```
!pip install shap xgboost --quiet

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.ensemble import RandomForestRegressor, RandomForestClassifier
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score, classification_report
import shap
```

```
crop_df = pd.read_csv("/content/Crop_recommendation.csv")         # crop dataset
fert_df = pd.read_csv("/content/Fertilizer Prediction.csv")       # fertilizer dataset
soil_df = pd.read_csv("/content/sensor_Crop_Dataset (1).csv")                    # soil dataset

print("Crop dataset:", crop_df.shape)
print("Fertilizer dataset:", fert_df.shape)
print("Soil dataset:", soil_df.shape)

display(crop_df.head(), fert_df.head(), soil_df.head())
```

```
Crop dataset: (2200, 8)
Fertilizer dataset: (99, 9)
Soil dataset: (20000, 10)
```

	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

	Temparature	Humidity	Moisture	Soil Type	Crop Type	Nitrogen	Potassium	Phosphorous	Fertilizer Name
0	26	52	38	Sandy	Maize	37	0	0	Urea
1	29	52	45	Loamy	Sugarcane	12	0	36	DAP
2	34	65	62	Black	Cotton	7	9	30	14-35-14
3	32	62	34	Red	Tobacco	22	0	20	28-28
4	28	54	46	Clayey	Paddy	35	0	0	Urea

	Nitrogen	Phosphorus	Potassium	Temperature	Humidity	pH_Value	Rainfall	Crop	Soil_Type	Variety
0	69.074766	53.954402	88.067625	17.261834	72.941652	4.631301	302.842639	Wheat	Clay	Soft Red
1	107.329352	70.102134	32.081067	21.846116	99.361954	4.761658	94.693847	Tomato	Clay	Beefsteak
2	130.634624	67.204533	28.294252	33.246895	81.506836	6.566007	83.563685	Sugarcane	Clay	Co 86032
3	15.169301	87.493181	14.336679	14.396289	59.274465	6.296297	31.508836	Sugarcane	Silt	Co 0238
4	21.881965	89.269712	38.833885	16.773218	51.191584	8.268274	295.193482	Maize	Sandy	Sweet

```
crop_df.rename(columns={'label':'Crop'}, inplace=True)

# Standardize fertilizer dataset column names
fert_df.rename(columns=lambda x: x.strip().capitalize().replace(" ", "_"), inplace=True)

# Merge datasets on crop
merged_df = pd.merge(crop_df, soil_df, on="Crop", how="inner")

print("Merged dataset:", merged_df.shape)
display(merged_df.head())
```

```
Merged dataset: (0, 17)
```

N	P	K	temperature	humidity	ph	rainfall	Crop	Nitrogen	Phosphorus	Potassium	Temperature	Humidity	pH_Value	Rainfall	Soi

✦ Gemini

```
print("Unique crops in crop_df:", crop_df['Crop'].unique())
print("Unique crops in soil_df:", soil_df['Crop'].unique())
print("Unique crops in fert_df:", fert_df['Crop_type'].unique())
```

```
Unique crops in crop_df: ['rice' 'maize' 'chickpea' 'kidneybeans' 'pigeonpeas' 'mothbeans'
 'mungbean' 'blackgram' 'lentil' 'pomegranate' 'banana' 'mango' 'grapes'
 'watermelon' 'muskmelon' 'apple' 'orange' 'papaya' 'coconut' 'cotton'
 'jute' 'coffee']
Unique crops in soil_df: ['Wheat' 'Tomato' 'Sugarcane' 'Maize' 'Potato' 'Rice']
Unique crops in fert_df: ['Maize' 'Sugarcane' 'Cotton' 'Tobacco' 'Paddy' 'Barley' 'Wheat' 'Millets'
 'Oil seeds' 'Pulses' 'Ground Nuts']
```

```
print(soil_df.columns.tolist())
soil_df.head()
```

```
['Nitrogen', 'Phosphorus', 'Potassium', 'Temperature', 'Humidity', 'pH_Value', 'Rainfall', 'Crop', 'Soil_Type', 'Variety']
```

	Nitrogen	Phosphorus	Potassium	Temperature	Humidity	pH_Value	Rainfall	Crop	Soil_Type	Variety
0	69.074766	53.954402	88.067625	17.261834	72.941652	4.631301	302.842639	Wheat	Clay	Soft Red
1	107.329352	70.102134	32.081067	21.846116	99.361954	4.761658	94.693847	Tomato	Clay	Beefsteak
2	130.634624	67.204533	28.294252	33.246895	81.506836	6.566007	83.563685	Sugarcane	Clay	Co 86032
3	15.169301	87.493181	14.336679	14.396289	59.274465	6.296297	31.508836	Sugarcane	Silt	Co 0238
4	21.881965	89.269712	38.833885	16.773218	51.191584	8.268274	295.193482	Maize	Sandy	Sweet

Next steps: Generate code with soil_df | View recommended plots | New interactive sheet

```
# Example synthetic dose calculation (you can modify)
soil_df['N_required'] = soil_df['Nitrogen'].apply(lambda x: max(0, 120 - x))
soil_df['P_required'] = soil_df['Phosphorus'].apply(lambda x: max(0, 60 - x))
soil_df['K_required'] = soil_df['Potassium'].apply(lambda x: max(0, 80 - x))
```

```
X_reg = soil_df[['Nitrogen','Phosphorus','Potassium','Temperature','Humidity','pH_Value','Rainfall']]
y_reg = soil_df[['N_required','P_required','K_required']]
```

```
# --- Cell 4: Regression Model (Dose Prediction) ---

# Step 1: Create synthetic target columns (dose required)
# Assuming threshold values: N=120, P=60, K=80 (adjust if needed)
soil_df['N_required'] = soil_df['Nitrogen'].apply(lambda x: max(0, 120 - x))
soil_df['P_required'] = soil_df['Phosphorus'].apply(lambda x: max(0, 60 - x))
soil_df['K_required'] = soil_df['Potassium'].apply(lambda x: max(0, 80 - x))

# Step 2: Define features (X) and target (y)
X_reg = soil_df[['Nitrogen','Phosphorus','Potassium','Temperature','Humidity','pH_Value','Rainfall']]
y_reg = soil_df[['N_required','P_required','K_required']]

# Step 3: Train-Test Split
X_train_reg, X_test_reg, y_train_reg, y_test_reg = train_test_split(X_reg, y_reg, test_size=0.2, random_state=42)

# Step 4: Train model (Random Forest Regressor)
reg_model = RandomForestRegressor(n_estimators=100, random_state=42)
reg_model.fit(X_train_reg, y_train_reg)

# Step 5: Predictions
y_pred_reg = reg_model.predict(X_test_reg)

# Step 6: Evaluation
print("Regression R² Score:", r2_score(y_test_reg, y_pred_reg))
```

```
Regression R² Score: 0.9981388919616414
```

```
print(fert_df.columns.tolist())
```

```
['Temparature', 'Humidity', 'Moisture', 'Soil_type', 'Crop_type', 'Nitrogen', 'Potassium', 'Phosphorous', 'Fertilizer_name']
```

```
# Features & target (with correct column names)
X_cls = fert_df[['Temparature','Humidity','Moisture','Soil_type','Crop_type','Nitrogen','Potassium','Phosphorous']]
y_cls = fert_df['Fertilizer_name']
```

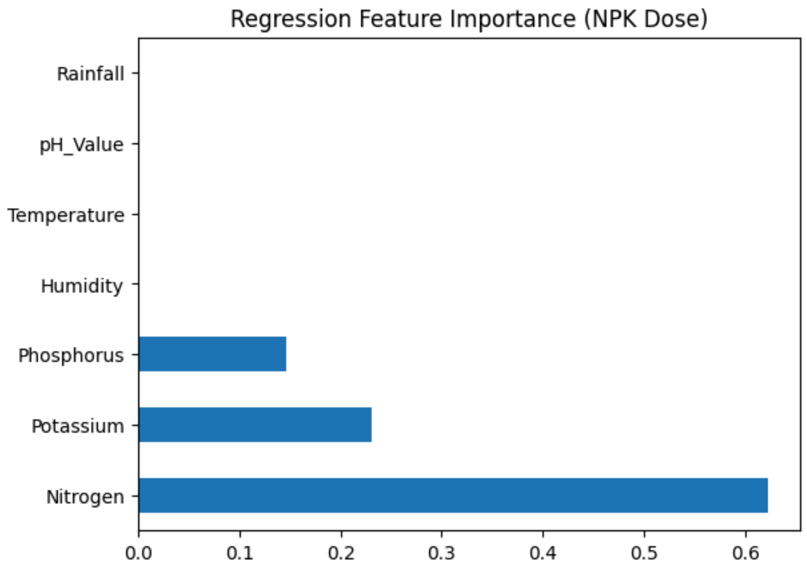
```
# One-hot encode categorical features (Soil_type, Crop_type)
X_cls = pd.get_dummies(X_cls)

# Train-test split
X_train_cls, X_test_cls, y_train_cls, y_test_cls = train_test_split(X_cls, y_cls, test_size=0.2, random_state=42)

# Train model
cls_model = RandomForestClassifier(n_estimators=100, random_state=42)
cls_model.fit(X_train_cls, y_train_cls)

# Predict & evaluate
y_pred_cls = cls_model.predict(X_test_cls)
print("Fertilizer Classification Accuracy:", accuracy_score(y_test_cls, y_pred_cls))
```

```
Fertilizer Classification Accuracy: 0.95
```

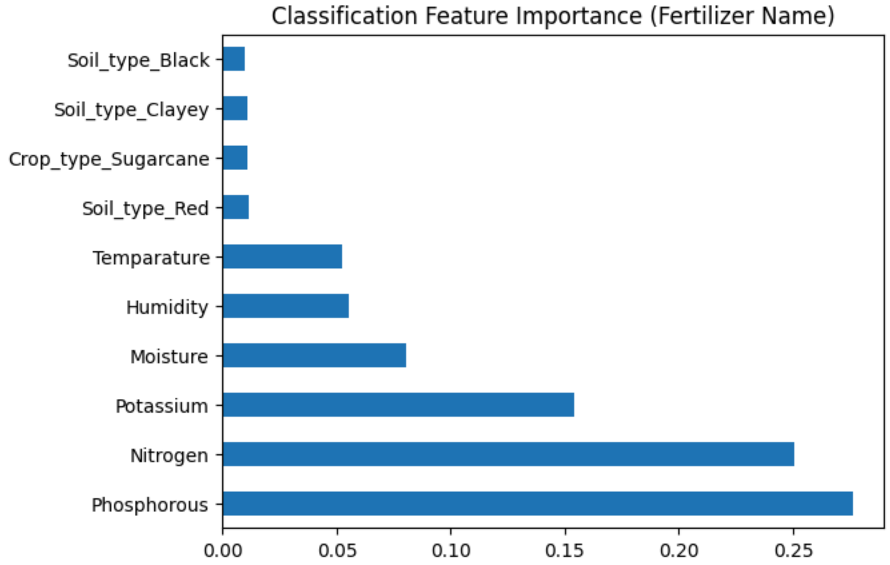
```
def recommend_fertilizer(input_data):
    """
    input_data: dict with keys - N,P,K,temperature,humidity,ph,rainfall,Moisture,Soil_type,Crop_type
    """
    # Dose prediction
    X_reg_input = pd.DataFrame([[
        input_data['N'], input_data['P'], input_data['K'],
        input_data['temperature'], input_data['humidity'],
        input_data['ph'], input_data['rainfall']
    ]], columns=X_reg.columns)

    npk_dose = reg_model.predict(X_reg_input)[0]

    # Fertilizer name prediction
    X_cls_input = pd.DataFrame([[
        input_data['temperature'], input_data['humidity'], input_data['Moisture'],
        input_data['Soil_type'], input_data['Crop_type'],
        input_data['N'], input_data['Potassium'], input_data['Phosphorous']
    ]], columns=['Temperature','Humidity','Moisture','Soil_type','Crop_type','Nitrogen','Potassium','Phosphorous'])

    X_cls_input = pd.get_dummies(X_cls_input)
    X_cls_input = X_cls_input.reindex(columns=X_cls.columns, fill_value=0)

    fert_name = cls_model.predict(X_cls_input)[0]

    return {
        "Recommended_N": round(npk_dose[0],2),
        "Recommended_P": round(npk_dose[1],2),
        "Recommended_K": round(npk_dose[2],2),
        "Fertilizer": fert_name
    }

# Example
test_input = {
    'N': 50, 'P': 30, 'K': 20,
    'temperature': 28, 'humidity': 70, 'ph': 6.5, 'rainfall': 200,
    'Moisture': 35, 'Soil_type': 'Sandy', 'Crop_type': 'Maize',
    'Potassium': 20, 'Phosphorous': 30
}

print(recommend_fertilizer(test_input))
```

```
{'Recommended_N': np.float64(71.22), 'Recommended_P': np.float64(29.95), 'Recommended_K': np.float64(61.97), 'Fertilizer': 'Urea'}
```

```
feat_importances = pd.Series(reg_model.feature_importances_, index=X_reg.columns)
feat_importances.nlargest(7).plot(kind='barh')
plt.title("Regression Feature Importance (NPK Dose)")
plt.show()

feat_importances_cls = pd.Series(cls_model.feature_importances_, index=X_cls.columns)
feat_importances_cls.nlargest(10).plot(kind='barh')
plt.title("Classification Feature Importance (Fertilizer Name)")
plt.show()
```

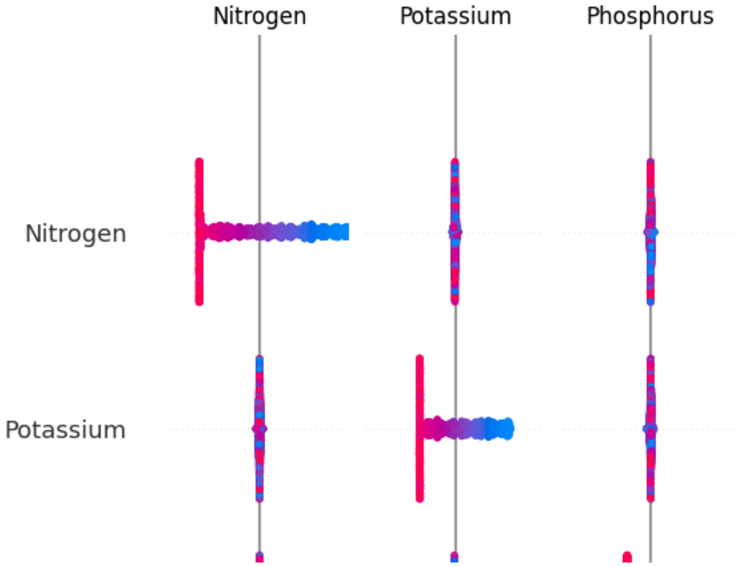


Regression Feature Importance (NPK Dose)



Classification Feature Importance (Fertilizer Name)

```
explainer = shap.TreeExplainer(reg_model)
shap_values = explainer.shap_values(X_test_reg)

shap.summary_plot(shap_values, X_test_reg, feature_names=X_reg.columns)
```


Nitrogen
Potassium
Phosphorus
Nitrogen
Potassium