1.Random Forest;

import pandas as pd

import numpy as np

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

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.preprocessing import LabelEncoder

# Load the data from the CSV file

file\_path = "/content/drive/MyDrive/crop\_yield.csv"

data = pd.read\_csv(file\_path)

# Print the first few rows to understand the structure of the dataset

print("Original Dataset:")

print(data.head())

# Check unique values in the Crop column

print("Unique values in Crop column:", data['Crop'].unique())

# Filter the dataset to include only rows where the crop type is 'rice'

rice\_data = data[data['Crop'].str.strip().str.lower() == 'rice'] # Ensure case insensitivity and strip spaces

# Print the filtered rows to confirm

if rice\_data.empty:

print("No data found for rice. Please check the Crop column for correct values.")

else:

print("Filtered Dataset (only rice data):")

print(rice\_data.head())

# Convert categorical columns to numerical using LabelEncoder

label\_encoder = LabelEncoder()

# Encode each categorical column in the filtered dataframe

for column in rice\_data.select\_dtypes(include=['object']).columns:

rice\_data[column] = label\_encoder.fit\_transform(rice\_data[column].astype(str).str.strip())

# Define features (X) and target (y) from the filtered data

X = rice\_data.drop('Yield', axis=1) # Features (adjust column names if necessary)

y = rice\_data['Yield'] # Target (yield)

# Split the dataset into training and testing sets

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

# Train a Random Forest model

rf\_model = RandomForestRegressor(n\_estimators=100, random\_state=42) # You can adjust the number of estimators

rf\_model.fit(X\_train, y\_train)

# Predict on test data

y\_pred = rf\_model.predict(X\_test)

# Evaluate the model using Mean Squared Error (MSE) and R-squared (R²)

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

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

# Print evaluation metrics

print(f"Random Forest Mean Squared Error (MSE): {mse:.4f}")

print(f"Random Forest R-squared (R²): {r2:.4f}")

# Calculate accuracy of predictions based on actual yield

accuracy = 100 \* (1 - (np.abs(y\_test - y\_pred) / y\_test))

accuracy\_df = pd.DataFrame({

'Actual Yield': y\_test,

'Predicted Yield': y\_pred,

'Accuracy': accuracy

})

# Select top 10 predictions based on actual yield

top\_10\_predictions = accuracy\_df.nlargest(10, 'Actual Yield')

print("Top 10 Yield Predictions:")

print(top\_10\_predictions)

OUTPUT:

Original Dataset:

Crop Crop\_Year Season State Area Production \

0 Arecanut 1997 Whole Year Assam 73814.0 56708

1 Arhar/Tur 1997 Kharif Assam 6637.0 4685

2 Castor seed 1997 Kharif Assam 796.0 22

3 Coconut 1997 Whole Year Assam 19656.0 126905000

4 Cotton(lint) 1997 Kharif Assam 1739.0 794

Annual\_Rainfall Fertilizer Pesticide Yield

0 2051.4 7024878.38 22882.34 0.796087

1 2051.4 631643.29 2057.47 0.710435

2 2051.4 75755.32 246.76 0.238333

3 2051.4 1870661.52 6093.36 5238.051739

4 2051.4 165500.63 539.09 0.420909

Unique values in Crop column: ['Arecanut' 'Arhar/Tur' 'Castor seed' 'Coconut ' 'Cotton(lint)'

'Dry chillies' 'Gram' 'Jute' 'Linseed' 'Maize' 'Mesta' 'Niger seed'

'Onion' 'Other Rabi pulses' 'Potato' 'Rapeseed &Mustard' 'Rice'

'Sesamum' 'Small millets' 'Sugarcane' 'Sweet potato' 'Tapioca' 'Tobacco'

'Turmeric' 'Wheat' 'Bajra' 'Black pepper' 'Cardamom' 'Coriander' 'Garlic'

'Ginger' 'Groundnut' 'Horse-gram' 'Jowar' 'Ragi' 'Cashewnut' 'Banana'

'Soyabean' 'Barley' 'Khesari' 'Masoor' 'Moong(Green Gram)'

'Other Kharif pulses' 'Safflower' 'Sannhamp' 'Sunflower' 'Urad'

'Peas & beans (Pulses)' 'other oilseeds' 'Other Cereals' 'Cowpea(Lobia)'

'Oilseeds total' 'Guar seed' 'Other Summer Pulses' 'Moth']

Filtered Dataset (only rice data):

Crop Crop\_Year Season State Area Production \

16 Rice 1997 Autumn Assam 607358.0 398311

17 Rice 1997 Summer Assam 174974.0 209623

18 Rice 1997 Winter Assam 1743321.0 1647296

51 Rice 1997 Kharif Karnataka 1031530.0 2340493

52 Rice 1997 Rabi Karnataka 53889.0 109350

Annual\_Rainfall Fertilizer Pesticide Yield

16 2051.4 5.780226e+07 188280.98 0.780870

17 2051.4 1.665228e+07 54241.94 1.060435

18 2051.4 1.659119e+08 540429.51 0.941304

51 1266.7 9.817071e+07 319774.30 2.233500

52 1266.7 5.128616e+06 16705.59 2.073846

<ipython-input-9-ae3f5e73134e>:34: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation:<https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

rice\_data[column] = label\_encoder.fit\_transform(rice\_data[column].astype(str).str.strip())

<ipython-input-9-ae3f5e73134e>:34: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation:<https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

rice\_data[column] = label\_encoder.fit\_transform(rice\_data[column].astype(str).str.strip())

<ipython-input-9-ae3f5e73134e>:34: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation:<https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

rice\_data[column] = label\_encoder.fit\_transform(rice\_data[column].astype(str).str.strip())

Random Forest Mean Squared Error (MSE): 0.2717

Random Forest R-squared (R²): 0.6052

Top 10 Yield Predictions:

Actual Yield Predicted Yield Accuracy

6266 8.778276 1.681338 79.153401

4394 4.730323 3.996572 84.488366

5745 4.184828 3.863784 92.328394

5746 3.997333 4.002210 99.878008

14529 3.923182 3.993441 98.209130

11788 3.834000 3.840115 99.840502

8789 3.660000 2.830348 87.331921

17139 3.610625 3.728431 96.737227

15184 3.585000 3.257414 90.862307

17138 3.575625 3.699482 96.536073

**2.Naive Bayes Algorithm :**

import pandas as pd

import numpy as np

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

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score, confusion\_matrix

from sklearn.preprocessing import LabelEncoder

# Load the data from the CSV file

file\_path = "/content/drive/MyDrive/crop\_yield.csv"

data = pd.read\_csv(file\_path)

# Print the first few rows to understand the structure of the dataset

print(data.head())

# Convert categorical columns to numerical using LabelEncoder

label\_encoder = LabelEncoder()

# Assuming 'Crop' is a categorical column, and 'Yield' is the target variable

# You need to adjust column names based on your dataset

for column in data.columns:

if data[column].dtype == 'object': # Check if the column is categorical

data[column] = label\_encoder.fit\_transform(data[column])

# Assuming 'Yield' is continuous, let's categorize it into classes (Low, Medium, High)

# You can define custom bins based on your data's distribution

data['Yield\_category'] = pd.cut(data['Yield'], bins=3, labels=['Low', 'Medium', 'High'])

# Define features (X) and target (y) with categorized yield

X = data.drop(['Yield', 'Yield\_category'], axis=1) # Features (adjust if necessary)

y = data['Yield\_category'] # Target (categorized yield)

# Split the dataset into training and testing sets

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

# Train a Naive Bayes model (GaussianNB)

nb\_model = GaussianNB()

nb\_model.fit(X\_train, y\_train)

# Predict on test data

y\_pred = nb\_model.predict(X\_test)

# Evaluate the model using accuracy score

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

print(f"Naive Bayes Classification Accuracy: {accuracy:.4f}")

# Confusion matrix to understand the classification results

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

print("Confusion Matrix:")

print(conf\_matrix)

# Perform iterations to observe model performance over multiple runs

num\_iterations = 10

accuracy\_scores = []

for iteration in range(num\_iterations):

nb\_model = GaussianNB()

nb\_model.fit(X\_train, y\_train)

y\_pred = nb\_model.predict(X\_test)

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

accuracy\_scores.append(accuracy)

print("Accuracy scores for 10 iterations (Naive Bayes Classifier):")

for i, score in enumerate(accuracy\_scores, 1):

print(f"Iteration {i}: {score:.4f}")

OUTPUT:

Crop Crop\_Year Season State Area Production \

0 Arecanut 1997 Whole Year Assam 73814.0 56708

1 Arhar/Tur 1997 Kharif Assam 6637.0 4685

2 Castor seed 1997 Kharif Assam 796.0 22

3 Coconut 1997 Whole Year Assam 19656.0 126905000

4 Cotton(lint) 1997 Kharif Assam 1739.0 794

Annual\_Rainfall Fertilizer Pesticide Yield

0 2051.4 7024878.38 22882.34 0.796087

1 2051.4 631643.29 2057.47 0.710435

2 2051.4 75755.32 246.76 0.238333

3 2051.4 1870661.52 6093.36 5238.051739

4 2051.4 165500.63 539.09 0.420909

Naive Bayes Classification Accuracy: 0.9964

Confusion Matrix:

[[ 0 1 1]

[ 0 3915 5]

[ 0 7 9]]

Accuracy scores for 10 iterations (Naive Bayes Classifier):

Iteration 1: 0.9232

Iteration 2: 0.9147

Iteration 3: 0.8972

Iteration 4: 0.9297

Iteration 5: 0.9465

Iteration 6: 0.9631

Iteration 7: 0.9277

Iteration 8: 0.9582

Iteration 9: 0.9612

Iteration 10: 0.9468

**3.Linear Regression :**

import pandas as pd

import numpy as np

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

from sklearn.linear\_model import LinearRegression # Import Linear Regression

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.preprocessing import LabelEncoder

# Load the data from the CSV file

file\_path = "/content/drive/MyDrive/crop\_yield.csv"

data = pd.read\_csv(file\_path)

# Print the first few rows to understand the structure of the dataset

print(data.head())

# Filter the data to only include rows where the Crop is "Rice"

rice\_data = data[data['Crop'] == 'Rice']

# Convert categorical columns to numerical using LabelEncoder

label\_encoder = LabelEncoder()

# Encode each categorical column in the dataframe

for column in rice\_data.select\_dtypes(include=['object']).columns:

rice\_data[column] = label\_encoder.fit\_transform(rice\_data[column].astype(str).str.strip())

# Define features (X) and target (y)

X = rice\_data.drop('Yield', axis=1) # Features (adjust column names if necessary)

y = rice\_data['Yield'] # Target (yield)

# Split the dataset into training and testing sets

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

# Train a Linear Regression model

lr\_model = LinearRegression()

lr\_model.fit(X\_train, y\_train)

# Predict on test data

y\_pred = lr\_model.predict(X\_test)

# Evaluate the model using Mean Squared Error (MSE) and R-squared (R²)

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

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

# Print evaluation metrics

print(f"Linear Regression Mean Squared Error (MSE): {mse:.4f}")

print(f"Linear Regression R-squared (R²): {r2:.4f}")

# Calculate accuracy of predictions based on actual yield

accuracy = 100 \* (1 - (np.abs(y\_test - y\_pred) / y\_test))

accuracy\_df = pd.DataFrame({

'Actual Yield': y\_test,

'Predicted Yield': y\_pred,

'Accuracy': accuracy

})

# Select top 10 predictions based on actual yield

top\_10\_predictions = accuracy\_df.nlargest(10, 'Actual Yield')

print("Top 10 Yield Predictions:")

print(top\_10\_predictions)

OUTPUT:

Crop Crop\_Year Season State Area Production \

0 Arecanut 1997 Whole Year Assam 73814.0 56708

1 Arhar/Tur 1997 Kharif Assam 6637.0 4685

2 Castor seed 1997 Kharif Assam 796.0 22

3 Coconut 1997 Whole Year Assam 19656.0 126905000

4 Cotton(lint) 1997 Kharif Assam 1739.0 794

Annual\_Rainfall Fertilizer Pesticide Yield

0 2051.4 7024878.38 22882.34 0.796087

1 2051.4 631643.29 2057.47 0.710435

2 2051.4 75755.32 246.76 0.238333

3 2051.4 1870661.52 6093.36 5238.051739

4 2051.4 165500.63 539.09 0.420909

Linear Regression Mean Squared Error (MSE): 0.4498

Linear Regression R-squared (R²): 0.3465

Top 10 Yield Predictions:

Actual Yield Predicted Yield Accuracy

6266 8.778276 1.723782 19.636909

4394 4.730323 3.713949 78.513649

5745 4.184828 2.594818 62.005370

5746 3.997333 2.832920 70.870237

14529 3.923182 4.094955 95.621590

11788 3.834000 3.713816 96.865302

8789 3.660000 2.302966 62.922565

17139 3.610625 3.328583 92.188562

15184 3.585000 3.031489 84.560362

17138 3.575625 3.227333 90.259261

<ipython-input-5-5699aba98df8>:23: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation:<https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

rice\_data[column] = label\_encoder.fit\_transform(rice\_data[column].astype(str).str.strip())

<ipython-input-5-5699aba98df8>:23: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation:<https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

rice\_data[column] = label\_encoder.fit\_transform(rice\_data[column].astype(str).str.strip())

<ipython-input-5-5699aba98df8>:23: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation:<https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

rice\_data[column] = label\_encoder.fit\_transform(rice\_data[column].astype(str).str.strip())

**4.DECISION TREE:**

**import pandas as pd**

**import numpy as np**

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

**from sklearn.tree import DecisionTreeRegressor**

**from sklearn.metrics import mean\_squared\_error, r2\_score**

**from sklearn.preprocessing import LabelEncoder**

**# Load the data from the CSV file**

**file\_path = "/content/drive/MyDrive/crop\_yield.csv"**

**data = pd.read\_csv(file\_path)**

**# Print the first few rows to understand the structure of the dataset**

**print("Original Dataset:")**

**print(data.head())**

**# Check unique values in the Crop column**

**print("Unique values in Crop column:", data['Crop'].unique())**

**# Filter the dataset to include only rows where the crop type is 'rice'**

**rice\_data = data[data['Crop'].str.strip().str.lower() == 'rice'] # Ensure case insensitivity and strip spaces**

**# Print the filtered rows to confirm**

**if rice\_data.empty:**

**print("No data found for rice. Please check the Crop column for correct values.")**

**else:**

**print("Filtered Dataset (only rice data):")**

**print(rice\_data.head())**

**# Convert categorical columns to numerical using LabelEncoder**

**label\_encoder = LabelEncoder()**

**# Encode each categorical column in the filtered dataframe**

**for column in rice\_data.select\_dtypes(include=['object']).columns:**

**rice\_data[column] = label\_encoder.fit\_transform(rice\_data[column].astype(str).str.strip())**

**# Define features (X) and target (y) from the filtered data**

**X = rice\_data.drop('Yield', axis=1) # Features (adjust column names if necessary)**

**y = rice\_data['Yield'] # Target (yield)**

**# Split the dataset into training and testing sets**

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

**# Train a Decision Tree model**

**dt\_model = DecisionTreeRegressor(random\_state=42)**

**dt\_model.fit(X\_train, y\_train)**

**# Predict on test data**

**y\_pred = dt\_model.predict(X\_test)**

**# Evaluate the model using Mean Squared Error (MSE) and R-squared (R²)**

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

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

**# Print evaluation metrics**

**print(f"Decision Tree Mean Squared Error (MSE): {mse:.4f}")**

**print(f"Decision Tree R-squared (R²): {r2:.4f}")**

**# Calculate accuracy of predictions based on actual yield**

**accuracy = 100 \* (1 - (np.abs(y\_test - y\_pred) / y\_test))**

**accuracy\_df = pd.DataFrame({**

**'Actual Yield': y\_test,**

**'Predicted Yield': y\_pred,**

**'Accuracy': accuracy**

**})**

**# Select top 10 predictions based on actual yield**

**top\_10\_predictions = accuracy\_df.nlargest(10, 'Actual Yield')**

**print("Top 10 Yield Predictions:")**

**print(top\_10\_predictions)**

OUTPUT:

Original Dataset:

Crop Crop\_Year Season State Area Production \

0 Arecanut 1997 Whole Year Assam 73814.0 56708

1 Arhar/Tur 1997 Kharif Assam 6637.0 4685

2 Castor seed 1997 Kharif Assam 796.0 22

3 Coconut 1997 Whole Year Assam 19656.0 126905000

4 Cotton(lint) 1997 Kharif Assam 1739.0 794

Annual\_Rainfall Fertilizer Pesticide Yield

0 2051.4 7024878.38 22882.34 0.796087

1 2051.4 631643.29 2057.47 0.710435

2 2051.4 75755.32 246.76 0.238333

3 2051.4 1870661.52 6093.36 5238.051739

4 2051.4 165500.63 539.09 0.420909

Unique values in Crop column: ['Arecanut' 'Arhar/Tur' 'Castor seed' 'Coconut ' 'Cotton(lint)'

'Dry chillies' 'Gram' 'Jute' 'Linseed' 'Maize' 'Mesta' 'Niger seed'

'Onion' 'Other Rabi pulses' 'Potato' 'Rapeseed &Mustard' 'Rice'

'Sesamum' 'Small millets' 'Sugarcane' 'Sweet potato' 'Tapioca' 'Tobacco'

'Turmeric' 'Wheat' 'Bajra' 'Black pepper' 'Cardamom' 'Coriander' 'Garlic'

'Ginger' 'Groundnut' 'Horse-gram' 'Jowar' 'Ragi' 'Cashewnut' 'Banana'

'Soyabean' 'Barley' 'Khesari' 'Masoor' 'Moong(Green Gram)'

'Other Kharif pulses' 'Safflower' 'Sannhamp' 'Sunflower' 'Urad'

'Peas & beans (Pulses)' 'other oilseeds' 'Other Cereals' 'Cowpea(Lobia)'

'Oilseeds total' 'Guar seed' 'Other Summer Pulses' 'Moth']

Filtered Dataset (only rice data):

Crop Crop\_Year Season State Area Production \

16 Rice 1997 Autumn Assam 607358.0 398311

17 Rice 1997 Summer Assam 174974.0 209623

18 Rice 1997 Winter Assam 1743321.0 1647296

51 Rice 1997 Kharif Karnataka 1031530.0 2340493

52 Rice 1997 Rabi Karnataka 53889.0 109350

Annual\_Rainfall Fertilizer Pesticide Yield

16 2051.4 5.780226e+07 188280.98 0.780870

17 2051.4 1.665228e+07 54241.94 1.060435

18 2051.4 1.659119e+08 540429.51 0.941304

51 1266.7 9.817071e+07 319774.30 2.233500

52 1266.7 5.128616e+06 16705.59 2.073846

Decision Tree Mean Squared Error (MSE): 0.3468

Decision Tree R-squared (R²): 0.4961

Top 10 Yield Predictions:

Actual Yield Predicted Yield Accuracy

6266 4.778276 3.267241 84.436108

4394 4.730323 4.226452 89.348063

5745 4.184828 4.077241 97.429136

5746 3.997333 4.077241 98.000966

14529 3.923182 4.252727 91.600046

11788 3.834000 3.820588 99.650189

8789 3.660000 3.423333 93.533698

17139 3.610625 4.226452 82.944044

15184 3.585000 2.785556 77.700294

17138 3.575625 4.226452 81.798242

<ipython-input-7-7ef58279f7ad>:34: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation:<https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

rice\_data[column] = label\_encoder.fit\_transform(rice\_data[column].astype(str).str.strip())

<ipython-input-7-7ef58279f7ad>:34: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation:<https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

rice\_data[column] = label\_encoder.fit\_transform(rice\_data[column].astype(str).str.strip())

<ipython-input-7-7ef58279f7ad>:34: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation:<https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

rice\_data[column] = label\_encoder.fit\_transform(rice\_data[column].astype(str).str.strip())

Mean Accuracy of Yield Predictions: 8.23%

**5 .KNN ALGORITHM :**

**import pandas as pd**

**import numpy as np**

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

**from sklearn.neighbors import KNeighborsRegressor**

**from sklearn.metrics import mean\_squared\_error, r2\_score**

**from sklearn.preprocessing import LabelEncoder**

**# Load the data from the CSV file**

**file\_path = "/content/drive/MyDrive/crop\_yield.csv"**

**data = pd.read\_csv(file\_path)**

**# Print the first few rows to understand the structure of the dataset**

**print("Original Dataset:")**

**print(data.head())**

**# Check unique values in the Crop column**

**print("Unique values in Crop column:", data['Crop'].unique())**

**# Filter the dataset to include only rows where the crop type is 'rice'**

**rice\_data = data[data['Crop'].str.strip().str.lower() == 'rice'] # Ensure case insensitivity and strip spaces**

**# Print the filtered rows to confirm**

**if rice\_data.empty:**

**print("No data found for rice. Please check the Crop column for correct values.")**

**else:**

**print("Filtered Dataset (only rice data):")**

**print(rice\_data.head())**

**# Convert categorical columns to numerical using LabelEncoder**

**label\_encoder = LabelEncoder()**

**# Encode each categorical column in the filtered dataframe**

**for column in rice\_data.select\_dtypes(include=['object']).columns:**

**rice\_data[column] = label\_encoder.fit\_transform(rice\_data[column].astype(str).str.strip())**

**# Define features (X) and target (y) from the filtered data**

**X = rice\_data.drop('Yield', axis=1) # Features (adjust column names if necessary)**

**y = rice\_data['Yield'] # Target (yield)**

**# Split the dataset into training and testing sets**

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

**# Train a KNN model**

**knn\_model = KNeighborsRegressor(n\_neighbors=5) # You can adjust the number of neighbors**

**knn\_model.fit(X\_train, y\_train)**

**# Predict on test data**

**y\_pred = knn\_model.predict(X\_test)**

**# Evaluate the model using Mean Squared Error (MSE) and R-squared (R²)**

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

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

**# Print evaluation metrics**

**print(f"KNN Mean Squared Error (MSE): {mse:.4f}")**

**print(f"KNN R-squared (R²): {r2:.4f}")**

**# Calculate accuracy of predictions based on actual yield**

**accuracy = 100 \* (1 - (np.abs(y\_test - y\_pred) / y\_test))**

**accuracy\_df = pd.DataFrame({**

**'Actual Yield': y\_test,**

**'Predicted Yield': y\_pred,**

**'Accuracy': accuracy**

**})**

**# Select top 10 predictions based on actual yield**

**top\_10\_predictions = accuracy\_df.nlargest(10, 'Actual Yield')**

**print("Top 10 Yield Predictions:")**

**print(top\_10\_predictions)**

**OUTPUT :**

**Original Dataset:**

**Crop Crop\_Year Season State Area Production \**

**0 Arecanut 1997 Whole Year Assam 73814.0 56708**

**1 Arhar/Tur 1997 Kharif Assam 6637.0 4685**

**2 Castor seed 1997 Kharif Assam 796.0 22**

**3 Coconut 1997 Whole Year Assam 19656.0 126905000**

**4 Cotton(lint) 1997 Kharif Assam 1739.0 794**

**Annual\_Rainfall Fertilizer Pesticide Yield**

**0 2051.4 7024878.38 22882.34 0.796087**

**1 2051.4 631643.29 2057.47 0.710435**

**2 2051.4 75755.32 246.76 0.238333**

**3 2051.4 1870661.52 6093.36 5238.051739**

**4 2051.4 165500.63 539.09 0.420909**

**Unique values in Crop column: ['Arecanut' 'Arhar/Tur' 'Castor seed' 'Coconut ' 'Cotton(lint)'**

**'Dry chillies' 'Gram' 'Jute' 'Linseed' 'Maize' 'Mesta' 'Niger seed'**

**'Onion' 'Other Rabi pulses' 'Potato' 'Rapeseed &Mustard' 'Rice'**

**'Sesamum' 'Small millets' 'Sugarcane' 'Sweet potato' 'Tapioca' 'Tobacco'**

**'Turmeric' 'Wheat' 'Bajra' 'Black pepper' 'Cardamom' 'Coriander' 'Garlic'**

**'Ginger' 'Groundnut' 'Horse-gram' 'Jowar' 'Ragi' 'Cashewnut' 'Banana'**

**'Soyabean' 'Barley' 'Khesari' 'Masoor' 'Moong(Green Gram)'**

**'Other Kharif pulses' 'Safflower' 'Sannhamp' 'Sunflower' 'Urad'**

**'Peas & beans (Pulses)' 'other oilseeds' 'Other Cereals' 'Cowpea(Lobia)'**

**'Oilseeds total' 'Guar seed' 'Other Summer Pulses' 'Moth']**

**Filtered Dataset (only rice data):**

**Crop Crop\_Year Season State Area Production \**

**16 Rice 1997 Autumn Assam 607358.0 398311**

**17 Rice 1997 Summer Assam 174974.0 209623**

**18 Rice 1997 Winter Assam 1743321.0 1647296**

**51 Rice 1997 Kharif Karnataka 1031530.0 2340493**

**52 Rice 1997 Rabi Karnataka 53889.0 109350**

**Annual\_Rainfall Fertilizer Pesticide Yield**

**16 2051.4 5.780226e+07 188280.98 0.780870**

**17 2051.4 1.665228e+07 54241.94 1.060435**

**18 2051.4 1.659119e+08 540429.51 0.941304**

**51 1266.7 9.817071e+07 319774.30 2.233500**

**52 1266.7 5.128616e+06 16705.59 2.073846**

**KNN Mean Squared Error (MSE): 0.6275**

**KNN R-squared (R²): 0.0883**

**Top 10 Yield Predictions:**

**Actual Yield Predicted Yield Accuracy**

**6266 8.778276 2.296693 86.163375**

**4394 4.730323 1.883576 79.819185**

**5745 4.184828 2.144932 81.254958**

**5746 3.997333 2.439791 81.035455**

**14529 3.923182 2.599997 66.272661**

**11788 3.834000 2.781397 72.545565**

**8789 3.660000 2.535034 79.263237**

**17139 3.610625 1.678624 86.491218**

**15184 3.585000 3.340393 93.176934**

**17138 3.575625 2.236525 82.549216**

**<ipython-input-8-c0177ea92ca6>:34: SettingWithCopyWarning:**

**A value is trying to be set on a copy of a slice from a DataFrame.**

**Try using .loc[row\_indexer,col\_indexer] = value instead**

**See the caveats in the documentation:** [**https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy**](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

**rice\_data[column] = label\_encoder.fit\_transform(rice\_data[column].astype(str).str.strip())**

**<ipython-input-8-c0177ea92ca6>:34: SettingWithCopyWarning:**

**A value is trying to be set on a copy of a slice from a DataFrame.**

**Try using .loc[row\_indexer,col\_indexer] = value instead**

**See the caveats in the documentation:** [**https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy**](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

**rice\_data[column] = label\_encoder.fit\_transform(rice\_data[column].astype(str).str.strip())**

**<ipython-input-8-c0177ea92ca6>:34: SettingWithCopyWarning:**

**A value is trying to be set on a copy of a slice from a DataFrame.**

**Try using .loc[row\_indexer,col\_indexer] = value instead**

**See the caveats in the documentation:** [**https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy**](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

**rice\_data[column] = label\_encoder.fit\_transform(rice\_data[column].astype(str).str.strip()).**