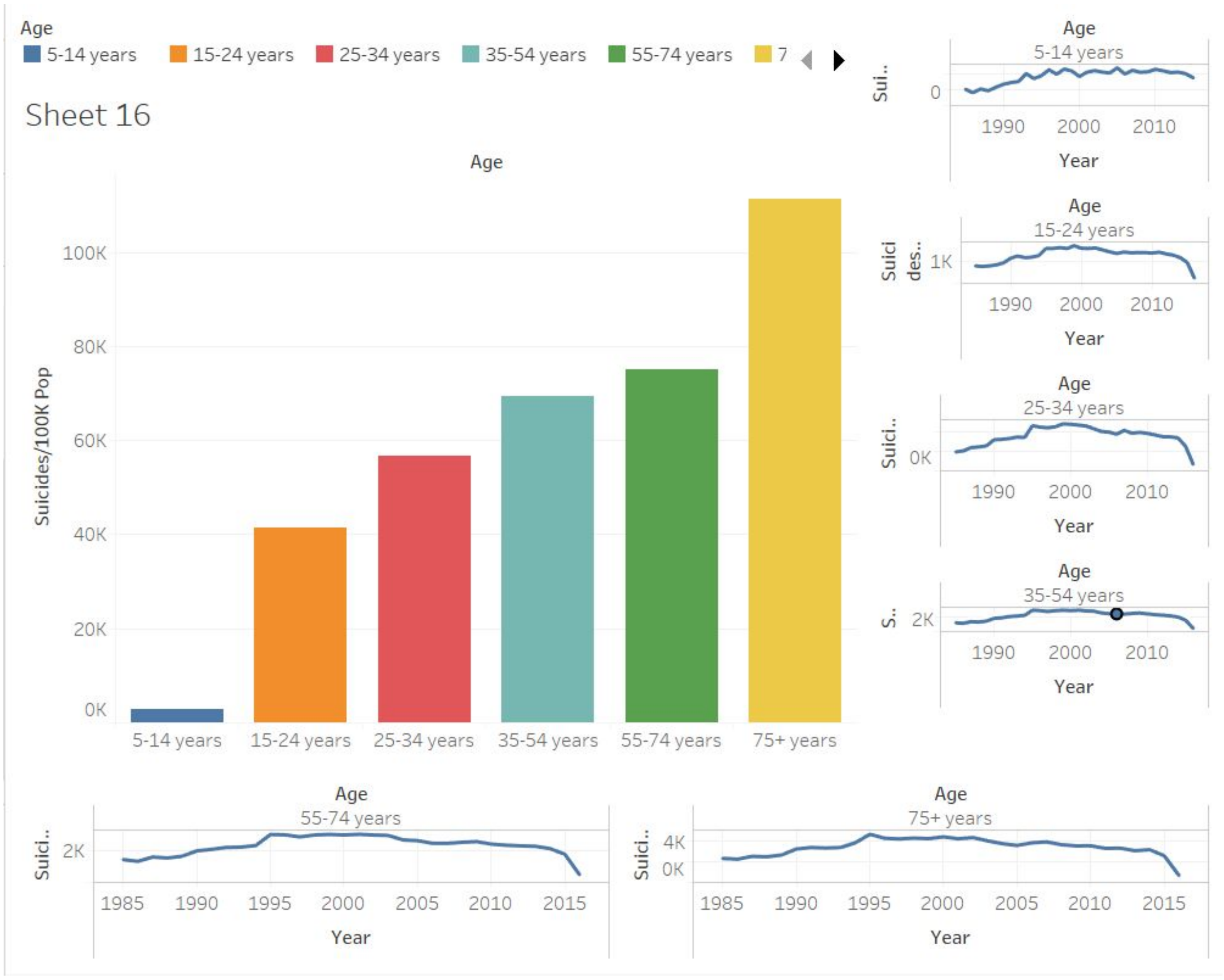


The graph above depicts the number of suicide per 100 thousand based on gender, and the graph below also includes the different trends male and female exhibit from 1985 to 2016. It is hard to miss that, globally, the suicide rate of men is significantly higher than the suicide rate of women; about 3 and a half times **(3.5x).** The trend by year shows that both men and women suicides peaked in 1995 and that perhaps is an interesting fact to look deeper into.



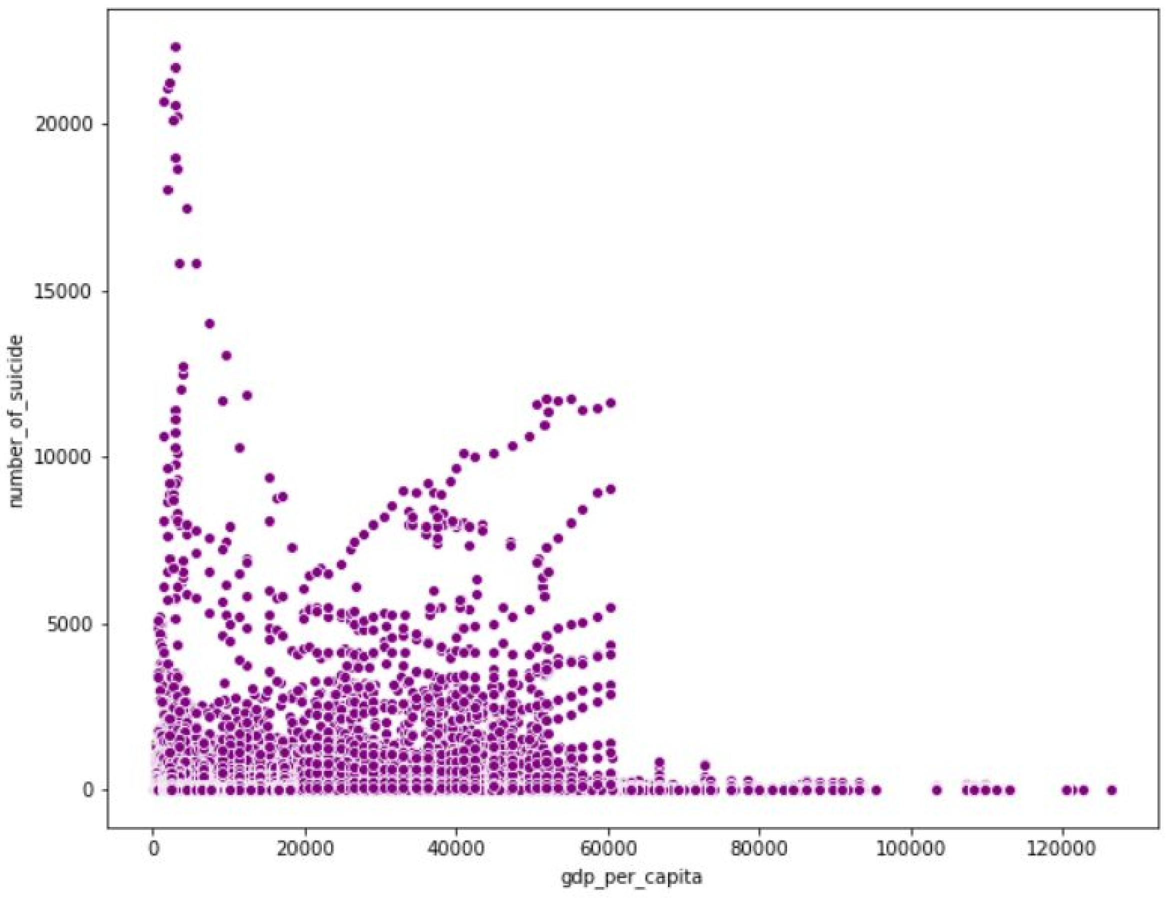
2.4 By Age

The graph below shows the relationship between age suicide numbers per 100 thousand. It also shows the evolution of suicide numbers in the age groups from 1985 to 2016. As per the graph, we can see that, globally, the suicide rate increases in a directly proportional manner with age; older people are more likely to commit suicide. Another key insight is the number of suicide for the age group 75+, it has decreased significantly after 1990. The reason might be longevity which does not extend, on average, so much after 75.



2.5 By Capital

It is an interesting question to answer whether more money brings fewer suicides. In our data set the close measurements to capital/money are GDP and HDI. The graph below shows the relationship between GDP and HDI with the number of suicides per 100 thousand. From the graph below, we can infer that GDP alone can not be an easily determining factor to predict suicide number per 100 thousand as it shows a very inconsistent trend and many outliers.



3. Data Cleaning

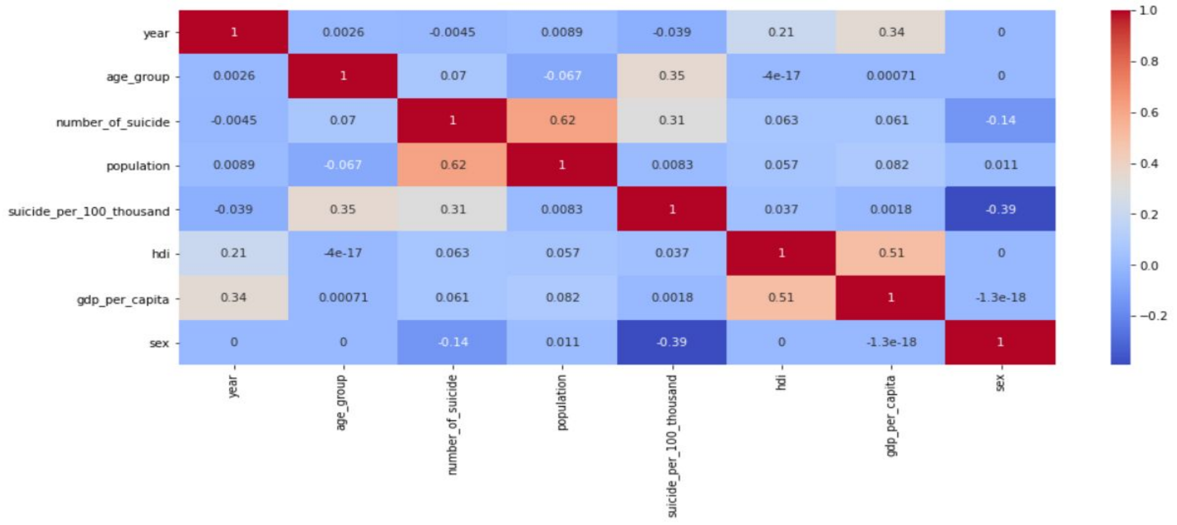
Data cleaning for the set was carried after the completion of adequate exploration and comprehension of the data. It is important to have column and row names that are easily understandable; hence, the first data cleaning that was performed was column name changes to some of the ambiguous names. The name changes were the following: ‘suicides_no’ to ‘number_of_suicides’, ‘suicides/100k’ to ‘suicide_per_100_thousand’, ‘HDI for year’ to ‘hdi’, ‘gdp_for_year ($)’ to ‘gdp’, and ‘gdp_per_capita ($)’ to ‘gdp_per_capita.’

The data set contained numerous NaN values that made it difficult to fully understand the data. Therefore, the filling of those values was performed as a cleaning mechanism. HDI had the most NaN values, however, after some research, it was clear that HDI remains constant every year and only changes incrementally, and so the missing values were filled by the mean of all the existing values. Furthermore, the data contained a column, 'country_year,' that concatenated the country and year as one string value. The importance of this column was rather confusing, and moreover not useful for the intended application of this project, so more cleaning was done by deleting the column.

More data cleaning was done to ensure accessibility and concise recordings for later utilization. The recordings for the groups of ages were listed in the format, 'xx-yy years', x and y being whole numbers. This format was inaccessible to use in the model, accordingly multiple if statements along with the *replace* function in Pandas was used to change the group numbers into a single mean value for concision and easy access. All newly changes mean values, then, were converted to float types. Additionally, gender in this data set was listed as male and female, however, to make it easily accessible a new column was created comprising of male values represented as a 0 and female values represented as 1. In a nutshell, it was evident that data cleaning was one of the most important steps in any analysis as real, raw data is more often than not is messy.

4. Modeling

The data was partitioned into the testing and training data in a 3:7 proportion. The test size was thirty percent of the data and the rest of the seventy being the training data. However, it was not the entire data that was partitioned. Based on the exploration and correlation coefficient analysis that was done, population, GDP per capita, year, and sex were determined as our X values, i.e., independent variables or predictors. As the model attempts to predict the suicide number, the dependent variable was suicide per 100 thousand. To identify the above mentioned predicting variables, a correlation coefficient heat map was used.



For modeling, Initially, a multiple regression was utilized to investigate whether the above listed independent variables can be used to predict the number of suicides. The model used was an ordinary least squares (OLS) regression model. This resulted in an R-squared value of about 0.416, hence accuracy of 41.6% in our training set and 44.3%. In our testing set. These numbers were obviously unsatisfactory and so better forms of regression were used to increase the accuracy number.

SckitLearn's Random Forest Regressor, relatively more accurate than linear regression, was utilized to investigate whether our independent variables, population, GDP per capita, year, and sex can be used to predict suicide number per 100 thousand. Results in the training set were a whopping 98.3% accuracy and an rmse of 113.58. Impressively, the testing set produced 89.02% and an rmse of 302.2. Thus, giving us a very decent model which was neither under not overfit.

Moreover, in the hopes of getting a better model, an ensemble regressor was used that produced 82.42% accuracy with rmse of 375.4 for the training set and 77.3% accuracy with rmse of 433 for the testing set. Relatively, this model was not a bad one, nevertheless, the Random forest regressor turned out to be our best model.

5. Interpretation

The interpretation of the model briefly is to utilize a given data to predict the number of suicide. Using or best model, and given a population, GDP per capita, year, and sex, we can predict the number of suicide per 100 thousand with a whopping accuracy of 98.3 percent. The results for the other models used in the data set were more or less satisfactory. However, our best model, the Random Forest Regressor, produced very good accuracy and hence precise predictions. Consequently, the model was very successful and insightful. Another model that can be used for this data set is a clustering model to group countries with a certain number of suicides.

In conclusion, it is important to understand that suicide has more nuances than what the data set has provided. People commit suicide for a plethora of reasons, including but not limited to psychological causes, demographics (race, ethnicity), and trauma. And so, the numbers in our predictions are highly dependent on these uncountable variables.

References

- Dataset: https://www.kaggle.com/russellyates88/suicide-rates-overview-1985-to-2016
- Scikit learn: https://scikit-learn.org/stable/supervised_learning.html#supervised-learning
- Other Sources: https://www.kaggle.com/tavoosi/suicide-data-full-interactive-dashboard
- Definition: https://www.investopedia.com/terms/h/human-development-index-hdi.asp#:~:text=The%20HDI%20is%20a%20measurement,to%20rank%20and%20compare%20countries.

Appendices

Ghgh

```
#!/usr/bin/env python

# coding: utf-8

# In[1]:

# import packages

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

# Seaborn visualization library

import seaborn as sns

import statsmodels.api as sm

# ###### Read our suicide rate data set

# In[50]:

df = pd.read_csv('master.csv')

df

# ###### Clean the data

# In[3]:

# change the columns names for better description and easy use

df.columns=["country","year","gender","age_group","number_of_suicide","population","suicide_per_100_thousand","country_year","hdi","gdp","gdp_per_capita","generation"]

df.head()

# In[4]:
```

```
# since hdi has a lot of missing values, we can replace it with the mean

df.fillna(df.mean(), inplace=True)

df

# In[5]

# delete column 'country_year' for concision

del df['country_year']

#delete hdi because of a lot of missing data

#del df['hdi']

df.head()

# In[6]:

# change the age group to text description

df.age_group.unique()

# In[7]:

len(df.country.unique())

# In[9]:

df.gender.unique()

# In[10]:

df["age_group"]=df["age_group"].apply(lambda x: str(x).replace('5-14 years','9.5') if '5-14
years' in str(x) else str(x))

df["age_group"]=df["age_group"].apply(lambda x: str(x).replace('15-24 years','19.5') if
'15-24 years' in str(x) else str(x))

df["age_group"]=df["age_group"].apply(lambda x: str(x).replace('25-34 years','29.5') if
'25-34 years' in str(x) else str(x))

df["age_group"]=df["age_group"].apply(lambda x: str(x).replace('35-54 years','44.5') if
'35-54 years' in str(x) else str(x))
```

```
df["age_group"]=df["age_group"].apply(lambda x: str(x).replace('55-74 years','64.5') if
'55-74 years' in str(x) else str(x))

df["age_group"]=df["age_group"].apply(lambda x: str(x).replace('75+ years','75') if '75+
years' in str(x) else str(x))

df.head()

# In[11]:

df['age_group'] = df['age_group'].astype(float)

# In[12]:

lst = []

for i in range(len(df.gender)):

    if df.gender[i] == 'male':

        lst.append(0)

    else:

        lst.append(1)

df['sex'] = lst

df.head()

# In[13]:

df

# ###### Data exploration and summary

# In[14]:

# Create the default pairplot

#sns.pairplot(df)

# ##### Suicide number by Year

# In[15]:
```

```
plt = plt.subplots(1, 1, figsize = (16 ,6))

plt = sns.barplot(x = df['year'], y = 'number_of_suicide',data = df, palette='Spectral')

# In[16]:

# arrange number of suicided from high to low

suicide_year = df.groupby('year')[['number_of_suicide']].sum().reset_index()

suicide_year.sort_values(by='number_of_suicide',
ascending=False).style.background_gradient(cmap='Greens',
subset=['number_of_suicide'])

# Suicide number with age groups

# ###### The highest number of suicides was in 1999.

# ###### The lowest number of suicides was in 2016.

# ##### Suicide number with age and sex

# In[17]:

import matplotlib.pyplot as plt

plt = plt.subplots(1, 1, figsize = (16 ,6))

plt1 = sns.barplot(x = df['age_group'], y = 'number_of_suicide', hue='sex', data = df,
palette = 'Accent')

# Males are, in all the years, more likely to commit suicide than females. Early edulthood
and adulthood are the age groups with the most suicide rate.

# ##### Suicide number with generation

# In[18]:

import matplotlib.pyplot as plt

# suicide based on generation

df.generation.dropna(inplace = True)

labels = df.generation.value_counts().index
```

```
sizes = df.generation.value_counts().values

plt.figure(0,figsize = (7,7))

plt.pie(sizes, labels=labels, autopct='%1.1f%%')

plt.title('Suicide According to Generation',color = 'black',fontsize = 17)

plt.show()

# ##### Suicide number with generation and sex

# In[19]:

import matplotlib.pyplot as plt

plt = plt.subplots(1, 1, figsize = (16 ,6))

plt2 = sns.barplot(x = df['generation'], y = 'number_of_suicide',

            hue='sex',data=df, palette='autumn')

# This visualizations shows that generation boomers have the highest suicide rate.

# ##### Number of suicide by country

# In[20]:

# top 10 countries by most suicide number

import matplotlib.pyplot as plt

plt = plt.subplots(1, 1, figsize = (16 ,6))

dfcountry =
df['number_of_suicide'].groupby(df.country).sum().sort_values(ascending=False)

plt3 = sns.barplot(dfcountry.head(10), dfcountry.head(10).index, palette='Reds_r')

# In[21]:
```

```
# least 10 countries by most suicide number

import matplotlib.pyplot as plt

plt = plt.subplots(1, 1, figsize = (16 ,6))

dfcountry =
df['number_of_suicide'].groupby(df.country).sum().sort_values(ascending=False)

plt4 = sns.barplot(dfcountry.tail(10), dfcountry.tail(10).index, palette='Blues_r')

# Russian Federation leads the world by number of suicide and Dominica records the
least number of suicides in the world from 1968 to 2016

# In[22]:

# To have a more accurate representation of suicide by country let us compare by their
suicide number per 100 thousand

# top 10 countries by most suicide number per 100 thousand people

import matplotlib.pyplot as plt

plt = plt.subplots(1, 1, figsize = (16 ,6))

dfcountry =
df['suicide_per_100_thousand'].groupby(df.country).sum().sort_values(ascending=False)

plt4 = sns.barplot(dfcountry.head(10), dfcountry.head(10).index, palette='Reds_r')

# In[23]:

# least 10 countries by most suicide number per 100 thousand people

import matplotlib.pyplot as plt

plt = plt.subplots(1, 1, figsize = (16 ,6))

dfcountry =
df['suicide_per_100_thousand'].groupby(df.country).sum().sort_values(ascending=False)

plt4 = sns.barplot(dfcountry.tail(10), dfcountry.tail(10).index, palette='Blues_r')

# The Russian Federation still leads and Dominica is the last; however, we can see
some variations in other countries. We can also see how close mose countries are when
compared using suicide per 100 thousand.
```

```
# Number of suicide and GDP

# In[24]:

(df.dtypes=="object").index[df.dtypes=="object"]

# In[25]:

# number of suicides and GDP per capita

import matplotlib.pyplot as plt

f, ax = plt.subplots(1,1, figsize=(10,8))

plt5 = sns.scatterplot(x="gdp_per_capita", y="number_of_suicide", data=df,
color='purple')

# Prediction

# In[26]:

df.corr()

# In[27]:

# plot a heat map for the correlation

import matplotlib.pyplot as plt

plt = plt.subplots(1, 1, figsize = (16 ,6))

plt6 = sns.heatmap(df.corr(),annot=True, cmap='coolwarm')

# In[28]:

from sklearn.metrics import accuracy_score

from sklearn.neighbors import KNeighborsClassifier

from sklearn.ensemble import RandomForestRegressor

import sklearn.metrics

from sklearn.metrics import r2_score,mean_squared_error
from sklearn.model_selection import train_test_split

from sklearn.ensemble import VotingRegressor

from sklearn.linear_model import LinearRegression

X_train, X_test, y_train, y_test = train_test_split(df[['population', 'sex', 'hdi', 'year',
'gdp_per_capita', 'age_group']],

                                        df['number_of_suicide'],test_size=.4, random_state=42)

# In[29]:

y = df.number_of_suicide

X = df[['population', 'sex', 'hdi', 'age_group']]

X = sm.add_constant(X)  # Adds a constant term to the predictor

# Build the regression with OLS

lr_model = sm.OLS(y, X).fit()

print(lr_model.summary())

# In[30]:

linearM = LinearRegression()

linearM.fit(X_train, y_train)

# In[31]:

y_pred = linearM.predict(X_train)

y_pred

# In[32]:

mse = mean_squared_error(y_train, y_pred)

rmse = np.sqrt(mse)

rmse
```

```
# In[33]:

sklearn.metrics.r2_score(y_train, y_pred)

# In[34]:

RandomForest = RandomForestRegressor()

RandomForest.fit(X_train, y_train)

# In[35]:

y_predi = RandomForest.predict(X_train)

y_predi

# In[36]:

mse = mean_squared_error(y_train, y_predi)

rmse = np.sqrt(mse)

rmse

# In[37]:

sklearn.metrics.r2_score(y_train, y_predi)

# In[38]:

voting = VotingRegressor([('lr', linearM), ('rf', RandomForest)])

y_prediction = voting.fit(X_train, y_train).predict(X_train)

y_prediction

# In[39]:

mse = mean_squared_error(y_train, y_prediction)

rmse = np.sqrt(mse)

rmse

# In[40]:
```

```
sklearn.metrics.r2_score(y_train, y_prediction)

# In[41]:

y_pred = linearM.predict(X_test)

y_pred

# In[42]:

mse = mean_squared_error(y_test, y_pred)

rmse = np.sqrt(mse)

rmse

# In[43]:

sklearn.metrics.r2_score(y_test, y_pred)

# In[44]:

y_predi = RandomForest.predict(X_test)

y_predi

# In[45]:

mse = mean_squared_error(y_test, y_predi)

rmse = np.sqrt(mse)

rmse

# In[46]:

sklearn.metrics.r2_score(y_test, y_predi)

# In[47]:

y_prediction = voting.predict(X_test)

y_prediction

# In[48]:
```

```
mse = mean_squared_error(y_test, y_prediction)

rmse = np.sqrt(mse)

rmse

# In[49]:

sklearn.metrics.r2_score(y_test, y_prediction)
```