AI BASED SMART FARMING SYSTEM IN AGRICULTURE

TEAM 8

MEMBERS

MALYALA SREE CHANDANA (221CV130)

MEDHINEE PADORTHY(221EC130)

ADITI PAGARIYA (221CV202)

RITIKA DWIWEDI (221CV138)

INTRODUCTION:

An AI-based smart farming system in agriculture leverages advanced technologies such as artificial intelligence, machine learning, Internet of Things (IoT), and data analytics to revolutionize traditional farming practices. Here's an introduction to how it works:

Data Collection: IoT sensors, drones, satellites, and other devices collect vast amounts of data from the farm environment. This data includes information on soil moisture, temperature, humidity, crop health, weather conditions, and more.

Data Analysis: Artificial intelligence algorithms analyze the collected data to extract valuable insights and patterns. Machine learning techniques are often used to train models that can predict crop yields, detect diseases, optimize irrigation schedules, and recommend appropriate actions for farmers.

Decision Support: AI-based smart farming systems provide farmers with real-time insights and actionable recommendations to optimize their farming practices. For example, they can advise on when to plant, irrigate, fertilize, and harvest crops, as well as which areas of the farm require attention.

Precision Farming: Precision farming techniques, enabled by AI, allow for targeted interventions at the plant level. This includes variable rate application of inputs such as water, fertilizer, and pesticides based on specific crop needs, leading to increased efficiency and reduced environmental impact.

Automation: AI-powered robotics and autonomous vehicles are increasingly being used in agriculture for tasks such as planting, weeding, spraying, and harvesting. These technologies improve productivity, reduce labor costs, and enable round-the-clock operations.

Remote Monitoring and Management: Farmers can remotely monitor and manage their farms using mobile applications and web-based dashboards. They can receive alerts about potential issues, track crop growth in real time, and make data-driven decisions from anywhere with an internet connection.

Sustainability and Resource Efficiency: AI-based smart farming systems promote sustainable agriculture by optimizing resource use and minimizing waste. By precisely controlling inputs and reducing reliance on chemical treatments, they help conserve water, soil, and energy while promoting biodiversity and environmental health.

Overall, AI-based smart farming systems hold the promise of transforming agriculture into a more efficient, sustainable, and resilient industry, capable of meeting the challenges of feeding a growing global population while mitigating the impacts of climate change.

OBJECTIVES:

1.Crop Recommendation:

Crop recommendation is a process where farmers are provided with suggestions on which crops to cultivate based on various factors such as soil type, climate, available resources, market demand, and the farmer's preferences and constraints. The objective of crop recommendation is to optimize crop productivity, maximize profitability, and ensure sustainable agricultural practices by matching the characteristics of the land and the farmer's capabilities with the most suitable crops. This can involve the use of technologies such as data analytics, machine learning, remote sensing, and expert knowledge to provide personalized recommendations tailored to specific farming conditions. Ultimately, the goal is to help farmers make informed decisions that lead to improved yields, reduced risks, and increased economic returns.

2.Yield Prediction:

Yield prediction is an objective commonly employed in agriculture and finance. In agriculture, it refers to estimating the amount of crops or produce that will be harvested from a given area of land. This prediction helps farmers plan for things like resource allocation, pricing, and overall management of their farms. In finance, yield prediction often refers to estimating the expected return on an investment, such as stocks, bonds, or other financial instruments. Accurate yield prediction in both contexts involves analyzing historical data, current conditions, and relevant factors such as weather patterns, market trends, and agricultural practices to make informed projections about future yields or returns.

LITERATURE REVIEW:

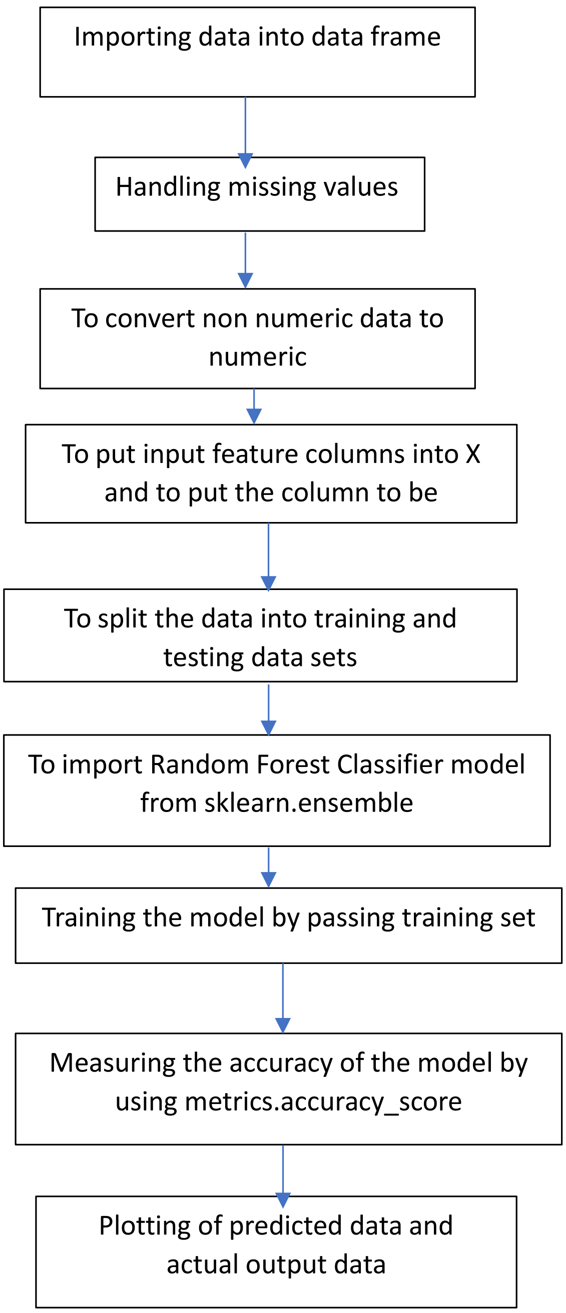
Table 1
Crop selection articles and sources.

Title	Author	Description	Source	Methods	Parameters
Future crop prediction	Praveen and Sharma (2020)	By considering climate variability prediction, future crop production	Journal of Public Affairs	autoregressive integrated moving average method (ARIMA)	Climate factors like rainfall, temperature
Corn and Soybean Crops yield prediction	Iniyan and Jebakumar (2022)	By using various phenotype factors predicting crop yield prediction for corn and soybean	Wireless Personal Communications	Multilayer Stacked Ensemble Regression (MSER)	maximum temperature, precipitation, minimum temperature, solar radiation
Crop prediction	Suruliandi and Raja (2021)	Environmental conditions also play an important role in crop production. By using different machine learning algorithms, predicting crop selection	Taylor and Francis	Recursive feature elimination with bagging results best accuracy algorithm. When compared with other algorithms	N, K, pH, Sulfur, Zinc, and many other factors
Maize yield prediction	Mupangwa et al. (2020)	Maize yield prediction in Eastern South Africa(ESA) by using different machine learning techniques	Springer	Linear discriminant analysis, Logistic regression, KNN, SVM	Low and high potential conditions of ESA regions
Crop yield prediction	van Klompenburg et al. (2020)	Analysis of various machine learning techniques for crop yield prediction	Elsevier	Artificial Neural Networks(ANN), Convolutional Neural Networks (CNN)	temperature, rainfall, and soil type
Crop type identification	Feng et al. (2019)	Crop type identification by using machine learning algorithms	IEEE	Random forest, support vector machines	Sentinel-2A images
Data management and IoT	Ouafiq et al. (2022b)	Explore the importance of big data and IoT in smart farming.	Agriculture	Big data with smart farming	Investigating IoT importance
Smart farming with IoT	Ouafiq et al. (2022a)	Implementing smart farming without impacting computing performance by using big data	Elsevier	Big data and IoT	Nitrogen, temperature, soil conditions
Rice Leaf Detection using ML	Pallathadka et al. (2022)	Demonstrated the importance of computer vision in rice leaf detection	Elsevier	SVM, Naïve Bayes and CNN	images of infected leaf
Smart farming with IoT	Phasinam et al. (2022)	By using IoT monitoring smart farming and also results in using energy as effective as possible.	Journal of Food Quality	SVM, logistic regression, and random forest classifiers	water volume, season, soil fertility, water quality, temperature

METHODOLOGY:

1. CROP RECOMMENDATION:

FLOW CHART:



- **Problem Definition**: To recommend a suitable crop based on Temperature, Humidity, Rainfall, N, P, K content data in the soil.

- **Data Collection**: To obtain datasets from various sources. Our Dataset contains the parameters called Temperature, Humidity, Rainfall, N, P, K content and Crop name.
- **Data Preprocessing**: Clean and preprocess the data to handle missing values, outliers, and inconsistencies. This may involve data normalization, scaling, encoding categorical variables, and feature engineering to extract meaningful insights.
- **Feature Selection**: Identify the most relevant features that contribute to crop recommendation. Use techniques like correlation analysis, feature importance, and domain knowledge to select the most informative features.
- **Model Selection**: Choose appropriate machine learning or deep learning models for crop recommendation. Common approaches include decision trees, random forests, support vector machines, gradient boosting algorithms, neural networks, and hybrid models.
    - The model used in this project is Random Forest Classifier which has accuracy of 99.98% which is more than any other models which have been tested.
- **Model Training**: Train the selected models using the preprocessed data. Use techniques like cross-validation and hyperparameter tuning to optimize model performance and prevent overfitting.
- **Model Evaluation**: Evaluate the trained models using the predefined evaluation metrics. Compare the performance of different models to select the best-performing one.
- **Giving Input Data:** giving all involved parameters as input in the form of list and predicting the suitable crop based on the input given

Analyzing the Dataset:

[98]:
```
import numpy as np
import pandas as pd
```

[134]:
```
crop=pd.read_csv("Crop_recommendation.csv")
crop.head()
```

[134]:

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice

[135]:
```
crop.shape
```

[135]:
```
(2200, 8)
```

[136]:
```
crop.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2200 entries, 0 to 2199
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   N            2200 non-null   int64  
 1   P            2200 non-null   int64  
 2   K            2200 non-null   int64  
 3   temperature  2200 non-null   float64
 4   humidity     2200 non-null   float64
```

[137]:
```
crop.duplicated().sum()
crop.describe()
```

[137]:

	N	P	K	temperature	humidity	ph	rainfall
count	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000
mean	50.551818	53.362727	48.149091	25.616244	71.481779	6.469480	103.463655
std	36.917334	32.985883	50.647931	5.063749	22.263812	0.773938	54.958389
min	0.000000	5.000000	5.000000	8.825675	14.258040	3.504752	20.211267
25%	21.000000	28.000000	20.000000	22.769375	60.261953	5.971693	64.551686
50%	37.000000	51.000000	32.000000	25.598693	80.473146	6.425045	94.867624
75%	84.250000	68.000000	49.000000	28.561654	89.948771	6.923643	124.267508
max	140.000000	145.000000	205.000000	43.675493	99.981876	9.935091	298.560117

[138]:
```
crop['label']=crop['label'].astype('category')
crop['label']=crop['label'].cat.codes
crop.head()
```

[138]:

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	20
1	85	58	41	21.770462	80.319644	7.038096	226.655537	20
2	60	55	44	23.004459	82.320763	7.840207	263.964248	20
3	74	35	40	26.491096	80.158363	6.980401	242.864034	20
4	78	42	42	20.130175	81.604873	7.628473	262.717340	20

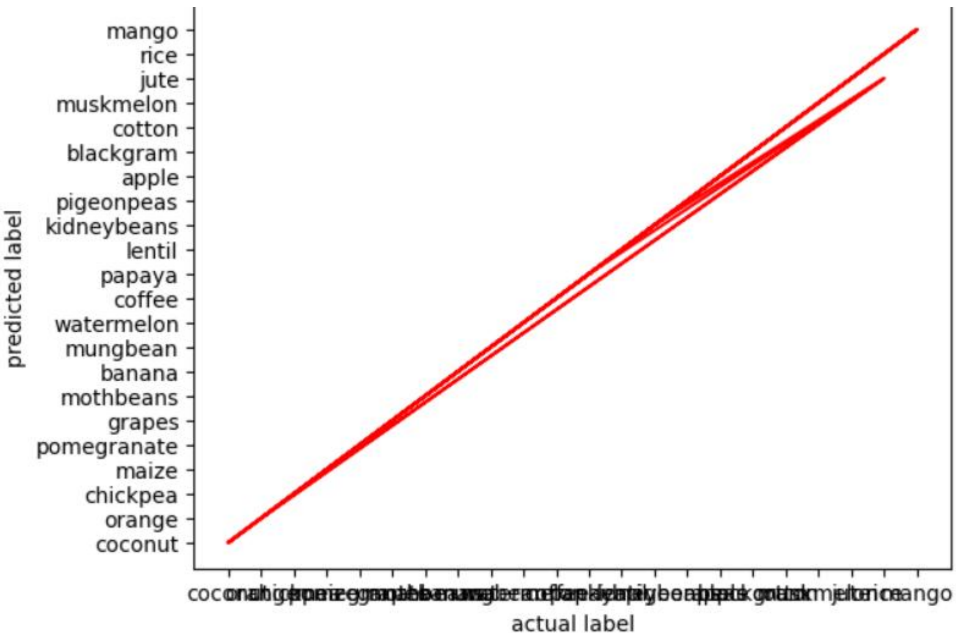
[105]:
```
crop.isnull().sum()
```

[105]:
```
N              0
P              0
K              0
temperature    0
humidity       0
ph             0
rainfall       0
label          0
dtype: int64
```

[139]:
```
crop['label'].value_counts()
```

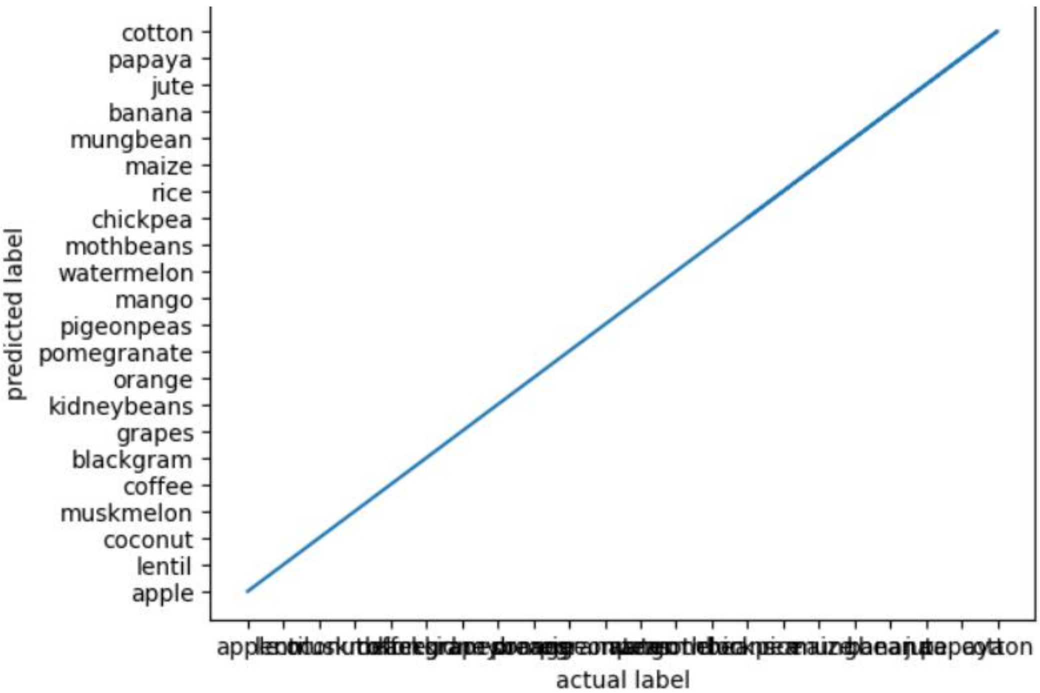
[139]:
```
label
20    100
11    100
8     100
6     100
4     100
17    100
16    100
0     100
15    100
21    100
7     100
12    100
1     100
19    100
10    100
2     100
14    100
13    100
18    100
9     100
3     100
5     100
```

GRAPHS:



The actual label tells what are the actual values present in testing data set

The predicted label tells what values are obtained for those samples in testing data set.



The actual label tells what are the actual values present in training data set

The predicted label tells what values are obtained for those samples in training data set.

Model Selection based on Accuracy:

[67]:
```python
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import ExtraTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score

# create instances of all models
models = {
    'Logistic Regression': LogisticRegression(),
    'Naive Bayes': GaussianNB(),
    'Support Vector Machine': SVC(),
    'K-Nearest Neighbors': KNeighborsClassifier(),
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(),
    'Bagging': BaggingClassifier(),
    'AdaBoost': AdaBoostClassifier(),
    'Gradient Boosting': GradientBoostingClassifier(),
    'Extra Trees': ExtraTreeClassifier(),
}
for name, md in models.items():
    md.fit(X_train,Y_train)
    ypred = md.predict(X_test)

    print(f"{name}  with accuracy : {accuracy_score(Y_test,ypred)}")
```

```
Logistic Regression  with accuracy : 0.95
Naive Bayes  with accuracy : 0.990909090909091
Support Vector Machine  with accuracy : 0.9681818181818181
K-Nearest Neighbors  with accuracy : 0.9818181818181818
Decision Tree  with accuracy : 0.9818181818181818
Random Forest  with accuracy : 0.9863636363636363
Bagging  with accuracy : 0.990909090909091
AdaBoost  with accuracy : 0.09545454545454546
Gradient Boosting  with accuracy : 0.9954545454545455
Extra Trees  with accuracy : 0.9045454545454545
```

Main Code in Python:

```
[54]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df=pd.read_csv("crop_recommendation.csv")

#sizes=df['label'].value_counts(sort=1)
#print(sizes)

#to drop irrelevant columns
#df.drop(['temperature'],axis=1,inplace=True)
#df.drop(['ph'],axis=1,inplace=True)

#handle missing values
#df=df.dropna()

#to convert nonnumeric data to numeric
df['label']=df['label'].astype('category')

Y=df['label'].values
X=df.drop(labels=['label'],axis=1)

from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test =train_test_split(X,Y,test_size=0.4,random_state=20)

from sklearn.ensemble import RandomForestClassifier
model=RandomForestClassifier(n_estimators=2,max_depth=19,min_samples_split=5,min_samples_leaf=1,random_state=30)

model.fit(X_train,Y_train)
prediction_test=model.predict(X_test)
print(prediction_test)

from sklearn import metrics
print("Accuracy= ",metrics.accuracy_score(Y_test,prediction_test))
```

```
from sklearn import metrics
print("Accuracy= ",metrics.accuracy_score(Y_test,prediction_test))

feature_list=list(X.columns)
feature_imp=pd.Series(model.feature_importances_,index=feature_list).sort_values(ascending=False)
print(feature_imp)
```

```
['coconut' 'pomegranate' 'chickpea' 'chickpea' 'maize' 'pomegranate'
 'chickpea' 'coconut' 'grapes' 'mothbeans' 'banana' 'maize' 'mungbean'
 'watermelon' 'coffee' 'papaya' 'lentil' 'banana' 'mungbean' 'chickpea'
 'grapes' 'lentil' 'kidneybeans' 'lentil' 'pigeonpeas' 'watermelon'
 'apple' 'apple' 'mungbean' 'pigeonpeas' 'coffee' 'orange' 'coconut'
 'mungbean' 'mungbean' 'watermelon' 'blackgram' 'kidneybeans' 'cotton'
 'coconut' 'blackgram' 'cotton' 'chickpea' 'papaya' 'muskmelon' 'coffee'
 'kidneybeans' 'jute' 'jute' 'chickpea' 'papaya' 'cotton' 'cotton'
 'pigeonpeas' 'mungbean' 'apple' 'watermelon' 'papaya' 'mango' 'orange'
 'jute' 'kidneybeans' 'banana' 'jute' 'lentil' 'coconut' 'mungbean'
 'papaya' 'lentil' 'blackgram' 'lentil' 'rice' 'cotton' 'pomegranate'
 'orange' 'mothbeans' 'mothbeans' 'watermelon' 'kidneybeans' 'lentil'
 'kidneybeans' 'mango' 'chickpea' 'cotton' 'watermelon' 'jute' 'muskmelon'
 'papaya' 'jute' 'orange' 'lentil' 'banana' 'pigeonpeas' 'mungbean'
 'pomegranate' 'coconut' 'grapes' 'orange' 'blackgram' 'jute' 'mothbeans'
 'kidneybeans' 'cotton' 'chickpea' 'mothbeans' 'lentil' 'watermelon'
 'coconut' 'pigeonpeas' 'watermelon' 'apple' 'blackgram' 'mango'
 'mothbeans' 'maize' 'lentil' 'lentil' 'pigeonpeas' 'lentil' 'jute'
```

```
Accuracy=  0.9397727272727273
humidity       0.223602
rainfall       0.213684
P              0.200829
K              0.114123
ph             0.088411
N              0.083267
temperature    0.076084
dtype: float64
```

Output:

```
[55]: sample=[[3,64,67,48.93,98.567,122.45,98]]
      output=model.predict(sample)
      print(output)

      ['papaya']
```

After training the model, in order to get the output based on given specifications, we are creating a sample.

In this sample, different values of features are given. This sample is then given to model and the output is predicted by using model.predict.

In above code,the model gives papaya as output

2.YIELD PREDICTION:

YEILD PREDICTION CODE-

[424]:

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

```
[425]: df.shape

[425]: (28242, 7)

[426]: df.info()

<class 'pandas.core.frame.DataFrame'>
Index: 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: 1.7+ MB
```

```
[427]: df.isnull().sum()
```

```
[427]: 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
```

```
[428]: df.duplicated().sum()
```

```
[428]: 2310
```

```
[429]: df.drop_duplicates(inplace=True)
```

```
[430]: df.duplicated().sum()
```

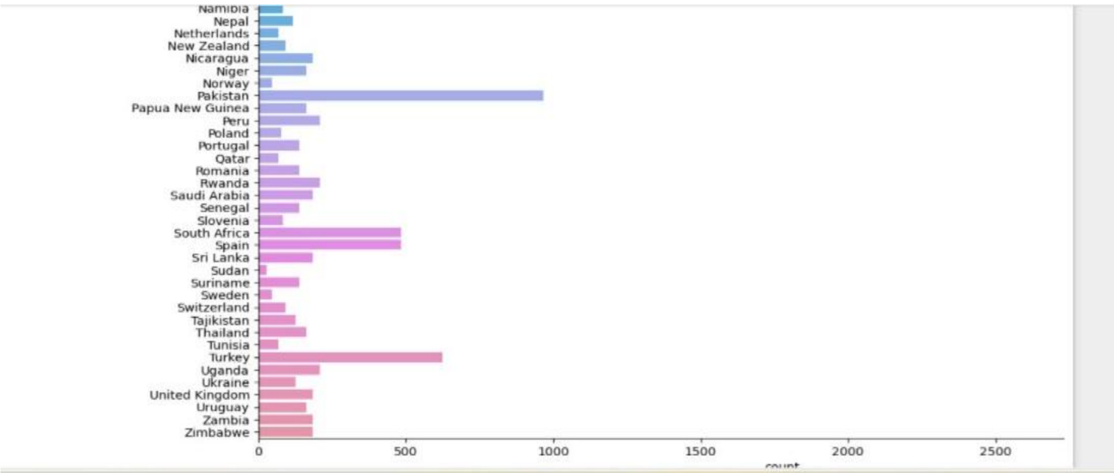
```
[430]: 0
```

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

Frequency vs Area Graph

```
[435]: 101
```

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



[437]:
```
(df['Area'].value_counts() < 500).sum()
```

[437]: 91

[438]:
```
country = df['Area'].unique()
yield_per_country = []
for state in country:
    yield_per_country.append(df[df['Area']==state]['hg/ha_yield'].sum())
```

[439]:
```
df['hg/ha_yield'].sum()
```

[439]: 1996196943

[440]:
```
yield_per_country
```

[440]:
```
[5711536,
 6711464,
 5722563,
 32864032,
 4524100,
 109111062,
 10852258,
 4608380,
 4384717,
 4443889,
 7720159,
 4704812,
 8442270,
 470651,
```

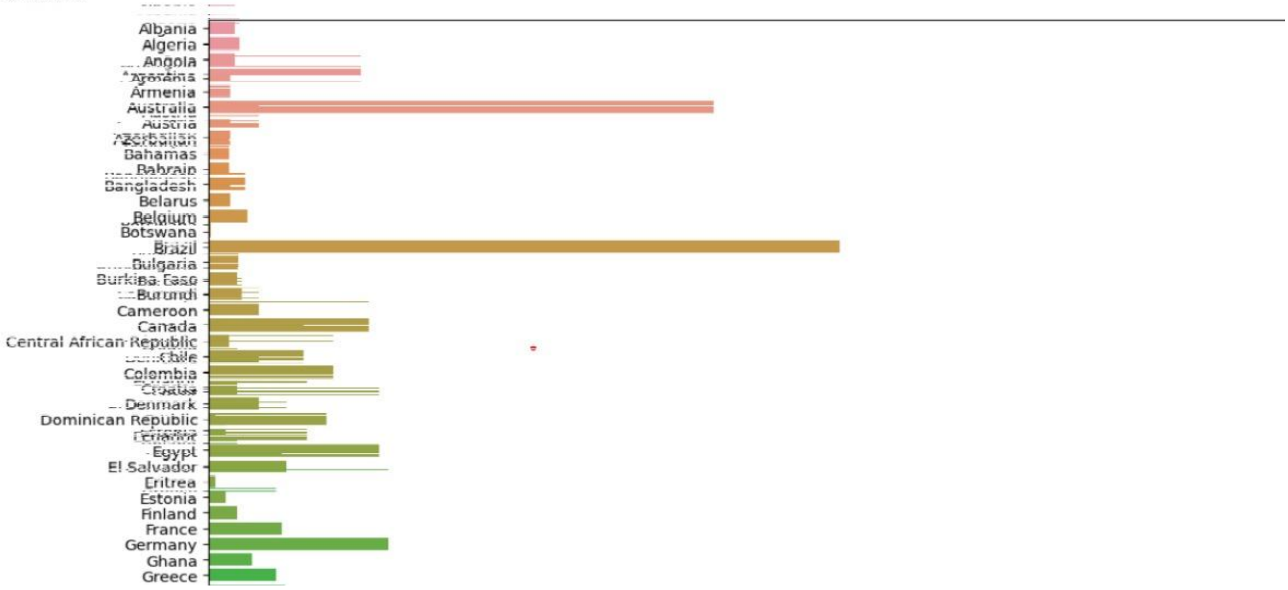
```
 49264956,
 [illegible],
 [illegible],
 6564711,
 [illegible],
 5995626,
 6006156,
 7741053,
 15171886,
 10542077,
 7197013,
 30999849,
 [illegible],
 11217741,
 1896346,
 14786468,
 8620653,
 [illegible],
 6295210,
 9511720,
 3724246,
 52263950,
 7494314,
 5496901,
 44335992,
 9539820,
 7254311,
 7408629]
```

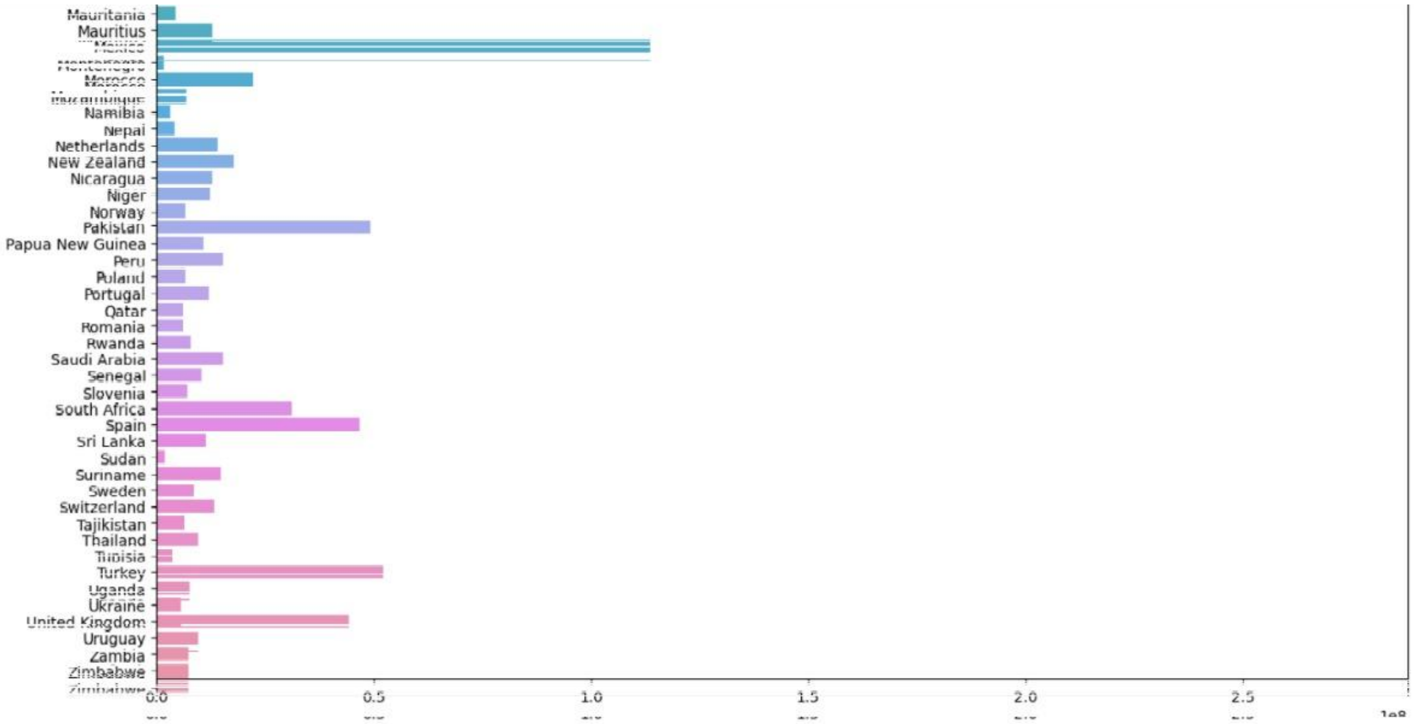
[441]:
```
plt.figure(figsize=(15, 20))
sns.barplot(y=country, x=yield_per_country)
```

Yield Per Country Graph

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

```
[441]: <Axes: >
```

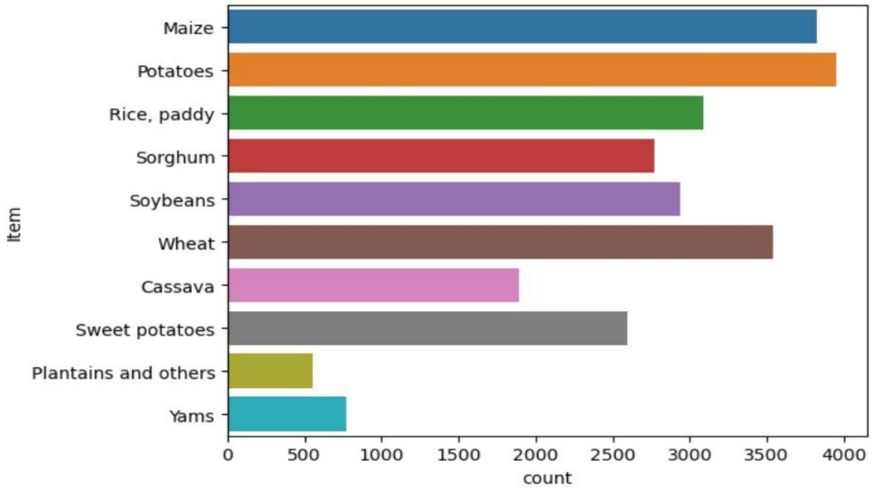




Graph- Frequency vs Item

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

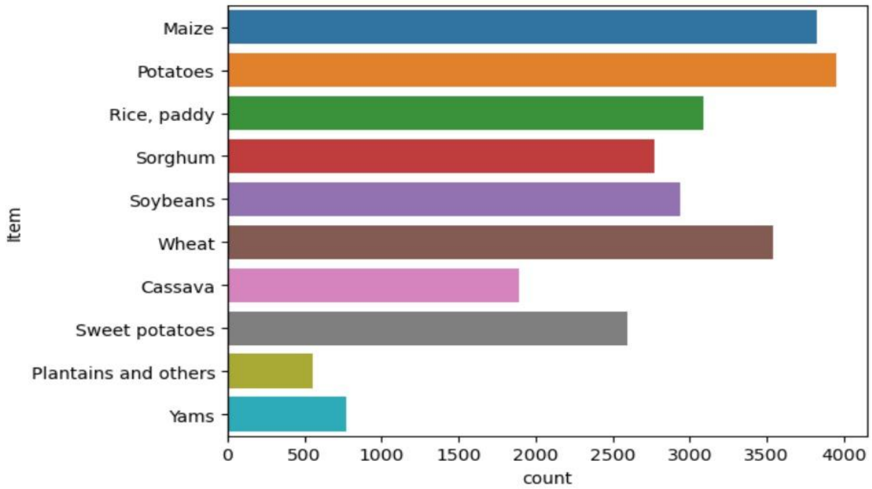
```
[442]: <Axes: xlabel='count', ylabel='Item'>
```




```
[443]: crops = df['Item'].unique()
        yield_per_crop = []
```


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

```
[442]: <Axes: xlabel='count', ylabel='Item'>
```

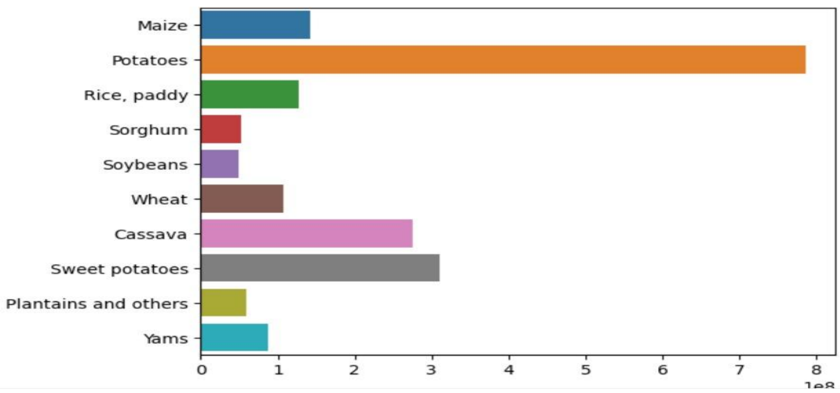

```
[443]: crops = df['Item'].unique()
        yield_per_crop = []
```


Yield Vs Item Graph

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

```
[444]: sns.barplot(y=crops,x=yield_per_crop)
```

```
[444]: <Axes: >
```



```
[445]: 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]
```

```
[446]: df.head(3)
```

[446]:

	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

```
[447]: 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)
```

```
[448]: 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'
```

```
[449]: X_train_dummy = preprocesser.fit_transform(X_train)
       X_test_dummy = preprocesser.transform(X_test)
```

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

```
[450]: 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_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_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)
```

```
[451]: #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 : 29907.509779056007 score : 0.7473129744324709
C:\Users\Aditi\anaconda3\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:592: ConvergenceWarning: Objective did not converge. You might wan
t to increase the number of iterations, check the scale of the features or consider increasing regularisation.
  model = cd_fast.sparse_enet_coordinate_descent(
```

```
[452]: dtr = DecisionTreeRegressor()
       dtr.fit(X_train_dummy,y_train)
       dtr.predict(X_test_dummy)

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

```
[453]: 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 preprocesser
           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 = 161.00
       avg_temp = 16.37
       Area = 'Albania'
```

```
Year = 1990
average_rain_fall_mm_per_year =1485.0
pesticides_tonnes = 161.00
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\Aditi\anaconda3\Lib\site-packages\sklearn\base.py:439: UserWarning: X does not have valid feature names, but StandardScaler was fitted with feat
ure names
  warnings.warn(
C:\Users\Aditi\anaconda3\Lib\site-packages\sklearn\base.py:439: UserWarning: X does not have valid feature names, but OneHotEncoder was fitted with featu
re names
  warnings.warn(

result

array([36613.])
```

```
[449]: X_train_dummy = preprocesser.fit_transform(X_train)
       X_test_dummy = preprocesser.transform(X_test)

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

[450]: 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)
```

```
[451]: #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():
```

INPUT & RESULT

```
Year = 1990
average_rain_fall_mm_per_year =1485.0
pesticides_tonnes = 161.00
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\Aditi\anaconda3\Lib\site-packages\sklearn\base.py:439: UserWarning: X does not have valid feature names, but StandardScaler was fitted with feat
ure names
  warnings.warn(
C:\Users\Aditi\anaconda3\Lib\site-packages\sklearn\base.py:439: UserWarning: X does not have valid feature names, but OneHotEncoder was fitted with featu
re names
  warnings.warn(
```

```
result
```

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

Algorithms Description

In conclusion, our exploration examined four machine learning algorithms for crop yield prediction:

Linear Regression: Provides a foundational approach. It establishes a straight-line relationship to predict yield based on input factors like weather data. While easy to understand and interpret, it struggles with complex, non-linear relationships common in agricultural data.

Decision Trees: Excel at capturing these non-linear relationships. They create a tree-like structure that splits data based on various factors, ultimately predicting yield based on the most relevant conditions. This flexibility makes them a strong choice for accurate crop yield prediction.

Lasso Regression: Builds upon linear regression by adding a penalty term that shrinks less important feature coefficients towards zero. This helps address issues with irrelevant features that might otherwise affect the model.

Ridge Regression: Another technique based on linear regression. It introduces a penalty term that shrinks all feature coefficients towards zero, reducing the model's complexity and mitigating overfitting.

While Lasso and Ridge regression offer improvements over linear regression, decision trees remain the preferred choice for accurate crop yield prediction due to their ability to handle the intricate relationships within agricultural data. By leveraging decision trees within an AI system, we can empower farmers with a powerful tool for informed decision-making and improved agricultural outcomes.

AI for Crop Yield Prediction: Impact and the Road Ahead

IMPORTANCE:

Enhanced Food Security: AI-powered prediction allows for optimized resource allocation, leading to increased yields, reduced waste, and ultimately contributing to global food security.

Empowered Farmers: Farmers gain valuable insights into potential yield outcomes, enabling data-driven decisions that can improve profitability and mitigate risks.

Sustainable Practices: AI systems promote sustainable agriculture by optimizing resource use and identifying areas for improvement, minimizing environmental impact while maximizing yields.

Early Warning Systems: Accurate forecasts can trigger early intervention for potential food shortages, allowing governments and humanitarian organizations to prepare mitigation strategies.

Market Stability: More accurate yield forecasts can inform market decisions, potentially smoothing out price fluctuations and benefiting both farmers and consumers.

Future Research:

- **Data Integration:** Incorporating data from diverse sources like remote sensing, soil moisture sensors, and real-time weather monitoring can further enhance prediction accuracy.
- **Advanced Techniques:** Exploring deep learning algorithms could potentially capture even more complex relationships within agricultural data.
- **Climate Adaptation:** Building models that factor in climate change scenarios can help farmers prepare for changing weather patterns and mitigate potential yield losses.
- **Precision Agriculture:** Integrating prediction systems with variable-rate technology can allow for targeted application of resources, optimizing inputs based on specific field conditions.
- **Explainable AI:** Developing interpretable models can empower farmers to understand the reasoning behind the predictions and make more informed choices.

References:

https://techvidvan.com/tutorials/crop-yield-prediction-python-machine-learning/

https://www.researchgate.net/publication/360401702_Smart_Farming_Using_Artificial_Intelligence_the_Internet_of_Things_and_Robotics_A_Comprehensive_Review

https://www.sciencedirect.com/science/article/pii/S258972172030012X

https://www.sciencedirect.com/science/article/pii/S0952197623000830#:~:text=Smart%20farming%20with%20artificial%20intelligence,are%20essential%20in%20smart%20farming

https://techvidvan.com/tutorials/crop-yield-prediction-python-machine-learning/

https://www.researchgate.net/publication/360401702_Smart_Farming_Using_Artificial_Intelligence_the_Internet_of_Things_and_Robotics_A_Comprehensive_Review

https://www.sciencedirect.com/science/article/pii/S258972172030012X

https://www.sciencedirect.com/science/article/pii/S0952197623000830#:~:text=Smart%20farming%20with%20artificial%20intelligence,are%20essential%20in%20smart%20farming