1. Introduction. An overview of the project and an outline of the shared work.

For our project, we are having a detail look into energy consumption and pollution related death. Try to find relationships between various energy consumption, GDP, health, and death caused by air pollution. Our group have decided to divide the project into four main parts: pre-processing, Model building and training, GUI, and summary. These works are shared by our three members, each of us are responsible for one part and deal with the final summary part together, then help each other with problems encountered in the projects in charge.

2. Description of your individual work. Provide some background information on the development of the algorithm and include necessary equations and figures.

My major work is to handle the pre-processing section, also hosted the meeting discussion and help designed the model.

For the pre-processing section, I have used some basic measures like data normalization, missing values imputation, data integration, and Noise identification. Data normalization is the process of minimizing redundancy from relation or set of relation. Select the specific datasets or data columns, using the function preprocessing from sklearn:

Formula
$\frac{X-\mu}{\sigma}$

: the formula for normalization of StandardScaler

And the purpose for missing values imputation is to fill up the variables that contain missing values. Data integration is an important step for preprocessing, the purpose of it is merging multiple datasets to let it be read by the model. Finding Noise is a step also necessary. Noise is a data which is unreasonable and should be detected during preprocessing. When finding noises out, we can decide if we are keeping them or drop them out from the dataset. Finally, cleaning datasets, dropping the columns and data which is unnecessary for the model.

For the modeling section, during the discussion we considered many models, and after evaluating the datasets we decided to use linear regression as the basis for our model. The first and most basic one is that a linear regression line has an equation of the form $\boldsymbol{Y = a + bX}$, where $\boldsymbol{X}$ is the explanatory variable and $\boldsymbol{Y}$ is the dependent variable. The slope of the line is $\boldsymbol{b}$, and $\boldsymbol{a}$ is the intercept (the value of $\boldsymbol{y}$ when $\boldsymbol{x} = 0$).

3. **Describe the portion of the work that you did on the project in detail. It can be figures, codes, explanation, pre-processing, training, etc.**

Main body of the preprocessor:

```
class preprocessing():
# DEFINE __INIT__ ,read datasets and set them base on Location
    def __init__(self):
        self.data_health =
pd.read_csv('https://raw.githubusercontent.com/alicanmutluu/DMDatasets/main/
2.12_Health_systems.csv',  sep = ',')
        self.data_clean =
pd.read_csv('https://raw.githubusercontent.com/alicanmutluu/ DMDatasets
/main/cleanFuelAndTech.csv',  sep = ',')
        self.data_pollution =
pd.read_csv('https://raw.githubusercontent.com/alicanmutluu/ DMDatasets
/main/death-rates-from-air-pollution.csv',  sep = ',')

self.data_GDP=pd.read_csv('https://raw.githubusercontent.com/alicanmutluu/
DMDatasets /main/Country_wise_GDP_from_1994_to_2017.csv', sep = ',')
        self.data_energy =
pd.read_csv('https://raw.githubusercontent.com/alicanmutluu/ DMDatasets
/main/Percentage_of_Energy_Consumption_by_Country.csv',  sep = ',')
        self.set1 = {i for i in self.data_health['World_Bank_Name']}
        self.set2 = {i for i in self.data_clean['Location']}
        self.set3 = {i for i in self.data_pollution['Entity']}
        self.set4 = {i for i in self.data_GDP['Country']}
        self.set5 = {i for i in self.data_energy['Entity']}
        self.setf = self.set1.intersection(self.set2)
        self.setf = self.setf.intersection(self.set3)
        self.setf = self.setf.intersection(self.set4)
        self.setf = self.setf.intersection(self.set5)
        self.setf2= self.set1.intersection(self.set2)
        self.setf2= self.setf2.intersection(self.set3)
        self.setf2= self.setf2.intersection(self.set4)
# DEFINE dfselection()
    def dfselection(self,s, df, v):
        ans = df.copy()
        for n, i in df.iterrows():
```

```
            if i[v] not in s:
                ans = ans.drop(n, axis=0)

        return ans
# DEFINE indoor2016() which select the data with year=2016 from the datasets
and merge them together by location
    def indoor2016(self):
        df1=self.dfselection(self.setf2,self.data_health,'World_Bank_Name')
        df2=self.dfselection(self.setf2,self.data_clean,'Location')
        df3=self.dfselection(self.setf2,self.data_pollution,'Entity')
        df4=self.dfselection(self.setf2,self.data_GDP,'Country')
        df2=df2[df2['Period']==2016]
        df3=df3[df3['Year']==2016]
        df4=df4[df4['Year']==2016]
        df1=df1.rename(columns={"World_Bank_Name": "Location"})
        df3=df3.rename(columns={'Entity':'Location'})
        df4=df4.rename(columns={'Country':'Location'})
        out=pd.merge(df1,df2,how='outer',on='Location')
        out=pd.merge(out,df3,how='outer',on="Location")
        out=pd.merge(out,df4,how='outer',on='Location')
        out=out.drop(columns='Province_State')

        return out

# DEFINE indoor() which select data from datasets and merge them by location
and year
    def indoor(self):
        # merge df2 df3 df4 on location and year, on=outer do not drop na
        df2=self.dfselection(self.setf2,self.data_clean,'Location')
        df3=self.dfselection(self.setf2,self.data_pollution,'Entity')
        df4=self.dfselection(self.setf2,self.data_GDP,'Country')
        df2 = df2.rename(columns={"Period": "Year"})
        df3 = df3.rename(columns={'Entity': 'Location'})
        df4 = df4.rename(columns={'Country': 'Location'})
        out = pd.merge(df2, df3, how='outer', on=['Location','Year'])
        out = pd.merge(out, df4, how='outer', on=['Location','Year'])

        return out
```

```
# DEFINE outdoor() which select data from datasets and merge them by
location and year
    def outdoor(self):
        # merge df3 df4 df5, on location and year, on=outer do not drop na
        df3=self.dfselection(self.setf,self.data_pollution,'Entity')
        df4=self.dfselection(self.setf,self.data_GDP,'Country')
        df3 = df3.rename(columns={'Entity': 'Location'})
        df4 = df4.rename(columns={'Country': 'Location'})
        out = pd.merge(df3, df4, how='outer', on=['Location', 'Year'])
        return out
# DEFINE Merge_energy() which select the data from datasets by location and
year, and dropping the features that is not necessary
    def Merge_energy(self):
        df5 = self.dfselection(self.setf2, self.data_energy, 'Entity')
        df3 = self.dfselection(self.setf2, self.data_pollution, 'Entity')
        df4 = self.dfselection(self.setf2, self.data_GDP, 'Country')
        df3 = df3.rename(columns={'Entity': 'Location'})
        df4 = df4.rename(columns={'Country': 'Location'})
        df5 = df5.rename(columns={'Entity': 'Location'})
        out = pd.merge(df5, df3, how='outer', on=['Location', 'Year'])
        out = pd.merge(out, df4, how='outer', on=['Location', 'Year'])
        out = out.drop(columns='Wind Generation -TWh')
        out = out.drop(columns='Solar Generation - TWh')
        out = out.drop(columns='Nuclear Generation - TWh')
        out = out.drop(columns='Hydro Generation - TWh')
        out = out.drop(columns='Geo Biomass Other - TWh')
        out = out.drop(columns='Gas Consumption - EJ')
        return out



#prepare data for indoor2016
pre=preprocessing()
indoor2016=pre.indoor2016()
print(indoor2016.columns)
print(indoor2016.shape)
#prepare data for indoor
pre=preprocessing()
indoor=pre.indoor()
```

```
print(indoor.columns)
print(indoor.shape)
#prepare data for outdoor
pre=preprocessing()
outdoor=pre.outdoor()
print(outdoor.columns)
print(outdoor.shape)
#prepare data for Merge_energy
pre=preprocessing()
Merge_energy=pre.Merge_energy()
print(Merge_energy.columns)
print(Merge_energy.shape)



# Missing data filling with most common value
# data_health.iloc[:, 1:-1] = data_health.iloc[:, 1:-1].apply(lambda x:
x.fillna(x.value_counts().index[0]))
# data_clean.iloc[:, 1:-1] = data_clean.iloc[:, 1:-1].apply(lambda x:
x.fillna(x.value_counts().index[0]))
# data_pollution.iloc[:, 1:-1] = data_pollution.iloc[:, 1:-1].apply(lambda x:
x.fillna(x.value_counts().index[0]))
# data_GDP.iloc[:, 1:-1] = data_GDP.iloc[:, 1:-1].apply(lambda x:
x.fillna(x.value_counts().index[0]))
# data_energy.iloc[:, 1:-1] = data_energy.iloc[:, 1:-1].apply(lambda x:
x.fillna(x.value_counts().index[0]))
```

But when I plot the histogram of these datasets, I find that they are not normal distribution, which if we fill the NA with most common data will make the model performed more worse so we decided to not fill up the NA values. Also, most missing values are distributed in features which finally doesn't relate highly to our target feature (examples like, solar energy generation and electricity from hydro) so we keep the NA and solve it at the model building code. On the other side, the distributions of our selected features are all having similar distributions (most of them are obviously right skewed) so we think that we don't have to modify the origin data distribution. Looking through heat map these feature also have high relation with each other, so this let me doublecheck that the datasets are good enough for putting into the model.

Same situation for noise detecting, I found that there is no noise which will make the model collapse but there are still lots of extreme observations, comparing to the

mean of the data. We decided to keep these extreme values of some observations and deal with it in model building.

```
# Plotting
import matplotlib.pyplot as plt
# clean_fuel, coal, oil, indoor death rate, outdoor death rate, GDP

# EDA of data_clean clean_fuel in histogram
data_clean_p = pd.read_csv('https://raw.githubusercontent.com/alicanmutluu/
DMDatasets /main/cleanFuelAndTech.csv',
                     sep=',')
dc = data_clean_p
num_bins = 50
n, bins, patches = plt.hist(dc['First Tooltip'], num_bins, facecolor='blue',
alpha=0.5)
plt.xlabel('First Tooltip')
plt.ylabel('count')
plt.title('data_clean clean_fuel')
plt.show()

# EDA of data_energy coal consumption in histogram
data_energy_p = pd.read_csv(
    'https://raw.githubusercontent.com/alicanmutluu/ DMDatasets
/main/Percentage_of_Energy_Consumption_by_Country.csv',
    sep=',')
de = data_energy_p
num_bins = 100
n, bins, patches = plt.hist(de['Coal Consumption - EJ'], num_bins,
facecolor='blue', alpha=0.5)
plt.xlabel('Coal Consumption - EJ')
plt.ylabel('count')
plt.title('data_energy Coal Consumption')
plt.show()

# EDA of data_energy oil consumption in histogram
num_bins = 100
n, bins, patches = plt.hist(de['Oil Consumption - EJ'], num_bins,
facecolor='blue', alpha=0.5)
```

```
plt.xlabel('Oil Consumption - EJ')
plt.ylabel('count')
plt.title('data_energy Oil Consumption')
plt.show()

# EDA of data_pollution outdoor pollution in histogram
data_pollution_p = pd.read_csv(
    'https://raw.githubusercontent.com/alicanmutluu/ DMDatasets /main/death-rates-from-air-pollution.csv', sep=',')
dp = data_pollution_p
num_bins = 75
n, bins, patches = plt.hist(dp['Outdoor particulate matter (deaths per 100,000)'], num_bins, facecolor='blue', alpha=0.5)
plt.xlabel('Outdoor particulate matter (deaths per 100,000)')
plt.ylabel('count')
plt.title('data_pollution Outdoor pollution')
plt.show()

# EDA of data_pollution indoor pollution in histogram
num_bins = 100
n, bins, patches = plt.hist(dp['Indoor air pollution (deaths per 100,000)'], num_bins, facecolor='blue', alpha=0.5)
plt.xlabel('Indoor air pollution (deaths per 100,000)')
plt.ylabel('count')
plt.title('data_pollution Indoor air pollution')
plt.show()

data_GDP_p = pd.read_csv(
    'https://raw.githubusercontent.com/alicanmutluu/ DMDatasets /main/Country_wise_GDP_from_1994_to_2017.csv', sep=',')
# EDA of data_GDP GDP per capita in histogram
dg = data_GDP_p
num_bins = 75
n, bins, patches = plt.hist(dg['GDP per capita (in USD)'], num_bins, facecolor='blue', alpha=0.5)
plt.xlabel('GDP per capita (in USD)')
plt.ylabel('count')
```

```
plt.title('data_GDP GDP per capita Year')
plt.show()


# Plotting heat map
# Encode location and drop code_x and code_y which are unnecessary.
le = preprocessing.LabelEncoder()
Merge_energy['Location'] =le.fit_transform(Merge_energy['Location'])
Merge_energy= Merge_energy.drop(columns='Code_x')
Merge_energy= Merge_energy.drop(columns='Code_y')

tc = Merge_energy.corr()
plt.figure(figsize=(16,20))
sns.heatmap(tc,cmap='coolwarm')
plt.show()
```

We can notice that there are extreme values (outliers) in each datasets, much more larger than the mean of datasets, and distributions of each datasets are extremely right skewed (which means that normalization won't work well in these datasets, normalization works better with normal distribution) through the EDA section.

4. **Results. Describe the results of your experiments, using figures and tables wherever possible. Include all results (including all figures and tables) in the main body of the report, not in appendices. Provide an explanation of each figure and table that you include. Your discussions in this section will be the most important part of the report.**

Merge_energy: the main datasets preprocessed using in models

- Columns:
  (['Location', 'Code_x', 'Year', 'Coal Consumption - EJ',
  'Oil Consumption - EJ', 'Code_y',
  'Air pollution (total) (deaths per 100,000)',
  'Indoor air pollution (deaths per 100,000)',
  'Outdoor particulate matter (deaths per 100,000)',
  'Outdoor ozone pollution (deaths per 100,000)', 'GDP (in USD)',
  'GDP Real (in USD)', 'GDP change (%)', 'GDP per capita (in USD)',
  'Pop. change (%)', 'Population'],
  dtype='object')
- Shape of this table:(5613, 16)

Outdoor: datasets preprocessed using in models

- Columns:(['Location', 'Code', 'Year',
  'Air pollution (total) (deaths per 100,000)',
  'Indoor air pollution (deaths per 100,000)',
  'Outdoor particulate matter (deaths per 100,000)',
  'Outdoor ozone pollution (deaths per 100,000)', 'GDP (in USD)',
  'GDP Real (in USD)', 'GDP change (%)', 'GDP per capita (in USD)',
  'Pop. change (%)', 'Population'],
  dtype='object')
- Shape of the table:(1820, 13)

Indoor: datasets preprocessed using in models

- Columns:
  (['Location', 'Indicator', 'Year', 'First Tooltip', 'Code',
  'Air pollution (total) (deaths per 100,000)',
  'Indoor air pollution (deaths per 100,000)',
  'Outdoor particulate matter (deaths per 100,000)',
  'Outdoor ozone pollution (deaths per 100,000)', 'GDP (in USD)',
  'GDP Real (in USD)', 'GDP change (%)', 'GDP per capita (in USD)',
  'Pop. change (%)', 'Population'],
  dtype='object')
- Shape of the table: (4234, 15)

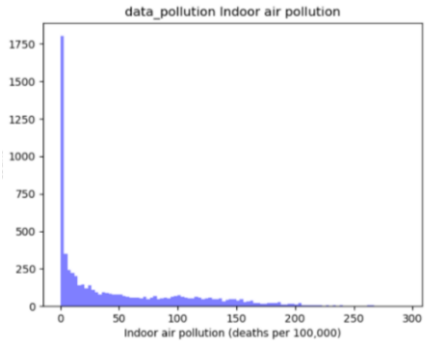
Indoor2016: datasets preprocessed using in models

- Columns:
  (['Country_Region', 'Location', 'Health_exp_pct_GDP_2016',
  'Health_exp_public_pct_2016', 'Health_exp_out_of_pocket_pct_2016',
  'Health_exp_per_capita_USD_2016', 'per_capita_exp_PPP_2016',
  'External_health_exp_pct_2016', 'Physicians_per_1000_2009-18',
  'Nurse_midwife_per_1000_2009-18',
  'Specialist_surgical_per_1000_2008-18',
  'Completeness_of_birth_reg_2009-18',
  'Completeness_of_death_reg_2008-16', 'Indicator', 'Period',
  'First Tooltip', 'Code', 'Year_x',
  'Air pollution (total) (deaths per 100,000)',
  'Indoor air pollution (deaths per 100,000)',
  'Outdoor particulate matter (deaths per 100,000)',
  'Outdoor ozone pollution (deaths per 100,000)', 'Year_y',
  'GDP (in USD)', 'GDP Real (in USD)', 'GDP change (%)',
  'GDP per capita (in USD)', 'Pop. change (%)', 'Population'],
  dtype='object')

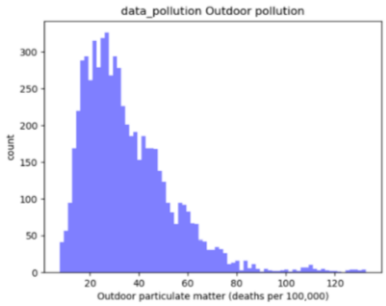
- Shape of the table: (146, 29)

EDA graphs:

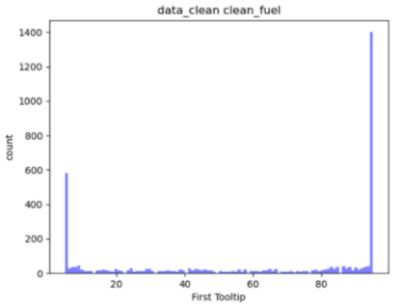
1. Histogram of indoor air pollution



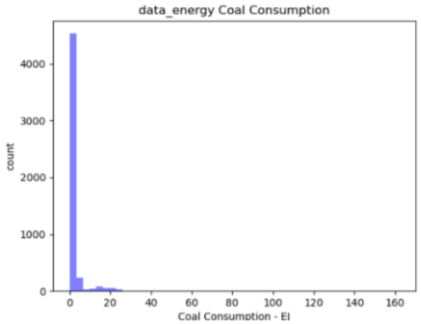
2. Histogram of outdoor air pollution



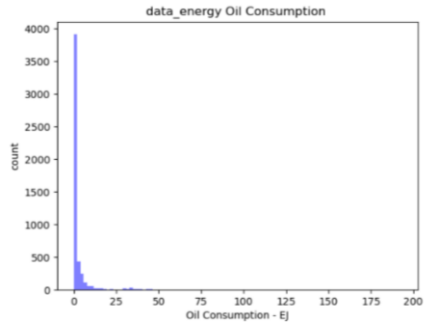
3. Histogram of clean fuel



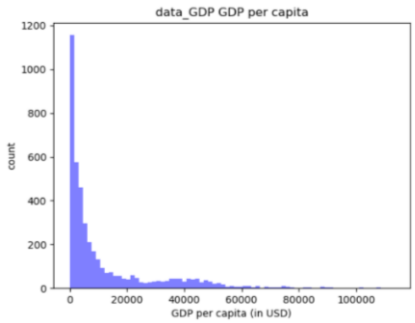
4. Histogram of Coal consumption



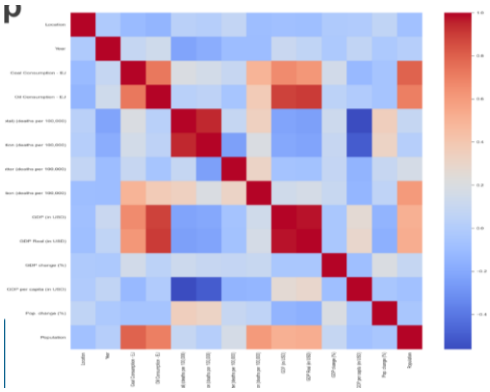
5. Histogram of oil consumption



6. Histogram of GDP



7. Heat map of major features



5. Summary and conclusions. Summarize the results you obtained, explain what you have learned, and suggest improvements that could be made in the future.

Data preprocessing is as important as building the model, this is the most touching summary to me in this report. Due to this time, I am the one who is responsible for handing all related pre-processing, and I have participated in the entire pre-processing process from beginning to end. Choosing and deciding which datasets should we use and which datasets fit our project better really takes time. Although datasets from Kaggle is already cleaned much better than other places, there is still lots of problems in the datasets and solving all the problems of datasets is the major work for me.

Having clear vision of the project target and purpose will be an improvement I should make in future. At the begin of this project we are not quite sure about what we are going to do, we just randomly collected datasets about air pollution. Repeatedly collect and then eliminate really waste lots of time. Next time I should

focus on understanding what I datasets I should look for directly instead of searching datasets without clear direction.

6. Calculate the percentage of the code that you found or copied from the internet. For example, if you used 50 lines of code from the internet and then you modified 10 of lines and added another 15 lines of your own code, the percentage will be 50 − 10 50 + 15 × 100.

All the codes are from class codes and lecture codes: 0

7.Reference