**6.1 APPENDIX**

* CROSS CORRELATION

**# importing the dataset**

df = pd.read\_excel ('final dataset.xlsx')

**# correlation**

data\_corr = df[['Ln\_CO2','Ln Ec', 'Ln Trade', 'Ln UP']]

corr\_mat = data\_corr.corr ()

corr\_mat

* GRAPHS OF THE VARIABLES OVER YEARS

**# importing libraries**

import matplotlib.pyplot as plt

**# CO2 emissions**

plt.plot (df['Year'], df["Ln\_CO2"])

plt.title ("CO2 consumption over the years")

**# Fossil Fuel Energy Consumption**

plt.plot (df['Year'], df['Ln Ec'])

plt.title ("Energy consumption over the years")

**# Trade Openness**

plt.plot (df ['Year'], df ['Ln Trade'])

plt.title ("Trade Openess over the years")

**# Urban Population**

plt.plot (df ['Year'], df ['Ln UP'])

plt.title ("Urban Population over the years")

* DESCRIPTIVE STATISTICS OF Ln\_CO2

data['Ln\_CO2'].describe ()

* ARDL BOUND TESTING

**# importing libraries**

import numpy as np

import pandas as pd

import statsmodels.api as sm

from scipy.stats import f

**# adding constant term**

df = sm.add\_constant (df)

**# defining dependent variable and independent variables**

y = df ['Ln\_CO2']

X = df [['const', 'Ln Ec', 'Ln Trade', 'Ln UP']]

**# estimate the ARDL model**

model = sm.OLS(y, X)

results = model.fit ()

**# calculating the residuals**

residuals = results.resid

**# defining lag length**

k = 3 **# You can adjust the lag length as needed**

**# calculating degrees of freedom**

n = len(y)

dfn = k \* (X.shape [1] - 1)

dfd = n - X.shape [1] \* k - 1

**# calculating SSR and SSE**

SSR = results.ssr

SSE = np.sum (residuals \*\* 2)

**# calculating F-statistic**

F\_statistic = (SSR / dfn) / (SSE / dfd)

**# defining significance level**

alpha = 0.05

**# calculating upper critical bound for F-statistic**

critical\_value = f.ppf (1 - alpha, dfn, dfd)

**# printing results**

print ("ARDL Bound Testing Results :")

print (f"F-statistic: {F\_statistic}")

print (f"Upper critical bound at {alpha} significance level: {critical\_value}")

**# checking for cointegration**

if F\_statistic > critical\_value:

print ("Variables are likely cointegrated (at the specified significance level).")

else:

print ("Variables are not likely cointegrated (at the specified significance level).")

* ARDL MODEL

**# importing libraries**

import numpy as np

import pandas as pd

import statsmodels.api as sm

from statsmodels.tsa.api import ARDL

from sklearn.model\_selection import train\_test\_split

**# importing data**

df = pd.read\_excel ('final dataset.xlsx')

**# defining exogenous variables**

exog = df [['Ln Ec', 'Ln Trade', 'Ln UP']]

**#training the data**

X\_train, X\_test, y\_train, y\_test = train\_test\_split (df [['Ln Ec', 'Ln Trade', 'Ln UP']], df ['Ln\_CO2'], test\_size=0.33, random\_state=42)

X\_train.head ()

model = ARDL (df.Ln\_CO2, 2, exog, 2)

ardl\_model = model.fit ()

**# model summary**

print (ardl\_model.summary ())

* ARDL DIAGNOSTIC TEST

**# importing libraries**

import numpy as np

import pandas as pd

import statsmodels.api as sm

from statsmodels.stats.diagnostic import het\_breuschpagan, het\_white

from statsmodels.stats.stattools import durbin\_watson

**# defining dependent variable and independent variables**

y = df ['Ln\_CO2']

X = df [['Ln Trade', 'Ln UP', 'Ln Ec']]

**# adding constant term to independent variables**

X = sm.add\_constant(X)

**# fitting the ARDL model**

model = sm.OLS(y, X)

results = model.fit ()

**# residuals**

residuals = results.resid

**# serial correlation**

dw\_statistic = durbin\_watson(residuals)

print ("Auto Serial correlation:", dw\_statistic)

**# heteroscedasticity**

bp\_test = het\_breuschpagan (residuals, X)

print ("\nHeteroscedasticity :")

print ("LM Statistic:", bp\_test[0])

print ("LM-Test p-value:", bp\_test[1])

print ("F-Statistic:", bp\_test[2])

print ("F-Test p-value:", bp\_test[3])

* DECISION TREE CLASSIFIER MODEL

**# importing necessary libraries**

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeRegressor

from sklearn import tree

import matplotlib.pyplot as plt

from sklearn.metrics import r2\_score

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, accuracy\_score

**# loading data**

data = pd.read\_excel('final dataset.xlsx')

**# separating features (X) and target variable (y)**

X = data[['Year']]

y = data['Ln\_CO2']

**# splitting the data into train and test sets**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**# creating the decision tree model**

model = DecisionTreeRegressor(random\_state=42)

model.fit(X\_train, y\_train)

**# predicting on the test set**

y\_pred = model.predict(X\_test)

**# evaluating the model**

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

rmse = np.sqrt(mse)

mae = mean\_absolute\_error(y\_test, y\_pred)

**# calculating Mean Squared Error (MSE)**

print("Mean Squared Error (MSE):", mse)

**# calculating Root Mean Squared Error (RMSE)**

print("Root Mean Squared Error (RMSE):", rmse)

**# calculating Mean Absolute Error (MAE)**

print("Mean Absolute Error (MAE):", mae)

**# calculating accuracy**

accuracy = model.score(X\_test, y\_test) \* 100

print("Accuracy (%):", accuracy)

**# creating a DataFrame to display actual values, predicted values, and years**

results\_df = pd.DataFrame({ 'Year': X\_test['Year'].values, 'Actual\_CO2': y\_test.values, 'Predicted\_CO2': y\_pred.round(2)})

**# printing the DataFrame**

print(results\_df)

**# plotting graph for actual CO2 values and predicted CO2 values**

plt.figure(figsize=(10, 5))

plt.scatter(results\_df['Year'], results\_df['Actual\_CO2'], label='Actual CO2', color='purple', marker='o')

plt.scatter(results\_df['Year'], results\_df['Predicted\_CO2'], label='Predicted CO2', color='red', marker='x')

**# adding labels and title**

plt.xlabel('Year')

plt.ylabel('CO2 Emissions')

plt.title('Actual vs Predicted CO2 Emissions')

plt.legend()

**# displaying the plot**

plt.grid(True)

plt.tight\_layout()

plt.show()

* SUPPORT VECTOR MACHINE MODEL

**# importing necessary libraries**

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.svm import SVR

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, accuracy\_score

import matplotlib.pyplot as plt

**# loading data**

data = pd.read\_excel('final dataset.xlsx')

**# separating features (X) and target variable (y)**

X = data[['Year']]

y = data['Ln\_CO2']

**# splitting the data into train and test sets**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**# creating SVM regressor model**

model = SVR(kernel='linear')

**# fitting the model to the training data**

model.fit(X\_train, y\_train)

# **predicting on the test set**

y\_pred = model.predict(X\_test)

**# evaluating the model**

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

rmse = np.sqrt(mse)

mae = mean\_absolute\_error(y\_test, y\_pred)

**# calculating Mean Squared Error (MSE)**

print("Mean Squared Error (MSE):", mse)

**# calculating Root Mean Squared Error (RMSE)**

print("Root Mean Squared Error (RMSE):", rmse)

**# calculating Mean Absolute Error (MAE)**

print("Mean Absolute Error (MAE):", mae)

**# calculating r2 score**

print("r2 score:", r2)

**# creating a DataFrame to display actual values, predicted values, and years**

results\_df\_svm = pd.DataFrame({'Year': X\_test['Year'].values, 'Actual\_CO2': y\_test.values,'Predicted\_CO2': y\_pred.round(2)})

**# printing the DataFrame**

print(results\_df\_svm)

**# plotting graph for actual CO2 values and predicted CO2 values**

plt.figure(figsize=(10, 5))

plt.scatter(results\_df\_svm['Year'], results\_df\_svm['Actual\_CO2'], label='Actual CO2', color='blue', marker='o')

plt.scatter(results\_df\_svm['Year'], results\_df\_svm['Predicted\_CO2'], label='Predicted CO2', color='green', marker='x')

**# adding labels and title**

plt.xlabel('Year')

plt.ylabel('CO2 Emissions')

plt.title('Actual vs Predicted CO2 Emissions')

plt.legend()

**# displaying the plot**

plt.grid(True)

plt.tight\_layout()

plt.show()

* COMPARING BOTH MODELS

**# creating plot**

plt.figure(figsize=(10, 5))

**# scatter plot for actual CO2 values**

plt.scatter(results\_df['Year'], results\_df['Actual\_CO2'], label='Actual CO2', color='purple', marker='o')

**# scatter plot for predicted CO2 values by SVM**

plt.scatter(results\_df\_svm['Year'], results\_df\_svm['Predicted\_CO2'], label='Predicted CO2 (SVM)', color='green', marker='x')

**# scatter plot for predicted CO2 values by Decision Tree**

plt.scatter(results\_df['Year'], results\_df['Predicted\_CO2'], label='Predicted CO2 (DT)', color='blue', marker='s')

**# customizing the plot**

plt.xlabel('Year')

plt.ylabel('CO2 Emissions')

plt.title('Actual vs Predicted CO2 Emissions')

plt.legend()

plt.grid(True)

**# displaying the plot**

plt.tight\_layout()

plt.show()