Crop Recommendation

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier, plot_tree

from sklearn.naive_bayes import GaussianNB

from sklearn.svm import SVC

from sklearn.ensemble import RandomForestClassifier, VotingClassifier

from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
```

Ensemble Model

```
# 1. Load the dataset

data = pd.read_csv('Crop_Recommendation.csv')  # Adjust path if needed

print(data.head())

# 2. Features and Target

X = data.drop('Crop', axis=1)

y = data['Crop']

# 3. Train-Test Split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 4. Feature Scaling (Not mandatory for tree-based models, but okay if used)

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)

# 5. Train Random Forest Model

rf_model = RandomForestClassifier(

    n_estimators=200,       # Number of trees

    max_depth=None,         # Let it grow fully (or you can set a depth like 10, 20 to control overfitting)

    random_state=42

)

rf_model.fit(X_train, y_train)  # No scaling needed for Random Forest (so using X_train directly)
```

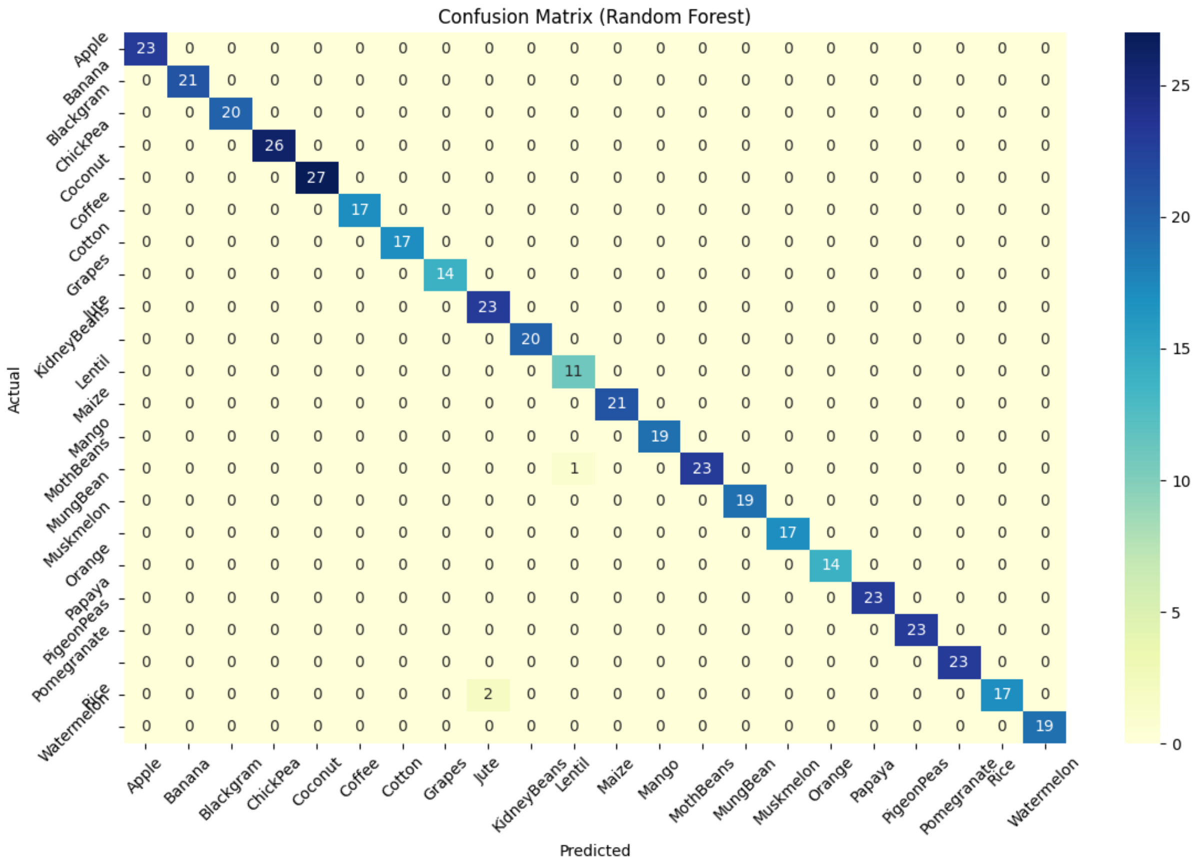
```
# 6. Predictions
y_pred = rf_model.predict(X_test)
# 7. Evaluation
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
# 8. Confusion Matrix
plt.figure(figsize=(12, 8))
sns.heatmap(
    confusion_matrix(y_test, y_pred),
    annot=True, fmt="d", cmap="YlGnBu",
    xticklabels=rf_model.classes_,
    yticklabels=rf_model.classes_
)
plt.title("Confusion Matrix (Random Forest)")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.xticks(rotation=45)
plt.yticks(rotation=45)
plt.tight_layout()
plt.show()
```

output

```
Nitrogen  Phosphorus  Potassium  Temperature   Humidity  pH_Value  \
0       90          42         43    20.879744  82.002744  6.502985
1       85          58         41    21.770462  80.319644  7.038096
2       60          55         44    23.004459  82.320763  7.840207
3       74          35         40    26.491096  80.158363  6.980401
4       78          42         42    20.130175  81.604873  7.628473
     Rainfall  Crop
0  202.935536  Rice
1  226.655537  Rice
2  263.964248  Rice
3  242.864034  Rice
4  262.717340  Rice
Accuracy: 0.9931818181818182
```

Classification Report:

	precision	recall	f1-score	support
Apple	1.00	1.00	1.00	23
Banana	1.00	1.00	1.00	21
Blackgram	1.00	1.00	1.00	20
ChickPea	1.00	1.00	1.00	26
Coconut	1.00	1.00	1.00	27
Coffee	1.00	1.00	1.00	17
Cotton	1.00	1.00	1.00	17
Grapes	1.00	1.00	1.00	14
Jute	0.92	1.00	0.96	23
KidneyBeans	1.00	1.00	1.00	20
Lentil	0.92	1.00	0.96	11
Maize	1.00	1.00	1.00	21
Mango	1.00	1.00	1.00	19
MothBeans	1.00	0.96	0.98	24
MungBean	1.00	1.00	1.00	19
Muskmelon	1.00	1.00	1.00	17
Orange	1.00	1.00	1.00	14
Papaya	1.00	1.00	1.00	23
PigeonPeas	1.00	1.00	1.00	23
Pomegranate	1.00	1.00	1.00	23
Rice	1.00	0.89	0.94	19
Watermelon	1.00	1.00	1.00	19
accuracy			0.99	440
macro avg	0.99	0.99	0.99	440
weighted avg	0.99	0.99	0.99	440



Support vector Machine

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 6. Feature scaling (important for SVM)

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)

# 7. Train the SVM model (you can adjust kernel, C, and other parameters)

model = SVC(kernel='linear')  # You can change the kernel to 'rbf', 'poly', etc.

model.fit(X_train_scaled, y_train)

# 8. Predictions

y_pred = model.predict(X_test_scaled)

# 9. Evaluation

print("Accuracy:", accuracy_score(y_test, y_pred))

print("\nClassification Report:\n", classification_report(y_test, y_pred))

# 10. Confusion Matrix

plt.figure(figsize=(12, 8))
```

sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt="d", cmap="Blues", xticklabels=model.classes_, yticklabels=model.classes_)

plt.title("Confusion Matrix")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.xticks(rotation=45)

plt.yticks(rotation=45)

plt.tight_layout()

plt.show()

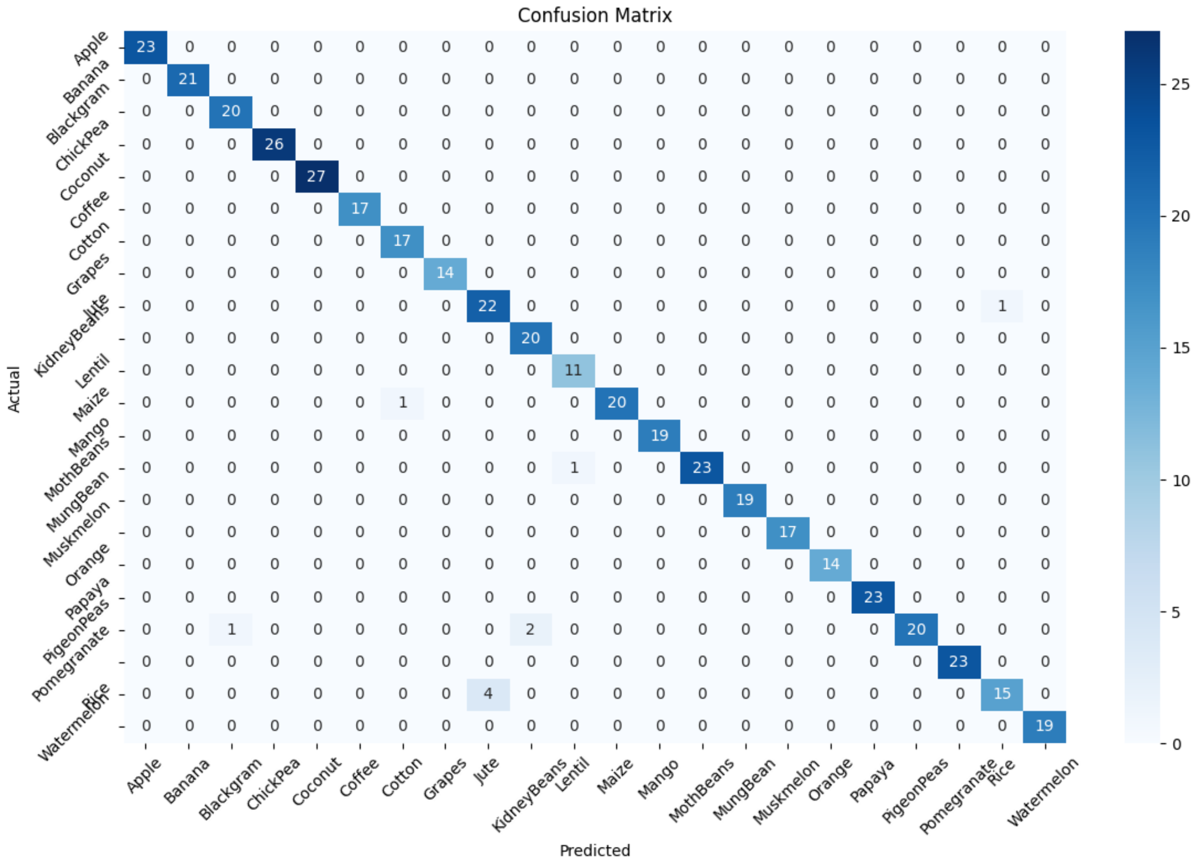
Output:

Accuracy: 0.9772727272727273

Classification Report:

	precision	recall	f1-score	support
Apple	1.00	1.00	1.00	23
Banana	1.00	1.00	1.00	21
Blackgram	0.95	1.00	0.98	20
ChickPea	1.00	1.00	1.00	26
Coconut	1.00	1.00	1.00	27
Coffee	1.00	1.00	1.00	17
Cotton	0.94	1.00	0.97	17
Grapes	1.00	1.00	1.00	14
Jute	0.85	0.96	0.90	23
KidneyBeans	0.91	1.00	0.95	20
Lentil	0.92	1.00	0.96	11
Maize	1.00	0.95	0.98	21
Mango	1.00	1.00	1.00	19
MothBeans	1.00	0.96	0.98	24
MungBean	1.00	1.00	1.00	19
Muskmelon	1.00	1.00	1.00	17
Orange	1.00	1.00	1.00	14
Papaya	1.00	1.00	1.00	23
PigeonPeas	1.00	0.87	0.93	23
Pomegranate	1.00	1.00	1.00	23
Rice	0.94	0.79	0.86	19
Watermelon	1.00	1.00	1.00	19
accuracy			0.98	440
macro avg	0.98	0.98	0.98	440

weighted avg 0.98 0.98 0.98 440



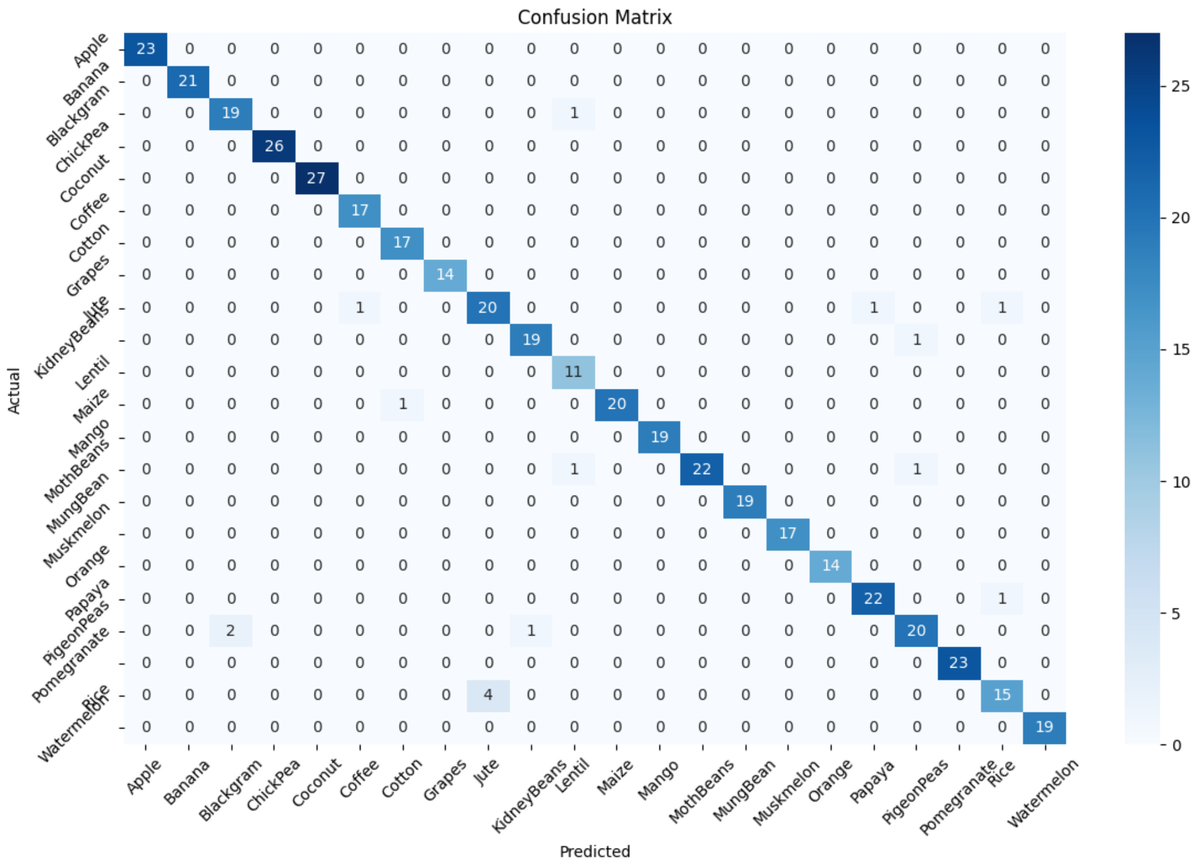
Logistic Regression:

```
# 3. Train-Test Split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 4. Feature Scaling

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)

# 5. Train Logistic Regression Model

model = LogisticRegression(

    max_iter=300,          # Increase max_iter if convergence warning occurs

    solver='lbfgs',        # Good solver for multinomial problems

    multi_class='multinomial'  # Because we have multiple crop classes

)

model.fit(X_train_scaled, y_train)

# 6. Predictions

y_pred = model.predict(X_test_scaled)

# 7. Evaluation

print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# 8. Confusion Matrix

plt.figure(figsize=(12, 8))

sns.heatmap(

    confusion_matrix(y_test, y_pred),

    annot=True, fmt="d", cmap="Blues",

    xticklabels=model.classes_,

    yticklabels=model.classes_

)

plt.title("Confusion Matrix")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.xticks(rotation=45)

plt.yticks(rotation=45)

plt.tight_layout()

plt.show()
```

Output:

Accuracy: 0.9636363636363636

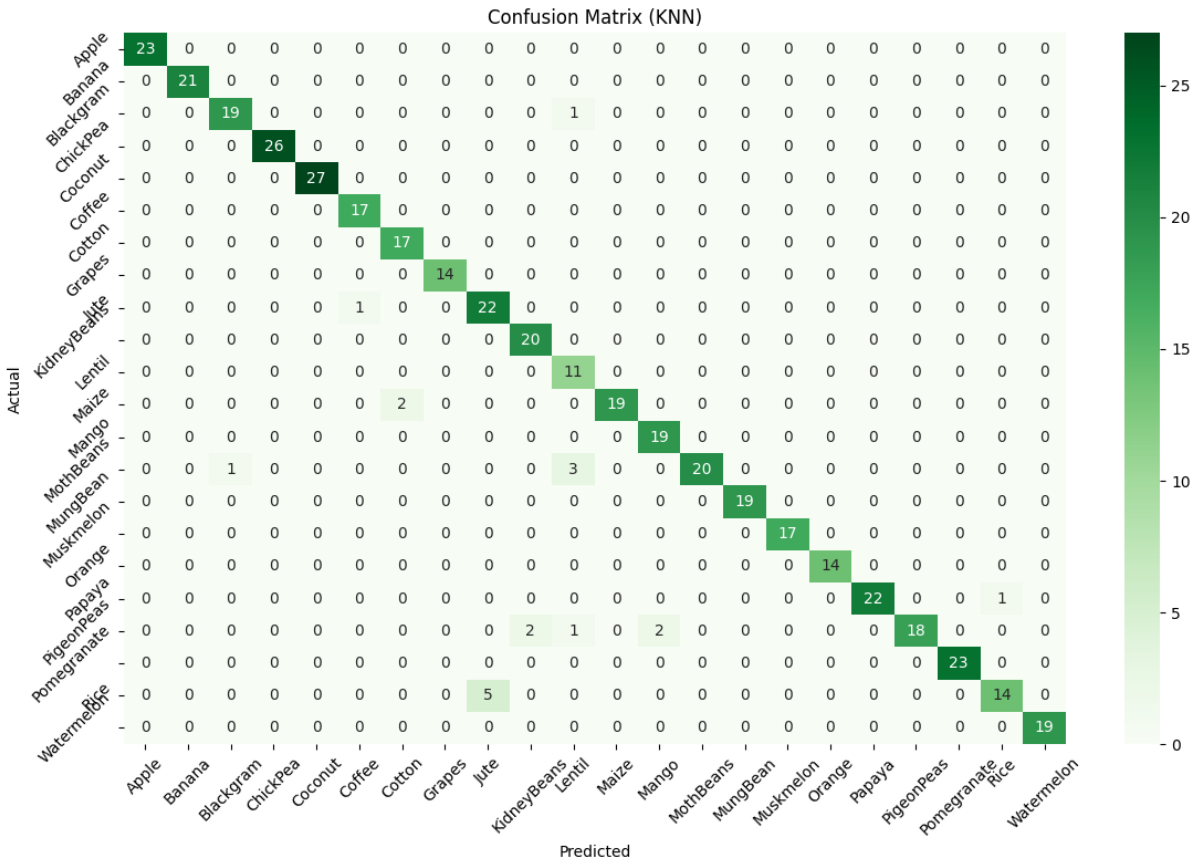


KNN Model:

```
# 3. Train-Test Split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 4. Feature Scaling (very important for KNN)

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)

# 5. Train KNN Model

knn_model = KNeighborsClassifier(n_neighbors=5)  # You can tune n_neighbors later

knn_model.fit(X_train_scaled, y_train)

# 6. Predictions

y_pred = knn_model.predict(X_test_scaled)

# 7. Evaluation

print("Accuracy:", accuracy_score(y_test, y_pred))

print("\nClassification Report:\n", classification_report(y_test, y_pred))

# 8. Confusion Matrix

plt.figure(figsize=(12, 8))

sns.heatmap(

    confusion_matrix(y_test, y_pred),

    annot=True, fmt="d", cmap="Greens",

    xticklabels=knn_model.classes_,

    yticklabels=knn_model.classes_

)

plt.title("Confusion Matrix (KNN)")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.xticks(rotation=45)

plt.yticks(rotation=45)

plt.tight_layout()

plt.show()
```

Output:

```
Accuracy: 0.9568181818181818

    accuracy                           0.96       440

   macro avg       0.96      0.96      0.95       440

weighted avg       0.96      0.96      0.96       440
```



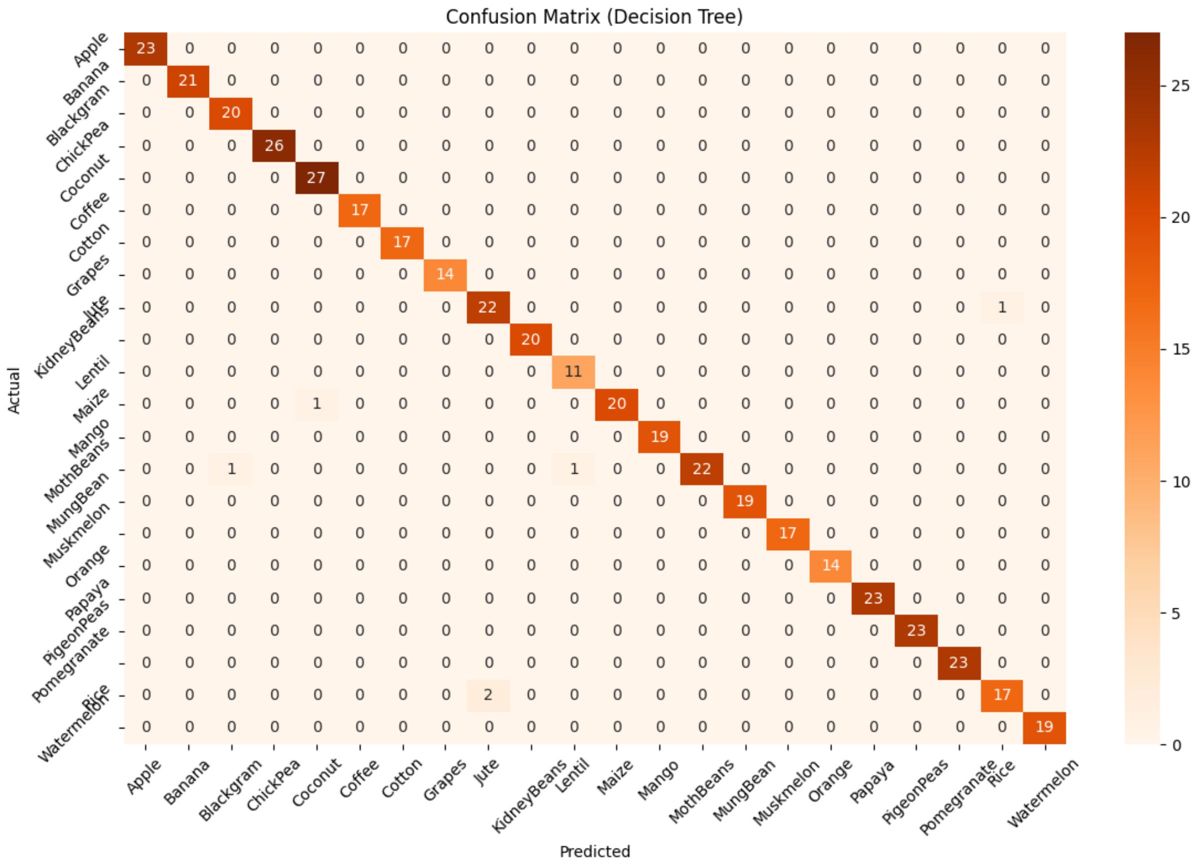
Decision Tree:

```
# 2. Features and Target

X = data.drop('Crop', axis=1)

y = data['Crop']

# 3. Train-Test Split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 4. Feature Scaling (NOT necessary for Decision Trees, but safe if you plan to combine later)

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)

# 5. Train Decision Tree Model

dt_model = DecisionTreeClassifier(random_state=42, max_depth=None)  # You can tune max_depth if needed

dt_model.fit(X_train, y_train)  # For tree models, scaling not necessary. So use X_train directly.

# 6. Predictions

y_pred = dt_model.predict(X_test)

# 7. Evaluation

print("Accuracy:", accuracy_score(y_test, y_pred))

print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

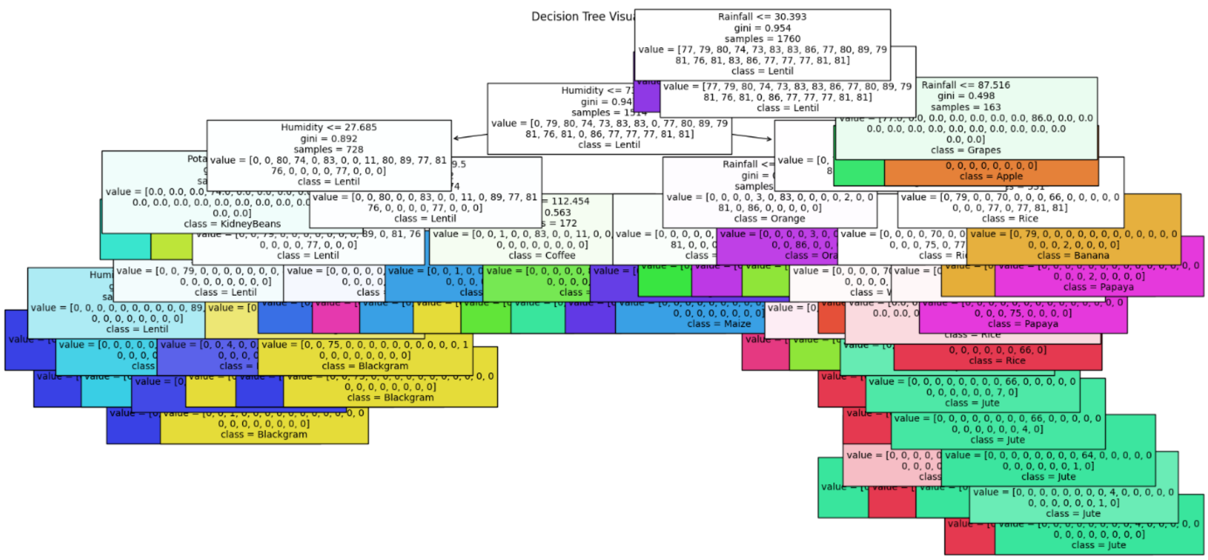
```python
# 8. Confusion Matrix
plt.figure(figsize=(12, 8))
sns.heatmap(
    confusion_matrix(y_test, y_pred),
    annot=True, fmt="d", cmap="Oranges",
    xticklabels=dt_model.classes_,
    yticklabels=dt_model.classes_
)
plt.title("Confusion Matrix (Decision Tree)")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.xticks(rotation=45)
plt.yticks(rotation=45)
plt.tight_layout()
plt.show()
plt.figure(figsize=(20, 10))
plot_tree(dt_model, filled=True, feature_names=X.columns, class_names=dt_model.classes_, fontsize=10)
plt.title("Decision Tree Visualization")
plt.show()
```

Output:

```
Accuracy: 0.9863636363636363
accuracy                          0.99      440
   macro avg      0.99      0.99      0.99      440
weighted avg      0.99      0.99      0.99      440
```



Confusion Matrix (Decision Tree)
Actual
Predicted
Apple
Banana
Blackgram
ChickPea
Coconut
Coffee
Cotton
Grapes
Jute
KidneyBeans
Lentil
Maize
Mango
MothBeans
MungBean
Muskmelon
Orange
Papaya
PigeonPeas
Pomegranate
Rice
Watermelon



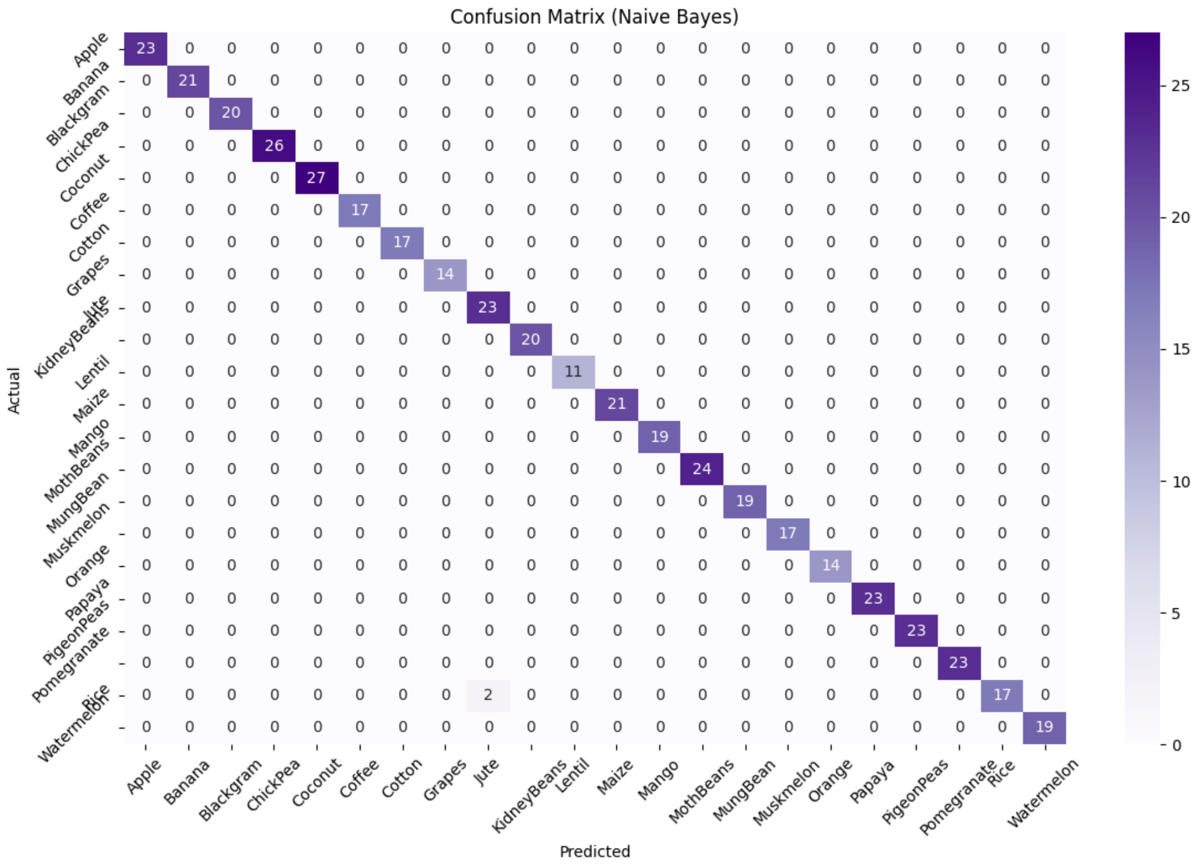
Rainfall <= 30.393
gini = 0.954
samples = 1760
class = Lentil

Naïve Base:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 4. Feature Scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
# 5. Train Naive Bayes Model
nb_model = GaussianNB()
nb_model.fit(X_train_scaled, y_train)
# 6. Predictions
y_pred = nb_model.predict(X_test_scaled)
# 7. Evaluation
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
# 8. Confusion Matrix
plt.figure(figsize=(12, 8))
sns.heatmap(
    confusion_matrix(y_test, y_pred),
    annot=True, fmt="d", cmap="Purples",
    xticklabels=nb_model.classes_,
    yticklabels=nb_model.classes_
)
plt.title("Confusion Matrix (Naive Bayes)")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.xticks(rotation=45)
plt.yticks(rotation=45)
plt.tight_layout()
plt.show()
```

Output:

```
Accuracy: 0.9954545454545455
accuracy                        1.00      440
  macro avg      1.00      1.00      1.00      440
weighted avg      1.00      1.00      1.00      440
```


Confusion Matrix (Naive Bayes)
Actual
Predicted
Apple
Banana
Blackgram
ChickPea
Coconut
Coffee
Cotton
Grapes
Jute
KidneyBeans
Lentil
Maize
Mango
MothBeans
MungBean
Muskmelon
Orange
Papaya
PigeonPeas
Pomegranate
Rice
Watermelon