```
# ========================================
# 1. Import Libraries
# ========================================

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
```

```
# ========================================
# 2. Load Dataset
# ========================================

df = pd.read_csv("crop_yield_dataset_50000.csv")

print(df.head())
print(df.info())
```

```
    Region  Crop_Type  Rainfall_mm  Temperature_C  Humidity_%  Soil_pH  \
0     West      Maize       432.52          24.47       59.29     7.11
1  Central      Maize       757.60          32.87       62.20     4.82
2     East      Maize      1567.35          37.62       40.71     5.62
3  Central     Cotton       597.73          30.68       36.53     8.23
4  Central  Sugarcane       698.53          29.32       52.22     5.45

   Fertilizer_kg_per_hectare  Pesticide_kg_per_hectare  Area_hectare  \
0                     132.31                      2.98          2.04
1                     159.85                      9.67         12.26
2                     142.69                      3.29          4.87
3                     184.17                      8.04          2.52
```

```
4                      89.29                      7.27          0.89

   Crop_Yield_ton_per_hectare
0                       19.22
1                       30.88
2                       33.47
3                       28.28
4                       23.04
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 10 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Region                      50000 non-null  object
 1   Crop_Type                   50000 non-null  object
 2   Rainfall_mm                 50000 non-null  float64
 3   Temperature_C               50000 non-null  float64
 4   Humidity_%                  50000 non-null  float64
 5   Soil_pH                     50000 non-null  float64
 6   Fertilizer_kg_per_hectare   50000 non-null  float64
 7   Pesticide_kg_per_hectare    50000 non-null  float64
 8   Area_hectare                50000 non-null  float64
 9   Crop_Yield_ton_per_hectare  50000 non-null  float64
dtypes: float64(8), object(2)
memory usage: 3.8+ MB
None
```

```
# ========================================
# 3. Data Preprocessing
# ========================================

# Check missing values
print(df.isnull().sum())

# Encode categorical variables
le = LabelEncoder()

df["Region"] = le.fit_transform(df["Region"])
df["Crop_Type"] = le.fit_transform(df["Crop_Type"])

# Feature & Target split
```

```
X = df.drop("Crop_Yield_ton_per_hectare", axis=1)
y = df["Crop_Yield_ton_per_hectare"]

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Feature Scaling
scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
Region                        0
Crop_Type                     0
Rainfall_mm                   0
Temperature_C                 0
Humidity_%                    0
Soil_pH                       0
Fertilizer_kg_per_hectare     0
Pesticide_kg_per_hectare      0
Area_hectare                  0
Crop_Yield_ton_per_hectare    0
dtype: int64
```

```
# ========================================
# 4. Model Training
# ========================================

# Linear Regression
lr = LinearRegression()
lr.fit(X_train, y_train)

# Random Forest
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
```

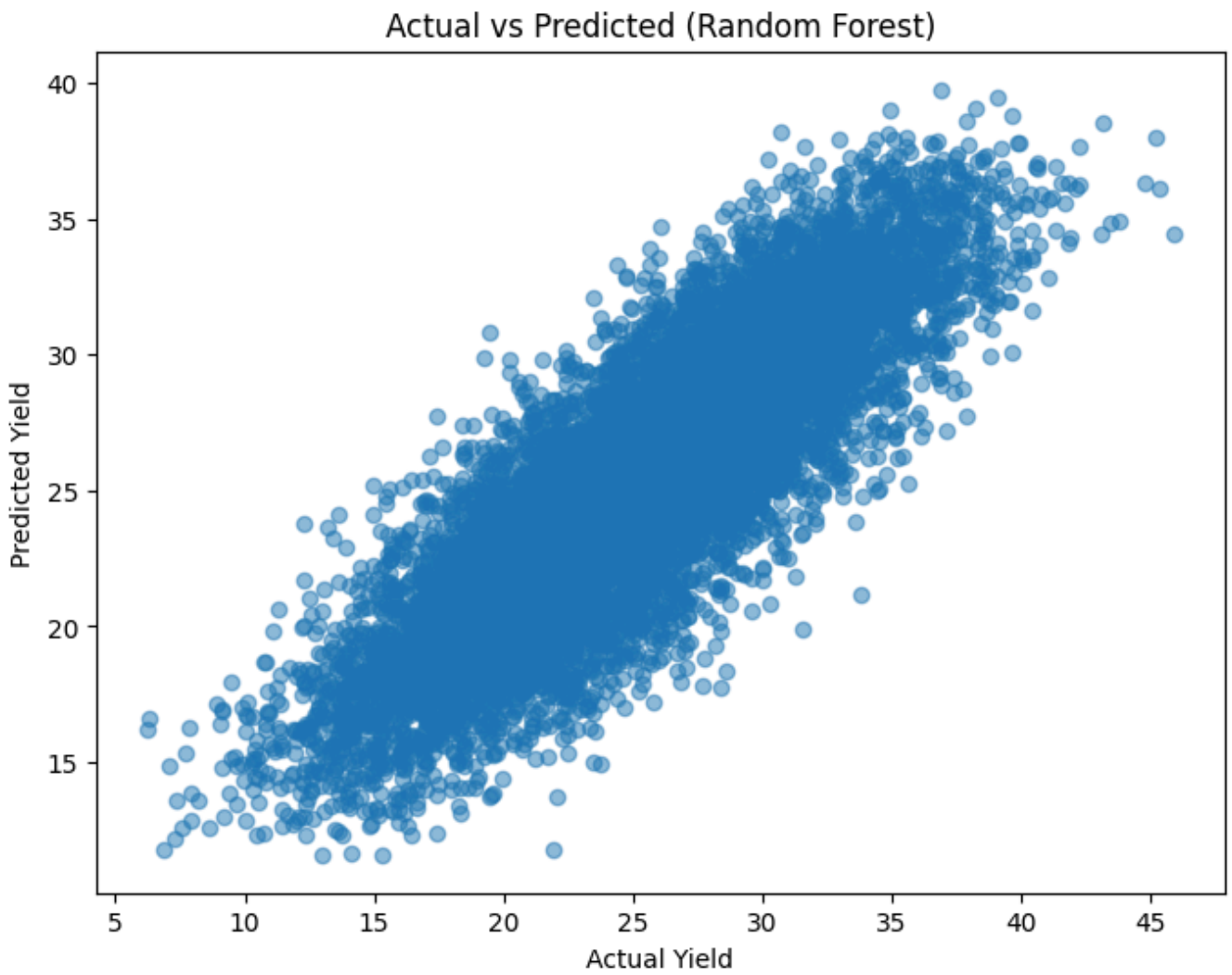
```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

```
# ========================================
# 5. Model Evaluation
# ========================================

def evaluate(model, X_test, y_test, name):

    y_pred = model.predict(X_test)

    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    print(f"--- {name} ---")
    print("RMSE:", rmse)
    print("MAE :", mae)
    print("R2  :", r2)
    print()

    return y_pred


lr_pred = evaluate(lr, X_test, y_test, "Linear Regression")
rf_pred = evaluate(rf, X_test, y_test, "Random Forest")
```

```
--- Linear Regression ---
RMSE: 2.9971372600077117
MAE : 2.378988679568162
R2  : 0.7441756874001002

--- Random Forest ---
RMSE: 3.1037788921975413
MAE : 2.46370429
```

```
R2  : 0.725646755290246
```

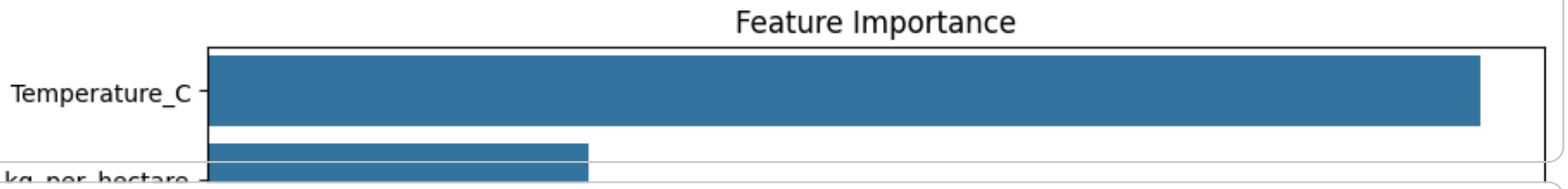
```
# ========================================
# 6. Visualization
# ========================================

plt.figure(figsize=(8,6))
plt.scatter(y_test, rf_pred, alpha=0.5)
plt.xlabel("Actual Yield")
plt.ylabel("Predicted Yield")
plt.title("Actual vs Predicted (Random Forest)")
plt.show()
```




```
# ========================================
# 7. Feature Importance
# ========================================

importance = rf.feature_importances_

features = X.columns
```

```
imp_df = pd.DataFrame({
    "Feature": features,
    "Importance": importance
}).sort_values(by="Importance", ascending=False)

print(imp_df)

# Plot
plt.figure(figsize=(10,6))
sns.barplot(x="Importance", y="Feature", data=imp_df)
plt.title("Feature Importance")
plt.show()
```

```
                    Feature  Importance
3               Temperature_C    0.554693
6   Fertilizer_kg_per_hectare    0.166173
2                 Rainfall_mm    0.129629
5                     Soil_pH    0.032482
4                  Humidity_%    0.032369
8                Area_hectare    0.031640
7    Pesticide_kg_per_hectare    0.031463
1                   Crop_Type    0.010856
0                      Region    0.010694
```




```
# ========================================
# 8. Sample Prediction
# ========================================

sample = X.iloc[0:1]
sample = scaler.transform(sample)

predicted_yield = rf.predict(sample)
```