import numpy as np # linear algebra  
import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)  
import seaborn as sns  
import matplotlib.pyplot as plt

df = pd.read\_csv('yield\_df.csv')

df.head()

Unnamed: 0 Area Item Year hg/ha\_yield \  
0 0 Albania Maize 1990 36613   
1 1 Albania Potatoes 1990 66667   
2 2 Albania Rice, paddy 1990 23333   
3 3 Albania Sorghum 1990 12500   
4 4 Albania Soybeans 1990 7000   
  
 average\_rain\_fall\_mm\_per\_year pesticides\_tonnes avg\_temp   
0 1485.0 121.0 16.37   
1 1485.0 121.0 16.37   
2 1485.0 121.0 16.37   
3 1485.0 121.0 16.37   
4 1485.0 121.0 16.37

df.drop('Unnamed: 0',axis=1,inplace=True) *# Drop irrelevant column*

df.shape

(28242, 7)

df.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 28242 entries, 0 to 28241  
Data columns (total 7 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 Area 28242 non-null object   
 1 Item 28242 non-null object   
 2 Year 28242 non-null int64   
 3 hg/ha\_yield 28242 non-null int64   
 4 average\_rain\_fall\_mm\_per\_year 28242 non-null float64  
 5 pesticides\_tonnes 28242 non-null float64  
 6 avg\_temp 28242 non-null float64  
dtypes: float64(3), int64(2), object(2)  
memory usage: 1.5+ MB

df.isnull().sum()

Area 0  
Item 0  
Year 0  
hg/ha\_yield 0  
average\_rain\_fall\_mm\_per\_year 0  
pesticides\_tonnes 0  
avg\_temp 0  
dtype: int64

df.duplicated().sum()

np.int64(2310)

df.drop\_duplicates(inplace=True)

df.duplicated().sum()

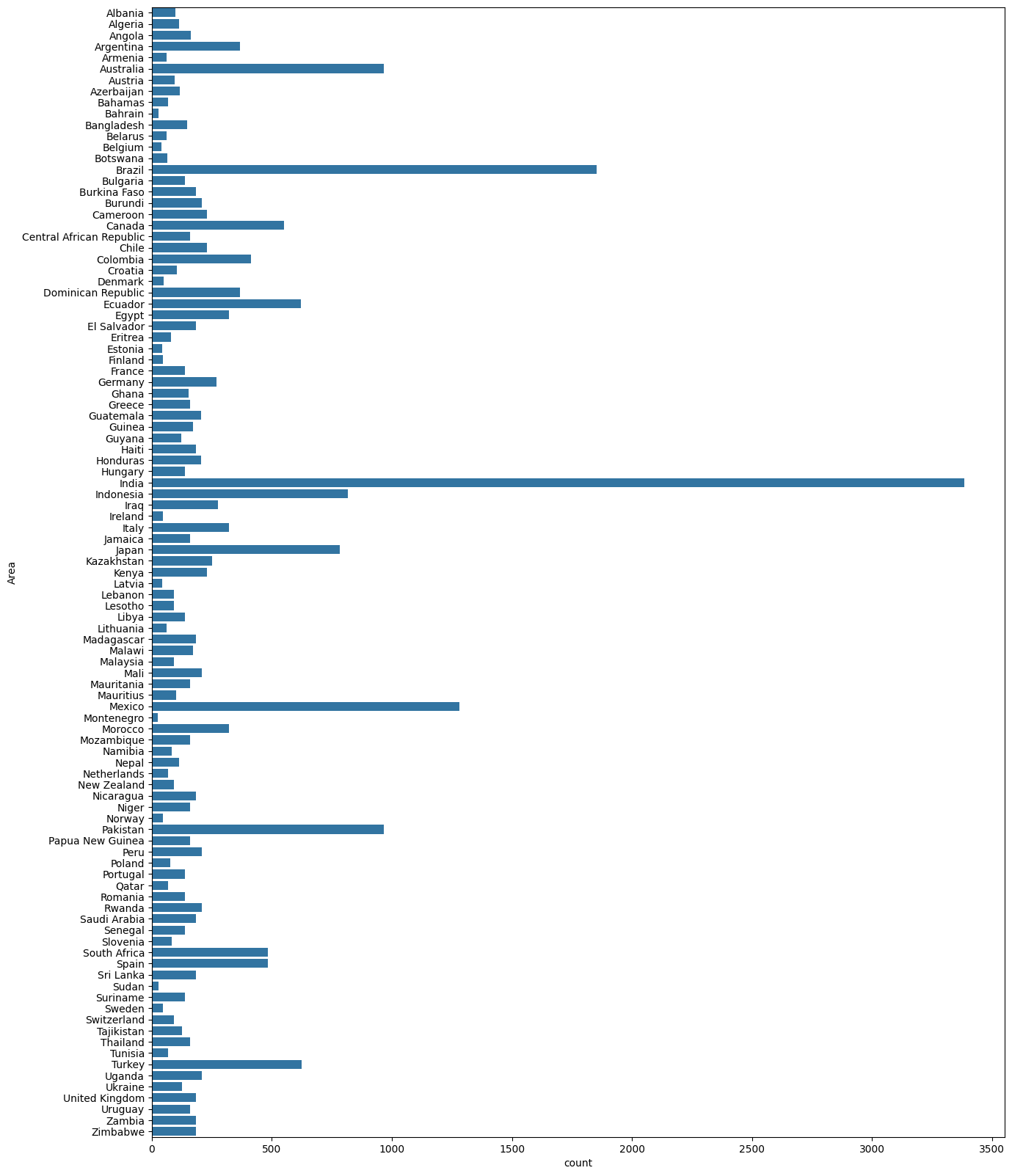
np.int64(0)

# Graph Frequency vs Area

len(df['Area'].unique())

101

plt.figure(figsize=(15,20))  
sns.countplot(y=df['Area'])  
plt.show()

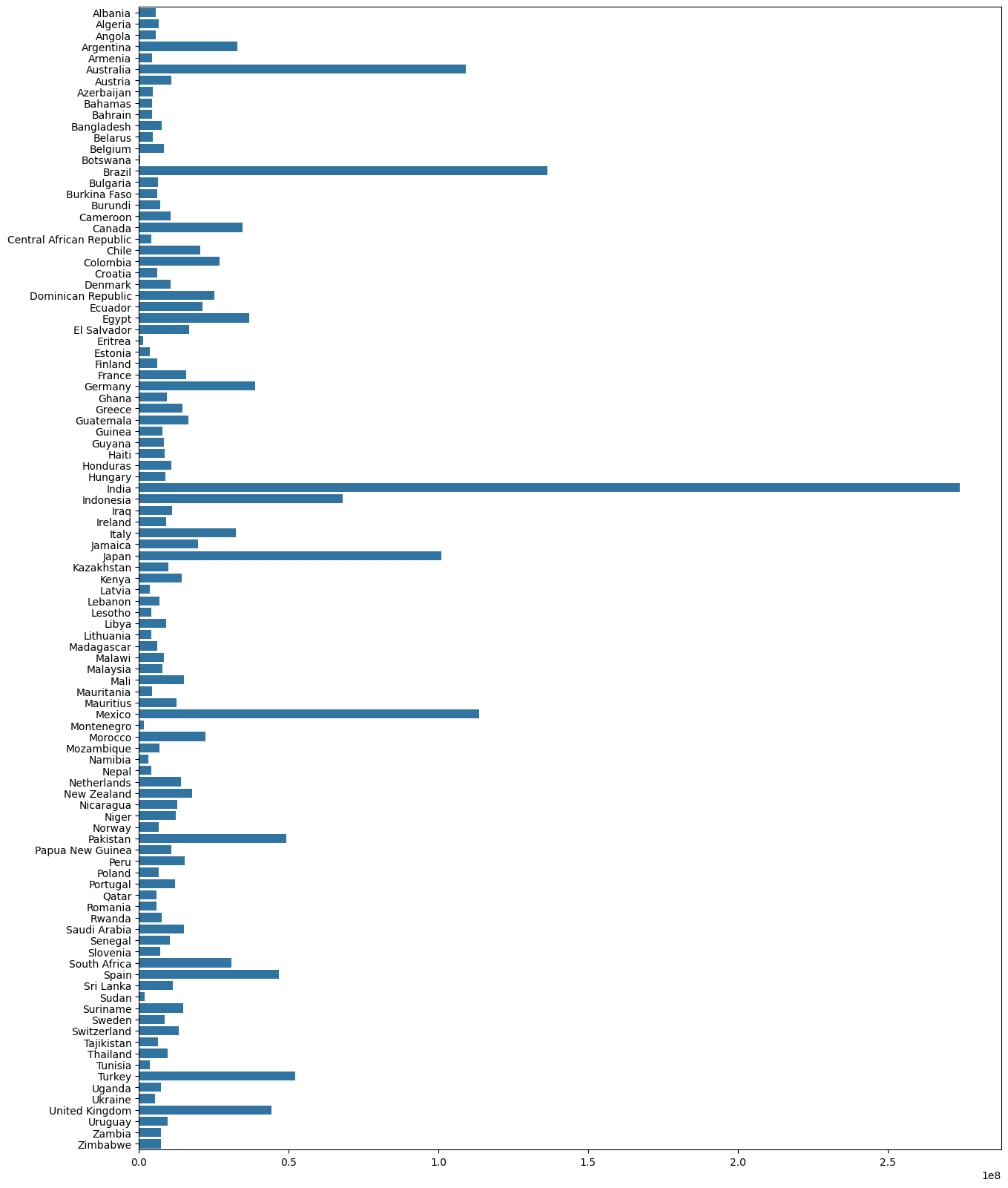


# Yield Per Country Graph

country = df['Area'].unique()  
yield\_per\_country = []  
for state in country:  
 yield\_per\_country.append(df[df['Area']==state]['hg/ha\_yield'].sum())

plt.figure(figsize=(15, 20))  
sns.barplot(y=country, x=yield\_per\_country)

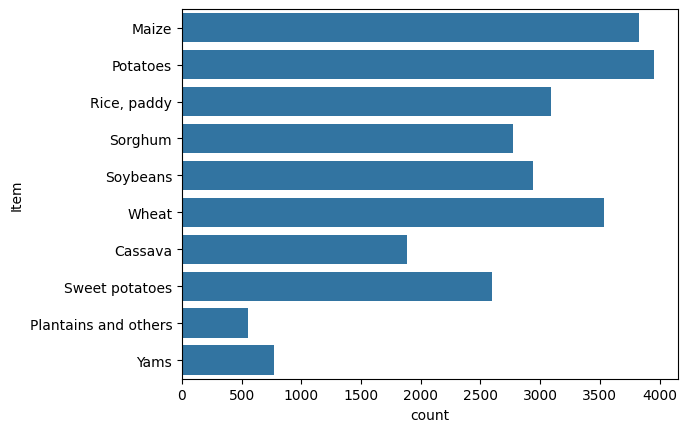
<Axes: >



# Graph Frequency vs Item

sns.countplot(y=df['Item'])

<Axes: xlabel='count', ylabel='Item'>

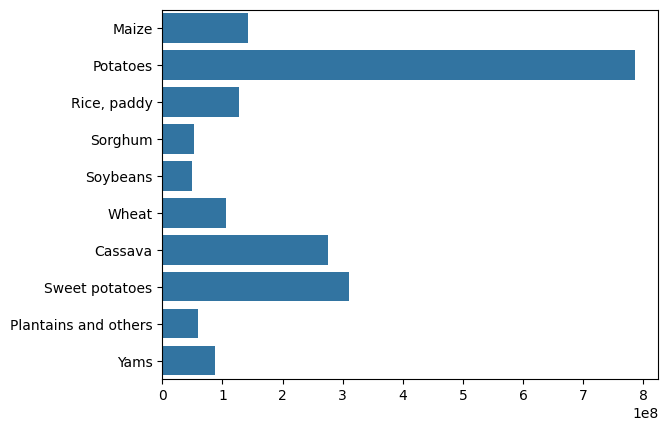


# Yield Vs Item

crops = df['Item'].unique()  
yield\_per\_crop = []  
for crop in crops:  
 yield\_per\_crop.append(df[df['Item']==crop]['hg/ha\_yield'].sum())

sns.barplot(y=crops,x=yield\_per\_crop)

<Axes: >



# Train Test split Rearranging Columns

col = ['Year', 'average\_rain\_fall\_mm\_per\_year','pesticides\_tonnes', 'avg\_temp', 'Area', 'Item', 'hg/ha\_yield']  
df = df[col]  
X = df.iloc[:, :-1]  
y = df.iloc[:, -1]

df.head(3)

Year average\_rain\_fall\_mm\_per\_year pesticides\_tonnes avg\_temp Area \  
0 1990 1485.0 121.0 16.37 Albania   
1 1990 1485.0 121.0 16.37 Albania   
2 1990 1485.0 121.0 16.37 Albania   
  
 Item hg/ha\_yield   
0 Maize 36613   
1 Potatoes 66667   
2 Rice, paddy 23333

from sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, train\_size=0.8, random\_state=0, shuffle=True)

# Converting Categorical to Numerical and Scaling the values

from sklearn.preprocessing import OneHotEncoder  
from sklearn.compose import ColumnTransformer  
from sklearn.preprocessing import StandardScaler  
ohe = OneHotEncoder(drop='first')  
scale = StandardScaler()  
  
preprocesser = ColumnTransformer(  
 transformers = [  
 ('StandardScale', scale, [0, 1, 2, 3]),  
 ('OHE', ohe, [4, 5]),  
 ],  
 remainder='passthrough'  
)

X\_train\_dummy = preprocesser.fit\_transform(X\_train)  
X\_test\_dummy = preprocesser.transform(X\_test)

preprocesser.get\_feature\_names\_out(col[:-1])

array(['StandardScale\_\_Year',  
 'StandardScale\_\_average\_rain\_fall\_mm\_per\_year',  
 'StandardScale\_\_pesticides\_tonnes', 'StandardScale\_\_avg\_temp',  
 'OHE\_\_Area\_Algeria', 'OHE\_\_Area\_Angola', 'OHE\_\_Area\_Argentina',  
 'OHE\_\_Area\_Armenia', 'OHE\_\_Area\_Australia', 'OHE\_\_Area\_Austria',  
 'OHE\_\_Area\_Azerbaijan', 'OHE\_\_Area\_Bahamas', 'OHE\_\_Area\_Bahrain',  
 'OHE\_\_Area\_Bangladesh', 'OHE\_\_Area\_Belarus', 'OHE\_\_Area\_Belgium',  
 'OHE\_\_Area\_Botswana', 'OHE\_\_Area\_Brazil', 'OHE\_\_Area\_Bulgaria',  
 'OHE\_\_Area\_Burkina Faso', 'OHE\_\_Area\_Burundi',  
 'OHE\_\_Area\_Cameroon', 'OHE\_\_Area\_Canada',  
 'OHE\_\_Area\_Central African Republic', 'OHE\_\_Area\_Chile',  
 'OHE\_\_Area\_Colombia', 'OHE\_\_Area\_Croatia', 'OHE\_\_Area\_Denmark',  
 'OHE\_\_Area\_Dominican Republic', 'OHE\_\_Area\_Ecuador',  
 'OHE\_\_Area\_Egypt', 'OHE\_\_Area\_El Salvador', 'OHE\_\_Area\_Eritrea',  
 'OHE\_\_Area\_Estonia', 'OHE\_\_Area\_Finland', 'OHE\_\_Area\_France',  
 'OHE\_\_Area\_Germany', 'OHE\_\_Area\_Ghana', 'OHE\_\_Area\_Greece',  
 'OHE\_\_Area\_Guatemala', 'OHE\_\_Area\_Guinea', 'OHE\_\_Area\_Guyana',  
 'OHE\_\_Area\_Haiti', 'OHE\_\_Area\_Honduras', 'OHE\_\_Area\_Hungary',  
 'OHE\_\_Area\_India', 'OHE\_\_Area\_Indonesia', 'OHE\_\_Area\_Iraq',  
 'OHE\_\_Area\_Ireland', 'OHE\_\_Area\_Italy', 'OHE\_\_Area\_Jamaica',  
 'OHE\_\_Area\_Japan', 'OHE\_\_Area\_Kazakhstan', 'OHE\_\_Area\_Kenya',  
 'OHE\_\_Area\_Latvia', 'OHE\_\_Area\_Lebanon', 'OHE\_\_Area\_Lesotho',  
 'OHE\_\_Area\_Libya', 'OHE\_\_Area\_Lithuania', 'OHE\_\_Area\_Madagascar',  
 'OHE\_\_Area\_Malawi', 'OHE\_\_Area\_Malaysia', 'OHE\_\_Area\_Mali',  
 'OHE\_\_Area\_Mauritania', 'OHE\_\_Area\_Mauritius', 'OHE\_\_Area\_Mexico',  
 'OHE\_\_Area\_Montenegro', 'OHE\_\_Area\_Morocco',  
 'OHE\_\_Area\_Mozambique', 'OHE\_\_Area\_Namibia', 'OHE\_\_Area\_Nepal',  
 'OHE\_\_Area\_Netherlands', 'OHE\_\_Area\_New Zealand',  
 'OHE\_\_Area\_Nicaragua', 'OHE\_\_Area\_Niger', 'OHE\_\_Area\_Norway',  
 'OHE\_\_Area\_Pakistan', 'OHE\_\_Area\_Papua New Guinea',  
 'OHE\_\_Area\_Peru', 'OHE\_\_Area\_Poland', 'OHE\_\_Area\_Portugal',  
 'OHE\_\_Area\_Qatar', 'OHE\_\_Area\_Romania', 'OHE\_\_Area\_Rwanda',  
 'OHE\_\_Area\_Saudi Arabia', 'OHE\_\_Area\_Senegal',  
 'OHE\_\_Area\_Slovenia', 'OHE\_\_Area\_South Africa', 'OHE\_\_Area\_Spain',  
 'OHE\_\_Area\_Sri Lanka', 'OHE\_\_Area\_Sudan', 'OHE\_\_Area\_Suriname',  
 'OHE\_\_Area\_Sweden', 'OHE\_\_Area\_Switzerland',  
 'OHE\_\_Area\_Tajikistan', 'OHE\_\_Area\_Thailand', 'OHE\_\_Area\_Tunisia',  
 'OHE\_\_Area\_Turkey', 'OHE\_\_Area\_Uganda', 'OHE\_\_Area\_Ukraine',  
 'OHE\_\_Area\_United Kingdom', 'OHE\_\_Area\_Uruguay',  
 'OHE\_\_Area\_Zambia', 'OHE\_\_Area\_Zimbabwe', 'OHE\_\_Item\_Maize',  
 'OHE\_\_Item\_Plantains and others', 'OHE\_\_Item\_Potatoes',  
 'OHE\_\_Item\_Rice, paddy', 'OHE\_\_Item\_Sorghum', 'OHE\_\_Item\_Soybeans',  
 'OHE\_\_Item\_Sweet potatoes', 'OHE\_\_Item\_Wheat', 'OHE\_\_Item\_Yams'],  
 dtype=object)

# Let's train our model

#linear regression  
from sklearn.linear\_model import LinearRegression,Lasso,Ridge  
from sklearn.neighbors import KNeighborsRegressor  
from sklearn.tree import DecisionTreeRegressor  
from sklearn.metrics import mean\_absolute\_error,r2\_score  
  
  
models = {  
 'lr':LinearRegression(),  
 'lss':Lasso(),  
 'Rid':Ridge(),  
 'Dtr':DecisionTreeRegressor()  
}  
for name, md in models.items():  
 md.fit(X\_train\_dummy,y\_train)  
 y\_pred = md.predict(X\_test\_dummy)  
   
 print(f"{name} : mae : {mean\_absolute\_error(y\_test,y\_pred)} score : {r2\_score(y\_test,y\_pred)}")

lr : mae : 29907.52454950499 score : 0.7473119790119538

c:\Users\thero\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\linear\_model\\_coordinate\_descent.py:658: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations. Duality gap: 67280771830.01953, tolerance: 14848622817.505228  
 model = cd\_fast.sparse\_enet\_coordinate\_descent(

lss : mae : 29893.99762450549 score : 0.7473261756207235  
Rid : mae : 29864.881035949216 score : 0.7473041855623102  
Dtr : mae : 3942.563909774436 score : 0.9799132947916251

# Select model

dtr = DecisionTreeRegressor()  
dtr.fit(X\_train\_dummy,y\_train)  
dtr.predict(X\_test\_dummy)

array([35286., 22814., 19295., ..., 15011., 34879., 79048.])

# Predictive System

def prediction(Year, average\_rain\_fall\_mm\_per\_year, pesticides\_tonnes, avg\_temp, Area, Item):  
 # Create an array of the input features  
 features = np.array([[Year, average\_rain\_fall\_mm\_per\_year, pesticides\_tonnes, avg\_temp, Area, Item]], dtype=object)  
  
 # Transform the features using the preprocessor  
 transformed\_features = preprocesser.transform(features)  
  
 # Make the prediction  
 predicted\_yield = dtr.predict(transformed\_features).reshape(1, -1)  
  
 return predicted\_yield[0]  
  
Year = 1990  
average\_rain\_fall\_mm\_per\_year =1485.0  
pesticides\_tonnes = 121.00  
avg\_temp = 16.37   
Area = 'Albania'  
Item = 'Maize'  
result = prediction(Year, average\_rain\_fall\_mm\_per\_year, pesticides\_tonnes, avg\_temp, Area, Item)

c:\Users\thero\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but StandardScaler was fitted with feature names  
 warnings.warn(  
c:\Users\thero\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but OneHotEncoder was fitted with feature names  
 warnings.warn(

result

array([36613.])

# 1990 1485.0 121.00 16.37 Albania Maize 36613  
# 2013 657.0 2550.07 19.76 Zimbabwe Sorghum 3066

# Pickle Files

import pickle  
pickle.dump(dtr,open('dtr.pkl','wb'))  
pickle.dump(preprocesser,open('preprocessor.pkl','wb'))

import sklearn  
print(sklearn.\_\_version\_\_)

1.5.2