School of Computer Science and Engineering

J Component report

Programme : B.Tech

Course Title: Social and Information Networks

Course Code: CSE3021

Slot: C2+TC2

Title: Analysis On COVID-19 Impact on the International Trade

Team Members: GUGGILAM AMARNATH / 20BCE1543

JAYANTH REDDY SIDDENKI / 20BCE1793

Faculty: Dr Punitha K

Sign:

Date:

A project report on

Analysis On COVID-19 Impact on Trade

Submitted in partial fulfilment for the course

Social Information Networks

by

GUGGILAM AMARNATH (20BCE1543)

JAYANTH REDDY (20BCE1793)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

April, 2023

DECLARATION

We hereby declare that the thesis entitled "Analysis on Covid 19 Impact on International Trade" submitted by Amarnath and Jayanth, for the completion of the course, Social Information Networks is a record of bonafide work carried out by me under the supervision of Dr Punitha K, our course instructor. We further declare that the work reported in this document has not been submitted and will not be submitted, either in part or in full, for any other courses in this institute or any other institute or university.

Place: Chennai

Amarnath

Jayanth Reddy

Signature of candidates

School of Computer Science and Engineering

CERTIFICATE

This is to certify that the report entitled **"Analysis on Covid 19 Impact on International Trade"** isprepared and submitted by Amaranth (**20BCE543**) and Jayanth Reddy (20BCE1793) to Vellore Institute of Technology, Chennai, in partial fulfilment of the requirement for the course, **Social Information Networks,** is a bonafide record carried out under my guidance. The project fulfills the requirements as per the regulations of this University and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and willnot be submitted either in part or in full, for the any other course and the same is certified.

Name: Dr.Punitha K Signature of the Faculty

Date:

ABSTRACT

International trade is the exchange of goods, services and capital across the countries and territories and is usually based on a set of complex relationships between different countries that can be modeled as an extremely dense network of interconnected entities. So in this project we would like to analyze the impact of COVID-19 on international trade by comparing the different network analysis measures on pre covid and post covid global trade data. This would help us to conclude the impact of Covid on the trade which plays a major role in countries growth and relation among the countries.

ACKNOWLEDGEMENT

In successfully completing this project, many people have helped me. I would like to thank all those who are related to this project.

I would like to thank my teacher **(Dr. Punitha)** who gave me this opportunity to work on this project and also for her patience, motivation, enthusiasm, and immense knowledge. Her guidance helped me in all the time of research and implementation of the project. I got to learn a lot from this project about how to search for the best datasets for the projects, how to make your project more effective, which will be very helpful in solving some real world problems and many other things. I would also like to thank our school HOD.

At last, I would like to extend my heartfelt thanks to my parents because without their help this project would not have been successful. Finally, I would like to thank my dear friends and my fellow students for many helpful discussions and good ideas along the way who have been with me all the time.

CONTENTS

List of Tables

CHAPTER-1

Introduction

The COVID-19 pandemic produced an enormous impact on human society with multiple health, social and economic effects. The impact on international trade is enormous. Thus, the World Trade Organization (WTO) states that “the COVID-19 pandemic represents an unprecedented disruption to the global economy and world trade, as production and consumption are scaled back across the globe”. Several reports of the WTO and the United Nations (UN) highlight the effects of the COVID-19 for world trade. A need for an effective public health response to the collapse of global trade is elucidated in.

The WTO and the UN reports depict the global change of world trade induced by the COVID-19. However, it is also important to analyse, how the trade relations between the countries are affected by the pandemic. with this aim Using the data of different product trades across the countries and their trade values we will be calculating different network analysis measures such as centrality measures then based on the measures, we will try to conclude whether there was a positive or negative impact of covid on trade. This would be very useful as trade plays an important role in the country’s GDP.

Advantages Of Trade Analysis

- Understanding the extent of the impact: By analysing the impact of COVID-19 on trade, we can gain a better understanding of the extent to which it has affected different sectors and regions. This information can help governments, businesses, and other organizations make more informed decisions about how to respond to the crisis.
- Identifying vulnerabilities: Studying the impact of COVID-19 on trade can help identify vulnerabilities in supply chains and trade networks. This information can help businesses and governments make adjustments to mitigate future disruptions.

- Understanding the impact of COVID-19 on trade can help businesses and governments plan for the future. This includes preparing for future pandemics and other global crises, as well as making strategic decisions about investment and trade.
- Overall, studying the impact of COVID-19 on trade can provide valuable insights and information that can help individuals, organizations, and governments respond to the crisis and prepare for the future.

Overview of Analysis on Covid 19 Impact on international Trade

The suggested system's outline is presented in this section. For this experiment, datasets were taken from UN Comtrade which is an online platform for using the real time databases. Many algorithms were applied to each datasets for the implementation. In this implementation the mainoutput is totally dependent on the dataset which we are using in this process.

Dataset –

We will be using UN Comtrade data (https://comtrade.un.org/data) of multiple product trades in the year 2018 and 2020 which has information regarding the imports and exports between different countries and their respective trade value. As it is a real time dataset we need to clean the data and then we will be applying different network analysis measures on the data.

PROPOSED WORK

- Network Analysis –

  In Network Analysis we will be implementing Trade Network Analysis, Eigen vector Centrality, Betweenness Centrality, Closeness Centrality, Eccentricity, Page rank, for analysis of the dataset to find the relationship between countries depending upon their imports, exports and trade value

- Pre-processing –

  The data pre-processing refers to the process of preparing raw data to make it appropriate for the construction and training of Machine Learning models. Here the job prediction datasets are preprocessed so as to improve the working of the system, as in this process the raw data is processed into proper data.

- Extracting features –

  Feature Extraction is a technique for reducing the number of features in a dataset by generating new ones from existing ones. The original set of features should then be able to summarize the majority of the information in the new reduced set of features.

- Trained data & test data –

  A training set is a subset of a larger data set that is used to fit a model for predicting or classifying values that are known in the training set but unknown in the rest of the data. The training set is used in conjunction with the validation and/or test sets to assess various models.

  A test set is a subset of a data set that is used in data mining to evaluate the likely future performance of a single prediction or classification model that has been chosen from a pool of competing models based on its performance with the validation set.

- Regression –

Regression is a method for understanding the relationship between independent variables or features and a dependent variable or outcome. Outcomes can then be predicted once the relationship between independent and dependent variables has been estimated.

- Performing Algorithm –

Here we will be performing various regression algorithms and analysis algorithms for visualizing the data and comparing the values with the previous values. Here we will also find the prediction for the similar products to be exported

- Packages –

Packages used in job prediction are as follows –

1) Pandas
2) NumPy
3) Matplotlib
4) Sklearn
5) Apriori
6) Association rule
7) Matplotlib
8) Seaborn
9) Linear Regression

Pandas –

Pandas are widely used open-source Python library for data science, data analysis, and machinelearning activities

Numpy –

NumPy (numerical Python) is a library that consists of multidimensional array objects and a collection of functions for manipulating them. NumPy allows you to conduct mathematical and logical operations on arrays.

Matplotlib –

Matplotlib is a Python package that allows you to create static, animated, and interactive visualisations.

Sklearn –

Sklearn is a Python-based machine learning package that is available for free. Support-vector machines, random forests, gradient boosting, and k-means are among the classification, regression, and clustering algorithms included.

Apriori –

Apriori algorithm is a machine learning model used in Association Rule Learning to identify frequent itemsets from a dataset. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database.

Association rule-

Association rule mining is a technique used to uncover hidden relationships between variables in large datasets. It is a popular method in data mining and machine learning and has a wide range of applications in various fields, such as market basket analysis, customer segmentation, and fraud detection.

Matplotlib-

Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open source alternative to MATLAB.

Seaborn-

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures. Seaborn helps you explore and understand your data.

Linear Regression-

Linear regression uses the relationship between the data-points to draw a straight line through all them. This line can be used to predict future values.

METHODOLOGY

NETWORK ANALYSIS (USING GEPHIE)

Trade Network Analysis-

The network is a mathematical representation of the state of a system at a given point in time in terms of nodes and links . In this project, we examine the trade interconnectedness and density using this methodology. Network analysis sees the relationship between different elements in terms of nodes and edges.

Degree Defined as the number of total edges connected with that vertex; it includes both arrows pointing toward it as well as arrows going outward from it.

In-degree Defined as the total number of arrows pointing toward the node; it represents import trade, that is, trade flowing toward the country (vertex).

Out-degree Defined as the total number of arrows pointing away from the node; it represents export trade, that is, trade flowing away from the country (vertex).

EigenVector Centrality-

Indicates how important a node is to the nodes around it; countries that carry a high value of eigenvector centrality are the ones that are connected to many other countries which are, in turn, connected to many others.

Betweenness Centrality-

The betweenness centrality for each vertex is the number of shortest paths that pass through the vertex.

Closeness Centrality-

A variable that tells how close one node (in terms of topological distance) is with respect to all other nodes. The smallest path connecting country i and country j is denoted by the geodesic distance between i and j.

Eccentricity-

The eccentricity of a node v in a network is the maximum distance from v to any other node.

Page Rank -

PageRank centrality is a variant of Eigen Centrality designed for ranking web content, using hyperlinks between pages as a measure of importance. It can be used for any kind of network, though.

Modularity-

Modularity is a system property which measures the degree to which densely connected compartments within a system can be decoupled into separate communities or clusters which interact more among themselves rather than other communities.

Graph Density-

Graph density represents the ratio between the edges present in a graph and the maximum number of edges that the graph can contain.

REGRESSION (USING PIPELINE)

Modelling the relationship between a dependent variable and one or more independent variables is a type of statistical analysis known as regression. Regression aims to forecast the dependent variable's value based on the values of the independent variables.

A machine learning workflow can be organised and automated using a pipeline. Data pre-processing, feature selection, model training, and model evaluation could be included in a regression pipeline. A regression pipeline looks something like this

Data preparation: Preparation of the data is the initial step. The data may need to be cleaned, variables changed, and the data divided into training and testing sets.

The next step is to choose the features that will be incorporated into the regression model. This can entail applying strategies like dimensionality reduction, feature importance ranking, or correlation analysis.

Regression model training: Using the practise data, a regression model is trained in the third stage. It might be necessary to use machine learning techniques like decision trees or neural networks, as well as methods like linear regression, polynomial regression, or other techniques.

Evaluation of the model: The regression model's performance on the test data is assessed as the last stage. This can entail applying measurements like accuracy, R-squared, or mean squared error.

We can automate the creation and testing of regression models by structuring these stages using a pipeline. This can reduce errors, save time, and make it easier to experiment with different techniques and Algorithms.

SENTIMENTAL ANALYSIS

Sentiment analysis is a machine learning tool that analyses texts for polarity, from positive to negative. There are a number of techniques and complex algorithms used to command and train machines to perform sentiment analysis. There are pros and cons to each. But, used together, they can provide exceptional results. The main purpose of sentiment analysis is to classify text data into positive, negative, or neutral sentiment categories. This information can then be used to gain insights into customer feedback, brand reputation, market trends, and public opinion. The machine learning models can be trained using various techniques, such as logistic regression, support vector machines, and neural networks. Once the model is trained, it can be used to classify new text data into sentiment categories

MARKET BASKET ANALYSIS:

Market basket analysis is a data mining technique, It involves analysing large data sets, such as purchase history, to reveal product groupings, as well as products that are likely to be purchased together.

In market basket analysis, association rules are used to predict the likelihood of products being purchased together. Association rules count the frequency of items that occur together, seeking to find associations that occur far more often than expected.

Algorithms that use association rules include AIS, SETM and Apriori. The Apriori algorithm is commonly cited by data scientists in research articles about market basket analysis and is used to identify frequent items in the database, then evaluate their frequency as the datasets are expanded to larger sizes.

MULTIPLE LINEAR REGRESSION

Multiple Linear Regression is a machine learning algorithm where we provide multiple independent variables for a single dependent variable. Several circumstances that influence the dependent variable simultaneously can be controlled through multiple regression analysis. Regression analysis is a method of analyzing the relationship between independent variables and dependent variables. n multiple linear regression, the dependent variable is the outcome or result from you're trying to predict. The independent variables are the things that explain your dependent variable.

Implementation –

TRADE NETWORK ANALYSIS

Every node has in-degree and out-degree. At the outset, we construct the trade structure from a network perspective which involves building centrality parameters, such as degree, closeness, eigenvector, and overall density. Where degree and closeness indicate the interconnectedness and geodesic distance, respectively. Whereas an eigenvector measures network connectivity . We will construct the network by preparing an undirected network matrix with N = 15 countries for the years 2019 and 2021.

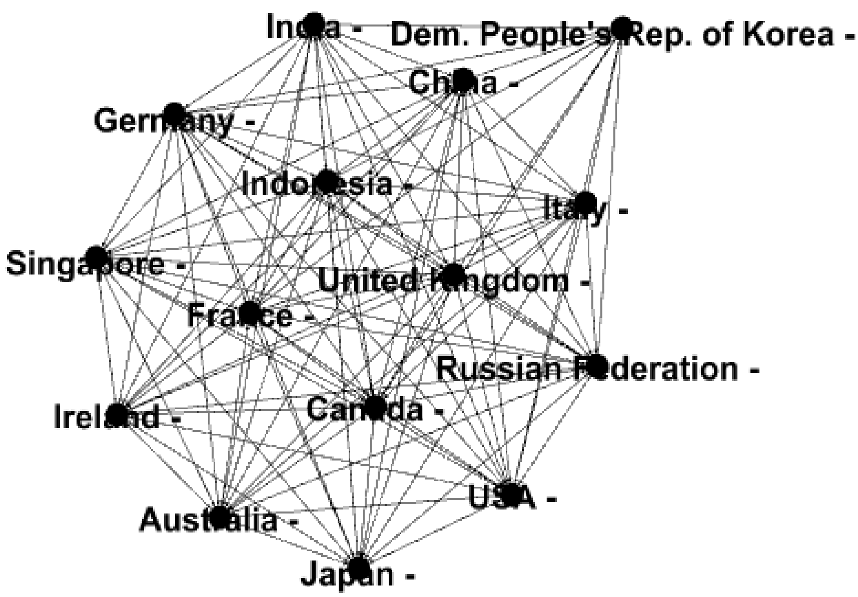


Figure 1.1 Trade-Network in 2019. This figure shows the network graph derived from Network Analysis for the period 2018. The number of countries included are the USA, Australia, Germany, France, Japan, South Korea, China, Italy, India, Indonesia, UK, Canada , Singapore, Russia and, Netherland.

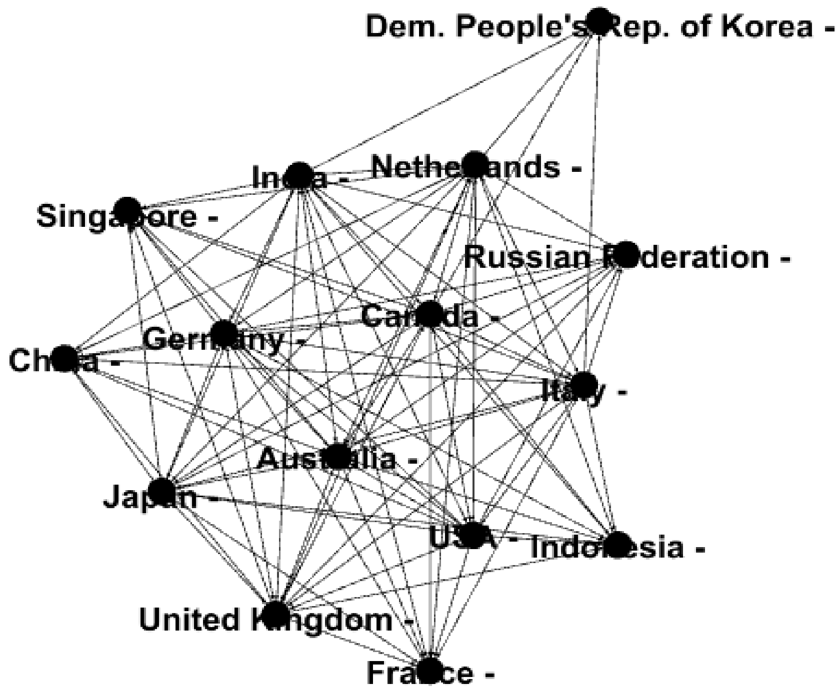


Figure 1.2 Trade-Network in 2021. This figure shows the network graph derived from Network Analysis for the period 2021. The number of countries included are the USA, Australia, Germany, France, Japan, South Korea, China, Italy, India, Indonesia, UK, Canada , and Singapore, Russia, Netherlands.

Country	indegree		outdegree		degree	
	2019	2021	2019	2021	2019	2021
Australia	13	8	13	13	26	21
Italy	13	8	14	14	27	22
Japan	13	8	13	13	26	21

Netherlands	13	8	13	13	26	21
Singapore	13	9	13	0	26	9
Germany	13	8	14	13	27	21
India	13	8	14	14	27	22
Dem. People's Rep. of Korea	10	4	0	0	20	4
USA	13	8	13	13	27	21
China	13	9	14	0	27	9
Canada	13	8	14	14	27	22
France	13	9	14	0	27	9
Indonesia	13	9	14	0	27	9
Russian Federation	13	9	13	0	26	9
United Kingdom	13	7	14	13	27	20

Table 1.1 indegree, outdegree and degree parameters of Trade data, 2019 and 2021.

EIGENVECTOR CENTRALITY

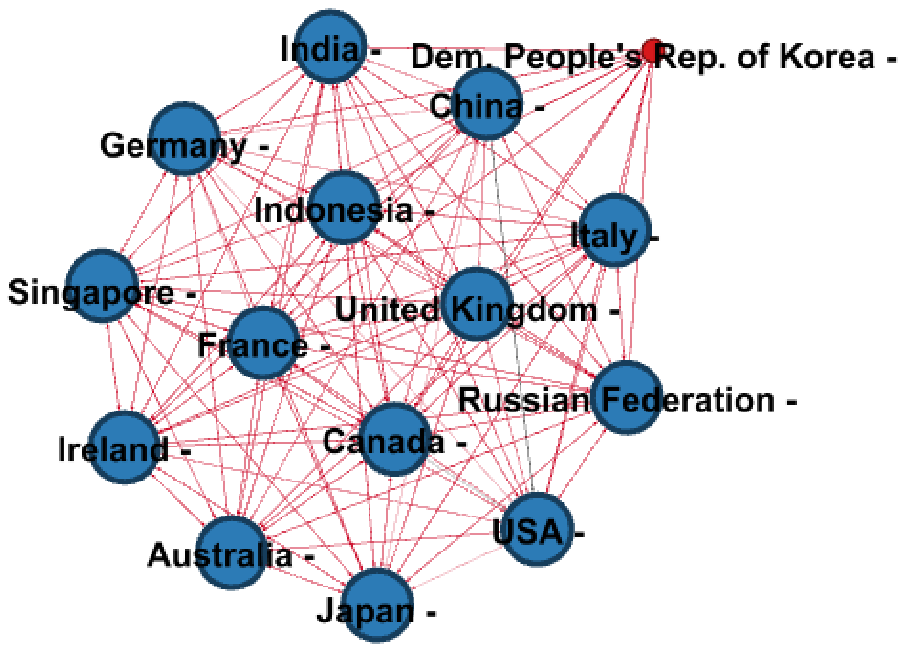


Figure 1.3 Eigenvector centrality on 2019 trade data

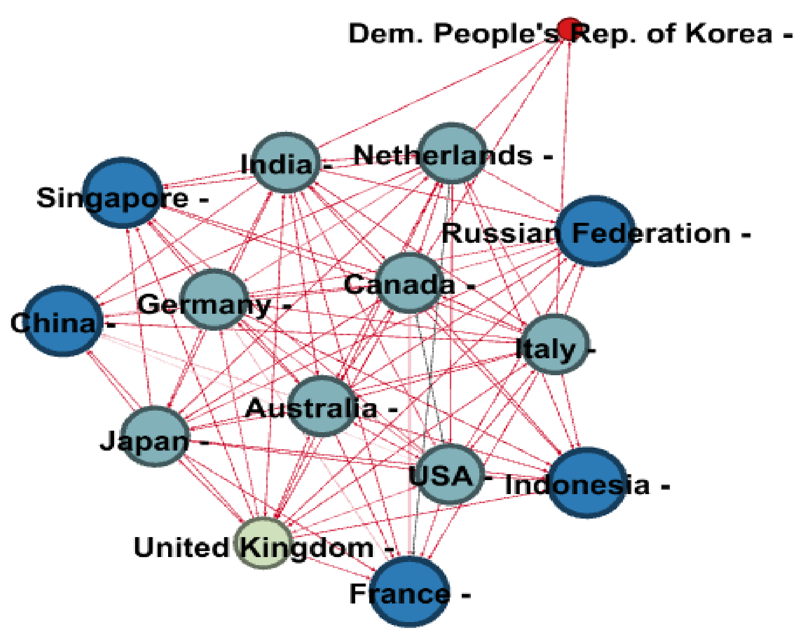
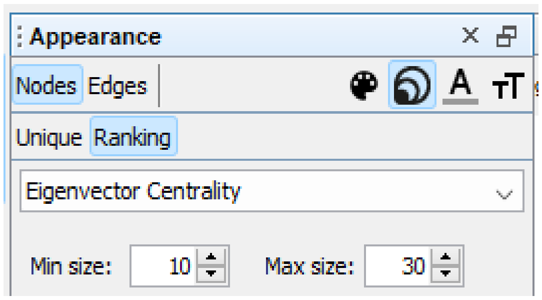
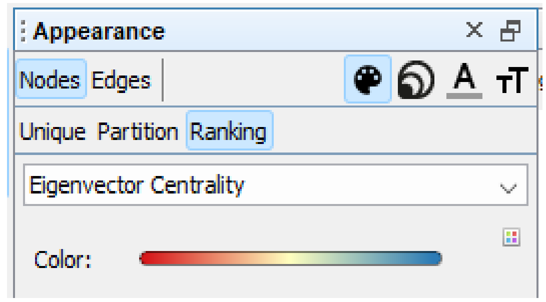


Figure 1.4 Eigenvector centrality on 2021 trade data



Color



Size

Country	Eigenvector Centrality	
	2018	2021
Australia	1	0.887491
Italy	1	0.887491
Japan	1	0.887491
Netherlands	1	0.887491
Singapore	1	1
Germany	1	0.887491
India	1	0.887491
Dem. People's Rep. of Korea	0.769231	0.450035
USA	1	0.887491
China	1	1

Canada	1	0.887491
France	1	1
Indonesia	1	1
Russian Federation	1	1
United Kingdom	1	0.787561

Table 1.2. Eigenvector Centrality Parameters, 2018 and 2021.

BETWEENNESS CENTRALITY

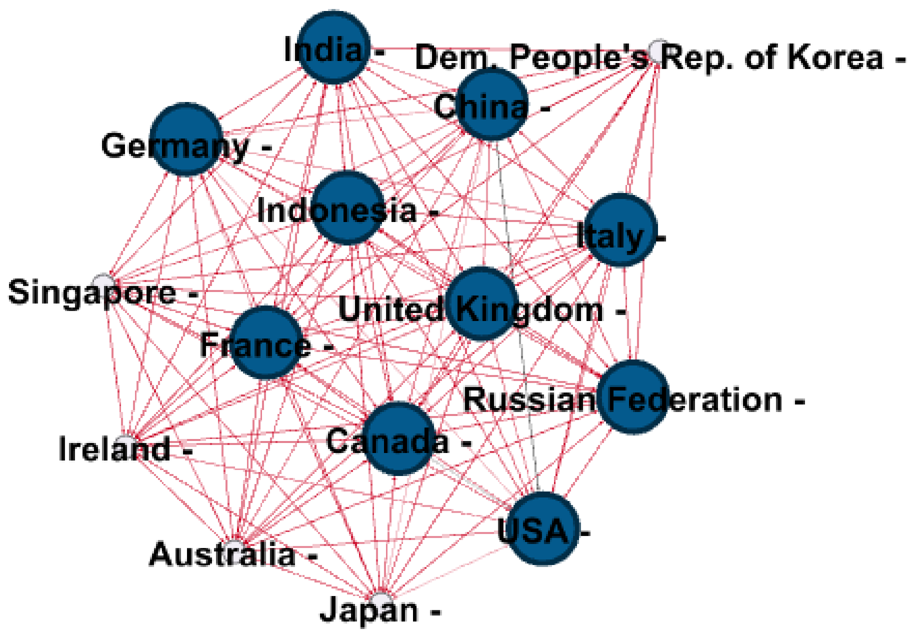


Figure 1.5. Betweenness Centrality of trade data in 2018

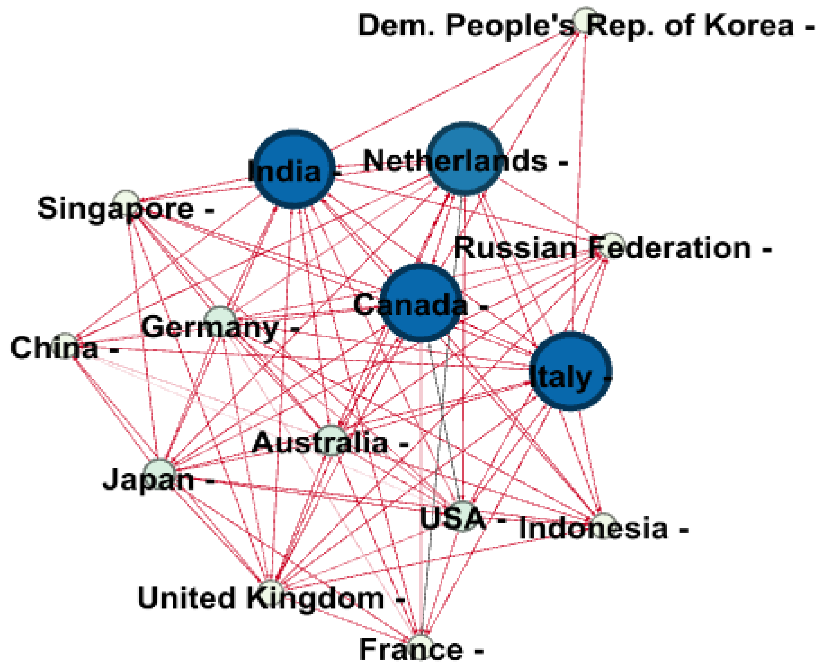
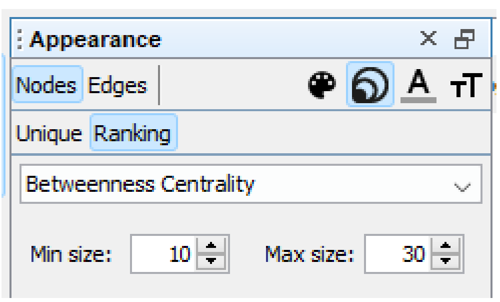
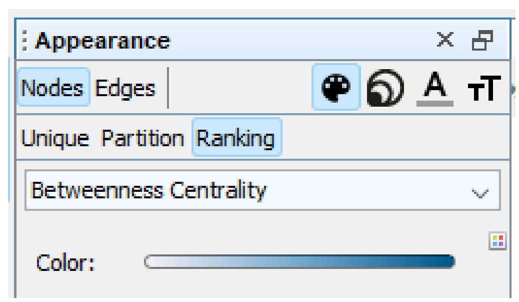


Figure 1.6. Betweenness Centrality of trade data in 2021



Color



Size

Country	Betweenness Centrality	
	2018	2021
Australia	0	0.142857
Italy	0.4	1.392857

Japan	0	0.142857
Netherlands	0	1.25
Singapore	0	0
Germany	0.4	0.142857
India	0.4	1.392857
Dem. People's Rep. of Korea	0	0
USA	0.4	0.142857
China	0.4	0
Canada	0.4	1.392857
France	0.4	0
Indonesia	0.4	0
Russian Federation	0.4	0
United Kingdom	0.4	0

Table1.3. Betweenness centrality parameters, 2018 and 2021

CLOSENESS CENTRALITY

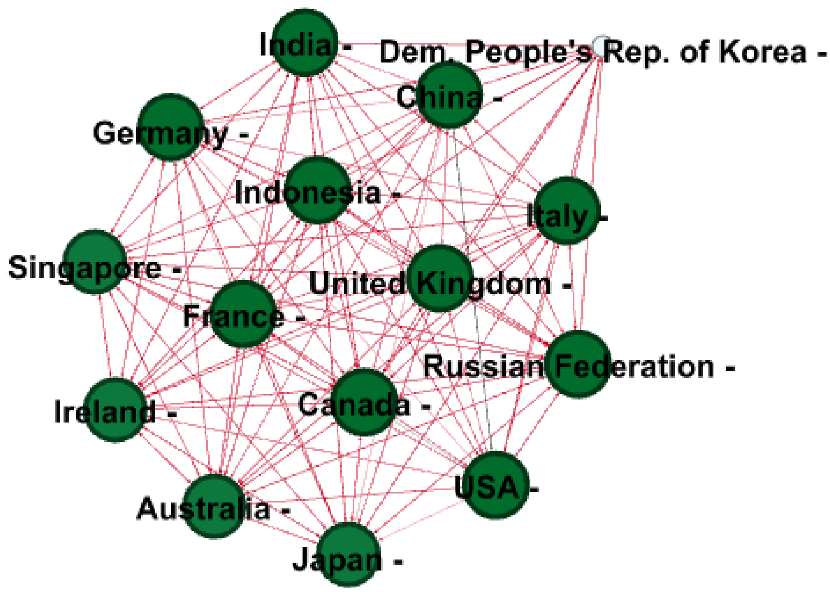


Figure 1.7 Closeness Centrality of trade data in 2018

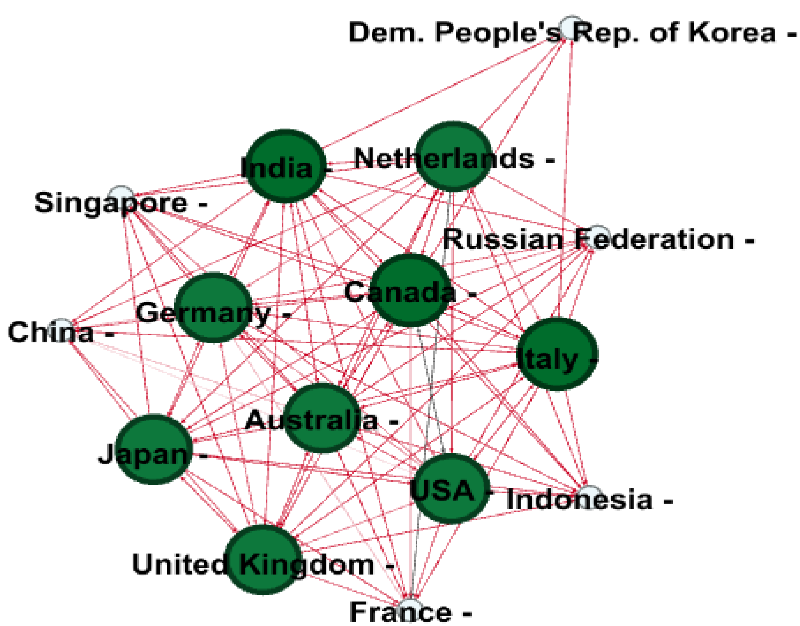
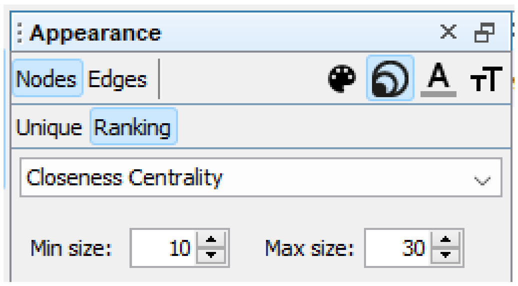
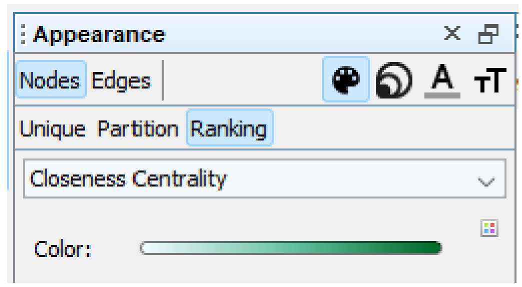


Figure 1.8 Closeness Centrality of trade data in 2021



Color



Size

Country	Closeness Centrality	
	2018	2021
Australia	0.933333	0.933333
Italy	1	1
Japan	0.933333	0.933333
Netherlands	0.933333	0.933333
Singapore	0.933333	0
Germany	1	0.933333
India	1	1
Dem. People's Rep. of Korea	0	0
USA	0.933333	0.933333
China	1	0
Canada	1	1

France	1	0
Indonesia	1	0
Russian Federation	1	0
United Kingdom	1	0.933333

Table 1.3. Closeness centrality parameters, 2018 and 2021

ECCENTRICITY

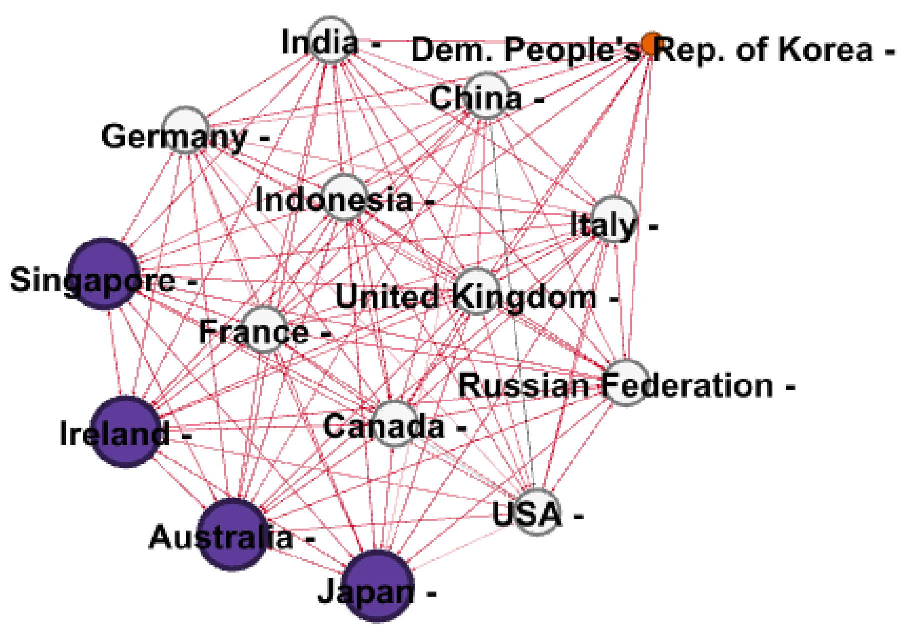


Figure 1.9. Eccentricity of trade data in 2018

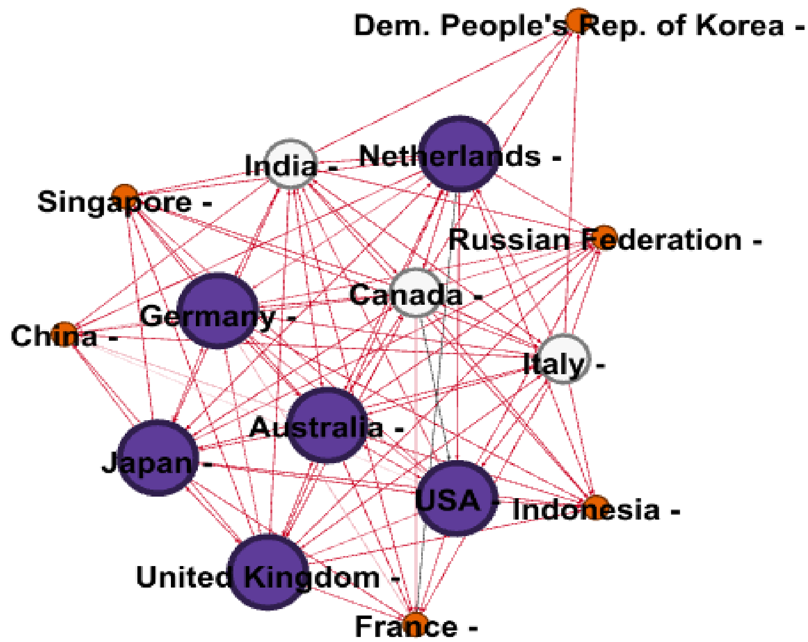
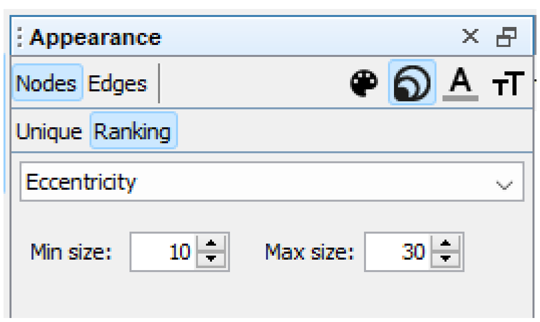
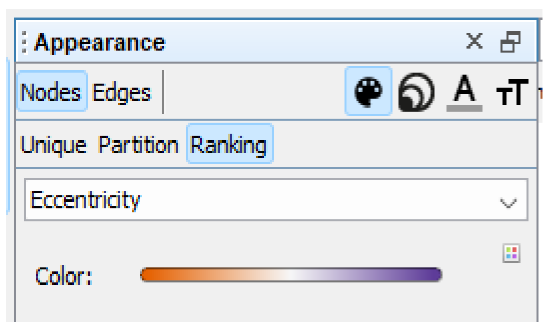


Figure 1.9.1 Eccentricity of trade data in 2021



Color



Size

Country	Eccentricity	
	2018	2021
Australia	2	2

Italy	1	1
Japan	2	2
Netherlands	2	2
Singapore	2	0
Germany	1	2
India	1	1
Dem. People's Rep. of Korea	0	0
USA	1	2
China	1	0
Canada	1	1
France	1	0
Indonesia	1	0
Russian Federation	1	0
United Kingdom	1	2

Table 1.4. Eccentricity parameters, 2018 and 2021

PAGE RANK

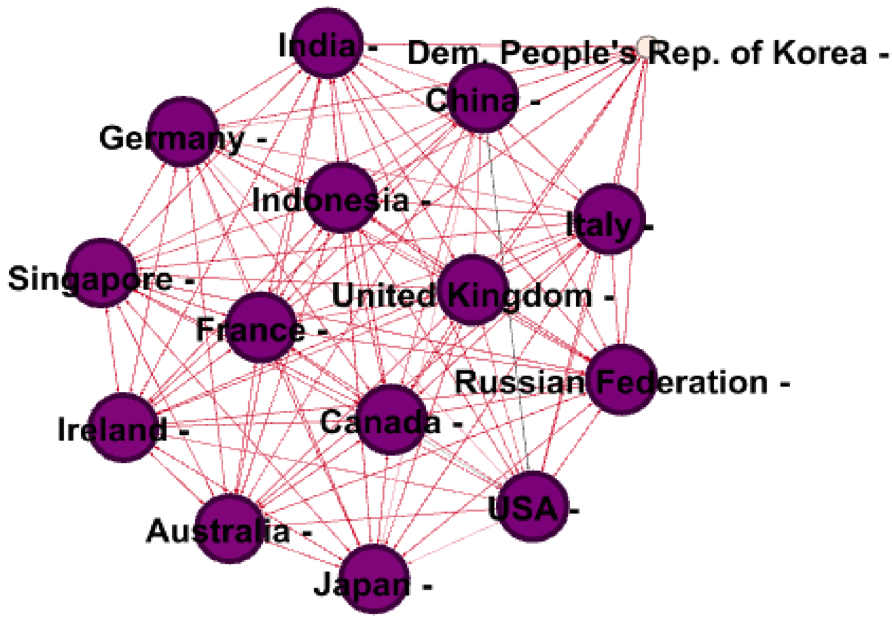


Figure 1.11. Page rank on trade data in 2018

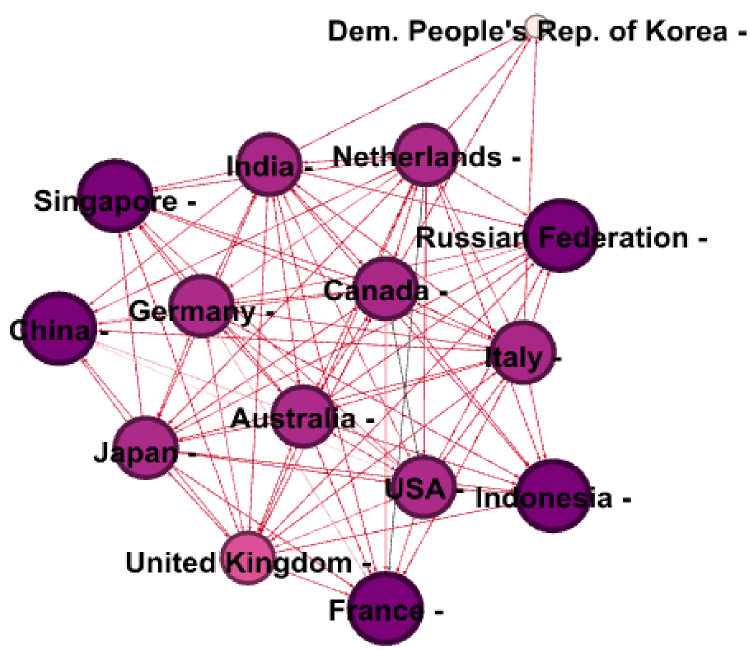
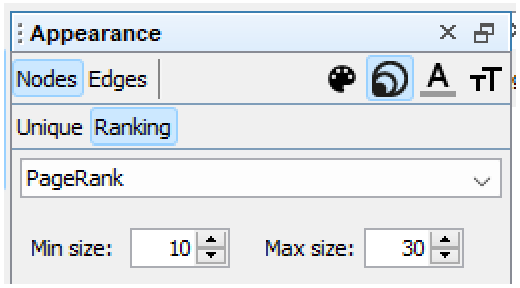
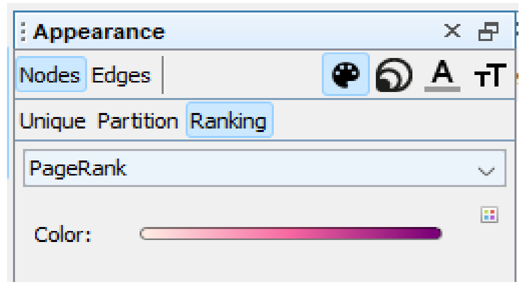


Figure 1.12. Page rank on trade data in 2021



Color



Size

Country	PageRank	
	2018	2021
Australia	0.06735	0.066573
Italy	0.067646	0.066866
Japan	0.06735	0.066573
Netherlands	0.06735	0.066573
Singapore	0.06735	0.070925
Germany	0.067646	0.066573
India	0.067646	0.066866
Dem. People's Rep. of Korea	0.054143	0.049428
USA	0.067646	0.066573
China	0.067646	0.070925

Canada	0.067646	0.066866
France	0.067646	0.070925
Indonesia	0.067646	0.070925
Russian Federation	0.067646	0.070925
United Kingdom	0.067646	0.062486

Table 1.5. PageRank parameters, 2018 and 2021

MODULARITY

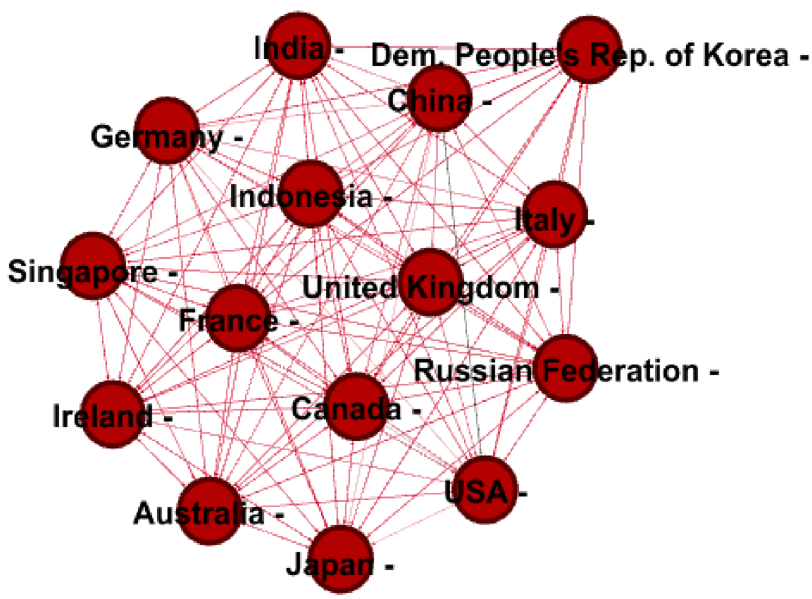


Figure 1.13. Modularity of trade data in 2018

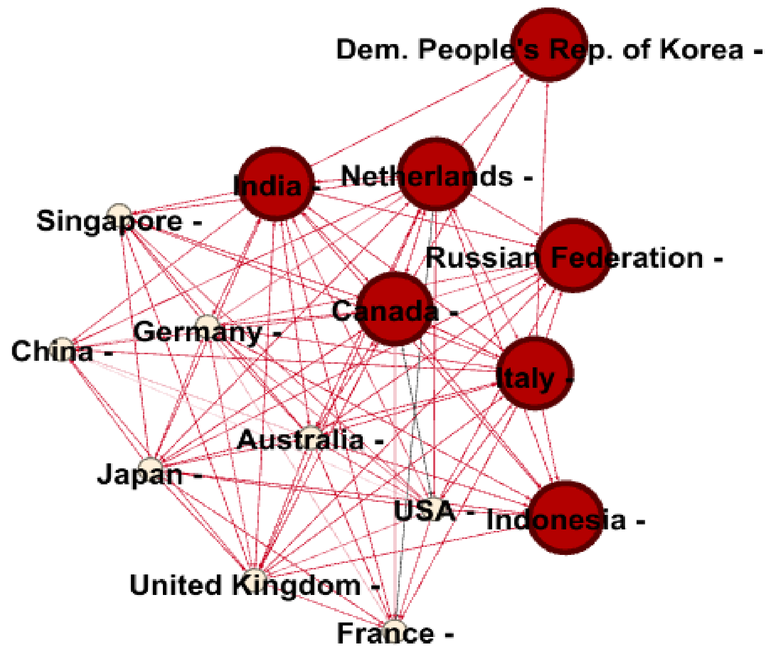
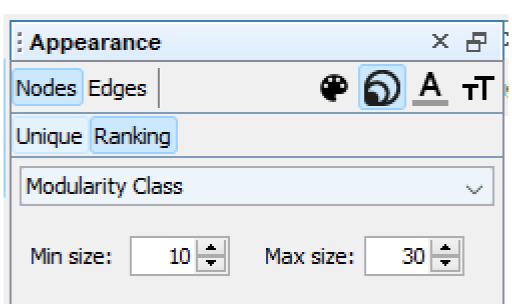
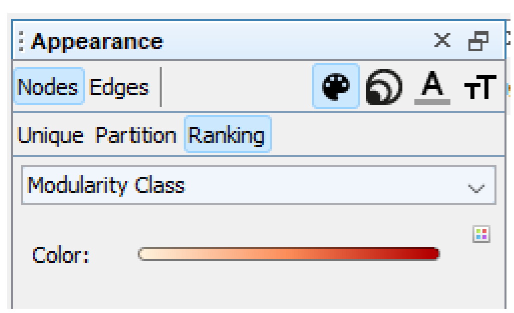


Figure 1.14. Modularity of trade data in 2021



Color



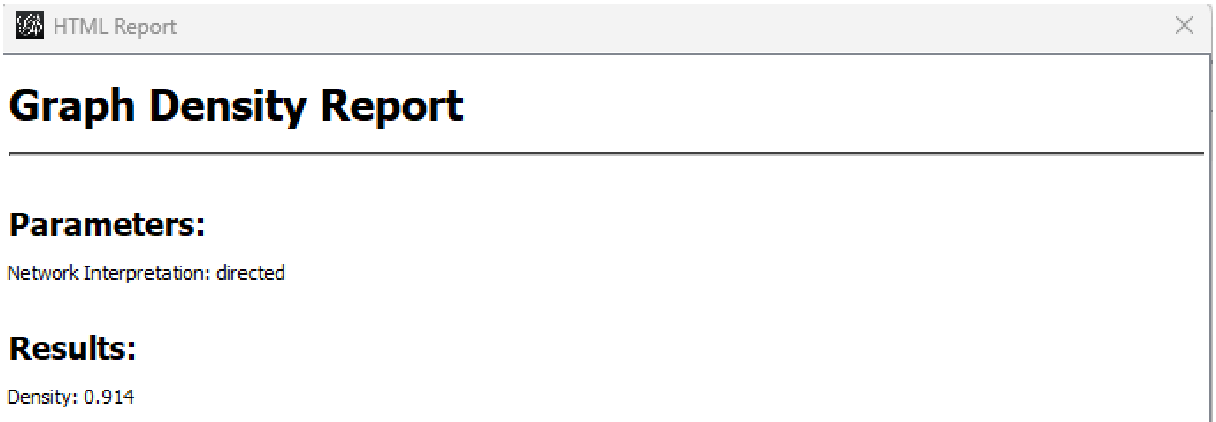
Size

Country	Modularity	
	2018	2021
Australia	0	0
Russia	0	1

France	0	0
India	0	1
USA	0	0
Canada	0	0
China	0	1
Germany	0	1
Japan	0	0
South Korea	0	0
United Kingdom	0	1
Indonesia	0	0
Ireland	0	1
Italy	0	1
Singapore	0	0

Table 1.6. Modularity Parameters, 2018 and 2021.

GRAPH DENSITY



HTML Report

Graph Density Report

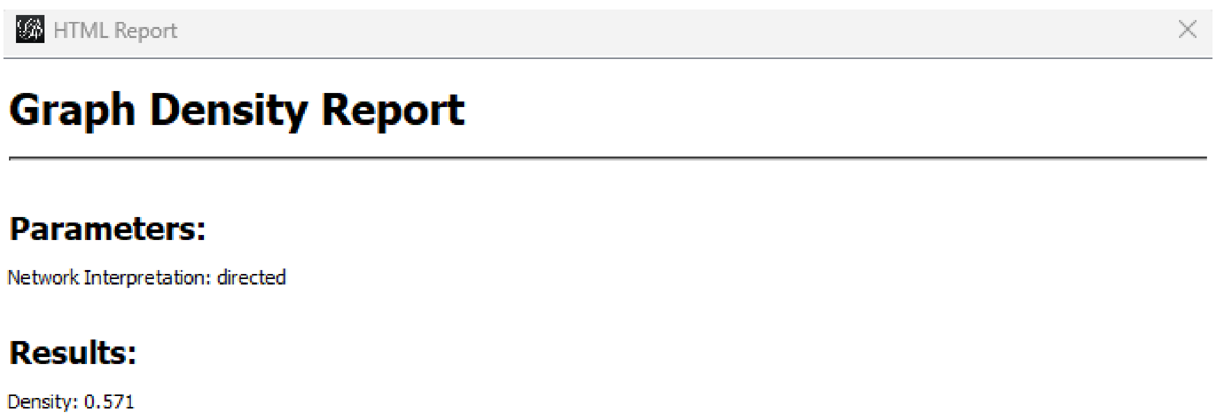
Parameters:

Network Interpretation: directed

Results:

Density: 0.914

Figure 1.15. Graph Density of trade data in 2018



HTML Report

Graph Density Report

Parameters:

Network Interpretation: directed

Results:

Density: 0.571

Figure 1.16. Graph Density of trade data in 2021

ANALYSIS:

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

```
data = pd.read_csv("2020 2021(full).csv")
data.head()
print(data)
data.dropna()
row = {'Reporter', 'Trade Value (US$)'}

data.plot.bar(x='Reporter', y='Trade Value (US$)')
```

```
<Axes: xlabel='Reporter'>
```

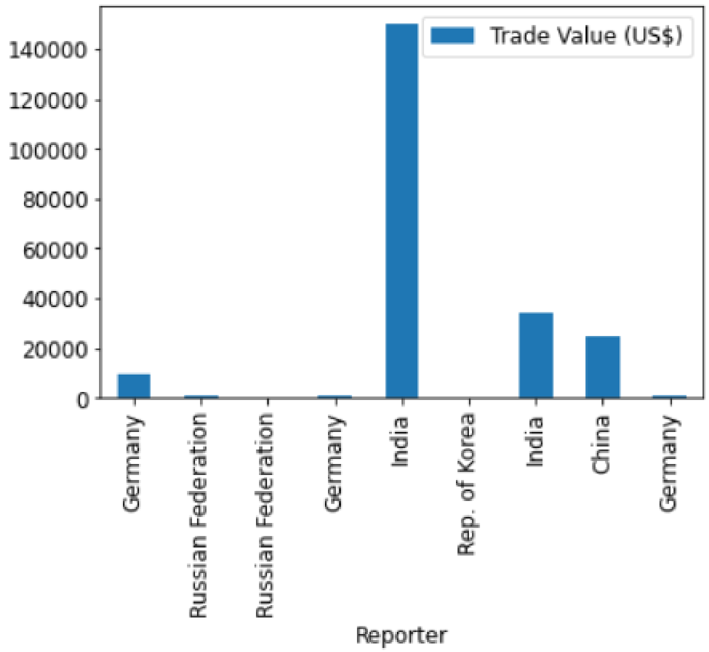


Fig2.1: Analysing the trade country wise

```
df = data["Trade Value (US$)"]
labels = data["Reporter"]
plt.figure(figsize=(7,7))
plt.pie(df, labels=labels, autopct='%1.1f%%', startangle=90, pctdistance=0.85, shadow =True)
central_circle = plt.Circle((0, 0), 0.5, color='white')
fig = plt.gcf()
fig.gca().add_artist(central_circle)
plt.rc('font', size=12)
plt.title("Distribution of the Trade", fontsize=20)
plt.show()
```

```
df = data["Trade Value (US$)"]
labels = data["Reporter"]
plt.figure(figsize=(7,7))
plt.pie(df, labels=labels, autopct='%1.1f%%', startangle=90, pctdistance=0.85, shadow =True)
central_circle = plt.Circle((0, 0), 0.5, color='white')
fig = plt.gcf()
fig.gca().add_artist(central_circle)
plt.rc('font', size=12)
plt.title("Distribution of the Trade", fontsize=20)
plt.show()
```

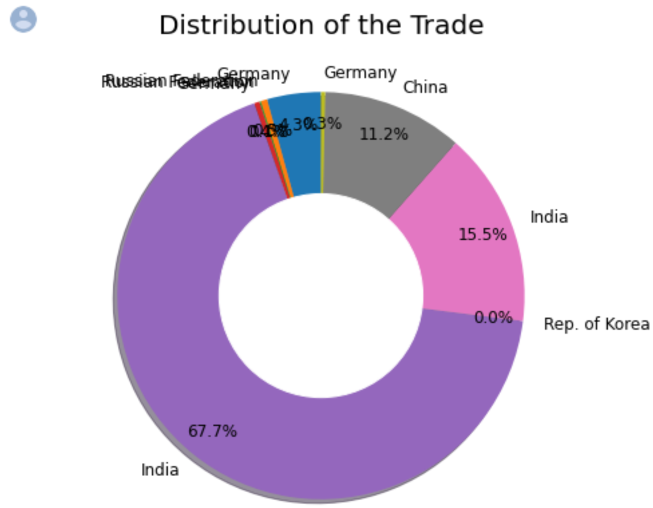


Fig2.2: Distribution of Trade

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# data analysis and wrangling
import pandas as pd
import numpy as np
import random as rnd

# visualization
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

#Data acquisition of the trade dataset
df_trade=pd.read_csv('trade dataset.csv')

df_trade.columns =['Year','Trade Flow Code','Trade Flow','Reporter Code','Reporter','Reporter ISO','Partner Code','Partner','Partner ISO','Commodity Code','Commodity','Qty Unit Code','Qty Unit','Qty','Netweight(kg)','Trade Value (US$)','Flag']
df_trade.columns
df_trade.dropna(inplace=True)
df_trade.head()

```
# data analysis and wrangling
import pandas as pd
import numpy as np
import random as rnd

# visualization
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

```
#Data acquisition of the trade  dataset
df_trade=pd.read_csv('trade dataset.csv')
df_trade.columns =['Year','Trade Flow Code','Trade Flow','Reporter Code','Reporter','Reporter ISO','Partner Code','Partner','Partner ISO','Commodity Code','Commodity','Qty Unit Code','Qty Unit','Qty','Netweight(kg)','Trade Value (US$
df_trade.columns
df_trade.dropna(inplace=True)
df_trade.head()
```

	Year	Trade Flow Code	Trade Flow	Reporter Code	Reporter	Reporter ISO	Partner Code	Partner	Partner ISO	Commodity Code	Commodity	Qty Unit Code	Qty Unit	Qty	Netweight(kg)	Trade Value (US$)	Flag
0	2019	1	Import	276	Germany	DEU	36	Australia	AUS	70200	Vegetables; tomatoes, fresh or chilled	8	Weight in kilograms	3372	3372.0	4391	0
1	2019	1	Import	276	Germany	DEU	76	Brazil	BRA	70200	Vegetables; tomatoes, fresh or chilled	8	Weight in kilograms	46719	46719.0	27134	0
2	2019	1	Import	276	Germany	DEU	156	China	CHN	70200	Vegetables; tomatoes, fresh or chilled	8	Weight in kilograms	25199	25199.0	68762	0
3	2019	1	Import	643	Russian Federation	RUS	156	China	CHN	70200	Vegetables; tomatoes, fresh or chilled	8	Weight in kilograms	69386320	69386320.0	84116572	0
4	2019	2	Export	643	Russian Federation	RUS	699	India	IND	70200	Vegetables; tomatoes, fresh or chilled	8	Weight in kilograms	634	634.0	1592	0

Fig2.3: Trade dataset for 2019

df = df_trade[df_trade['Year']==2019][df_trade["Reporter"]=="Germany"]
df

```
df = df_trade[df_trade['Year']==2019][df_trade["Reporter"]=="Germany"]
df
```

```
<ipython-input-39-9b329331b575>:1: UserWarning: Boolean Series key will be reindexed to match DataFrame index.
  df = df_trade[df_trade['Year']==2019][df_trade["Reporter"]=="Germany"]
```

	Year	Trade Flow Code	Trade Flow	Reporter Code	Reporter	Reporter ISO	Partner Code	Partner	Partner ISO	Commodity Code	Commodity	Qty Unit Code	Qty Unit	Qty	Netweight(kg)	Trade Value (US$)	Flag
0	2019	1	Import	276	Germany	DEU	36	Australia	AUS	70200	Vegetables; tomatoes, fresh or chilled	8	Weight in kilograms	3372	3372.0	4391	0
1	2019	1	Import	276	Germany	DEU	76	Brazil	BRA	70200	Vegetables; tomatoes, fresh or chilled	8	Weight in kilograms	46719	46719.0	27134	0
2	2019	1	Import	276	Germany	DEU	156	China	CHN	70200	Vegetables; tomatoes, fresh or chilled	8	Weight in kilograms	25199	25199.0	68762	0
20	2019	1	Import	276	Germany	DEU	36	Australia	AUS	80211	Nuts, edible; almonds, fresh or dried, in shell	8	Weight in kilograms	277	277.0	944	0
44	2019	1	Import	276	Germany	DEU	36	Australia	AUS	80910	Fruit, edible; apricots, fresh	8	Weight in kilograms	1625	1625.0	14436	0
45	2019	1	Import	276	Germany	DEU	156	China	CHN	80910	Fruit, edible; apricots, fresh	8	Weight in kilograms	2465	2465.0	4278	0
63	2019	1	Import	276	Germany	DEU	36	Australia	AUS	910	Ginger, saffron, tumeric (curcuma), thyme, bay...	8	Weight in kilograms	9416	9416.0	140444	6
64	2019	2	Export	276	Germany	DEU	36	Australia	AUS	910	Ginger, saffron, tumeric (curcuma), thyme, bay...	8	Weight in kilograms	48840	48840.0	376809	6
65	2019	1	Import	276	Germany	DEU	76	Brazil	BRA	910	Ginger, saffron, tumeric (curcuma), thyme, bay...	8	Weight in kilograms	878898	878898.0	4886421	6

Fig2.4: Selecting the values for the country Germany

df = df_trade[df_trade['Year']==2019]
frequency = np.unique(df["Trade Flow"])
country = np.unique(df["Reporter"])

```
countryList = {}
for i in country:
  list = {}
  for j in frequency:
    list[j] = len(df[(df["Trade Flow"]==j) & (df["Reporter"]==i)])

  countryList[i] = list
countryList
dff = pd.DataFrame(countryList).transpose()
dff["Country"] = dff.index
dff
```

```
df = df_trade[df_trade['Year']==2019]
frequency = np.unique(df["Trade Flow"])
country = np.unique(df["Reporter"])
countryList = {}
for i in country:
  list = {}
  for j in frequency:
    list[j] = len(df[(df["Trade Flow"]==j) & (df["Reporter"]==i)])

  countryList[i] = list
countryList
dff = pd.DataFrame(countryList).transpose()
dff["Country"] = dff.index
dff
```

	Export	Import	Re-Import	Country
China	15	16	4	China
Germany	15	24	0	Germany
India	16	14	0	India
Rep. of Korea	14	18	0	Rep. of Korea
Russian Federation	11	14	0	Russian Federation

Fig2.5: Analysing the export and import values of the countries in year 2019

```
dff.plot(x="Country",
        kind='bar',
        stacked=False,
        title='Grouped Bar Graph with dataframe')
```

```
dff.plot(x="Country",
         kind='bar',
         stacked=False,
         title='Grouped Bar Graph with dataframe')
```

```
<Axes: title={'center': 'Grouped Bar Graph with dataframe'}, xlabel='Country'>
```

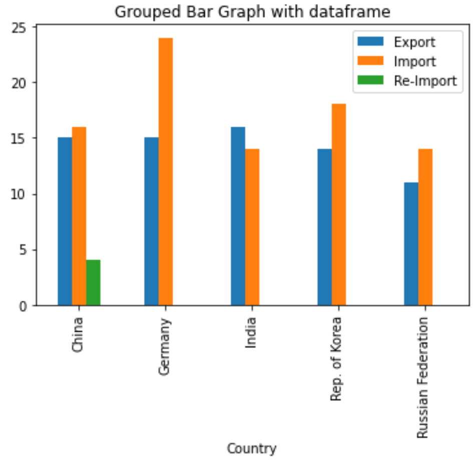


Fig2.6: Bar plot the export and import

```
df = df_trade[df_trade['Year']==2021]
frequency = np.unique(df["Trade Flow"])
country = np.unique(df["Reporter"])
countryList = {}
for i in country:
  list = {}
  for j in frequency:
    list[j] = len(df[(df["Trade Flow"]==j) & (df["Reporter"]==i)])

  countryList[i] = list
countryList
dff = pd.DataFrame(countryList).transpose()
dff["Country"] = dff.index
dff
```

```python
df = df_trade[df_trade['Year']==2021]
frequency = np.unique(df["Trade Flow"])
country = np.unique(df["Reporter"])
countryList = {}
for i in country:
  list = {}
  for j in frequency:
    list[j] = len(df[(df["Trade Flow"]==j) & (df["Reporter"]==i)])

  countryList[i] = list
countryList
dff = pd.DataFrame(countryList).transpose()
dff["Country"] = dff.index
dff
```

	Export	Import	Re-Import	Country
China	13	16	4	China
Germany	16	21	0	Germany
India	15	12	0	India
Rep. of Korea	13	22	0	Rep. of Korea
Russian Federation	13	16	0	Russian Federation

Fig2.7: Analysing the export and import values of the countries in year 2021

```python
dff.plot(x="Country",
    kind='bar',
    stacked=False,
    title='Grouped Bar Graph with dataframe')
```

```python
dff.plot(x="Country",
        kind='bar',
        stacked=False,
        title='Grouped Bar Graph with dataframe')
```

```
<Axes: title={'center': 'Grouped Bar Graph with dataframe'}, xlabel='Country'>
```

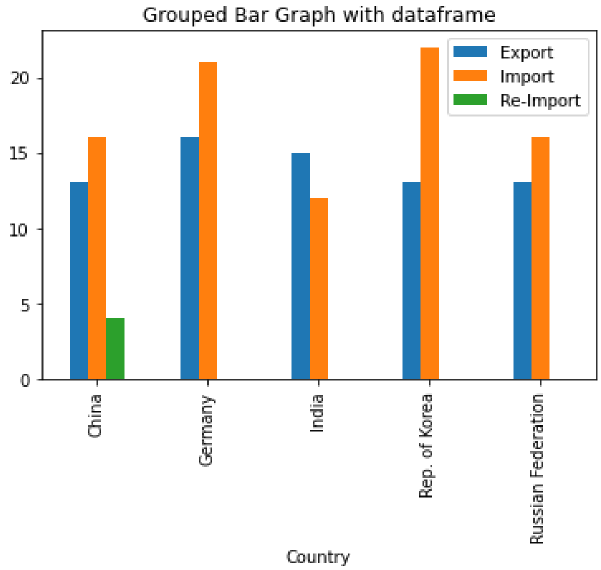


Fig 2.8: Bar plot the export and import

df = df_trade[df_trade['Year']==2019][df_trade["Trade Flow"]=="Import"][df_trade["Reporter"]=="India"][df_trade["Commodity"]=="Rice"]
df

df = df_trade[df_trade['Year']==2021][df_trade["Trade Flow"]=="Import"][df_trade["Reporter"]=="India"][df_trade["Commodity"]=="Rice"]
df

```python
df = df_trade[df_trade['Year']==2019][df_trade["Trade Flow"]=="Import"][df_trade["Reporter"]=="India"][df_trade["Commodity"]=="Rice"]
df
```

```
<ipython-input-44-ee31968a9434>:1: UserWarning: Boolean Series key will be reindexed to match DataFrame index.
  df = df_trade[df_trade['Year']==2019][df_trade["Trade Flow"]=="Import"][df_trade["Reporter"]=="India"][df_trade["Commodity"]=="Rice"]
```

	Year	Trade Flow Code	Trade Flow	Reporter Code	Reporter	Reporter ISO	Partner Code	Partner	Partner ISO	Commodity Code	Commodity	Qty Unit Code	Qty Unit	Qty	Netweight(kg)	Trade Value (US$)	Flag
177	2019	1	Import	699	India	IND	36	Australia	AUS	1006	Rice	8	Weight in kilograms	148000	148000.0	48149	0
179	2019	1	Import	699	India	IND	76	Brazil	BRA	1006	Rice	8	Weight in kilograms	4	4.0	237	0
181	2019	1	Import	699	India	IND	156	China	CHN	1006	Rice	8	Weight in kilograms	2	2.0	57	0

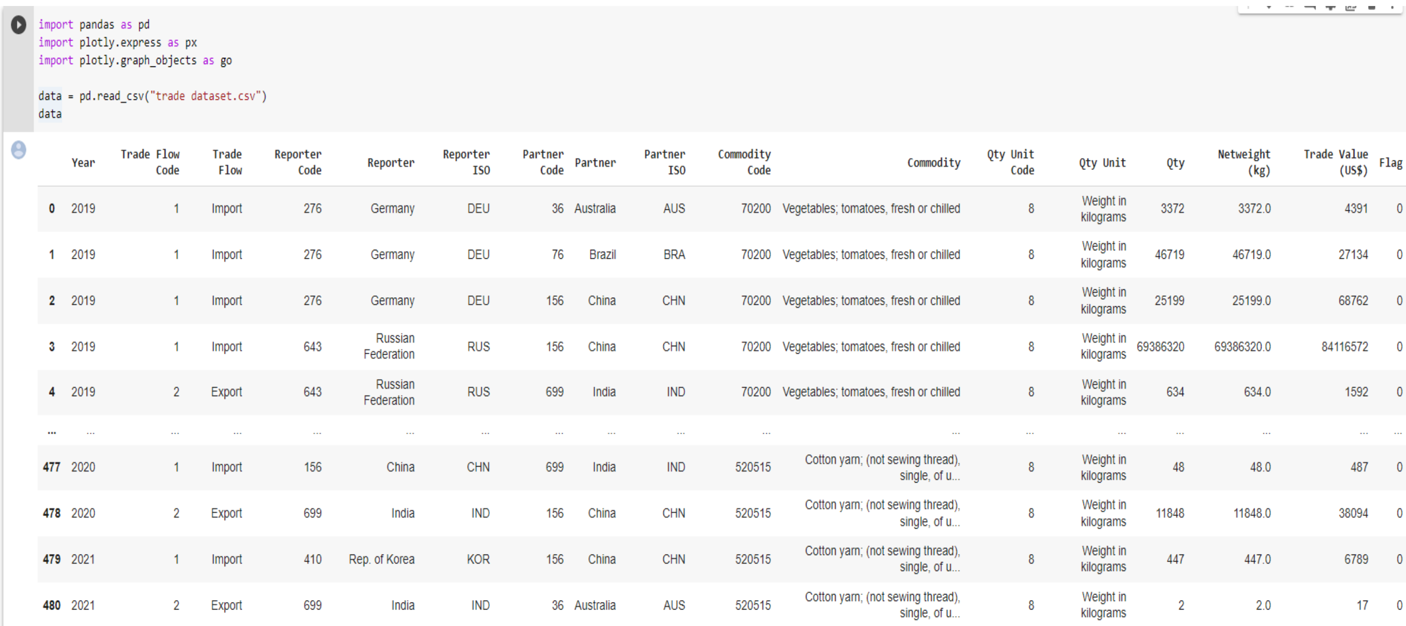
```python
df = df_trade[df_trade['Year']==2021][df_trade["Trade Flow"]=="Import"][df_trade["Reporter"]=="India"][df_trade["Commodity"]=="Rice"]
df
```

```
<ipython-input-45-0fe4e461c613>:1: UserWarning: Boolean Series key will be reindexed to match DataFrame index.
  df = df_trade[df_trade['Year']==2021][df_trade["Trade Flow"]=="Import"][df_trade["Reporter"]=="India"][df_trade["Commodity"]=="Rice"]
```

	Year	Trade Flow Code	Trade Flow	Reporter Code	Reporter	Reporter ISO	Partner Code	Partner	Partner ISO	Commodity Code	Commodity	Qty Unit Code	Qty Unit	Qty	Netweight(kg)	Trade Value (US$)	Flag
227	2021	1	Import	699	India	IND	36	Australia	AUS	1006	Rice	8	Weight in kilograms	63800	63800.0	35199	0
229	2021	1	Import	699	India	IND	76	Brazil	BRA	1006	Rice	8	Weight in kilograms	7	7.0	345	0

Sentimental Analysis

```
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go

data = pd.read_csv("trade dataset.csv")
data
```



```
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go

data = pd.read_csv("trade dataset.csv")
data
```

	Year	Trade Flow Code	Trade Flow	Reporter Code	Reporter	Reporter ISO	Partner Code	Partner	Partner ISO	Commodity Code	Commodity	Qty Unit Code	Qty Unit	Qty	Netweight (kg)	Trade Value (US$)	Flag
0	2019	1	Import	276	Germany	DEU	36	Australia	AUS	70200	Vegetables; tomatoes, fresh or chilled	8	Weight in kilograms	3372	3372.0	4391	0
1	2019	1	Import	276	Germany	DEU	76	Brazil	BRA	70200	Vegetables; tomatoes, fresh or chilled	8	Weight in kilograms	46719	46719.0	27134	0
2	2019	1	Import	276	Germany	DEU	156	China	CHN	70200	Vegetables; tomatoes, fresh or chilled	8	Weight in kilograms	25199	25199.0	68762	0
3	2019	1	Import	643	Russian Federation	RUS	156	China	CHN	70200	Vegetables; tomatoes, fresh or chilled	8	Weight in kilograms	69386320	69386320.0	84116572	0
4	2019	2	Export	643	Russian Federation	RUS	699	India	IND	70200	Vegetables; tomatoes, fresh or chilled	8	Weight in kilograms	634	634.0	1592	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
477	2020	1	Import	156	China	CHN	699	India	IND	520515	Cotton yarn; (not sewing thread), single, of u...	8	Weight in kilograms	48	48.0	487	0
478	2020	2	Export	699	India	IND	156	China	CHN	520515	Cotton yarn; (not sewing thread), single, of u...	8	Weight in kilograms	11848	11848.0	38094	0
479	2021	1	Import	410	Rep. of Korea	KOR	156	China	CHN	520515	Cotton yarn; (not sewing thread), single, of u...	8	Weight in kilograms	447	447.0	6789	0
480	2021	2	Export	699	India	IND	36	Australia	AUS	520515	Cotton yarn; (not sewing thread), single, of u...	8	Weight in kilograms	2	2.0	17	0

Fig 3.1: importing the dataset

```
data["Reporter"].value_counts()
data["Partner"].value_counts()
data.columns
```

```
[ ] data["Reporter"].value_counts()

Germany               115
China                 104
Rep. of Korea          98
India                  85
Russian Federation     80
Name: Reporter, dtype: int64
```

```
[ ] data["Partner"].value_counts()

China        153
Australia    143
India        104
Brazil        82
Name: Partner, dtype: int64
```

```
data.columns

Index(['Year', 'Trade Flow Code', 'Trade Flow', 'Reporter Code', 'Reporter',
       'Reporter ISO', 'Partner Code', 'Partner', 'Partner ISO',
       'Commodity Code', 'Commodity', 'Qty Unit Code', 'Qty Unit', 'Qty',
       'Netweight (kg)', 'Trade Value (US$)', 'Flag'],
      dtype='object')
```

Fig 3.2: Counting the values

```
code = data["Reporter ISO"].unique().tolist()
country = data["Reporter ISO"].unique().tolist()
tnw = []
ttv = []

for i in country:
   tnw.append((data.loc[data["Reporter ISO"] == i, "Trade Value (US$)"]).sum())
   ttv.append((data.loc[data["Reporter ISO"] == i, "Netweight (kg)"]).sum())

new_data = pd.DataFrame(list(zip(code, country, tnw,ttv,)),
                        columns = ["Code", "Country",
                                   "Total trade value", "Total NetWeight",
                                   ])
```

```
data1 = new_data.sort_values(by=["Total trade value"], ascending=False)
data1.head()
```

```
code = data["Reporter ISO"].unique().tolist()
country = data["Reporter ISO"].unique().tolist()
tnw = []
ttv = []

for i in country:
    tnw.append((data.loc[data["Reporter ISO"] == i, "Trade Value (US$)"]).sum())
    ttv.append((data.loc[data["Reporter ISO"] == i, "Netweight (kg)"]).sum())

new_data = pd.DataFrame(list(zip(code, country, tnw,ttv,)),
                               columns = ["Code", "Country",
                                          "Total trade value", "Total NetWeight",
                                          ])
```

```
data1 = new_data.sort_values(by=["Total trade value"], ascending=False)
data1.head()
```

	Code	Country	Total trade value	Total NetWeight
3	CHN	CHN	6451378540	1.216477e+09
4	KOR	KOR	1463560713	7.110245e+08
2	IND	IND	1333873562	1.643919e+09
0	DEU	DEU	733953090	2.102995e+08
1	RUS	RUS	584106529	6.181938e+08

Fig 3.3: Sorting the values along with trade value and total netweight

```
data1 = data1.head(10)
print(data1)
figure = px.bar(data1, y='Total trade value', x='Country',
        title="Countries with Highest Trade Value")
figure.show()
```

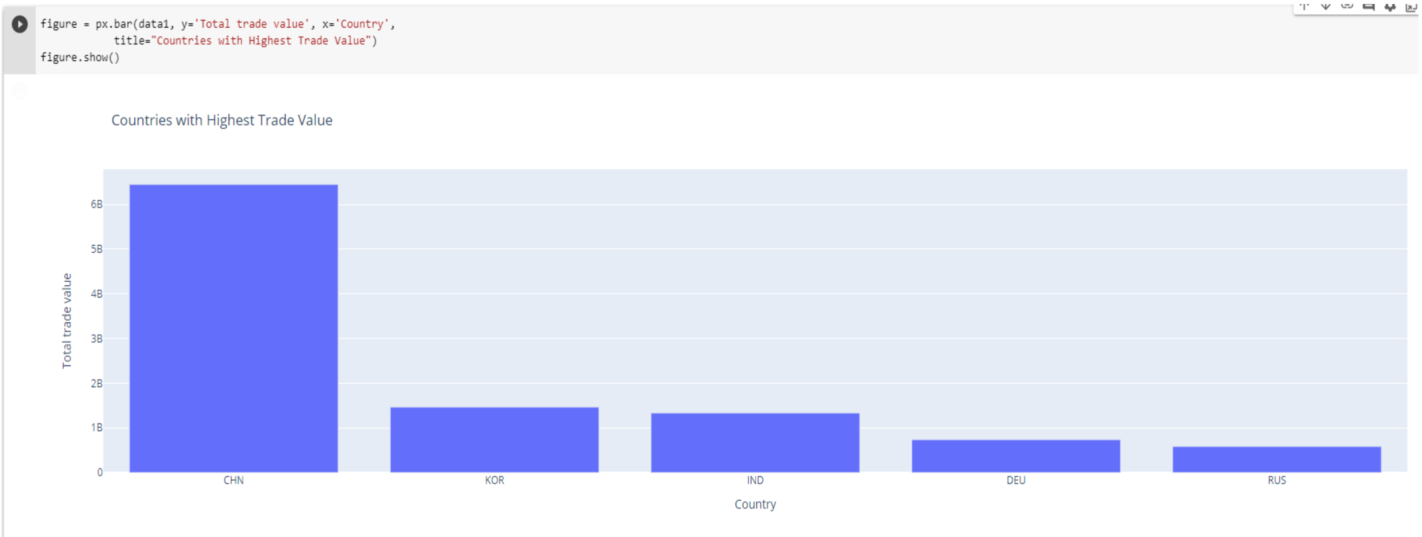


Fig3.4: Countires with high trade

```
figure = px.bar(data1, y=’Total NetWeight’, x=’Country’,
         title=”Countries with  highest netweight “)
figure.show()
```

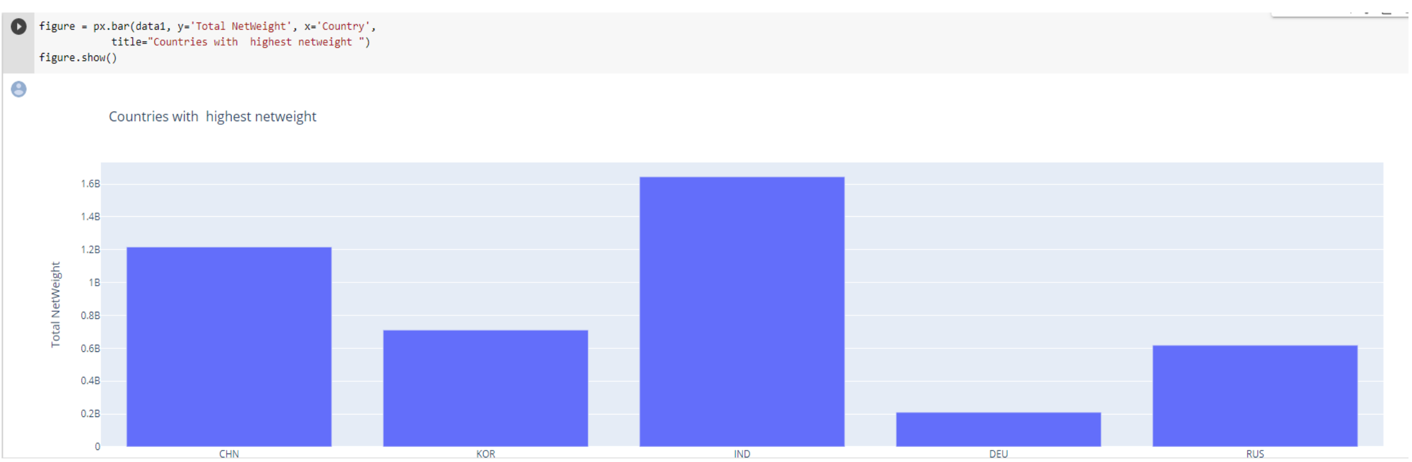


Fig 3.5: Countries with highest netweight

```
fig = go.Figure()
fig.add_trace(go.Bar(
   x=data1[“Country”],
   y=data1[“Total trade value”],
   name=’Total trade value’,
   marker_color=’indianred’
))
fig.add_trace(go.Bar(
   x=data1[“Country”],
   y=data1[“Total NetWeight”],
```

```
    name='Total NetWeight',
    marker_color='lightsalmon'
))
fig.update_layout(barmode='group', xaxis_tickangle=-45)
fig.show()
```

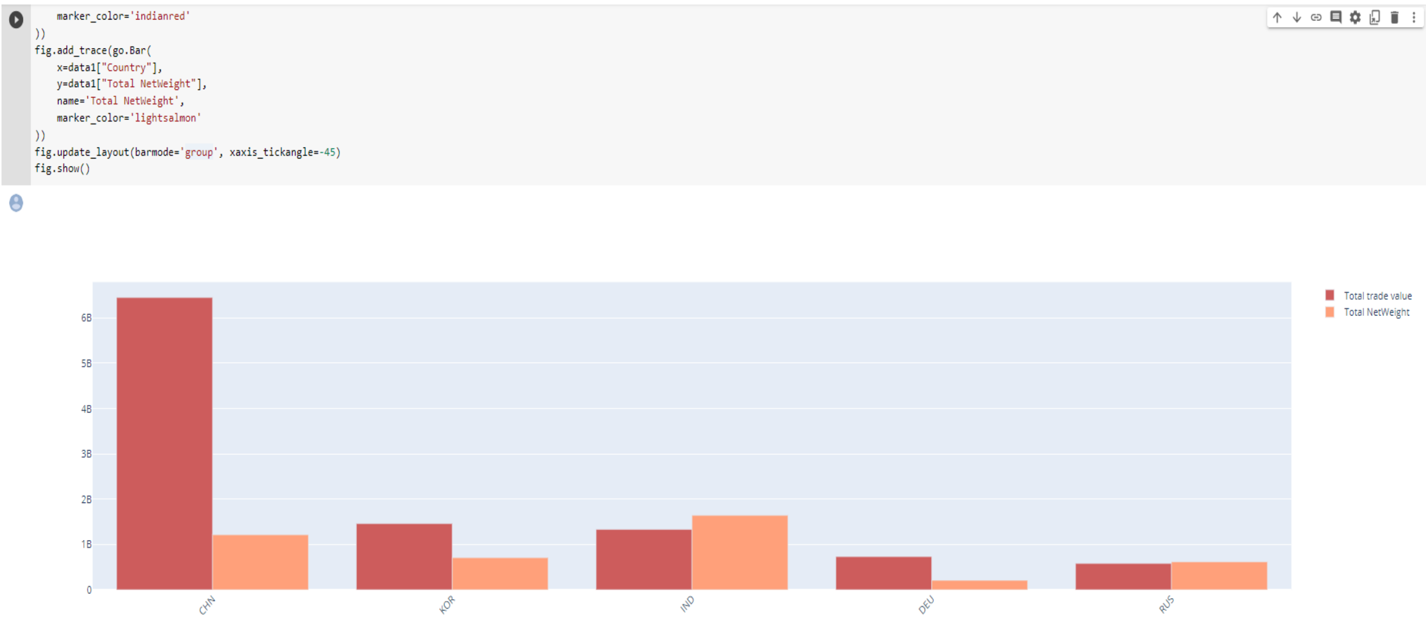


Fig3.6 :Barplot for trade value and netweight

```
ttv = data1["Total trade value"].sum()
tnw = data1["Total NetWeight"].sum()

labels = ["Total trade value ", "Total NetWeight "]
values = [ttv,tnw]

fig = px.pie(data, values=values, names=labels,
             title='Percentage of Total trade value and netweight', hole=0.5)
fig.show()
```

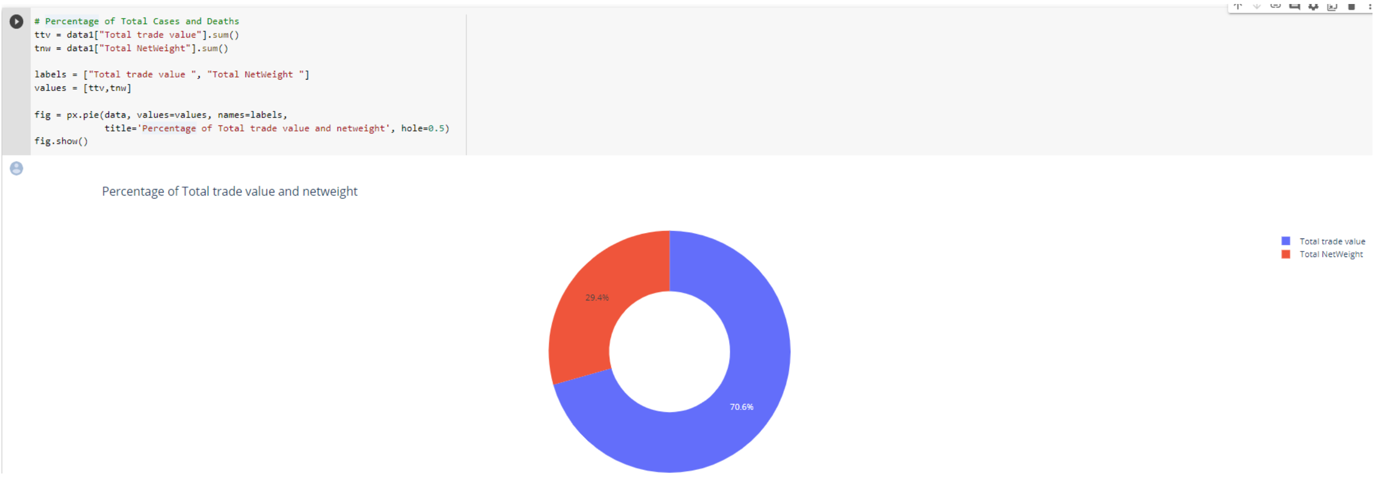


Fig3.7 :Percentage total trade and netweight

REGRESSION:

```
# Need for converting country name
!pip install -q pycountry
!pip install -q pycountry-convert
!pip install -q jellyfish
```

```
import numpy as np
import pandas as pd

import plotly.graph_objects as go
from plotly.subplots import make_subplots

import pycountry
import pycountry_convert as pc

from difflib import SequenceMatcher
import jellyfish

import re

TEMPLATE = 'simple_white'
WIDTH = 1200
```

```
INDIA_ORANGE = "rgb(255, 154, 47)"
INDIA_GREEN = "rgb(10, 137, 1)"
INDIA_BLUE = "rgb(0, 0, 137)"

COLOR_DICT = {
    'North America':INDIA_ORANGE,
    'Asia':INDIA_GREEN,
    'Oceania':INDIA_BLUE,
    'Europe':'GoldenRod',
    'Africa':'LightSeaGreen',
    'South America':'PaleVioletRed'
}
```

```
india_imports_df = pd.read_csv("/content/India_exports_FY22.csv", thousands=',')
india_exports_df = pd.read_csv("/content/India_imports_FY22.csv", thousands=',')
```

```
print("##India imports data##")
display(india_imports_df.head(2))
india_imports_df.info()

print("\n##India exports data##")
display(india_exports_df.head(2))
india_exports_df.info()
```

```python
import numpy as np
import pandas as pd

import plotly.graph_objects as go
from plotly.subplots import make_subplots

import pycountry
import pycountry_convert as pc

from difflib import SequenceMatcher
import jellyfish

import re

TEMPLATE = 'simple_white'
WIDTH = 1200

INDIA_ORANGE = "rgb(255, 154, 47)"
INDIA_GREEN = "rgb(10, 137, 1)"
INDIA_BLUE = "rgb(0, 0, 137)"

COLOR_DICT = {
    'North America':INDIA_ORANGE,
    'Asia':INDIA_GREEN,
    'Oceania':INDIA_BLUE,
    'Europe':'GoldenRod',
    'Africa':'LightSeaGreen',
    'South America':'PaleVioletRed'
}
```

Fig:

```
india_imports_df = pd.read_csv("/content/India_exports_FY22.csv", thousands=',')
india_exports_df = pd.read_csv("/content/India_imports_FY22.csv", thousands=',')
```

```
print("##India imports data##")
display(india_imports_df.head(2))
india_imports_df.info()

print("\n##India exports data##")
display(india_exports_df.head(2))
india_exports_df.info()
```

##India imports data##

	S.No.	Country	2020-2021	%Share	2021-2022	%Share1	%Growth
0	1	AFGHANISTAN	825.78	0.283	554.47	0.1314	-32.85
1	2	ALBANIA	46.75	0.016	55.16	0.0131	17.99

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 235 entries, 0 to 234
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   S.No.      235 non-null    int64
 1   Country    235 non-null    object
 2   2020-2021  232 non-null    float64
 3   %Share     235 non-null    object
 4   2021-2022  230 non-null    float64
 5   %Share1    235 non-null    object
 6   %Growth    235 non-null    object
dtypes: float64(2), int64(1), object(4)
memory usage: 13.0+ KB
```

##India exports data##

	S.No.	Country	2020-2021	%Share	2021-2022	%Share1	%Growth
0	1	AFGHANISTAN	509.49	0.1292	510.93	0.0834	0.28
1	2	ALBANIA	8.30	0.0021	30.71	0.005	269.78

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 227 entries, 0 to 226
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   S.No.      227 non-null    int64
 1   Country    227 non-null    object
 2   2020-2021  225 non-null    float64
 3   %Share     227 non-null    object
 4   2021-2022  216 non-null    float64
 5   %Share1    227 non-null    object
 6   %Growth    227 non-null    object
dtypes: float64(2), int64(1), object(4)
memory usage: 12.5+ KB
```

Fig4.2: Imports and Exports

```
def prep_df(df):
   num_cols = ['2020-2021', '%Share', '2021-2022', '%Share1', '%Growth']

   df.columns = [c.strip() for c in df.columns] # remove space
   df[num_cols] = df[num_cols].apply(pd.to_numeric, errors='coerce') # convert dtype
   df = df.fillna(0) # fillna
   return df

india_imports_df = prep_df(india_imports_df)
india_exports_df = prep_df(india_exports_df)
print("##India imports data##")
display(india_imports_df.head(10))
india_imports_df.info()
```

print(“##India exports data##”)
display(india_exports_df.head(10))
india_exports_df.info()

```
def prep_df(df):
    num_cols = ['2020-2021', '%Share', '2021-2022', '%Share1', '%Growth']

    df.columns = [c.strip() for c in df.columns] # remove space
    df[num_cols] = df[num_cols].apply(pd.to_numeric, errors='coerce') # convert dtype
    df = df.fillna(0) # fillna
    return df

india_imports_df = prep_df(india_imports_df)
india_exports_df = prep_df(india_exports_df)
print("##India imports data##")
display(india_imports_df.head(10))
india_imports_df.info()

print("##India exports data##")
display(india_exports_df.head(10))
india_exports_df.info()
```

##India imports data##

	S.No.	Country	2020-2021	%Share	2021-2022	%Share1	%Growth
0	1	AFGHANISTAN	825.78	0.2830	554.47	0.1314	-32.85
1	2	ALBANIA	46.75	0.0160	55.16	0.0131	17.99
2	3	ALGERIA	594.74	0.2038	703.25	0.1667	18.24
3	4	AMERI SAMOA	0.58	0.0002	0.78	0.0002	33.68
4	5	ANDORRA	0.02	0.0000	0.05	0.0000	141.87
5	6	ANGOLA	259.60	0.0890	452.45	0.1072	74.29
6	7	ANGUILLA	0.05	0.0000	0.09	0.0000	80.67
7	8	ANTARTICA	1.05	0.0004	0.07	0.0000	-92.98
8	9	ANTIGUA	1.62	0.0006	2.47	0.0006	53.17
9	10	ARGENTINA	687.84	0.2357	1425.94	0.3380	107.31

Fig4.3: Imports dataset

```
 5   %Share1     235 non-null    float64
 6   %Growth     235 non-null    float64
dtypes: float64(5), int64(1), object(1)
memory usage: 13.0+ KB
##India exports data##
```

	S.No.	Country	2020-2021	%Share	2021-2022	%Share1	%Growth
0	1	AFGHANISTAN	509.49	0.1292	510.93	0.0834	0.28
1	2	ALBANIA	8.30	0.0021	30.71	0.0050	269.78
2	3	ALGERIA	408.79	0.1036	1004.24	0.1639	145.66
3	4	AMERI SAMOA	0.96	0.0002	0.00	0.0000	-99.74
4	5	ANDORRA	0.01	0.0000	0.00	0.0000	0.00
5	6	ANGOLA	1879.74	0.4766	2725.08	0.4448	44.97
6	7	ANGUILLA	0.18	0.0000	0.03	0.0000	-83.84
7	8	ANTARTICA	0.67	0.0002	1.22	0.0002	82.37
8	9	ANTIGUA	0.14	0.0000	0.13	0.0000	-6.05
9	10	ARGENTINA	2627.05	0.6660	4201.74	0.6859	59.94

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 227 entries, 0 to 226
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   S.No.       227 non-null    int64
 1   Country     227 non-null    object
 2   2020-2021   227 non-null    float64
 3   %Share      227 non-null    float64
 4   2021-2022   227 non-null    float64
 5   %Share1     227 non-null    float64
 6   %Growth     227 non-null    float64
dtypes: float64(5), int64(1), object(1)
memory usage: 12.5+ KB
```

Fig4.4:Exports dataset

```
# Merge
temp1 = india_imports_df.drop(columns='S.No.')
temp1.columns = ['Country'] + [f"{c}_import" for c in temp1.columns if c != 'Country']
temp2 = india_exports_df.drop(columns='S.No.')
temp2.columns = ['Country'] + [f"{c}_export" for c in temp2.columns if c != 'Country']
india_df = pd.merge(temp1, temp2, on='Country')

# Add Total
for i, c in enumerate(['2020-2021', '2021-2022']):
   india_df[f"{c}_total"] = india_df[f"{c}_import"] + india_df[f"{c}_export"]
   n = '' if i==0 else '1'
   india_df[f"%Share{n}_total"] = 100 * india_df[f"{c}_total"] / india_df[f"{c}_total"].sum()

india_df[f"%Growth_total"] = 100*(india_df["2021-2022_total"] – india_df["2020-2021_total"]) / india_df["2020-2021_total"]
```

india_df.head()

```python
# Merge
temp1 = india_imports_df.drop(columns='S.No.')
temp1.columns = ['Country'] + [f"{c}_import" for c in temp1.columns if c != 'Country']
temp2 = india_exports_df.drop(columns='S.No.')
temp2.columns = ['Country'] + [f"{c}_export" for c in temp2.columns if c != 'Country']
india_df = pd.merge(temp1, temp2, on='Country')

# Add Total
for i, c in enumerate(['2020-2021', '2021-2022']):
    india_df[f"{c}_total"] = india_df[f"{c}_import"] + india_df[f"{c}_export"]
    n = '' if i==0 else '1'
    india_df[f"%Share{n}_total"] = 100 * india_df[f"{c}_total"] / india_df[f"{c}_total"].sum()

india_df[f"%Growth_total"] = 100*(india_df["2021-2022_total"] - india_df["2020-2021_total"]) / india_df["2020-2021_total"]

india_df.head()
```

	Country	2020-2021_import	%Share_import	2021-2022_import	%Share1_import	%Growth_import	2020-2021_export	%Share_export	2021-2022_export	%Share1_export	%Growth_export	2020-2021_total	%Share_total	2021-2022_total	%Share1_tot
0	AFGHANISTAN	825.78	0.2830	554.47	0.1314	-32.85	509.49	0.1292	510.93	0.0834	0.28	1335.27	0.194577	1065.40	0.1029
1	ALBANIA	46.75	0.0160	55.16	0.0131	17.99	8.30	0.0021	30.71	0.0050	269.78	55.05	0.008022	85.87	0.0083
2	ALGERIA	594.74	0.2038	703.25	0.1667	18.24	408.79	0.1036	1004.24	0.1639	145.66	1003.53	0.146235	1707.49	0.1650
3	AMERI SAMOA	0.58	0.0002	0.78	0.0002	33.68	0.96	0.0002	0.00	0.0000	-99.74	1.54	0.000224	0.78	0.0000
4	ANDORRA	0.02	0.0000	0.05	0.0000	141.87	0.01	0.0000	0.00	0.0000	0.00	0.03	0.000004	0.05	0.0000

Fig4.5: Merging of data

```
def convert_country_name(x):
    cname = x[:]

    try:
        # Search without all spaces in existing country names and add them to the candidate list.
        Cands = [c.name for c in pycountry.countries.search_fuzzy(cname.replace(“ “, “”))]
    except:
        cands = []

    try:
        # The abbreviations ‘U’ and ‘RP’ of the existing country names are changed to ‘United’ and ‘RP’, res
pectively.
        Cname = re.sub(‘[^a-zA-z]’, “ “, cname)
        for st, tt in zip([‘U’, ‘RP’, ‘IS’, ‘DP’], [‘United’, ‘Republic’, ‘Islands’, ‘Democratic People’]):
            cname = re.sub(f’{st}[\s]+’,f”{tt} “, cname)
            cname = re.sub(f’[\s]+{st}’, f” {tt}”,cname)

        # Then, for each token, add a group of candidates using pycountry’s fuzzy search.
        For sub in cname.split(“ “):
            if len(sub)<=2:
                continue
```

```
        try:
            cands += [c.name for c in pycountry.countries.search_fuzzy(sub)]
        except:
            continue

      # Using the string similarity of jellyfish, find the most similar country name among the candidate gro
ups.
      Cand_sim = [jellyfish.jaro_distance(cname.lower(), c.lower()) for c in cands]
      return cands[np.argmax(cand_sim)]
   except:
      print(f"Error : {x}")
      return np.nan
```

```
print("#### Sample Test ####")
for c in ['U ARAB EMTS ', 'U S A', 'CHINA P RP', 'KOREA RP', 'BOSNIA-
HRZGOVIN', 'N. MARIANA IS.', 'BAHARAIN IS', 'ANTARTICA']:
   print(f"{c} ➔ {convert_country_name©}")
```

```
print("#### Sample Test ####")
for c in ['U ARAB EMTS ', 'U S A', 'CHINA P RP', 'KOREA RP', 'BOSNIA-HRZGOVIN', 'N. MARIANA IS.', 'BAHARAIN IS', 'ANTARTICA']:
    print(f"{c} ==> {convert_country_name(c)}")
```

```
#### Sample Test ####
U ARAB EMTS  ==> United Arab Emirates
U S A ==> United States
CHINA P RP ==> China
KOREA RP ==> Korea, Republic of
BOSNIA-HRZGOVIN ==> Bosnia and Herzegovina
N. MARIANA IS. ==> Northern Mariana Islands
BAHARAIN IS ==> Bahamas
Error : ANTARTICA
ANTARTICA ==> nan
```

```
print("#### Convert All ####")
india_df['Name'] = india_df['Country'].apply(lambda x : convert_country_name(x))

error_country = {
   "ANTARTICA " : "Antarctica",
   "KYRGHYZSTAN " : "Kyrgyzstan",
   "NETHERLANDANTIL " : "Netherlands Antilles",
   "SWAZILAND " : "Eswatini",
```

```
    "UNSPECIFIED " : "UNSPECIFIED"
}

india_df['Name'] = india_df['Name'].fillna(india_df['Country'])
india_df['Name'] = india_df['Name'].apply(lambda x : error_country[x] if x in error_country else x)
```

```
print("#### Convert All ####")
india_df['Name'] = india_df['Country'].apply(lambda x : convert_country_name(x))

error_country = {
    "ANTARTICA " : "Antarctica",
    "KYRGHYZSTAN " : "Kyrgyzstan",
    "NETHERLANDANTIL " : "Netherlands Antilles",
    "SWAZILAND " : "Eswatini",
    "UNSPECIFIED " : "UNSPECIFIED"
}

india_df['Name'] = india_df['Name'].fillna(india_df['Country'])
india_df['Name'] = india_df['Name'].apply(lambda x : error_country[x] if x in error_country else x)
```

```
#### Convert All ####
Error : ANTARTICA
Error : KYRGHYZSTAN
Error : NETHERLANDANTIL
Error : SWAZILAND
Error : UNSPECIFIED
```

```
def get_alph_3_name(x):
    try:
        return pycountry.countries.lookup(x).alpha_3
    except:
        return 'ZZZ' # No data name

india_df['Name_3'] = india_df['Name'].apply(lambda x : get_alph_3_name(x))
india_df[['Country', 'Name', 'Name_3']].head(3)
```

```
def get_alph_3_name(x):
    try:
        return pycountry.countries.lookup(x).alpha_3
    except:
        return 'ZZZ' # No data name

india_df['Name_3'] = india_df['Name'].apply(lambda x : get_alph_3_name(x))
india_df[['Country', 'Name', 'Name_3']].head(3)
```

	Country	Name	Name_3
0	AFGHANISTAN	Afghanistan	AFG
1	ALBANIA	Albania	ALB
2	ALGERIA	Algeria	DZA

```
def get_continet_name(name_3):
    try:
        alpah2 = pc.country_alpha3_to_country_alpha2(name_3)
        continent_code = pc.country_alpha2_to_continent_code(alpah2)
        continent_name = pc.convert_continent_code_to_continent_name(continent_code)
    except:
        continent_name = ‘Others’
    return continent_name

india_df[‘Continent’] = india_df[‘Name_3’].apply(lambda x : get_continet_name(x))
india_df[[‘Country’, ‘Name’, ‘Name_3’, ‘Continent’]].head(3)
```

```
def get_continet_name(name_3):
    try:
        alpah2 = pc.country_alpha3_to_country_alpha2(name_3)
        continent_code = pc.country_alpha2_to_continent_code(alpah2)
        continent_name = pc.convert_continent_code_to_continent_name(continent_code)
    except:
        continent_name = 'Others'
    return continent_name

india_df['Continent'] = india_df['Name_3'].apply(lambda x : get_continet_name(x))
india_df[['Country', 'Name', 'Name_3', 'Continent']].head(3)
```

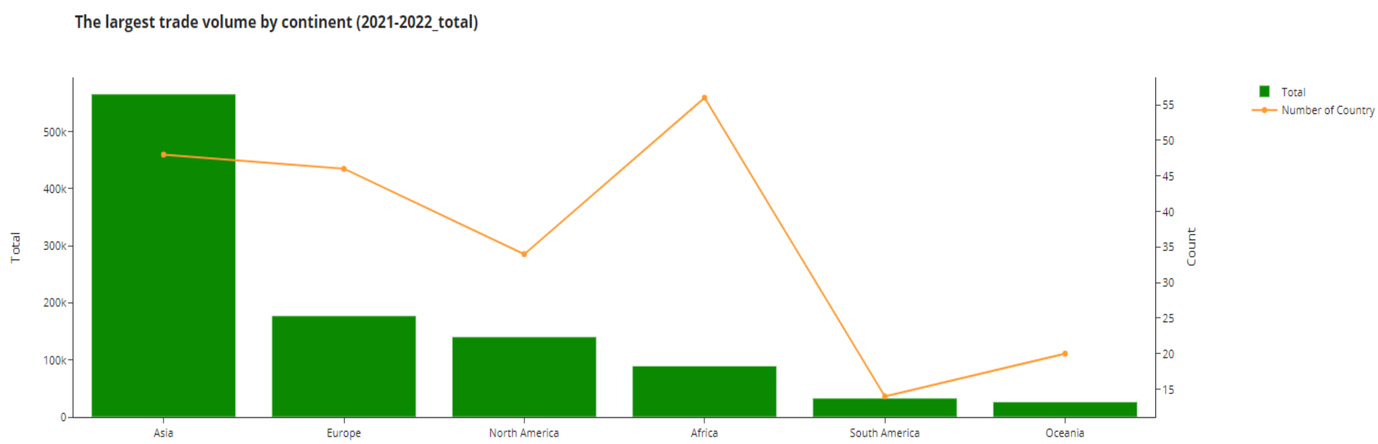
	Country	Name	Name_3	Continent
0	AFGHANISTAN	Afghanistan	AFG	Asia
1	ALBANIA	Albania	ALB	Europe
2	ALGERIA	Algeria	DZA	Africa

target = ‘2021-2022_total’
show_df = india_df[india_df[‘Continent’]!=’Others’].groupby([‘Continent’])[[target]].sum().sort_values(by=[target], ascending=False)
show_df[‘Cnt’] = india_df.groupby([‘Continent’])[[target]].size()

fig = make_subplots(specs=[[{“secondary_y”: True}]])
fig.add_trace(go.Bar(x=show_df.index, y=show_df[target], marker_color=INDIA_GREEN, name=’Total’))
fig.add_trace(go.Scatter(x=show_df.index, y=show_df[‘Cnt’], marker_color=INDIA_ORANGE, name=’Number of Country’), secondary_y=True)

fig.update_layout(title_text=f’The largest trade volume by continent ({target})’, template=TEMPLATE)
fig.update_yaxes(title=’Total’)
fig.update_yaxes(title=’Count’, secondary_y=True)
fig.show()

```
show_df['Cnt'] = india_df.groupby(['Continent'])[[target]].size()

fig = make_subplots(specs=[[{"secondary_y": True}]])
fig.add_trace(go.Bar(x=show_df.index, y=show_df[target], marker_color=INDIA_GREEN, name='Total'))
fig.add_trace(go.Scatter(x=show_df.index, y=show_df['Cnt'], marker_color=INDIA_ORANGE, name='Number of Country'), secondary_y=True)

fig.update_layout(title_text=f'<b>The largest trade volume by continent ({target})</b>', template=TEMPLATE)
fig.update_yaxes(title='Total')
fig.update_yaxes(title='Count', secondary_y=True)
fig.show()
```




```
temp_show_df = pd.DataFrame()
for cate in ['total', 'import', 'export']:
    c1 = f'2020-2021_{cate}'
    c2 = f'2021-2022_{cate}'
    temp_ = pd.DataFrame(india_df[[c1, c2]].sum()).T
    temp_[f'%Growth_{cate}'] = 100* (temp_[c2] – temp_[c1]) / temp_[c1]
    temp_.columns = ['2020-2021', '2021-2022', 'Growth_Rate(%)']
    temp_show_df = temp_show_df.append(temp_)

temp_show_df.index = ['total', 'import', 'export']
display(temp_show_df)

BASE_GR = {
    'total': 50.748297,
    'import': 55.312213,
    'export' : 44.579255
}
```

```
for cate in ['total', 'import', 'export']:
    c1 = f'2020-2021_{cate}'
    c2 = f'2021-2022_{cate}'
    temp_ = pd.DataFrame(india_df[[c1, c2]].sum()).T
    temp_[f'%Growth_{cate}'] = 100* (temp_[c2] - temp_[c1]) / temp_[c1]
    temp_.columns = ['2020-2021', '2021-2022', 'Growth_Rate(%)']
    temp_show_df = temp_show_df.append(temp_)

temp_show_df.index = ['total', 'import', 'export']
display(temp_show_df)

BASE_GR = {
    'total': 50.748297,
    'import': 55.312213,
    'export' : 44.579255
}
```

```
<ipython-input-28-89f533d27193>:8: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.

<ipython-input-28-89f533d27193>:8: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.

<ipython-input-28-89f533d27193>:8: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
```

	2020-2021	2021-2022	Growth_Rate(%)
total	686243.34	1034500.15	50.748297
import	291807.41	421892.98	44.579255
export	394435.93	612607.17	55.312213

REGRESSION USING PIPELINE

y_col = ‘2021-2022_total’
num_cols = [‘2020-2021_import’, ‘%Share_import’, ‘2020-2021_export’, ‘%Share_export’, ‘2020-2021_total’, ‘%Share_total’]
cat_cols = [‘Continent’]
x_cols = num_cols + cat_cols

print(f”Y : {y_col}”)
print(f”X numeric : {num_cols}”)
print(f”X category : {cat_cols}”)
random_seed = 1
lost_rate = 0.3
sample_n = int(lost_rate * len(india_df))

df = india_df[[‘Country’, y_col]+x_cols].copy()
test_name = df[‘Country’].sample(n=sample_n, random_state=random_seed).tolist()
test_df = df[df[‘Country’].isin(test_name)]
train_df = df[~df[‘Country’].isin(test_name)]

display(df.head(3))
print(f”Total data size : {len(df)}”)
print(f”Test data size : {len(test_df)}”)
print(f”Train data size : {len(train_df)}”)

X_train = train_df.drop(columns=[y_col])
y_train = train_df[y_col]

X_test = test_df.drop(columns=[y_col])
y_test = test_df[y_col]

```
random_seed = 1
lost_rate = 0.3
sample_n = int(lost_rate * len(india_df))


df = india_df[['Country', y_col]+x_cols].copy()
test_name = df['Country'].sample(n=sample_n, random_state=random_seed).tolist()
test_df = df[df['Country'].isin(test_name)]
train_df = df[~df['Country'].isin(test_name)]

display(df.head(3))
print(f"Total data size : {len(df)}")
print(f"Test data size : {len(test_df)}")
print(f"Train data size : {len(train_df)}")

X_train = train_df.drop(columns=[y_col])
y_train = train_df[y_col]

X_test = test_df.drop(columns=[y_col])
y_test = test_df[y_col]
```

	Country	2021-2022_total	2020-2021_import	%Share_import	2020-2021_export	%Share_export	2020-2021_total	%Share_total	Continent
0	AFGHANISTAN	1065.40	825.78	0.2830	509.49	0.1292	1335.27	0.194577	Asia
1	ALBANIA	85.87	46.75	0.0160	8.30	0.0021	55.05	0.008022	Europe
2	ALGERIA	1707.49	594.74	0.2038	408.79	0.1036	1003.53	0.146235	Africa

```
Total data size : 225
Test data size : 67
Train data size : 158
```

Fig5.1: Data Preparation

from sklearn.metrics import mean_absolute_error
from sklearn.pipeline import Pipeline

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.model_selection import KFold, cross_val_score
from sklearn import set_config
set_config(display="diagram")

from sklearn.linear_model import LinearRegression

from sklearn.neighbors import KNeighborsRegressor

from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor

from xgboost import XGBRegressor


num_pipeline = Pipeline([
    ('scaler', StandardScaler())
])
cat_pipeline = Pipeline([
    ('encoder', OneHotEncoder())
])

col_trans = ColumnTransformer([
    ('num_trans', num_pipeline, num_cols),
    ('cat_trans', cat_pipeline, cat_cols),
])

model_pipeline = Pipeline([
    ('prep', col_trans),
    ('reg', LinearRegression())
])

model_pipeline
```

```python
from sklearn.metrics import mean_absolute_error
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.model_selection import KFold, cross_val_score
from sklearn import set_config
set_config(display="diagram")

from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor
from xgboost import XGBRegressor


num_pipeline = Pipeline([
    ('scaler', StandardScaler())
])
cat_pipeline = Pipeline([
    ('encoder', OneHotEncoder())
])

col_trans = ColumnTransformer([
    ('num_trans', num_pipeline, num_cols),
    ('cat_trans', cat_pipeline, cat_cols),
])

model_pipeline = Pipeline([
    ('prep', col_trans),
    ('reg', LinearRegression())
])

model_pipeline
```

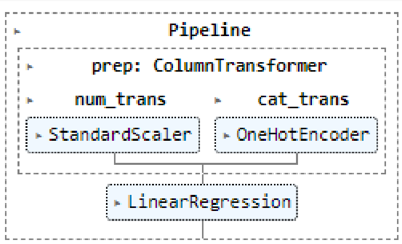


Fig 5.2: Making Pipeline

```python
def run_pipeline_cv(X_train, y_train, model=LinearRegression()):
    num_pipeline = Pipeline([
        ('scaler', StandardScaler())
    ])
    cat_pipeline = Pipeline([
        ('encoder', OneHotEncoder())
    ])

    col_trans = ColumnTransformer([
        ('num_trans', num_pipeline, num_cols),
        ('cat_trans', cat_pipeline, cat_cols),
    ])
```

```
    model_pipeline = Pipeline([
        (‘prep’, col_trans),
        (‘reg’, model)
    ])

    kf = KFold(n_splits=5, shuffle=True, random_state=123)

    model_cv = cross_val_score(model_pipeline, X_train, y_train, scoring=’neg_mean_absolute_error’, cv
=kf)
    return model_cv

ms = []
es = []
st = []
for m in [LinearRegression(), KneighborsRegressor(), RandomForestRegressor(), AdaBoostRegressor(), X
GBRegressor()]:
    cur_model = m.__class__.__name__
    cur_score = run_pipeline_cv(X_train, y_train, model=m)
    cur_error = -1*cur_score.mean() # Since we used negative score in cross validation
    cur_error_std = cur_score.std()
    ms.append(cur_model)
    es.append(cur_error)
    st.append(cur_error_std)
test_result = pd.DataFrame({‘Model’:ms, ‘MAE’:es, ‘MAE_std’:st}).sort_values(by=[‘MAE’], ascending
=True)

# visualization
fig = go.Figure()
fig.add_trace(go.Bar(x=test_result[‘Model’], y=test_result[‘MAE’], error_y=dict(type=’data’, array=test_r
esult[‘MAE_std’]), marker_color=INDIA_GREEN))
fig.update_layout(title_text=f’<b>Cross validation error by model</b>’, template=TEMPLATE)
fig.update_yaxes(title=’MAE’)
fig.show()
```

```python
def run_pipeline_cv(X_train, y_train, model=LinearRegression()):
    num_pipeline = Pipeline([
        ('scaler', StandardScaler())
    ])
    cat_pipeline = Pipeline([
        ('encoder', OneHotEncoder())
    ])

    col_trans = ColumnTransformer([
        ('num_trans', num_pipeline, num_cols),
        ('cat_trans', cat_pipeline, cat_cols),
    ])

    model_pipeline = Pipeline([
        ('prep', col_trans),
        ('reg', model)
    ])

    kf = KFold(n_splits=5, shuffle=True, random_state=123)

    model_cv = cross_val_score(model_pipeline, X_train, y_train, scoring='neg_mean_absolute_error', cv=kf)
    return model_cv

ms = []
es = []
st = []
for m in [LinearRegression(), KNeighborsRegressor(), RandomForestRegressor(), AdaBoostRegressor(), XGBRegressor()]:
    cur_model = m.__class__.__name__
    cur_score = run_pipeline_cv(X_train, y_train, model=m)
    cur_error = -1*cur_score.mean() # Since we used negative score in cross validation
    cur_error_std = cur_score.std()
    ms.append(cur_model)
    es.append(cur_error)
    st.append(cur_error_std)
test_result = pd.DataFrame({'Model':ms, 'MAE':es, 'MAE_std':st}).sort_values(by=['MAE'], ascending=True)

# visualization
fig = go.Figure()
fig.add_trace(go.Bar(x=test_result['Model'], y=test_result['MAE'], error_y=dict(type='data', array=test_result['MAE_std']), marker_color=INDIA_GREEN))
fig.update_layout(title_text=f'<b>Cross validation error by model</b>', template=TEMPLATE)
fig.update_yaxes(title='MAE')
fig.show()
```

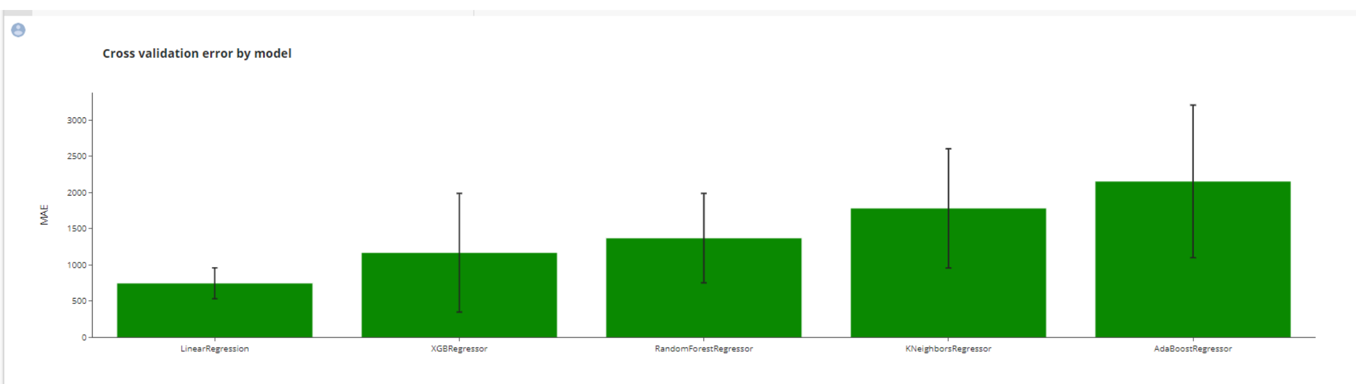


Fig5.3: Cross Validation error by model

```python
# Split
test_df = df[df['Country'].isin(test_name)].reset_index(drop=True).copy()
train_df = df[~df['Country'].isin(test_name)].reset_index(drop=True).copy()
```

```
# Standard Scaling
for c in num_cols:
    ss = StandardScaler()
    train_df.loc[:, c] = ss.fit_transform(train_df[[c]])

# One-hot Encoding
for c in cat_cols:
    oh = OneHotEncoder(sparse=False)
    temp_ = pd.DataFrame(oh.fit_transform(train_df[[c]]))
    temp_.columns = [i.replace(" ", "")for i in oh.get_feature_names_out().tolist()]
    train_df = pd.concat([train_df.drop(columns=[c]), temp_], axis=1)

X_train = train_df.drop(columns=['Country', y_col])
y_train = train_df[y_col]

# Linear Regression
import statsmodels.api as sm # for summary
X_train_cons = sm.add_constant(X_train)
model = sm.OLS(y_train, X_train_cons).fit()
print(model.summary())
```

```
# Split
test_df = df[df['Country'].isin(test_name)].reset_index(drop=True).copy()
train_df = df[~df['Country'].isin(test_name)].reset_index(drop=True).copy()

# Standard Scaling
for c in num_cols:
    ss = StandardScaler()
    train_df.loc[:, c] = ss.fit_transform(train_df[[c]])

# One-hot Encoding
for c in cat_cols:
    oh = OneHotEncoder(sparse=False)
    temp_ = pd.DataFrame(oh.fit_transform(train_df[[c]]))
    temp_.columns = [i.replace(" ", "")for i in oh.get_feature_names_out().tolist()]
    train_df = pd.concat([train_df.drop(columns=[c]), temp_], axis=1)

X_train = train_df.drop(columns=['Country', y_col])
y_train = train_df[y_col]

# Linear Regression
import statsmodels.api as sm # for summary
X_train_cons = sm.add_constant(X_train)
model = sm.OLS(y_train, X_train_cons).fit()
print(model.summary())
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning:

`sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.

                            OLS Regression Results
==============================================================================
Dep. Variable:        2021-2022_total   R-squared:                       0.989
Model:                            OLS   Adj. R-squared:                  0.989
Method:                 Least Squares   F-statistic:                     1378.
Date:                Mon, 10 Apr 2023   Prob (F-statistic):          7.18e-140
Time:                        16:04:27   Log-Likelihood:                -1379.3
No. Observations:                 158   AIC:                             2781.
Df Residuals:                     147   BIC:                             2814.
Df Model:                          10
Covariance Type:            nonrobust
==========================================================================================
```

	coef	std err	t	P>|t|	[0.025	0.975]
const	4331.0524	138.185	31.342	0.000	4057.966	4604.139
2020-2021_import	-1.465e+06	5.89e+06	-0.249	0.804	-1.31e+07	1.02e+07
%Share_import	2.206e+06	6.85e+06	0.322	0.748	-1.13e+07	1.57e+07
2020-2021_export	-1.818e+05	5.54e+06	-0.033	0.974	-1.11e+07	1.08e+07
%Share_export	1.167e+06	8.05e+06	0.145	0.885	-1.47e+07	1.71e+07
2020-2021_total	-7.911e+05	2.69e+06	-0.294	0.769	-6.1e+06	4.52e+06
%Share_total	-7.911e+05	2.69e+06	-0.294	0.769	-6.1e+06	4.52e+06
Continent_Africa	650.2057	261.114	2.490	0.014	134.184	1166.228
Continent_Asia	670.3279	282.485	2.373	0.019	112.071	1228.585
Continent_Europe	446.5778	272.783	1.637	0.104	-92.506	985.661
Continent_NorthAmerica	418.8387	300.735	1.393	0.166	-175.484	1013.162
Continent_Oceania	944.3014	375.425	2.515	0.013	202.375	1686.228
Continent_Others	445.1362	619.071	0.719	0.473	-778.292	1668.565
Continent_SouthAmerica	755.6645	469.760	1.609	0.110	-172.690	1684.019

Omnibus:	173.916	Durbin-Watson:	2.010
Prob(Omnibus):	0.000	Jarque-Bera (JB):	8268.262
Skew:	3.842	Prob(JB):	0.00
Kurtosis:	37.596	Cond. No.	3.60e+16

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 6.6e-31. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

Fig5.4: Removing the factors with high p value

```
# Split
test_df = df[df['Country'].isin(test_name)].reset_index(drop=True).copy()
train_df = df[~df['Country'].isin(test_name)].reset_index(drop=True).copy()

# Standard Scaling
for c in num_cols:
    ss = StandardScaler()
    train_df.loc[:, c] = ss.fit_transform(train_df[[c]])

# One-hot Encoding
for c in cat_cols:
    oh = OneHotEncoder(sparse=False)
    temp_ = pd.DataFrame(oh.fit_transform(train_df[[c]]))
    temp_.columns = [i.replace(" ", "")for i in oh.get_feature_names_out().tolist()]
    train_df = pd.concat([train_df.drop(columns=[c]), temp_], axis=1)

X_train = train_df.drop(columns=['Country', y_col])
test_cols = [
        #'2020-2021_import', # 1
```

```
            '%Share_import',
            #'2020-2021_export', # 2
            '%Share_export',
            #'2020-2021_total', # 4
            #'%Share_total', # 3
            # 'Continent_Africa', # 9
            # 'Continent_Asia', # 10
            #'Continent_Europe', # 6
            #'Continent_NorthAmerica', # 7
            # 'Continent_Oceania', # 11
            #'Continent_Others', # 5
            #'Continent_SouthAmerica' # 8
           ]
X_train = X_train[test_cols]
y_train = train_df[y_col]

# Linear Regression
import statsmodels.api as sm # for summary
X_train_cons = sm.add_constant(X_train)
model = sm.OLS(y_train, X_train_cons).fit()
print(model.summary())

# Cross Validation
kf = KFold(n_splits=5, shuffle=True, random_state=123)
model_cv = cross_val_score(LinearRegression(), X_train, y_train, scoring='neg_mean_absolute_error', cv
=kf)

# Append result
test_result = test_result.append(pd.DataFrame({'Model':['LinearRegression_FS'], 'MAE':[-
1*model_cv.mean()], 'MAE_std':[model_cv.std()]})).sort_values(by=['MAE'], ascending=True)

# visualization
fig = go.Figure()
fig.add_trace(go.Bar(x=test_result['Model'], y=test_result['MAE'], error_y=dict(type='data', array=test_res
ult['MAE_std']), marker_color=INDIA_GREEN))
fig.update_layout(title_text=f'<b>Cross validation error by model</b>', template=TEMPLATE)
fig.update_yaxes(title='MAE')
fig.show()
```

```python
# Split
test_df = df[df['Country'].isin(test_name)].reset_index(drop=True).copy()
train_df = df[~df['Country'].isin(test_name)].reset_index(drop=True).copy()

# Standard Scaling
for c in num_cols:
    ss = StandardScaler()
    train_df.loc[:, c] = ss.fit_transform(train_df[[c]])

# One-hot Encoding
for c in cat_cols:
    oh = OneHotEncoder(sparse=False)
    temp_ = pd.DataFrame(oh.fit_transform(train_df[[c]]))
    temp_.columns = [i.replace(" ", "")for i in oh.get_feature_names_out().tolist()]
    train_df = pd.concat([train_df.drop(columns=[c]), temp_], axis=1)

X_train = train_df.drop(columns=['Country', y_col])
test_cols = [
             #'2020-2021_import', # 1
             '%Share_import',
             #'2020-2021_export', # 2
             '%Share_export',
             #'2020-2021_total', # 4
             #'%Share_total', # 3
             # 'Continent_Africa', # 9
             # 'Continent_Asia', # 10
             #'Continent_Europe', # 6
             #'Continent_NorthAmerica', # 7
             # 'Continent_Oceania', # 11
             #'Continent_Others', # 5
             #'Continent_SouthAmerica' # 8
            ]
X_train = X_train[test_cols]
y_train = train_df[y_col]

# Linear Regression
import statsmodels.api as sm # for summary
X_train_cons = sm.add_constant(X_train)
model = sm.OLS(y_train, X_train_cons).fit()
```

Fig5.5: Applying the predefined modules

```python
print(model.summary())

# Cross Validation
kf = KFold(n_splits=5, shuffle=True, random_state=123)
model_cv = cross_val_score(LinearRegression(), X_train, y_train, scoring='neg_mean_absolute_error', cv=kf)

# Append result
test_result = test_result.append(pd.DataFrame({'Model':['LinearRegression_FS'], 'MAE':[-1*model_cv.mean()], 'MAE_std':[model_cv.std()]})).sort_values(by=['MAE'], ascending=True)

# visualization
fig = go.Figure()
fig.add_trace(go.Bar(x=test_result['Model'], y=test_result['MAE'], error_y=dict(type='data', array=test_result['MAE_std']), marker_color=INDIA_GREEN))
fig.update_layout(title_text=f'<b>Cross validation error by model</b>', template=TEMPLATE)
fig.update_yaxes(title='MAE')
fig.show()
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        2021-2022_total   R-squared:                       0.989
Model:                            OLS   Adj. R-squared:                  0.989
Method:                 Least Squares   F-statistic:                     7172.
Date:                Mon, 10 Apr 2023   Prob (F-statistic):          1.76e-153
Time:                        16:09:36   Log-Likelihood:                -1380.3
No. Observations:                 158   AIC:                             2767.
Df Residuals:                     155   BIC:                             2776.
Df Model:                           2
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
const          4932.2960    120.955     40.778      0.000    4693.363    5171.229
%Share_import  7522.4500    170.557     44.105      0.000    7185.534    7859.366
%Share_export  8165.1865    170.557     47.874      0.000    7828.270    8502.103
==============================================================================
Omnibus:                      176.778   Durbin-Watson:                   2.004
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             8587.267
Skew:                           3.943   Prob(JB):                         0.00
Kurtosis:                      38.245   Cond. No.                         2.40
==============================================================================
```

Fig5.6: Regression Results

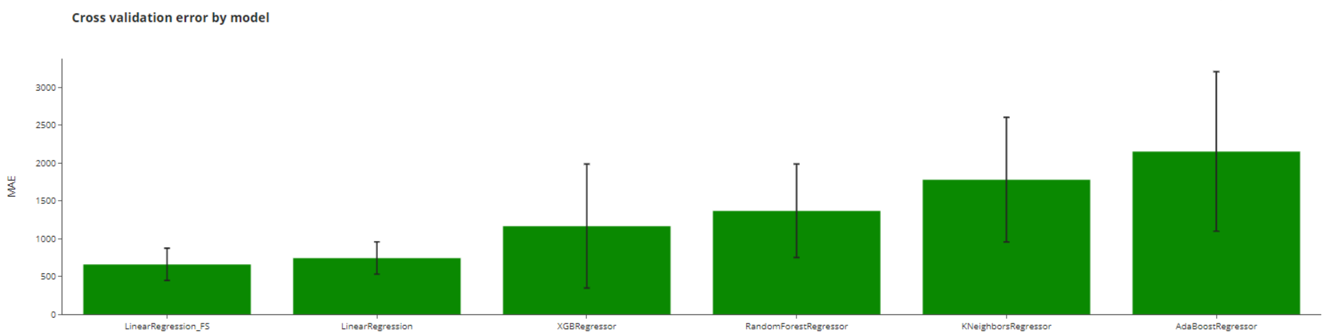


Fig5.7: Results of cross validation

```
# Split
test_df = df[df['Country'].isin(test_name)].reset_index(drop=True).copy()
train_df = df[~df['Country'].isin(test_name)].reset_index(drop=True).copy()

# Standard Scaling
for c in num_cols:
    ss = StandardScaler()
    train_df.loc[:, c] = ss.fit_transform(train_df[[c]])
    test_df.loc[:, c] = ss.transform(test_df[[c]]) # test

# One-hot Encoding : No valid

# X & y
X_train = train_df.drop(columns=['Country', y_col])
y_train = train_df[y_col]
X_test = test_df.drop(columns=['Country', y_col])
y_test = test_df[['Country', y_col]]

# Linear Regression Feature Selection : Train & Test
fs = ['%Share_import', '%Share_export']
model = sm.OLS(y_train, sm.add_constant(X_train[fs])).fit()
y_pred = model.predict(sm.add_constant(X_test[fs]))

# Result
```

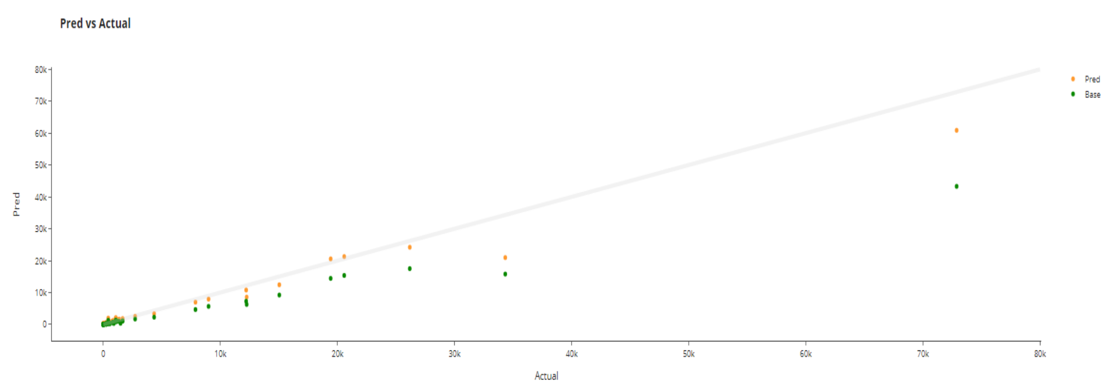
```
pred_df = y_test.copy()
pred_df['Pred'] = y_pred
## Baseline
pred_df = pd.merge(pred_df, df[['Country', '2020-2021_total']].rename(columns={'2020-
2021_total':'Base'}), on='Country')

display(pred_df)

# visualization
fig = go.Figure()
fig.add_trace(go.Scatter(x=pred_df[y_col], y=pred_df['Pred'], text=pred_df['Country'], name='Pred',
                         mode = 'markers',
                         marker_color=INDIA_ORANGE)
             )
fig.add_trace(go.Scatter(x=pred_df[y_col], y=pred_df['Base'], text=pred_df['Country'], name='Base',
                         mode = 'markers',
                         marker_color=INDIA_GREEN)
             )

# Tune marker appearance and layout
fig.update_layout(title_text=f'<b>Pred vs Actual</b>', template=TEMPLATE)
fig.add_shape(type="line", x0=0, y0=0, x1=80000, y1=80000, line=dict(color="LightGray", width=5))
fig.update_yaxes(title="Pred")
fig.update_xaxes(title="Actual")
fig.show()

print(f"Model MAE : {mean_absolute_error(pred_df[y_col], pred_df['Pred'])}")
print(f"Baseline MAE : {mean_absolute_error(pred_df[y_col], pred_df['Base'])}")
```

```
fig.add_trace(go.Scatter(x=pred_df[y_col], y=pred_df['Base'], text=pred_df['Country'], name='Base',
                         mode = 'markers',
                         marker_color=INDIA_GREEN)
             )

# Tune marker appearance and layout
fig.update_layout(title_text=f'<b>Pred vs Actual</b>', template=TEMPLATE)
fig.add_shape(type="line", x0=0, y0=0, x1=80000, y1=80000, line=dict(color="LightGray", width=5))
fig.update_yaxes(title="Pred")
fig.update_xaxes(title="Actual")
fig.show()

print(f"Model MAE : {mean_absolute_error(pred_df[y_col], pred_df['Pred'])}")
print(f"Baseline MAE : {mean_absolute_error(pred_df[y_col], pred_df['Base'])}")
```

	Country	2021-2022_total	Pred	Base
0	AFGHANISTAN	1065.40	2140.044612	1335.27
1	ANDORRA	0.05	176.454745	0.03
2	ARUBA	7.99	182.089322	3.60
3	BAHARAIN IS	1653.19	1717.798991	1075.74
4	BARBADOS	17.03	195.019821	11.71
...	...	...	...	...
62	US MINOR OUTLYING ISLANDS	0.46	178.348426	1.26
63	VENEZUELA	423.78	1978.516078	1271.34
64	VIRGIN IS US	4.72	295.138256	75.06
65	YEMEN REPUBLC	1074.38	1449.617862	808.93
66	ZIMBABWE	208.26	461.576269	181.43

67 rows × 4 columns

Fig5.8: Predicted Values




```
Model MAE : 806.3473294776055
Baseline MAE : 1504.3489552238805
```

Fig5.9: Predicted vs Actual

Market Basket analysis

import pandas as pd

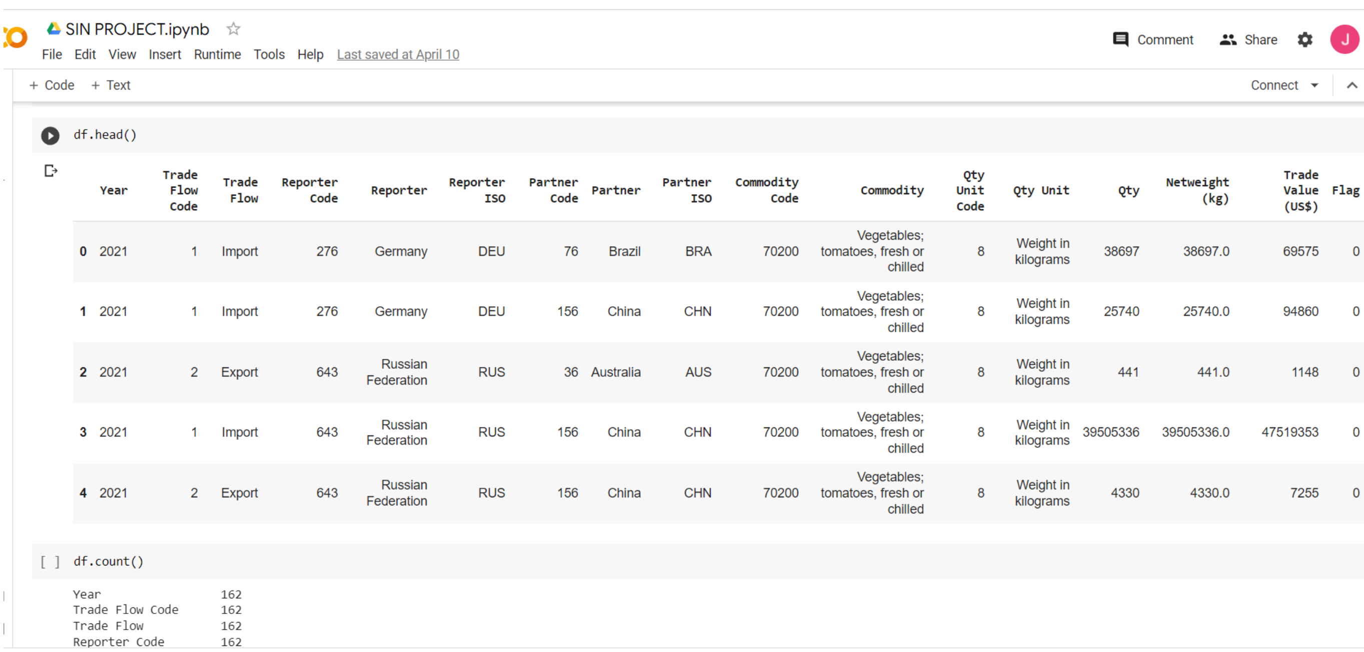
import numpy as np

from mlxtend.frequent_patterns import apriori

from mlxtend.frequent_patterns import association_rules

df=pd.read_csv('/content/sin 20201Book1.csv')

df.head()



	Year	Trade Flow Code	Trade Flow	Reporter Code	Reporter	Reporter ISO	Partner Code	Partner	Partner ISO	Commodity Code	Commodity	Qty Unit Code	Qty Unit	Qty	Netweight (kg)	Trade Value (US$)	Flag
0	2021	1	Import	276	Germany	DEU	76	Brazil	BRA	70200	Vegetables; tomatoes, fresh or chilled	8	Weight in kilograms	38697	38697.0	69575	0
1	2021	1	Import	276	Germany	DEU	156	China	CHN	70200	Vegetables; tomatoes, fresh or chilled	8	Weight in kilograms	25740	25740.0	94860	0
2	2021	2	Export	643	Russian Federation	RUS	36	Australia	AUS	70200	Vegetables; tomatoes, fresh or chilled	8	Weight in kilograms	441	441.0	1148	0
3	2021	1	Import	643	Russian Federation	RUS	156	China	CHN	70200	Vegetables; tomatoes, fresh or chilled	8	Weight in kilograms	39505336	39505336.0	47519353	0
4	2021	2	Export	643	Russian Federation	RUS	156	China	CHN	70200	Vegetables; tomatoes, fresh or chilled	8	Weight in kilograms	4330	4330.0	7255	0

```
[ ] df.count()

Year               162
Trade Flow Code    162
Trade Flow         162
Reporter Code      162
```

Fig 6.1: Retrieving the data

df.count()

df = df.drop("Netweight (kg)", axis = 1)

df = df.drop("Reporter ISO", axis = 1)

df = df.drop("Partner ISO", axis = 1)

df = df.drop("Commodity Code", axis = 1)

df = df.drop("Flag", axis = 1)

df = df.drop("Reporter Code", axis = 1)

df = df.drop("Partner", axis = 1)

df = df.drop("Trade Flow", axis = 1)

df = df.drop("Qty Unit", axis = 1)

df = df.drop("Partner Code", axis = 1)

df.head()

df.count()

df.Reporter.value_counts()

df.Commodity.value_counts()

```
[ ] df.head()
```

	Trade Flow Code	Reporter	Commodity	Qty Unit Code	Qty	Trade Value (US$)
0	1	Germany	Vegetables; tomatoes, fresh or chilled	8	38697	69575
1	1	Germany	Vegetables; tomatoes, fresh or chilled	8	25740	94860
2	2	Russian Federation	Vegetables; tomatoes, fresh or chilled	8	441	1148
3	1	Russian Federation	Vegetables; tomatoes, fresh or chilled	8	39505336	47519353
4	2	Russian Federation	Vegetables; tomatoes, fresh or chilled	8	4330	7255

```
df.count()
```

```
Trade Flow Code      162
Reporter             162
Commodity            162
Qty Unit Code        162
Qty                  162
Trade Value (US$)    162
dtype: int64
```

```
df.Reporter.value_counts()
```

```
Germany               37
Rep. of Korea         35
China                 33
Russian Federation    29
India                 28
Name: Reporter, dtype: int64
```

```
df.Commodity.value_counts()
```

Fig 6.2: Counting the total number of values

#Selecting India from the dataset

df1=df[df.Reporter=='India']

+ Code   + Text

	Trade Flow Code	Reporter	Commodity	Qty Unit Code	Qty	Trade Value (US$)
13	1	India	Nuts, edible; almonds, fresh or dried, in shell	8	23585000	99085868
14	2	India	Nuts, edible; almonds, fresh or dried, in shell	8	5933	29610
19	1	India	Fruit, edible; apricots, fresh	8	3019	9529
46	1	India	Ginger, saffron, tumeric (curcuma), thyme, bay...	8	6020	7391
47	2	India	Ginger, saffron, tumeric (curcuma), thyme, bay...	8	3698127	13057137
48	1	India	Ginger, saffron, tumeric (curcuma), thyme, bay...	8	256995	59910
49	2	India	Ginger, saffron, tumeric (curcuma), thyme, bay...	8	2286570	2880324
50	1	India	Ginger, saffron, tumeric (curcuma), thyme, bay...	8	814183	852902
51	2	India	Ginger, saffron, tumeric (curcuma), thyme, bay...	8	23839184	17629161
70	1	India	Rice	8	63800	35199
71	2	India	Rice	8	58186902	53139249
72	1	India	Rice	8	7	345
73	2	India	Rice	8	42732	36513
74	2	India	Rice	8	1208396556	369391949
106	1	India	Bread, pastry, cakes, biscuits, other bakers' ...	8	6301	25635
107	2	India	Bread, pastry, cakes, biscuits, other bakers' ...	8	5470681	14686502
108	1	India	Bread, pastry, cakes, biscuits, other bakers' ...	8	1161	5316

Fig6.3 : Selecting one country from the reporter

df1.value_counts()

#stripping out the extra spaces presened in the commodity of the dataset

df1['Commodity'].str.strip()

df2=df1['Commodity'].str.strip()

df2

#removing negative values from quantities if any present.

df1 = df1[df1.Qty >0]

df1[df1.Reporter == 'India'].head(20)

basket = pd.pivot_table(data=df1,index='Trade Flow Code',columns='Commodity',values='Qty', \

aggfunc='sum',fill_value=0)

basket.head()

#this to check correctness after binning it to 1 at below code..

basket['Fruit, edible; apricots, fresh'].head(20)

basket['Rice'].head(20)

```
[ ] #this to check correctness after binning it to 1 at below code..
    basket['Fruit, edible; apricots, fresh'].head(20)

    Trade Flow Code
    1    3019
    2       0
    Name: Fruit, edible; apricots, fresh, dtype: int64

    basket['Rice'].head(20)

    Trade Flow Code
    1         63807
    2    1266626190
    Name: Rice, dtype: int64
```

Fig 6.4: Sample of Basket Values

#using binary to represent if a reporte has exported the item or not

def convert_into_binary(x):

```
    if x > 0:
        return 1
    else:
        return 0
basket_sets = basket.applymap(convert_into_binary)
basket_sets['Fruit, edible; apricots, fresh'].head()
basket_sets['Rice']
frequent_itemsets = apriori(basket_sets, min_support=0.10, use_colnames=True)
#it will generate frequent itemsets using two step approch
frequent_itemsets
```

+ Code + Text

```
[ ] basket_sets['Fruit, edible; apricots, fresh'].head()

    Trade Flow Code
    1    1
    2    0
    Name: Fruit, edible; apricots, fresh, dtype: int64

[ ] basket_sets['Rice']

    Trade Flow Code
    1    1
    2    1
    Name: Rice, dtype: int64

[ ] frequent_itemsets = apriori(basket_sets, min_support=0.10, use_colnames=True)

    #it will generate frequent itemsets using two step approch
    frequent_itemsets
```

	support	itemsets
0	1.0	(Bread, pastry, cakes, biscuits, other bakers'...
1	0.5	(Cotton yarn; (not sewing thread), single, of ...
2	0.5	(Fruit, edible; apricots, fresh)
3	1.0	(Ginger, saffron, tumeric (curcuma), thyme, ba...
4	1.0	(Nuts, edible; almonds, fresh or dried, in shell)
...	...	...

Fig 6.5 : Generating the frequent items

rules_mlxtend = association_rules(frequent_itemsets, metric="lift", min_threshold=1)

rules_mlxtend.head()

rules_mlxtend

+ Code + Text

```
[ ] rules_mlxtend = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
    rules_mlxtend.head()
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
0	(Cotton yarn; (not sewing thread), single, of ...	(Bread, pastry, cakes, biscuits, other bakers'...	0.5	1.0	0.5	1.0	1.0	0.0
1	(Bread, pastry, cakes, biscuits, other bakers'...	(Cotton yarn; (not sewing thread), single, of ...	1.0	0.5	0.5	0.5	1.0	0.0
2	(Fruit, edible; apricots, fresh)	(Bread, pastry, cakes, biscuits, other bakers'...	0.5	1.0	0.5	1.0	1.0	0.0
3	(Bread, pastry, cakes, biscuits, other bakers'...	(Fruit, edible; apricots, fresh)	1.0	0.5	0.5	0.5	1.0	0.0
4	(Ginger, saffron, tumeric (curcuma), thyme, ba...	(Bread, pastry, cakes, biscuits, other bakers'...	1.0	1.0	1.0	1.0	1.0	0.0

```
rules_mlxtend
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
0	(Cotton yarn; (not sewing thread), single, of ...	(Bread, pastry, cakes, biscuits, other bakers'...	0.5	1.0	0.5	1.0	1.0	0.00
1	(Bread, pastry, cakes, biscuits, other bakers'...	(Cotton yarn; (not sewing thread), single, of ...	1.0	0.5	0.5	0.5	1.0	0.00
2	(Fruit, edible; apricots, fresh)	(Bread, pastry, cakes, biscuits, other bakers'...	0.5	1.0	0.5	1.0	1.0	0.00
3	(Bread, pastry, cakes, biscuits, other bakers'...	(Fruit, edible; apricots, fresh)	1.0	0.5	0.5	0.5	1.0	0.00
4	(Ginger, saffron, tumeric (curcuma), thyme, ba...	(Bread, pastry, cakes, biscuits, other bakers'...	1.0	1.0	1.0	1.0	1.0	0.00
...	...	...	...	...	...	...	...	...
727	(Bread, pastry, cakes, biscuits, other bakers'...	(Cotton yarn; (not sewing thread), single, of ...	1.0	0.5	0.5	0.5	1.0	0.00
728	(Rice)	(Bread, pastry, cakes, biscuits, other bakers'...	1.0	0.5	0.5	0.5	1.0	0.00

Fig6.6 : Predicting the combination of products to be exported

Predicting the commodity value using decision tree classfier

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

df = pd.read_csv("/content/sin 20201Book1.csv")

df.head()

df = df.drop("Year", axis = 1)

df = df.drop("Netweight (kg)", axis = 1)

df = df.drop("Qty Unit", axis = 1)

df = df.drop("Trade Flow", axis = 1)

df = df.drop("Reporter Code", axis = 1)

df = df.drop("Reporter ISO", axis = 1)

df = df.drop("Partner Code", axis = 1)

df = df.drop("Partner ISO", axis = 1)

df = df.drop("Commodity Code", axis = 1)

df = df.drop("Flag", axis = 1)

df = df.drop("Qty Unit Code", axis = 1)

df.head()

df['Reporter'] = pd.Categorical(df['Reporter']).codes

df['Partner'] = pd.Categorical(df['Partner']).codes

df['Commodity'] = pd.Categorical(df['Commodity']).codes

```
df.head()
```

	Trade Flow Code	Reporter	Partner	Commodity	Qty	Trade Value (US$)
0	1	Germany	Brazil	Vegetables; tomatoes, fresh or chilled	38697	69575
1	1	Germany	China	Vegetables; tomatoes, fresh or chilled	25740	94860
2	2	Russian Federation	Australia	Vegetables; tomatoes, fresh or chilled	441	1148
3	1	Russian Federation	China	Vegetables; tomatoes, fresh or chilled	39505336	47519353
4	2	Russian Federation	China	Vegetables; tomatoes, fresh or chilled	4330	7255

```
[5] df['Reporter'] = pd.Categorical(df['Reporter']).codes
    df['Partner'] = pd.Categorical(df['Partner']).codes
    df['Commodity'] = pd.Categorical(df['Commodity']).codes
```

```
[6] df.head()
```

	Trade Flow Code	Reporter	Partner	Commodity	Qty	Trade Value (US$)
0	1	1	1	7	38697	69575
1	1	1	2	7	25740	94860
2	2	4	0	7	441	1148
3	1	4	2	7	39505336	47519353

Fig 7.1 Modifying the dataset

from sklearn.tree import DecisionTreeClassifier

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score

Extract features and target variable

```
features = df[['Trade Flow Code', 'Reporter', 'Qty', 'Trade Value (US$)']]
target = df['Commodity']
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)
# Initialize the decision tree classifier
clf = DecisionTreeClassifier()
# Fit the model to the training data
clf.fit(X_train, y_train)
# Make predictions on the test data
y_pred = clf.predict(X_test)
y_pred
```

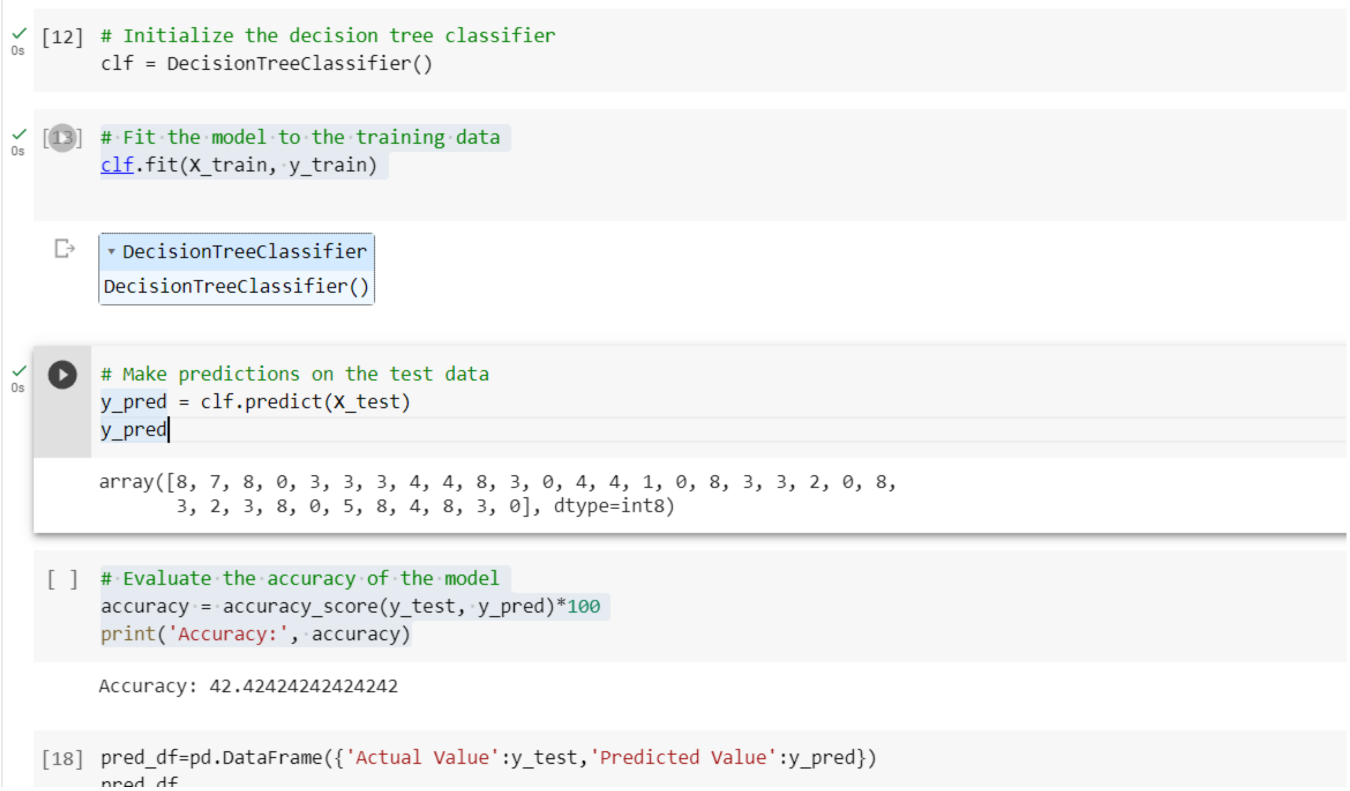


Fig 7.2 Implementing Decision Tree classfier

```
pred_df=pd.DataFrame({'Actual Value':y_test,'Predicted Value':y_pred})
pred_df
```

```
pred_df=pd.DataFrame({'Actual Value':y_test,'Predicted Value':y_pred})
pred_df
```

	Actual Value	Predicted Value
158	8	8
109	0	7
131	8	8
55	5	0
94	0	3
29	3	3
101	0	3
51	3	4
100	0	4
144	8	8
19	2	3
84	0	0
15	2	4
66	5	4
24	3	1

Fig 7.3 Predicted values

Multilinear Regression

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

df = pd.read_csv("/content/sin 20201Book1.csv")

df.head()

df.isna().sum()

df.info()

df.head()

df['Reporter'] = pd.Categorical(df['Reporter']).codes

df['Partner'] = pd.Categorical(df['Partner']).codes df['Commodity'] = pd.Categorical(df['Commodity']).codes

df.head()

```
df.head()
```

	Trade Flow Code	Reporter	Partner	Commodity	Qty	Trade Value (US$)
0	1	Germany	Brazil	Vegetables; tomatoes, fresh or chilled	38697	69575
1	1	Germany	China	Vegetables; tomatoes, fresh or chilled	25740	94860
2	2	Russian Federation	Australia	Vegetables; tomatoes, fresh or chilled	441	1148
3	1	Russian Federation	China	Vegetables; tomatoes, fresh or chilled	39505336	47519353
4	2	Russian Federation	China	Vegetables; tomatoes, fresh or chilled	4330	7255

+ Code + Text

:onverting Categorical values into Numerical Values

```
df['Reporter'] = pd.Categorical(df['Reporter']).codes
```

```
df['Partner'] = pd.Categorical(df['Partner']).codes
```

```
df['Commodity'] = pd.Categorical(df['Commodity']).codes
```

```
df.head()
```

	Trade Flow Code	Reporter	Partner	Commodity	Qty	Trade Value (US$)

Fig 8.1: Sorting the dataset and converting them into numerical values

x = df.drop("Trade Value (US$)", axis = 1)

y = df.pop("Trade Value (US$)")

np.any(np.isnan(x))

np.all(np.isfinite(x))

df.dropna(inplace=True)

df.replace([np.inf, -np.inf], np.nan, inplace=True)

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression lrmodel = LinearRegression()

x_train, x_test , y_train, y_test = train_test_split(x,y,test_size = 0.5, random_state = 1)

lrmodel.fit(x_train,y_train)

lrmodel.score(x_train,y_train)

y_pred=lrmodel.predict(x_test)

y_pred

Predicting the test results

```
y_pred=lrmodel.predict(x_test)
y_pred
```

```
array([7.73506608e+06, 5.15072473e+06, 4.59458138e+06, 2.86914621e+06,
       5.52599721e+06, 9.06700880e+06, 1.40118419e+07, 3.81727777e+06,
       1.37885397e+07, 5.17127267e+06, 3.03795948e+06, 6.67293581e+04,
       3.00284554e+05, 2.78360595e+06, 1.41699472e+07, 1.02534844e+06,
       2.16639040e+06, 7.06490669e+06, 2.66968247e+06, 1.62825975e+06,
       5.22698549e+06, 6.62848519e+06, 1.03241489e+07, 1.33007099e+07,
       9.38550560e+06, 6.52789623e+06, 1.45071047e+07, 9.53551927e+06,
       1.28945350e+06, 2.60494685e+06, 8.27604499e+06, 3.55077553e+08,
       3.60764168e+06, 2.30607535e+06, 9.74895868e+04, 6.50773979e+06,
       1.63983055e+07, 1.55789639e+07, 7.00664297e+06, 1.58391866e+07,
       2.96054699e+06, 1.03533403e+07, 2.21949037e+07, 8.41386259e+06,
       4.70553285e+06, 1.23240357e+07, 1.56614752e+07, 8.89086244e+06,
       1.06292702e+07, 6.24440769e+06, 4.17803886e+06, 8.57859592e+06,
       1.18091819e+07, 2.47290268e+06, 6.70624132e+06, 1.19415889e+07,
       6.14337239e+06, 1.25522213e+07, 1.17833558e+07, 8.75688730e+06,
       6.09441807e+06, 2.40076641e+06, 1.24577900e+07, 1.06936265e+07,
       5.38986132e+07, 5.83597735e+06, 1.39662510e+07, 1.36108283e+07,
       1.40375478e+07, 7.94014808e+06, 7.53021002e+06, 8.35527453e+06,
       1.02203799e+07, 1.19609233e+07, 1.31755434e+07, 1.09250882e+07,
       9.86495721e+06, 1.92787506e+07, 8.60223640e+06, 6.69254570e+06,
       6.81579551e+06])
```

```
[20] from sklearn.metrics import r2_score
```

Fig 8.2: Predicting the test values

plt.scatter(y_test,y_pred);

plt.xlabel('Actual');

plt.ylabel('Predicted');

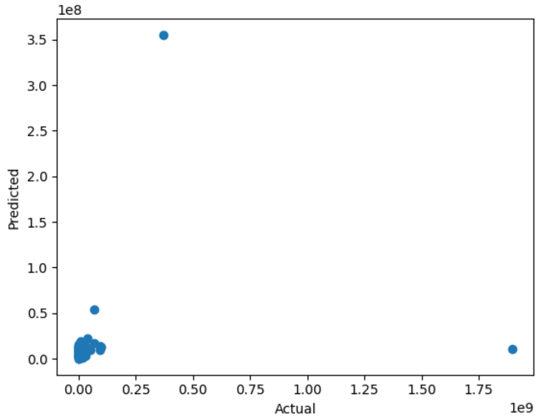


Fig 8.3: Plotting the values between the actual values and predicted values

```
sns.regplot(x=y_test,y=y_pred,ci=None,color ='red');
```

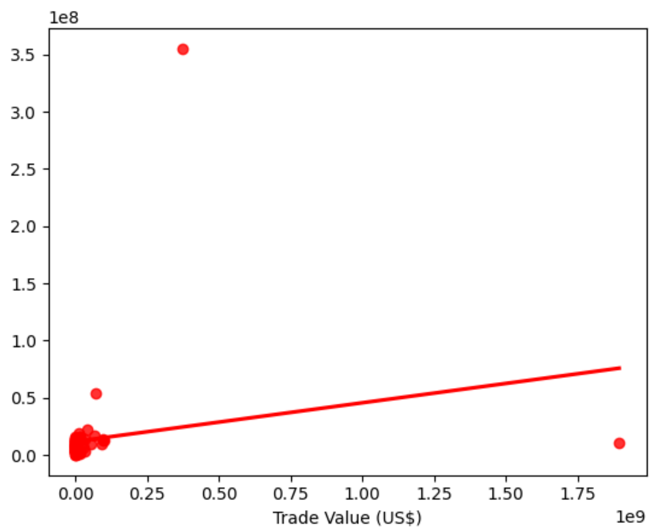
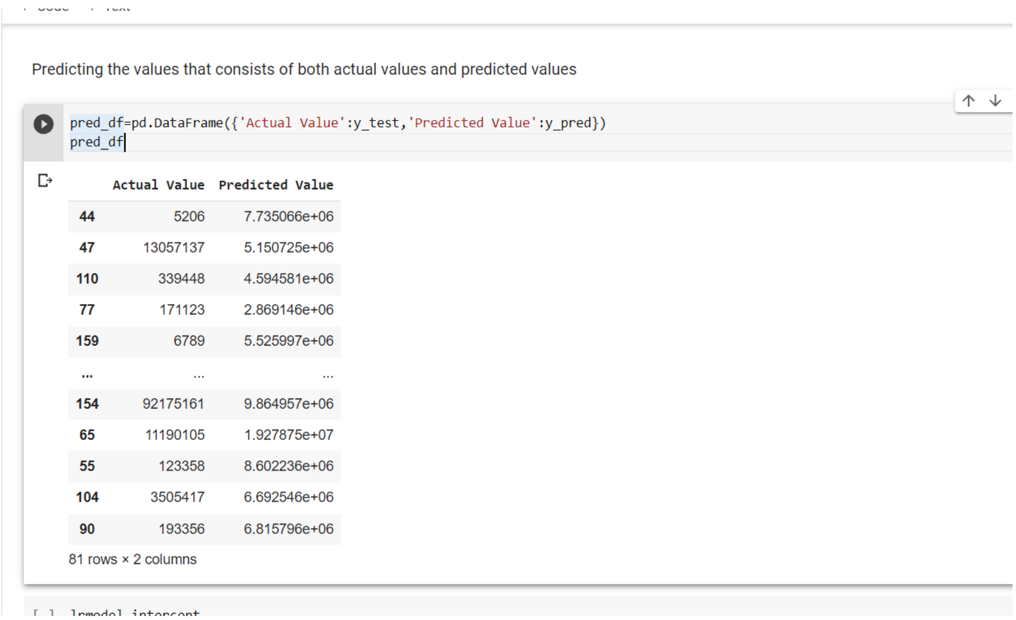


Fig 8.4: Identifying the realtionship between two values

```
pred_df=pd.DataFrame({'Actual Value':y_test,'Predicted Value':y_pred})

pred_df

lrmodel.score(x_train,y_train)
```



Predicting the values that consists of both actual values and predicted values

```
pred_df=pd.DataFrame({'Actual Value':y_test,'Predicted Value':y_pred})
pred_df
```

	Actual Value	Predicted Value
44	5206	7.735066e+06
47	13057137	5.150725e+06
110	339448	4.594581e+06
77	171123	2.869146e+06
159	6789	5.525997e+06
...	...	...
154	92175161	9.864957e+06
65	11190105	1.927875e+07
55	123358	8.602236e+06
104	3505417	6.692546e+06
90	193356	6.815796e+06

81 rows × 2 columns

Fig8.5: Predicting the values that consists of both actual values and predicted values

Conclusion

The COVID-19 pandemic has had a stressful impact on the world economy. World GDP and world trade experienced a massive contraction in recent years. The pandemic originated in China, but the spread effect was huge by spreading through the whole world. In this study, we selected leading trading economies in the world to address the issue of (i) measuring the trade interconnectedness and density before and after the COVID-19 outbreak period We analyzed the trade interconnectedness and density among the leading trading economies in the world are: Canada, US, UK, Germany, France, Italy, Japan, South Korea, China, Australia, India, Indonesia, Russia, Netherlands, and Singapore. We applied trade network analysis for two specific points of time, 2018 and 2021. Trade density has decreased considerably from 0.914 to 0.571. The COVID-19 pandemic has severely hit countries such as Germany, Italy, France, USA, UK. These countries show a steep reduction in degree centrality. Evidently, there is noticeable change in the trade network structure in 2021 compared to 2018.

However, we could find that China was the key center of the trade during 2018 and is slightly repositioned toward ‘circle of center’ by 2021. It clearly indicates that even though the COVID-19 pandemic originated in China in December 2019 and impacted its trade, the country’s relative position in the trade network has not changed drastically. The overall findings show that there will be a significant decline in trade in these economies due to the adverse impact of COVID-19 pandemic.

References

[1] C. T. Vidya and K. P. Prabheesh, Taylor and Francis Group, Vol. 56, NO. 10 (2020)

[2] Celestin Coquide, Jose Lages*, arXiv, Vol. 1 (2022)

[3] Kozo Kiyota, Elsevier, Vol. 78 (2021)