In [2]:

```
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
```

In [3]:

```
df = pd.read_csv('yield_df.csv')
```

In [4]:

```
df.head()
```

Out[4]:

	Unnamed: 0	Area	Item	Year	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp
0	0	Albania	Maize	1990	36613	1485.0	121.0	16.37
1	1	Albania	Potatoes	1990	66667	1485.0	121.0	16.37
2	2	Albania	Rice, paddy	1990	23333	1485.0	121.0	16.37
3	3	Albania	Sorghum	1990	12500	1485.0	121.0	16.37
4	4	Albania	Soybeans	1990	7000	1485.0	121.0	16.37

In [5]:

```
df.drop('Unnamed: 0',axis=1,inplace=True)
```

In [6]:

```
df.shape
```

Out[6]:

```
(28242, 7)
```

In [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
```

In [8]:

```
df.isnull().sum()
```

Out[8]:

```
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
```

In [9]:

```
df.duplicated().sum()
```

Out[9]:

```
2310
```

In [10]:

```
df.drop_duplicates(inplace=True)
```

In [11]:

```
df.duplicated().sum()
```

Out[11]:

```
0
```

Transforming average_rain_fall_mm_per_year

In summary, this code identifies the indices of rows in the DataFrame df where the values in the column 'average_rain_fall_mm_per_year' are not numeric strings. These rows can be considered for removal or further processing, depending on the specific use case.

In [12]:

```
def isStr(obj):
    try:
        float(obj)
        return False
    except:
        return True
to_drop = df[df['average_rain_fall_mm_per_year'].apply(isStr)].index
```

In [13]:

```
df = df.drop(to_drop)
```

In [14]:

```
df
```

Out[14]:

```
          Area            Item  Year  hg/ha_yield  average_rain_fall_mm_per_year  pesticides_tonnes  avg_temp
    0     Albania          Maize  1990        36613                         1485.0             121.00     16.37
    1     Albania       Potatoes  1990        66667                         1485.0             121.00     16.37
    2     Albania    Rice, paddy  1990        23333                         1485.0             121.00     16.37
    3     Albania        Sorghum  1990        12500                         1485.0             121.00     16.37
    4     Albania       Soybeans  1990         7000                         1485.0             121.00     16.37
  ...         ...            ...   ...          ...                            ...                ...       ...
28237    Zimbabwe    Rice, paddy  2013        22581                          657.0            2550.07     19.76
28238    Zimbabwe        Sorghum  2013         3066                          657.0            2550.07     19.76
28239    Zimbabwe       Soybeans  2013        13142                          657.0            2550.07     19.76
28240    Zimbabwe Sweet potatoes  2013        22222                          657.0            2550.07     19.76
28241    Zimbabwe          Wheat  2013        22888                          657.0            2550.07     19.76

25932 rows × 7 columns
```

In [15]:

```
df['average_rain_fall_mm_per_year'] = df['average_rain_fall_mm_per_year'].astype(np.float64)
```

Graph Frequency vs Area

In [16]:

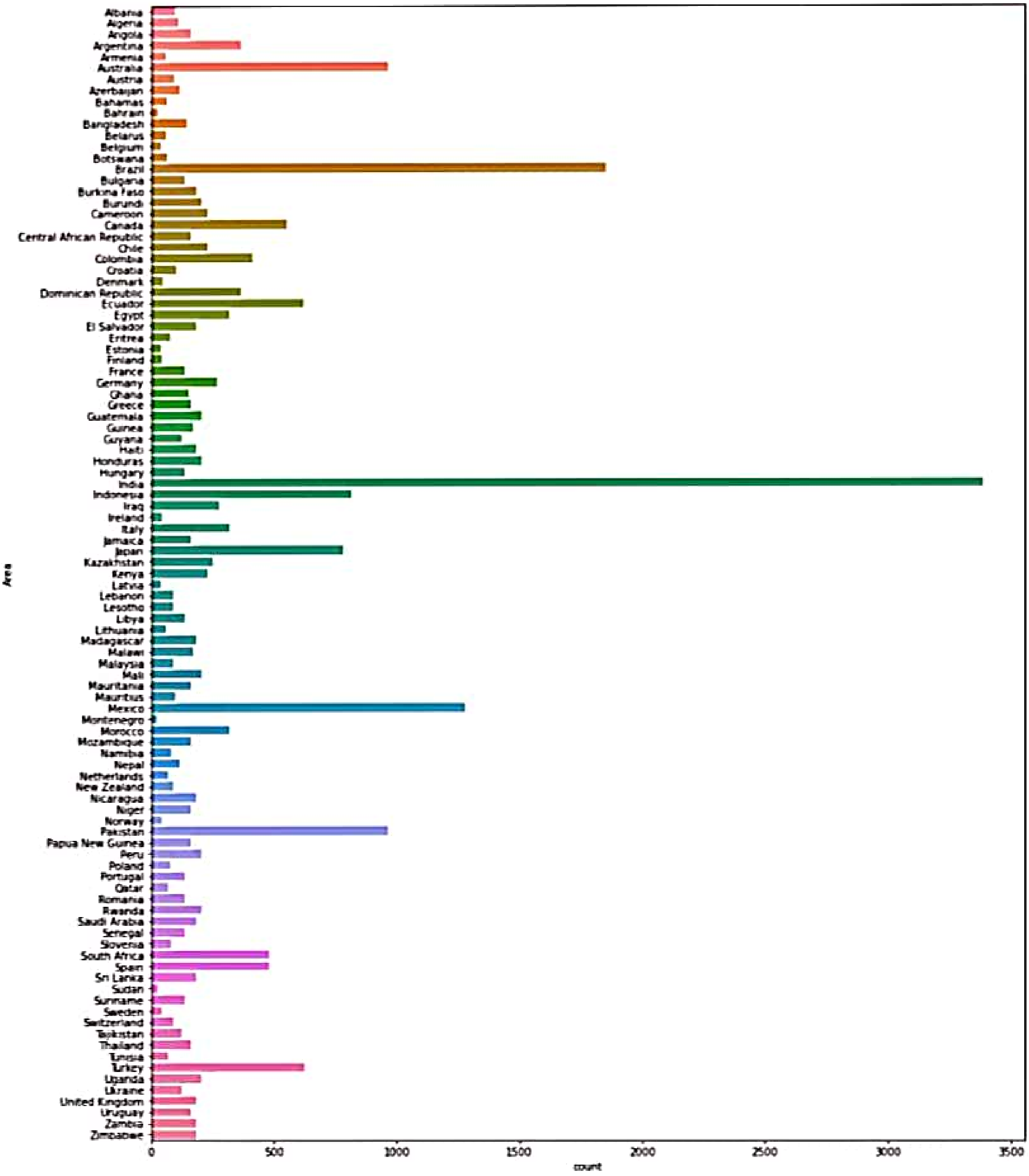
```
len(df['Area'].unique())
```

Out[16]:

```
101
```

In [17]:

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



In [18]:

```
(df['Area'].value_counts() < 500).sum()
```

Out[18]:

```
91
```

yield_per_country

```
In [19]:
country = df['Area'].unique()
yield_per_country = []
for state in country:
    yield_per_country.append(df[df['Area']==state]['hg/ha_yield'].sum())
```

```
In [20]:
df['hg/ha_yield'].sum()
```

```
Out[20]: 1996196943
```

```
In [21]:
yield_per_country
```

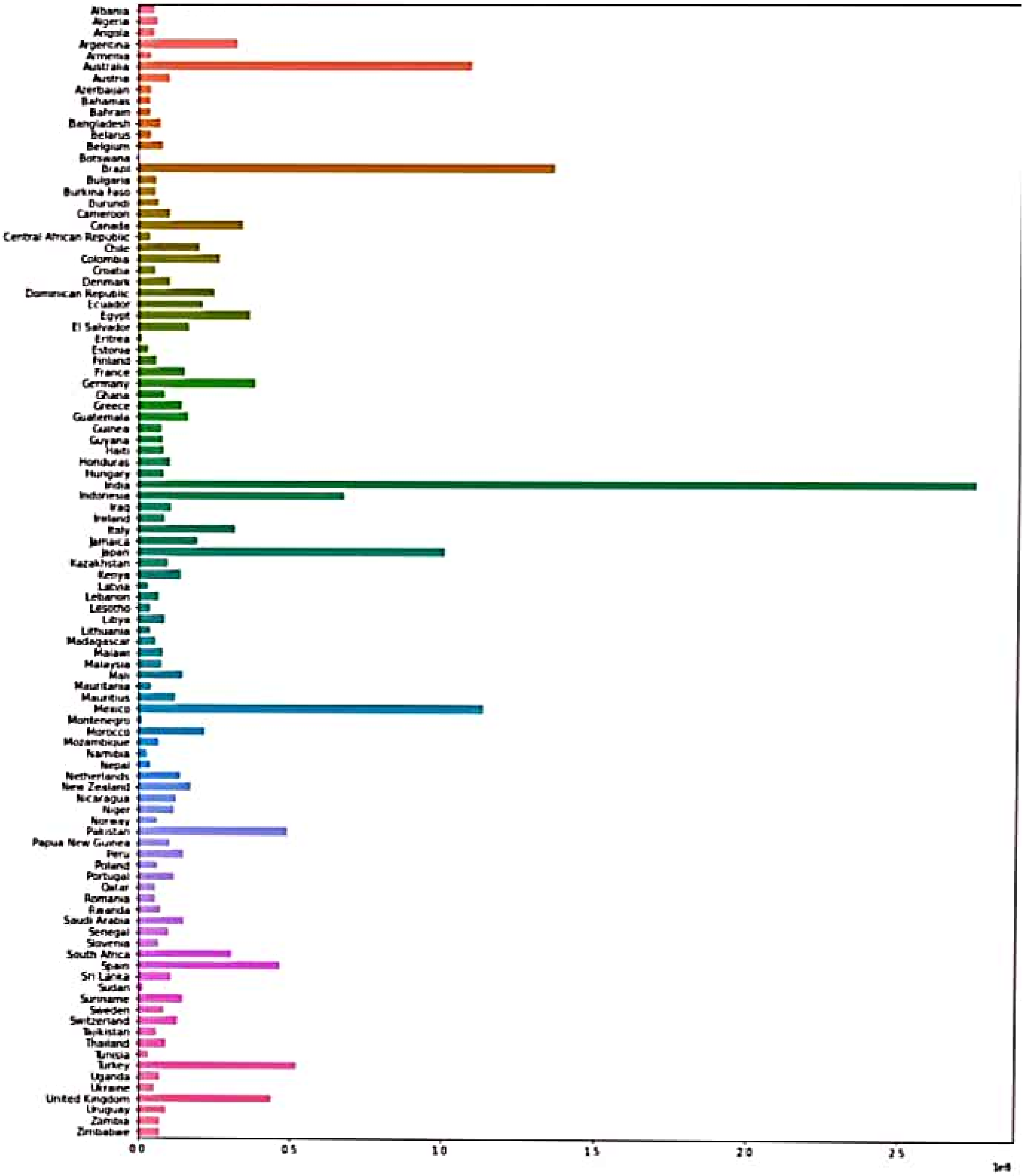
```
Out[21]: [5711536,
 6711464,
 5722563,
 32864032,
 4524100,
 109111062,
 10852258,
 4608380,
 4384717,
 4443889,
 7720159,
 4704812,
 8442270,
 470651,
 136340329,
 6263075,
 6083337,
 7031146,
 10717883,
 34706922,
 4255627,
 20561214,
 26927138,
 6083167,
 10701651,
 25312166,
 21315591,
 36828848,
 16855944,
 1452416,
 3595638,
 6210668,
 15790618,
 38780463,
 9260371,
 14571925,
 16508723,
 7975743,
 8361103,
 8619194,
 10920131,
 8824110,
 274219558,
 68067328,
 10984722,
 9104030,
 32280700,
 19698007,
 100924145,
 9965212,
 14391737,
 3698588,
 6956804,
 4258932,
 9016288,
 4174348,
 6103523,
 8346715
```

Yield Per Country Graph

In [22]:

```
plt.figure(figsize=(15, 20))
sns.barplot(y=country, x=yield_per_country)
```

Out[22]: `<AxesSubplot:>`

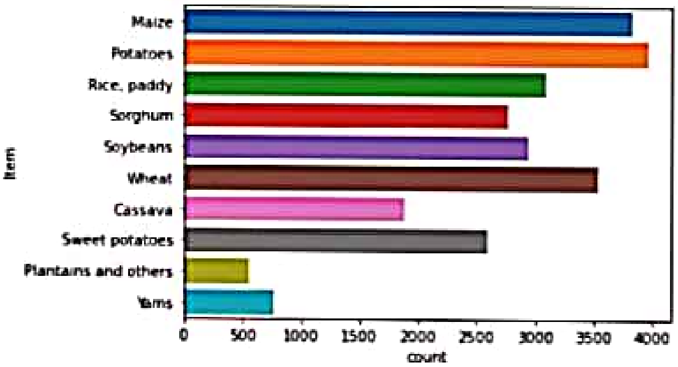


Graph Frequency vs Item

In [23]:

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

Out[23]: `<AxesSubplot:xlabel='count', ylabel='Item'>`



Yield Vs Item

In [24]:

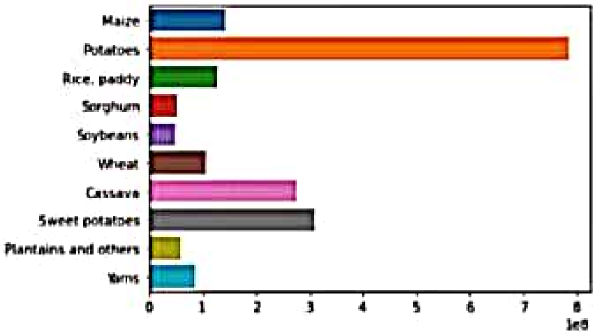
```
crops = df['Item'].unique()
yield_per_crop = []
for crop in crops:
    yield_per_crop.append(df[df['Item']==crop]['hg/ha_yield'].sum())
```

In [25]:

```
sns.barplot(y=crops,x=yield_per_crop)
```

Out[25]:

```
<AxesSubplot:>
```



Train Test split Rearranging Columns

In [26]:

```
col = ['Year', 'average_rain_fall_mm_per_year','pesticides_tonnes', 'avg_temp', 'Area', 'Item', 'hg/ha_
df = df[col]
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
```

In [27]:

```
df.head(3)
```

Out[27]:

	Year	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp	Area	Item	hg/ha_yield
0	1990	1485.0	121.0	16.37	Albania	Maize	36613
1	1990	1485.0	121.0	16.37	Albania	Potatoes	66667
2	1990	1485.0	121.0	16.37	Albania	Rice, paddy	23333

In [28]:

```
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

In [29]:

```
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'
)
```

In [30]:

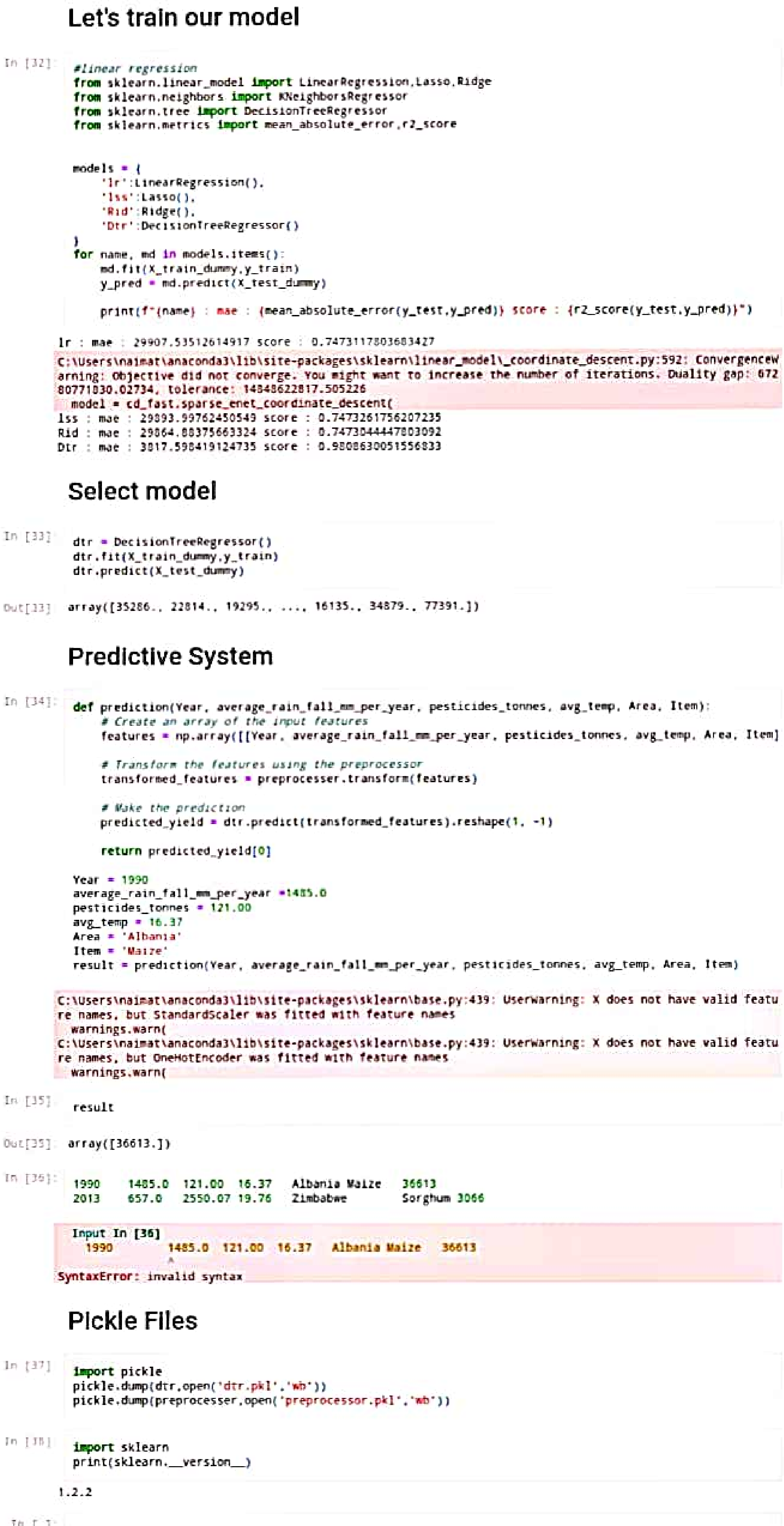
```
X_train_dummy = preprocesser.fit_transform(X_train)
X_test_dummy = preprocesser.transform(X_test)
```

In [31]:

```
preprocesser.get_feature_names_out(col[:-1])
```

Out[31]:

```
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',
```



Let's train our model

In [32]:

```
#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 : 29907.53512614917 score : 0.7473117803683427
C:\Users\naimat\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:592: ConvergenceW
arning: Objective did not converge. You might want to increase the number of iterations. Duality gap: 672
80771030.02734, tolerance: 14848622817.505226
  model = cd_fast.sparse_enet_coordinate_descent(
lss : mae : 29893.99762450549 score : 0.7473261756207235
Rid : mae : 29864.88375663324 score : 0.7473044447803092
Dtr : mae : 3817.598419124735 score : 0.9808630051556833
```

Select model

In [33]:

```
dtr = DecisionTreeRegressor()
dtr.fit(X_train_dummy,y_train)
dtr.predict(X_test_dummy)
```

Out[33]:

```
array([35286., 22814., 19295., ..., 16135., 34879., 77391.])
```

Predictive System

In [34]:

```
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]

    # 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\naimat\anaconda3\lib\site-packages\sklearn\base.py:439: UserWarning: X does not have valid featu
re names, but StandardScaler was fitted with feature names
  warnings.warn(
C:\Users\naimat\anaconda3\lib\site-packages\sklearn\base.py:439: UserWarning: X does not have valid featu
re names, but OneHotEncoder was fitted with feature names
  warnings.warn(
```

In [35]:

```
result
```

Out[35]:

```
array([36613.])
```

In [36]:

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

```
  Input In [36]
    1990        1485.0  121.00  16.37   Albania Maize   36613
                ^
SyntaxError: invalid syntax
```

Pickle Files

In [37]:

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

In [38]:

```
import sklearn
print(sklearn.__version__)
```

```
1.2.2
```

In [ ]: