| Team Number: | 2202547 |
|---|---|
| Problem Chosen: | E |

2023 APMCM summary sheet

This article gives a scientific development strategy of the nuclear weapon of the world to prevent the potential hazards of nuclear weapons to the earth and the environment we live.

To predict what the world's distribution of nuclear weapons might be in the next 100 years, we work out a logical strategy. **For problem 2(a)**, firstly, to reduce computational complexity, We **select potential countries** by status(1, or 2 but not 3 now). Then we choose **two indicator**(GDP and military expenditure) to represent a country's economic and military level, which is highly related to nuclear weapons manufacture. Thirdly, we train a **logistic regression model** to set economic and military criteria for possession of nuclear weapons. Fourthly, we train a **GRU model** to predict the GDP and military expenditure of potential countries in the next 100 years and judge whether and when they can have nuclear weapons associated with **logistic regression model**. **For problem 2(b)**, we first divide countries into countries with nuclear weapons now and potential countries. Then we use **Kaye model** and **Saperstein model** in **game theory** to judge whether each of two of countries with nuclear weapons in the **religion MAD**, which indicates **nuclear deterrence works**. Then we train a **polynomial regression model** with GDP and military expenditure of countries who have had nuclear weapons and use the model to predict number of nuclear weapons produced by potential countries in the next 100 years.

**For problem 3(a)**, we first calculate maximum explosion radius we have. We **subdivide** the earth's surface into n **equilateral triangles meshes**, and then we use **optimal-covering algorithm** to calculate the minimum number of circle(radius equals to explosion radius). We find the minimum value by writing a **algorithm** cover equilateral triangles with the **least coincident area** (the least number of circles). **For problem 3(b)**, we calculate the number of existing nuclear weapons and compare with 3(b) to judge. **For problem 3(c)**, we use **population density map** together with **pixel counting algorithm** in residential sites to calculate living space. Then we can calculate the number of nuclear weapons to destroy human living environment by scaling down. **For problem 4**, we write an non-technical article to U.N. and explain our findings and suggestions in five aspects.

**Keywords:** Saperstein model game theory optimal-covering algorithm earth surface mesh division algorithm logistic regression model polynomial regression model GRU model pixel-counting algorithm

# Contents

# I. Introduction

## 1.1 Background and Significance

Nuclear weapons refer to huge lethal weapons related to nuclear reaction, including hydrogen bombs, atomic bombs, neutron bombs, etc. Nuclear weapons are one of the most powerful weapons ever developed by human beings, and they often remind people of the scene of destroying heaven and earth. The instantaneous explosion temperature of an atomic bomb can reach tens of millions of degrees. The explosive yield of the atomic bomb is about tens of thousands to hundreds of thousands of tons of TNT equivalent. The explosion of an atomic bomb and the area of its radiation can destroy a city.

Facing such a powerful weapon, how to use it scientifically and reasonably has naturally become an important topic. We urgently need to propose a more reasonable and effective nuclear weapons development strategy through calculating the development trend of nuclear weapons in the future and analyzing the probable damage it may cause to earth.

## 1.2 Purpose

The purpose of this article is to establish a mathematical model about the development of nuclear weapons in different countries and use the model to calculate the development trend of nuclear weapons in the future and the probable damage it may cause to earth. With these mathematical results, we can work out a sensible global strategy for developing nuclear weapons.

# II. The Description of the Problem

## 2.1 Problems need to solve

**1)Basic data analysis**

Problem I requires us to be familiar with and basically analyze the data given in question E. We use python and excel to extract and classify the data we need from the dataset and do some simple calculation like **summing** or **sorting**. We also use matplotlib in python to do some **visualization** to make the result clearer and intuitive.

**2)Predict the number of countries and nuclear weapons**

**Problem 2(a)**

Problem 2(a) requires us to build a mathematical model to predict countries with nuclear weapons in the next 100 years. We first select some **main factors** related to nuclear weapons manufacturing,and then we use **logistic regression** model to predict the **criteria** of these factors. Then we use **GRU model** to predict what these factors will be like for potential countries. By **comparing** prediction results with criteria, we can obtain whether the potential country can possess nuclear weapons.

**Problem 2(b)**

Problem 2(b) requires us to total number, change trend of nuclear weapons in the next 100 years and calculate the number of nuclear weapons for each country.

We found that when a country **initially owned nuclear weapons**, the number of nuclear weapons would go through a rapid rise and then decline process. For a country that **has owned nuclear bombs for many years**, we observed that their number of nuclear bombs tended to be stable. So we divide countries into **two categories**: countries that already have nuclear bombs and potential countries.

We first use **Saperstein model** in **game theory** to confirm that the number of countries who have had nuclear weapons will be stable(or minor fluctuation). For **potential countries**, Our prediction of the number of nuclear weapons of potential countries in the second question still works in the third question. However, **logistic regression model** can only do the second classification, so in this case it only judges when a country can have nuclear bombs, but it **cannot** estimate the number of nuclear bombs. So for problem 2(b), we introduce **polynomial regression**. establish a mathematical model to predict the number of nuclear weapons and predict the change trend of the number of nuclear weapons in the next 100 years, the total number of nuclear weapons in 2123, and the number of nuclear weapons in each country.

**3) Protect our planet**

**Problem 3(a)**

As required in Problem 3(a), a mathematical model is to be established for the detonation position of nuclear weapons, and the number of nuclear bombs are required at least to destroy the earth.

To get the smallest number ,the **"Big Ivan"** ,the most powerful nuclear bomb known in the world at present, is applied to the model of destroying the Earth.After calculation,the damage radius of a bomb is known.

After that,since it's troublesome to determine the optimal covering of a sphere with equal circles, **dividing the earth surface into small equilateral triangular meshes** is a good way.

Finally, we use the **optimal covering algorithm** by *Kari J. Nurmela* to find the covering with least overlapping area, which can also output the arrangements of the circles,i.e. the centers.In that case, we can get the detonation position of nuclear weapons.After multiplying the results, we also get the least number of nuclear bombs to destroy the earth.

**Problem 3(b)**

In this problem, in order to find out if the nuclear weapons we have now can destroy the earth, we calculate the sum of nuclear weapons currently we have, and compare it with the number we obtain from problem 3(a).

**4) Write a non-technical article to the U.N.**

Problem 4 requires us to write a non-technical article to the United Nations to explain our finding and put forward some proposals. We divide our article into five parts, including power, growing trend, hazards, nuclear deterrence and our proposal. In this part, we explain the results we get in common language, and put forward some ideas on stopping endless growing of nuclear weapons and how to take full use of nuclear weapons' advantages.

## 2.2 Problem Analysis

### 2.2.1 *Problem I*

**Basic analysis of data**

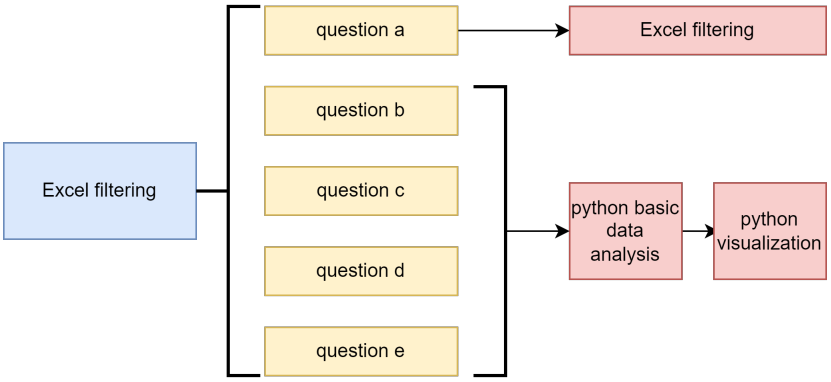



**Figure 1 basic analysis flow chart**

### 2.2.2 *Problem II*

**a) Predict countries with nuclear weapons in the next 100 years**

1) Choosing potential countries

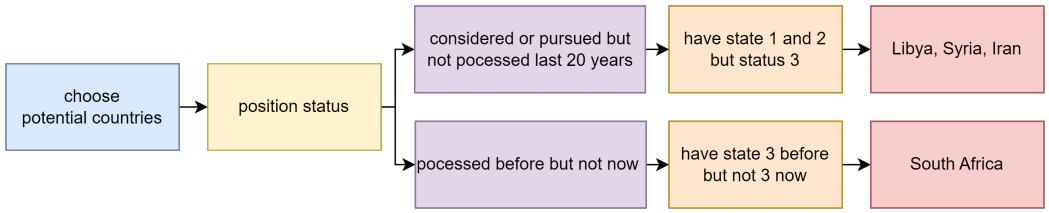



**Figure 2 2(a) choose country flow chart**

2) Choosing evaluation indicator

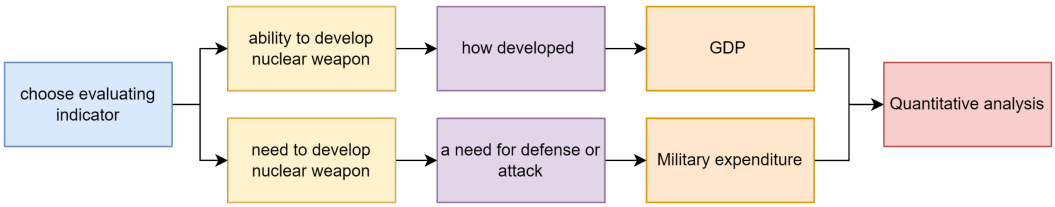



**Figure 3 2(a) choose evaluation indicator flow chart**

3) Predict factor(GDP and military expenditure)

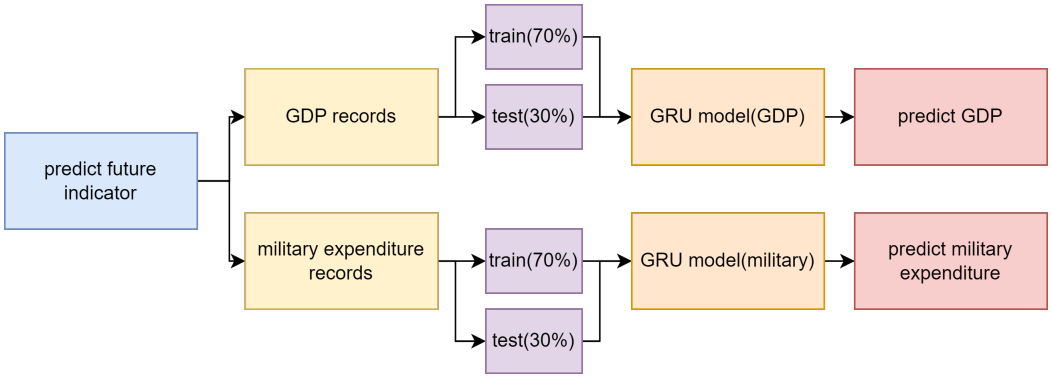



**Figure 4 2(a) use GRU to predict factors flow chart**

4) classify countries with nuclear weapons using logistic regression

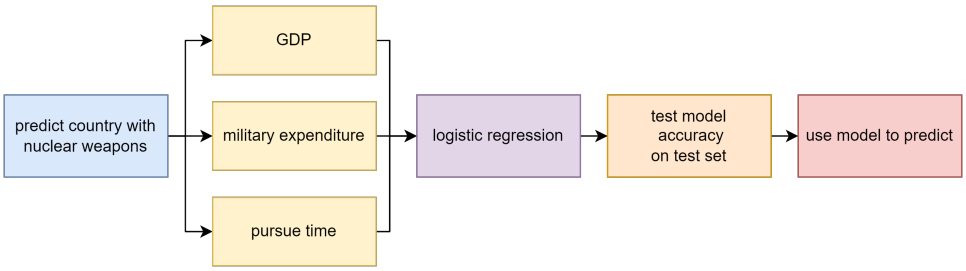



**Figure 5 2(a) use logistic regression to classify countries with nuclear weapons**

**b) Predict countries with nuclear weapons in the next 100 year**

1) use Saperstein model to determine countries' military state

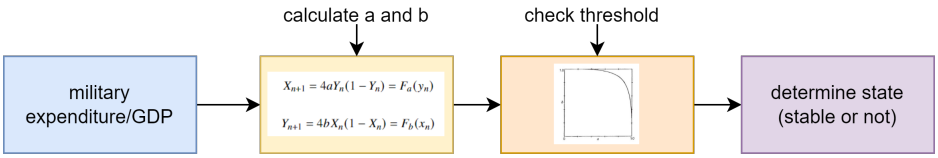



**Figure 6 2(b) use Saperstein model to determine state**

2) use multiple linear regression to predict countries' nuclear weapons number

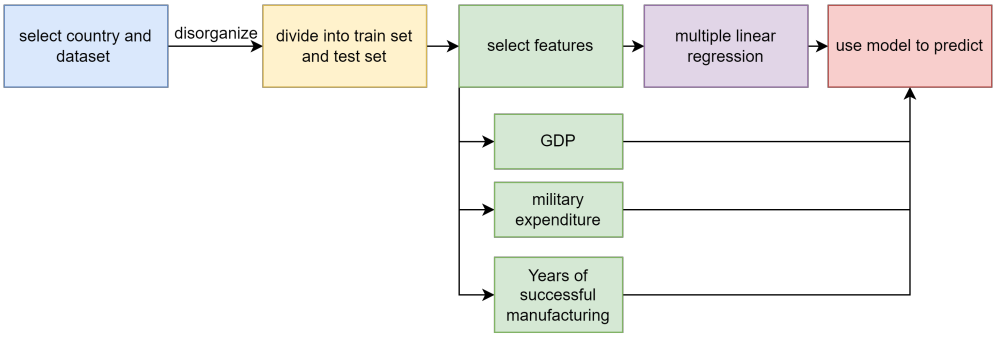



**Figure 7 2(b) use multiple linear regression to predict nuclear weapons number**

### 2.2.3 *Problem III*

a) calculate position of nuclear weapons, and the number of nuclear bombs required at least to destroy the earth

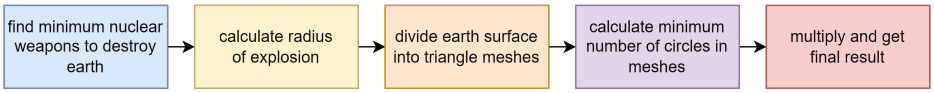



**Figure 8 3(a) calculate the minimum number of nuclear weapons to destroy earth**

b) check whether current nuclear weapons can destroy earth or not

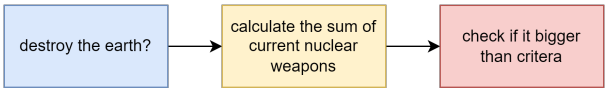



**Figure 9 3(b) check if current nuclear weapons can destroy the earth**

c) calculate limited number of nuclear weapons to protect earth

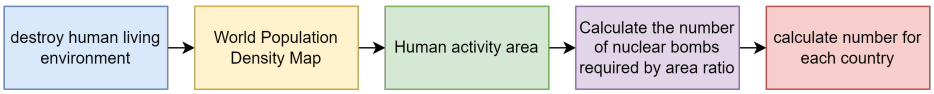



**Figure 10 3(b) calculate number of nuclear weapons to destroy human environment**

### 2.2.4 *Problem IV*

write a non-technical article to the United Nations

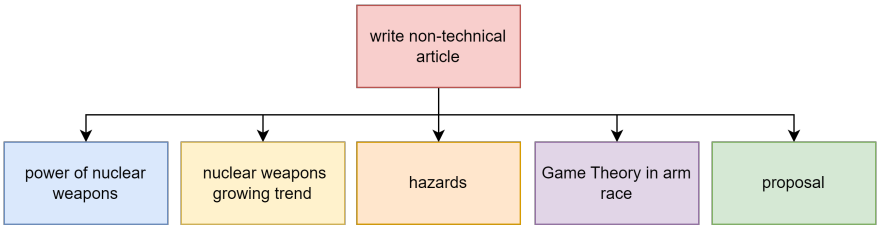



**Figure 11 4 write article to UN**

# III. Models

## 3.1 Terms, Definitions and Symbols

| Symbol | Definition |
|---|---|
| $GDP_i$ | the total gdp of a country in year i |
| $ME_i$ | the total military expenditure of a country in year i |
| $YAC_i$ | the year after a country started considering nuclear weapons |
| $YAP_i$ | the year after a country possessing nuclear weapons |
| $R^2$ | the Coefficient of Determination |
| $y_i$ | the label of the $i$th sample |
| $y_{ipred}$ | the predicted label of the $i$th sample |
| $S_i$ | the stockpile of a country in year i |
| $X_n$ | Proportion of military expenditure to GDP in the nth year of country X |
| $Y_n$ | Proportion of military expenditure to GDP in the nth year of country Y |
| $(a, b)$ | parameters in Saperstein model |
| $OL_n$ | area of n circles / area of a triangle for n circles covering a equilateral triangle |

## 3.2 Assumptions

1) We don't consider war and other emergencies.
2) The country's economic and military expenditure development is not sudden. The economic and expenditure development of the next year is based on the economic changes of the previous n years.
3) We only consider economic(GDP) and military(military expenditure) in predicting the number of nuclear weapons, we discard other factors like religion and geographical position.
4) We believe that the economic factor(GDP) and military factor(military expenditure) when potential countries invent nuclear weapons are similar to the economic and military situation of the countries that have nuclear weapons when they invented nuclear weapons, so we take GDP and military expenditure of the countries that have nuclear weapons as training set.
5)We assume that countries that are developing or have ideas to develop will still maintain the status of developing or having ideas to develop in the next 100 years.
6) When we divide the sphere into 20480 facets, we ignore the difference between the surface area of the sphere and that of 20480 facets.
7) We assume "tsar bomb" mentioned in the "How Many Nuclear Bombs can Destroy the Earth?" is the most powerful nuclear bomb with the largest explosion range.
8) When selecting the human living area, we chose the land area with a population density greater than $30/qskm$, ignoring the polar regions, islands, deserts and other regions with a population density less than $30/qskm$.

## 3.3 Model Establishment and Solution

### 3.3.1 *Problem 1*

a)Which countries have ever possessed nuclear weapons?

With reference to the given data, we find that United States, Russia, United Kingdom, France, China, India, Pakistan, Israel and North Korea have possessed nuclear weapons.

b) Which country has the largest reduction or increase in its nuclear weapons stockpiles in the last 20 years?

After processing the given data, we can see in the graph below that United States has the largest reduction in its nuclear weapons stockpiles in the last 20 years, which amounts to 6749 stockpiles.

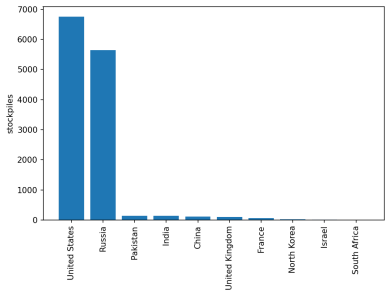


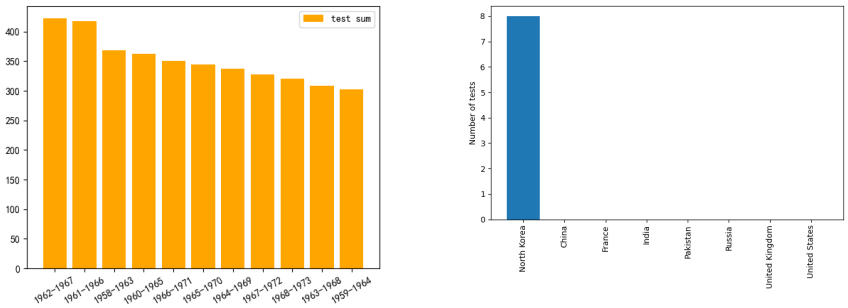

**Figure 12 histogram of change in nuclear weapons stockpiles in the last 20 years**

c) During which five years did nuclear weapon tests occur the most?

using the ”test.xlsx” table from ”2022_APMCM_E_Data.xlsx”, we calculate the sum of nuclear tests for five consecutive years and sort them in descending order.We extract the top ten records and draw the histogram as follows. we find that the most nuclear weapon test occurs during 1962 and 1967, which reaches to 422.

d) Which country has been the most active in nuclear weapons research in the last 10 years?

After processing the given data, we can see in the graph below that North Korea has been the most active in nuclear weapons research in the last 10 years, because it has conducted the most nuclear tests, which amounts to 8 times.




(a) histogram of nuclear test sum for every five years (b) histogram of nuclear test sum in the last 10 years

**Figure 13 Problem1(c) and Problem1(d)**

e) Which country has made the fastest transition from ”not considering nuclear weapons” to ”possessing nuclear weapons”?

According to the given data,the United States is the fastest to transfer from ”not considering nuclear weapons” to ”possessing nuclear weapons”, which took 7 years from 1938 to 1945.

| Country | United States | | Russia | | China | | France | | India | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0/3 | 1938 | 1945 | 1941 | 1949 | 1951 | 1964 | 1944 | 1960 | 1947 | 1987 |
| Transition year | 7 | | 8 | | 13 | | 16 | | 40 | |
| Country | United Kingdom | | Pakistan | | Israel | | North Korea | | | |
| 0/3 | 1939 | 1952 | 1971 | 1987 | 1947 | 1987 | 1961 | 2006 | | |
| Transition year | 13 | | 16 | | 19 | | 45 | | | |

**Table 1 Table of transition years**

### 3.3.2 *Problem 2(a)*

**I data collection, selection and process**

1) Choosing potential countries

Firstly, we exclude those countries that do not consider producing nuclear weapons in the latest 20 years. After filtering, four countries,Libya ,Syria, South Africa and Iran,are likely to possess nuclear weapons in the next 100 years.

2) Select factors highly related to nuclear development

The factors that determine whether a country can possess nuclear weapons are mainly divided into **two aspects**: firstly, the ability to develop nuclear weapons, and second, the need to develop nuclear weapons.We believe that the development of nuclear weapons is closely related to the **country's economic situation**. Only a country that is sufficiently developed and has sufficient financial resources can promote the development of nuclear weapons. So we first choose **GDP** to represent the economic situation of a country. Secondly, when a country has the ability to develop nuclear weapons, another factor is whether it has the **need** to develop nuclear weapons to defend itself or attack other countries. We have selected a **country's military investment** as a quantitative indicator of national demand. The higher a country's military investment is, the more it needs to develop nuclear weapons.

3) Missing value process

After determining the factors to use, we collected these required data from information sources like world bank and other authoritative sources. However, the problem of data missing appeared. The world bank only provides data on GDP and military expenditure from 1960 to 2021, which means we need to complete the data from 1945 to 1959. We used the fitting tool in matlab to fit the real data with **exponential function** and **Gaussian function** to complete the missing data.

**II train and test a GRU model**

The following is the flow chart about how we train and test a GRU model and use it to predict GDP and military expenditure of potential countries.

We first separate the data set into train set(70 percent of data) and test set(30 percent of data). After that, we build the basic code of GRU model and set an initial value for each parameter. Then we train the model and test it.Then we **modulate the parameters** of the model by the results obtained

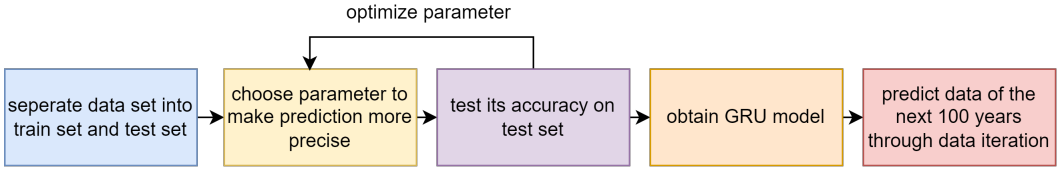
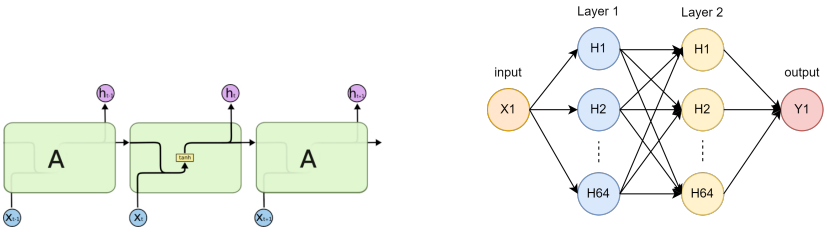



**Figure 14 train and test a GRU model to predict GDP and ME**

on the test set. We repeat the above steps to obtain the **optimal parameters**. The final parameters we obtained are shown below.

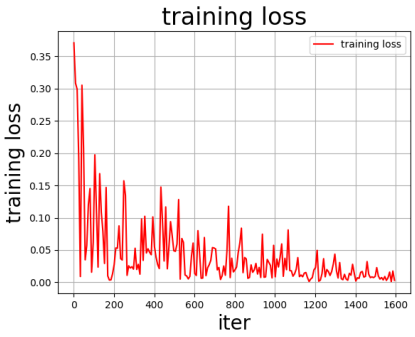
```
train_para = {
          'input_dim': 1,      'batch_size': 4,
          'hidden_dim': 64,    'dropout': 0.5,
          'num_layers': 2,     'epoch': 200,
          'output_dim': 1,     'backward': 9,
          'lr': 0.001,         'feature': 'Military'
}
```

We develop a **GRU model** for GDP and military expenditure of each potential country, and the pictures below are their training loss. We can see from the result that the training loss of our GRU model reduce to zero, which indicates that our GRU model can fit the data very well.




(a) working principle (b) GRU structure

**Figure 15 GRU model**




(a) train loss of Iran GDP

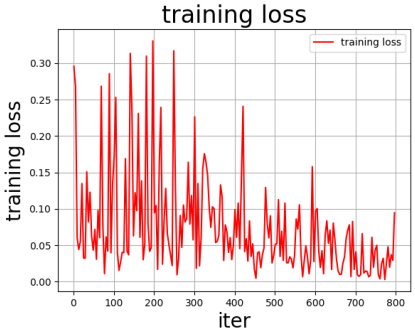

(b) train loss of Libya GDP


(c) train loss of Syria GDP

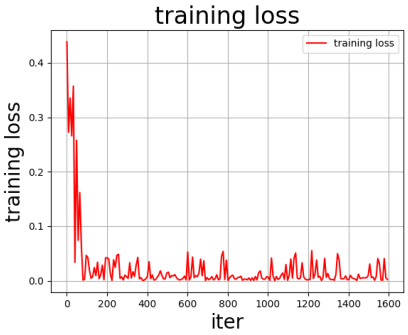

(d) train loss of South Africa GDP

**Figure 16 train loss of four potential countries' GDP**

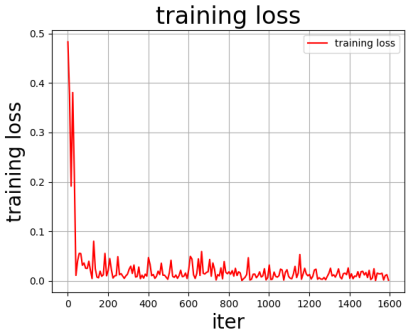

(a) train loss of Iran military expenditure




(b) train loss of Libya military expenditure




(c) train loss of Syria military expenditure




(d) train loss of South Africa military expenditure

**Figure 17 train loss of four potential countries' military expenditure**

Then we use the GRU model we trained to predict GDP and military expenditure in the next

100 years. The GRU neural network we trained cannot predict the data in the next 100 years at one time, So we choose to **iterate the data** in the following way described below.

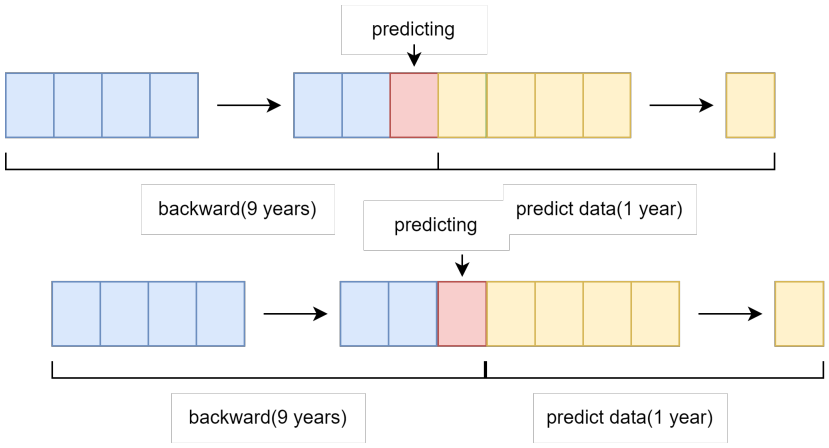
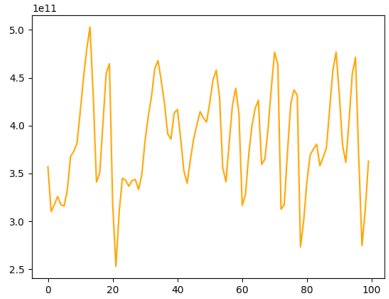


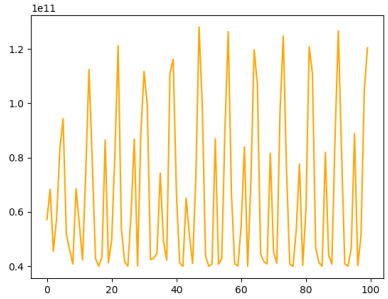

**Figure 18 predicting process using GRU**

We first input the last 9 real data(GDP and military expenditure) and we can get a prediction result through GRU model. Then we take last 8 data and the prediction result as our input. For example, We input data from 2010 to 2018 and we can predict data for 2019. After that, we input data from 2011 to 2019... We repeat this process until we get the data of the next 100 years.

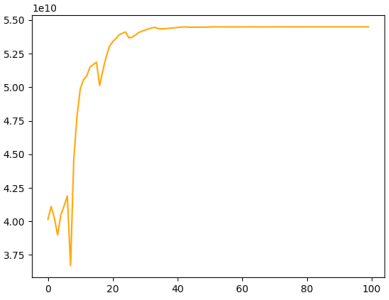
The following figures are our prediction results of potential counties' GDP.



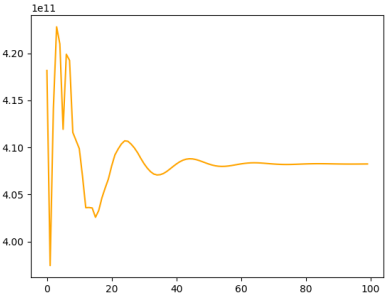

(a) prediction of Iran GDP




(b) prediction of Libya GDP



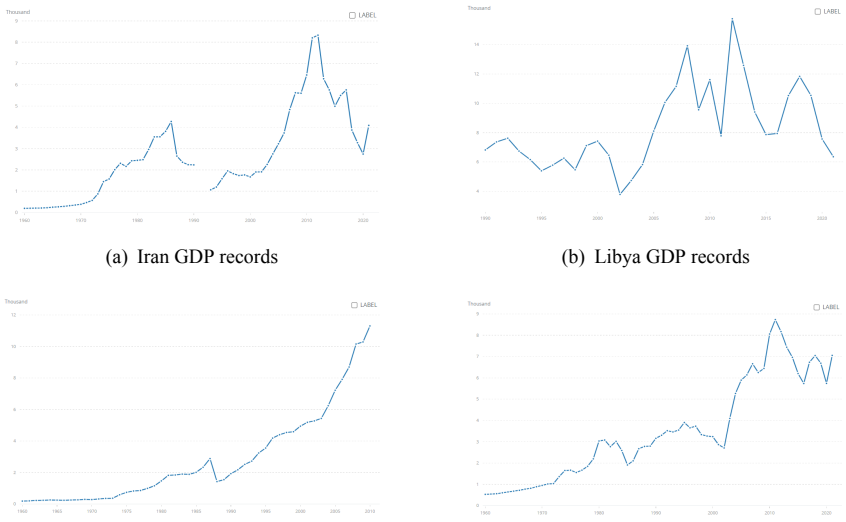

(c) prediction of Syria GDP




(d) prediction of South Africa GDP

**Figure 19 prediction of potential countries' GDP**

We also download GDP records of the four potential countries and draw the pictures below.



(a) Iran GDP records

(b) Libya GDP records

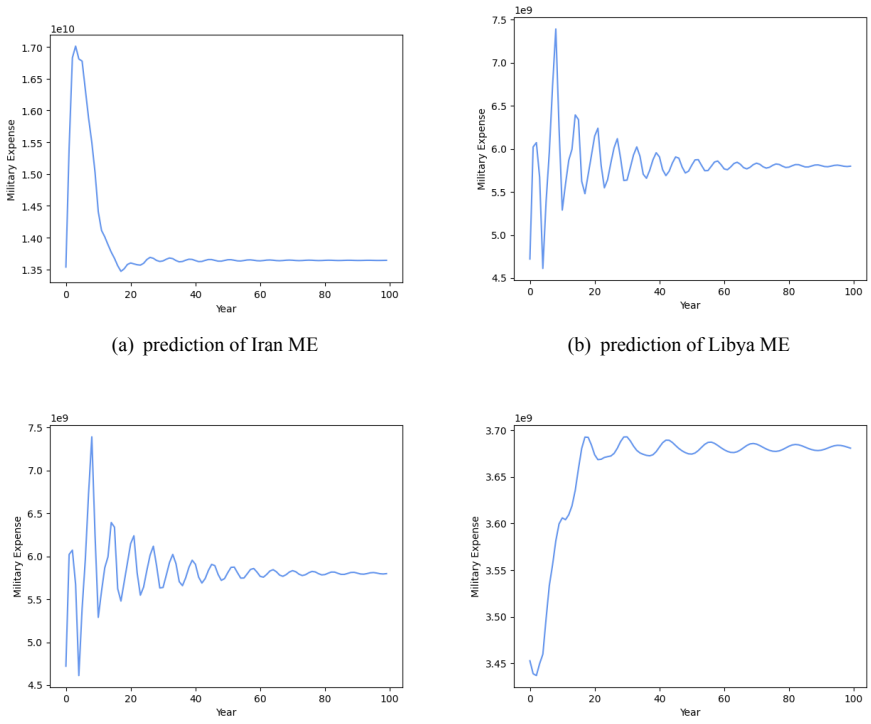
(c) Syria GDP records

(d) South Africa GDP records

**Figure 20 records of potential countries' GDP**

We compared the past GDP data with the predicted GDP data, and found that the GDP numerical changes have **similar cycles and trends**, indicating that our prediction is **highly reasonable**.

The followings below are our prediction results of potential counties' military expenditure.




(a) prediction of Iran ME

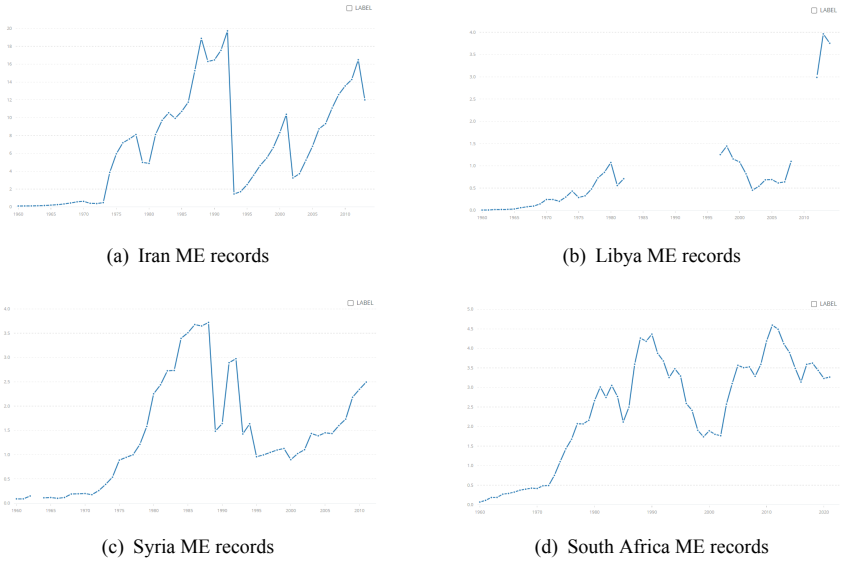
(b) prediction of Libya ME

(c) prediction of Syria ME

(d) prediction of South Africa ME

**Figure 21 prediction of potential countries' military expenditure**

We also download military expenditure records of the four potential countries and draw the pictures below.



(a) Iran ME records (b) Libya ME records

(c) Syria ME records (d) South Africa ME records

**Figure 22 records of potential countries' military expenditure**

We compared the past military expenditure data with the predicted military expenditure data, and found that the military expenditure numerical changes have similar cycles and trends.

I have to mention that, as we know, a country's military expenditure has much to do with the **political environment** in which the country is located. However, the political situation of a country is **prone to sudden changes**, which is difficult to predict with models. So we admit that the effect of predicting a country's military expenditure will have **some errors**, but it is **inevitable**.

**II train and test a Logistic Regression model**

To predict the **countries with nuclear weapons in the next 100 years**,a multi-factor Logistic Regression model is built, which can output the year when a country can possess nuclear weapons.

Since the probability of possessing nuclear weapons is closely related to the **GDP**, annual military expenditure($\boldsymbol{ME_i}$) and the year after the country considers nuclear weapons($\boldsymbol{YAC_i}$). To train the model, it takes the three factors of countries with nuclear weapons as training samples.

More specifically,we label sample data with 0 if a country doesn't possess nuclear weapons this year(the position equals to 0 or 1 or 2),while label sample data with 1 if the country already possesses nuclear weapons this year(the position equals to 3).In that case, the model can figure out the rule of transferring from "considering nuclear weapons" to "possessing nuclear weapons".

We apply the training sample data shown in the table 2.

| # | Column | Non-Null Count | Dtype |
|---|---|---|---|
| 0 | $GDP_i$ | 468 non-null | float64 |
| 1 | $ME_i$ | 468 non-null | float64 |
| 2 | $YAC_i$ | 468 non-null | int64 |
| 3 | label | 468 non-null | int64 |

**Table 2 Sample Data of training set of Logistic Regression model**

We separate the data set into train set(70 percent of data) and test set(30 percent of data). Following is the **training loss** of the Logistic Regression model,and the **accuracy rate** of the model is

0.9096774193548387.

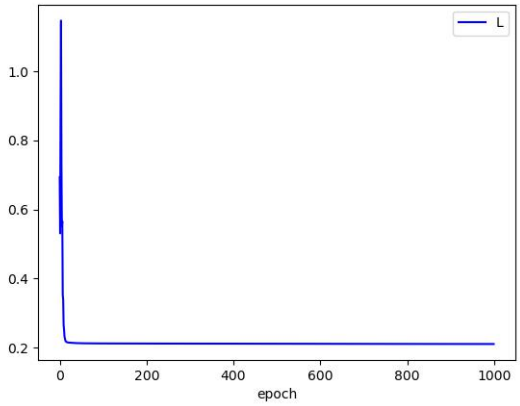



**Figure 23 training loss of Logistic Regression model**

Last,the model takes $\boldsymbol{GDP_i}$, $\boldsymbol{ME_i}$ and $\boldsymbol{YAC_i}$ in the next 100 years of the four alternative countries, Libya,Syria, South Africa and Iran, as input.It then outputs the year when a country can possess nuclear weapons.

Input data sample is shown in the table 3.

| # | Column | Non-Null Count | Dtype |
|---|---|---|---|
| 0 | Year | 100 non-null | int64 |
| 1 | $GDP_i$ | 100 non-null | int64 |
| 2 | $ME_i$ | 100 non-null | float64 |
| 3 | $YAC_i$ | 100 non-null | int64 |

**Table 3 Sample Data of input of Logistic Regression model**

Following table presents the **result of the model**.As is reflected in table 4,Iran is predicted to possess nuclear in 2046, Libya in 2069, South Africa in 2096 and Syria in 2125.

All in all, three countries,Iran, Libya and South Africa,are likely to possess nuclear weapons in the next 100 years.

| Country | Predicted Year | Country | Predicted Year |
|---|---|---|---|
| Iran | 2046 | South Africa | 2096 |
| Libya | 2069 | Syria | 2125 |

**Table 4 Output of Logistic Regression model**

### 3.3.3 *Problem 2(b)*

We divide the countries with weapons in the next 100 years into two categories, the countries that have had nuclear weapons now and the countries that will have nuclear weapons.

We believe these two categories will act differently in nuclear weapons manufacture and we calculate their nuclear weapons numbers in two different ways.

- For countries who do not have nuclear weapons now:

By observing the changes in the number of nuclear weapons of countries with nuclear weapons so far, we find that the trends of these countries are roughly the same. We have reason to believe that the new nuclear weapon countries will also fluctuate in a similar shape.

- For countries who do have nuclear weapons now:

We have no experience in predicting the future of countries that already possess nuclear weapons. However, we have found that the number of nuclear weapons of countries with nuclear weapons has stabilized this year. We use Saperstein model to calculate the military status of countries with nuclear weapons and judge whether they will remain stable.

**I use Saperstein model to judge status**

First, through Kaye model, we can determine the conditions for the balance of war.

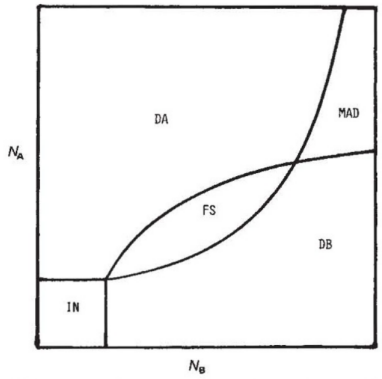



**Figure 24 Kaye model**

The **Kaye model** is described by a Cartesian plot in which the two orthogonal axes represent the number of warheads or launch vehicles available to each of the two sides. The inequalities are represented by different regions of the plot.For example, in the figure above, the number of launch vehicles possessed by side $A$, $N_A$, is plotted against $N_B$, the corresponding number for the opposing side $B$. In the region designated $DA$, $A$ is dominant; it can punish $B$ with a successful disarming first strike without fear of severe loss to itself. Similarly $B$ is dominant in region $DB$.**Region MAD** is the **stable region** of mutually assured destruction whereas $FS$ is the unstable deterrence region in which each side has the incentive to strike first. The **remaining region**, $IN$, represents that in which neither side has sufficient warheads to do conclusive damage to the other. Thus there are **stable** and **nonstable** areas which, topologically are simply connected and have simple boundary curves. Except for the boundary curves themselves, nearby points in the plot belong to the same region. Thus small uncertainties about the actual numbers need not lead to changes in the perception of stability or instability. In this sense, the static model also **permits practical predictions**.

Then we apply **Saperstein model** to further calculate countries' state(in stable or nonstable status).

**Saperstein model**

A bilateral arms race between two competing countries X and Y is assumed to occur in stages designated by an integer indicator n. The dependent variables $X_n$, $Y_n$ represent the fraction of the two nations' resources devoted to armaments: $0 \leq X_n, Y_n \leq 1$. A nation's armaments at the next

stage depend upon its perception of its opponent's armaments at the present stage;Saperstein model assume $X_{n+1}$ to be proportional to $Y_n$ and conversely.

But the perceived threat also depends upon how much of the opponent's resources remain potentially convertible to armaments at a future stage. If all of the opponent's substance is devoted to arms, there is no need to be concerned about a future increase in threat (assuming fixed total resources). Hence Saperstein model also assume $X_{n+1}$ to be proportional to $1 - Y_n$ and conversely. **Saperstein model** [3] thus consists of the pair of sequences of mappings:

$$X_{n+1} = 4aY_n(1 - Y_n) = F_a(y_n) \tag{1}$$

$$Y_{n+1} = 4bX_n(1 - X_n) = F_b(x_n) \tag{2}$$

Due to space limitation, we have omitted the calculation of critical values of a and b. The following figure is the relationship between a and b obtained by Saperstein model.

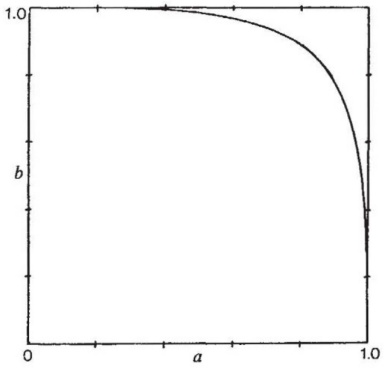



**Figure 25 parameter threshold**

When the (a, b) we calculate is below the line, it indicates that the country is in **stable status**; when the (a, b) we calculate is above the line, it indicates that the country is in **nonstable status** which can lead to change in the number of nuclear weapons.

We calculate parameter a and b for each two of countries which have nuclear weapons now and the below are our results.

Note: We can't find proportion of military expenditure to GDP of North Korea, considering North Korea produce few nuclear weapons these year, we discard North Korea in this part.

| Relaion | China | US | France | Israel | UK | India | Pakistan | Russia |
|---|---|---|---|---|---|---|---|---|
| China | | 0.13 | 0.24 | 0.09 | 0.22 | 0.18 | 0.11 | 0.12 |
| US | 0.55 | | 0.51 | 0.19 | 0.47 | 0.38 | 0.24 | 0.25 |
| France | 0.3 | 0.16 | | 0.11 | 0.26 | 0.21 | 0.13 | 0.14 |
| Israel | 0.83 | 0.42 | 0.77 | | 0.71 | 0.57 | 0.36 | 0.38 |
| UK | 0.33 | 0.17 | 0.31 | 0.11 | | 0.23 | 0.14 | 0.15 |
| India | 0.42 | 0.22 | 0.4 | 0.15 | 0.37 | | 0.18 | 0.2 |
| Pakistan | 0.59 | 0.3 | 0.55 | 0.2 | 0.51 | 0.41 | | 0.27 |
| Russia | 0.63 | 0.32 | 0.59 | 0.22 | 0.54 | 0.43 | 0.27 | |

**Table 5 (a,b) parameter we calculate by Saperstein model**

We can see from the table that all of (a, b) are below the line, which means **all countries** who have had nuclear weapons are in **stable status**. We can conclude from the result that the number of nuclear weapons will stay the same(or in small range fluctuation).

From the data in the above chart, we can see that the number of nuclear bombs in the above eight countries will stabilize at the current value (there may be small fluctuations). However, by observing the changes in the number of nuclear bombs in India and Pakistan in recent years, we find that the number of nuclear bombs in **India and Pakistan** is still **on the rise**.We believe that this is due to the fact that India and Pakistan are still compete with each other in the MAD (nuclear deterrence) region and **produce redundant nuclear weapons**. Therefore, we take India and Pakistan as **exceptions**, and also put them into the following **GRU and polynomial regression prediction system** to predict the number of nuclear weapons.

**II Train and test a polynomial regression model**

By observing the countries which has already had nuclear weapons, we can find from the graph below that the amount of stockpiles of these countries mostly increased at first, then went down to a steady value.

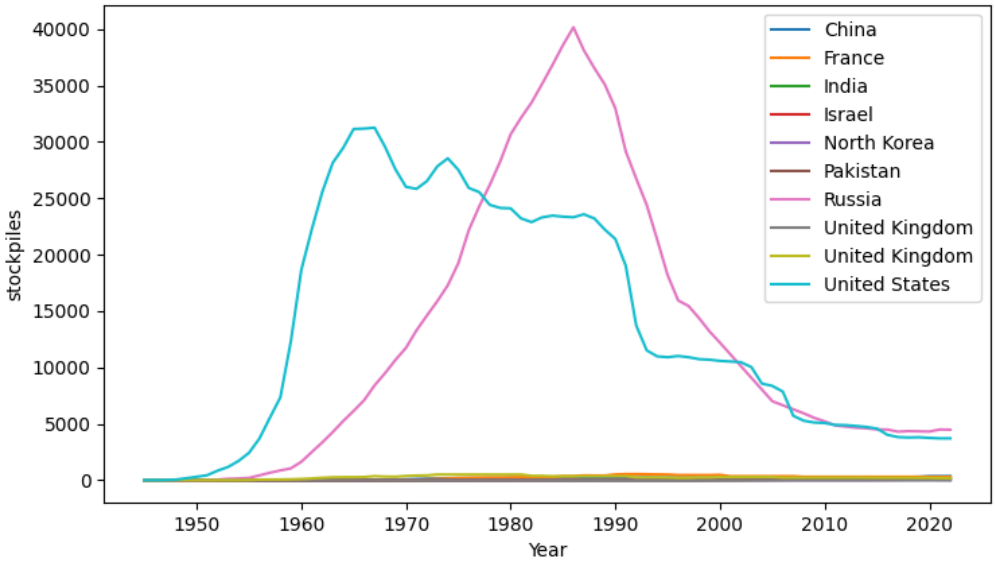



**Figure 26 Change of stockpiles of all countries**

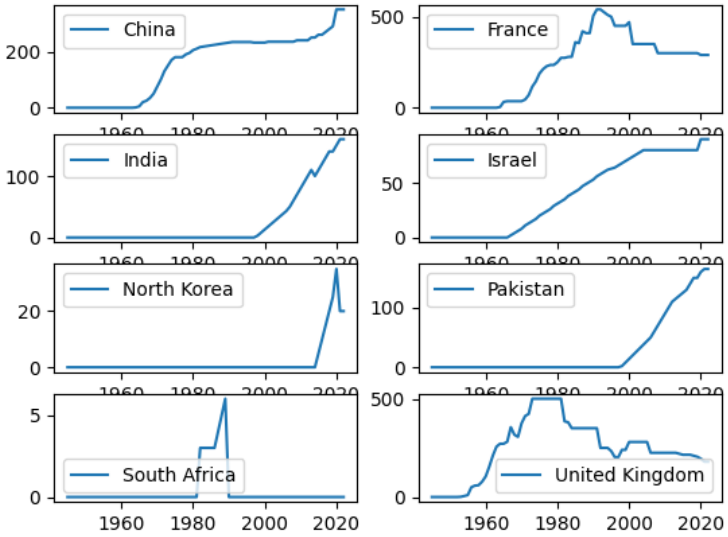



**Figure 27 Change of stockpiles of countries with small stockpiles**

Moreover, the difference of stockpiles between United States, Russia and other countries is impossible to neglect because of some well-known political and historical reasons. We don't think it reasonable to consider the United States, Russia and other countries side by side. We regard the United States, Russia as countries forced by arms race and others as countries driven by potential political threats in peacetime. Also, under the assumption that there won't be a world war in the next 100 years, we can infer that when it comes to the countries that will have nuclear weapons in the future, the number of their nuclear weapons will change in accordance with the historical pattern of countries driven by potential political threats in peacetime.

Under this premise, we introduce the multiple linear regression model. Particularly, we train and test the model with the data of China, France, India, Israel, Pakistan and United Kingdom from the year they begun producing nuclear weapons to 2022. We dismiss North Korea because it's difficult to find the GDP and military expenditure of North Korea. Also, we dismiss South Africa because although it once had nuclear weapons, it doesn't possess it now.

We use the sample data shown as the table below.

| # | Column | Non-Null Count | Dtype |
|---|---|---|---|
| 0 | $GDP_i$ | 320 non-null | float64 |
| 1 | $ME_i$ | 320 non-null | float64 |
| 2 | $YAP_i$ | 320 non-null | int64 |
| 3 | $S_i$ | 320 non-null | in64 |

**Table 6 Sample Data of Polynomial Regression**

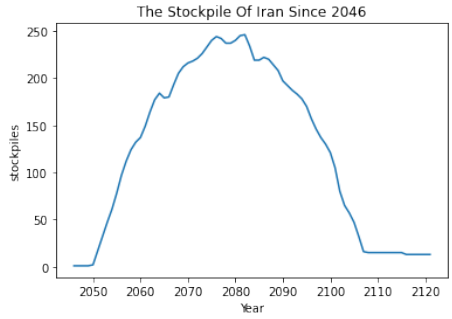
We divide the sample data set into training set(80%) and test set(20%). To evaluate the accuracy of the model, we introduce the coefficient of determination $R^2$.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - y_{ipred})^2}{\sum_{i=1}^{n}(y_{ipred} - \overline{y})^2} \tag{3}$$

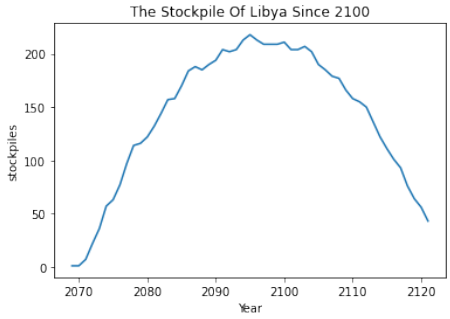
Through the test set, we can get that

$$R^2 = 0.6871 \tag{4}$$

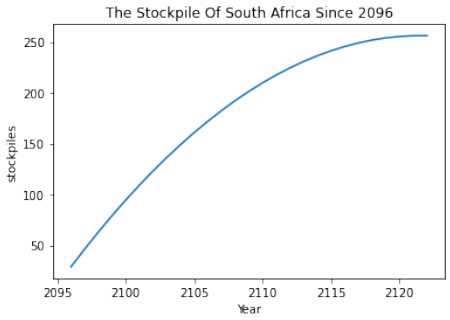
Then, we put the data of countries that were screened out by the previous logistic regression model into this polynomial regression model, which are Iran, Libya and South Africa, we can get the following result.




(a) Stockpiles of Iran Since 2046




(b) Stockpiles of Libya Since 2069



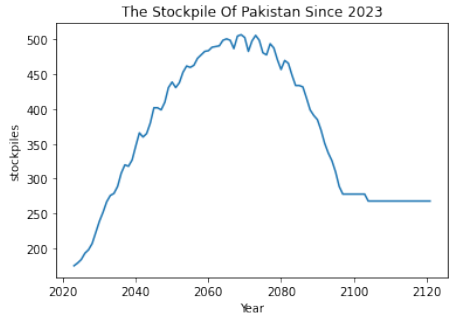

(c) Stockpiles of South Africa Since 2096

**Figure 28 Prediction of Potential Countries' Stockpiles**

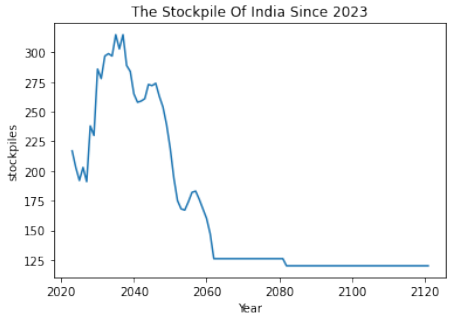
When it comes to the prediction of stockpiles of the countries already have nuclear weapons, we choose the data of the countries whose stockpiles have been stable for a long time to train the model. In this case, we can get

$$R^2 = 0.90593 \tag{5}$$

Similarly, we use the data predicted by the GRU model to obtain the stockpiles of Pankistan and India in the next 100 years. Here are the results.




(a) Stockpiles of Pakistan Since 2023




(b) Stockpiles of India Since 2023

**Figure 29 Prediction of Stockpiles of Pakistan and India**

As a conclusion, we can get the world's overall change of stockpiles in the next 100 years. From the line chart below, we can see the change trend of the number of nuclear weapons in the next 100 years is first increase, then decrease, and finally gradually goes down to a steady value.

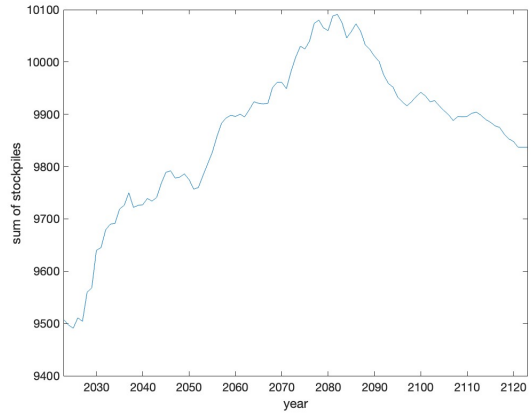



**Figure 30 Prediction of The World's stockpiles in the next 100 years**

The total number of nuclear weapons in 2123 is 9837, and the number of nuclear weapons in each country in 2123 is shown in the table below. More data about each country's stockpiles of each year is shown in an attachment file.

| Country | South Africa | Libya | Pakistan | India | UK | US |
|---|---|---|---|---|---|---|
| Stockpiles of 2123 | 278 | 43 | 268 | 120 | 180 | 3708 |
| Country | China | North Korea | Israel | Russia | France | Iran |
| Stockpiles of 2123 | 350 | 20 | 90 | 4477 | 290 | 13 |

**Table 7 The Number of Nuclear Weapons in Each Country in 2123**

### 3.3.4 *Problem 3(a)*

As required, an mathematical model is built to calculate **the number of nuclear bombs** required at least to destroy the earth. If nuclear weapons destroy the earth, it does not mean that they can blow the earth into pieces, but that the living environment of human beings and creatures on the earth has been destroyed.According to modern estimates, the surface area of the Earth is approximately 510 million square km($5.1 \times 10^8\ km^2$) or 196,900,000 square miles.

**I Calculate the Damage Radius of Nuclear Weapons**

The "Big Ivan" ,the most powerful nuclear bomb known in the world at present, is applied to the model of destroying the Earth.The damage power of the nuclear weapon basically comes from air blast, nuclear fireball, ionizing radiation and other long term radiation effects. Each of them can cause different degrees of damage, different areas of the damaging scope. In order to destroy the earth right after the nuclear explosion, we choose the air blast as the primary mode of destruction. The evaluation of blast wave is based on the over pressure caused by it, which is given in formula below.

$$\Delta p = 22.5 \left(\frac{m}{V}\right)^{0.72} \text{ bars} \tag{6}$$

where:

- m = (kilograms) net explosive mass calculated using all explosive materials and their relative effectiveness

- V = (cubic meters) volume of given area (primarily used to determine volume within an enclosed space)

Also, we find the overpressure data from the internet, which is shown as below. [4]

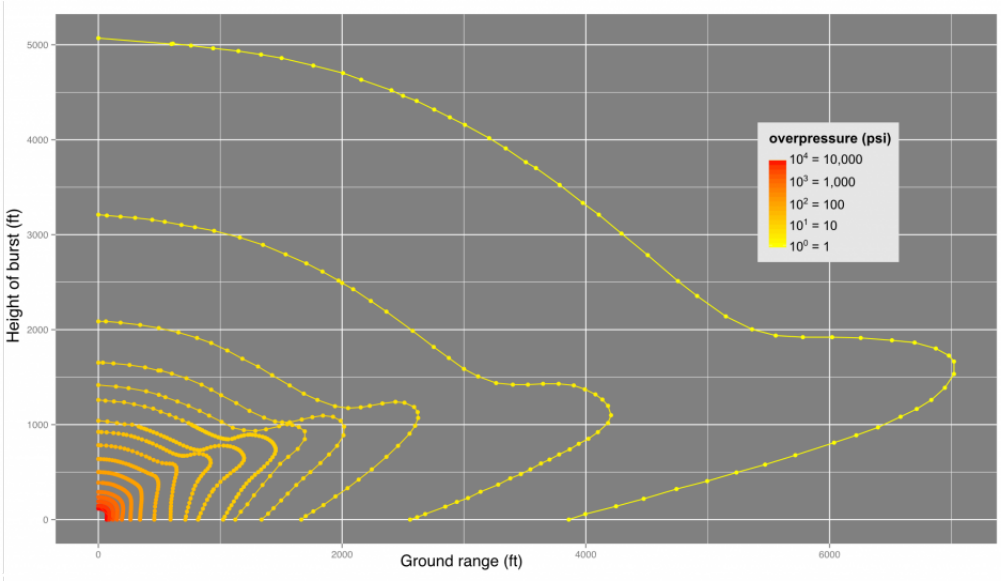



**Figure 31 Overpressure Data For 1-kiloton Burst on the Ground**

According to the given data, we estimate the destruction scope of the most powerful nuclear weapon over history – ”Big Ivan” as the table shown below.

| Effective Radius | Overpressure/psi | Effective Distance/km | Effective Area/km^2 |
|---|---|---|---|
| Air blast radius | 300 | 2.7 | 22.91 |
| Heavy blast damage radius | 20 | 8.03 | 202.39 |
| Moderate blast damage radius | 5 | 16.92 | 899.74 |

**Table 8 The Destruction Area of ”Big Ivan”**

In the air blast radius, which is at 300psi, everything will be destroyed completely. In the heavy blast damage radius, which is at 20psi, concrete buildings are severely damaged or demolished, fatalities approach 100%. In the moderate blast damage radius, which is at 5psi, most residential buildings collapse, injuries are universal, fatalities are widespread, which means in this radius, the damage is not complete. As a conclusion, we choose the moderate blast damage radius as our basic radius to cover the earth, because the overlap of coverage is inevitable, and the damage decreases as the radius increases. Under the choice of the moderate blast damage radius, overlapping coverage areas are the superposition of such relatively minor damage.

**II Find Optimal Coverings of Equilateral Triangle with Damage circles**

To destroy the earth with least nuclear bombs, it means **covering the earth surface with damage circles** and making **overlapping area** of the circles **as small as possible**.

| n | radius | norm. rad. | $OL_n$ | n | radius | norm. rad. | $OL_n$ |
|---|---|---|---|---|---|---|---|
| 1 | 0.577350 | 1 | 2.42 | 15 | 0.115470 | 1 | 1.46 |
| 3 | 0.288675 | 1 | 1.81 | 21 | 0.096225 | 1 | 1.41 |
| 6 | 0.192450 | 1 | 1.61 | 28 | 0.082479 | 1 | 1.38 |
| 10 | 0.144336 | 1 | 1.52 | 36 | 0.072169 | 1 | 1.51 |

**Table 9 Properties of the coverings with normalized radius = 1**

Since it's difficult and hardy to have a sphere covered by n equal circles with least overlapping area.Therefore, it is more suggested to divide the earth surface into small regions and determine the optimal covering of a small region. In that case, the objective is to determine **optimal coverings** with n equal circles of a small area such as **an equilateral triangle** rather than the whole earth sphere.

According to relevant research, some scientists have found the algorithm of determining good coverings of an an equilateral triangle with n equal circles so that the radius of the circles is as small as possible.With circles of fixed radius(16.1km), among coverings for different number of circles,selecting the the one with smallest overlapping area can solve our problem.

Kari J. presents a computational method to find good, optimal coverings of an equilateral triangle with up to 36 equal circles, which shows all the best known coverings for $1 \leq n \leq 36$. [2]

- The description of the optimal covering algorithm:The algorithm consists of two nested levels: on the inner level the uncovered area of the triangle is minimized by a local optimization routine while the radius of the circles is kept constant. The radius is adapted on the outer level to find a locally optimal covering. Good coverings are obtained by applying the algorithm repeatedly to random initial configurations.

As the paper shows,the smaller the normalized radius, the more efficient the cover.Moreover,the smallest normalized radius is 1 among all the optimal coverings($1 \leq n \leq 36$). Only when n = 1, 3, 6, 10, 15, 21, 28, 36,the normalized radius is equal to 1.Therefore, to get the number n with least overlapping area,overlapping rate(($OL_n$)) is calculated and compared among coverings for n = 1, 3, 6, 10, 15, 21, 28, 36.

Table 8 shows some data relative to the optimal coverings for n = 1, 3, 6, 10, 15, 21, 28, 36: the radius of the circles and the radius normalized with respect to coverings by triangular numbers of circles.The result is calculated under the condition that the length of each side of equilateral triangle is 1.

As is mirrored in the table 8, the overlapping rate for n = 28($OL_{28}$) is the smallest.Therefore, the optimal solution is to cover an with **28 circles**,and the **ratio of radius to the length of triangle** is **0.082479**.Since the damage radius is 16.1 km,the length of side of the equilateral triangle is approximately **204.90 km**.

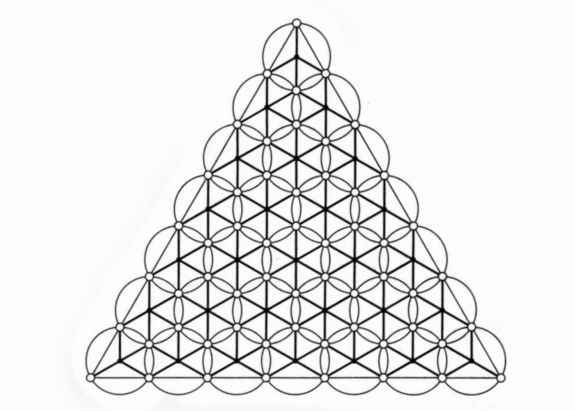


**Figure 32 Covering for n = 28**

## III Dividing Earth Surface with Equilateral Triangular Meshes

As discussed earlier, the earth surface is subdivided into small equilateral triangular meshes of which the side length is **204.90 km**.

The choice of spherical polygons used to tile the sphere consists of triangles, quadrilaterals, and hexagons (with a small mix of pentagons), but not all combinations result in a high quality mesh.**A regular icosahedron** produces **a superior mesh** compared to a tetrahedron or an octahedron. [1]In geometry, a regular icosahedron has five equilateral triangular faces meeting at each vertex.If the edge length of a regular icosahedron is $a$, the radius of a circumscribed sphere (one that touches the icosahedron at all vertices) is

$$r_u = \frac{a}{2}\sqrt{\varphi\sqrt{5}} = \frac{a}{4}\sqrt{10+2\sqrt{5}} = a\sin\frac{2\pi}{5} \approx 0.951\,056\,5163 \cdot a \tag{7}$$

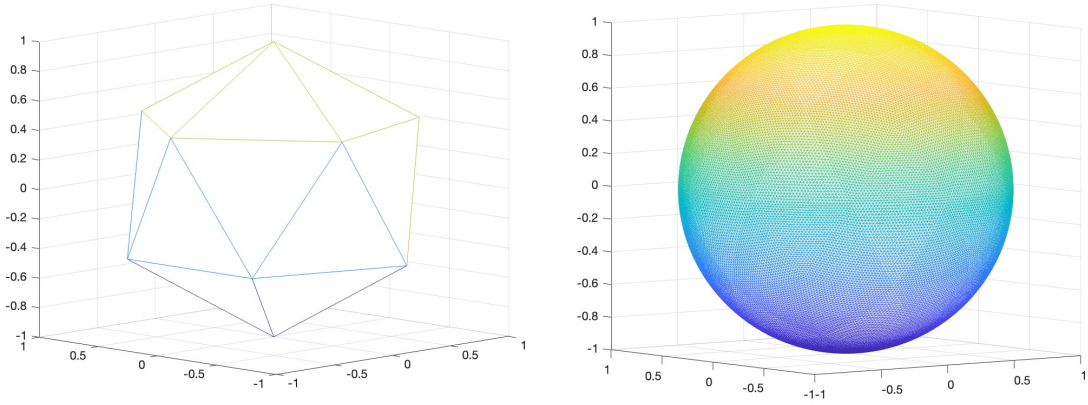
where $\varphi$ is the golden ratio.

The radius of the Earth(circumscribed sphere) is 6371km,so the edge length of inscribed regular icosahedron is approximately 6698.8658 km.


**Figure 33 Training Loss of Polynomial Regression**

As figure 34 implies,the subdivision of equilateral triangles is quite simple and the triangles can be subdivided continuously.

By continuously subdividing the faces of the regular icosahedron, it turns into **a polyhedron with 20480 equilateral triangular meshes** while its edge length is close to the number we need,i.e. 204.90 km.The area of the polyhedron is 80 % of the area of the earth($5.1 \times 10^8\ km^2$).But there is overlapping area and $OL_{28}$ is 1.38(see table 8) so the reduction of the area can be neglected.



(a) a regular icosahedron (b) polyhedron with 20480 equilateral triangular meshes

**Figure 34 a regular icosahedron before and after subdivision**

The locations of the vertices of a regular icosahedron can be described using spherical coordinates as latitude and longitude. If two vertices are taken to be at the north and south poles (latitude ±90°), then the other ten vertices are at latitude ± arctan $\frac{1}{2}$ = ± 26.57°. These ten vertices are at evenly spaced longitudes (36° apart), alternating between north and south latitudes.

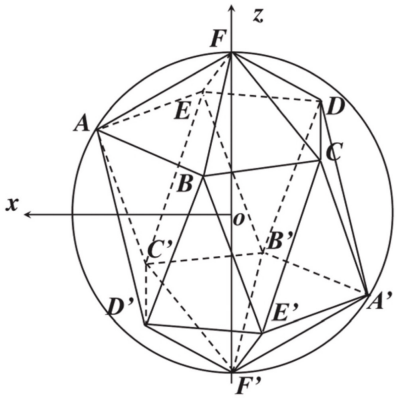



**Figure 35 Subdivision of an equilateral triangle**

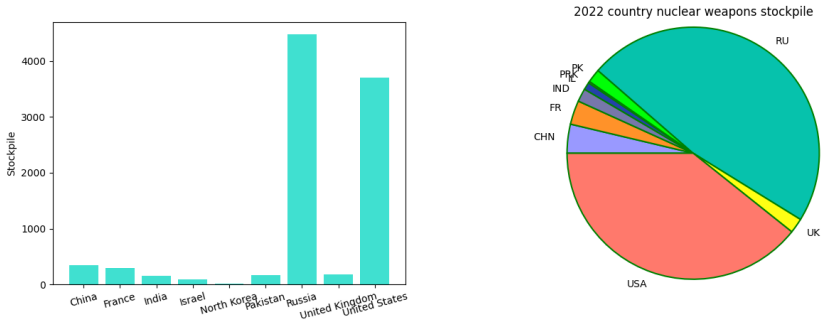
The coordinates of point F are (0,0,R)(R is the radius of circumscribed sphere).A is located on the plane $xoz$, a mirror plane of the icosahedron.The coordinates of points B, C, D, and F, can be obtained by $\frac{2\pi}{5}$ rotations of point A around the z axis.

With all vertexes know, we can approximately get the vertexes of subdivision of the 20 surfaces. Assuming the earth to be a sphere, 3D coordinates of its center is (0, 0, 0). **face_vertex_order.xlsx** in attachment shows the **order numbers of three vertexes that constitute each face** of the the polyhedron with 20480 equilateral triangular meshes. **coordinates_of_all_vertexes.xlsx** shows the **coordinates(x, y, z) of all vertexes**. Then, the centers of 28 circles covering each equilateral triangular mesh can be easily calculated since the structure is in the dihedral group(see picture 33). [2]

All in all, the number of nuclear bombs are required at least to destroy the earth is $20480 \times 28 = 573440$. Moreover, as discussed above, in each equilateral triangular mesh, the detonation positions of nuclear weapons is the center of the 28 circles, which are calculated.

### 3.3.5 *Problem 3(b)*

Using the data in sheet ”stockpile” of the Table ”2022APMCMEData.xlsx”, We calculate the sum of nuclear weapons of each country in 2022, which is equal to 9440. The sum is less that the number that can destroy the earth we calculated in problem 3(a), so the stockpile we have now is not enough to destroy the earth.




(a) 2022 stockpile bar chart (b) 2022 stockpile pie chart

**Figure 36 countries nuclear weapons of 2022**

### 3.3.6 *Problem 3(c)*

Our idea is to first calculate the **proportion of human living area** in the area of the earth, and then calculate the number of nuclear bombs needed to blow up human living area by **multiplying** the number of nuclear bombs that destroy the earth’s surface solved in the first question by the corresponding proportion.

We found on the Internet that the area of human habitation is 124.5 million square kilometers, but it includes a large number of human living areas with very small population density. So we decided to calculate the living area of human beings by the method we come up with ourselves.

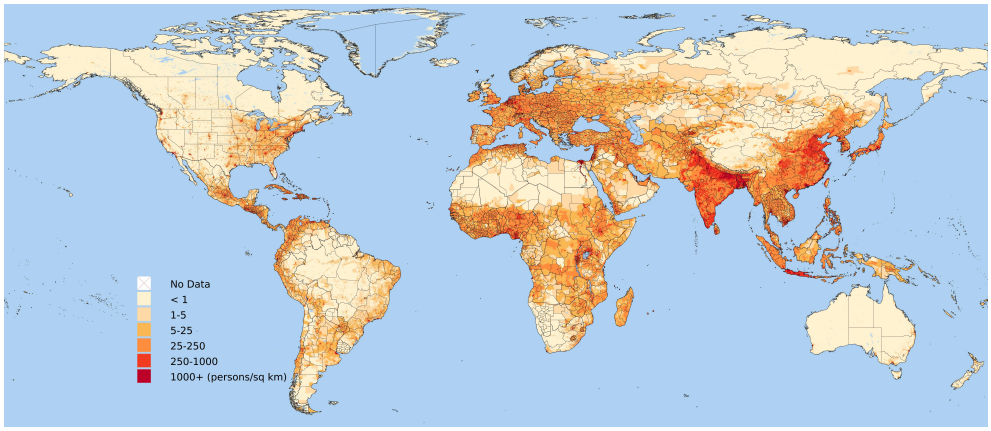



**Figure 37 population density map**

We first convert each pixel of the image to the corresponding RGB values, and then filter the RGB values to get the size of the pixel that needs color. In this case, We selected the area with population density equal to or greater than 30/qs $km$.

| Number of pixels | Proportion | Land area | residential area |
|---|---|---|---|
| 2157319 | 0.3351 | 0.149 billion | 49923870 |

**Table 10 calculate residential area**

We can calculate that the human living area (after density filtering) accounts for the earth area is

$$proportion = \frac{49923870}{5.1 * 10^9} = 0.98\%$$

then we have the number of nuclear weapons needed to destroy human being living environment is equal to

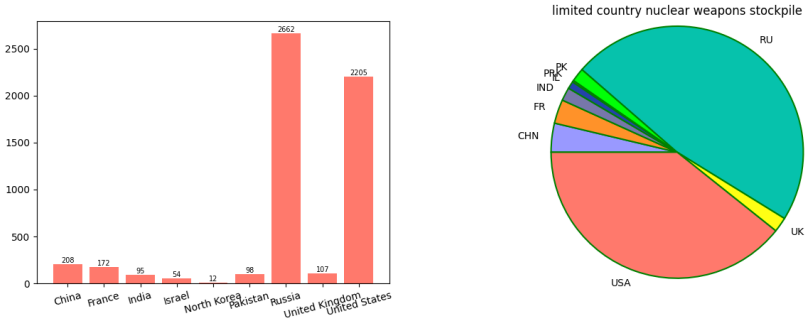
$$num_{human} = proportion \times num_{earth} = 0.0098 * 573440 = 5613$$

so the number of nuclear weapons should be limited to 5613.

We allocate the limited nuclear sum according to the proportion of the number of nuclear bombs in each country to the total number in 2022, we get

| | China | France | India | Israel | North Korea | Pakistan | Russia | UK | US |
|---|---|---|---|---|---|---|---|---|---|
| limit | 208 | 172 | 95 | 53 | 11 | 98 | 2662 | 107 | 2204 |

**Table 11 limited nuclear weapons number**




(a) nuclear weapons bar chart (b) nuclear weapons pie chart

**Figure 38 limited nuclear weapons number**

### 3.3.7 *Problem 4*

Nuclear weapons refer to huge lethal weapons related to nuclear reaction, including hydrogen bombs, atomic bombs, neutron bombs, etc. Nuclear weapons are one of the most powerful weapons ever developed by human beings, and they often remind us of the scene of destroying heaven and earth.

In our project, we select GDP and military expenditure as indicators and use machine learning methods to predict the number of nuclear weapons in the future. However, the results are not optimistic.We find that in the next 100 years, the number of countries with nuclear weapons will increase to 13, and the number of nuclear weapons will continue to grow. At the beginning of the 22nd century(2123), the total number of nuclear weapons will grow increase to 9837.

Perhaps this number does not look terrible, but it is devastating to the earth.We divide the earth's surface into 20480 meshes, and use covering algorithm to find minimum number of nuclear bombs. According to our results, destroying a planet only requires 573440 tsar bombs! Furthermore, if we only project nuclear bombs to human activity intensive points, the number of nuclear bombs needed will even be reduced to 5613. Actually, we seem to be defending our country by researching and manufacturing of nuclear weapons, but we are actually destroying ourselves.

Do we really need so many nuclear weapons to protect our national security? Through our calculation, the answer is no. We collected the data set on ratio of military expenditure of countries with nuclear weapons to GDP, and calculated that each country is at stable status through Saperstein model. The result illustrates that, between every two of the eight countries, both of them are assured that they can inflict unacceptable damage upon their opponents even after suffering the worst first-strike devastation their opponent is capable of delivering. So there is no need to increase nuclear weapons to protect their country. However, we still see some disharmonious phenomena, such as India and Pakistan, two enemy countries. Although both countries are in stable regions, the number of nuclear weapons is still increasing.In this case, more expenditure on nuclear weapons does not have a corresponding protective effect.

Is there a better way to make the world develop peacefully? Is there a way to defend national security other than continuously developing the military which is a tit for that way?

Firstly, we propose that the hostile parties can sign an agreement to reduce or even stop the manufacture of nuclear weapons, which can not only deter the other party from nuclear weapons, but also reduce the waste of economy and resources while protecting the environment. Secondly, we hope that the United Nations will act as a supervisor. Every country with nuclear weapons can hand over part of its nuclear weapons to the United Nations. And when countries with nuclear weapons launch wars against other countries and use nuclear weapons, the United Nations can help the attacked countries to fight back. We believe that through the deterrent effect of nuclear weapons, we can not only protect some vulnerable countries, but also reduce the number of countries that want to develop nuclear weapons.

We should make it clear that nuclear weapon itself has both good and bad aspects. It will make countries more cautious to make war because of worries about the terrible consequences of nuclear weapons, it can also make a war much more destructive and cruel. We will continue to explore how to use nuclear weapons to better serve the earth and the world.

# IV. Conclusions

## 4.1 Conclusions of the problem

In conclusion, we work out a reasonable strategy predicting the nuclear stockpiles of the future, which includes multiple mathematical models of mutual verification. Apart from this, we use some scientific algorithms to develop a concrete mathematical model to calculate the potential destruction of massive nuclear weapons, which is of great significance to remind people that nuclear weapons should be limited.

## 4.2 Methods used in our models

**Curve fitting**

We use curve fitting to complete the missing data of historical GDP and military expenditure.

**GRU model**

We use GRU to predict the GDP and military expenditure in the next 100 years.

**Logistic regression**

We use logistic regression to predict the year in which a country likely to develop a nuclear weapon will develop one in the future.

**Polynomial regression**

We use polynomial regression to predict the stockpile of countries beginning to possess nuclear weapon in the future.

**Game theory(Saperstein model)**

We use the game theory to explain the steady stockpiles of some countries in the future.

**Spherical subdivision algorithm**

We use this algorithm to translate the three-dimensional spherical surface of the earth into a two-dimensional plane formed by concatenation of equilateral triangles.

**Optimal-covering algorithm**

We use this algorithm to work out the least number of bomb destruction circles to cover the earth.

**Pixel counting algorithm**

We use this algorithm to calculate the area of human living environment.

## 4.3 Strength and Weakness

**Strength**

1) We combine various machine learning methods(GRU model, logistic regression, polynomial regression) and establish a sophisticated predicting system.

2) Our model has strong mathematical theory support, such as spherical subdivision algorithm and pixel counting algorithm. We also write some principle mathematical proofs of the model in the paper.

3) We also analyzed the details. We have made special analysis on some data that do not quite conform to the model results.When it is found that the change of the number of nuclear bombs in

India and Pakistan is inconsistent with Saperstein model, we used GRU and polynomial regression method to make a special prediction for them.

**Weakness**

1) For the sake of reducing workload, we only choose three factors as the input of the regression model, which is not enough.

2) Although some of our approximation is close to the reality, there still exist some small error comparing to the realty.

### 4.4 Applications of our models

1) Our model gives a reference to the government of different countries about the limited amount of nuclear weapons.

2) The prediction strategy can be extended to other prediction problems.

3) The thinking of translating three-dimensional sphere to two-dimensional plane can also be used in other covering problems.

## V. Future Work

### 5.1 Model Improvements

1) If we take more related factors into consideration, we can have more data to train our model, thus the better accuracy of prediction.

2) As to the destruction area of a nuclear bomb, we haven't take the difference of land and sea into consideration. If we have the scientific data of that, our model can be more detailed and precise.

### 5.2 Model Extension

All of our model is based on the assumption that there won't be a world war in the future. We have only considered about the situation of peacetime. More work could be done in the event of another world war in the future. To extend our model to situation of war, we can use data in past world wars to train our model to predict the likely distribution of nuclear stockpiles in future wars.

## VI. References

[1] Vladimir Florinski, Dinshaw S. Balsara, Sudip Garain, and Katharine F. Gurski. Technologies for supporting high-order geodesic mesh frameworks for computational astrophysics and space sciences. Computational Astrophysics and Cosmology, 7:1, 2020.

[2] Kari J. Nurmela. Conjecturally optimal coverings of an equilateral triangle with up to 36 equal circles. Experimental Mathematics, 9(2):241–250, 2000.

[3] Alvin M Saperstein. Chaos—a model for the outbreak of war. Nature, 309(5966):303–305, 1984.

[4] Alex Wellerstein. Effect distances for a 50 megaton airburst. https://nuclearsecrecy.com/nukemap/?&hob_opt=2&psi=20,5,300&crater=1&ff=3&linked=1&kt=50000&lat=51.5073509&lng=-0.1277583&hob_psi=5&hob_ft=200&zm=9.

# VII. Appendix

**Listing 1:** problem 1(b)

```
f = open("stockpiles.txt",'rt')
lines = f.readlines()
changeDict = {}
country = []
f.close()
for i in range(len(lines)):
    line = lines[i].split("\t")
    line[3] = line[3].strip("\n")
    if line[2] == '2002' or line[2] == '2022':
        country.append(line)
j = 0
while j < len(country) - 1:
    changeDict[country[j][0]] = abs(int(country[j][3]) - int(country[j+1][3]))
    j += 2
sortedchange = sorted(changeDict.items(),key = lambda x:x[1],reverse = True)
print(sortedchange)
```

**Listing 2:** problem 1(c)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_excel("1_c_material.xlsx", usecols=["Country", "Year", "Tests"])
dict_weapon = {}
for i in range(df.shape[0]):
   if df["Year"][i] not in dict_weapon.keys():
      dict_weapon[df["Year"][i]] = df["Tests"][i]
   else:
      dict_weapon[df["Year"][i]] += df["Tests"][i]
year = []
sum = []
flag = True
result = {}
for key in dict_weapon.keys():
   x = "%d-%d" % (key, key + 5)
   year.append("%d-%d" % (key, key + 5))
   result[x] = 0
   sum_ = 0
   for i in range(5):
      if dict_weapon.get(key + i) != None:
         sum_ += dict_weapon.get(key + i)
      else:
         flag = False
         break
```

```
    if flag == False:
        break
    sum.append(sum_)
    result[x] = sum_
year = np.array(year)
sum = np.array(sum)
max_index = np.argmax(sum)
maxyear = year[max_index]
maxsum = sum[max_index]
# output the maximum sum of five year nuclear test and its corresponding years
print(maxyear, maxsum)
# The descending order of the sum of five years' experiments (all)
my_result = sorted(result.items(),key=lambda x:x[1],reverse=True)

# Data visualization
sort_less_result = []
plt.figure("Descending histogram of nuclear test every five years")
plt.rcParams["font.sans-serif"]=["SimHei"]
for k, v in result.items():
    if v > 301:
        sort_less_result.append([k, v])
sort_less_result = sorted(sort_less_result,key=lambda x:x[1],reverse=True)
# The descending order of the sum of five years' experiments (top ten)
# print(sort_less_result)
x = []
y = []
for i in sort_less_result:
    x.append(i[0])
    y.append(i[1])
plt.bar(x, y, color='orange', label='test sum')
plt.xticks(rotation=30)
plt.legend()
plt.show()
```

**Listing 3:** problem 1(d)

```
f = open("test.txt",'rt')
lines = f.readlines()
testDict = {}
country = []
f.close()
for i in range(len(lines)):
    line = lines[i].split("\t")
    line[3] = line[3].strip("\n")
    if int(line[2]) >= 2009 and int(line[2]) <= 2019:
        country.append(line)
for j in range(len(country)):
    if country[j][0] not in testDict:
        testDict[country[j][0]] = 0
```

```
    testDict[country[j][0]] += int(country[j][3])
sortedsum = sorted(testDict.items(),key = lambda x:x[1],reverse = True)
print(sortedsum)
```

**Listing 4:** problem 2(a)

```
"""-----------------------original_process.py---------------------"""
# function: extract useful data from dataset on GDP
import numpy as np
import pandas as pd
import pickle

df = pd.read_csv('1960_2021各国GDP.csv')
Abb = ['DZA', 'ARG', 'AUS', 'BRA', 'EGY', 'DEU', 'IDN', 'IRN', 'IRQ', 'ITA', 'JPN',
    'LBY', 'NOR', 'ROU', 'SRB', 'KOR', 'SWE', 'CHE', 'SYR', 'ZAF']
year = np.arange(1960, 2022)
df = df.set_index(keys='Country Code')

for i in df.index:
   if i in Abb:
      GDP_ext = df.loc[i][0: 63]
      GDP_ext = np.array(GDP_ext)
      df2 = pd.read_excel("example.xlsx", usecols=["Year", "GDP"])
      to_write1 = np.empty((62, 1), dtype=int)
      to_write2 = np.empty((62,1),dtype=float)
      to_write = np.concatenate((to_write1, to_write2), axis=1)
      to_write[:, 0] = year
      to_write[:, 1] = GDP_ext
      with open("GDP/%s_GDP.csv"%(i), 'wb') as f:
         pickle.dump(to_write,f)
```

**Listing 5:** problem 2(a)

```
"""-----------------------military_process.py---------------------"""
# function: extract useful data from dataset on military expenditure
import numpy as np
import pandas as pd

df = pd.read_csv('military-expenditure-total.csv', usecols=['Entity', 'Year',
    'military_expenditure'])
country = ['Libya', 'Syria', 'South Africa', 'Iran']

i = 0
while i < df.shape[0]:
   if df['Entity'][i] in country:
      countryName = df['Entity'][i]
      year_arr = []
      milit_arr = []
      while df['Entity'][i] in country:
```

```
        year_arr.append(df['Year'].iloc[i])
        milit_arr.append(df['military_expenditure'].iloc[i])
        i += 1
    columns1 = ['Year', 'Military']
    num = len(year_arr)
    index1 = np.arange(0, num)
    df2 = pd.DataFrame(columns=columns1, index=index1)
    df2['Year'] = np.array(year_arr).T
    df2['Military'] = np.array(milit_arr).T
    print(df2)
    df2.to_csv("military/%s_military.csv"%countryName, index=False)
  i += 1
```

**Listing 6:** problem 2(a)

```
"""-----------------------data_process.py---------------------"""
# function: process data to prepare for training and testing GRU
# note: because the size of data from different countries is different, we write 8
    data_process(_xxx).py to process data independently. The functions in these 8
    python files are similar, so we only add data_process_SA.py here to save space and
    make our paper more concise.The other 7 data_process(_xxx).py are included in our
    supporting material.
import numpy as np
from torch.utils.data import Dataset
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler

"""
data preprocess and visualization
"""
def loading_data(feature, backward):
    """
    :param feature: Selected training characteristics
    :param backward: Years to go forward
    :return: Training data, test data, feature list, timeline
    """
    with open('../military/SouthAfrica_military.csv', 'r') as file:
        df = pd.read_csv(file, usecols=["Year", "Military"])
        df = df.replace({np.NaN: 0})
        aver = df["Military"].sum()/62
        df = df.replace({0:aver})
        data = df[['Year', 'Military']]
        data_header = data.keys().tolist()
        time_line = df['Year']
        num_train = int(df.shape[0] * 0.7) + 1
        time_train = time_line.iloc[:num_train].values
        time_eval = time_line.iloc[num_train:].values
        t_time = []
```

```
        e_time = []
        for i in range(len(time_train)):
            t_time.append(str(time_train[i]))
        for i in range(len(time_eval)):
            e_time.append(str(time_eval[i]))
        train_data = df[[feature]].iloc[:num_train]
        eval_data = df[[feature]].iloc[num_train:]
        return train_data, eval_data, data_header, time_line
```

**Listing 7:** problem 2(a)

```
"""------------------------test.py----------------------"""
# function: training and testing GRU and use the trained GRU to predict future GDP and
    military.
# note: because the size of data from different countries is different, we write 8
    test(_xxx).py to train GRU model independently. The functions in these 8 python
    files are similar, so we only add test_SA.py here to save space and make our paper
    more concise.The other 7 test(_xxx).py are included in our supporting material.
from torch.utils.data import DataLoader
import data_process_SA
import torch.nn as nn
import torch.optim as optim
import matplotlib.pyplot as plt
import torch
from sklearn.preprocessing import MinMaxScaler
import numpy as np
import pandas as pd

future = 1
"""
data
"""



def process_train_data(dataset, backward):
    """
    Load Training Data
    :param dataset: dataset
    :param n: Years to backward
    :return: Training set, training tag, test set, test tag
    """
    n = backward + future
    scaler = MinMaxScaler(feature_range=(-1, 1))
    data_raw = scaler.fit_transform(dataset.values.reshape(-1, 1))
    datatable = []
    for index in range(0, len(data_raw) - n):
        datatable.append(data_raw[index: index + n])
    data = np.array(datatable, dtype=np.float64)
    data = data.reshape(data.shape[0], n)
```

```
    x_train = data[:, :-future]
    y_train = data[:, -future:]
    return [x_train, y_train]


def process_eval_data(dataset, backward):
    """
    Load validation data
    :param dataset: dataset
    :param n: Years to backward
    :return: Test data, test label
    """
    global future
    n = backward + future
    scaler = MinMaxScaler(feature_range=(-1, 1))
    data_raw = scaler.fit_transform(dataset.values.reshape(-1, 1))
    datatable = []
    for index in range(0, len(dataset) - n, future):
        datatable.append(data_raw[index: index + n])
    data = np.array(datatable, dtype=np.float64)
    data = data.reshape(data.shape[0], n)
    eval_data = data[:, :-future]
    eval_label = data[:, -future:]
    eval_label = scaler.inverse_transform(eval_label.reshape(-1, 1))
    return eval_data, eval_label


"""
model
"""


class GRU(nn.Module):
    def __init__(self, input_size):
        super(GRU, self).__init__()
        self.gru = nn.GRU(
            input_size=input_size,
            hidden_size=128,
            num_layers=4,
            batch_first=True,
            bidirectional=False
        )
        self.out = nn.Sequential(
            nn.Linear(128, future)
        )

    def forward(self, x):
        r_out, hidden = self.gru(x, None)
        out = self.out(r_out)
```

```
        return out


"""
train
"""
train_para = {
    'input_dim': 1, # Characteristic number of data
    'hidden_dim': 64, # Number of neurons in hidden layer
    'num_layers': 2, # Number of layers of GRU
    'output_dim': 1, # Characteristic number of predicted value
    'lr': 0.001, # learning rate
    'batch_size': 4,
    'dropout': 0.5,
    'epoch': 200, # Number of training rounds
    'backward': 9, # Years to backward
    'feature': 'Military'
} # Training parameters

train_data, Eval_data, data_header, time_line =
     data_process_SA.loading_data(train_para['feature'], train_para['backward'])
train_data, train_label = process_train_data(train_data, train_para['backward'])

train_dataset = data_process_SA.LSTMDataset(train_data, train_label)

train_loader = DataLoader(train_dataset, batch_size=4, shuffle=True, drop_last=True)

"""
print('=============train_dataset =============')
# Shape and label of the output dataset
print(train_dataset.__getitem__(1)[0].shape, train_dataset.__getitem__(1)[1])
# Length of the output dataset
print(train_dataset.__len__())
print('=============eval_dataset =============')
"""

def draw_process(title, color, iters, data, label):
    plt.title(title, fontsize=24)
    plt.xlabel("iter", fontsize=20)
    plt.ylabel(label, fontsize=20)
    plt.plot(iters, data, color=color, label=label)
    plt.legend()
    plt.grid()
    plt.show()


def train(model):
    print('start training...')
    model.train()
```

```
    criterion = nn.MSELoss()
    opt = optim.Adam(model.parameters(), lr=train_para['lr'])
    steps = 0
    Iters, train_loss, test_loss, epo = [], [], [], []
    for epoch in range(train_para['epoch']):
        for batch_id, data in enumerate(train_loader):
            opt.zero_grad()
            steps += 1
            stock = data[0]
            label = data[1]
            stock = stock.float()
            label = label.float()
            output = model(stock)
            loss = criterion(output, label)
            if batch_id % 50 == 0:
                Iters.append(steps)
                train_loss.append(loss.item())
                print("epoch: {}, batch_id: {}, loss is: {}".format(epoch, batch_id,
                    loss.item()))
            loss.backward()
            opt.step()
        model.eval()
        model.train()
    draw_process("training loss", "red", Iters, train_loss, "training loss")
    torch.save(model.state_dict(), '../model/GRU_SA')
    print('finish!')


Model = GRU(train_para['backward'])
train(Model)

"""
eval
"""
backward = train_para['backward']
a, b, c, d = data_process_SA.loading_data('Military', backward)
eval_data, eval_label = process_eval_data(b, backward)
time_eval = d.iloc[42:].values
Model = GRU(backward)
state_dict = torch.load('../model/GRU_SA')
Model.load_state_dict(state_dict) # Load model parameters
Model.eval() # Authentication Mode
result = []
scaler = MinMaxScaler(feature_range=(-1, 1)).fit(eval_label.reshape(-1, 1))
for data in eval_data:
    stock = torch.from_numpy(data.reshape(-1, backward))
    stock = stock.float()
    output = Model(stock)
    output = output.detach().numpy()
```

```
    output = scaler.inverse_transform(output.reshape(-1, 1))
    result.append(output)
result = np.array(result)
result = result.reshape(-1, 1)
plt.figure("accurancy on test set of South Africa(military)")

diff = eval_label - result
x = np.array([i for i in range(eval_label.shape[0])])
plt.plot(x, eval_label, color='red', label='Truth')
plt.plot(x, result, color='blue', label='Predict')
plt.plot(x, diff, color='green', label='diff')
plt.xlabel("Test Year")
plt.ylabel("Military Expense")
plt.legend()

with open('../military/SouthAfrica_military.csv', 'r') as file:
    df = pd.read_csv(file, usecols=["Year", "Military"])
    df = df.replace({np.NaN: 0})
    aver = df["Military"].sum() / 62
    df = df.replace({0: aver})
    year = np.arange(0, 100)
    a = df["Military"][-backward:]
    a = scaler.fit_transform(a.values.reshape(-1, 1))
    a = np.array([a])
    predGDP = torch.from_numpy(a.reshape(-1, backward))
    predGDP = torch.flatten(predGDP, 1)
    predGDP = predGDP.to(torch.float32)
    result1 = []
    for i in range(100):
        output = Model(predGDP)
        for j in range(-backward, -1):
            predGDP[0][j] = predGDP[0][j + 1]
        predGDP[0][-1] = output
        output = output.detach().numpy()
        output = scaler.inverse_transform(output.reshape(-1, 1))
        result1.append(output)
    result2 = []
    for x in result1:
        result2.append(x[0][0])
    plt.figure("predict military of South Africa in next 100 year")
    plt.plot(year, result2, color='cornflowerblue')
    plt.xlabel("Year")
    plt.ylabel("Military Expense")
    plt.show()

    columns1 = ['Year','Military']
    index1 = np.arange(0,100)
    SA = pd.DataFrame(columns=columns1,index=index1)
    SA['Year'] = year
```

```
    SA['Military'] = result2
    SA.to_csv("M_future_SA.csv", index=False)
```

**Listing 8:** problem 2(b)

```
from sklearn.linear_model import LinearRegression
import pandas as pd
import numpy as np
import pandas as pd
from pandas import DataFrame
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures

df = pd.read_csv("modelData.CSV", header=0)
df.info()
X = df.iloc[:,0:-1]
X = np.array(X)

#print(X)
#数据min_max归一化到（0，1）间
min_max_scaler = preprocessing.MinMaxScaler(feature_range=[0, 1])
X_Scaler = min_max_scaler.fit_transform(X)

Y = df.iloc[:, -1]
Y = np.array(Y)
X_train,X_test,Y_train,Y_test = train_test_split(X_Scaler,Y,test_size=0.2)


polynomial = PolynomialFeatures(degree = 3, include_bias = False, interaction_only =
    False)
X_train_polynomial = polynomial.fit_transform(X_train)
X_test_polynomial = polynomial.fit_transform(X_test)
linear = LinearRegression()
model = linear.fit(X_train_polynomial, Y_train)
y_predict = model.predict(X_test_polynomial)


def R2(Y_Test, y_pre):
    u = np.sum((Y_test - y_pre)**2)
    v = np.sum((Y_test - np.mean(y_pre))**2)
    print("R2 = ", 1 - (u/v))
R2(Y_test, y_predict)
import matplotlib.pyplot as plt
IR = pd.read_csv("pakistan_predictData.csv", header=0)
IR.info()
X_IR = IR.iloc[:,0:-1]
#print(X_IR)
X_IR = np.array(X_IR)
```

```
l = len(X_IR)
#print(l)
X_Mix = np.vstack((X_IR, X))
X_Mix = min_max_scaler.fit_transform(X_Mix)
#print(X)
X_IR_scaler = X_Mix[0:l,:]
#print(X_SA_scaler)
X_IR_Poly = polynomial.fit_transform(X_IR_scaler)
y_IR = model.predict(X_IR_Poly)
s = 280
for i in range(len(y_IR)):
    y_IR[i] = int(y_IR[i])

for i in range(len(y_IR)):
    print(y_IR[i])

print(y_IR)

year = IR.iloc[:,-1]
#print(year)
year_list = np.array(year)
plt.plot(year_list, y_IR)
plt.xlabel('Year')
plt.ylabel('stockpiles')
plt.title("The Stockpile Of Pakistan Since 2023")
SYR = pd.read_csv("India_predictData.csv", header=0)
SYR.info()
X_SYR = SYR.iloc[:,0:-1]
#print(X_SYR)
X_SYR = np.array(X_SYR)
l = len(X_SYR)
#print(l)
X_Mix = np.vstack((X_SYR, X))
X_Mix = min_max_scaler.fit_transform(X_Mix)
#print(X)
X_SYR_scaler = X_Mix[0:l,:]
#print(X_SA_scaler)
X_SYR_Poly = polynomial.fit_transform(X_SYR_scaler)
y_SYR = model.predict(X_SYR_Poly)
k = 128
for i in range(len(y_SYR)):
    y_SYR[i] = y_SYR[i]/20
    y_SYR[i] = int(y_SYR[i])
    if(y_SYR[i] < 132 and i <= 58):
        y_SYR[i] = k - 2
    if(y_SYR[i] < 132 and i > 58):
        y_SYR[i] = 120

for i in range(len(y_SYR)):
```

```
    print(y_SYR[i])
print(y_SYR)
year = SYR.iloc[:,-1]
#print(year)
year_list = np.array(year)
plt.plot(year_list, y_SYR)
plt.xlabel('Year')
plt.ylabel('stockpiles')
plt.title("The Stockpile Of India Since 2023")
```

**Listing 9:** problem 2(b)

```
from sklearn.linear_model import LinearRegression
import pandas as pd
import numpy as np
import pandas as pd
from pandas import DataFrame
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures

df = pd.read_csv("randomData", header=0)
df.info()
X = df.iloc[:,0:-1]
X = np.array(X)

#数据min_max归一化到（0, 1）间
min_max_scaler = preprocessing.MinMaxScaler(feature_range=[0, 1])
X_Scaler = min_max_scaler.fit_transform(X)

Y = df.iloc[:, -1]
Y = np.array(Y)
X_train,X_test,Y_train,Y_test = train_test_split(X_Scaler,Y,test_size=0.2)

polynomial = PolynomialFeatures(degree = 3, include_bias = False, interaction_only =
    False)
X_train_polynomial = polynomial.fit_transform(X_train)
X_test_polynomial = polynomial.fit_transform(X_test)
linear = LinearRegression()
model = linear.fit(X_train_polynomial, Y_train)
y_predict = model.predict(X_test_polynomial)

def R2(Y_Test, y_pre):
    u = np.sum((Y_test - y_pre)**2)
    v = np.sum((Y_test - np.mean(y_pre))**2)
    print("R2 = ", 1 - (u/v))

R2(Y_test, y_predict)
SA = pd.read_csv("SouthAfricaData.csv", header=0)
```

```
SA.info()
X_SA = SA.iloc[:,0:-1]
X_SA = np.array(X_SA)
l = len(X_SA)
X_Mix = np.vstack((X_SA, X))
X_Mix = min_max_scaler.fit_transform(X_Mix)
X_SA_scaler = X_Mix[0:l,:]
X_SA_Poly = polynomial.fit_transform(X_SA_scaler)
y_SA = model.predict(X_SA_Poly)
for i in range(len(y_SA)):
    y_SA[i] = int(y_SA[i])
    print(y_SA[i])
print(y_SA)

year = SA.iloc[:,-1]
year_list = np.array(year)
import matplotlib.pyplot as plt
plt.plot(year_list, y_SA)
plt.xlabel('Year')
plt.ylabel('stockpiles')
plt.title("The Stockpile Of South Africa Since 2096")

IR = pd.read_csv("iranData.csv", header=0)
IR.info()
X_IR = IR.iloc[:,0:-1]
#print(X_IR)
X_IR = np.array(X_IR)
l = len(X_IR)
#print(l)
X_Mix = np.vstack((X_IR, X))
X_Mix = min_max_scaler.fit_transform(X_Mix)
#print(X)
X_IR_scaler = X_Mix[0:l,:]
#print(X_SA_scaler)
X_IR_Poly = polynomial.fit_transform(X_IR_scaler)
y_IR = model.predict(X_IR_Poly)
for i in range(len(y_IR)):
    y_IR[i] = int(y_IR[i])
for i in range(len(y_IR)):
    print(y_IR[i])
print(y_IR)
year = IR.iloc[:,-1]
#print(year)
year_list = np.array(year)
plt.plot(year_list, y_IR)
plt.xlabel('Year')
plt.ylabel('stockpiles')
plt.title("The Stockpile Of Iran Since 2046")
LBY = pd.read_csv("LBYData.csv", header=0)
```

```
LBY.info()
X_LBY = LBY.iloc[:,0:-1]
#print(X_LBY)
X_LBY = np.array(X_LBY)
l = len(X_LBY)
#print(l)
X_Mix = np.vstack((X_LBY, X))
X_Mix = min_max_scaler.fit_transform(X_Mix)
#print(X)
X_LBY_scaler = X_Mix[0:l,:]
#print(X_SA_scaler)
X_LBY_Poly = polynomial.fit_transform(X_LBY_scaler)
y_LBY = model.predict(X_LBY_Poly)
for i in range(len(y_LBY)):
    y_LBY[i] = int(y_LBY[i])
    if(y_LBY[i] <= 0):
        y_LBY[i] = 1
    print(y_LBY[i])
#print(y_LBY)

year = LBY.iloc[:,-1]
#print(year)
year_list = np.array(year)
plt.plot(year_list, y_LBY)
plt.xlabel('Year')
plt.ylabel('stockpiles')
plt.title("The Stockpile Of Libya Since 2069")
```

**Listing 10:** problem 2(a)

```
 import math
import numpy as np
import pandas as pd
from pandas import DataFrame
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from pylab import scatter, show, legend, xlabel, ylabel
import matplotlib.pyplot as plt
import os

df = pd.read_csv("各国各项信息整合2.csv", header=0)
df.info()
f3 = pd.read_csv("future_LBY.csv", header=0)


X = df.iloc[:,2:-1]
df.info
X = np.array(X)
min_max_scaler = preprocessing.MinMaxScaler(feature_range=(-10,10))
```

```
X = min_max_scaler.fit_transform(X)
Y = df.iloc[:, -1]
Y = np.array(Y)
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.33)

def Sigmoid(z):
    sig_z = 1/(1 + np.exp(-z))
    return sig_z

def Hypothesis(theta, x):
    z = np.dot(theta, x)
    return Sigmoid(z)

def Cost_Function(X,Y,theta):
    Jsum = 0
    lamb = 10
    for i in range(len(X)):
        Jsum += (Y[i] * np.log(Hypothesis(theta, X[i])) + (1 - Y[i]) * np.log(1 -
            Hypothesis(theta, X[i])))
    J = - Jsum / len(X)
    return J

from numpy import ndarray


def Gradient_Descent(X,Y,theta,alpha):
    new_theta = theta.copy()
    for j in range(len(new_theta)):
        sum = 0
        for i in range(len(X)):
            sum += (Hypothesis(theta, X[i]) - Y[i]) * X[i][j]
        new_theta[j] = new_theta[j] - alpha / len(X) * sum
    return new_theta

def Logistic_Regression(X,Y,alpha,theta,num_iters):
    epoch_loss = []
    for i in range(num_iters):
        epoch_loss.append(Cost_Function(X,Y,theta))
        theta = Gradient_Descent(X,Y,theta,alpha)
    return theta, epoch_loss

column = 3
initial_theta = np.array([0.0 for i in range(column)])
alpha = 0.1
iterations = 1000

opt_theta, epoch_loss =
    Logistic_Regression(X_train,Y_train,alpha,initial_theta,iterations)
```

```
def cost_plot(epoch_losses, iterations, figure_name, imgPath = "./"):
    plt.switch_backend('Agg')
    plt.figure()
    x = list(range(0,iterations,1))
    plt.plot(x, epoch_losses,'b',label = 'loss')
    plt.xlabel('epoch')
    plt.legend('Logistic Regression Multiple')
    plt.savefig(os.path.join(imgPath,figure_name))

figure_name = "1_recon_loss_multiple_iris.jpg"
cost_plot(epoch_loss,iterations,figure_name)

def test(X,Year,theta):
    length = len(X)
    score = 0
    for i in range(length):
        prediction = round(Hypothesis(X[i],theta))
        if prediction == 1:
            print(Year[i])
            break

def evaluation(X_test,Y_test,theta):
    length = len(X_test)
    score = 0
    for i in range(length):
        prediction = round(Hypothesis(X_test[i],theta))
        answer = Y_test[i]
        if prediction == answer:
            score += 1
    score = float(score) / float(length)
    print('Accuracy = ',score)

evaluation(X_test,Y_test,opt_theta)

f1 = pd.read_csv("future_ZAF.csv", header=0)
X1 = f1.iloc[:,1:]
X1 = np.array(X1)
min_max_scaler = preprocessing.MinMaxScaler(feature_range=(-10,10))
X1 = min_max_scaler.fit_transform(X1)
Year1 = f1.iloc[:, 0]
Year1 = np.array(Year1)
test(X1,Year1,opt_theta)

f4 = pd.read_csv("future_IRN.csv", header=0)

X4 = f4.iloc[:,1:]
X4 = np.array(X4)
min_max_scaler = preprocessing.MinMaxScaler(feature_range=(-10,10))
X4 = min_max_scaler.fit_transform(X4)
```

```
Year4 = f4.iloc[:, 0]
Year4 = np.array(Year4)
test(X4,Year4,opt_theta)

f2 = pd.read_csv("future_SYR.csv", header=0)
X2 = f2.iloc[:,1:]
f2.info()
X2 = np.array(X2)
min_max_scaler = preprocessing.MinMaxScaler(feature_range=(-10,10))
X2 = min_max_scaler.fit_transform(X2)
Year2 = f2.iloc[:, 0]
Year2 = np.array(Year2)

f3 = pd.read_csv("future_LBY.csv", header=0)

X3 = f3.iloc[:,1:]
X3 = np.array(X3)
min_max_scaler = preprocessing.MinMaxScaler(feature_range=(-10,10))
X3 = min_max_scaler.fit_transform(X3)
Year3 = f3.iloc[:, 0]
Year3 = np.array(Year3)
test(X3,Year3,opt_theta)
```

**Listing 11:** problem 2(b)

```
import matplotlib.pyplot as plt
x = [i for i in range(1945, 2023)]
f = open("stockpiles.txt",'rt')
lines = f.readlines()
China = []
France = []
India = []
Israel = []
NorthKorea = []
Pakistan = []
Russia = []
SouthAfrica = []
UnitedKingdom = []
UnitedStates = []
for i in range(len(lines)):
    line = lines[i].split("\t")
    line[3] = line[3].strip("\n")
    #print(line[0])
    if line[0] == "China":
        China.append(int(int(line[3])))
    elif line[0] == "France":
        France.append(int(line[3]))
    elif line[0] == "India":
        India.append(int(line[3]))
```

```
    elif line[0] == "Israel":
        Israel.append(int(line[3]))
    elif line[0] == "North Korea":
        NorthKorea.append(int(line[3]))
    elif line[0] == "Pakistan":
        Pakistan.append(int(line[3]))
    elif line[0] == "Russia":
        Russia.append(int(line[3]))
    elif line[0] == "South Africa":
        SouthAfrica.append(int(line[3]))
    elif line[0] == "United Kingdom":
        UnitedKingdom.append(int(line[3]))
    elif line[0] == "United States":
        UnitedStates.append(int(line[3]))
#print(China)
#plt.title('')
#plt.rcParams['font.sans-serif'] = ['SimHei']
plt.xlabel('Year')
plt.ylabel('stockpiles')
plt.subplot(4,2,1)
plt.plot(x, China, label = "China")
plt.legend()
plt.subplot(4,2,2)
plt.plot(x, France, label = "France")
plt.legend()
plt.subplot(4,2,3)
plt.plot(x, India, label = "India")
plt.legend()
plt.subplot(4,2,4)
plt.plot(x, Israel, label = "Israel")
plt.legend()
plt.subplot(4,2,5)
plt.plot(x, NorthKorea, label = "North Korea")
plt.legend()
plt.subplot(4,2,6)
plt.plot(x, Pakistan, label = "Pakistan")
plt.legend()
plt.subplot(4,2,7)
plt.plot(x, SouthAfrica, label = "South Africa")
plt.legend()
plt.subplot(4,2,8)
plt.plot(x, UnitedKingdom, label = "United Kingdom")
plt.legend()
"""
plt.plot(x, Russia, label = "Russia")
plt.plot(x, UnitedStates, label = "United States")
"""
plt.legend()
plt.show()
```

**Listing 12:** problem 2(b)

```
"""--------------------Saperstein_calculate.py----------------------"""
import pandas as pd
import numpy as np

df = pd.read_csv("processed_prop.csv", usecols=["name", "2019", "2020"])
df["2019"] = df["2019"]/100
df["2020"] = df["2020"]/100
column1 = ["relation", "a", "b"]
index1 = np.arange(0, 28)
df2 = pd.DataFrame(columns=column1, index=index1)

column3 = ["China", "United States", "France", "Israel", "United Kingdom", "India",
    "Pakistan", "Russia"]
index3 = np.arange(0, 8)
df3 = pd.DataFrame(columns=column3, index=index3)
df3["Relaion"] = ["China", "United States", "France", "Israel", "United Kingdom",
    "India", "Pakistan", "Russia"]
df3 = df3.set_index("Relaion")

j = 0
for x in df.index:
   for y in range(x+1, df.shape[0]):
      name = "%s and %s" % (df["name"].iloc[x], df["name"].iloc[y])
      a = round((df["2020"].iloc[x] / (df["2019"].iloc[y] * (1 - df["2019"].iloc[y]) *
          4)), 2)
      b = round((df["2020"].iloc[y] / (df["2019"].iloc[x] * (1 - df["2019"].iloc[x]) *
          4)), 2)
      df2["relation"][j] = name
      df2["a"][j] = a
      df2["b"][j] = b
      x_name = df["name"].iloc[x]
      y_name = df["name"].iloc[y]
      df3[y_name].loc[x_name] = a
      df3[x_name].loc[y_name] = b
      j += 1
df3 = df3.reset_index()
df3.to_csv("chaos_relation.csv", index=False)
```

**Listing 13:** problem 3(a) GetRegularIcosahedron.m

```
function [vertexs, faces]= GetRegularIcosahedron
    t = 0:2*pi/5:(2*pi-2*pi/5);
    vertexs = [cos(t'),sin(t'),zeros(5,1)];
    t = t'+pi/5;
    a = 2*sin(pi/5);
    vertexs=[vertexs;cos(t),sin(t),a*sqrt(3)*ones(5,1)/2];
    h = sqrt(0.75*a^2-cos(pi/5)^2);
    vertexs = [0 0 -h;vertexs;0 0 a*sqrt(3)/2+h];
```

```
   vertexs = vertexs - [zeros(12,2), ones([12,1])*0.5];
   for i = 1 :12
      vertexs(i, :) = vertexs(i, :)/norm(vertexs(i, :), 2);
   end
   faces = [1 2 3
         1 3 4
          1 4 5
          1 5 6
          1 6 2
          2 3 7
          2 6 11
          2 7 11
          3 7 8
          3 4 8
          4 8 9
          4 5 9
          5 9 10
          5 6 10
          6 10 11
          7 8 12
          8 9 12
          9 10 12
          10 11 12
          7 11 12];
    change = 0;
    for i =1 : 20
       if(det([vertexs(faces(i, 1), :);
               vertexs(faces(i, 2), :);
               vertexs(faces(i, 3), :)])<0)
           change = change+1;
           faces(i, :) = [faces(i, 1);faces(i, 3);faces(i, 2)];
    end
end
```

**Listing 14:** problem 3(a) Subdivision.m

```
function [vertex, face] = Subdivision(vertex, face)
 face_num = size(face, 1);
 vertex_num = size(vertex, 1);
 new_vertexs = zeros(face_num*3, 3);
 new_faces = zeros(face_num*3, 3);
 for i = 1 : face_num
    new_vertexs(3*(i-1)+1, :)=(vertex(face(i, 1), :)+vertex(face(i, 2), :))/2;
    new_vertexs(3*(i-1)+1, :)= new_vertexs(3*(i-1)+1, :)/norm(new_vertexs(3*(i-1)+1,
        :),2);
    new_vertexs(3*(i-1)+2, :)=(vertex(face(i, 2), :)+vertex(face(i, 3), :))/2;
    new_vertexs(3*(i-1)+2, :)= new_vertexs(3*(i-1)+2, :)/norm(new_vertexs(3*(i-1)+2,
        :),2);
    new_vertexs(3*(i-1)+3, :)=(vertex(face(i, 3), :)+vertex(face(i, 1), :))/2;
```

```
    new_vertexs(3*(i-1)+3, :)= new_vertexs(3*(i-1)+3, :)/norm(new_vertexs(3*(i-1)+3,
        :),2);
    new_faces(3*(i-1)+1, :) = [face(i, 1), vertex_num+3*(i-1)+1, vertex_num+3*(i-1)+3];
    new_faces(3*(i-1)+2, :) = [face(i, 2), vertex_num+3*(i-1)+2, vertex_num+3*(i-1)+1];
    new_faces(3*(i-1)+3, :) = [face(i, 3), vertex_num+3*(i-1)+3, vertex_num+3*(i-1)+2];
    face(i, :) = [vertex_num+3*(i-1)+1, vertex_num+3*(i-1)+2, vertex_num+3*(i-1)+3];
 end
face = [face; new_faces];
vertex = [vertex; new_vertexs];
function [outputArg1,outputArg2] = untitled(inputArg1,inputArg2)
```

**Listing 15:** problem 3(a)plotsphere.mlx

```
% plot equilateral triangle meshes of a sphere
% the radius of circumscribed shpere is 1
[x,y] = GetRegularIcosahedron
trimesh(y,x(:,1),x(:,2),x(:,3));
[x1,y1] = Subdivision(x, y);
trimesh(y1,x1(:,1),x1(:,2),x1(:,3));
[x2,y2] = Subdivision(x1, y1);
[x3,y3] = Subdivision(x2, y2);
[x4,y4] = Subdivision(x3, y3);
[x5,y5] = Subdivision(x4, y4);
% x6 is the vertexes of all the meshes
[x6,y6] = Subdivision(x5, y5)
trimesh(y6,x6(:,1),x6(:,2),x6(:,3))
x7 = 6371*x6 % multiply the radius of earth
```

**Listing 16:** problem 3(b)

```
"""--------------------------sum.py----------------------------"""
# function: calculate the sum of nuclear weapon in 2022, and draw corresponding bar
    chart and pie chart.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_csv("sum.csv", usecols=['Country', 'Year','Stockpile'])
country = ["China", "United States", "France", "Israel", "United Kingdom", "India",
    "Pakistan", "Russia", "North Korea"]
columns1 = ['name', 'Stockpile']
index1 = np.arange(0, 9)
df2 = pd.DataFrame(columns=columns1, index=index1)

j = 0
for i in df.index:
   if df['Country'].iloc[i] in country and df['Year'].iloc[i] == 2022:
      df2['name'].iloc[j] = df['Country'].iloc[i]
      df2['Stockpile'].iloc[j] = df['Stockpile'].iloc[i]
```

```
    j += 1
# 2022 country nuclear weapons stockpile
print(df2)
# 2022 Stockpile
sum1 = np.sum(df2['Stockpile'])
print(sum1)
# visualization
plt.figure("2022 country nuclear weapons stockpile")
plt.bar(df2['name'], df2['Stockpile'], color='turquoise')
plt.ylabel("Stockpile")
plt.xticks(rotation=15)
plt.show()

plt.rcParams['axes.unicode_minus']=False
plt.axes(aspect='equal')

stockpile = df2['Stockpile']
labels =['CHN','FR','IND','IL','PRK','PK','RU', 'UK','USA']
colors = ['#9999ff', '#ff9229', '#7777aa', '#2442aa', '#dd5555',
    '#01ff07','#06c2ac','#ffff14','#ff796c']

plt.pie(x=stockpile,
       labels=labels,
       colors=colors,
       labeldistance=1.1,
       startangle=180,
       radius=1.2,
       counterclock=False,
       wedgeprops={'linewidth':1.5, 'edgecolor':'green'},
       textprops={'fontsize':10, 'color':'black'},
       )
plt.title('2022 country nuclear weapons stockpile')
plt.show()
```

**Listing 17:** problem 3(c)

```
from PIL import Image
import numpy as np

L_path='map.png'
L_image=Image.open(L_path)
out = L_image.convert("RGB")
img=np.array(out)
print(out)
print(out.size)
h, w, x = img.shape
print(h, w)
print(img.shape)
```

```
land_size = 0
human_size = 0
c = 1

for i in range(h):
    for j in range(w):
        print(c)
        p = img[i][j]
        r = p[0]
        g = p[1]
        b = p[2]
        human = True
        if r==150 and g==150 and b==150:
            human = False
        if r==255 and g==255 and b==255:
            human = False
        if r==255 and g==255 and b==204:
            human = False
        if r==255 and g==237 and b==160:
            human = False
        if human:
            human_size += 1
        c += 1
        if img[i][j]!= [150, 150, 150]:
            land_size += 1

print(land_size)
print(human_size)
print(human_size/land_size)
```

**Listing 18:** problem 3(c)

```
"""--------------------------distribution.py-----------------------"""
# function: Calculate the number of nuclear bombs per country to protect earth
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_csv("2022_bomb.csv", usecols=['name', 'Stockpile'])
sum_bomb = 5613
sum1 = np.sum(df['Stockpile'])
df['percent'] = df['Stockpile']/sum1
df['limit'] = sum_bomb * df['percent']
print(df)
df.to_csv("limit_bomb.csv")

labels =['CHN','FR','IND','IL','PRK','PK','RU', 'UK','USA']
colors = ['#9999ff', '#ff9229', '#7777aa', '#2442aa', '#dd5555',
    '#01ff07','#06c2ac','#ffff14','#ff796c']
```

```
plt.pie(x=df['limit'],
       labels=labels,
       colors=colors,
       labeldistance=1.1,
       startangle=180,
       radius=1.2,
       counterclock=False,
       wedgeprops={'linewidth':1.5, 'edgecolor':'green'},
       textprops={'fontsize':10, 'color':'black'},
       )
plt.title('limited country nuclear weapons stockpile')

plt.figure("nuclear distribution")
plt.bar(df['name'], df['limit'], color='#ff796c')
for a, b in zip(df['name'], df['limit']):
   plt.text(a, b + 0.1, '%.0f' % b, ha='center', va='bottom', fontsize=7)
plt.xticks(rotation=15)
plt.show()
```