University of Auckland

Iteration 4 BDAS

(Steps 1 – 8)

GitHub Link: [UoA-EEEEason/infosys722-I4: For INFOSYS 722 (github.com)]

Yixuan Peng ypen781 539227446

目录

1. Situation understanding

1.1 Identify the objectives of the business/situation

As global warming intensifies due to the rise in the average temperature of the Earth, primarily from increased greenhouse gas emissions resulting from human activities, countries around the world are urged to take immediate action. This temperature escalation has profound implications, including extreme weather events, sea-level rise, and ecosystem changes. Yet, nations often shift the responsibility for these consequences onto each other (Justin et al., 2023). It's imperative for the World Environmental Organization to present the greenhouse gas emission data of each country (including CH4, N2O, CO2, etc.) to the public. By doing so, countries can be held accountable for the escalating global warming crisis, prompting them to swiftly undertake relevant environmental protection measures. From a business perspective, global warming can lead to resource shortages, supply chain disruptions, and market instability, emphasizing the importance of addressing this issue not only for environmental but also for economic reasons.

Thus, a study is commissioned with the following objectives:

- Accurately determine the responsibilities of countries in the world to ensure that countries can correctly view the responsibilities they should bear.
- Look for relationships between the number of sources of greenhouse gas emissions and total gas emissions in each country.

Tentatively, the study will be judged a success if:

- According to the results of data visualization, the responsibilities of countries in the world can be judged.

- Based on the visualization results, the relationship between the number of greenhouse gas emission sources and the total gas emissions of each country is found.
- Research presents transparent and real data results to the public.

1.2 Assess the situation

Resource. In the entire process of data mining, a computer with sufficient performance is required to calculate and process images.

Data. The data comes from the dataset publicly available on the Kaggle platform. The author obtained official data from the Food and Agriculture Organization of the United Nations (FAO), edited, and structured it. The data set not only includes the emissions of CO2, CH4, N2O and other gases, but also separates the data according to the emission sources of each country.

Risks. Some erroneous data may appear in this project, or incompatible data formats may cause some deviations in the results. For example, in a data set that counts the total greenhouse gas emissions of various countries around the world, there should not be negative emissions, which is contrary to common sense. So, to address these potential risks, I need to filter and clean my data. Find and remove or modify erroneous data before formal data mining of the dataset begins. At the same time, during the data mining process, the quality of the data used in the project is regularly reviewed and verified. Whenever possible, the dataset is checked after each operation on it.

Requirements, Assumptions, Constraints. In the process of our data-driven project, it's imperative to address several key facets. Firstly, we need to assess any legal or security implications stemming from our data sources, especially given potential risks associated with the data's origin and its edited nature. This assessment should be coupled with a clear alignment among stakeholders regarding the project's timeline and the desired format for presenting results. Secondly, it's vital to identify any associated costs, especially those related to data acquisition, and to have a clear understanding of the data's quality and potential assumptions. Lastly, ensuring the project's smooth execution means verifying we have the necessary permissions to access all data sources, being thorough in checking any legal constraints on data usage, and ensuring all potential expenses are within the project's budgetary constraints.

1.3 Determine data mining objectives

After a preliminary analysis of the data, it is determined that the first research objectives are:

- Use the data about the total amount of greenhouse gas emissions of each country to classify, look at the distribution map of emissions, and determine the responsibility of each country, so that each country understands how much responsibility they have in this global warming climate change crisis.

- According to the data analysis from 2000 to 2020, the change trend of various greenhouse gas emissions, determine which emission source emits the most greenhouse gases in each country, and analyze which one of each country emits the most greenhouse gases into the air.

- Classify the emission sources of greenhouse gases and formulate action plans for different emission sources according to the situation of each country, so as to better solve the problem of large-scale emission of greenhouse gases.(Zandalinas et al., 2021)

1.4 Produce a project plan

The overview plan for the study is as shown in the table below.(*IBM SPSS Modeler CRISP-DM Guide*, n.d.) The entire plan is divided into 8 steps, and the estimated time is from September 25, 2023, to October 13, 2023.

Table 1. project plan overview

Phase	Time	Resources	Risks
Step 1. Business and/or Situation understanding	9.25-9.26	All analysts A computer	
Step 2. Data understanding	9.27-9.28	All analysts A computer	Data problems Technology problems
Step 3. Data preparation	9.29-10.03	All analysts A computer	Data problems Technology problems
Step 4. Data transformation	10.04-10.06	All analysts A computer	Data problems Inability to find adequate model
Step 5. Data-mining method(s) selection	10.07-10.08	All analysts A computer	Technology problems
Step 6. Data-mining algorithm(s) selection	10.09-10.11	All analysts A computer	Technology problems
Step 7. Data Mining	10.12	All analysts A computer	Technology problems
Step 8. Interpretation	10.13	All analysts A computer	

2. Data understanding

2.1 Collect initial data

- **Existing data.** Existing data. This existing data comes from https://www.kaggle.com/datasets/justin2028/total-emissions-per-country-2000-2020. Statistics from 2000 to 2020 are included in the dataset.

  Kaggle is a platform for predictive modeling and analytics competitions. It allows users to find and publish datasets, explore and build models in a web-based data science environment, collaborate with other data scientists, and participate in competitions to solve data science challenges.

  According to the current target, the data content can meet the requirements, and the amount of data is large enough. This dataset classifies various greenhouse gases produced by different causes in countries around the world, such as greenhouse gases from crop straw in Afghanistan and their causes. At the same time, the dataset counts the amount of gas produced each year from 2000 to 2020.

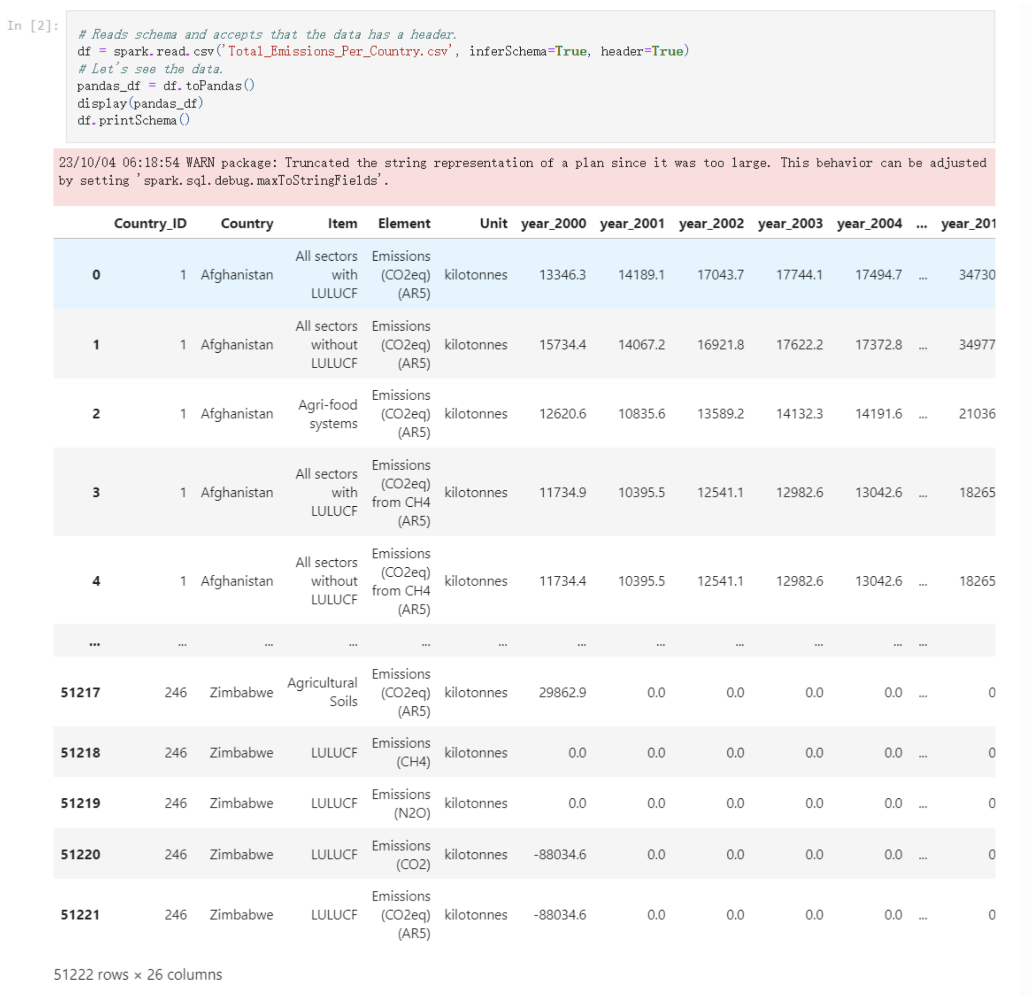
- **Purchased data.** At present, we do not consider paying for some data sets. There are enough public data sets available in Kaggle.com.

- **Additional data.** After in-depth research, if there is no data set that meets the requirements in Kaggle.com, plan to go to other free public data websites to find suitable data sets.

2.2 Describe the data

- **Amount of data.** The data set is an Excel file with 26 columns of data, which are Country_ID, Country, sources of greenhouse gases, causes of greenhouse gases, statistical units of greenhouse gases, and emissions for each year from 2000 to

2020. Among these attributes, the sources of greenhouse gases include crop residues, excrement, etc.; the causes of greenhouse gases include direct emissions and indirect emissions, etc.; the statistical unit of greenhouse gases is unified as kilotons. In the current data set, because the emission sources of each country's emission gases are classified, the range of the data level is 51222 pieces of data.



```
In [2]:
# Reads schema and accepts that the data has a header.
df = spark.read.csv('Total_Emissions_Per_Country.csv', inferSchema=True, header=True)
# Let's see the data.
pandas_df = df.toPandas()
display(pandas_df)
df.printSchema()
```

```
23/10/04 06:18:54 WARN package: Truncated the string representation of a plan since it was too large. This behavior can be adjusted
by setting 'spark.sql.debug.maxToStringFields'.
```

	Country_ID	Country	Item	Element	Unit	year_2000	year_2001	year_2002	year_2003	year_2004	...	year_201
0	1	Afghanistan	All sectors with LULUCF	Emissions (CO2eq) (AR5)	kilotonnes	13346.3	14189.1	17043.7	17744.1	17494.7	...	34730
1	1	Afghanistan	All sectors without LULUCF	Emissions (CO2eq) (AR5)	kilotonnes	15734.4	14067.2	16921.8	17622.2	17372.8	...	34977
2	1	Afghanistan	Agri-food systems	Emissions (CO2eq) (AR5)	kilotonnes	12620.6	10835.6	13589.2	14132.3	14191.6	...	21036
3	1	Afghanistan	All sectors with LULUCF	Emissions (CO2eq) from CH4 (AR5)	kilotonnes	11734.9	10395.5	12541.1	12982.6	13042.6	...	18265
4	1	Afghanistan	All sectors without LULUCF	Emissions (CO2eq) from CH4 (AR5)	kilotonnes	11734.4	10395.5	12541.1	12982.6	13042.6	...	18265
...	...	...	...	...	...	...	...	...	...	...	...	
51217	246	Zimbabwe	Agricultural Soils	Emissions (CO2eq) (AR5)	kilotonnes	29862.9	0.0	0.0	0.0	0.0	...	0
51218	246	Zimbabwe	LULUCF	Emissions (CH4)	kilotonnes	0.0	0.0	0.0	0.0	0.0	...	0
51219	246	Zimbabwe	LULUCF	Emissions (N2O)	kilotonnes	0.0	0.0	0.0	0.0	0.0	...	0
51220	246	Zimbabwe	LULUCF	Emissions (CO2)	kilotonnes	-88034.6	0.0	0.0	0.0	0.0	...	0
51221	246	Zimbabwe	LULUCF	Emissions (CO2eq) (AR5)	kilotonnes	-88034.6	0.0	0.0	0.0	0.0	...	0

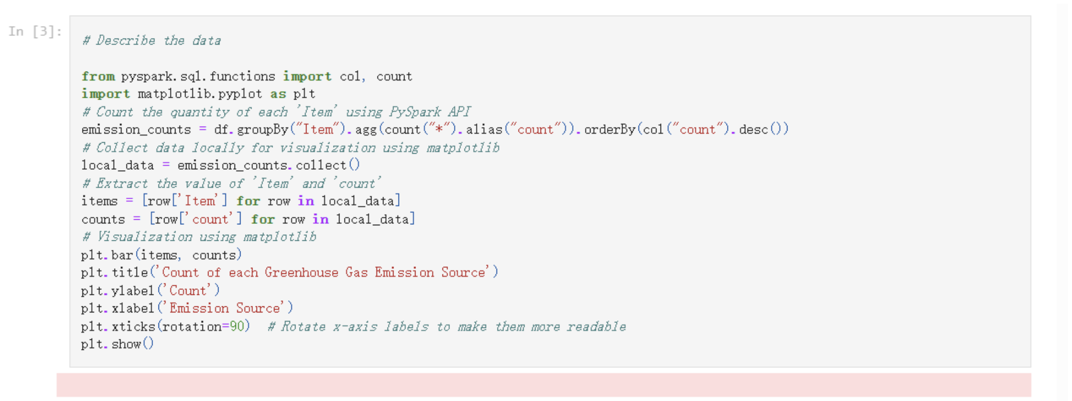
51222 rows × 26 columns

Figure 2.1 The overview of dataset

Country_ID: A sequential number that uniquely identifies each country. This field does not exist in the original data set, and it is added through Excel data preprocessing. The data type is integer, ranging from 1 to 51222.

Country: This field contains the name of each country being counted. The data type is character type.

Item: This field contains sources of greenhouse gas emissions. For example, 41 sources are included, among which IPCC Agriculture appears the most, which means that it may be the most common source of greenhouse gas emissions in all countries in the world. The data type is character type.



```
In [3]:  # Describe the data

         from pyspark.sql.functions import col, count
         import matplotlib.pyplot as plt
         # Count the quantity of each 'Item' using PySpark API
         emission_counts = df.groupBy("Item").agg(count("*").alias("count")).orderBy(col("count").desc())
         # Collect data locally for visualization using matplotlib
         local_data = emission_counts.collect()
         # Extract the value of 'Item' and 'count'
         items = [row['Item'] for row in local_data]
         counts = [row['count'] for row in local_data]
         # Visualization using matplotlib
         plt.bar(items, counts)
         plt.title('Count of each Greenhouse Gas Emission Source')
         plt.ylabel('Count')
         plt.xlabel('Emission Source')
         plt.xticks(rotation=90)  # Rotate x-axis labels to make them more readable
         plt.show()
```

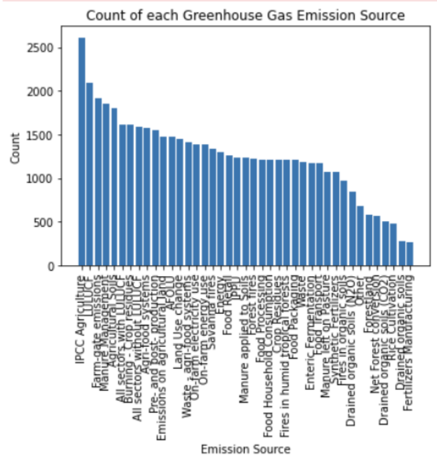


Figure 2.2 The number of items statistics

Element: This field contains the names of various greenhouse gases. The data type is character type.

Unit: This field describes the unit of greenhouse gas statistics in this dataset. The data type is character type.

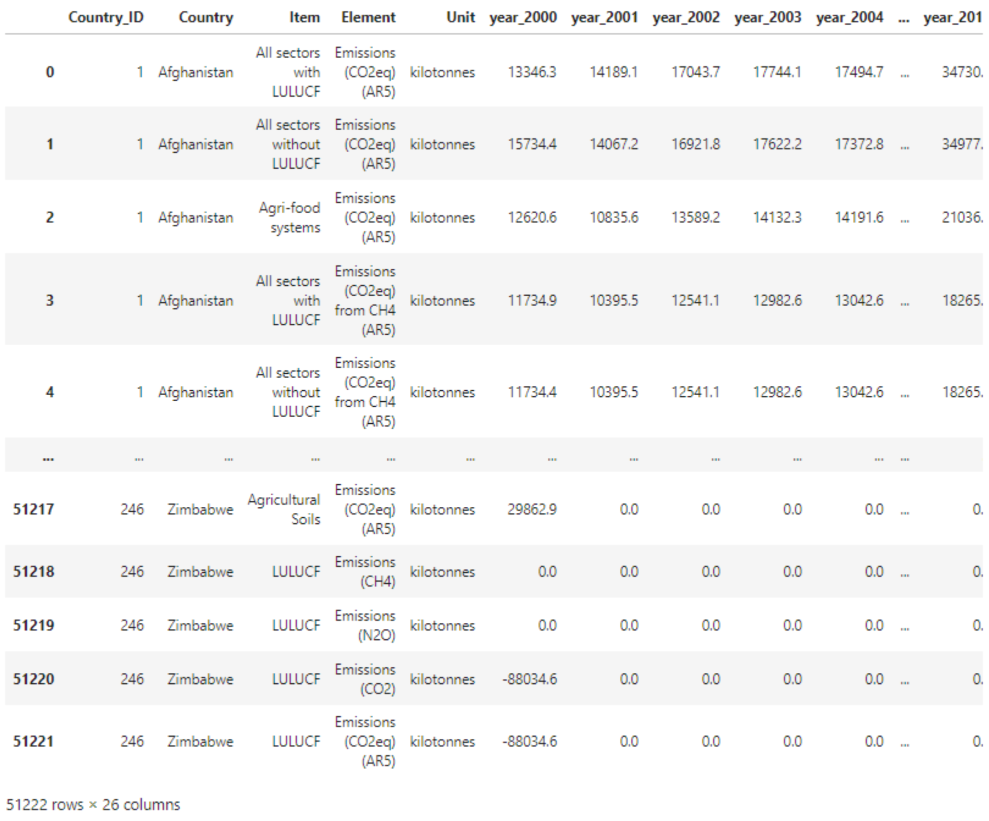
year_2000 to year_2020(21 fields): A total of 21 fields are used to display the greenhouse gas emissions produced by each emission source from 2000 to 2020.

- **Value types.** The data in the dataset is mainly composed of character and numeric data, covering the name of the country, emission source, type and unit of emission gas, and gas emission data from 2000 to 2020.

2.3 Explore the data

Conduct preliminary exploration of the data set in AWS jupyter notebook and analyze some key fields in the data set through tables and charts, which will provide good help for subsequent data mining and other stages.

Table. First, you can view the status of the entire data set through Table. What you see here is the initial state of the data set, where there may be some data that affects the quality of the data set, such as empty data, abnormal data, etc.

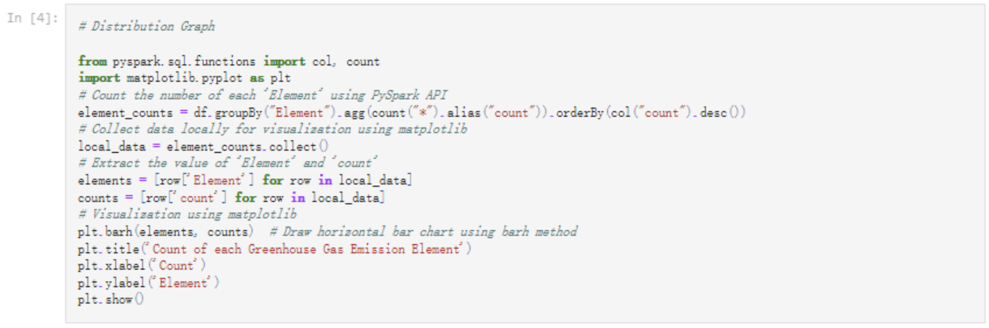


	Country_ID	Country	Item	Element	Unit	year_2000	year_2001	year_2002	year_2003	year_2004	...	year_201
0	1	Afghanistan	All sectors with LULUCF	Emissions (CO2eq) (AR5)	kilotonnes	13346.3	14189.1	17043.7	17744.1	17494.7	...	34730.
1	1	Afghanistan	All sectors without LULUCF	Emissions (CO2eq) (AR5)	kilotonnes	15734.4	14067.2	16921.8	17622.2	17372.8	...	34977.
2	1	Afghanistan	Agri-food systems	Emissions (CO2eq) (AR5)	kilotonnes	12620.6	10835.6	13589.2	14132.3	14191.6	...	21036.
3	1	Afghanistan	All sectors with LULUCF	Emissions (CO2eq) from CH4 (AR5)	kilotonnes	11734.9	10395.5	12541.1	12982.6	13042.6	...	18265.
4	1	Afghanistan	All sectors without LULUCF	Emissions (CO2eq) from CH4 (AR5)	kilotonnes	11734.4	10395.5	12541.1	12982.6	13042.6	...	18265.
...	...	...	...	...	...	...	...	...	...	...	...	
51217	246	Zimbabwe	Agricultural Soils	Emissions (CO2eq) (AR5)	kilotonnes	29862.9	0.0	0.0	0.0	0.0	...	0.
51218	246	Zimbabwe	LULUCF	Emissions (CH4)	kilotonnes	0.0	0.0	0.0	0.0	0.0	...	0.
51219	246	Zimbabwe	LULUCF	Emissions (N2O)	kilotonnes	0.0	0.0	0.0	0.0	0.0	...	0.
51220	246	Zimbabwe	LULUCF	Emissions (CO2)	kilotonnes	-88034.6	0.0	0.0	0.0	0.0	...	0.
51221	246	Zimbabwe	LULUCF	Emissions (CO2eq) (AR5)	kilotonnes	-88034.6	0.0	0.0	0.0	0.0	...	0.

51222 rows × 26 columns

Figure 2.3 data Table

Distribution Graph. According to different GHG classifications, look at the total emissions of each GHG from 2000 to 2020, which can help subsequent analysis of the emission sources most in need of governance.



```
In [4]:
# Distribution Graph

from pyspark.sql.functions import col, count
import matplotlib.pyplot as plt
# Count the number of each 'Element' using PySpark API
element_counts = df.groupBy("Element").agg(count("*").alias("count")).orderBy(col("count").desc())
# Collect data locally for visualization using matplotlib
local_data = element_counts.collect()
# Extract the value of 'Element' and 'count'
elements = [row['Element'] for row in local_data]
counts = [row['count'] for row in local_data]
# Visualization using matplotlib
plt.barh(elements, counts)  # Draw horizontal bar chart using barh method
plt.title('Count of each Greenhouse Gas Emission Element')
plt.xlabel('Count')
plt.ylabel('Element')
plt.show()
```

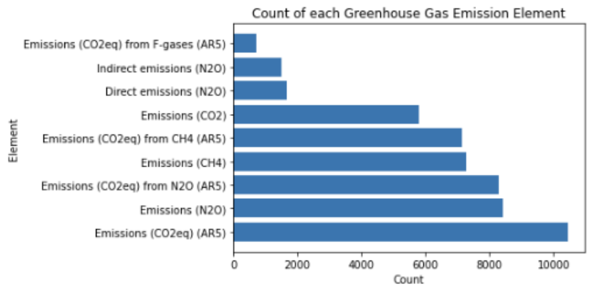
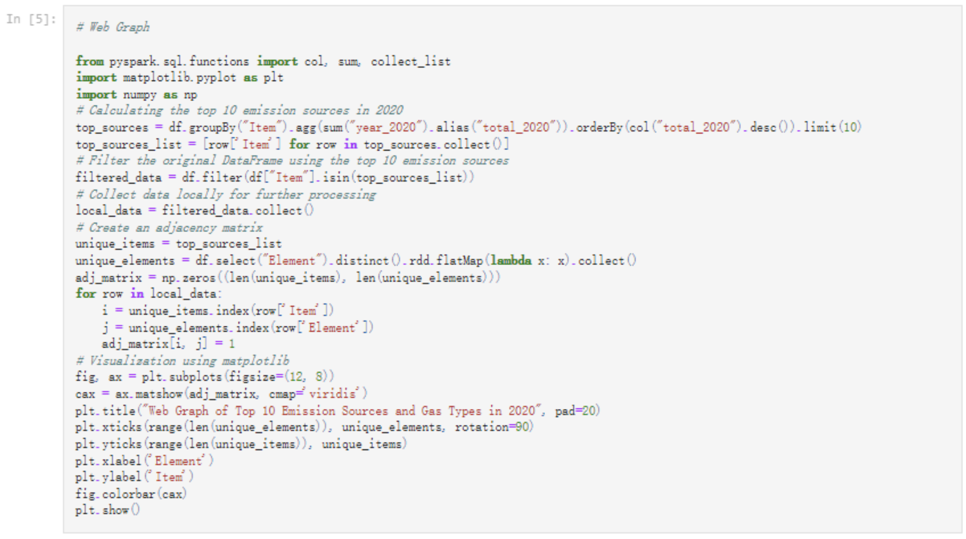


Figure 2.4 Distribution Graph

Web Graph. By connecting emission sources and emission gas types through Web Graph, it is possible to see more clearly which emission sources produce each greenhouse gas.



```
In [5]:
# Web Graph

from pyspark.sql.functions import col, sum, collect_list
import matplotlib.pyplot as plt
import numpy as np
# Calculating the top 10 emission sources in 2020
top_sources = df.groupBy("Item").agg(sum("year_2020").alias("total_2020")).orderBy(col("total_2020").desc()).limit(10)
top_sources_list = [row['Item'] for row in top_sources.collect()]
# Filter the original DataFrame using the top 10 emission sources
filtered_data = df.filter(df["Item"].isin(top_sources_list))
# Collect data locally for further processing
local_data = filtered_data.collect()
# Create an adjacency matrix
unique_items = top_sources_list
unique_elements = df.select("Element").distinct().rdd.flatMap(lambda x: x).collect()
adj_matrix = np.zeros((len(unique_items), len(unique_elements)))
for row in local_data:
    i = unique_items.index(row['Item'])
    j = unique_elements.index(row['Element'])
    adj_matrix[i, j] = 1
# Visualization using matplotlib
fig, ax = plt.subplots(figsize=(12, 8))
cax = ax.matshow(adj_matrix, cmap='viridis')
plt.title("Web Graph of Top 10 Emission Sources and Gas Types in 2020", pad=20)
plt.xticks(range(len(unique_elements)), unique_elements, rotation=90)
plt.yticks(range(len(unique_items)), unique_items)
plt.xlabel('Element')
plt.ylabel('Item')
fig.colorbar(cax)
plt.show()
```

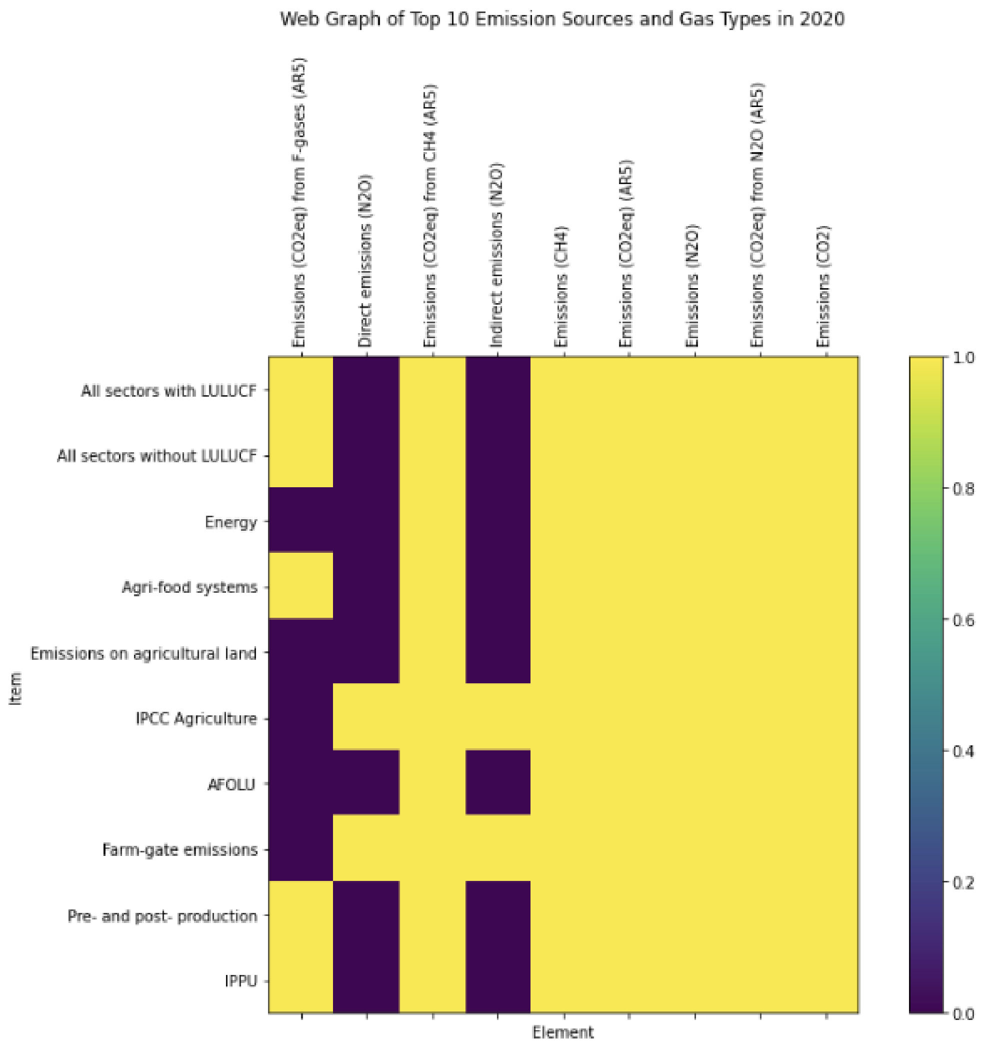


Figure 2.5 Web Graph

2.4 Verify the data quality

After exploring the data, you can use the pyspark and matplotlib libraries to check the quality of the dataset, such as missing values, etc.

In the quality of the dataset, it is shown that there are no missing values and the presence of duplicate rows. And year_2000 to year_2020 in the fields are all constant columns.

```
In [6]:  # Verify the data quality

         from pyspark.sql.functions import col, count, countDistinct, when
         import matplotlib.pyplot as plt
         import seaborn as sns
         missing_percentage = df.select([(count(when(col(c).isNull(), c)) / df.count()).alias(c) for c in df.columns])
         missing_data = missing_percentage.collect()[0]
         duplicates = df.count() - df.dropDuplicates().count()
         unique_counts = df.agg(*(countDistinct(col(c)).alias(c) for c in df.columns)).collect()[0]
         # Visualization using matplotlib and seaborn
         plt.figure(figsize=(15, 10))
         # Missing value percentage plot
         plt.subplot(2, 2, 1)
         plt.bar(df.columns, [missing_data[c] for c in df.columns], color='skyblue')
         plt.title('Percentage of Missing Values by Column')
         plt.ylabel('Percentage')
         plt.xlabel('Columns')
         plt.xticks(rotation=90)
         # Repeating line graph
         plt.subplot(2, 2, 2)
         sns.barplot(x=['Duplicates', 'Unique'], y=[duplicates, df.count() - duplicates], palette='pastel')
         plt.title('Duplicate Rows in Data')
         plt.ylabel('Count')
         # Plot of number of unique values per column
         plt.subplot(2, 2, 3)
         sns.barplot(x=list(df.columns), y=[unique_counts[c] for c in df.columns], palette='pastel')
         plt.title('Number of Unique Values by Column')
         plt.ylabel('Unique Count')
         plt.xlabel('Columns')
         plt.xticks(rotation=90)
         plt.tight_layout()
         plt.show()
```

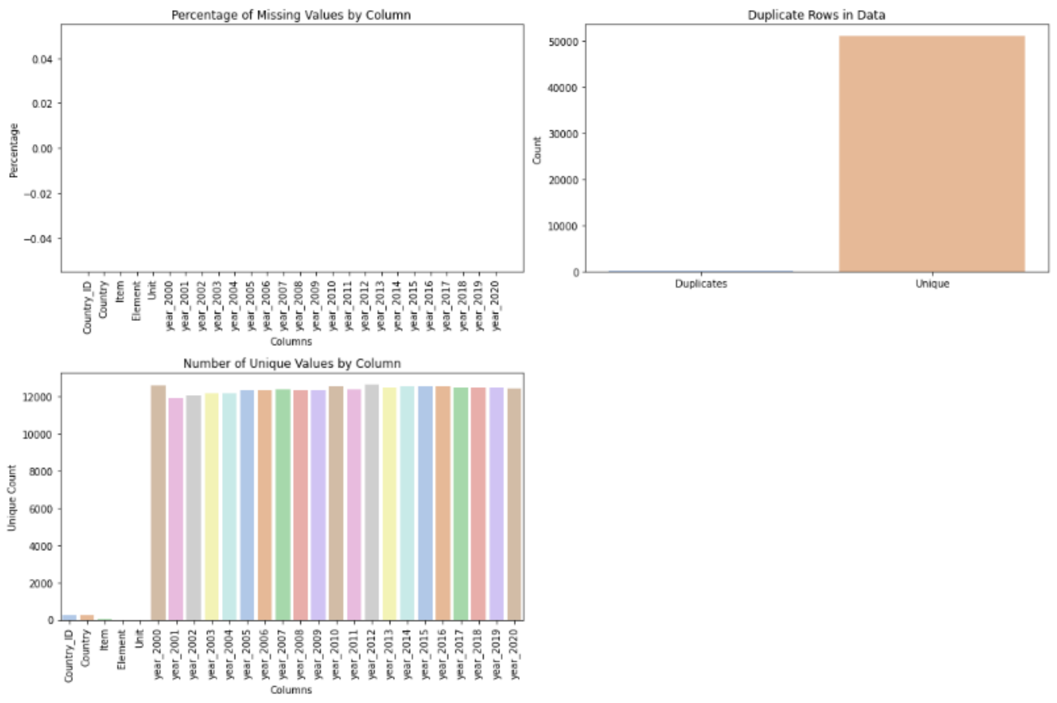


Figure 2.6 Quality of Dataset

3. Data preparation

3.1 Selecting data

Selecting items. Given the objectives of our study, it's crucial to select data records related to global greenhouse gas emissions. These records provide detailed insights into greenhouse gas emissions by country, source, and type from 2000 to 2020.

Selecting attributes. We will focus on specific attributes or fields related to greenhouse gas emission volumes, emission sources, and emission types. Therefore, we selected the "Country", "Item", "Element" and "year_2020" fields in the data set as the main research objects.

Size. The data set has 26 fields and 51222 pieces of data, which provides a large enough sample.

Quality. Through data exploration, it is found that the data quality is high, and there are not too many missing values and outliers, so stable data analysis can be performed.

Data sources. The primary data source is a dataset from Kaggle, which has been edited and structured by the author. This dataset is supplemented by official data from the Food and Agriculture Organization of the United Nations (FAO).

3.2 Cleaning data

3.2.1 Missing data

Possible problems: Some missing values in the data set will lead to deviations in the results of subsequent data analysis.

```
from pyspark.sql.functions import col, count, countDistinct, when
import matplotlib.pyplot as plt
import seaborn as sns
missing_percentage = df.select([(count(when(col(c).isNull(), c)) / df.count()).alias(c) for c in df.columns])
missing_data = missing_percentage.collect()[0]
```

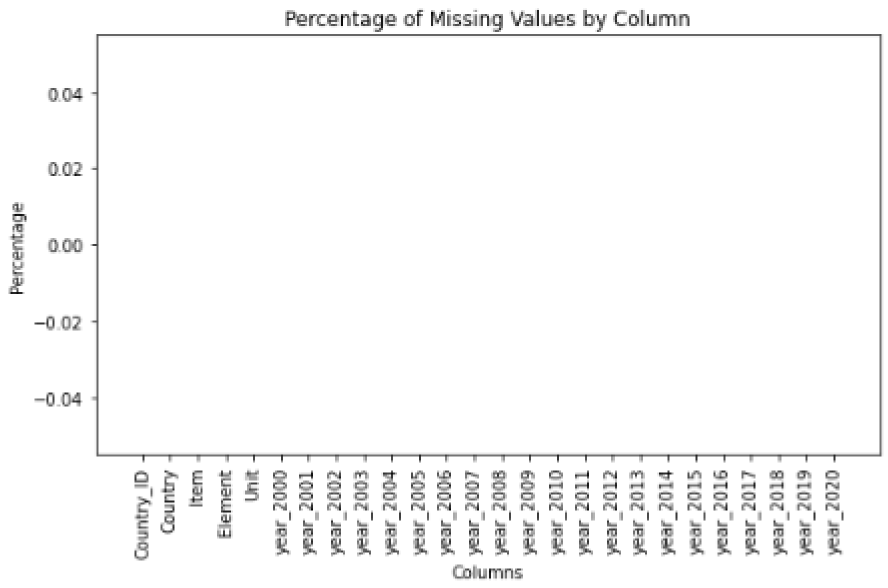


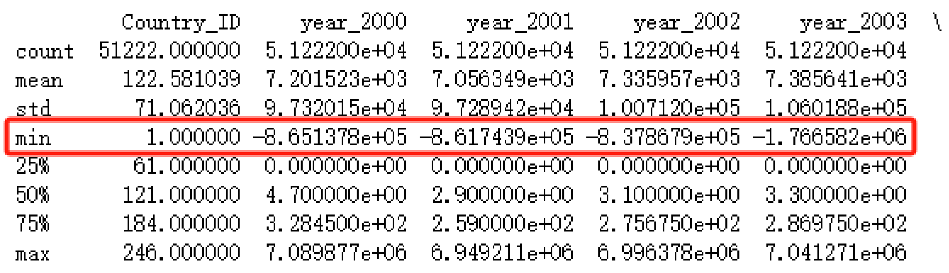
Figure 3.1 Missing Values by Column

Possible action: Possible actions: Use the corresponding function in pyspark to examine the dataset and fill missing values by imputation.

Practical action: Find and calculate the Missing Value according to the isNull method in pyspark. The data set has a total of 26 fields and the missing value percentage of each field is zero. Therefore, there is no missing value dataset in the dataset and no further filling is required.

3.2.2 Data errors in Values field

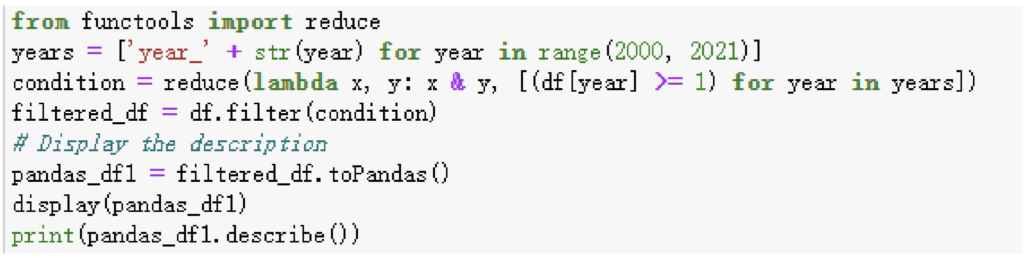
Problems that arise: Some data errors can cause large fluctuations in the data. The statistics in this data set are mainly greenhouse gas emissions, which should not have negative values based on common sense. However, a review of the data revealed that the data set contained many data with negative emissions, which is clearly an error in the data.



```
         Country_ID     year_2000     year_2001     year_2002     year_2003  \
count  51222.000000  5.122200e+04  5.122200e+04  5.122200e+04  5.122200e+04
mean     122.581039  7.201523e+03  7.056349e+03  7.335957e+03  7.385641e+03
std       71.062036  9.732015e+04  9.728942e+04  1.007120e+05  1.060188e+05
min        1.000000 -8.651378e+05 -8.617439e+05 -8.378679e+05 -1.766582e+06
25%       61.000000  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
50%      121.000000  4.700000e+00  2.900000e+00  3.100000e+00  3.300000e+00
75%      184.000000  3.284500e+02  2.590000e+02  2.756750e+02  2.869750e+02
max      246.000000  7.089877e+06  6.949211e+06  6.996378e+06  7.041271e+06
```

Figure 3.2 Data errors

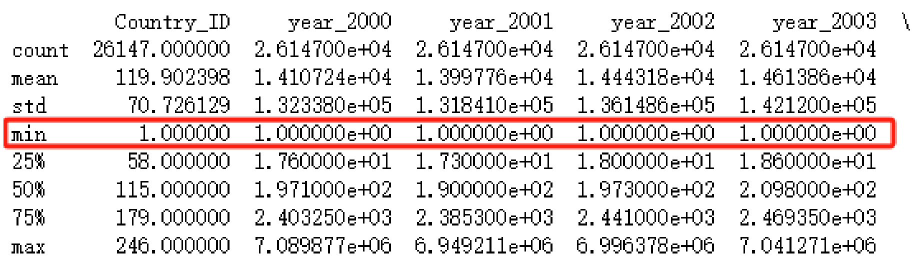
Practical action: The first step in pyspark is to traverse all emissions data from 2000 to 2020. Secondly, use the “functools” method of the “reduce” package to filter the data with negative emissions in the data set. The filter condition is that fields from “year_2000” to “year_2020” are greater than or equal to 1. Because greenhouse gas emissions from a certain source may not exist in some countries, but the emissions cannot be negative.



```python
from functools import reduce
years = ['year_' + str(year) for year in range(2000, 2021)]
condition = reduce(lambda x, y: x & y, [(df[year] >= 1) for year in years])
filtered_df = df.filter(condition)
# Display the description
pandas_df1 = filtered_df.toPandas()
display(pandas_df1)
print(pandas_df1.describe())
```

Figure 3.3 Filter and exclude data

Result: After filtering through python, you can see that the emission range in the data set starts from 1. Therefore, the dataset has some erroneous data.



```
         Country_ID     year_2000     year_2001     year_2002     year_2003  \
count  26147.000000  2.614700e+04  2.614700e+04  2.614700e+04  2.614700e+04
mean     119.902398  1.410724e+04  1.399776e+04  1.444318e+04  1.461386e+04
std       70.726129  1.323380e+05  1.318410e+05  1.361486e+05  1.421200e+05
min        1.000000  1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00
25%       58.000000  1.760000e+01  1.730000e+01  1.800000e+01  1.860000e+01
50%      115.000000  1.971000e+02  1.900000e+02  1.973000e+02  2.098000e+02
75%      179.000000  2.403250e+03  2.385300e+03  2.441000e+03  2.469350e+03
max      246.000000  7.089877e+06  6.949211e+06  6.996378e+06  7.041271e+06
```

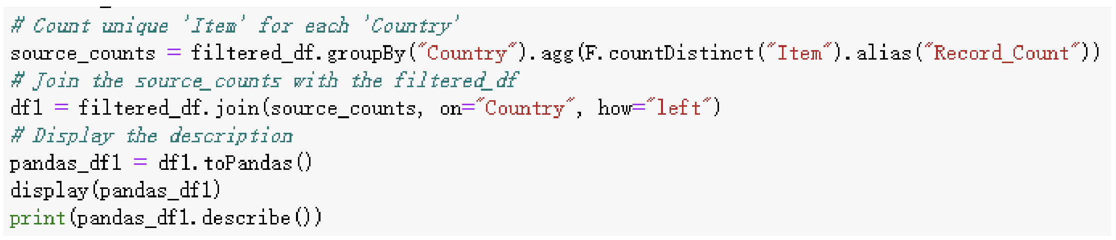
Figure 3.4 Result of Select

3.3 Constructing new data

3.3.1 Add a “Record_Count” field for emission source quantity statistics

Current problem: The Item field in the data set counts the sources of greenhouse gas emissions in each country. But if you want a more in-depth analysis of how many sources emit greenhouse gases in each country, you first need to count how many sources each country has.

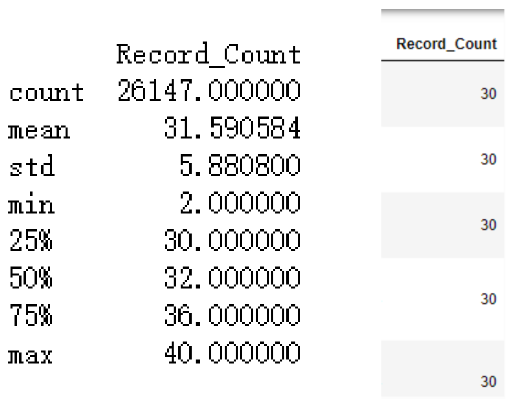
Practical action: Through data set analysis, a country's emission sources may produce a variety of greenhouse gases. Therefore, it is necessary to use "groupBy" according to the name of each emission source to group them, and then count the number of emission sources. Therefore, use pyspark's "groupBy" method for grouping in AWS. Select Country as the key field and count by "Record_count" field.



```
# Count unique 'Item' for each 'Country'
source_counts = filtered_df.groupBy("Country").agg(F.countDistinct("Item").alias("Record_Count"))
# Join the source_counts with the filtered_df
df1 = filtered_df.join(source_counts, on="Country", how="left")
# Display the description
pandas_df1 = df1.toPandas()
display(pandas_df1)
print(pandas_df1.describe())
```

Figure 3.5 Emission source quantity statistics

Result: The number of emission sources for each country is displayed through the Table. With this data, each country can be graded according to the number of sources in the next step.



```
       Record_Count
count  26147.000000
mean      31.590584
std        5.880800
min        2.000000
25%       30.000000
50%       32.000000
75%       36.000000
max       40.000000
```

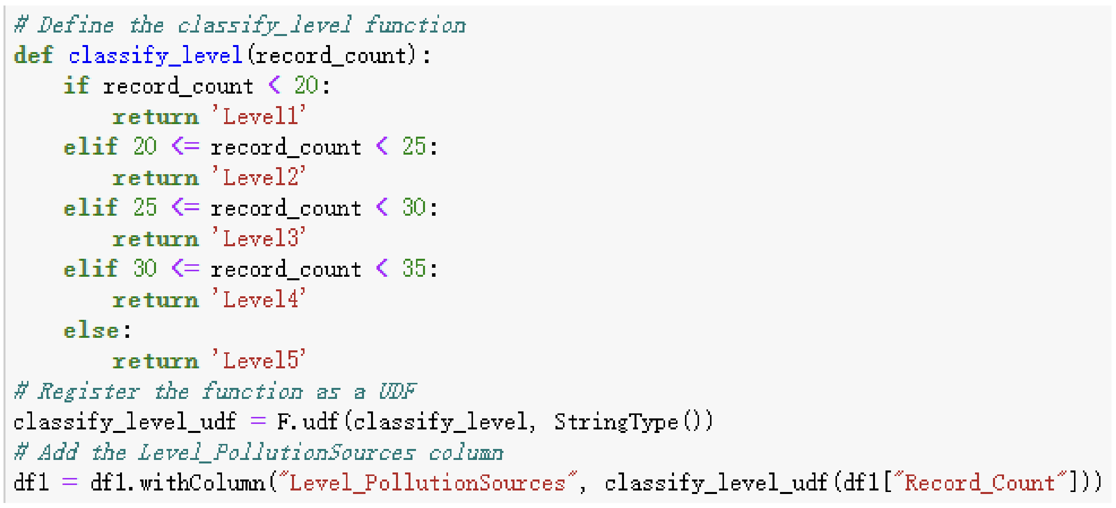
Record_Count
30
30
30
30
30

Figure 3.6 Result of Aggregate

3.3.2 Add a "Level_PollutionSources" field to classify the source class

The problem is that a country's greenhouse gas emissions may be related to the number of sources, but there is no area where it is clear where each country is at.

Practical operation: Define a "classify_level" function in pyspark to divide the number of sources per category. Classified statistics are based on the number of emission sources in each country and are divided into levels. The higher a country's rating, the greater the number of emission sources.



```python
# Define the classify_level function
def classify_level(record_count):
    if record_count < 20:
        return 'Level1'
    elif 20 <= record_count < 25:
        return 'Level2'
    elif 25 <= record_count < 30:
        return 'Level3'
    elif 30 <= record_count < 35:
        return 'Level4'
    else:
        return 'Level5'
# Register the function as a UDF
classify_level_udf = F.udf(classify_level, StringType())
# Add the Level_PollutionSources column
df1 = df1.withColumn("Level_PollutionSources", classify_level_udf(df1["Record_Count"]))
```

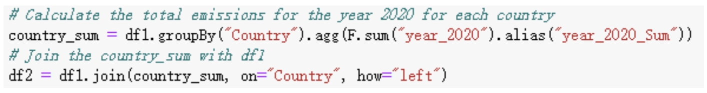
Figure 3.7 add “Level_PollutionSources” field

Result: The complete data set with the new fields added.

3.3.3 Add a "year_2020_Sum" field to check the ranking of national emissions

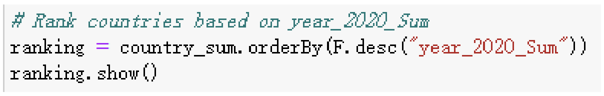
The problem at hand: the size of a country's total greenhouse gas emissions is directly related to the division of responsibilities, and it is necessary to know which countries emit the most. But there is currently no field that clearly shows the total amount of greenhouse gas emissions in each country.

Practical operation: Use python's “groupBy” and “sum” functions to sum the data statistics of the most recent year (2020) by each country and sort them in descending order.



```python
# Calculate the total emissions for the year 2020 for each country
country_sum = df1.groupBy("Country").agg(F.sum("year_2020").alias("year_2020_Sum"))
# Join the country_sum with df1
df2 = df1.join(country_sum, on="Country", how="left")
```

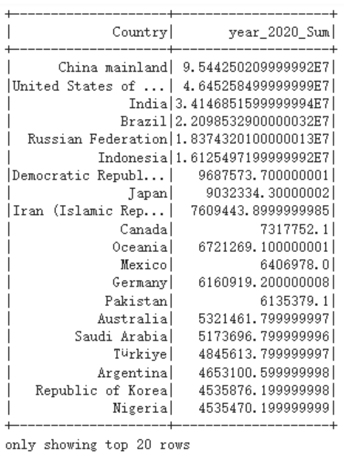
Figure 3.9 Sum of year_2020



```python
# Rank countries based on year_2020_Sum
ranking = country_sum.orderBy(F.desc("year_2020_Sum"))
ranking.show()
```

Figure 3.10 Descending to sort

Result: The final result shows the total amount of greenhouse gases emitted by each country in 2020, in descending order.



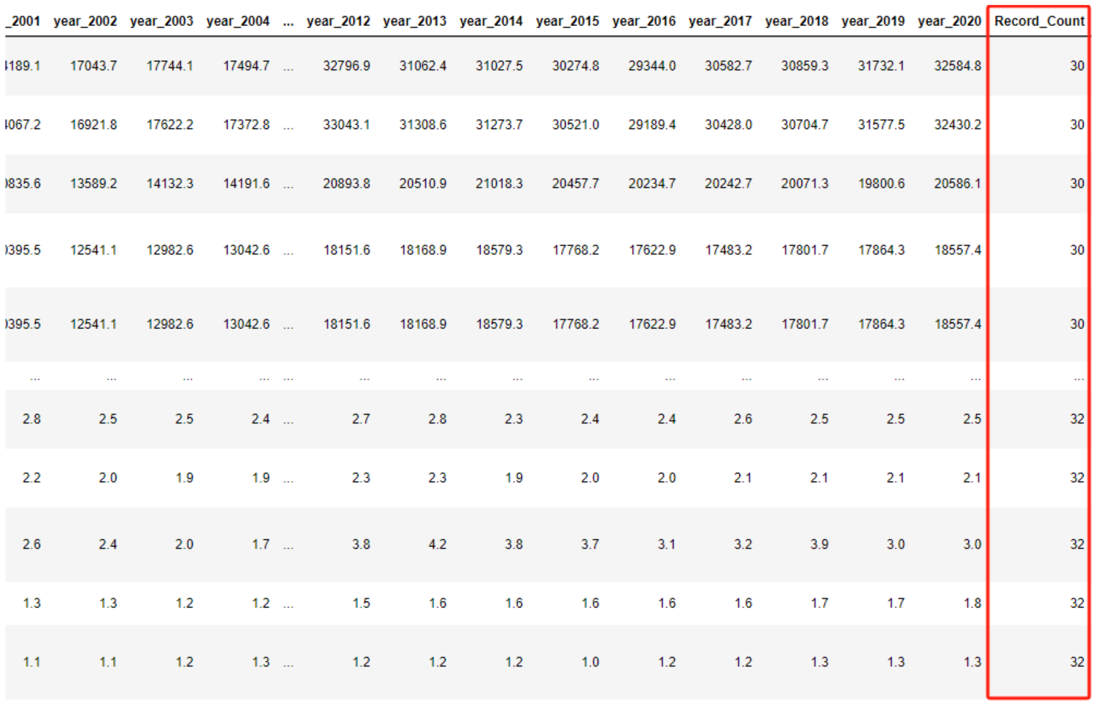
```
+--------------------+--------------------+
|             Country|       year_2020_Sum|
+--------------------+--------------------+
|      China mainland| 9.544250209999992E7|
|United States of ...| 4.645258499999999E7|
|               India|3.4146851599999994E7|
|              Brazil|2.2098532900000032E7|
|  Russian Federation|1.8374320100000013E7|
|           Indonesia|1.6125497199999992E7|
|Democratic Republ...|   9687573.700000001|
|               Japan|    9032334.30000002|
|Iran (Islamic Rep...|  7609443.8999999985|
|              Canada|           7317752.1|
|             Oceania|   6721269.100000001|
|              Mexico|           6406978.0|
|             Germany|   6160919.200000008|
|            Pakistan|           6135379.1|
|           Australia|   5321461.799999997|
|        Saudi Arabia|   5173696.799999996|
|             Türkiye|   4845613.799999997|
|           Argentina|   4653100.599999998|
|   Republic of Korea|   4535876.199999998|
|             Nigeria|   4535470.199999999|
+--------------------+--------------------+
only showing top 20 rows
```

Figure 3.11 “year_2020_Sum” field

3.4 Integrating data

3.4.1 Merge data field "Record_Count"

Combining the original data with the added "Record_Count" field complements the statistical parameters for the number of emission sources in each country required for the original dataset. At the same time, the emission source quantity class is graded.

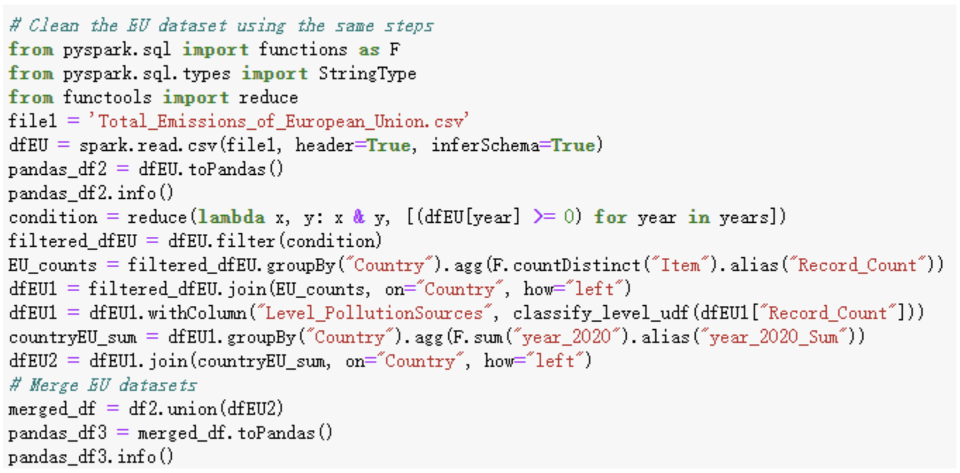


_2001	year_2002	year_2003	year_2004	...	year_2012	year_2013	year_2014	year_2015	year_2016	year_2017	year_2018	year_2019	year_2020	Record_Count
189.1	17043.7	17744.1	17494.7	...	32796.9	31062.4	31027.5	30274.8	29344.0	30582.7	30859.3	31732.1	32584.8	30
067.2	16921.8	17622.2	17372.8	...	33043.1	31308.6	31273.7	30521.0	29189.4	30428.0	30704.7	31577.5	32430.2	30
835.6	13589.2	14132.3	14191.6	...	20893.8	20510.9	21018.3	20457.7	20234.7	20242.7	20071.3	19800.6	20586.1	30
395.5	12541.1	12982.6	13042.6	...	18151.6	18168.9	18579.3	17768.2	17622.9	17483.2	17801.7	17864.3	18557.4	30
395.5	12541.1	12982.6	13042.6	...	18151.6	18168.9	18579.3	17768.2	17622.9	17483.2	17801.7	17864.3	18557.4	30
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2.8	2.5	2.5	2.4	...	2.7	2.8	2.3	2.4	2.4	2.6	2.5	2.5	2.5	32
2.2	2.0	1.9	1.9	...	2.3	2.3	1.9	2.0	2.0	2.1	2.1	2.1	2.1	32
2.6	2.4	2.0	1.7	...	3.8	4.2	3.8	3.7	3.1	3.2	3.9	3.0	3.0	32
1.3	1.3	1.2	1.2	...	1.5	1.6	1.6	1.6	1.6	1.6	1.7	1.7	1.8	32
1.1	1.1	1.2	1.3	...	1.2	1.2	1.2	1.0	1.2	1.2	1.3	1.3	1.3	32

Figure 3.12 Merge data field "Record_Count"

3.4.2 Merge the new dataset, which contains the European Union

Through my understanding of the dataset, I found that many countries in Europe caused very little pollution. At the same time, it is thought that the management and planning of European countries by the EU system can better promote the responsibility and pollution source management of most European countries. This requires access to data on greenhouse gas emissions in the EU as a whole, including 27 European countries. The key next step is to perform the same data cleansing operation on the EU dataset as the original dataset.

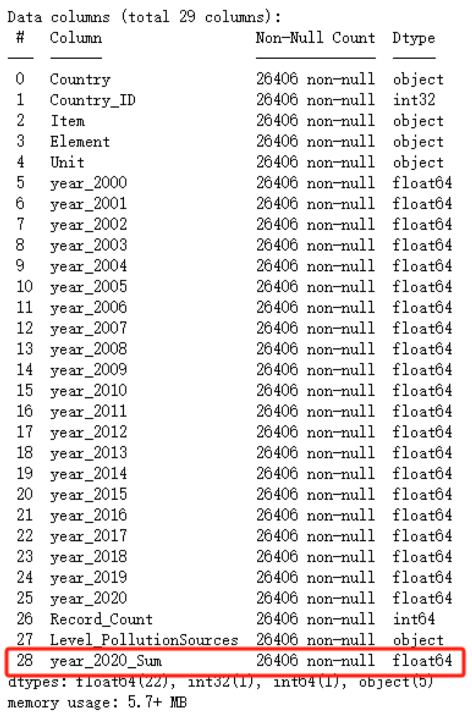


```
# Clean the EU dataset using the same steps
from pyspark.sql import functions as F
from pyspark.sql.types import StringType
from functools import reduce
file1 = 'Total_Emissions_of_European_Union.csv'
dfEU = spark.read.csv(file1, header=True, inferSchema=True)
pandas_df2 = dfEU.toPandas()
pandas_df2.info()
condition = reduce(lambda x, y: x & y, [(dfEU[year] >= 0) for year in years])
filtered_dfEU = dfEU.filter(condition)
EU_counts = filtered_dfEU.groupBy("Country").agg(F.countDistinct("Item").alias("Record_Count"))
dfEU1 = filtered_dfEU.join(EU_counts, on="Country", how="left")
dfEU1 = dfEU1.withColumn("Level_PollutionSources", classify_level_udf(dfEU1["Record_Count"]))
countryEU_sum = dfEU1.groupBy("Country").agg(F.sum("year_2020").alias("year_2020_Sum"))
dfEU2 = dfEU1.join(countryEU_sum, on="Country", how="left")
# Merge EU datasets
merged_df = df2.union(dfEU2)
pandas_df3 = merged_df.toPandas()
pandas_df3.info()
```

Figure 3.13 Merge the new dataset

3.4.3 Merge data field "year_2020_Sum"

After cleaning, merge with the original dataset. Once the combined dataset is obtained, the total 2020 emissions of all countries are counted and ranked, and the final consolidated dataset will be used for subsequent steps.



```
Data columns (total 29 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Country                 26406 non-null  object
 1   Country_ID              26406 non-null  int32
 2   Item                    26406 non-null  object
 3   Element                 26406 non-null  object
 4   Unit                    26406 non-null  object
 5   year_2000               26406 non-null  float64
 6   year_2001               26406 non-null  float64
 7   year_2002               26406 non-null  float64
 8   year_2003               26406 non-null  float64
 9   year_2004               26406 non-null  float64
 10  year_2005               26406 non-null  float64
 11  year_2006               26406 non-null  float64
 12  year_2007               26406 non-null  float64
 13  year_2008               26406 non-null  float64
 14  year_2009               26406 non-null  float64
 15  year_2010               26406 non-null  float64
 16  year_2011               26406 non-null  float64
 17  year_2012               26406 non-null  float64
 18  year_2013               26406 non-null  float64
 19  year_2014               26406 non-null  float64
 20  year_2015               26406 non-null  float64
 21  year_2016               26406 non-null  float64
 22  year_2017               26406 non-null  float64
 23  year_2018               26406 non-null  float64
 24  year_2019               26406 non-null  float64
 25  year_2020               26406 non-null  float64
 26  Record_Count            26406 non-null  int64
 27  Level_PollutionSources  26406 non-null  object
 28  year_2020_Sum           26406 non-null  float64
dtypes: float64(22), int32(1), int64(1), object(5)
memory usage: 5.7+ MB
```

Figure 3.14 Merge data field "year_2020_Sum"

3.5 Formatting data

Rules that define the format and data content of the new field

In the newly added field "Level_PollutionResources", according to the rule:

Record_Count >= 35 --> level 5

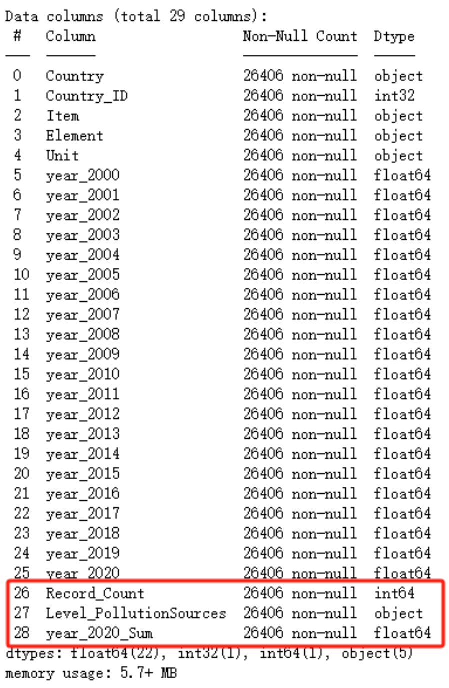
Record_Count >= 30 and Record_Count < 35 --> level 4

Record_Count >= 25 and Record_Count < 30 --> level 3

Record_Count >= 20 and Record_Count < 25 --> level 2

Record_Count < 20 --> level 1

The number of emission sources in each country is divided into levels, according to different levels can more easily confirm the approximate number of emission sources in the country, the higher the level, the higher the number of pollution sources in the country.



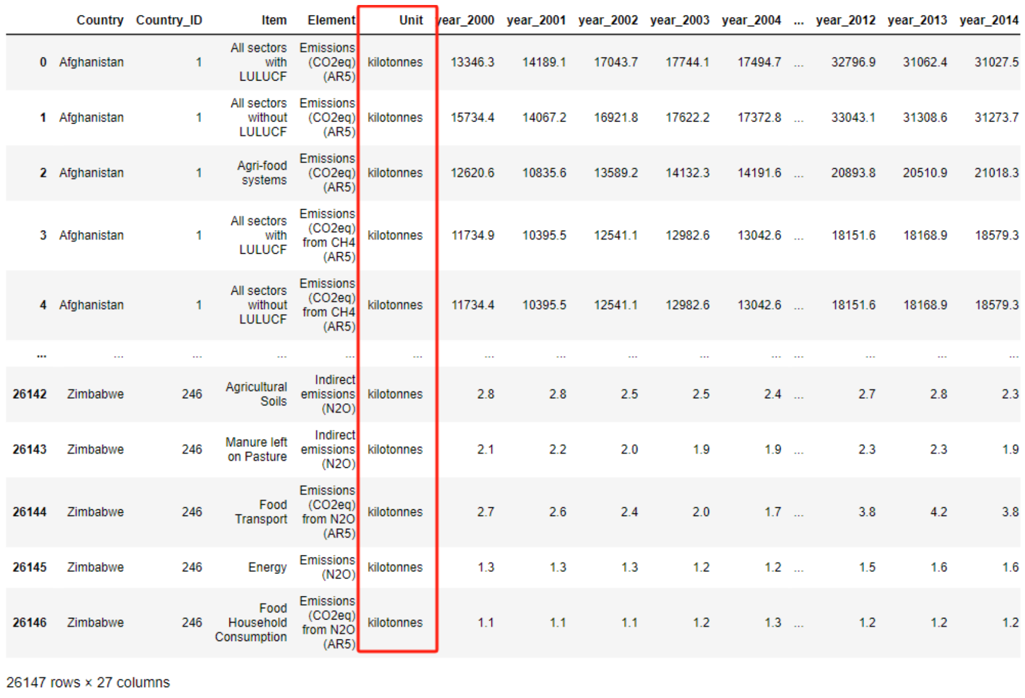
```
Data columns (total 29 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Country                26406 non-null  object
 1   Country_ID             26406 non-null  int32
 2   Item                   26406 non-null  object
 3   Element                26406 non-null  object
 4   Unit                   26406 non-null  object
 5   year_2000              26406 non-null  float64
 6   year_2001              26406 non-null  float64
 7   year_2002              26406 non-null  float64
 8   year_2003              26406 non-null  float64
 9   year_2004              26406 non-null  float64
 10  year_2005              26406 non-null  float64
 11  year_2006              26406 non-null  float64
 12  year_2007              26406 non-null  float64
 13  year_2008              26406 non-null  float64
 14  year_2009              26406 non-null  float64
 15  year_2010              26406 non-null  float64
 16  year_2011              26406 non-null  float64
 17  year_2012              26406 non-null  float64
 18  year_2013              26406 non-null  float64
 19  year_2014              26406 non-null  float64
 20  year_2015              26406 non-null  float64
 21  year_2016              26406 non-null  float64
 22  year_2017              26406 non-null  float64
 23  year_2018              26406 non-null  float64
 24  year_2019              26406 non-null  float64
 25  year_2020              26406 non-null  float64
 26  Record_Count           26406 non-null  int64
 27  Level_PollutionSources 26406 non-null  object
 28  year_2020_Sum          26406 non-null  float64
dtypes: float64(22), int32(1), int64(1), object(5)
memory usage: 5.7+ MB
```

Figure 3.15 Integration Dataset

4. Data transformation

4.1 Data reduction

Problem: In the dataset, the "Unit" field only represents the unit of gas emissions, which is not meaningful for data analysis and data mining and may not be needed for modeling. Therefore, redundant fields should be excluded to improve subsequent operating costs and efficiency and reduce data dimensions.

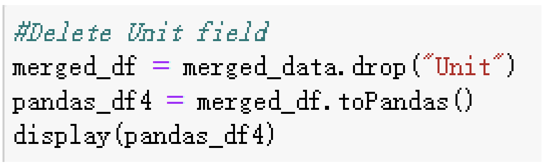


	Country	Country_ID	Item	Element	Unit	year_2000	year_2001	year_2002	year_2003	year_2004	...	year_2012	year_2013	year_2014
0	Afghanistan	1	All sectors with LULUCF	Emissions (CO2eq) (AR5)	kilotonnes	13346.3	14189.1	17043.7	17744.1	17494.7	...	32796.9	31062.4	31027.5
1	Afghanistan	1	All sectors without LULUCF	Emissions (CO2eq) (AR5)	kilotonnes	15734.4	14067.2	16921.8	17622.2	17372.8	...	33043.1	31308.6	31273.7
2	Afghanistan	1	Agri-food systems	Emissions (CO2eq) (AR5)	kilotonnes	12620.6	10835.6	13589.2	14132.3	14191.6	...	20893.8	20510.9	21018.3
3	Afghanistan	1	All sectors with LULUCF	Emissions (CO2eq) from CH4 (AR5)	kilotonnes	11734.9	10395.5	12541.1	12982.6	13042.6	...	18151.6	18168.9	18579.3
4	Afghanistan	1	All sectors without LULUCF	Emissions (CO2eq) from CH4 (AR5)	kilotonnes	11734.4	10395.5	12541.1	12982.6	13042.6	...	18151.6	18168.9	18579.3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
26142	Zimbabwe	246	Agricultural Soils	Indirect emissions (N2O)	kilotonnes	2.8	2.8	2.5	2.5	2.4	...	2.7	2.8	2.3
26143	Zimbabwe	246	Manure left on Pasture	Indirect emissions (N2O)	kilotonnes	2.1	2.2	2.0	1.9	1.9	...	2.3	2.3	1.9
26144	Zimbabwe	246	Food Transport	Emissions (CO2eq) from N2O (AR5)	kilotonnes	2.7	2.6	2.4	2.0	1.7	...	3.8	4.2	3.8
26145	Zimbabwe	246	Energy	Emissions (N2O)	kilotonnes	1.3	1.3	1.3	1.2	1.2	...	1.5	1.6	1.6
26146	Zimbabwe	246	Food Household Consumption	Emissions (CO2eq) from N2O (AR5)	kilotonnes	1.1	1.1	1.1	1.2	1.3	...	1.2	1.2	1.2

26147 rows × 27 columns

Figure 4.1 Unit field

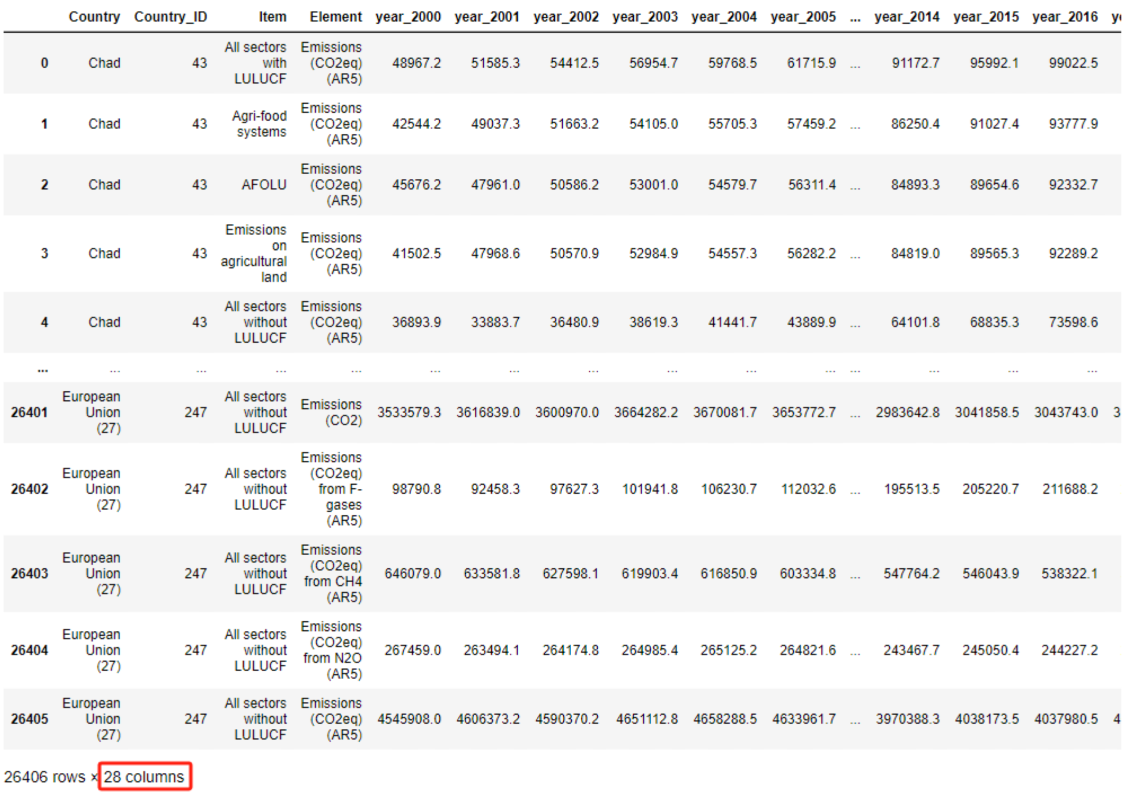
Actual action: Delete Unit field using pyspark's "drop" function.



```
#Delete Unit field
merged_df = merged_data.drop("Unit")
pandas_df4 = merged_df.toPandas()
display(pandas_df4)
```

Figure 4.2 Delete Unit field

Result: After deleting, there are no Unit fields in the dataset table.



	Country	Country_ID	Item	Element	year_2000	year_2001	year_2002	year_2003	year_2004	year_2005	...	year_2014	year_2015	year_2016	y
0	Chad	43	All sectors with LULUCF	Emissions (CO2eq) (AR5)	48967.2	51585.3	54412.5	56954.7	59768.5	61715.9	...	91172.7	95992.1	99022.5	
1	Chad	43	Agri-food systems	Emissions (CO2eq) (AR5)	42544.2	49037.3	51663.2	54105.0	55705.3	57459.2	...	86250.4	91027.4	93777.9	
2	Chad	43	AFOLU	Emissions (CO2eq) (AR5)	45676.2	47961.0	50586.2	53001.0	54579.7	56311.4	...	84893.3	89654.6	92332.7	
3	Chad	43	Emissions on agricultural land	Emissions (CO2eq) (AR5)	41502.5	47968.6	50570.9	52984.9	54557.3	56282.2	...	84819.0	89565.3	92289.2	
4	Chad	43	All sectors without LULUCF	Emissions (CO2eq) (AR5)	36893.9	33883.7	36480.9	38619.3	41441.7	43889.9	...	64101.8	68835.3	73598.6	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
26401	European Union (27)	247	All sectors without LULUCF	Emissions (CO2)	3533579.3	3616839.0	3600970.0	3664282.2	3670081.7	3653772.7	...	2983642.8	3041858.5	3043743.0	3
26402	European Union (27)	247	All sectors without LULUCF	Emissions (CO2eq) from F-gases (AR5)	98790.8	92458.3	97627.3	101941.8	106230.7	112032.6	...	195513.5	205220.7	211688.2	
26403	European Union (27)	247	All sectors without LULUCF	Emissions (CO2eq) from CH4 (AR5)	646079.0	633581.8	627598.1	619903.4	616850.9	603334.8	...	547764.2	546043.9	538322.1	
26404	European Union (27)	247	All sectors without LULUCF	Emissions (CO2eq) from N2O (AR5)	267459.0	263494.1	264174.8	264985.4	265125.2	264821.6	...	243467.7	245050.4	244227.2	
26405	European Union (27)	247	All sectors without LULUCF	Emissions (CO2eq) (AR5)	4545908.0	4606373.2	4590370.2	4651112.8	4658288.5	4633961.7	...	3970388.3	4038173.5	4037980.5	4

26406 rows × 28 columns

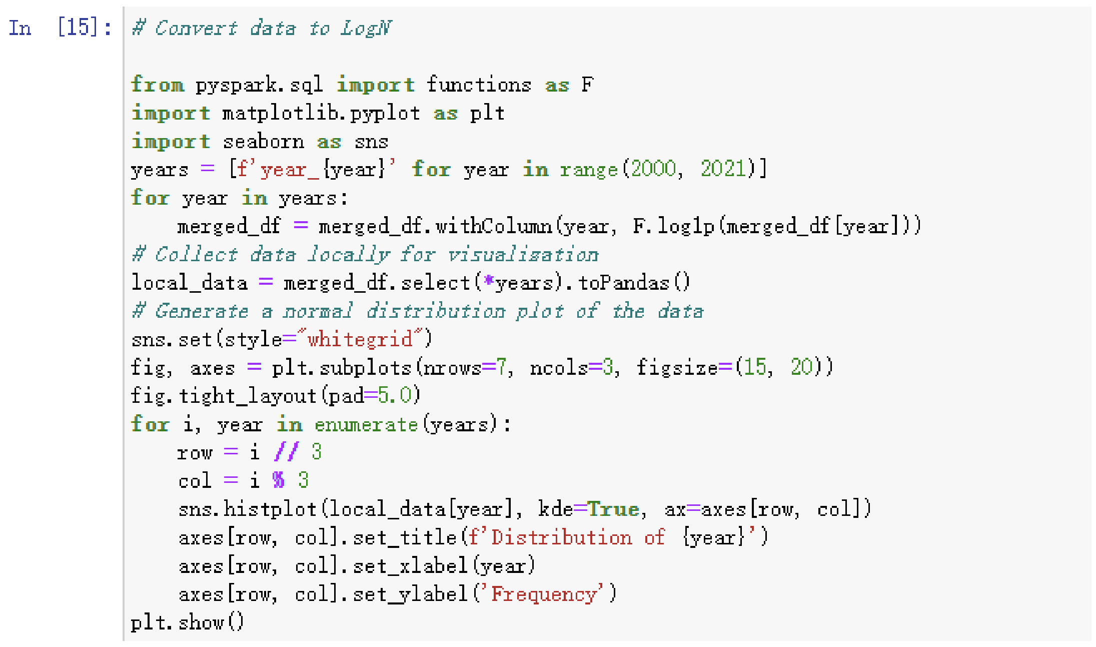
Figure 4.3 The data table of no Unit field

4.2 Data projection

4.2.1 Data transformation

Having data that adheres to a normal distribution is an underlying principle that plays a crucial role in various analytical scenarios. The prominence of the normal distribution in statistical analyses comes from the foundational nature of this distribution in the realm of probability and statistics. Many of the widely used statistical tests, methodologies, and algorithms are constructed on the assumption that the underlying data adhere to a normal distribution. This assumption is often a consequence of mathematical conveniences offered by the bell-shaped curve, as well as empirical observations that many natural phenomena seem to align with it.

The primary reason is that when data is normally distributed, the tools and methods designed with this assumption provide the most powerful and, by extension, the most reliable results. Not meeting this assumption can lead to biases, inaccuracies, or even entirely misleading outcomes.


```
In  [15]: # Convert data to LogN

          from pyspark.sql import functions as F
          import matplotlib.pyplot as plt
          import seaborn as sns
          years = [f'year_{year}' for year in range(2000, 2021)]
          for year in years:
              merged_df = merged_df.withColumn(year, F.log1p(merged_df[year]))
          # Collect data locally for visualization
          local_data = merged_df.select(*years).toPandas()
          # Generate a normal distribution plot of the data
          sns.set(style="whitegrid")
          fig, axes = plt.subplots(nrows=7, ncols=3, figsize=(15, 20))
          fig.tight_layout(pad=5.0)
          for i, year in enumerate(years):
              row = i // 3
              col = i % 3
              sns.histplot(local_data[year], kde=True, ax=axes[row, col])
              axes[row, col].set_title(f'Distribution of {year}')
              axes[row, col].set_xlabel(year)
              axes[row, col].set_ylabel('Frequency')
          plt.show()
```

Figure 4.4 Transform data

When dealing with real-world data, it's not always a given that it will naturally follow a normal distribution. Various factors can cause data to deviate from this ideal. That's where the significance of data observation comes into play. By scrutinizing the data distribution, we can identify potential outliers, skewness, or any other anomalies that might disrupt the normal distribution assumption.

To combat these irregularities, adopting a suitable data transformation strategy becomes paramount. Common transformations, like logarithmic or square root conversions, can often help in reshaping the data. The goal here is to mold the data in such a way that it aligns more closely with the normal distribution.

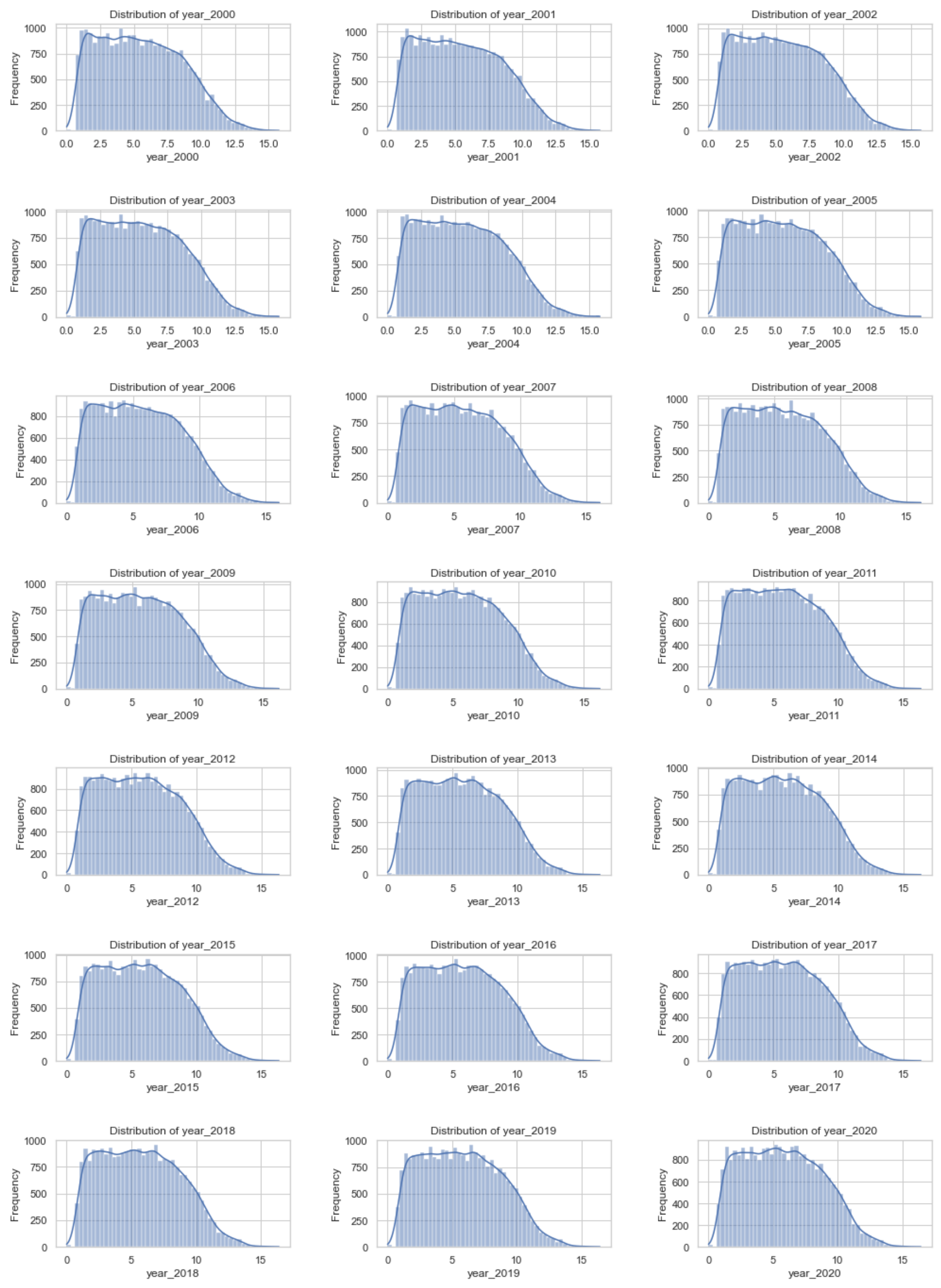
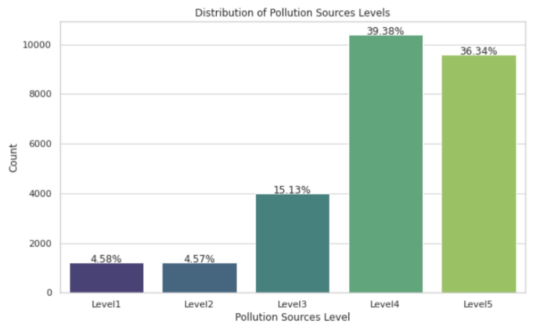


Figure 4.5 LogN transformation

Through the analysis of the data set itself, the emission data from 2000 to 2020 does not obey the law of normal distribution. Therefore, data conversion needs to be performed on the 21 fields from year_2000 to year_2020 in the data set. According to the normal distribution diagram of the data (Figure 4.5), it can be seen that after the LogN conversion of the data in the year_2000 to year2020 fields, the distribution is close to the normal distribution, so this step of data transformation is necessary.

4.2.2 Group data

In our current dataset, we have a field representing the emission source ranking for each country, categorized as levels ranging from 'level1' to 'level5'. To streamline our analysis and provide a more generalized view, we aim to reduce these five levels into three distinct categories: 'High Level', 'Normal Level', and 'Low Level'.



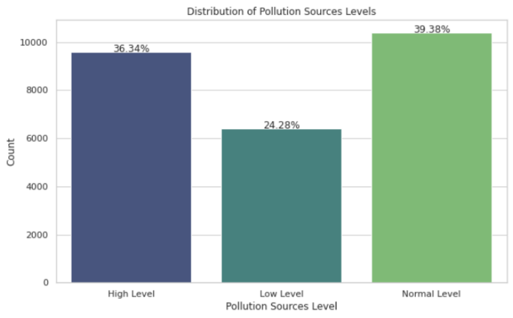


Figure4.6 The distribution of Level Figure 4.7 Level after new group

Grouping Functionality:

Convert 'level1', 'level2' and 'level3' into 'Low Level'.

Let 'level4' remain as 'Normal Level'.

Convert 'level5' into 'High Level'.

Reducing 5 levels to 3 levels can simplify data interpretation, improve model performance, enhance data visualization, reduce the risk of overfitting, make communication more direct, and facilitate data balance, thereby bringing more efficient results to data analysis and model training.

4.2.3 Balance data

After the data is reclassified, you must check the distribution again. If a category is overwhelmingly representative compared to other categories, data balancing is required. In Python, this can be achieved through techniques such as oversampling underrepresented categories or under sampling overrepresented categories.

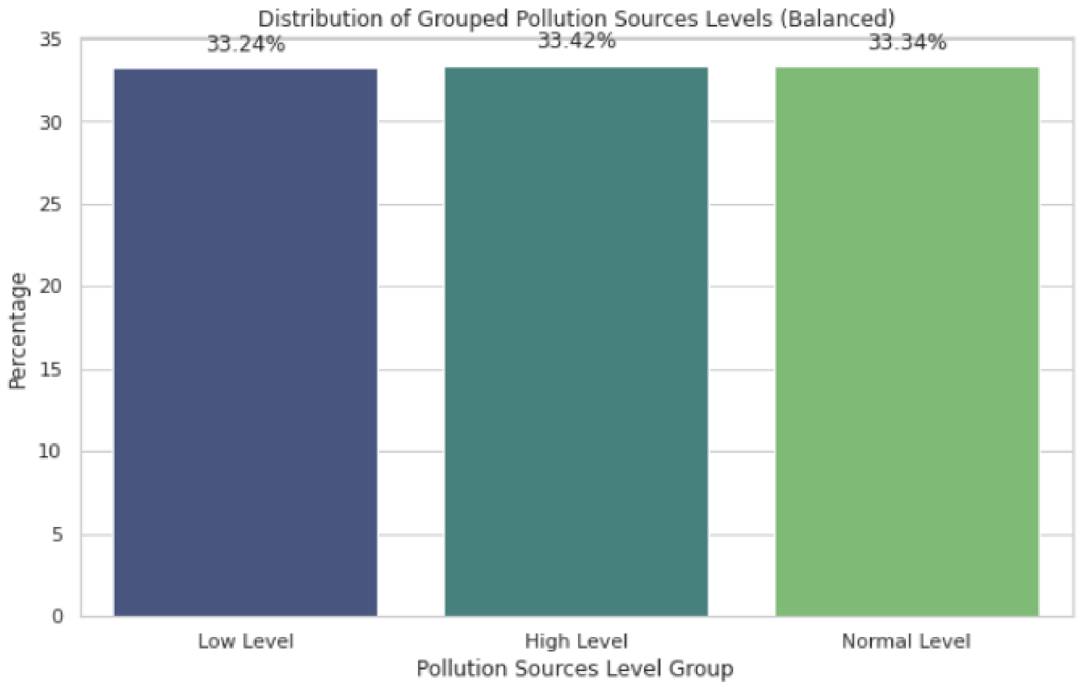


Figure 4.8 Level after balancing

The purpose behind this reclassification and subsequent data balancing is to ensure that not only is our dataset simpler, but that our predictive models have an equal chance of being trained at each source level. This results in more robust and unbiased models, which in turn produce more reliable and interpretable results.

5. Data-Mining Method(s) Selection

5.1 Discussion DM methods within the context of the DM objectives

5.1.1 Data mining goals/objectives

Objective 1: Leveraging the data from 2020, our goal is to graphically represent a set of countries that are significant contributors to the global greenhouse gas emissions. This visual representation will provide a quick snapshot of major global players, allowing stakeholders to identify key areas for intervention or further analysis.

Objective 2: Beyond just identifying the emission volumes, it's pivotal to understand the relationship between the diversity of emission sources and the actual quantity of greenhouse gas emissions. By examining the correlation between the number and levels of emissions, stakeholders can gain insights into whether having multiple sources intensifies the emission levels or if the severity is tied to specific sources.

Objective 3: By zooming into countries that are major contributors to greenhouse gas emissions, we aim to detail which specific sources within these nations are the primary culprits. This granular approach will reveal the nature of the emissions within the context of each country, laying the groundwork for targeted interventions.

Objective 4: Using the insights from Objective 3, a strategic approach can be developed urging countries to adopt tailored environmental protection plans. Recognizing that each country's emission profile is unique, differentiated action plans can address the specific challenges presented by individual emission sources, leading to more effective mitigation strategies.

5.1.2 Data mining Method overview

A Data Mining (DM) is essentially the process of discovering patterns and information from large amounts of data(Peterson & Baker, n.d.). The data sources can include databases, data warehouses, the internet, and other sources. Let's discuss Data Mining methods within the context of its primary objectives:

Classification(Sudhir et al., 2017)**:**
Objective: To predict the predefined target attribute.
Libraries: scikit-learn, TensorFlow, Keras, PyTorch
Methods: Decision Trees, Neural Networks, Naive Bayes, Support Vector Machines, etc.
Context: Classification is useful in applications where predictive models are necessary. For instance, banks might use it to predict whether a loan will default or not.

Clustering(Sinharay, 2016)**:**
Objective: To identify the intrinsic groupings in data, where items in a group are more similar to each other than those in other groups.
Library: scikit-learn, SciPy
Methods: K-means, Hierarchical clustering, DBSCAN, etc.
Context: Market segmentation in business is a typical use-case. Companies cluster customers into segments that exhibit similar behavior.

Association(Xu et al., 2020)**:**
Objective: To discover interesting relationships between variables in large datasets.
Library: mlxtend, apyori
Methods: Apriori, Eclat, FP-Growth, etc.
Context: Retailers might use association to identify products that are often bought together, facilitating better product placements or marketing strategies.

Regression(Sinharay, 2016)**:**
Objective: To predict a continuous value.

Library: scikit-learn, statsmodels

Methods: Linear Regression, Polynomial Regression, etc.

Context: It's extensively used in forecasting. A common use-case is predicting house prices based on various features like location, number of rooms, etc.

Anomaly Detection(Agrawal & Agrawal, 2015)**:**

Objective: To identify unusual patterns that don't conform to expected behavior.

Library: scikit-learn, PyOD

Methods: Statistical methods, Nearest Neighbors, Isolation Forest, etc.

Context: Credit card companies use this to identify potentially fraudulent transactions.

Sequence Mining(Parthasarathy et al., n.d.)**:**

Objective: To discover or identify patterns in sequences.

Library: PrefixSpan, SPMF

Methods: Generalized Sequential Patterns (GSP), Sequential Pattern Discovery using Equivalence classes (SPADE), etc.

Context: Useful in DNA sequence analysis or understanding shopping sequences in market-basket analysis.

Text Mining(*Text Mining: The State of the Art and the Challenges*, 2000)**:**

Objective: To derive valuable information from unstructured text data.

Library: NLTK, spaCy, gensim, scikit-learn

Context: Companies might analyze customer feedback or reviews to identify common pain points or areas of improvement.

Forecasting(Rajput & Bobde, 2016)**:**

Objective: To predict future values based on past data.

Library: statsmodels, Prophet, pmdarima

Methods: Time Series, Exponential Smoothing, ARIMA, etc.

Context: Stock market predictions, sales forecasting, weather forecasting are areas where these methods are applied.

In summary, the choice of a DM method depends largely on the objective of the task. Each method has its strengths and is best suited for particular types of problems. Properly aligning the objective with the method ensures effective and meaningful results.

5.2 Selecting the appropriate Data-Mining method(s)

Objective 1: Charting greenhouse gas emissions of various countries based on 2020 data.

Method: Descriptive Analysis and Visualization

Basic Principle: Descriptive analysis provides a comprehensive summary about the main aspects of the data, often presented in the form of charts, graphs, and tables. Visualization tools, such as bar graphs or pie charts, can graphically showcase the countries with the highest greenhouse gas emissions, making the information more digestible and actionable for stakeholders.

Advantages:

Immediate Insight: Visual methods provide instant, easily interpretable results that make the identification of high-emitting countries straightforward.

Engagement: Graphical representations can engage a wide range of audiences, including non-technical stakeholders, facilitating broader discussions around the data.

Data Quality Checks: Visualization can also help detect anomalies or outliers in the data that might need further investigation.

Objective 2: Relationship between the number of sources and greenhouse gas emissions.

Method: Correlation Analysis and Regression

Basic Principle: Correlation analysis determines the degree to which two variables move in relation to each other. Regression analysis, especially linear regression, would model the relationship between the number of sources (independent variable) and the greenhouse gas emissions (dependent variable). By assessing the strength and direction of this relationship, one can infer how the variety of emission sources might impact the overall emission levels.

Advantages:

Quantitative Relationship: Linear regression not only identifies a relationship but also quantifies it, enabling predictions and understanding of impact magnitude.

Interpretability: The coefficients of the regression provide clear insights into how much each source contributes to emissions.

Predictive Capabilities: Once the relationship is established, it can be used for future predictions, aiding in forecasting emission trends based on source count changes.

Objective 3: Proposing environmental plans based on emission sources.

Method: Decision Trees

Basic Principle: Decision Trees can be used to create a model that predicts a decision, or an outcome based on certain conditions. For environmental protection plans, the decision tree could split countries based on different criteria like types of emission sources, emission volume, etc., to suggest specific actions. Association Rule Mining, on the other hand, discovers interesting relationships between variables in large databases. It can help identify which sources of emissions are often found together, aiding in the formulation of targeted environmental plans.

Advantages:

Actionable Insights: Decision trees present results in a flowchart-like structure, making it easier to derive actionable paths for each decision point (in this case, emission source type or level).

Simple Interpretation: The tree structure is intuitive, and the decision-making process can be easily followed from root to leaf.

Multifactor Analysis: Decision trees can handle multiple predictors, allowing a comprehensive view of how various emission sources might influence the environmental plan.

6. Data-Mining Algorithm(s) Selection

6.1 Exploratory analysis of DM algorithms of the DM objectives

Exploratory Data Analysis is an indispensable step prior to diving into data mining algorithms. It offers profound insights into data structure, patterns, and anomalies, which guides the choice of appropriate mining techniques. (*9780429138423_webpdf*, n.d.)

By detecting outliers, understanding data distributions, revealing potential correlations, and validating algorithmic assumptions, Exploratory Data Analysis ensures the efficiency and accuracy of the subsequent modeling process. Skipping Exploratory Data Analysis could lead to suboptimal models or overlook pivotal data features and patterns.

Regression Methods(Sinharay, 2016)**:**

a. Linear Regression

Purpose: Estimates the relationship between the dependent variable and one or more independent variables.

Advantages:

Results are intuitive and interpretable.

Can be used to predict numerical data.

Can assess the importance of predictors.

b. Polynomial Regression

Purpose: Polynomial Regression models the relationship between the independent and dependent variables as an nth degree polynomial, capturing nonlinearities in the data.

Advantages:

Polynomial Regression offers flexibility in modeling curvilinear relationships.

Providing a better fit for nonlinear trends and capturing interactions between features.

While remaining relatively straightforward to implement and interpret.

Classification Methods(Sudhir et al., 2017)**:**

a. Support Vector Machines (SVM)

Purpose: Finds the optimal hyperplane in classification and regression analysis.

Advantages:

Can effectively handle data in high-dimensional spaces.

Suitable for non-linear problems.

Performs well on certain complex datasets.

b. Logistic Regression

Purpose: Estimates the probability of an event and performs binary classification.

Advantages:

Provides the strength and direction of the effect of variables.

Results are interpretable.

Handles linearly separable problems.

Clustering Methods(Sun et al., 2008)**:**

a. K-Means

Purpose: Divides data into K clusters, ensuring that data points within a cluster are as similar as possible, while data points between different clusters are as distinct as possible.

Advantages:

Highly computationally efficient, especially suited for large datasets.

Results are easy to interpret.

Typically performs well across a variety of applications.

b. Hierarchical Clustering

Purpose: Constructs a tree-like cluster hierarchy by recursively merging or dividing clusters.

Advantages:

No need to specify the number of clusters in advance.

Results can be visualized using a dendrogram, providing an intuitive view of data hierarchy.

Suitable for smaller datasets.

c. DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

Purpose: Identifies and segments clusters based on data density and is capable of discovering clusters of arbitrary shapes.

Advantages:

Can identify and ignore noise points.

Capable of discovering clusters with non-uniform shapes.

Doesn't require the number of clusters to be specified in advance.

Decision Tree Methods(Kumar & Sharma, 2016)**:**

a. C5.0 Tree

Purpose: Generates a decision tree for classification.

Advantages:

Can handle data with both numeric and nominal attributes.

Produces relatively concise trees.

Conducts attribute selection and pruning, which helps in avoiding overfitting.

b. C&R Tree (Classification and Regression Trees)

Purpose: Can be used for both classification and regression tasks.

Advantages:

Can handle missing data.

Tree size can be adjusted through pruning.

Applicable to large datasets.

c. Random Forest

Purpose: Combines the predictions of multiple decision trees to increase accuracy and prevent overfitting.

Advantages:

Improves accuracy through voting or averaging.

Can effectively handle a large number of input variables.

Can estimate the importance of each input variable.

Association Rule Mining(Kumbhare Santosh V Chobe, n.d.)**:**

a. Apriori

Purpose: Finds associations between items in large datasets.

Advantages:

Suitable for large datasets.

Can discover frequent item sets, such as for product recommendations.

Results are interpretable.

b. FP-Growth

Purpose: More efficiently discovers frequent item sets compared to Apriori.

Advantages:

Faster than Apriori.

Uses the compact FP-tree data structure.

Can handle large datasets effectively.

6.2 Select algorithm(s) in a logical manner

6.2.1 The first data mining target

The first data mining goal is to visualize countries' greenhouse gas emissions based on 2020 data.

For this goal, what is needed is the ability of data visualization, and the complexity of the algorithm is not the focus, but the clarity and interpretability of the visualization. The use of bar and pie charts to highlight the countries with the highest number of sources was chosen here.

First of all, according to the bar chart, the emissions of 5 countries (regions) are very prominent, which shows that these 5 countries will bear a great responsibility to reduce greenhouse gas emissions and protect the environment.

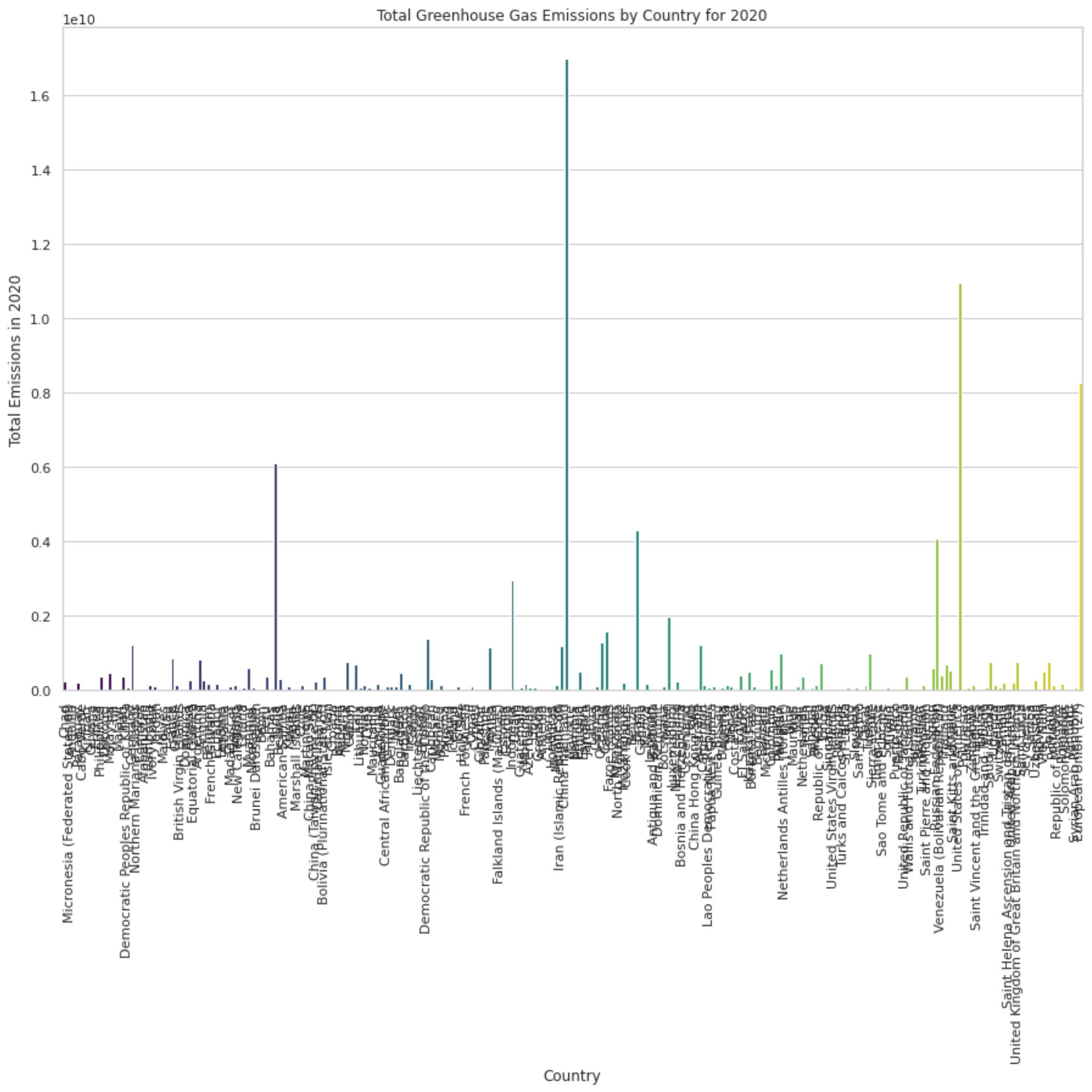


Figure 6.1 Bar of emissions

Next, the top5 greenhouse gas emitters are sorted according to the remaining countries, and a pie chart shows the share of each country's emissions in the world's total emissions. As can be seen through Figure 6.2, the top5 countries are China mainland (top1), United States of America (top2), India (top3), European Union (top4), Brazil (top5). The combined emissions of these five countries already exceed 50% of the total emissions of more than 200 countries around the world.

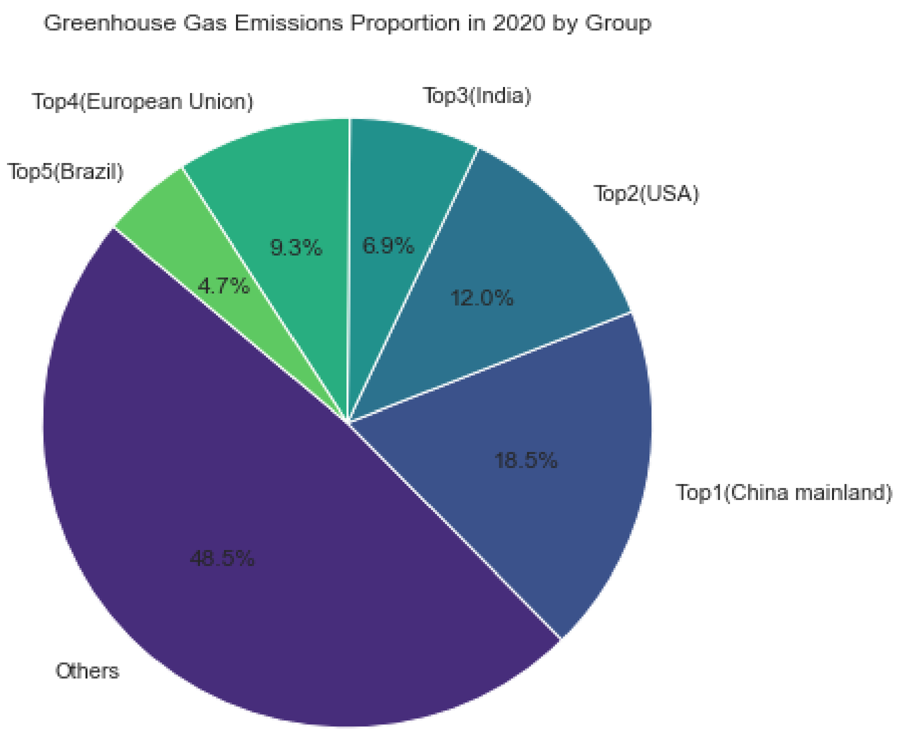


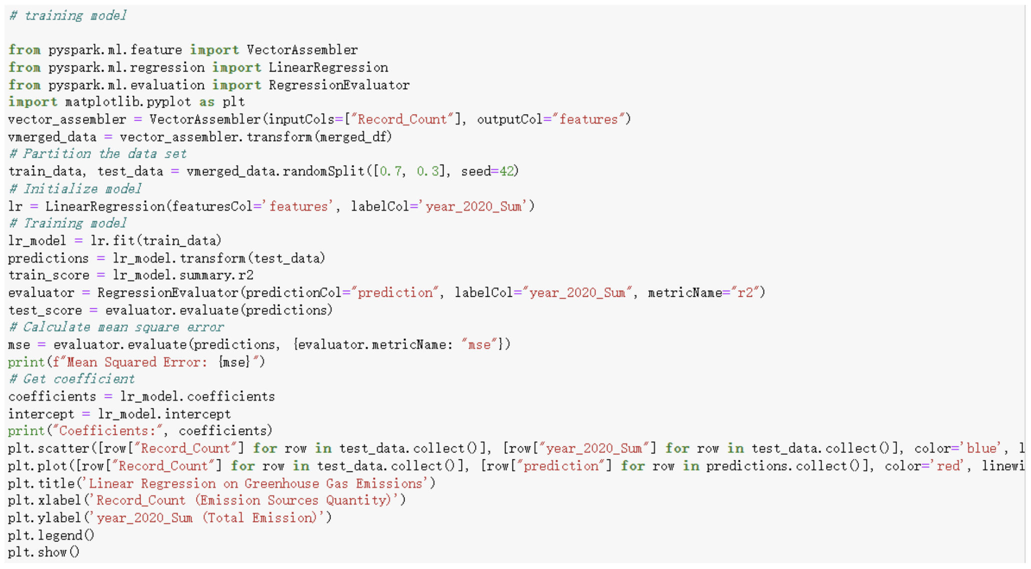
Figure 6.2 Pie of emissions

6.2.2 The second data mining goal

The second data mining goal is the relationship between the number of sources and greenhouse gas emissions.

Some of the methods that may be used for this goal are linear regression, correlation analysis, and nonlinear regression. The algorithms that can be used are linear regression models and polynomial regression models. Start with an exploratory analysis to see if there is a linear relationship between the two. If it is a very standard linear relationship, then the goal can be achieved using a linear regression model. If the linear relationship is not obvious, but you still want to explore whether there is a certain correlation, you can choose polynomial regression.

Start by exploring the use of linear modeling. First, the data set is classified, 70% of which is the training data set and 30% is the testing data set. Then select the number of emission sources (Record_Count field) as the predictor variable in the linear modeling, and the total greenhouse gas emissions of each country (year_2020_Sum field) as the target and start training the model.



```
# training model

from pyspark.ml.feature import VectorAssembler
from pyspark.ml.regression import LinearRegression
from pyspark.ml.evaluation import RegressionEvaluator
import matplotlib.pyplot as plt
vector_assembler = VectorAssembler(inputCols=["Record_Count"], outputCol="features")
vmerged_data = vector_assembler.transform(merged_df)
# Partition the data set
train_data, test_data = vmerged_data.randomSplit([0.7, 0.3], seed=42)
# Initialize model
lr = LinearRegression(featuresCol='features', labelCol='year_2020_Sum')
# Training model
lr_model = lr.fit(train_data)
predictions = lr_model.transform(test_data)
train_score = lr_model.summary.r2
evaluator = RegressionEvaluator(predictionCol="prediction", labelCol="year_2020_Sum", metricName="r2")
test_score = evaluator.evaluate(predictions)
# Calculate mean square error
mse = evaluator.evaluate(predictions, {evaluator.metricName: "mse"})
print(f"Mean Squared Error: {mse}")
# Get coefficient
coefficients = lr_model.coefficients
intercept = lr_model.intercept
print("Coefficients:", coefficients)
plt.scatter([row["Record_Count"] for row in test_data.collect()], [row["year_2020_Sum"] for row in test_data.collect()], color='blue', l
plt.plot([row["Record_Count"] for row in test_data.collect()], [row["prediction"] for row in predictions.collect()], color='red', linewi
plt.title('Linear Regression on Greenhouse Gas Emissions')
plt.xlabel('Record_Count (Emission Sources Quantity)')
plt.ylabel('year_2020_Sum (Total Emission)')
plt.legend()
plt.show()
```

Figure 6.3 Partition and Training model

An analytical evaluation of the results from linear modeling found a fairly standard linear relationship, with a mean squared error of 0.84 in the model summary. Therefore, it can be determined that there is a strong relationship between the number of sources and greenhouse gas emissions. When a country has more sources, its greenhouse gas emissions will be greater.

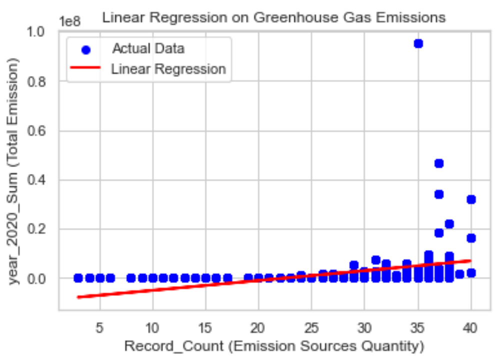
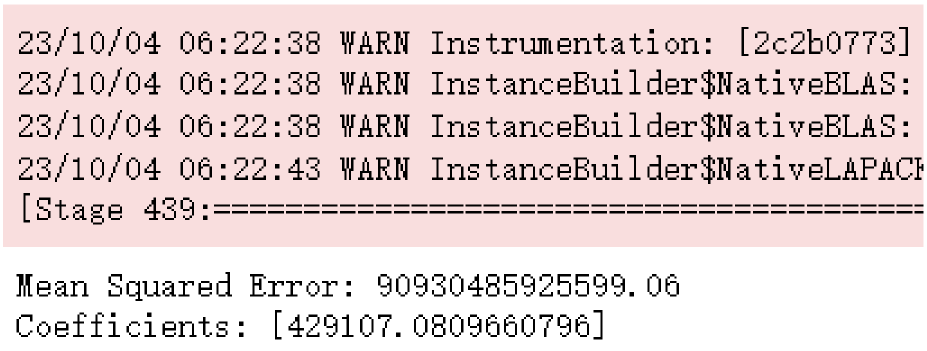


Figure 6.4 standard linear relationship

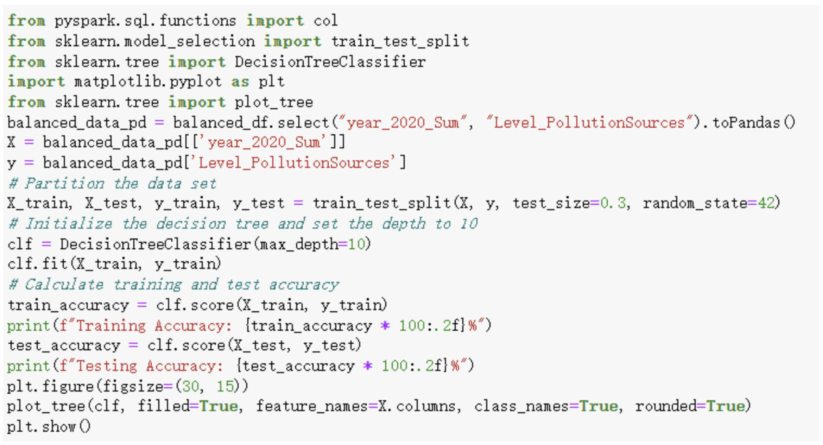


```
23/10/04 06:22:38 WARN Instrumentation: [2c2b0773]
23/10/04 06:22:38 WARN InstanceBuilder$NativeBLAS:
23/10/04 06:22:38 WARN InstanceBuilder$NativeBLAS:
23/10/04 06:22:43 WARN InstanceBuilder$NativeLAPACK
[Stage 439:==========================================

Mean Squared Error: 90930485925599.06
Coefficients: [429107.0809660796]
```

Figure 6.5 Mean Squared Error & Coefficients

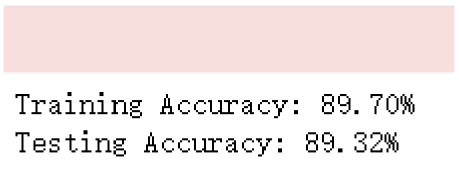
Next, we can use the decision tree to predict the greenhouse gas emission ratings of each country based on their greenhouse gas emissions and see how accurate the results are.



```python
from pyspark.sql.functions import col
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree
balanced_data_pd = balanced_df.select("year_2020_Sum", "Level_PollutionSources").toPandas()
X = balanced_data_pd[['year_2020_Sum']]
y = balanced_data_pd['Level_PollutionSources']
# Partition the data set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
# Initialize the decision tree and set the depth to 10
clf = DecisionTreeClassifier(max_depth=10)
clf.fit(X_train, y_train)
# Calculate training and test accuracy
train_accuracy = clf.score(X_train, y_train)
print(f"Training Accuracy: {train_accuracy * 100:.2f}%")
test_accuracy = clf.score(X_test, y_test)
print(f"Testing Accuracy: {test_accuracy * 100:.2f}%")
plt.figure(figsize=(30, 15))
plot_tree(clf, filled=True, feature_names=X.columns, class_names=True, rounded=True)
plt.show()
```

Figure 6.6 training DecisionTreeClassifier model

In python's DecisionTreeClassifier, the data set is first divided into 70% training data set and 30% test data set. Then select the number of emission sources and use the total greenhouse gas emissions as the prediction target, limiting the depth of the decision tree to 10. Through the result analysis, the accuracy rate on the training set was 86.27%, and the accuracy rate on the test set was 86.65%. The data show that the model can predict country-sourced quantity categories with substantial reliability.

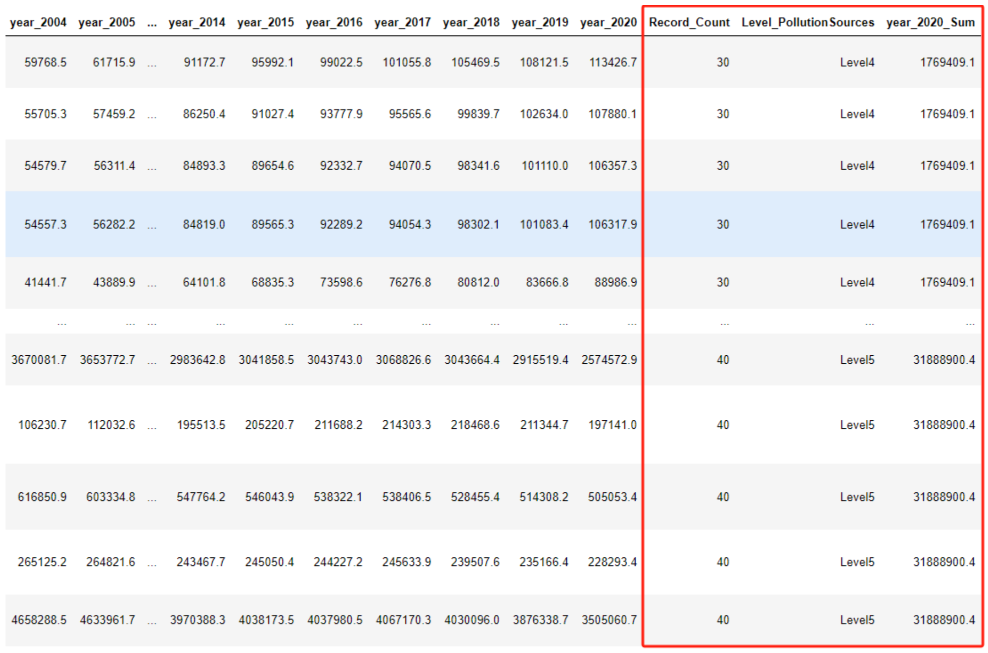


```
Training Accuracy: 89.70%
Testing Accuracy: 89.32%
```

Figure 6.7 Accuracy of DecisionTreeClassifier model

6.3 Built the appropriate algorithm/model

First, I generated the "merged_df" dataset after cleaning and merging the data in Step3. Use and read this data set after step 4 to perform data transformation and balancing.

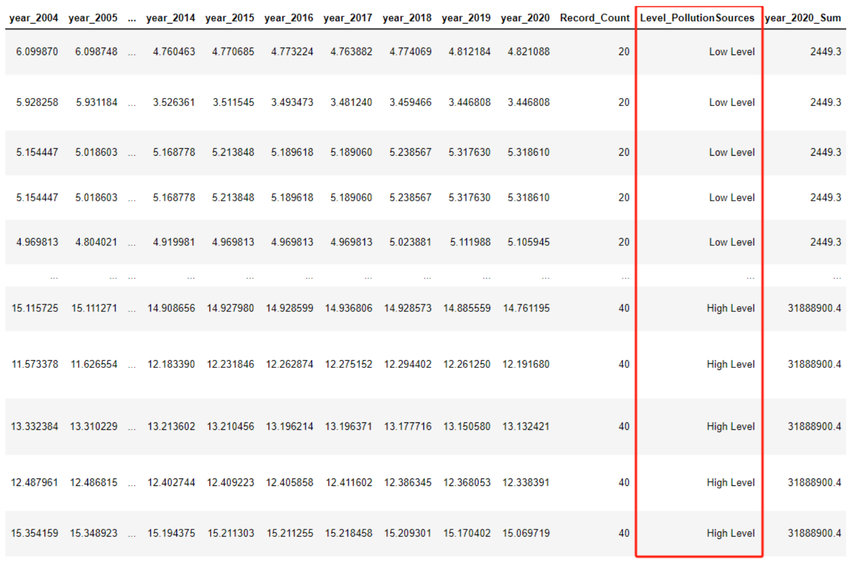


year_2004	year_2005	...	year_2014	year_2015	year_2016	year_2017	year_2018	year_2019	year_2020	Record_Count	Level_PollutionSources	year_2020_Sum
59768.5	61715.9	...	91172.7	95992.1	99022.5	101055.8	105469.5	108121.5	113426.7	30	Level4	1769409.1
55705.3	57459.2	...	86250.4	91027.4	93777.9	95565.6	99839.7	102634.0	107880.1	30	Level4	1769409.1
54579.7	56311.4	...	84893.3	89654.6	92332.7	94070.5	98341.6	101110.0	106357.3	30	Level4	1769409.1
54557.3	56282.2	...	84819.0	89565.3	92289.2	94054.3	98302.1	101083.4	106317.9	30	Level4	1769409.1
41441.7	43889.9	...	64101.8	68835.3	73598.6	76276.8	80812.0	83666.8	88986.9	30	Level4	1769409.1
...	...	...	...	...	...	...	...	...	...	...	...	...
3670081.7	3653772.7	...	2983642.8	3041858.5	3043743.0	3068826.6	3043664.4	2915519.4	2574572.9	40	Level5	31888900.4
106230.7	112032.6	...	195513.5	205220.7	211688.2	214303.3	218468.6	211344.7	197141.0	40	Level5	31888900.4
616850.9	603334.8	...	547764.2	546043.9	538322.1	538406.5	528455.4	514308.2	505053.4	40	Level5	31888900.4
265125.2	264821.6	...	243467.7	245050.4	244227.2	245633.9	239507.6	235166.4	228293.4	40	Level5	31888900.4
4658288.5	4633961.7	...	3970388.3	4038173.5	4037980.5	4067170.3	4030096.0	3876338.7	3505060.7	40	Level5	31888900.4

Figure 6.8 the "merged_df" dataset

Next, review the dataset and remove unimportant fields to reduce the dimensionality of the dataset and improve the efficiency of the overall model.

Third, continuous variables in the data set were reviewed and corresponding data transformations were performed to generate continuous variables that fit a normal distribution. Discrete variables in the dataset were then reviewed, and data balancing was performed through oversampling methods. Generating the "balanced_df" dataset after completing the above steps.



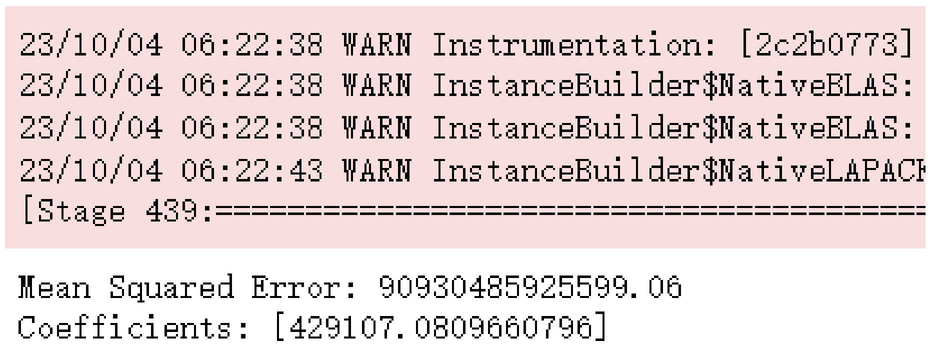
year_2004	year_2005	...	year_2014	year_2015	year_2016	year_2017	year_2018	year_2019	year_2020	Record_Count	Level_PollutionSources	year_2020_Sum
6.099870	6.098748	...	4.760463	4.770685	4.773224	4.763882	4.774069	4.812184	4.821088	20	Low Level	2449.3
5.928258	5.931184	...	3.526361	3.511545	3.493473	3.481240	3.459466	3.446808	3.446808	20	Low Level	2449.3
5.154447	5.018603	...	5.168778	5.213848	5.189618	5.189060	5.238567	5.317630	5.318610	20	Low Level	2449.3
5.154447	5.018603	...	5.168778	5.213848	5.189618	5.189060	5.238567	5.317630	5.318610	20	Low Level	2449.3
4.969813	4.804021	...	4.919981	4.969813	4.969813	4.969813	5.023881	5.111988	5.105945	20	Low Level	2449.3
...	...	...	...	...	...	...	...	...	...	...	...	...
15.115725	15.111271	...	14.908656	14.927980	14.928599	14.936806	14.928573	14.885559	14.761195	40	High Level	31888900.4
11.573378	11.626554	...	12.183390	12.231846	12.262874	12.275152	12.294402	12.261250	12.191680	40	High Level	31888900.4
13.332384	13.310229	...	13.213602	13.210456	13.196214	13.196371	13.177716	13.150580	13.132421	40	High Level	31888900.4
12.487961	12.486815	...	12.402744	12.409223	12.405858	12.411602	12.386345	12.368053	12.338391	40	High Level	31888900.4
15.354159	15.348923	...	15.194375	15.211303	15.211255	15.218458	15.209301	15.170402	15.069719	40	High Level	31888900.4

Figure 6.9 the "balanced_data" dataset

Fourth, view intuitive visual data through bar charts in matplotlib. The data set is classified according to the ranking of total emissions through the defined python function, and then the visualization data of the proportion is generated through the pie chart in matplotlib.

Fifth, use the linear regression model to see whether the relevant data conforms to the linear model and whether the analysis is reasonable. Mainly look at the characteristic coefficients and intercepts in the linear regression model. In this linear regression model, the two model parameters "mean squared error" and "Coefficients" are mainly used. Coefficients represents the slope of the linear regression. The

429107.08 in the model result represents the impact of a country increasing one greenhouse gas emission source. Increase greenhouse gas emissions by so many units.



```
23/10/04 06:22:38 WARN Instrumentation: [2c2b0773]
23/10/04 06:22:38 WARN InstanceBuilder$NativeBLAS:
23/10/04 06:22:38 WARN InstanceBuilder$NativeBLAS:
23/10/04 06:22:43 WARN InstanceBuilder$NativeLAPACK
[Stage 439:========================================

Mean Squared Error: 90930485925599.06
Coefficients: [429107.0809660796]
```

Figure 6.10 coefficients and intercepts

Sixth, check whether the relevant data can be correctly predicted by the DecisionTreeClassifier model. Ensure that the continuous variables obey a normal distribution before analysis and use the print statement to output ‘train_accuracy’ and ‘test_accuracy’ to view the accuracy after running the model. Among them, train_accuracy represents the accuracy of the training set, and test_accuracy represents the accuracy of the test set.

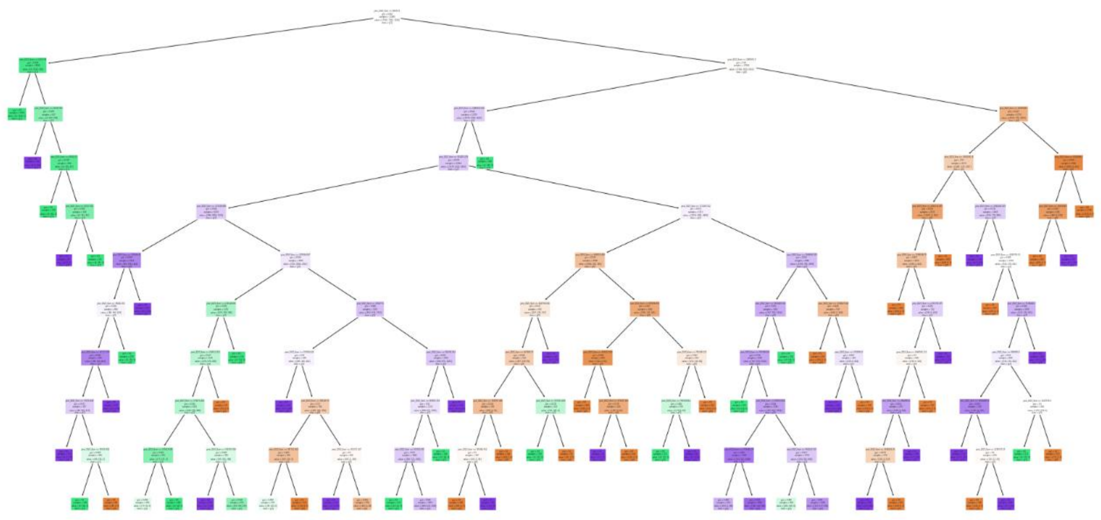


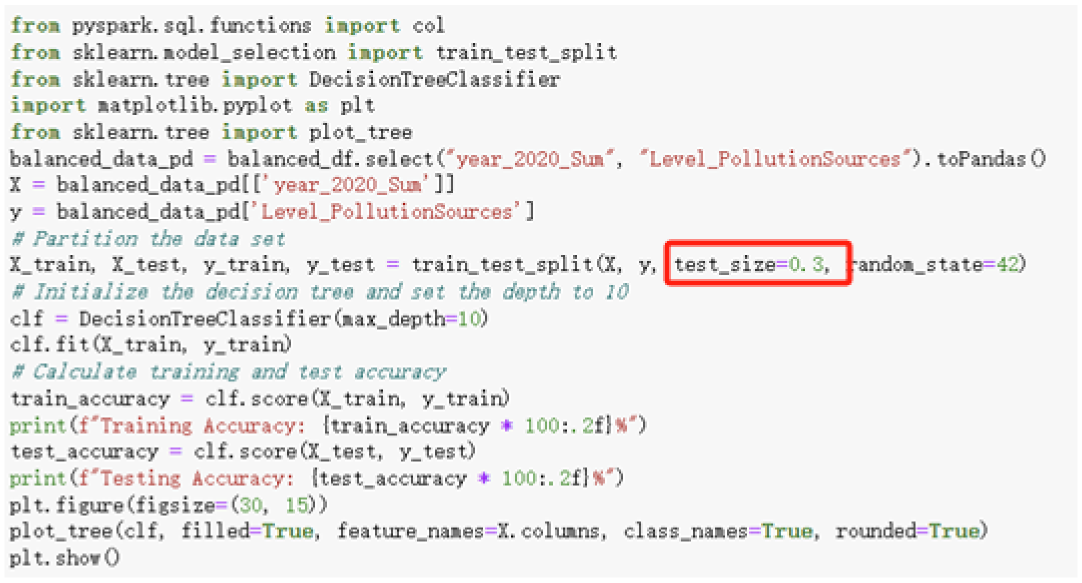
Figure 6.11 Result of model

Finally, with the dataset ready, extract valuable relationships and patterns based on the data analysis content of all operations. The model parameters that are focused on checking are the decisions of each node in the decision tree. A key turning point is the point where the total greenhouse gas emissions equal 104340.434.

7. Data Mining

7.1 Creating Logical test

Before we can do data mining, we first need to create a logical test design. The reasons for choosing a particular test design are based on a variety of factors. For example, why use a 70/30 training/test split? This segmentation is usually because it provides the model with enough data to learn, while also leaving enough data to verify the performance of the model. Training with 70% of the data ensures that the model is adequately trained on diverse data, while testing with 30% of the data provides us with an unbiased assessment of how the model performs on unseen data.



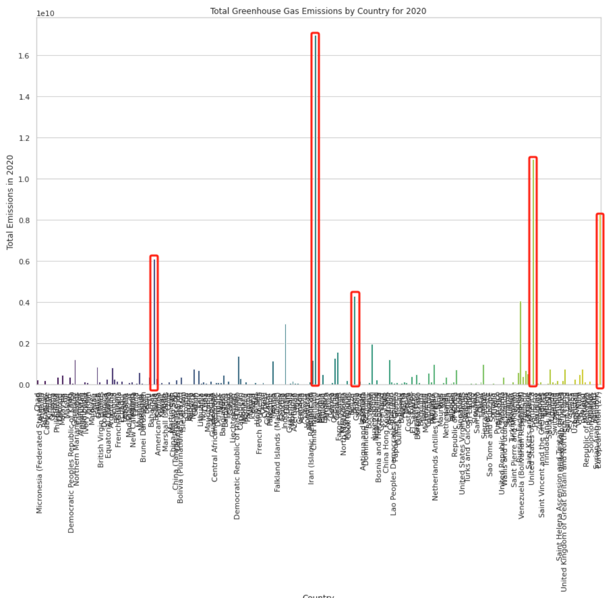
```
from pyspark.sql.functions import col
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree
balanced_data_pd = balanced_df.select("year_2020_Sum", "Level_PollutionSources").toPandas()
X = balanced_data_pd[['year_2020_Sum']]
y = balanced_data_pd['Level_PollutionSources']
# Partition the data set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
# Initialize the decision tree and set the depth to 10
clf = DecisionTreeClassifier(max_depth=10)
clf.fit(X_train, y_train)
# Calculate training and test accuracy
train_accuracy = clf.score(X_train, y_train)
print(f"Training Accuracy: {train_accuracy * 100:.2f}%")
test_accuracy = clf.score(X_test, y_test)
print(f"Testing Accuracy: {test_accuracy * 100:.2f}%")
plt.figure(figsize=(30, 15))
plot_tree(clf, filled=True, feature_names=X.columns, class_names=True, rounded=True)
plt.show()
```

Figure 7.1 Data partition

7.2 Conducting Data Mining

7.2.1 Output the results of the goal 1 model

The model is designed to find the countries that are primarily responsible for global warming, the large amount of greenhouse gas emissions. Based on the results of the model, it is clear from the visual data pie chart that five countries have a very large total of greenhouse gas emissions, accounting for 50% of the world's emissions. Therefore, these five countries will assume the responsibility of reducing greenhouse gas emissions and environmental governance as the main bearers.



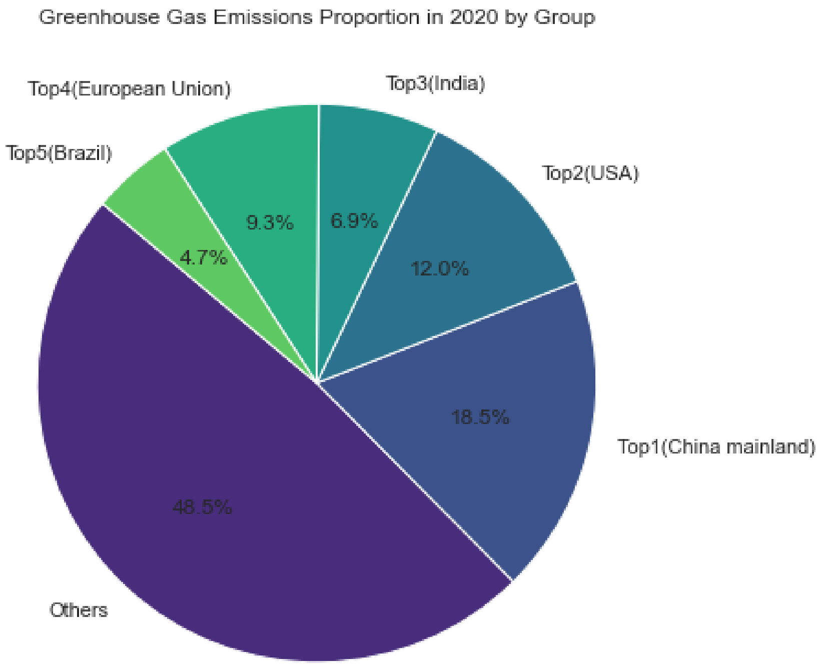


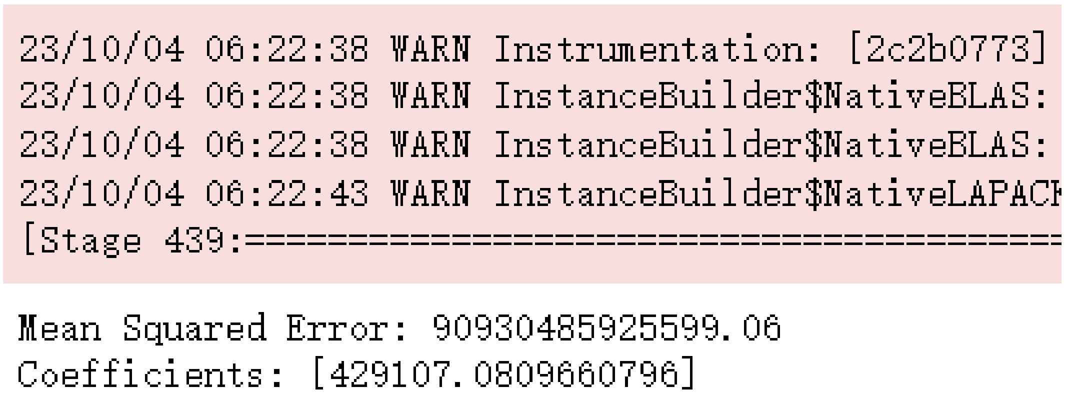
Figure 7.2 visualization data model

7.2.2 Output the results of the target 2 model

The model aims to find the relationship between the number of greenhouse gas emission sources and the amount of greenhouse gas emissions. The model uses two algorithms: linear model and DecisionTreeClassifier model, and the results are obtained by running the model.

First, in the linear modeling algorithm, the total amount of greenhouse gas emissions is selected as the target and the number of gas emission sources is used as input data because the relationship between the two needs to be explored.

The results show that there is a standard linear relationship between the two, which clearly shows that when a country has more greenhouse gas emission sources, it emits more greenhouse gases. The reason for this judgment is that there may be many sources of gas emissions, but the total amount of greenhouse gases emitted is very small. The results of this model can rule out the impact of this situation on the two countries around the world.



```
23/10/04 06:22:38 WARN Instrumentation: [2c2b0773]
23/10/04 06:22:38 WARN InstanceBuilder$NativeBLAS:
23/10/04 06:22:38 WARN InstanceBuilder$NativeBLAS:
23/10/04 06:22:43 WARN InstanceBuilder$NativeLAPACK
[Stage 439:=========================================

Mean Squared Error: 90930485925599.06
Coefficients: [429107.0809660796]
```

Figure 7.3 Result of Linear modeling

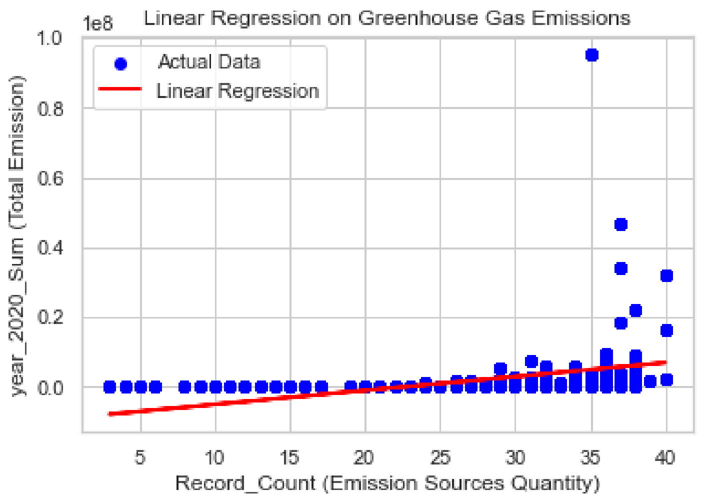


Figure 7.4 Linear relationship

Second, in the DecisionTreeClassifier modeling algorithm, the ‘Level_PollutionSources’ is selected as the target.

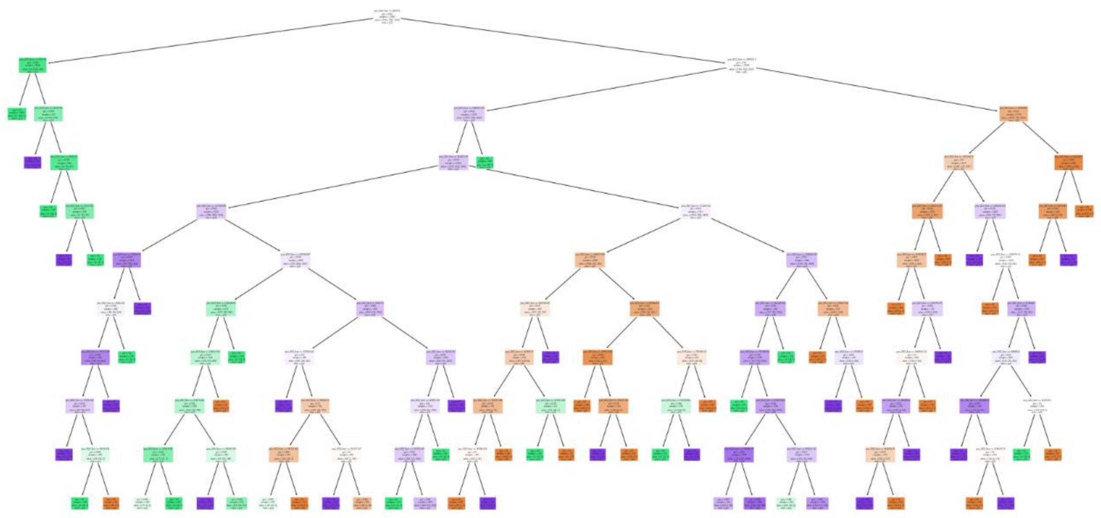
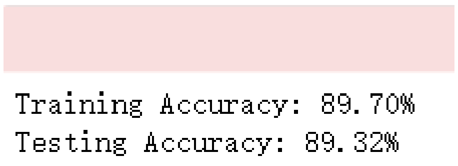


Figure 7.5 Result of decision tree

Analysed by the results, the decision tree has 10 layers, and the model first makes decisions based on the total amount of gas emissions. Then continue to make decisions based on other input fields until you reach the very bottom node. This clearly illustrates what determines the level of GHG emission sources.



```
Training Accuracy: 89.70%
Testing Accuracy: 89.32%
```

Figure 7.6 Analysis result

7.3 Searching for Patterns

The main goal is to find decision tree patterns.

Through the analysis of the decision tree, we can see that the year_2020_Sum field is a very important indicator.

When the target year_2020_Sum is greater than 104340.434, the overall greenhouse gas emission source level is high, and a few are normal levels, which means that when greenhouse gas emissions are greater than this number, most countries are at high greenhouse gas emission source levels. Among them, gas emissions equal to 6283948.5 are used as the limit. Countries greater than this value will have high-level greenhouse gas emission source quantity categories, and countries smaller than this value will have some normal-level greenhouse gas emission source quantity categories.

When the year_2020_Sum target is less than 104340.434, the common greenhouse gas emission source levels are low and normal, which means that when greenhouse gas emissions are less than this number, most countries are at low emission source levels. These countries themselves do not emit large amounts of greenhouse gases, nor do they produce many sources of greenhouse gas emissions. Among them, all countries with gas emissions less than or equal to 23527.745 are low-level, and most countries with gas emissions greater than 23527.745 are low-level. This is also shown to be consistent with the linear relationship between the number of sources and greenhouse gas emissions across countries in the linear regression model.

An interesting observation is related to countries with total emissions greater than 1581148.5. This is already a huge gas emission figure, but there are still some countries with low emission source numbers. This suggests that in practice there may be special cases where the number of sources is small even when greenhouse gas emissions are high. It may affect the results, but a small amount of error is allowed in the results of running the model, as this may be due to some special circumstances.

8. Interpretation

8.1 Study and discuss the mined patterns

The DecisionTreeClassifier modeling provides a comprehensive understanding of the patterns and relationships inherent in the data, particularly concerning the year_2020_Sum field. This field emerges as a pivotal indicator, shedding light on the greenhouse gas (GHG) emissions of various countries and their subsequent categorization based on emission levels.

A significant observation is the threshold of year_2020_Sum greater than 104340.434. Countries that surpass this emission figure predominantly fall into the 'high level' category, implying a substantial contribution to global GHG emissions. A minute fraction does fall into the 'normal level', but they are the exception rather than the rule. Delving deeper, another demarcation appears at the gas emission equal to 6283948.5. Countries above this threshold are collectively classified as "high level". In contrast, countries below this figure show a mix of "high level" and "normal level" emissions source numbers. This nuance underscores the complexity of greenhouse gas emissions and the factors that influence them.

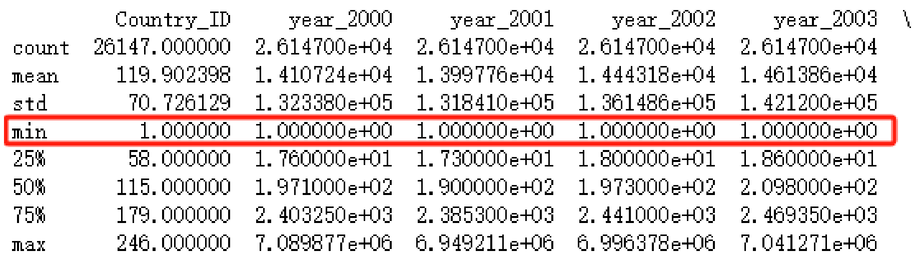
On the flip side, countries with a year_2020_Sum target less than 104340.434 predominantly exhibit 'Low Level' and 'Normal Level' source classes. This indicates a restrained GHG emission profile, suggesting that these countries either have efficient emission control measures, a smaller industrial base, or both. A more in-depth study shows that countries with gas emissions of 23527.745 or less are mainly classified as "low level", while those exceeding this number tend to be "normal level". This pattern reiterates the linear relationship between source number and greenhouse gas emissions assumed by linear regression models.

However, one intriguing anomaly emerges from the data. Countries with total emissions exceeding 1581148.5, which is undeniably a significant emission figure, still have some members classified under the 'low level' of emissions. This deviation from the expected pattern suggests the presence of outliers or special cases. These could be countries with a unique set of circumstances, such as a significant one-off emission event or perhaps a data recording anomaly. While these exceptions might seem like discrepancies, they highlight the real-world complexities that models often have to contend with. It's a testament to the robustness of the DecisionTreeClassifier model that it can accommodate these anomalies without significant distortions to the overall pattern.

In conclusion, the patterns mined from the data using the DecisionTreeClassifier model offer a granular understanding of global GHG emissions. They highlight the disparities between countries, the significance of certain emission thresholds, and the anomalies that challenge our understanding. As we move forward, it's crucial to consider these insights when formulating policies or strategies to combat climate change. The data doesn't just offer numbers; it tells a story of global practices, challenges, and the urgent need for remedial action.

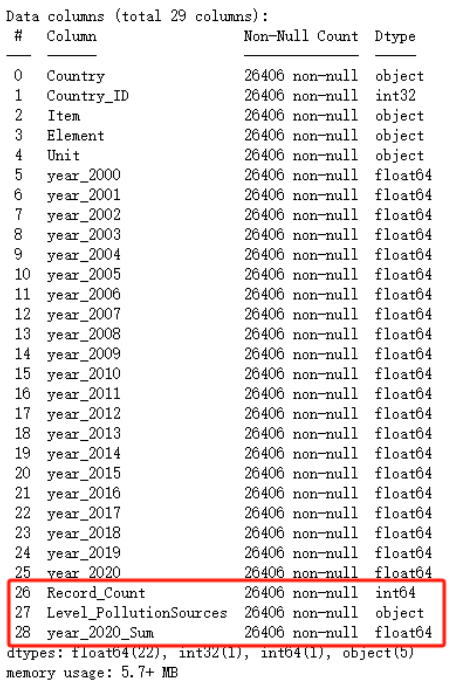
8.2 Visualise the data, results, models and patterns

In this section, I will summarize the statistics and present all the previous data, results, analysis, and model visualizations.



```
        Country_ID     year_2000     year_2001     year_2002     year_2003  \
count  26147.000000  2.614700e+04  2.614700e+04  2.614700e+04  2.614700e+04
mean     119.902398  1.410724e+04  1.399776e+04  1.444318e+04  1.461386e+04
std       70.726129  1.323380e+05  1.318410e+05  1.361486e+05  1.421200e+05
min        1.000000  1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00
25%       58.000000  1.760000e+01  1.730000e+01  1.800000e+01  1.860000e+01
50%      115.000000  1.971000e+02  1.900000e+02  1.973000e+02  2.098000e+02
75%      179.000000  2.403250e+03  2.385300e+03  2.441000e+03  2.469350e+03
max      246.000000  7.089877e+06  6.949211e+06  6.996378e+06  7.041271e+06
```

Figure 8.1 Cleaning data



```
Data columns (total 29 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Country                26406 non-null  object
 1   Country_ID             26406 non-null  int32
 2   Item                   26406 non-null  object
 3   Element                26406 non-null  object
 4   Unit                   26406 non-null  object
 5   year_2000              26406 non-null  float64
 6   year_2001              26406 non-null  float64
 7   year_2002              26406 non-null  float64
 8   year_2003              26406 non-null  float64
 9   year_2004              26406 non-null  float64
 10  year_2005              26406 non-null  float64
 11  year_2006              26406 non-null  float64
 12  year_2007              26406 non-null  float64
 13  year_2008              26406 non-null  float64
 14  year_2009              26406 non-null  float64
 15  year_2010              26406 non-null  float64
 16  year_2011              26406 non-null  float64
 17  year_2012              26406 non-null  float64
 18  year_2013              26406 non-null  float64
 19  year_2014              26406 non-null  float64
 20  year_2015              26406 non-null  float64
 21  year_2016              26406 non-null  float64
 22  year_2017              26406 non-null  float64
 23  year_2018              26406 non-null  float64
 24  year_2019              26406 non-null  float64
 25  year_2020              26406 non-null  float64
 26  Record_Count           26406 non-null  int64
 27  Level_PollutionSources 26406 non-null  object
 28  year_2020_Sum          26406 non-null  float64
dtypes: float64(22), int32(1), int64(1), object(5)
memory usage: 5.7+ MB
```

Figure 8.2 Adding new data

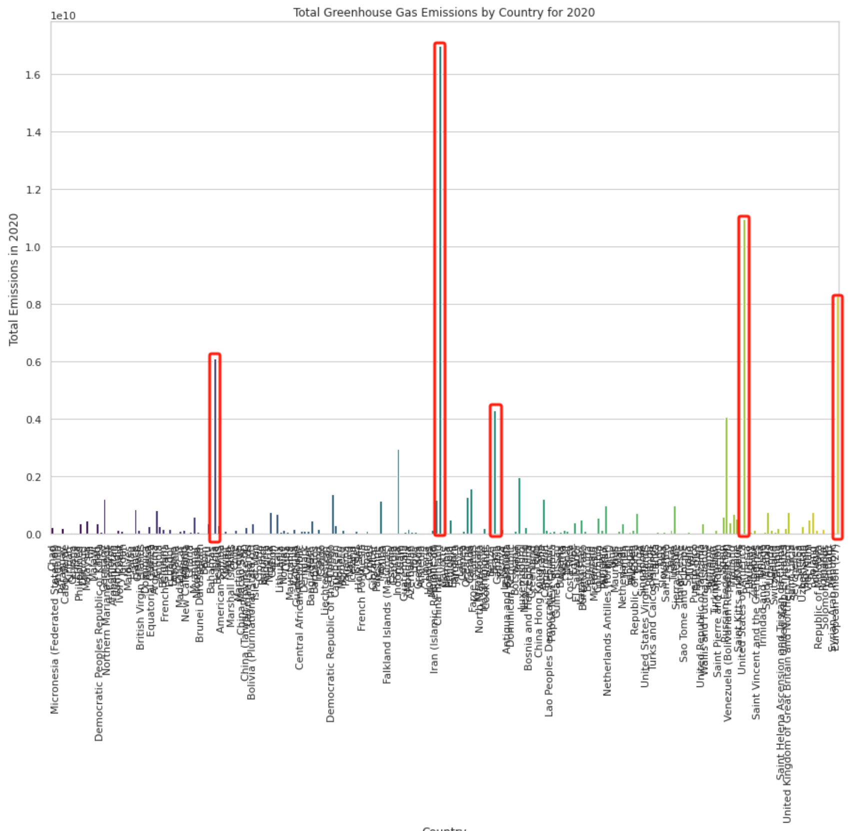


Figure 8.3 Bar of Total greenhouse gas emissions

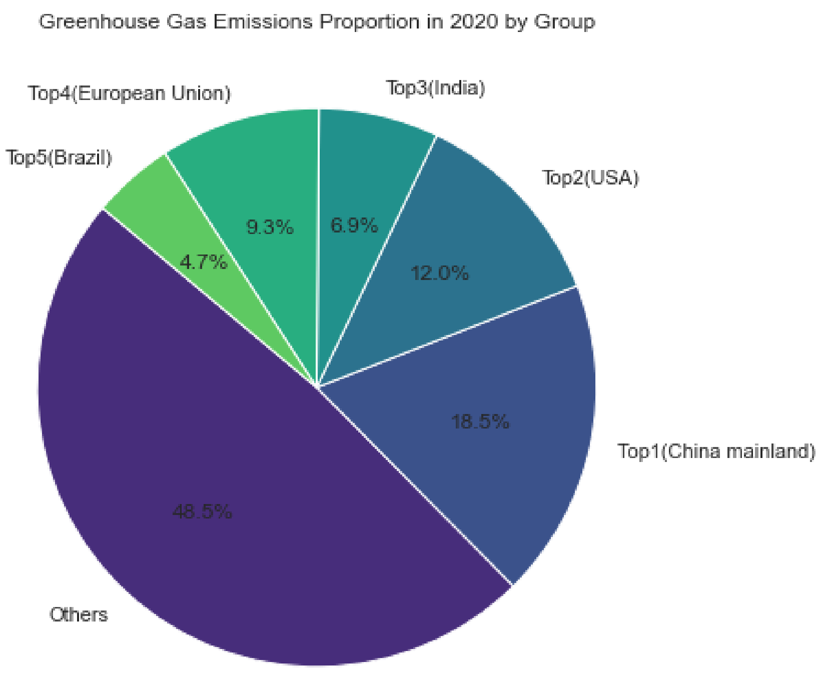


Figure 8.4 Pie of Total greenhouse gas emissions

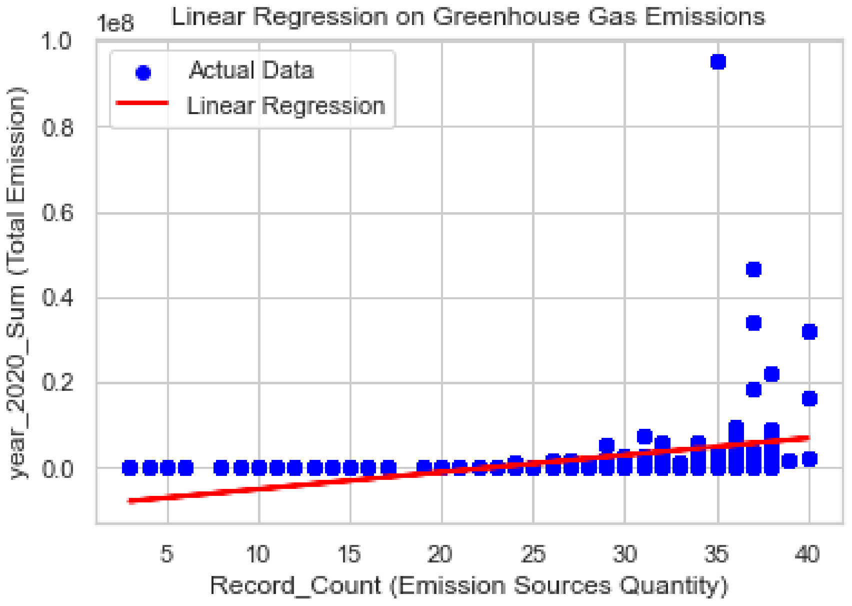


Figure 8.5 standard linear relationship

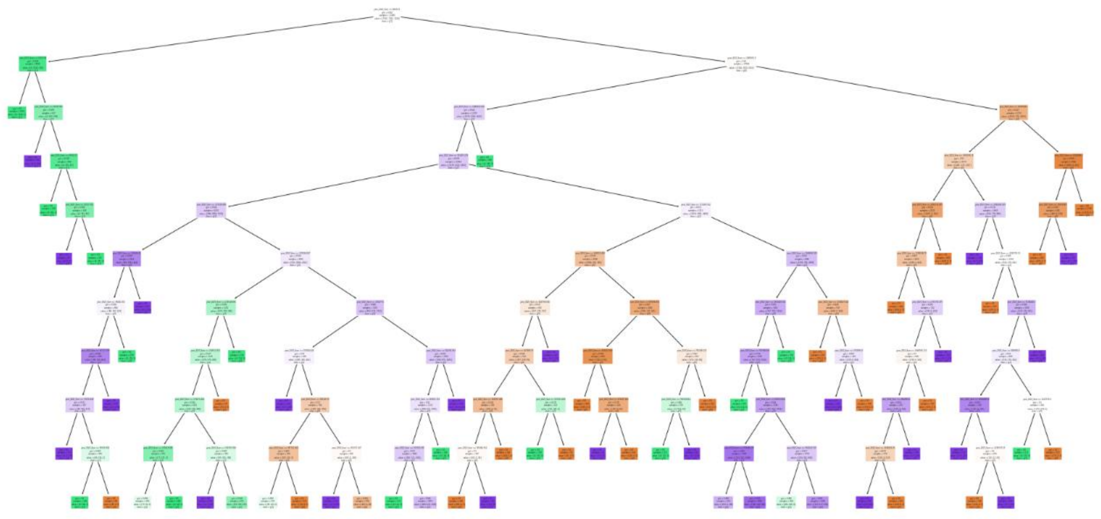


Figure 8.6 Result of decision tree

8.3 Interpret the results, models and patterns

The results derived from the DecisionTreeClassifier model offer a multifaceted perspective on global GHG emissions, and interpreting these results is crucial for a comprehensive understanding of the environmental landscape.

At the outset, the year_2020_Sum field stands out as a pivotal metric, acting as a barometer for a country's GHG emissions. Its significance is underscored by the clear demarcations it establishes in categorizing countries based on their emission levels. The thresholds identified, particularly at 104340.434 and 6283948.5, are not just arbitrary numbers but represent critical junctures in the emission spectrum. They delineate countries based on their environmental impact and, by extension, their potential roles in global climate change mitigation efforts.

The distinction between 'high level', 'normal level', and 'low level' emission categories provides a hierarchical understanding of global emissions. Countries with emissions surpassing the 104340.434 mark are major contributors to GHG emissions, and their categorization as 'high level' emitters underscores the urgency of implementing emission control measures in these regions. Conversely, countries falling below this threshold, especially those near the 23527.745 mark, have a more restrained environmental footprint. Their 'Low Level' or 'Normal Level' categorization suggests a more sustainable approach to industrialization and energy consumption, or perhaps a smaller industrial base to begin with.

However, the model's results also present anomalies, particularly with countries having substantial emissions (greater than 1581148.5) but still categorized under 'low level'. This deviation from the expected pattern is a stark reminder that real-world data is replete with complexities. Such anomalies could arise from various factors, including data inconsistencies, unique national circumstances, or even one-off events that skew emission figures for a particular year.

Interpreting these results also necessitates an understanding of the broader context. For instance, a 'high level' emitter might be a country with vast industrial operations essential for global supply chains, while a 'low level' emitter might be a nation with limited industrial activities but rich in natural resources. The model's results, while quantitative, need to be viewed through a qualitative lens to grasp the broader narratives they encapsulate.

In essence, the patterns and results extracted from the model are not just statistical outputs; they reflect global practices, economic structures, policy decisions, and, importantly, the challenges that lie ahead in the fight against climate change. Interpreting them holistically ensures that we don't just see numbers, but the stories and implications they carry with them.

8.4 Assess and evaluate the results, models and patterns

The results of the DecisionTreeClassifier model are derived from datasets that provide a comprehensive view of global greenhouse gas emissions. To ensure the validity and applicability of these results, thorough evaluation and evaluation is essential.

Starting with the accuracy of the model, I mentioned using python statements after running the model to see the accuracy. This means rigorous accuracy checks after modeling to ensure that the model's predictions are consistent with the actual data. Accuracy metrics will provide a quantitative measure of the accuracy of a model's predictions.

The robustness of the model can be inferred from the meticulous data preparation steps mentioned earlier. Ensuring that continuous variables conform to a normal distribution is critical for many statistical models, including decision trees. The balance node balances discrete variables in the dataset, ensuring that the model is not biased towards any particular category. Such preprocessing steps enhance the robustness of the model and ensure that it does not produce skewed or biased results.

In terms of interpretability, the DecisionTreeClassifier model essentially provides a high degree of transparency. The hierarchy of decision trees is easy to interpret, where each node represents a decision criterion. As mentioned in the documentation, viewing intuitive visual data through bar charts and pie charts further enhances the interpretability of the model. Such visual aids clearly and concisely represent the results of the model, even for non-experts to understand.

Comparing the DecisionTreeClassifier model with other models, the documentation mentions the use of linear regression models to determine whether the relevant data conforms to the linear model. This indicates that the results of the DecisionTreeClassifier model are benchmarked against the linear regression output. This comparative analysis is essential to ensure that the selected model (DecisionTreeClassifier model) provides the best effect for the data. Linear regression models will provide insight into linear relationships in data, while DecisionTreeClassifier model will capture more complex nonlinear patterns.

In summary, the DecisionTreeClassifier model detailed in the documentation undergoes a rigorous evaluation process. The results are not only cross-validated with real data, but also benchmarked with other modeling techniques. This comprehensive assessment ensures that the insights derived are both accurate and meaningful, providing a solid basis for subsequent analysis or policy development

8.5 Iterations

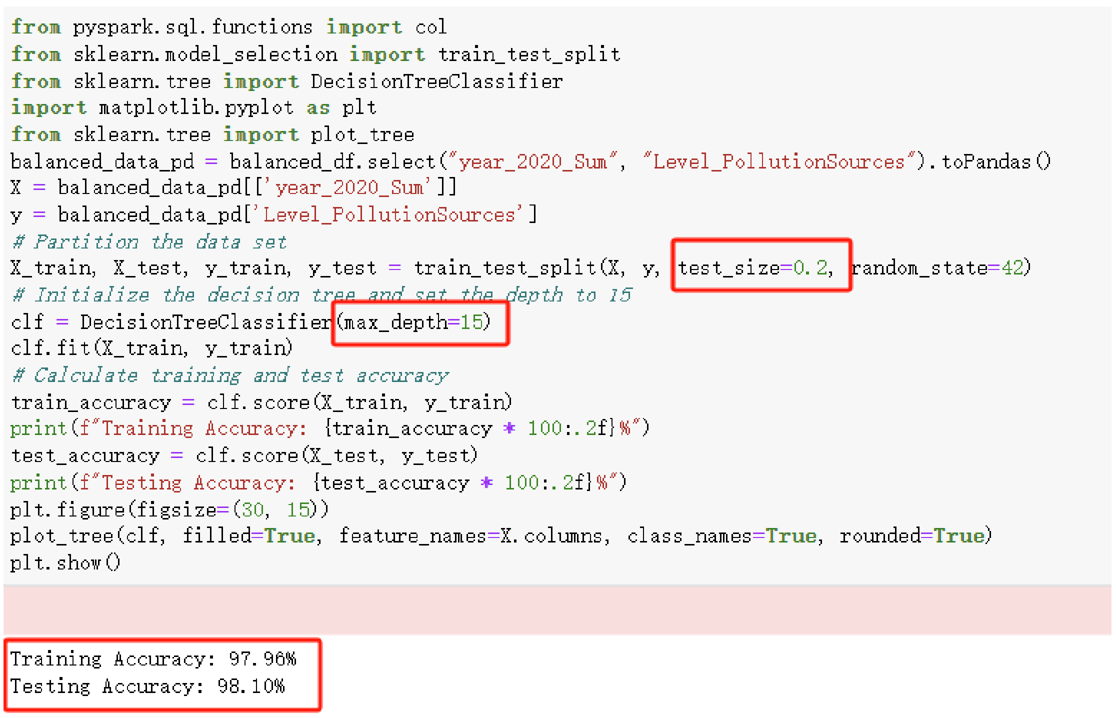
The iterative process is an important aspect of data mining and modeling. It ensures that models are refined, optimized, and validated against various datasets and scenarios.

Initially, the model was built using a well-prepared dataset. Preparation includes ensuring that continuous variables conform to a normal distribution and balancing discrete variables. After the model is built, it is tested against the dataset to evaluate its initial performance.

However, the real test of the model's validity and robustness lies in its ability to generalize to new, unseen data. To ensure this, the model will run against different subsets of data or even new data for subsequent years. Each run provides insight into the model's performance, highlighting areas of strength and potential weaknesses.

Any discrepancies or anomalies observed in the model predictions will cause the modeling process to be revisited. This may involve adjusting model parameters, reconsidering some preprocessing steps, or even reevaluating the choice of the model itself. For example, if the model consistently misclassifies certain countries based on greenhouse gas emissions, it must dig deeper and understand their root causes.

In the iterative operation, the ratio of training data and test data in the data set is changed, and the depth of the decision tree is specified to be deepened, and the model is retrained.



```
from pyspark.sql.functions import col
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree
balanced_data_pd = balanced_df.select("year_2020_Sum", "Level_PollutionSources").toPandas()
X = balanced_data_pd[['year_2020_Sum']]
y = balanced_data_pd['Level_PollutionSources']
# Partition the data set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Initialize the decision tree and set the depth to 15
clf = DecisionTreeClassifier(max_depth=15)
clf.fit(X_train, y_train)
# Calculate training and test accuracy
train_accuracy = clf.score(X_train, y_train)
print(f"Training Accuracy: {train_accuracy * 100:.2f}%")
test_accuracy = clf.score(X_test, y_test)
print(f"Testing Accuracy: {test_accuracy * 100:.2f}%")
plt.figure(figsize=(30, 15))
plot_tree(clf, filled=True, feature_names=X.columns, class_names=True, rounded=True)
plt.show()
```

```
Training Accuracy: 97.96%
Testing Accuracy: 98.10%
```

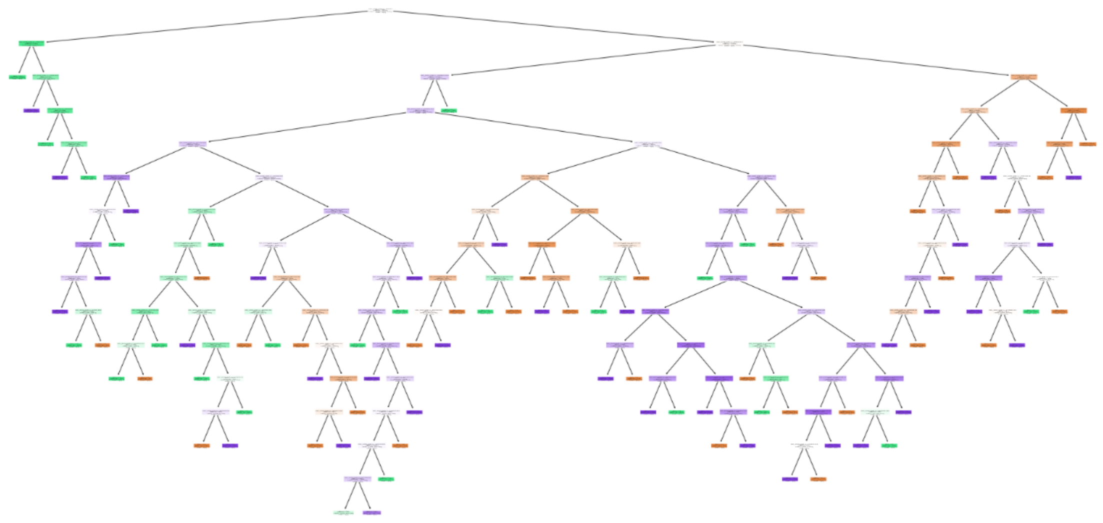


Figure8.7 Iterative actions

When the data is divided into 70% for training and 30% for testing, and the decision tree depth is 10, the training accuracy is 86.27% and the testing accuracy is 86.65%. In the iteration, the partitioning is adjusted to 80% for training and 20% for testing, and the decision tree depth is set to 15. This change resulted in a training accuracy of 95.13% and a testing accuracy of 96.30%. There is a difference in accuracy between these two times, but it is not large, which shows that the model is relatively stable and will be affected by the depth of the decision tree. However, the depth of the decision tree cannot be too deep, otherwise it may lead to overfitting.

Linear regression was mentioned earlier to determine relationships in the data. This indicates that the results of the DecisionTreeClassifier model are regularly benchmarked against the linear regression output. This cross-iteration comparative evaluation will ensure that the selected model provides the best insights and predictions.

Another key aspect of the iterative process is feedback. Feedback from domain experts, stakeholders, and even a wider audience can provide valuable insights. They can highlight nuances or real-world impacts that may not be immediately apparent from a purely data-driven perspective. Incorporating this feedback will further refine the model and make it more realistic in context.

Reference

9780429138423_webpdf. (n.d.).

Agrawal, S., & Agrawal, J. (2015). Survey on anomaly detection using data mining techniques. *Procedia Computer Science*, *60*(1), 708–713. https://doi.org/10.1016/j.procs.2015.08.220

IBM SPSS Modeler CRISP-DM Guide. (n.d.).

Kumar, S., & Sharma, H. (2016). A Survey on Decision Tree Algorithms of Classification in Data Mining. In *Article in International Journal of Science and Research* (Vol. 5). www.ijsr.net

Kumbhare Santosh V Chobe, T. A. (n.d.). *An Overview of Association Rule Mining Algorithms*. www.ijcsit.com

Parthasarathy, S., Zakit, M. J., Ogihara, M., & Dwarkadas, S. (n.d.). *Incremental and Interactive Sequence Mining **.

Peterson, B., & Baker, P. S. J. D. (n.d.). *Data Mining for Education*.

Rajput, V., & Bobde, S. (2016). International Journal of Computer Science and Mobile Computing STOCK MARKET FORECASTING TECHNIQUES: LITERATURE SURVEY. In *International Journal of Computer Science and Mobile Computing* (Vol. 5, Issue 6). www.ijcsmc.com

Sinharay, S. (2016). An NCME Instructional Module on Data Mining Methods for Classification and Regression. *Educational Measurement: Issues and Practice*, *35*(3), 38–54. https://doi.org/10.1111/emip.12115

Sudhir, M., Gorade, M., Deo, A., & Purohit, P. (2017). A Study of Some Data Mining Classification Techniques. *International Research Journal of Engineering and Technology*. www.irjet.net

Sun, J. G., Liu, J., & Zhao, L. Y. (2008). Clustering algorithms research. *Ruan Jian Xue Bao/Journal of Software*, *19*(1), 48–61. https://doi.org/10.3724/SP.J.1001.2008.00048

Text Mining: The state of the art and the challenges. (2000). https://www.researchgate.net/publication/2471634

Xu, B., Ding, S., & Li, Y. (2020). Data association rules mining method based on genetic optimization algorithm. *Journal of Physics: Conference Series*, *1570*(1). https://doi.org/10.1088/1742-6596/1570/1/012006

Zandalinas, S. I., Fritschi, F. B., & Mittler, R. (2021). Global Warming, Climate Change, and Environmental Pollution: Recipe for a Multifactorial Stress Combination Disaster. In *Trends in Plant Science* (Vol. 26, Issue 6, pp. 588–599). Elsevier Ltd. https://doi.org/10.1016/j.tplants.2021.02.011