Analysis of the world suicide population from 1985 to 2016

Name Jinyun Liu

Student ID 246149448

UPI jliu737

Contents

1. Business and/or Situation understanding

1.1 Identify the objectives of the situation.

Suicide is a dangerous behavior that an individual intentionally or voluntarily takes various means to end his life under the action of complex psychological activities. According to statistics, nearly one million people die by suicide every year in the world. In most European countries, more people die of suicide than traffic accidents each year. Globally, the number of suicide deaths exceeds the combined number of deaths due to murder and war. Suicide has become one of the top ten causes of death in humans. The WHO has listed suicide as a global public health problem and calls on all human society to take measures to prevent suicide tragedies from occurring.

It is difficult for us to directly prevent people from committing suicide, but we can make reasonable suggestions to relevant departments by analyzing the factors affecting suicide and predicting the trend of suicide rate. Therefore, it has become an important issue in the world to predict the future trend of suicide rate and the relationship between suicide and various factors, such as economy, age, gender.

Thus, a study is commissioned with the follow:

- Analyze the future trend of suicide rate.

  By identifying the future trend of suicide, we can give some warning to the world and remind the human society to pay attention to the suicide group.

- Find out the relationship between the suicide rate and different factors and identify the factors that most influence the suicide rate.

  Finding these relationships mean that if we know that these factors affect people's suicide rates, we can try to change these factors to reduce the suicide rate.

Tentatively, the study will be judged a success if:

- Accurate analysis of future suicide rate trends. The margin of error is less than 15%.
- Through analysis, sum up various factors affecting suicide rate and provide effective advice to relevant organizations. After improving the related factors, the suicide rate can be significantly reduced.
- The project is finished in time.

1.2 Assess the situation

1.2 Assess the situation

- Resource Inventory

  All hardware and software should be prepared for this project. In this project we have prepared hardware computer and software python3.8 for project analysis. We have a license to use the software. The data source used in this project is "Suicide Rate Overview from 1985 to 2016" provided by Kaggle which is a reliable, freely-available dataset website. We can get the dataset by click: https://www.kaggle.com/russellyates88/suicide-rates-overview-1985-to-2016

- Requirements

  In order to effectively analyze the trend of suicide rate and the factors affecting suicide, we require professional data experts. I will execute this project as the person in charge. If I encounter difficulties, I can also get help and support from lecturers and tutors. The analysis of the data set should not violate the law on security and privacy. In this project, we will use the open source dataset which is legal and does not involve personal privacy.

- Assumption

  Not 100% of the variables affecting suicide were listed in the data set. For example, diseases, serious diseases can cause people to commit suicide. We assume that these unlisted factors remain unchanged and will not affect our analysis results. It is also believed that the data quality is high so that our analysis results are credible.

- Constraints

  Since we have legal access to the data set, and there is no law that states that we cannot analyze the suicide rate, there is no obvious constraints other than that we need to complete the project on time.

- Risks and contingencies So far we haven't seen any risks. However, the possible risks and its corresponding contingencies plan are shown in below.

Table 1.1: project plan

Possible Risks	**Contingeny Plan**
There is no rich project background	We need to spend time learning a lot of corresponding knowledge.
Time risk	Data development needs to be accelerated and effective.
Failure to analyze effective results is also a risk	Try several models before choosing the final model, compare their performance, and choose the best performing model.
The data in the dataset is cluttered (for example, there are many missing values). The risk is poor data quality can affect the analysis results	We need to spend more time in the data processing phase cleaning the data into effective data sets to provide better and more accurate data for our analysis.

- Cost/Benefit Analysis

In order to conduce this project, we may need to cost our times and manpower resource. The benefit is, we can achieve our goals which can predict the trend of suicide and know the factors which can affect suicide rate. Then, according to the result, we can give effective advice to the relevant agencies in order to prevent the loss of lives.

1.3 Determine data mining objectives

Based on the business objectives, the data mining objectives for the project are as follows:

- Analyze the historical data of the world suicide rate through different data analysis models. Find the lowest error according to the performance of different models, and the best performing model to predict the future development trend of suicide rate.
- Determine the important factors that affect the suicide rate by analyzing the relationship between different factors in the data set and the suicide rate, and rank them according to the degree of impact. Finally, we give the visual effect of data analysis to provide support and suggestions to relevant departments more clearly and intuitively.

1.3.1 Data Mining Success Criteria

- Methods for Model Assessment

  In this study, for Model Assessment, we will use two argument to measure. One is MAE which is mean absolute error. It reflects the difference between the real value and the predicted value. The smaller the value, the better the model performance. The other one is linear correlation. The higher linear correlation the model obtains, the better the model is.

- Benchmark for Evaluating Success

  We plan to set specify number to judge if the evaluating is success. If mean absolute error is less than 0.5 and the linear correlation is higher than 0.9, we say the evaluating is success.

- Subjective Measurements and the Arbiter of Success

  In this study, the measurement of success is if the model predict the objectives accurately and we have given the specify number to measure it.

- Successful deployment of model results a part of data mining success?

  Yes, successful deployment of model results is a part of data mining success. We will deploy the results of different model.

1.4 Producing a Project Plan

- The overview plan for the study is as shown in the table below.

Table 1.2: project plan

Phase	**Time**	**Risks**	**Description**
Business understanding	2 days	Change of policy	Find data mining goals
Data understanding	2 days	Technology problems	Explore and understand the data
Data preparation	3 day	Data & technology problems	Do some operations in data to make it more conducive to analyzing the target
Modeling	3 days (multiple iterations occur at this phase)	Technology problems, time to find adequate model	Run results with different models, multiple iterations occur at this stage
Evaluation	2 days	Inability to implement results	Evaluate the results
Explanation	2 days	Misunderstanding	Explain the results

- Gantt chart of project schdule

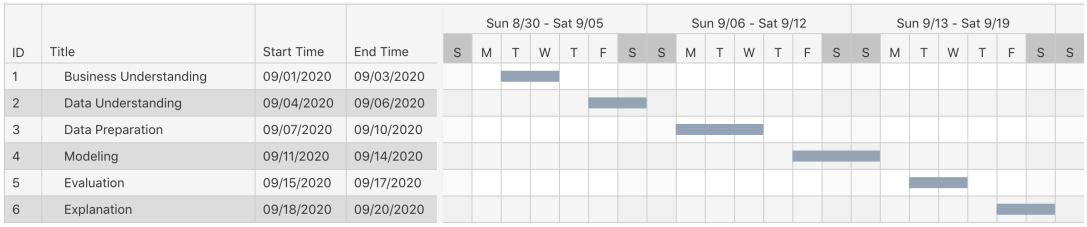

Figure 1.1: Gantt chart of project schdule

2. Data understanding

2.1 Collect initial data

The dataset used in this study comes from the free and open source website Kaggele. I downloaded the dataset named Suicide Rates Overview 1985 to 2016 from kaggle by this URL: https://www.kaggle.com/russellyates88/suicide-rates-overview-1985-to-2016. This dataset pulled from four other datasets linked by time and place. The data in the dataset comes from trusted data sources like World Bank and United Nations Development Program.

2.2 Describe the data

- Amount of data:

  This data set records 27,820 records and 12 attributes of suicide data.

```
In [1]: import pandas as pd

In [2]: datas = pd.read_csv("suicide.csv")

In [3]: datas
Out[3]:
          country  year  ... gdp_per_capita ($)       generation
0         Albania  1987  ...                796     Generation X
1         Albania  1987  ...                796          Silent
2         Albania  1987  ...                796     Generation X
3         Albania  1987  ...                796  G.I. Generation
4         Albania  1987  ...                796          Boomers
...           ...   ...  ...                ...              ...
27815  Uzbekistan  2014  ...               2309     Generation X
27816  Uzbekistan  2014  ...               2309          Silent
27817  Uzbekistan  2014  ...               2309     Generation Z
27818  Uzbekistan  2014  ...               2309     Generation Z
27819  Uzbekistan  2014  ...               2309          Boomers

[27820 rows x 12 columns]

In [4]: datas.shape
Out[4]: (27820, 12)
```

Figure 2.1: Date Amount1

The time interval of this data is from 1985 to 2016. The size of the file is about 2.58 MB. The code output is shown in below.

```
In [5]: datas["year"].min()
Out[5]: 1985

In [6]: datas["year"].max()
Out[6]: 2016

In [7]: import os

In [8]: os.path.getsize("suicide.csv")/1024**2
Out[8]: 2.581000328063965
```

Figure 2.2: Date Amount2

- Format of data: The overview of data types in this dataset has list in below.

```
In [9]: datas.dtypes
Out[9]:
country                 object
year                     int64
sex                     object
age                     object
suicides_no              int64
population               int64
suicides/100k pop      float64
country-year            object
HDI for year           float64
 gdp_for_year ($)       object
gdp_per_capita ($)       int64
generation              object
dtype: object
```

Figure 2.3: Date types

- Value types

  - country: names of different countries

  - year: year of data in this row

  - sex: the sex of the person who committed suicide

  - age: the age range of the person who committed suicide

  - suicides_no: number of suicides that fit this line of information

  - population: number of population that fit this line of information

  - suicides100k pop: suicide rate(suicides_no/population)

  - country-year: record the country's name and year together

- HDI for year: this year's Human Development Index
- gdp_for_year ($): the GDP for the country
- gdp_per_capita ($): per capita GDP
- generation: which generation belongs to

2.3 Explore the data

In this step, we need to explore the data. In order to understand the data more intuitively, the first thing to display is an overview of the dataset. It can be seen the three columns of suicides_no, population and suicides/100k pop. In this study, we mainly study what factors affect the suicide rate. The suicides/100k pop is got by suicides_no divide by population. Therefore, in the next study, we should remove the two factors(population and suicides/100k pop).

```
In [55]: datas
Out[55]:
          country  year     sex          age  suicides_no  population  \
0         Albania  1987    male  15-24 years           21      312900
1         Albania  1987    male  35-54 years           16      308000
2         Albania  1987  female  15-24 years           14      289700
3         Albania  1987    male    75+ years            1       21800
4         Albania  1987    male  25-34 years            9      274300
...           ...   ...     ...          ...          ...         ...
27815  Uzbekistan  2014  female  35-54 years          107     3620833
27816  Uzbekistan  2014  female    75+ years            9      348465
27817  Uzbekistan  2014    male   5-14 years           60     2762158
27818  Uzbekistan  2014  female   5-14 years           44     2631600
27819  Uzbekistan  2014  female  55-74 years           21     1438935

       suicides/100k pop    country-year  HDI for year  gdp_for_year ($)   \
0                   6.71     Albania1987           NaN      2,156,624,900
1                   5.19     Albania1987           NaN      2,156,624,900
2                   4.83     Albania1987           NaN      2,156,624,900
3                   4.59     Albania1987           NaN      2,156,624,900
4                   3.28     Albania1987           NaN      2,156,624,900
...                  ...             ...           ...                ...
27815               2.96  Uzbekistan2014         0.675     63,067,077,179
27816               2.58  Uzbekistan2014         0.675     63,067,077,179
27817               2.17  Uzbekistan2014         0.675     63,067,077,179
27818               1.67  Uzbekistan2014         0.675     63,067,077,179
27819               1.46  Uzbekistan2014         0.675     63,067,077,179

       gdp_per_capita ($)       generation
0                     796     Generation X
1                     796           Silent
2                     796     Generation X
3                     796  G.I. Generation
4                     796          Boomers
...                   ...              ...
27815                2309     Generation X
27816                2309           Silent
27817                2309     Generation Z
27818                2309     Generation Z
27819                2309          Boomers

[27820 rows x 12 columns]
```

Figure 2.4: Overview of dataset

We also can display some descriptive statistics of the data.

From the output, we can see that the field named "HDI for year" has only 8364 pieces of data, which is far less than other fields. Based on this, we

```
In [27]: datas.describe()
Out[27]:
                year   suicides_no    population  suicides/100k pop  \
count  27820.000000  27820.000000  2.782000e+04       27820.000000
mean    2001.258375    242.574407  1.844794e+06          12.816097
std        8.469055    902.047917  3.911779e+06          18.961511
min     1985.000000      0.000000  2.780000e+02           0.000000
25%     1995.000000      3.000000  9.749850e+04           0.920000
50%     2002.000000     25.000000  4.301500e+05           5.990000
75%     2008.000000    131.000000  1.486143e+06          16.620000
max     2016.000000  22338.000000  4.380521e+07         224.970000

       HDI for year  gdp_per_capita ($)
count   8364.000000        27820.000000
mean       0.776601        16866.464414
std        0.093367        18887.576472
min        0.483000          251.000000
25%        0.713000         3447.000000
50%        0.779000         9372.000000
75%        0.855000        24874.000000
max        0.944000       126352.000000
```

Figure 2.5: Data Description

can infer that there are many missing values in this column, and we need to process these data later. We noticed that there are only six columns of data description, because the other data types are not numeric. In order to better analyze, we will need to transform these data later.

- Visualization & Analysis:

  One of the purposes of this project is to analyze the future trend of suicide rates, so we need to explore the distribution of suicide rates.

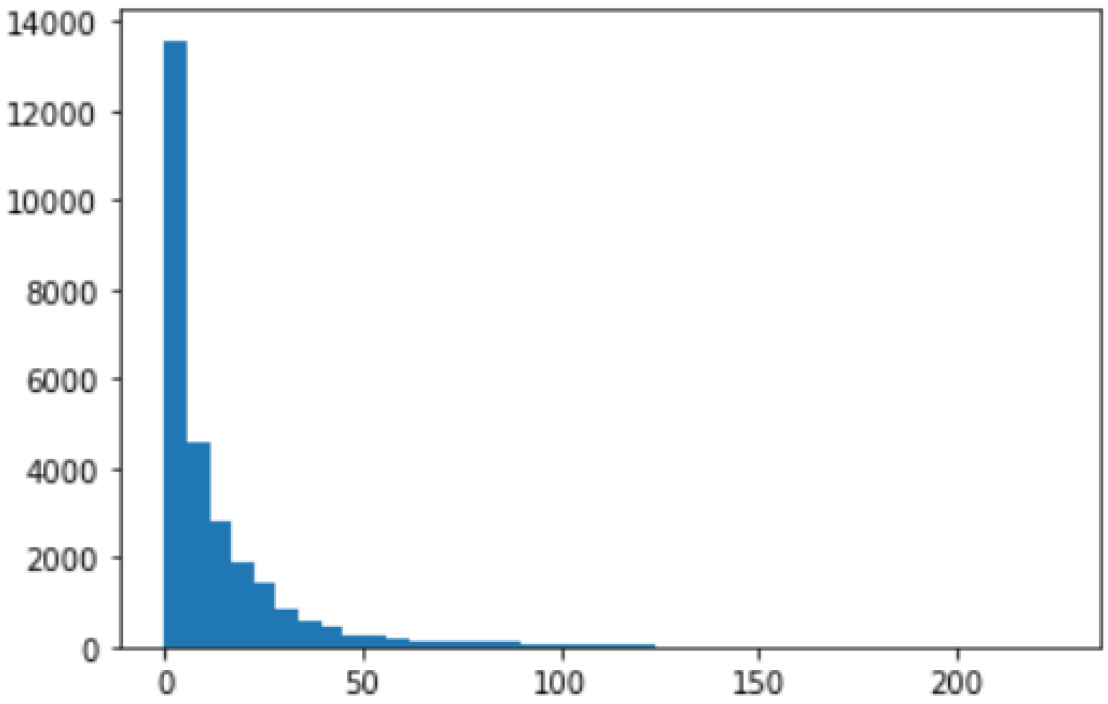

Figure 2.6: The distribution of suicide rates

The Figure2.6 shows the histogram of suicide rate. We can get it is not like normal distribution, so we need to consider to transform it before we creating model. In order to see the trend of suicide rate more clearly, We can draw a scatter plot which describe the relationship between year and suicide rates. It has shown in Figure 2.7.

```
In [41]: plt.scatter(datas["year"], datas["suicides/100k pop"], label="scatter_plot",marker="." )
    ...: plt.xlabel("Year")
    ...: plt.ylabel("suicides/100k pop")
Out[41]: Text(0, 0.5, 'suicides/100k pop')
```

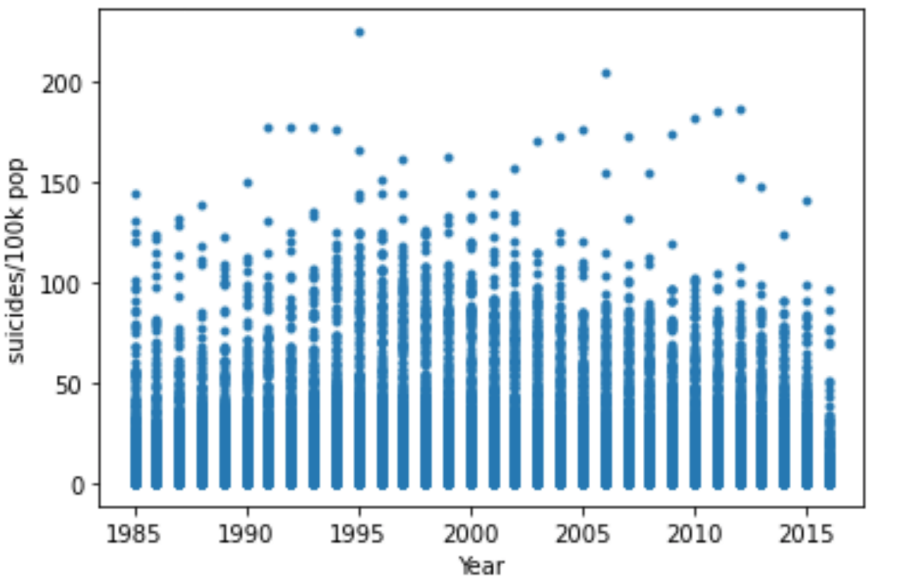

Figure 2.7: The relationship between year and suicide rates

In order to find more factors which can affect the rate of suicide. We use correlation coefficient function to check the importance of features.

```
In [43]: print(datas.corr()["suicides/100k pop"])
year                -0.039037
suicides_no          0.306604
population           0.008285
suicides/100k pop    1.000000
HDI for year         0.074279
gdp_per_capita ($)   0.001785
Name: suicides/100k pop, dtype: float64
```

Figure 2.8: The correlation coefficient

From the result shows in Figure 2.8, we can get the 'year' may be less important than the other factors like 'HDI for year' and 'gdp_per_capita'. However, the year is an important unit of time measurement, so we should take it into consideration. In order to see the relationship between these variables more intuitively, we overall plot the relationships between these variables in Figure 2.9.

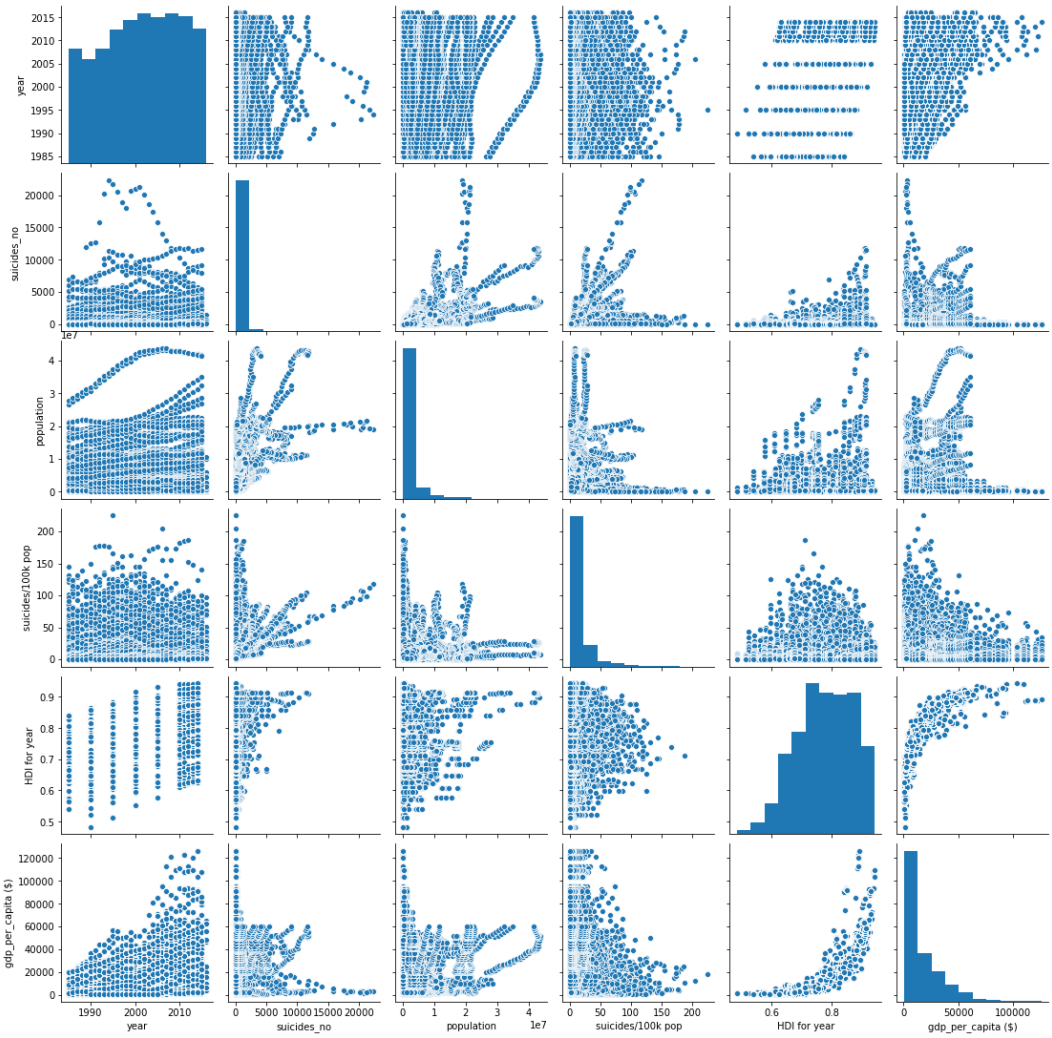

Figure 2.9: The overall plot of correlation coefficient

What we need to note here is that the correlation coefficient function only analyzes the fields of numerical type in the dataset. Therefore, the relationship here is only a preliminary judgment, and a more comprehensive judgment needs to be obtained after transforming the data.

In order to get a more comprehensive correlation analysis, we have drawn the relationship between suicide rate and some non-numerical factors. We drew the relationship between sex and suicide rate as a box plot. From the graph, we can clear see that the males has a higher suicide rate. This means gender is an important factor affecting suicide rates.

```
In [47]: import seaborn as sns

In [48]: sns.catplot(x="sex", y="suicides/100k pop", kind = "box", data=datas)
Out[48]: <seaborn.axisgrid.FacetGrid at 0x122f13370>
```

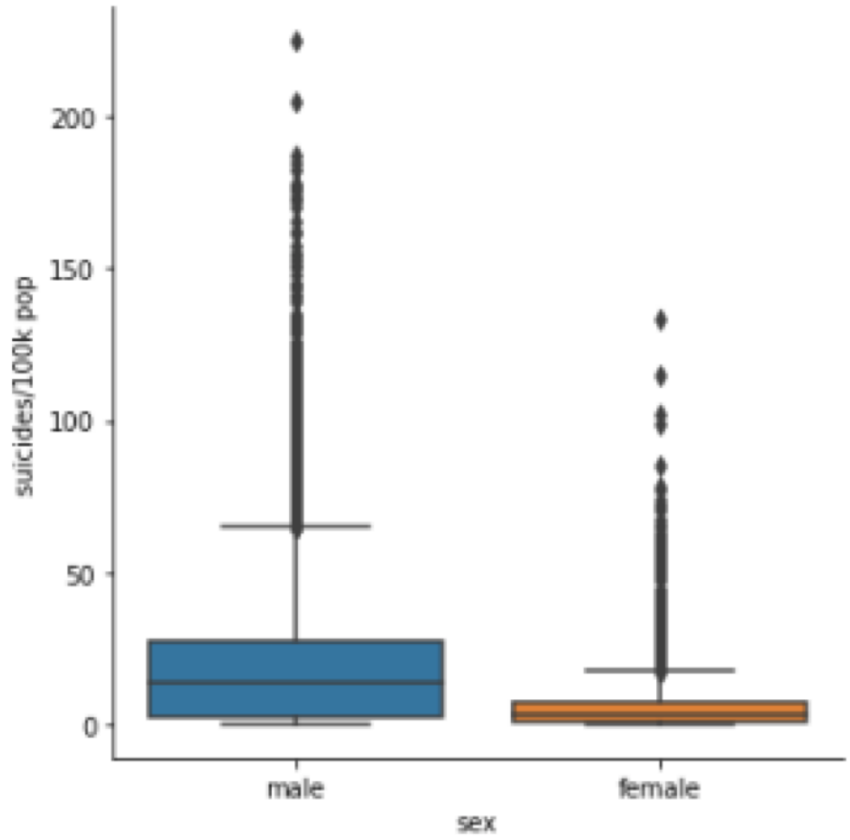

Figure 2.10

We also plot the relationship between age and suicide rate in Figure 2.11. It can be seen that age has an effect on suicide rate. The suicide rate for those younger than 14 is very low.

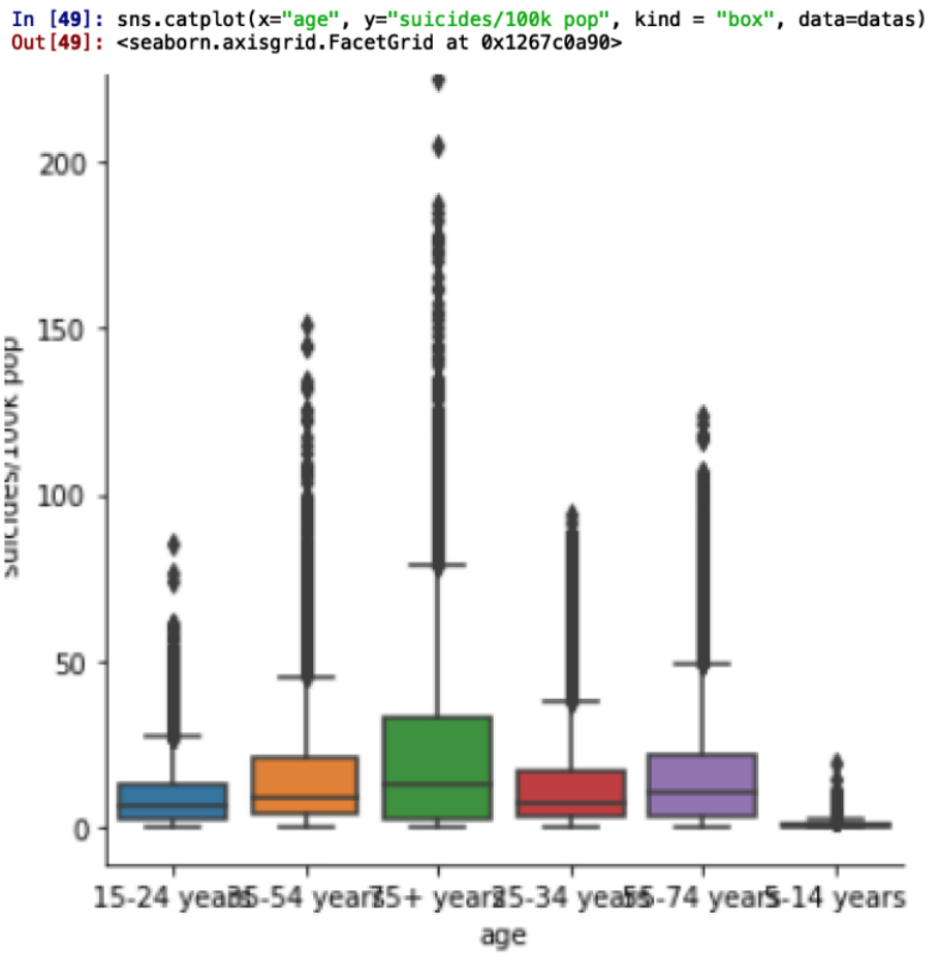

Figure 2.11

Different countries have different cultural and historical backgrounds, which can affect people's personalities. So different countries may also be an important factor affecting suicide rate. Based on this guess, we plot their relation in below. By observing Figure 2.12, different colors represent the suicide rates of different countries. It is not difficult to see that the suicide rates in different countries are significantly different, so the country should also be a factor affecting the suicide rate.

```
In [61]: sns.catplot(x="country", y="suicides/100k pop", data=datas)
Out[61]: <seaborn.axisgrid.FacetGrid at 0x129d5c1f0>
```

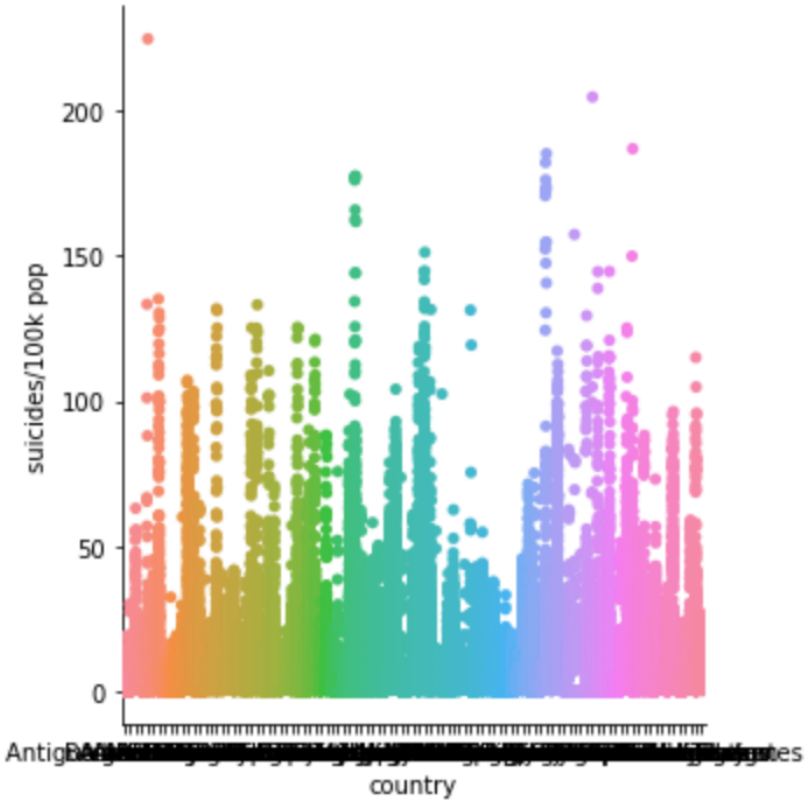

Figure 2.12

2.4 Verify the data quality

- Missing data:

```
In [62]: datas.isnull().any()
Out[62]:
country               False
year                  False
sex                   False
age                   False
suicides_no           False
population            False
suicides/100k pop     False
country-year          False
HDI for year           True
 gdp_for_year ($)     False
gdp_per_capita ($)    False
generation            False
dtype: bool
```

Figure 2.13

Through Figure 2.13, we can see that most fields have no miss values. Most of them are 100% completed. However, it can be found that there are null values in the "HDI for year" field, which is a value describing the human happiness index. It can be found in Figure 2.14 the data in the "HDI for year" field is about only 30% completed. Because there are too many missing values in "HDI for year" field, the data in this filed cannot

be effective for our prediction, so we are going to delete this column in data cleaning state in this project.

```
In [63]: datas.isnull().apply(pd.value_counts)
Out[63]:
       country     year      sex      age  suicides_no  population  \
False  27820.0  27820.0  27820.0  27820.0      27820.0     27820.0
True       NaN      NaN      NaN      NaN          NaN         NaN

       suicides/100k pop  country-year  HDI for year   gdp_for_year ($)   \
False            27820.0       27820.0          8364              27820.0
True                 NaN           NaN         19456                 NaN

       gdp_per_capita ($)  generation
False             27820.0     27820.0
True                  NaN         NaN
```

Figure 2.14

- Outliers and extreme values:

```
In [27]: datas.describe()
Out[27]:
                year   suicides_no    population  suicides/100k pop  \
count  27820.000000  27820.000000  2.782000e+04       27820.000000
mean    2001.258375    242.574407  1.844794e+06          12.816097
std        8.469055    902.047917  3.911779e+06          18.961511
min     1985.000000      0.000000  2.780000e+02           0.000000
25%     1995.000000      3.000000  9.749850e+04           0.920000
50%     2002.000000     25.000000  4.301500e+05           5.990000
75%     2008.000000    131.000000  1.486143e+06          16.620000
max     2016.000000  22338.000000  4.380521e+07         224.970000

       HDI for year  gdp_per_capita ($)
count   8364.000000        27820.000000
mean       0.776601        16866.464414
std        0.093367        18887.576472
min        0.483000          251.000000
25%        0.713000         3447.000000
50%        0.779000         9372.000000
75%        0.855000        24874.000000
max        0.944000       126352.000000
```

Figure 2.15: Data Description

From the description of the data, we can preliminarily infer that there are outliers and extreme values in features suicides_no, population, suicides/100k pop, HDI for year and gdp_per_capita ($). These may be caused by data errors.

In order to more accurately determine the extreme values in the data, I used the method named IQR(interquartile range). IQR is a measure of statistical dispersion. In IQR, all the values are arranged from small to large and divided into four equal parts. The calculation formula of the quartile distance is IQR = Q3-Q1. In this project, we determined if the values are greater than Q3+1.5*(Q3-Q1) or less than Q1-1.5*(Q3-Q1), these values are outliers and extremes.

```
In [113]: Q1 = datas.quantile(0.25)

In [114]: Q3 = datas.quantile(0.75)

In [115]: IQR = Q3 - Q1

In [116]: print((datas<(Q1 - 1.5*IQR)) | (datas>(Q3 + 1.5*IQR)))
       gdp_for_year ($)   HDI for year    age  country  country-year  \
0                 False         False  False    False         False
1                 False         False  False    False         False
2                 False         False  False    False         False
3                 False         False  False    False         False
4                 False         False  False    False         False
...                 ...           ...    ...      ...           ...
27815             False         False  False    False         False
27816             False         False  False    False         False
27817             False         False  False    False         False
27818             False         False  False    False         False
27819             False         False  False    False         False

       gdp_per_capita ($)  generation  population    sex  suicides/100k pop  \
0                   False       False       False  False              False
1                   False       False       False  False              False
2                   False       False       False  False              False
3                   False       False       False  False              False
4                   False       False       False  False              False
...                   ...         ...         ...    ...                ...
27815               False       False        True  False              False
27816               False       False       False  False              False
27817               False       False       False  False              False
27818               False       False       False  False              False
27819               False       False       False  False              False
```

Figure 2.16

In Figure 2.16, "False" means the value is not outliers or extremes and "True" means the value is outliers or extremes.

```
In [117]: datas_outliners = (datas<(Q1 - 1.5*IQR)) | (datas>(Q3 + 1.5*IQR))

In [118]: datas_outliners.sum()
Out[118]:
 gdp_for_year ($)        0
HDI for year            12
age                      0
country                  0
country-year             0
gdp_per_capita ($)    1016
generation               0
population            4180
sex                      0
suicides/100k pop     2046
suicides_no           3909
year                     0
dtype: int64
```

Figure 2.17

After this, we sum up the amount of outliers and extremes, which has clearly listed in Figure 2.17. Outliers and extreme values exist in features suicides_no, population, suicides/100k pop, HDI for year and gdp_per_capita ($). These values may caused by data errors or measurement errors. We need to take care and transform them in following stages.

3. Data preparation

3.1 Data Selection

In the previous step, we have a deeper understanding of the data set. We will select data according to the following goals:

1. Whether the data is meaningless.

   After careful observation of the data set, we found the field population and suicides_no are meaningless because our aim in this project is to find factors that most influence the suicide rate. The suicide rate(suicides/100k pop) is obtained by dividing the number of suicides(suicides_no) by the total number of people(population), so these two fields are meaningless for our analysis. We need to remove them.

2. Whether there are repetitive items.

   Repetitive items mean that the meaning of the fields is repetitive. For example, the field country-year is repetitive item. It combines the field country and year. It is meaningless for our prediction. Therefore in this project, we will remove the field of country-year.

```
In [119]: del datas["population"]

In [120]: del datas["suicides_no"]

In [121]: del datas["country-year"]

In [122]: datas.head()
Out[122]:
   country  year     sex          age  suicides/100k pop  HDI for year  \
0  Albania  1987    male  15-24 years               6.71           NaN
1  Albania  1987    male  35-54 years               5.19           NaN
2  Albania  1987  female  15-24 years               4.83           NaN
3  Albania  1987    male    75+ years               4.59           NaN
4  Albania  1987    male  25-34 years               3.28           NaN

   gdp_for_year ($)   gdp_per_capita ($)       generation
0      2,156,624,900                  796     Generation X
1      2,156,624,900                  796           Silent
2      2,156,624,900                  796     Generation X
3      2,156,624,900                  796  G.I. Generation
4      2,156,624,900                  796          Boomers
```

Figure 3.1

3.2 Cleaning the data

3.2.1. Missing value issues

In the previous step, we have found the most missing value is in the filed of "HDI for year". This filed is only 30% completed, which has shown in Figure 3.2.

```
In [63]: datas.isnull().apply(pd.value_counts)
Out[63]:
       country     year      sex      age  suicides_no  population  \
False  27820.0  27820.0  27820.0  27820.0      27820.0     27820.0
True       NaN      NaN      NaN      NaN          NaN         NaN

       suicides/100k pop  country-year  HDI for year   gdp_for_year ($)   \
False            27820.0       27820.0          8364              27820.0
True                 NaN           NaN         19456                 NaN

       gdp_per_capita ($)  generation
False              27820.0     27820.0
True                   NaN         NaN
```

Figure 3.2

Although this is a large-scale defect, we still try to fix it first. We decided to replace missing values with mean value. Using the mean value can disturb the data as little as possible. Figure 3.3 shows there is no missing value in the dataset now. However, so many

```
In [125]: mean = datas.mean()

In [126]: datas.fillna(mean, inplace = True)

In [127]: datas.isnull().apply(pd.value_counts)
Out[127]:
       country   year    sex    age  suicides/100k pop  HDI for year  \
False    27820  27820  27820  27820              27820         27820

        gdp_for_year ($)   gdp_per_capita ($)  generation
False               27820               27820       27820
```

Figure 3.3

fixed missing data may makes this column of data meaningless for our prediction, so in the next step we don't rule out the possibility of deleting it directly.

3.2.2. Outliers and extremes value issues

We have found the outliers and extremes values in chapter 2.4. These outliers and extremes values can have a negative effect on our prediction. In order to deal with outliers and extreme values, we have the following methods:

1. Coerce. Replaces outliers and extreme values with the nearest value that would not be considered extreme.

2. Discard. Discards records with outlying or extreme values for the specified field.

3. Nullify. Replaces outliers and extremes with the null or system-missing value.

4. Coerce outliers / discard extremes. Discards extreme values only.

5. Coerce outliers / nullify extremes. Nullifies extreme values only.

Considering that the number of the outliers and extreme values only accounts for a very small part of the entire data set, so we plan to use IQR to discard them. I have shown the process in Figure 3.4.

```
In [135]: Q1 = datas.quantile(0.25)

In [136]: Q3 = datas.quantile(0.75)

In [137]: IQR = Q3 - Q1

In [138]: datas_cl = datas[~((datas<(Q1 - 1.5*IQR)) | (datas>(Q3 + 1.5*IQR))).any(axis=1)]

In [139]: datas_cl.shape
Out[139]: (17470, 9)
```

Figure 3.4

3.3 Constructing a new data

In this project, I would like to explore the suicide problem of the elderly, so I want to create a new column to distinguish whether the suicide population is the elderly or the young. I will set 55 as the threshold, and people younger than 55 are considered young, and people older than 55 are considered old.

```
In [149]: datas_cl["old_or_young"] = np.where((datas_cl["age"]=="75+ years") | (datas_cl["age"]=="55-74 years"),1,0)
<ipython-input-149-cd2d6af0b691>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy
  datas_cl["old_or_young"] = np.where((datas_cl["age"]=="75+ years") | (datas_cl["age"]=="55-74 years"),1,0)

In [150]: datas_cl.head()
Out[150]:
   country  year     sex          age  suicides/100k pop  HDI for year  \
0  Albania  1987    male  15-24 years               6.71      0.776601
1  Albania  1987    male  35-54 years               5.19      0.776601
2  Albania  1987  female  15-24 years               4.83      0.776601
3  Albania  1987    male    75+ years               4.59      0.776601
4  Albania  1987    male  25-34 years               3.28      0.776601

   gdp_for_year ($)   gdp_per_capita ($)       generation  old_or_young
0      2,156,624,900                 796     Generation X             0
1      2,156,624,900                 796           Silent             0
2      2,156,624,900                 796     Generation X             0
3      2,156,624,900                 796  G.I. Generation             1
4      2,156,624,900                 796          Boomers             0
```

Figure 3.5

After these operations, we can get a new field named "old_or_young", which has shown in Figure 3.5. "1" in the new field means the elderly and "0" means the young.

3.4 Data Integration

In this step, we merged the two csv files to one. The two csv files' data are all about the suicide. They have the same fields, so it is easy to merge them together.

```
data1 = pd.read_csv("suicide-1.csv")
data2 = pd.read_csv("suicide.csv")
datas = data1.append(data2)
```

Figure 3.6: Merge two csv files

3.5 Formatting Data

In this project, our goal is to analyze the data in the dataset and draw the conclusions we want. Therefore, the data type in the data set should be transformed into a state suitable for analysis. We plan to use regression model to try to analyze the data, so the fields we need in the model should be numeric. As we can see in Figure 3.7, the type of many fields are not numeric, so we need to transform them.

```
In [9]: datas.dtypes
Out[9]:
country                 object
year                     int64
sex                     object
age                     object
suicides_no              int64
population               int64
suicides/100k pop      float64
country-year            object
HDI for year           float64
 gdp_for_year ($)       object
gdp_per_capita ($)       int64
generation              object
dtype: object
```

Figure 3.7: Date types

In this data set, the format of "sex", "age", "generation", "country" and "gdp_for_year ($) " fields are string and we decide to change them to numeric.

The code has shown in Figure 3.8.

In this operation, we use 0 to instead "female" and 1 to instead "male" in the field of sex.

```
le = preprocessing.LabelEncoder()
datas_cl["sex"] = le.fit_transform(datas_cl["sex"])
datas_cl["age"] = le.fit_transform(datas_cl["age"])
datas_cl["generation"] = le.fit_transform(datas_cl["generation"])
datas_cl["country"] = le.fit_transform(datas_cl["country"])
datas_cl[' gdp_for_year ($) '] =datas_cl[' gdp_for_year ($) '].apply(lambda x: "".join(x.split(','))).astype('int')
```

Figure 3.8

In the "age" field, we set "15-24 years" as 0, 25-34 years as 1, 35-54 years as 2, 5-14 years as 3, 55-74 years as 4, 75+ years as 5.

In the "Generation" field, there are six kind of generation. They are G.I. Generation, Generation X, Silent, Boomers, Millenials and Generation Z. The Boomers are people born between 1946 and 1964 and we will use 0 to instead this generation. The G.I. Generation are people born between 1900 and 1924 and we will use 1 to instead this generation. The Generation X are people born between 1965 and 1979 and we will use 2 to instead this generation. The Generation Z or Generation Y are people born between 1995 and 2009 and we will use 3 to instead this generation. The Millennials or Generation Y are people born between 1980 and 2000 and we will use 4 to instead this generation. The Silent Generation are people born between 1925 and 1945 and we will use 5 to instead this generation.

In the "country" field, there are 101 countries in this dataset and we use 1–101 to instead these countries(in alphabetical order).

In the "gdp_for_year ($) " field, the original data is of string type separated by commas, so I use code to changed it to be integer, which has shown in Figure 3.8.

After these operations, the type of these fields is int or float, which can be checked in Figure 3.9.

```
In [223]: datas_cl.head()
Out[223]:
   country  year  sex  age  suicides/100k pop  HDI for year  \
0        0  1987    1    0               6.71      0.776601
1        0  1987    1    2               5.19      0.776601
2        0  1987    0    0               4.83      0.776601
3        0  1987    1    5               4.59      0.776601
4        0  1987    1    1               3.28      0.776601

    gdp_for_year ($)   gdp_per_capita ($)  generation  old_or_young
0          2156624900                 796           2             0
1          2156624900                 796           5             0
2          2156624900                 796           2             0
3          2156624900                 796           1             1
4          2156624900                 796           0             0

In [224]: datas_cl.dtypes
Out[224]:
country                 int64
year                    int64
sex                     int64
age                     int64
suicides/100k pop     float64
HDI for year          float64
 gdp_for_year ($)       int64
gdp_per_capita ($)      int64
generation              int64
old_or_young            int64
dtype: object
```

Figure 3.9

4. Data transformation

4.1 Data Reduction

In this step, we will consider three aspects to reduce the data.

1. In the previous steps, when we explored the dataset, we found the field of "HDI for year" is only 30% completed. Although in chapter 3.2.1, we have dealt with this problem with random numbers, we still decided to delete this column considering that the percentage of missing is too large.

2. We also need to reduce the filed which is meaningless. It can be seen the three columns of suicides_no, population and suicides/100k pop. In this study, we mainly study what factors affect the suicide rate. The suicides/100k pop is got by suicides_no divide by population. Therefore, we should remove the two factors(population and suicides_no). The field country-year is repetitive item. It combines the field country and year. It is meaningless for our prediction, so we need to reduce them. We have reduced the three fields in Chapter3.1(Data Selection). Here we are just reiterating why we should reduce these three items.

3. In this project, we want to predict the rate of suicide and the factors which will have important influence in suicide rate, so the factor which is not important should be reduced. We used correlation coefficient function to find the relationship between suicide rate and the other factors. The result shows that the coefficient of variation of year is not important. However, year is an important time variable, so we cannot delete it. Therefore, in this aspect, we will reduce nothing.

```
In [230]: print(datas_cl.corr()["suicides/100k pop"])
country               8.330782e-02
year                 -2.460524e-02
sex                   3.713294e-01
age                   2.726214e-02
suicides/100k pop     1.000000e+00
HDI for year          1.337690e-15
 gdp_for_year ($)     7.737804e-02
gdp_per_capita ($)    1.304312e-01
generation           -1.036409e-01
old_or_young          1.854609e-01
Name: suicides/100k pop, dtype: float64
```

Figure 4.1: The result of Feature Selection

To sum up, in this step, we only need to reduce the fields "HDI for year".

```
In [231]: del datas_cl["HDI for year"]

In [232]: datas_cl.head()
Out[232]:
   country  year  sex  age  suicides/100k pop   gdp_for_year ($)   \
0        0  1987    1    0               6.71         2156624900
1        0  1987    1    2               5.19         2156624900
2        0  1987    0    0               4.83         2156624900
3        0  1987    1    5               4.59         2156624900
4        0  1987    1    1               3.28         2156624900

   gdp_per_capita ($)  generation  old_or_young
0                 796           2             0
1                 796           5             0
2                 796           2             0
3                 796           1             1
4                 796           0             0
```

Figure 4.2: Reduce data

4.2 Project the data

In chapter2.3 named explore the data, we have plot the histogram of suicide rate. The distribution is not a normal distribution, which is inconvenient for us to use regression model.

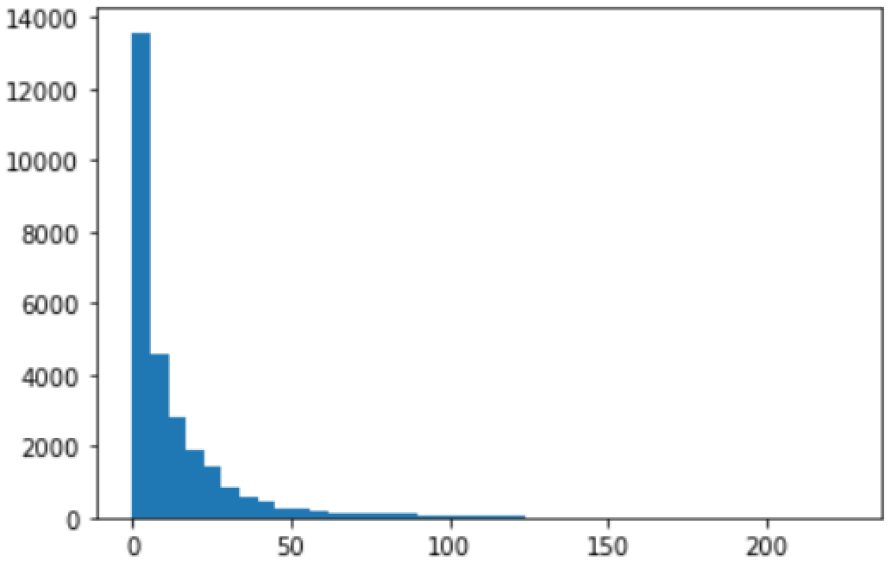

Figure 4.3: The distribution of suicide rates

Therefore, we decide to project the data by statistical transformations (the log of a distribution). In order to conduct this process, first we need to remove the number of zero in "suicides/100k pop" field, because zero can not be logged.

```
In [240]: datas_cl =  datas_cl[datas_cl["suicides/100k pop"] != 0]

In [241]: datas_cl["suicides/100k pop"] = np.log(datas_cl["suicides/100k
pop"])
<ipython-input-241-ea018df14fad>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  datas_cl["suicides/100k pop"] = np.log(datas_cl["suicides/100k pop"])

In [242]: datas_cl.head()
Out[242]:
   country  year  sex  age  suicides/100k pop   gdp_for_year ($)   \
0        0  1987    1    0           1.903599          2156624900
1        0  1987    1    2           1.646734          2156624900
2        0  1987    0    0           1.574846          2156624900
3        0  1987    1    5           1.523880          2156624900
4        0  1987    1    1           1.187843          2156624900

   gdp_per_capita ($)  generation  old_or_young
0                 796           2             0
1                 796           5             0
2                 796           2             0
3                 796           1             1
4                 796           0             0
```

Figure 4.4: Reduce data

After transformation, we plot the suicide rate data again. The distribution of it is close to normal distribution now.

```
In [252]: sns.set_style("darkgrid")

In [253]: sns.distplot(datas_cl["suicides/100k pop"])
Out[253]: <matplotlib.axes._subplots.AxesSubplot at 0x1312e1d60>
```

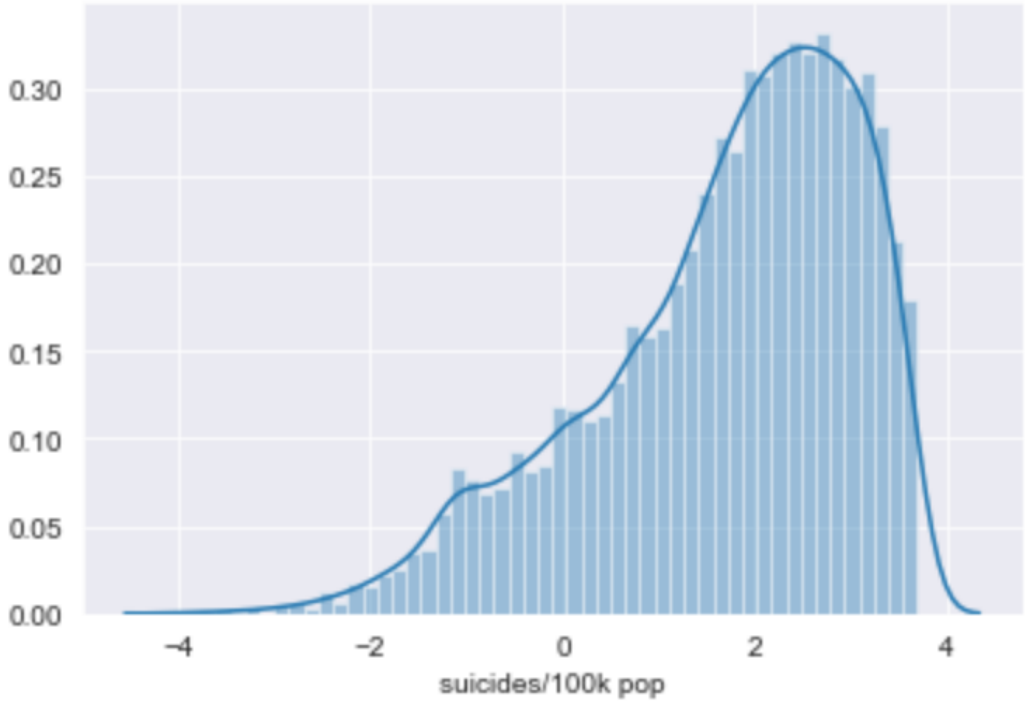

Figure 4.5: Distribution of suicide rate

5. Data-Mining Method(s) Selection

5.1 Discussion of Data Mining Methods in Context of Data Ming Objectives

We have two objectives in this study.

- Analyze the future trend of suicide rate.
- Find out the relationship between the suicide rate and different factors and identify the factors that most influence the suicide rate.

According to the above objectives, our data mining objectives is:

- Analyze the historical data of the world suicide rate through different data analysis models to predict the future trend of suicide rate. Find the lowest error according to the performance of different models, and the best performing model to predict the future development trend of suicide rate.
- Determine the important factors that affect the suicide rate by analyzing the relation-ship between different factors in the data set and the suicide rate, and rank them according to the degree of impact. Finally, we give the visual effect of data analysis to provide support and suggestions to relevant departments more clearly and intuitively.

The methods of data mining can be roughly divided into two categories. They are Supervised Method and Unsupervised Machine Learning Algorithms.

- Supervised Learning

  Supervised Learning is a method in machine learning, which can learn from training materials or establish a pattern (function / learning model), and infer new examples according to this model.

  There are two types of learning supervision problems, One of them is regression problem and another is classification problem.

- Regression

  Regression method is a supervised learning algorithm for predicting and modeling numerical continuous random variables. The characteristic of regression task is that the labeled dataset has numerical target variables. Use cases generally include house price forecast, stock trend or test results and other continuous changes.

- Classification

  Classification method is a supervised learning algorithm for modeling or predicting discrete random variables. Classification algorithms are usually used to predict a category (or the probability of a category) rather than continuous values. Use cases include email filtering, financial fraud, and forecasting employee turnover.

The objective in this project is to predict the trend of suicide rates. The target we want to predict is a numerical variables, so based on the previous discussion, the regression method is more suitable for our project.

- Unsupervised Learning

  The feature of unsupervised learning is that the data of model learning has no label, so the goal of unsupervised learning is to reveal the inherent characteristics and laws of data by learning these unlabeled samples, and its representative is clustering.

  - Clustering

    Clustering is a method of partitioning the dataset into groups. The goal is to split up the data in such a way that points within single cluster are very similar and points in different clusters are different. It determines grouping among unlabeled data

  Our project is to predict the trend of suicide rate and find the factors which can affect the suicide rate. Our target suicide rate has labeled in the dataset, so this method may not suitable for our prediction.

5.2 Select the appropriate data-mining methods

Based on the previous discussion, we will choose regression model of supervised learning to conduct our objectives. Because the objective of this is to predict the trend of suicide rate and find the factors which can affect the suicide rate. The fields in this dataset have been labeled and the target value is real value, so regression model is the most suitable one. We will not choose the classification method because in the classification model, the target value

should be categorical. We will not use the clustering method because our target value is already labeled in the dataset. Neither these two methods is applicable to this project.

6. Data-mining algorithms selection

Sklearn is a third party module commonly used in machine learning, which encapsulates commonly used machine learning methods, including Regression, Dimensionality Reduction, Classfication, Clustering, etc. Based on the discussion in Chapter 5, we would like to choose Supervised regression method. Therefore, we will conduct this kind of model with the Sklearn package approach such as Random Forest, Linear Regression and KNN models. These models have advantages and disadvantages so we need to analyze them further.

6.1 Conduct exploratory analysis and discuss

- Random Forest model
  - Definition

    Random Forest is an advanced implementation of a bagging algorithm with a tree model as the base model. In random forests, each tree in the ensemble is built from a sample drawn with replacement (for example, a bootstrap sample) from the training set. When splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among a random subset of the features. Because of this randomness, the bias of the forest usually slightly increases (with respect to the bias of a single non-random tree) but, due to averaging, its variance also decreases, usually more than compensating for the increase in bias, hence yielding an overall better model.[3] After a series of base learners are obtained, their prediction results are synthesized as the final output of the integration model.
  - Requirements

    The input value and target value of Random Forest model should be numeric.
  - Strengths

It has great advantages in performance compared with other algorithms. A random forest can process high-dimensional data (that is, data with many features) without having to make feature selection. After the training, which characteristics can be given by the random forest are more important.

- Neural Net model
  - Definition

    A neural network can approximate a wide range of predictive models with minimal demands on model structure and assumption. The form of the relationships is determined during the learning process. If a linear relationship between the target and predictors is appropriate, the results of the neural network should closely approximate those of a traditional linear model. If a nonlinear relationship is more appropriate, the neural network will automatically approximate the "correct" model structure.[1]
  - Requirements

    There must be at least one response variable and one input variable.
  - Strengths

    The trade-off for this flexibility is that the neural network is not easily interpretable. If you are trying to explain an underlying process that produces the relationships between the target and predictors, it would be better to use a more traditional statistical model. However, if model interpretability is not important, you can obtain good predictions using a neural network.[1]
- Linear Regression Model
  - Definition

    Linear regression is a common statistical technique for classifying records based on the values of numeric input fields. Linear regression fits a straight line or surface that minimizes the discrepancies between predicted and actual output values. Only numeric fields can be used in a regression model. [2]
  - Requirements

    There must be a Target and at least one Input. By default, fields with predefined roles of Both or None are not used. The target must be continuous (scale).

- KNN Regression Model
  - Definition

    K-nearest neighbor algorithm (KNN) is a supervised learning algorithm, which means that the training data set needs to have label or category. The goal of KNN is to label the unlabeled data points (samples) automatically or predict the category they belong to. KNN can also be used for regression.
  - Requirements

    KNN Regression Model requires the input should be numeric and the target value can be numeric or categorical value.

6.2 Select data-mining algorithms based on the discussion

We have discussed the different data-mining algorithms in the former steps. Our data mining goals are:

- Analyze the historical data of the world suicide rate through different data analysis models to predict the future trend of suicide rate. Find the lowest error according to the performance of different models, and the best performing model to predict the future development trend of suicide rate.
- Determine the important factors that affect the suicide rate by analyzing the relation-ship between different factors in the data set and the suicide rate, and rank them ac-cording to the degree of impact. Finally, we give the visual effect of data analysis to provide support and suggestions to relevant departments more clearly and intuitively.

Combined with the goal of data mining and the discussion in the previous section, we decide to use the following data mining algorithms:

- Linear Regression Model

  Firstly, we decide to use Linear Regression Model to generate our model. Linear Regression Model is a straightforward and easy-understanding model so we wiil try it first. According to the requirements of the linear regression model listed in the previous discussion, our data set meets this requirement, so we can use Linear Regression Model in sklearn package to predict the trend of suicide rates directly.
- Random Forest model

The second method we will use is Random Forest model. According to the discussion in Chapter6.1, Random Forest model is very suitable for our project. It can be used to find the fields which affect the suicide rates.

- KNN Regression Model

  Finally, we will use KNN Regression Model. We wil use the output of this model to compare with the other model. Then, conclude the best model and result of our project.

We will not use Neural Net model, because the neural network is not easily to interpret. We can use the more traditional model to instead of it.

6.3 Build/Select model(s) with algorithm parameters

6.3.1 Linear Regression Model

In the previous chapter, we have introduced Linear regression is a common statistical technique for classifying records based on the values of numeric input fields.

We will use Linear Regression from sklearn package to generate a simple Linear Regression model.

```
#Linear Regression
from sklear.Linear_model import LinearRegression
Linear_model = LinearRegression()
Linear_model.fit(x_train,y_train)
```

Figure 6.1

The parameters we used:

The Linear Regression Model in this project is not complicated, so we do not need special setting. We decided to use default settings in this model.

- fit_intercept, default=True

  Whether to calculate the intercept for this model. If set to False, no intercept will be used in calculations (i.e. data is expected to be centered).

- normalize, default=False

  This parameter is ignored when fit_intercept is set to False. If True, the regressor X will be normalized before regression by subtracting the mean and dividing by the l2-norm.

- copy_X, default=True

  If True, X will be copied; else, it may be overwritten.

- n_jobs, default=None

  The number of jobs to use for the computation. This will only provide speedup for n_targets > 1 and sufficient large problems. None means 1 unless in a joblib.parallel_backend context. -1 means using all processors. See Glossary for more details.

6.3.2 Random Forest model

We will use Random Forest model as the second one to build our model. The input value and target value of Random Forest model should be numeric. Our input and target values meet the requirements. The code has shown in below:

```
#Random Forest Model
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(criterion="mae", max_depth=15, n_estimators=10,random_state=1)
model.fit(x_train,y_train)
```

Figure 6.2

The parameters we used:

- bootstrap, default=True

  Whether bootstrap samples are used when building trees. If False, the whole dataset is used to build each tree.

- criterion="mae"

  Use absolute mean error MAE (mean absolute error), this indicator uses the median value of leaf nodes to minimize L1 loss.

- ma_depth=15

  The maximum depth of the tree. If we set it as a large number, the model will be overfit. If we set it as a small number, the model will be underfit. Therefore, we set it to be 15. It should be a suitable number.

- n_estimators=10

  The number of trees in the forest. We set up 10 trees for this model, because the more trees there are, the higher the accuracy. The more trees there are, the more time it takes. Therefore, we think 10 is a suitable number in our project.

- random_state=1

  Controls both the randomness of the bootstrapping of the samples used when building trees (if bootstrap=True) and the sampling of the features to consider when looking for the best split at each node. We set it to be 1.

6.3.3 KNN Regression Model

We will use KNN Regression Model in this step. KNN is K-nearest neighbor algorithm. It can predict the nearest neighbors of the target value. The code we used is shown in Figure 6.3.

```
#KNN Regression Model
from sklearn import neighbors
knn = neighbors.KNeighborsRegressor()
knn.fit(x_train,y_train)
```

Figure 6.3

The parameters we used: In the parameters of this model, we mostly use the default settings.

- n_neighborsint: default=5

  Number of neighbors to use by default for kneighbors queries.

- leaf_size : int, default=30

  Leaf size passed to BallTree or KDTree. This can affect the speed of the construction and query, as well as the memory required to store the tree. The optimal value depends on the nature of the problem.

- p: default=2

  Power parameter for the Minkowski metric. When p = 1, this is equivalent to using manhattan_distance (l1), and euclidean_distance (l2) for p = 2. For arbitrary p, minkowski_distance (l_p) is used.

- algorithm: default='auto'

  'auto' will attempt to decide the most appropriate algorithm based on the values passed to fit method.

7. Data Mining

7.1 Create and justify test design

If we use the same data set to fit and test the model, our tests will be meaningless because the model is built on the same data set. We're going to get a pretty good result from this test but it doesn't mean anything. Therefore, we need to split the data into training and testing sets. In machine learning, we generally need to divide the sample into two separate training sets (train set) and test set (test set). The training set is used to construct the model, and the test set is used to test the accuracy of the model. For example, we can divide the data set into training set and data set according to 80% and 20%. In this project, we will use Pareto principle to divide our dataset, which we will use 80% to be train set and 20% to be test set. This operation can be done by model_selection in sklearn package.

```
from sklearn.model_selection import train_test_split
datacolumns = datas_cl.columns
inputdata = datas_cl[[c for c in datacolumns if c not in ["suicides/100k pop"]]]
target = datas_cl["suicides/100k pop"]
x_train, x_test, y_train, y_test = train_test_split(inputdata, target, test_size=0.2, random_state=10 )
```

Figure 7.1: Create training and test data

7.2 Conduct data mining(running the model)

In this section, we have split the data. We have 80% data is training data and 20% data is test data. We will conduct data mining with these data. After running the model, we will use the score in metrics and mean_squared_error to help me to evaluate and find the accuracy of this model.

7.2.1 Linear Regression Model

We will use the default parameters to run the model and check mean_squared_error and score in the last step. The code has shown below. As we can see, the MSE is 1.06 and Score is 0.439. The lower the MSE (closer to 0), the better the

```
In [280]: from sklearn.linear_model import LinearRegression

In [281]: Linear_model = LinearRegression()

In [282]: Linear_model.fit(x_train,y_train)
Out[282]: LinearRegression()

In [283]: predictions = Linear_model.predict(x_test)

In [284]: mse = mean_squared_error(predictions, y_test)

In [285]: print("MSE: \n",mse)
MSE:
 1.061985838643618

In [286]: print("Linear Model Score: \n",Linear_model.score(x_test,y_test))
Linear Model Score:
 0.4390798182375979
```

Figure 7.2

model is, and the higher the score, the better the model is. In this model, MSE seems to be relatively high and score is relatively low.

In order to check the fitting effect of the model more intuitively, we draw the residual plot in Figure 7.3.

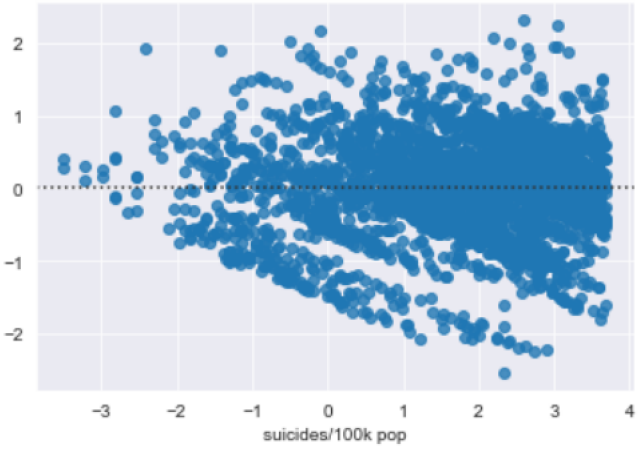

Figure 7.3

If the model fits well, the residuals should change up and down 0, and the changes will be in a fixed range, and there will be no increase or decrease in the range of changes. As we can see in Figure 7.3, there is an increase in the range of changes. According to the MSE and score of the model, the performance of the model seems to be unsatisfactory.

This is the first model we fitted. We will use the result as a reference to compare the quality of the following two models.

7.2.2 Random Forest Model

In this step, we used random forest model to fit the model and the code is shown in below. The MSE of Random Forest Model is 0.201 and the score is

```
In [344]: from sklearn.ensemble import RandomForestRegressor

In [345]: model = RandomForestRegressor(criterion="mae", max_depth=15, n_estimators=10,random_state=1)

In [346]: model.fit(x_train,y_train)
Out[346]:
RandomForestRegressor(criterion='mae', max_depth=15, n_estimators=10,
                      random_state=1)

In [347]: predict_rf = model.predict(x_test)

In [348]: error_rf = mean_squared_error(predict_rf, y_test)

In [349]: print('MSE: \n',error_rf)
MSE:
 0.2011498153559424

In [350]: score_rf = model.score(x_test,y_test)

In [351]: print('Random forest Score : \n',score_rf)
Random forest Score :
 0.8937565955351764
```

Figure 7.4

0.893. The MSE is low and the score is high, which means the performance of this model is good.

In order to check the fitting effect of the model more intuitively, we draw the residual plot of Random Forest Model in Figure 7.5.

```
##plot

import seaborn as sns
import matplotlib.pyplot as plt
sns.residplot(y_test,predict_rf)
```

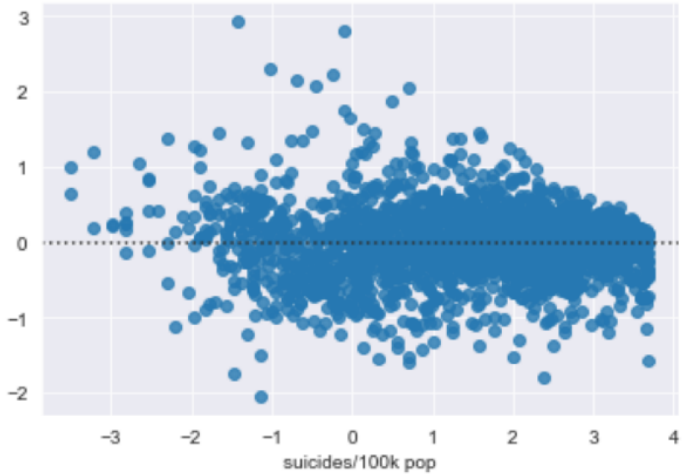

Figure 7.5

As we can see, the residuals change up and down 0, and the changes are almost in a fixed range. Combined with the MSE and score of the model, the performance of this model is fairly good.

7.2.3 KNN Regression Model

The last model is KNN Regression Model, which we have introduced in the previous chapter. We fitted the model and printed the MSE and score of it. The code is shown in below.

```
In [352]: from sklearn import neighbors

In [353]: knn = neighbors.KNeighborsRegressor()

In [354]: knn.fit(x_train,y_train)
Out[354]: KNeighborsRegressor()

In [355]: preds = knn.predict(x_test)

In [356]: error_kn = mean_squared_error(preds, y_test)

In [357]: print('MSE: \n',error_kn)
MSE:
 2.0829408413144077

In [358]: score_knn = knn.score(x_test,y_test)

In [359]: print('knn Score : \n',score_knn)
knn Score :
  0.10016867720445055
```

Figure 7.6

As we can see in Figure 7.6, the MSE of KNN Regression Model is 2.08 which is higher than the other two models. The score of it is 0.10 which is lower than the other two models. It seems to have a poor performance. Let's plot the residual plot to double check.

The residual plot in Figure 7.7 does not look very good. The changes of residuals is not in a fixed range and the range seems to be relatively large in the residual plot. Combined with the MSE and score of the model, it should not be a good model fit.

7.2.4 Assessment

After comparing the performance of the three models(Linear Re-gression Model, Random Forest Model, KNN Regression Model),we find that the per-formance of Random Forest Model is the mostoutstanding. It has the lowest MSE and the highest score(accuracy)in these three models.

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.residplot(y_test,preds)
```

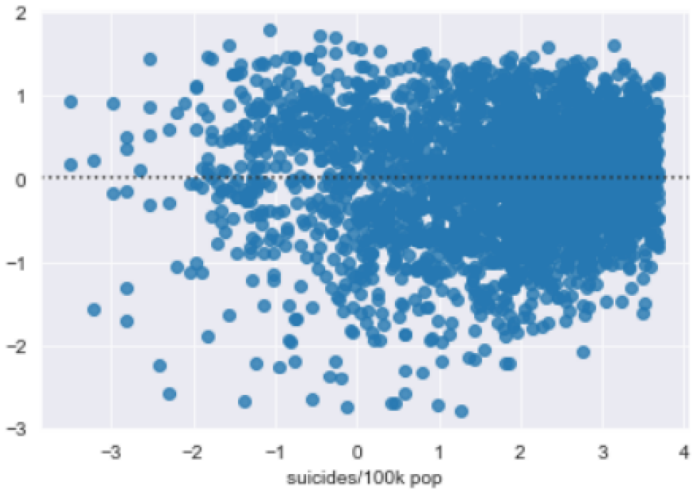

Figure 7.7

7.3 Search for patterns

After the discussion in the previous section, we list the performance of the various models which is shown in below.

Table 7.1

Algorithm	**MSE (mean squared error)**	**Score of Model**
Linear Regression	1.06	0.439
Random Forest	0.201	0.893
KNN Regression	2.08	0.10

From the table, we can see, Random Forest model has the lowest MSE and highest score. In the previous steps, we have explained that the lower the MSE (closer to 0), the better the model is, and the higher the score, the better the model is. Therefore, in these three models, Random Forest model has the best performance. It shows a high accuracy and low mean squared error. The accuracy of Linear Regression and KNN Regression are far lower than Random Forest model and The Mean Squared Error of them are higer than Random Forest model, so we will focus on Random Forest model in the rest of this project.

Now, let's list the output of Random Forest model. It can be concluded that the field of age is the most important field which can influence suicide rate in this model and the importance of country, gdp_for_year, gdp_for_capita and

sex followed close behind.

```
# Random Forest Regression Model
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(criterion="mae", max_depth=15, n_estimators=10,random_state=1)
model.fit(x_train,y_train)
predict_rf = model.predict(x_test)

coef1 = pd.Series(model.feature_importances_,inputdata.columns).sort_values()
coef1.plot(kind='bar', title='Model Coefficients')
```

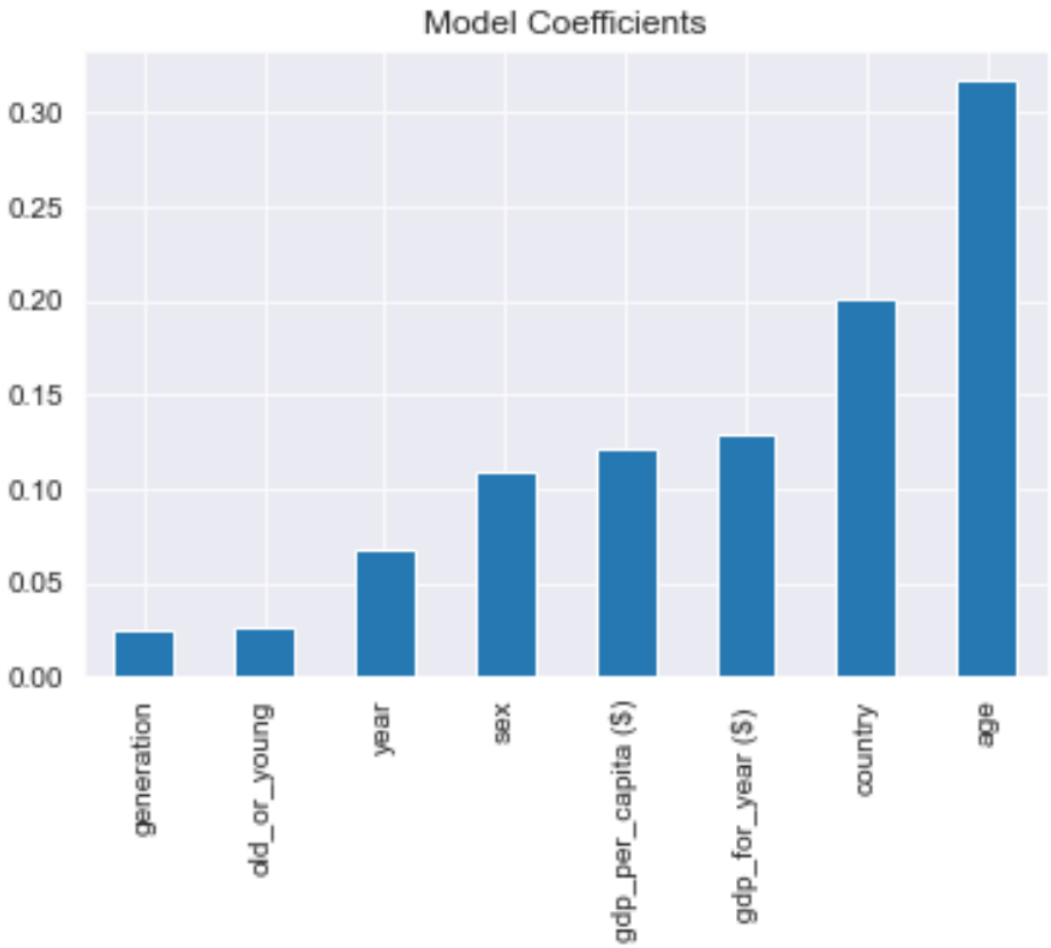

Figure 7.8: The analysis of Random Forest model

8. Interpretation

8.1 Study and discuss the mined patterns.

The data we used in this project is from kaggle which is a open-source website. Firstly, we collect initial data and choose the useful data fields. Then we clean the data which wasn't appropriate. We discarded the extremes and transformed the missing values. After that, We construct a new data named "old_or_young" in the dataset, because we want to explore whether the factor of young or old has an effect on suicide rate. Then, we format the data, we changed the category data into numeric data in order to better fit the model. In the next step, we found the distribution of our target field "suicides/100k pop" is not normal distribution, so we use log to transform it. Because we want to fit model to predict the trend of suicide and find the factors which can influence the suicide rate, so we list some different models and compare them and try to find the best one. We choose three different models to fit and list their performance(MSE and Score of the models). They are Linear Regression Model, Random Forest model and KNN Regression Model. After comparison, we found Random Forest model performs the best and the other two model's accuracy is not so good.

Therefore, we decide to discard the other two models and use Random Forest model to predict the trend of suicide rate and use this model to find the important factors which can affect suicide rate. We list the patterns we found:

- In order of importance field to the suicide rate, age, country, gdp, sex, year, old_or_young and generation. The most important field is age.
- Random Forest model has a higher score than other models and the MAE(mean absolute error) is the lowest. A higher score means the accuracy of Random Forest model is very high. The lowest MSE means the difference between true value and predict value is low.

In conclusion, we will use Random Forest model to conduct the model and find the trend of suicide, important factors.

```
In [344]: from sklearn.ensemble import RandomForestRegressor

In [345]: model = RandomForestRegressor(criterion="mae", max_depth=15, n_estimators=10,random_state=1)

In [346]: model.fit(x_train,y_train)
Out[346]:
RandomForestRegressor(criterion='mae', max_depth=15, n_estimators=10,
                      random_state=1)

In [347]: predict_rf = model.predict(x_test)

In [348]: error_rf = mean_squared_error(predict_rf, y_test)

In [349]: print('MSE: \n',error_rf)
MSE:
 0.2011498153559424

In [350]: score_rf = model.score(x_test,y_test)

In [351]: print('Random forest Score : \n',score_rf)
Random forest Score :
 0.8937565955351764
```

Figure 8.1

```
In [376]: frame = pd.DataFrame({"fields":inputdata.columns, "importances":model.feature_importances_})

In [377]: frame.sort_values(by='importances', ascending=False)
Out[377]:
              fields  importances
3                age     0.316990
0            country     0.201751
4   gdp_for_year ($)     0.129884
5  gdp_per_capita ($)     0.121429
2                sex     0.108581
1               year     0.068536
7       old_or_young     0.027139
6         generation     0.025690
```

Figure 8.2

8.2 Visualize the data, results, models, and patterns.

8.2.1 Visualize the data

- Visualize Data Distribution

```
In [27]: datas.describe()
Out[27]:
                year   suicides_no    population  suicides/100k pop  \
count  27820.000000  27820.000000  2.782000e+04       27820.000000
mean    2001.258375    242.574407  1.844794e+06          12.816097
std        8.469055    902.047917  3.911779e+06          18.961511
min     1985.000000      0.000000  2.780000e+02           0.000000
25%     1995.000000      3.000000  9.749850e+04           0.920000
50%     2002.000000     25.000000  4.301500e+05           5.990000
75%     2008.000000    131.000000  1.486143e+06          16.620000
max     2016.000000  22338.000000  4.380521e+07         224.970000

       HDI for year  gdp_per_capita ($)
count   8364.000000        27820.000000
mean       0.776601        16866.464414
std        0.093367        18887.576472
min        0.483000          251.000000
25%        0.713000         3447.000000
50%        0.779000         9372.000000
75%        0.855000        24874.000000
max        0.944000       126352.000000
```

Figure 8.3: Data Description

- The histogram of target value(log_suicide_rate)

```
In [252]: sns.set_style("darkgrid")

In [253]: sns.distplot(datas_cl["suicides/100k pop"])
Out[253]: <matplotlib.axes._subplots.AxesSubplot at 0x1312e1d60>
```

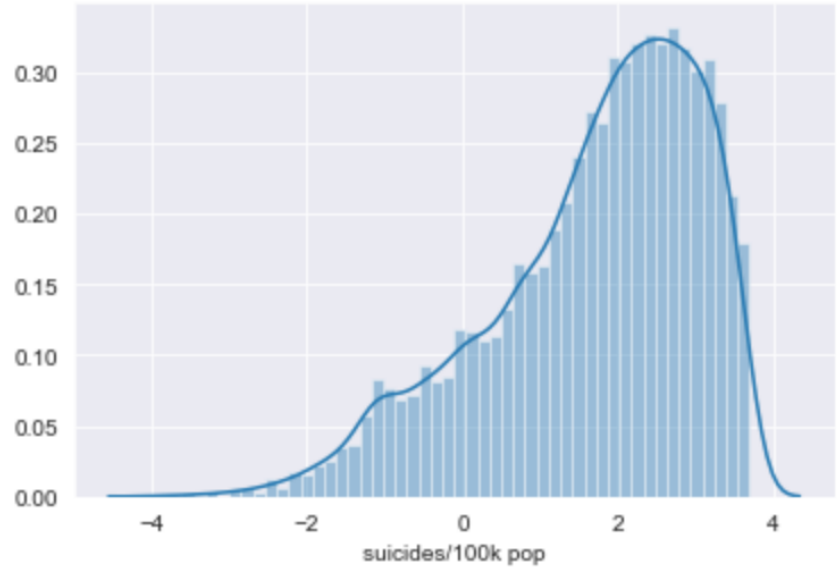

Figure 8.4: Distribution of suicide rate

- Plot the relationship between each field and the target value

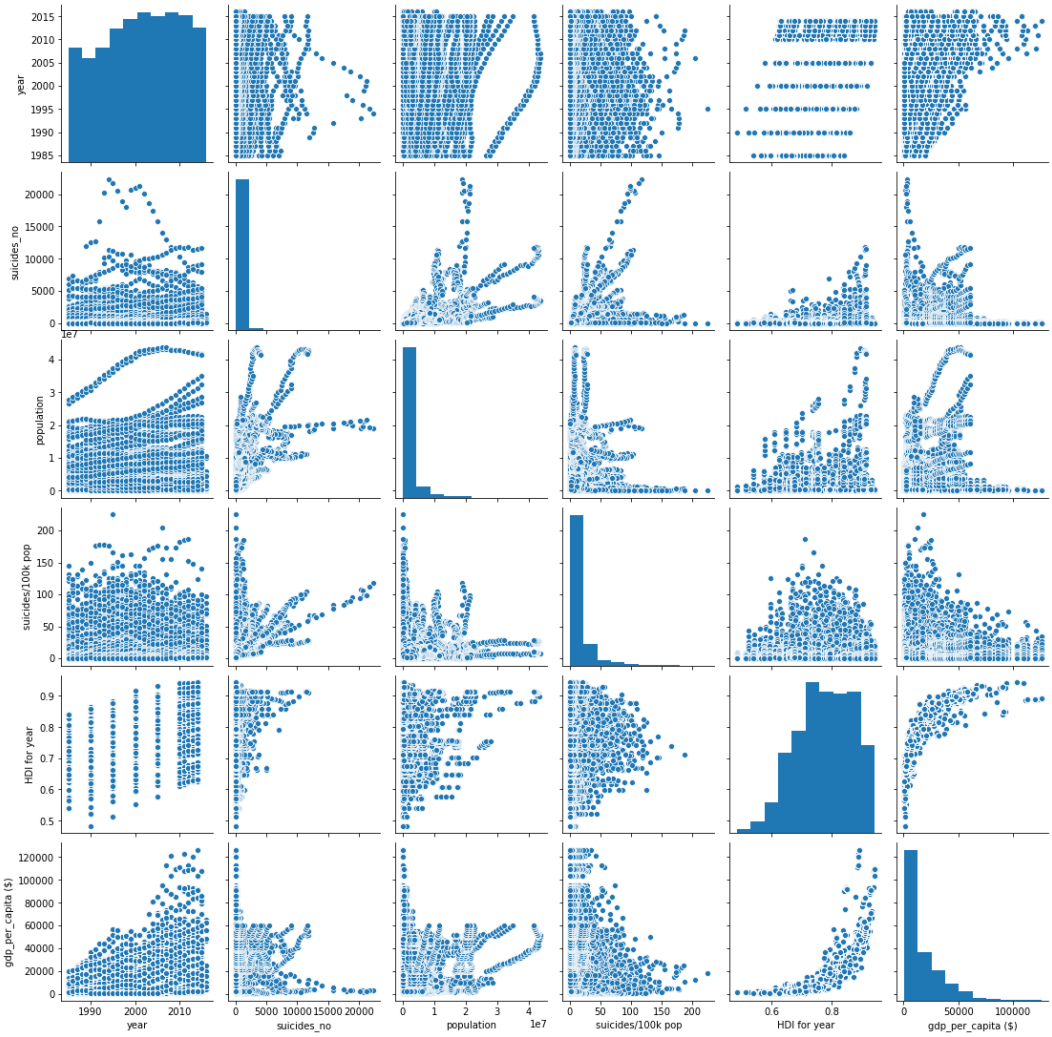

Figure 8.5

8.2.2 Visualize the models and result

– Linear Regression Model

In this section, we will plot the output of the Linear Regression Model. In Figure 8.6, we plot the coefficients of every fields which can also used to represent the field's importance. In Figure 8.7, we can see the fitting process of the model and the MSE and score of the model. In Figure 8.8, we plot the residuals plot of this model.

```
In [389]: framelinear =  pd.DataFrame({"fields":inputdata.columns, "coef":Linear_model.coef_})

In [390]: framelinear.sort_values(by='coef', ascending=False)
Out[390]:
                fields          coef
7        old_or_young  3.047370e+00
2                 sex  9.283038e-01
1                year  4.849060e-03
0             country  1.921904e-03
5  gdp_per_capita ($)  1.054765e-05
4   gdp_for_year ($)  -2.265120e-14
6          generation -2.415111e-01
3                 age -5.995057e-01
```

Figure 8.6

```
In [280]: from sklearn.linear_model import LinearRegression

In [281]: Linear_model = LinearRegression()

In [282]: Linear_model.fit(x_train,y_train)
Out[282]: LinearRegression()

In [283]: predictions = Linear_model.predict(x_test)

In [284]: mse = mean_squared_error(predictions, y_test)

In [285]: print("MSE: \n",mse)
MSE:
 1.061985838643618

In [286]: print("Linear Model Score: \n",Linear_model.score(x_test,y_test))
Linear Model Score:
 0.4390798182375979
```

Figure 8.7

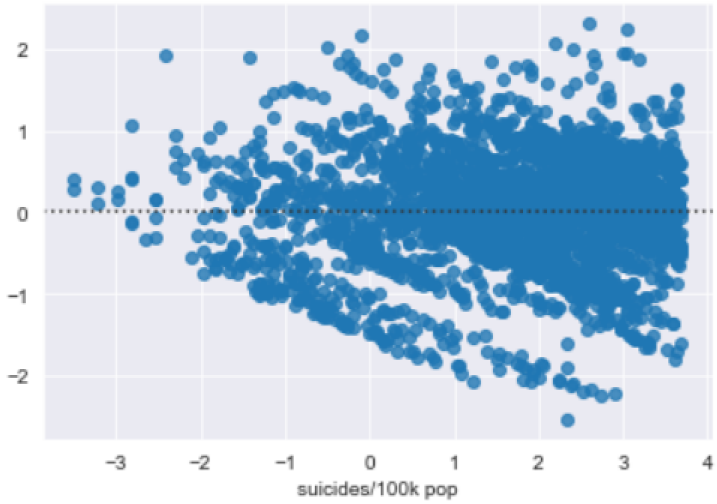

Figure 8.8

– Random Forest Model

In this section, we will plot the output of the Random Forest Model. In Figure 8.9, we plot the importance of every fields. In Figure 8.10, we can see the fitting process of the model and the MSE and score of the model. In Figure 8.11, we plot the residuals plot of this model.

```
In [376]: frame = pd.DataFrame({"fields":inputdata.columns, "importances":model.feature_importances_})

In [377]: frame.sort_values(by='importances', ascending=False)
Out[377]:
              fields  importances
3                age     0.316990
0            country     0.201751
4   gdp_for_year ($)     0.129884
5  gdp_per_capita ($)     0.121429
2                sex     0.108581
1               year     0.068536
7       old_or_young     0.027139
6         generation     0.025690
```

Figure 8.9

```
In [344]: from sklearn.ensemble import RandomForestRegressor

In [345]: model = RandomForestRegressor(criterion="mae", max_depth=15, n_estimators=10,random_state=1)

In [346]: model.fit(x_train,y_train)
Out[346]:
RandomForestRegressor(criterion='mae', max_depth=15, n_estimators=10,
                      random_state=1)

In [347]: predict_rf = model.predict(x_test)

In [348]: error_rf = mean_squared_error(predict_rf, y_test)

In [349]: print('MSE: \n',error_rf)
MSE:
 0.2011498153559424

In [350]: score_rf = model.score(x_test,y_test)

In [351]: print('Random forest Score : \n',score_rf)
Random forest Score :
 0.8937565955351764
```

Figure 8.10

```
##plot

import seaborn as sns
import matplotlib.pyplot as plt
sns.residplot(y_test,predict_rf)
```

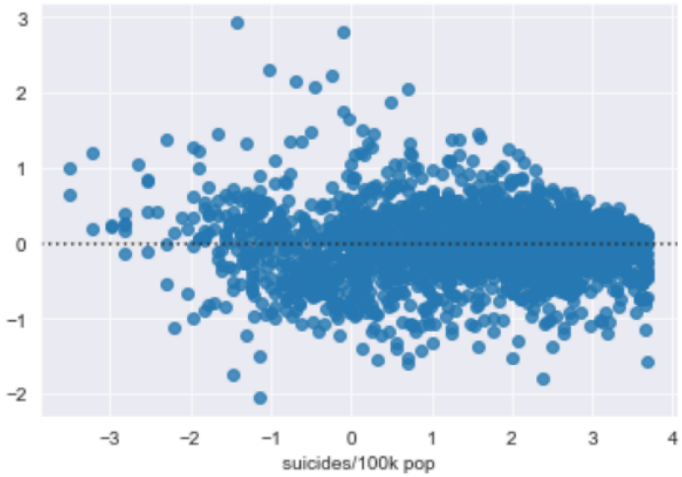

Figure 8.11

– KNN Regression Model

  In this section, we will plot the output of the RKNN Regression Model. In Figure 8.12, we can see the fitting process of the model and the MSE and score of the model. In Figure 8.13, we plot the residuals plot of this model.

```
In [352]: from sklearn import neighbors

In [353]: knn = neighbors.KNeighborsRegressor()

In [354]: knn.fit(x_train,y_train)
Out[354]: KNeighborsRegressor()

In [355]: preds = knn.predict(x_test)

In [356]: error_kn = mean_squared_error(preds, y_test)

In [357]: print('MSE: \n',error_kn)
MSE:
 2.0829408413144077

In [358]: score_knn = knn.score(x_test,y_test)

In [359]: print('knn Score : \n',score_knn)
knn Score :
  0.10016867720445055
```

Figure 8.12

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.residplot(y_test,preds)
```

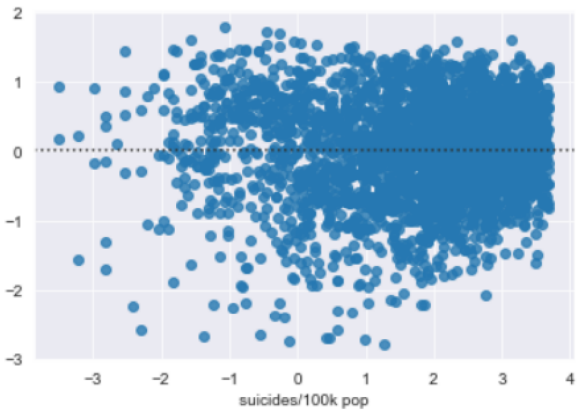

Figure 8.13

After comparing the performance of the three models(Linear Regression Model, Random Forest Model, KNN Regression Model), we find that the performance of Random Forest Model is the most outstanding. It has the lowest MSE and the highest score(accuracy) in these three models. Therefore, we decided to plot the first five important factors and target value(suicide rate) based on the output of the random forest model. From the previous discussion, we can know, in random forest model, the field of age is the most important field which can influence suiciderate in this model and the importance of country, gdp_for_year, gdp_for_capita and sex followed close behind. We will plot their relationship with the target value(suicide rate) in order of importance.

* The relationship between age and suicide rate

```
In [49]: sns.catplot(x="age", y="suicides/100k pop", kind = "box", data=datas)
Out[49]: <seaborn.axisgrid.FacetGrid at 0x1267c0a90>
```

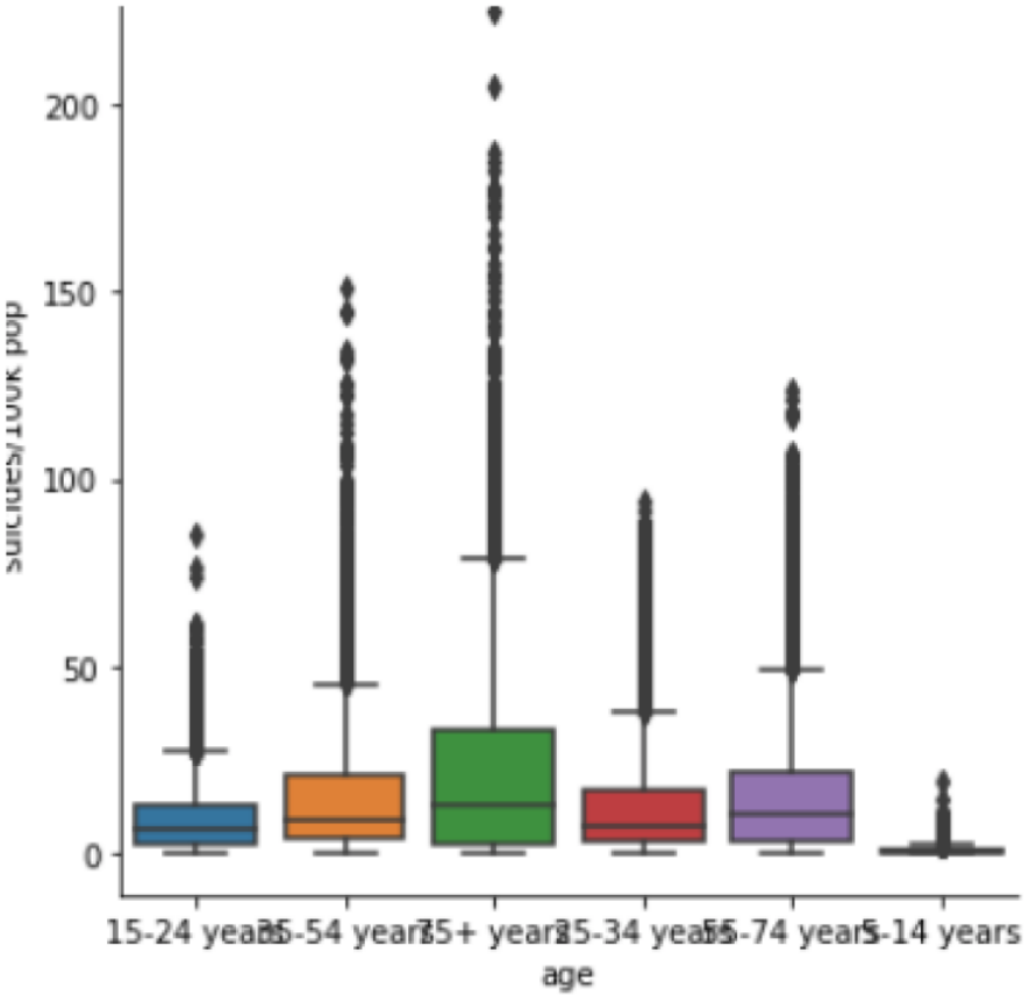

Figure 8.14

* The relationship between country and suicide rate

```
In [61]: sns.catplot(x="country", y="suicides/100k pop", data=datas)
Out[61]: <seaborn.axisgrid.FacetGrid at 0x129d5c1f0>
```

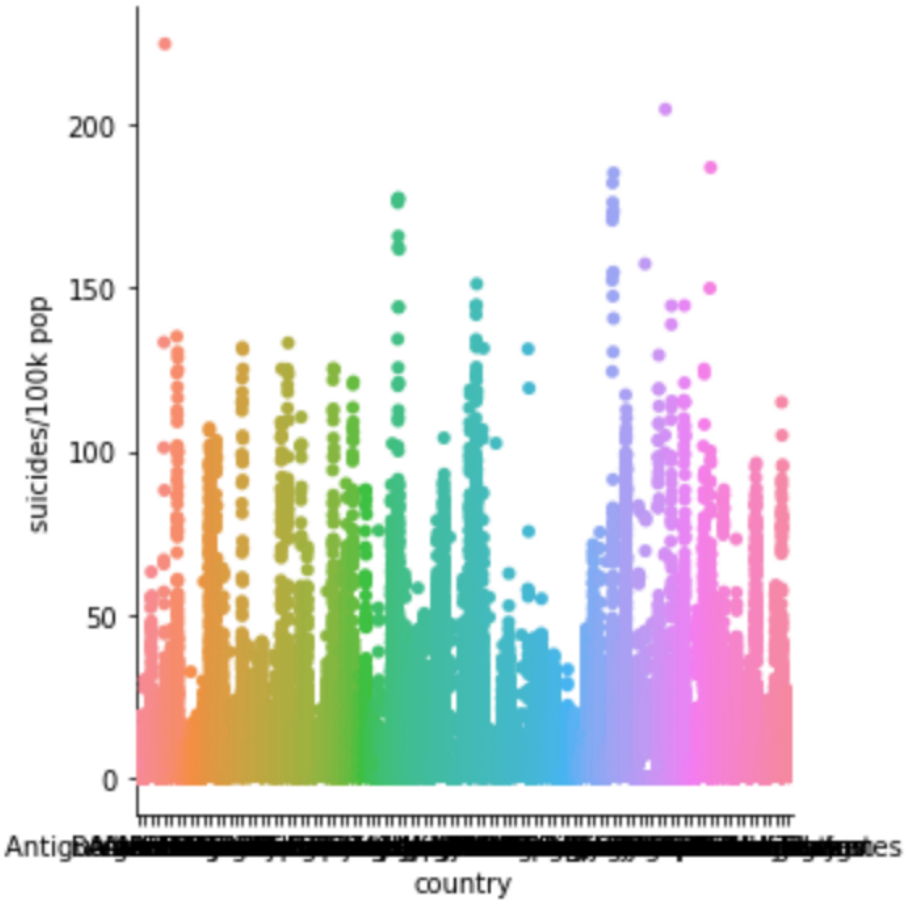

Figure 8.15

∗ The relationship between gdp_for_year and suicide rate

```
sns.relplot(x=' gdp_for_year ($) ', y="suicides/100k pop", kind="line",data=datas )
```

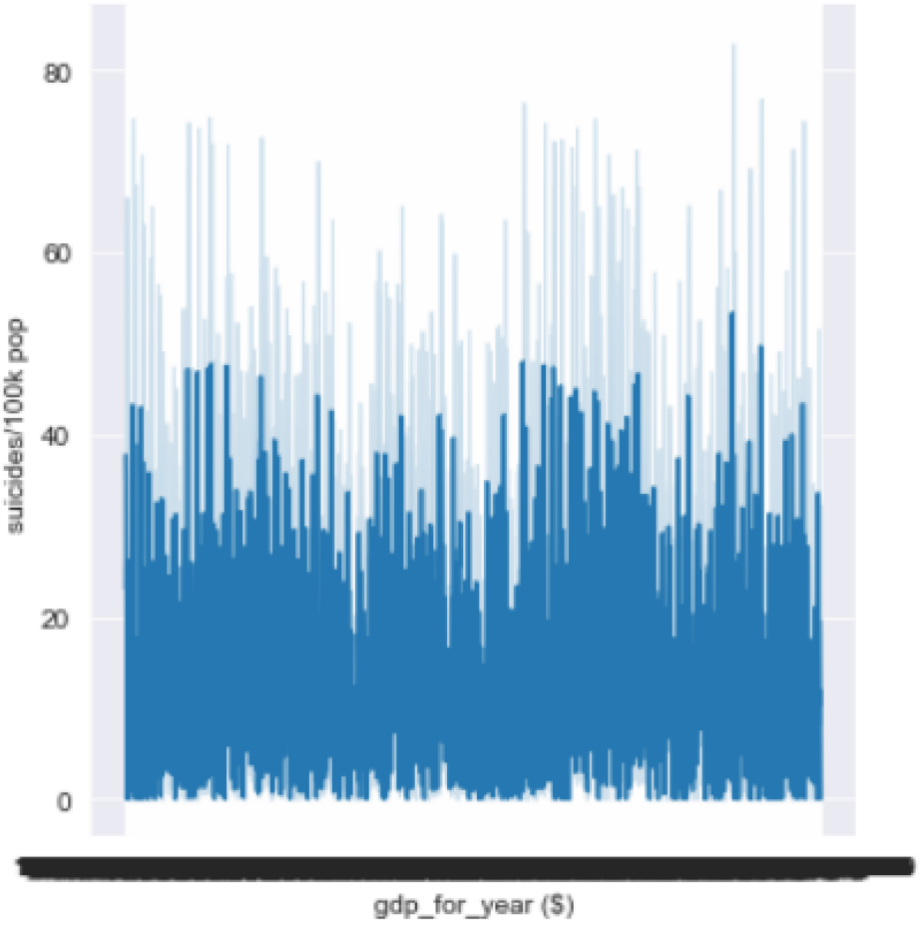

Figure 8.16

* The relationship between gdp_for_capita and suicide rate

```
sns.relplot(x="gdp_per_capita ($)", y="suicides/100k pop", kind="line",data=datas )
```

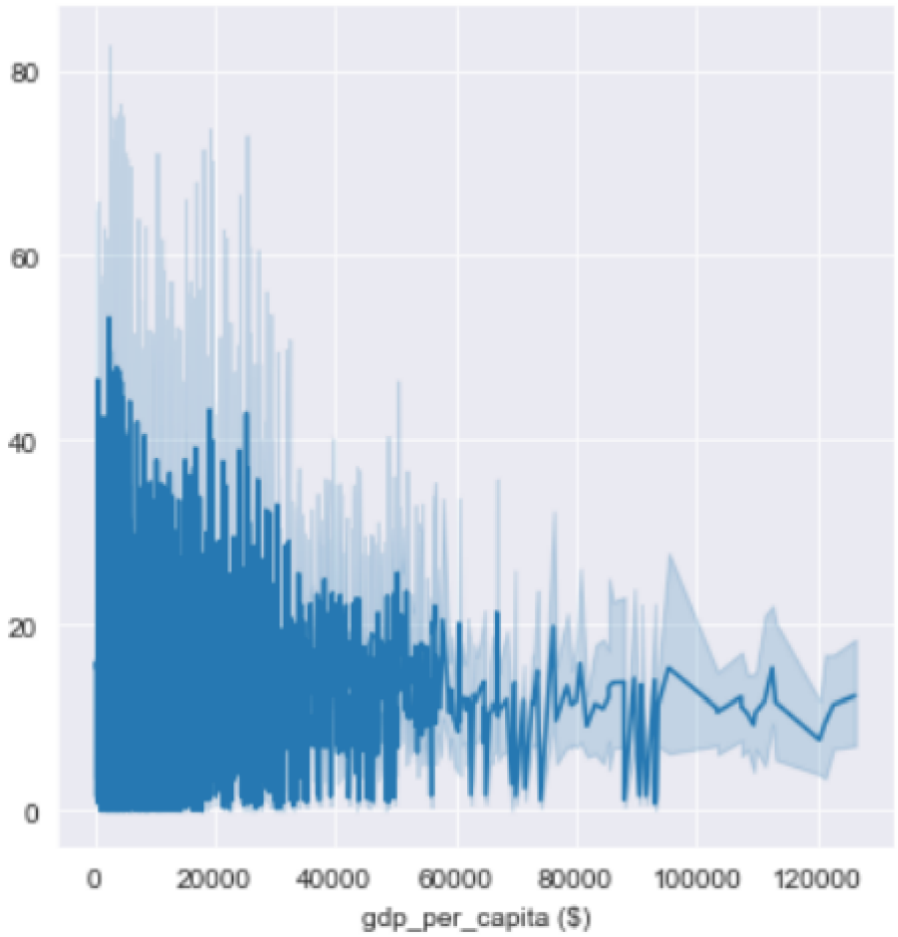

Figure 8.17

∗ The relationship between sex and suicide rate

```
In [47]: import seaborn as sns

In [48]: sns.catplot(x="sex", y="suicides/100k pop", kind = "box", data=datas)
Out[48]: <seaborn.axisgrid.FacetGrid at 0x122f13370>
```

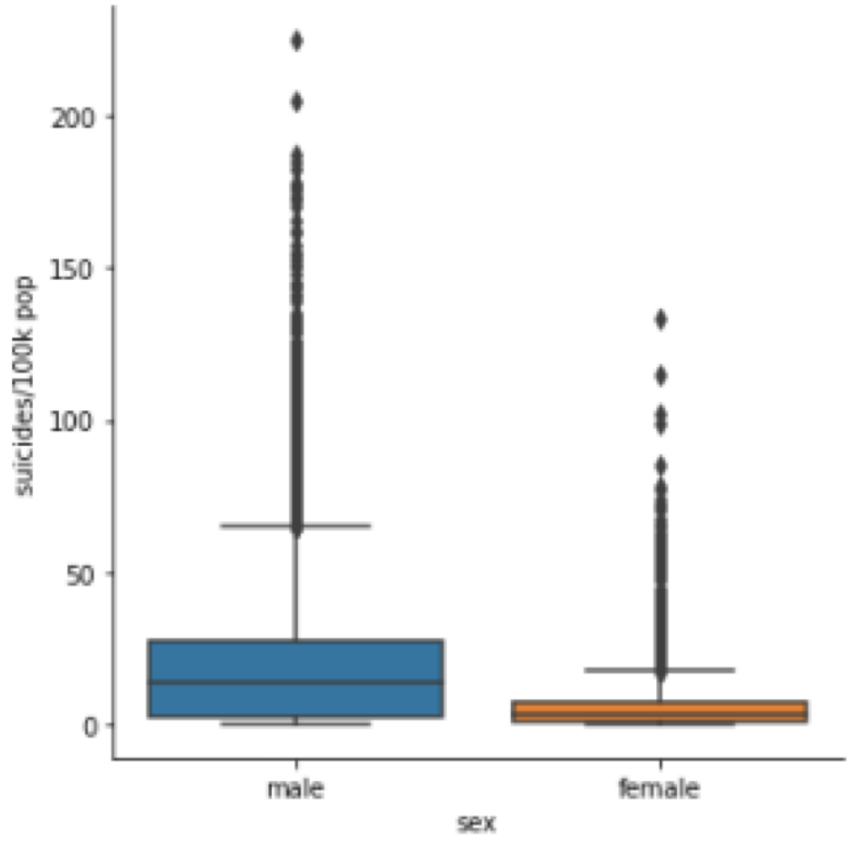

Figure 8.18

8.3 Interpret the results, models, and patterns

We have used three different model. Through the imformance of these models in Table 8.1, we can get the performance of Random Forest Model is the best. It has the highest score (accuracy) and lowest Mean absolute error. The second best performer is Linear Regression Model and the last is KNN Regression Model.

Table 8.1

Algorithm	**MSE (mean squared error)**	**Score of Model**
Linear Regression	1.06	0.439
Random Forest	0.201	0.893
KNN Regression	2.08	0.10

In this step, We will interpret the results and patterns of Random Forest Model which is the most suitable model in these models. Let's list the output of the random forest model again.

```
In [344]: from sklearn.ensemble import RandomForestRegressor

In [345]: model = RandomForestRegressor(criterion="mae", max_depth=15, n_estimators=10,random_state=1)

In [346]: model.fit(x_train,y_train)
Out[346]:
RandomForestRegressor(criterion='mae', max_depth=15, n_estimators=10,
                      random_state=1)

In [347]: predict_rf = model.predict(x_test)

In [348]: error_rf = mean_squared_error(predict_rf, y_test)

In [349]: print('MSE: \n',error_rf)
MSE:
 0.2011498153559424

In [350]: score_rf = model.score(x_test,y_test)

In [351]: print('Random forest Score : \n',score_rf)
Random forest Score :
 0.8937565955351764
```

Figure 8.19

The MSE (mean squared error) of Random Forest Model is 0.201. MSE (mean squared error) is the error between the actual value and predicted value, and the smaller the value the better the model's performance. 0.201 is a small number which is not far from 0, so we can conclude the difference between the value predicted by Random Forest Model and actual value is not large.

The Score of Random Forest Model is 0.893. The Score of model is used to measure how accurate the model. The range of it is between 0 to 1. The closer the value of a is to 1, the higher the accuracy. The Score of Random Forest Model is 0.893. There is no doubt that this is a relatively high number, indicating that the accuracy of this model has reached about 90%.

```
##plot

import seaborn as sns
import matplotlib.pyplot as plt
sns.residplot(y_test,predict_rf)
```

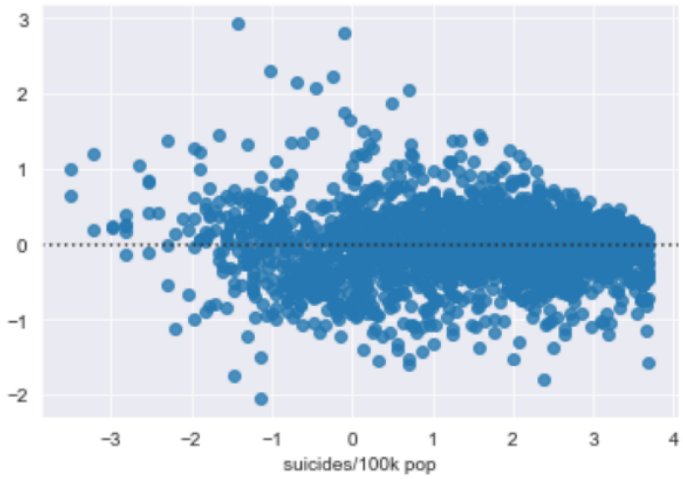

Figure 8.20

We also give the residual plot of Random Forest Model in Figure 8.20. If the model fits well, the residuals should change up and down 0, and the changes will be in a fixed range, and there will be no increase or decrease in the range of changes. From the figure, we can see the residuals change up and down 0, and the changes are almost in a fixed range, which shows the model fits well. Combined with the MSE and score of the model, the performance of this model is fairly good

Next, we plot the importance of every fields. Based on the output in Figure 8.21. From the figure, we can know, in random forest model, the field of age is the most important field which can influence suicide rate in this model and the importance of country, gdp_for_year, gdp_for_capita and sex followed close behind.

```
In [376]: frame = pd.DataFrame({"fields":inputdata.columns, "importances":model.feature_importances_})

In [377]: frame.sort_values(by='importances', ascending=False)
Out[377]:
              fields  importances
3                age     0.316990
0            country     0.201751
4   gdp_for_year ($)     0.129884
5  gdp_per_capita ($)     0.121429
2                sex     0.108581
1               year     0.068536
7       old_or_young     0.027139
6         generation     0.025690
```

Figure 8.21

We have plotted the relationship between age, gender, country and suicide rate in last step. We combined the five figures in one which has shown in Figure 8.22.(The figure may not clear, if you want to see clearer, please go to

chapter8.2.2)

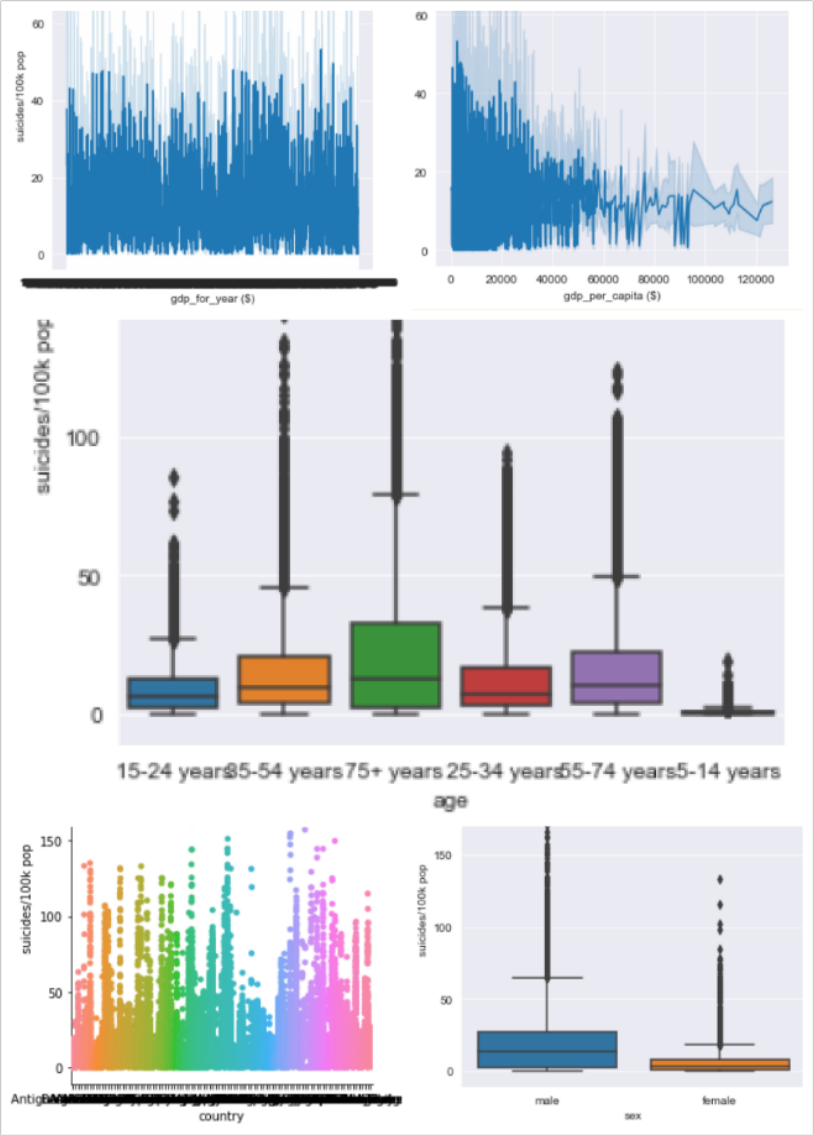

Figure 8.22

From the relationship between age and suicide rate, we can clearly see that the suicide rate of young people who is less than 14-year-old is very low, while the suicide rate of the elderly over 55 years old is relatively high.

From the relationship between gdp_for_year, gdp_for_capita and suicide rate, we can see the level of gdp has an impact on the suicide rate, which is more obvious in the relationship between gdp per capita and suicide rate. The lower the per capita gdp, the higher the suicide rate.

From the relationship between country and suicide rate, we can clearly see that that suicide rates vary greatly in different countries.

From the relationship between sex and suicide rate, we can see that the suicide rate of men was significantly higher than that of women.

Based on above patterns discovered, we think We believe that older men with lower GDP have the highest suicide rate, especially in some countries with high suicide rate. We plot the boxplot of age, gender and suicide rate in Figure 8.23, which proved our guess. The suicide rate of male elderly is relatively high.

```
In [409]: sns.boxplot(x="age", y="suicides/100k pop", hue="sex", data=datas);
```

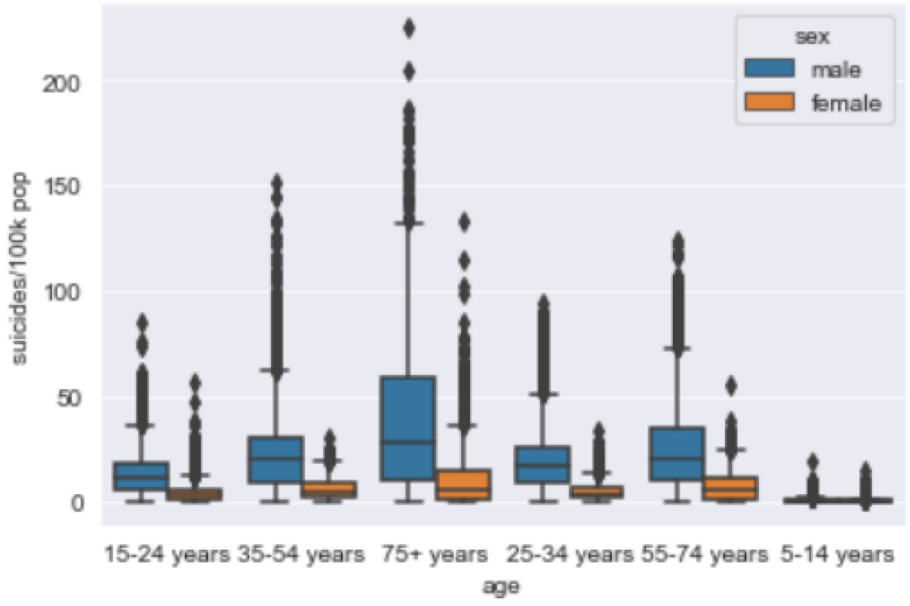

Figure 8.23

8.4 Assess and evaluate results, models, and patterns

- Assess and evaluate results

  We have discussed the result in the previous steps and we concluded Age, country, gdp and gender can have a great influence on suicide rates.

  - The higher the age, the higher the suicide rate.
  - Suicide rates vary significantly in different countries.
  - The lower GDP, the higher suicide rate.
  - The suicide rate of men is higher than that of women.

  Therefore, we should give advice to relevant agencies concerning the lives and mental health of male and elderly people, and remind relevant agencies of several countries where suicide rates have increased, and suggest that these countries should strengthen national psychological development to prevent the loss of lives.

- Assessment for Prediction of model

  In order to view the model forecast results, we print the describe of predicted data in Figure 8.24. The describe of the data shows that the dispersion of data is small and there is almost no extreme value. The quality of the predicted data seems good, but it can not mean the Prediction of model is good.

```
In [410]: data_prediction=pd.DataFrame({'predict_suicide_r':predict_rf})
     ...:
     ...: data_prediction.describe().T
Out[410]:
                  count      mean       std       min       25%       50%  \
predict_suicide_r  2886.0  1.706483  1.283324 -2.847572  1.012692  1.962258

                       75%      max
predict_suicide_r  2.713841  3.60009
```

Figure 8.24

  We have known the performance of Random Forest model is the best in the three models. The mean absolute error of it is 0.201 which is lower than that of the other models and the score of it is 0.893 which means Random Forest model has nearly 90% accuracy in prediction. The number is very high, so the accuracy of Random Forest model is high. Therefore, we can use Random Forest model to predict the trend of suicide rate.

- Assess and evaluate of patterns

The importance fields of Random Forest model shows that age, country,gdp and sex are important factors which can affect the suicide rate. Therefore, we should advice the relevant agencies to pay attention of these features in order to prevent the loss of lives.

8.5 Iterate prior steps (1 - 7) as required

8.5.1 Iteration for Step 1 - Business Understanding

For this step, we need to find a topic which is worth to explore. We selected the study of suicide. Suicide has become one of the top ten causes of death in humans, so this topic is worth to explore. We set the objectives and success criteria for this study in step-1. In this step, we should note that we need to connect the data mining target to the business target.

8.5.2 Iteration for Step 2 - Data Understanding

For this step, we collected the data from the open-source website Kaggle and started to explore the data. We checked the format of fields in the dataset and found some fields which are meaningless for our project. We may remove them in the next steps. Then, we visualized the data and verify the quality of the data. We found the dataset had some missing value, outliers and extreme values, which we need to handle in the next stage.

8.5.3 Iteration for Step 3 - Data Preparation

In this step, we integrate two csv files into one and select the data. We replaced the missing values by the mean number and we discarded the outliers and extremes value issues. We also grouped the population by age for better implementation of the study. Then, we formatted the data, because in some model, the input value should be numeric such as regression model. We changed the format of sex, age, generation,suicide population and country fields to be numeric.

8.5.4 Iteration for Step 4 - Data Transformation

For this step, we reduced the fields("HDI for year") which we do not need first. Then we found the target value(suicides/100k pop) 's distribution is not normal distribution, which has a bad influence in some model, so we decided to transform the target data. We decided to log it. After that, we found the distribution of data after log is near normal distribution, which can help us to fit a linear regression model later.

8.5.5 Data Mining Methods Selection

In this step, we choose regression methods as our data mining method in this research, because the type of response variable used by classification method is category and our target value(suicides/100k pop) is continuous in

our dataset. After discussion, We decided to use regression momdel in this research.

8.5.6 Iteration for Step 6 - Data Mining Algorithms Selection

For this step, we discuss the advantages and disadvantages of several data mining methods combined with data mining objectives. At last we choosed three model to fit our data. They are Linear Regression Model, Random Forest model and KNN Regression Model. Then we set algorithm parameters for these selected model(s). In Linear Regression Model and KNN Regression Model, most of our choices are default settings because they are all simple model, we do not need special settings in this project. In Random Forest model, we set the maximum depth of the tree as 15, because if we set it as a large number, the model will be overfit and if we set it as a small number, the model will be underfit. We think 15 is a suitable number. We also set the number of trees in the forest as 10, because the more trees there are, the higher the accuracy and the more treesthere are, the more time it takes. We think 10 is a suitable number.

8.5.7 IIteration for Step 7- Data Mining

In this step, we split the data into training data(80%) and test data(20%). Then, we used the split data to fit model with the different three models which we selected in the previous steps. Then, we listed the output and analysis of these model. The output of model include the mse and score of the different three models. We also draw the residual plot in order to better analyze the quality of the model. After comparing the performance of the three models, we got the mining result, patterns. At last of this step, we compared the performance of the three models and made a conclusion for the mining pattern.

9. Disclaimer

I acknowledge that the submitted work is my own original work in accordance with the University of Auckland guidelines and policies on academic integrity and copyright. (See: https://www.auckland.ac.nz/en/students/forms-policies- and-guidelines/student-policies-and-guidelines/academic-integrity-copyright.html.).

I also acknowledge that I have appropriate permission to use the data that I have utilised in this project. (For example, if the data belongs to an organisation and the data has not been published in the public domain then the data must be approved by the rights holder.) This includes permission to upload the data file to Canvas. The University of Auckland bears no responsibility for the student's misuse of data.

References

[1] IBM Knowledge Center-Neural Net node. 2020.

[2] IBM Knowledge Center-Regression model. 2020.

[3] L. Breiman. "random forests,". *Machine Learning*, pages 5–32, 2001.