Crop Recommendation Model Using Random Forest and LIME

Introduction

This project implements a **Crop Recommendation System** using a **Random Forest Classifier**. The model is trained on a dataset of soil and environmental factors to predict the most suitable crop. Additionally, **LIME (Local Interpretable Model-Agnostic Explanations)** is used to provide interpretability for individual predictions.

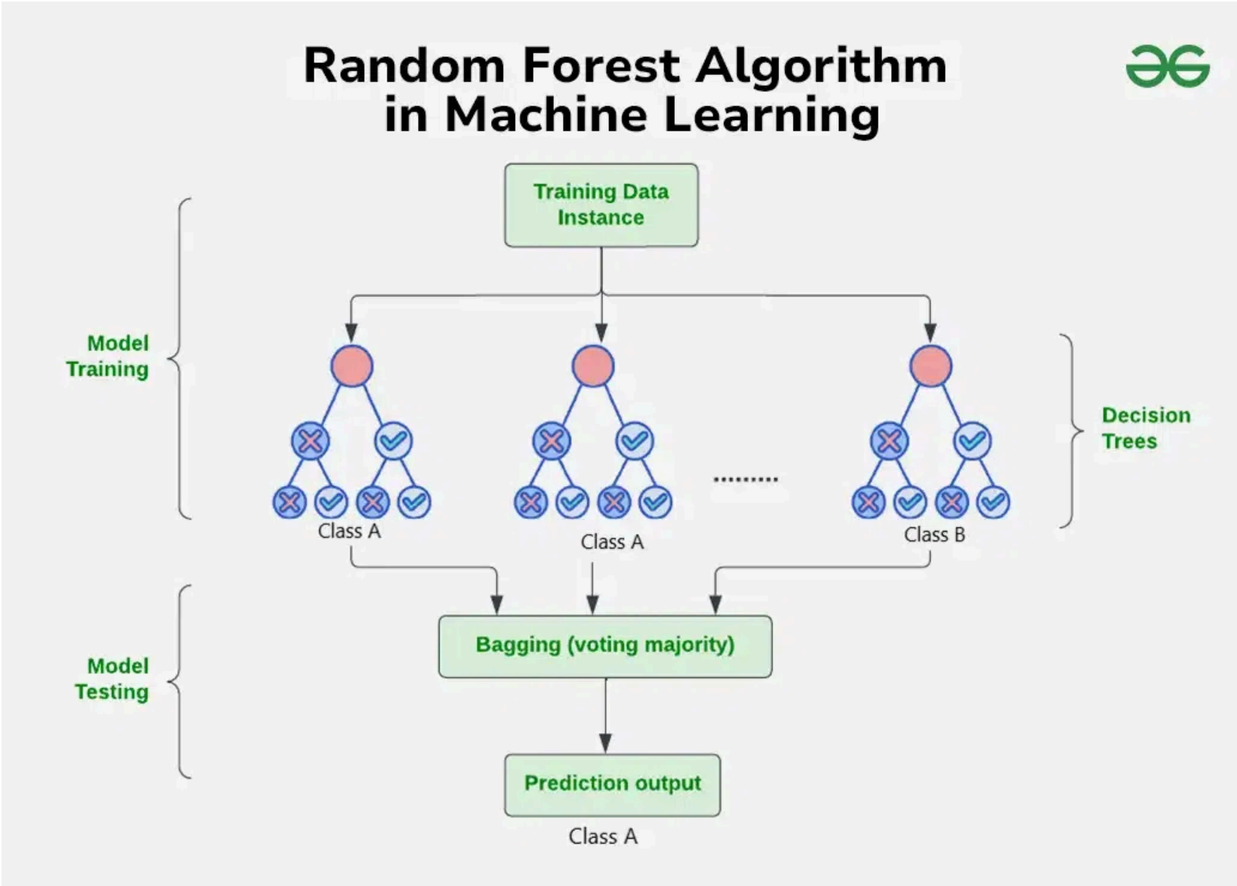
1. Random Forest Classifier

1.1 What is Random Forest?

Random Forest is an **ensemble learning** method that builds multiple decision trees and combines their outputs to make a more accurate and stable prediction. It is widely used for **classification and regression** tasks due to its ability to handle large datasets and reduce overfitting.

1.2 How Random Forest Works?

1. **Bootstrapping (Random Sampling):**
    - The dataset is randomly sampled multiple times with replacement to create multiple **training subsets**.
2. **Building Decision Trees:**
    - Each subset is used to train an individual **Decision Tree**.
    - Instead of considering all features for a split, only a **random subset of features** is used at each decision node.
3. **Aggregating Predictions:**
    - For **classification problems**, Random Forest takes the **majority vote** from all the trees (i.e., the most common prediction).
    - For **regression problems**, it takes the **average prediction** of all trees.



Random forest working

Prerequisites

To run this code, ensure you have the following Python libraries installed:

```
pip install lime pandas numpy scikit-learn matplotlib
```

Dataset

The dataset is read from a CSV file named `Crop_recommendation.csv`, containing various soil and climate features. The target variable (`label`) represents the crop recommendation.

Steps in the Code

1. Import Necessary Libraries

```python
from IPython.display import display
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import lime
import lime.lime_tabular
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
import random
```

These libraries are required for data processing, model training, evaluation, and explainability.

2. Load Dataset

```python
df = pd.read_csv("/content/Crop_recommendation.csv")
X = df.drop(columns=["label"])
y = df["label"]
print(y.value_counts())
```

- `X` contains the feature variables (e.g., nitrogen, phosphorus, potassium levels, temperature, humidity, etc.).
- `y` contains the target variable (`label`) representing the recommended crop.

3. Train-Test Split

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

- Splits the dataset into 80% training and 20% testing.

4. Train a Random Forest Model

```python
rf_model = RandomForestClassifier(n_estimators=500,
random_state=42)
rf_model.fit(X_train, y_train)
```

- A **Random Forest Classifier** is trained with 500 trees.

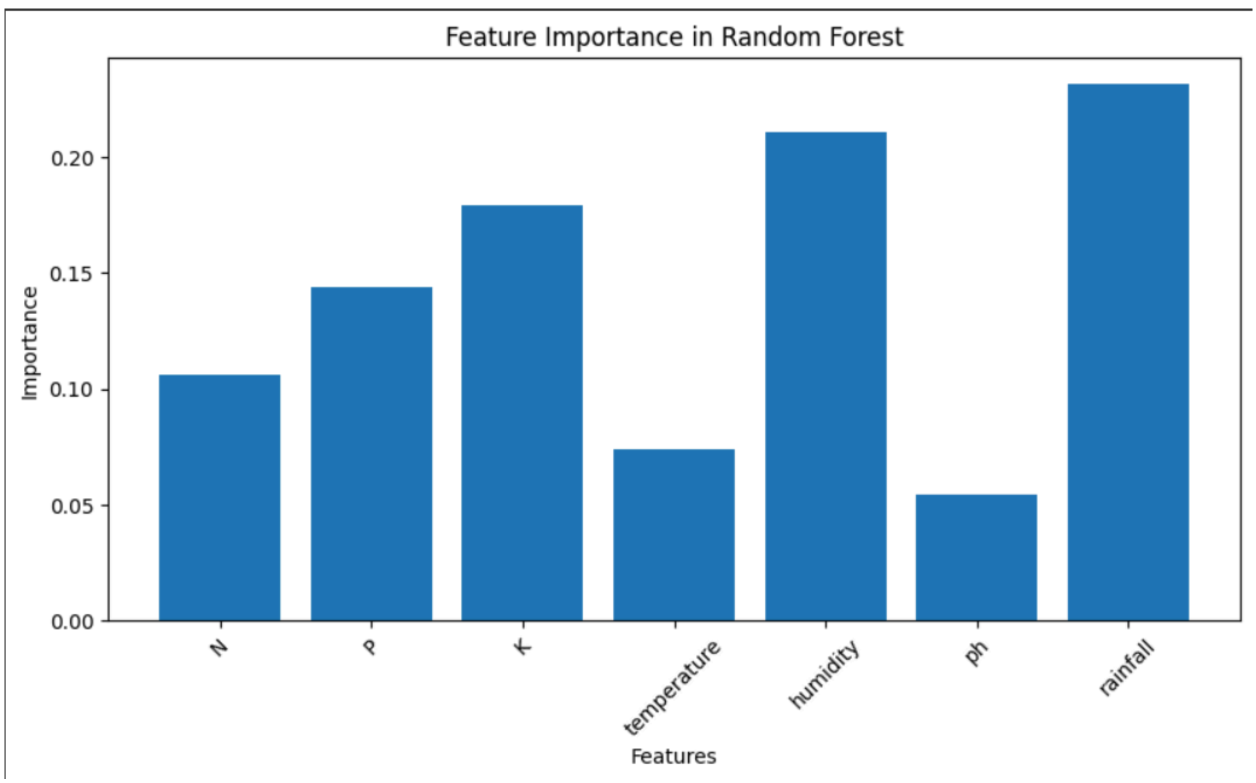
5. Model Evaluation

```python
y_pred = rf_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

Model Accuracy

6. Feature Importance Visualization

```python
feature_importances = rf_model.feature_importances_
plt.figure(figsize=(10, 5))
plt.bar(X.columns, feature_importances)
plt.xlabel("Features")
plt.ylabel("Importance")
plt.title("Feature Importance in Random Forest")
plt.xticks(rotation=45)
plt.show()
```



Feature Importance in Random Forest

2. LIME (Local Interpretable Model-agnostic Explanations)

2.1 What is LIME?

LIME (Local Interpretable Model-agnostic Explanations) is an algorithm designed to explain the **predictions** of any machine learning model in a human-understandable way. It is **model-agnostic**, meaning it can work with any machine learning model (e.g., Random Forest, Neural Networks, SVM, etc.).

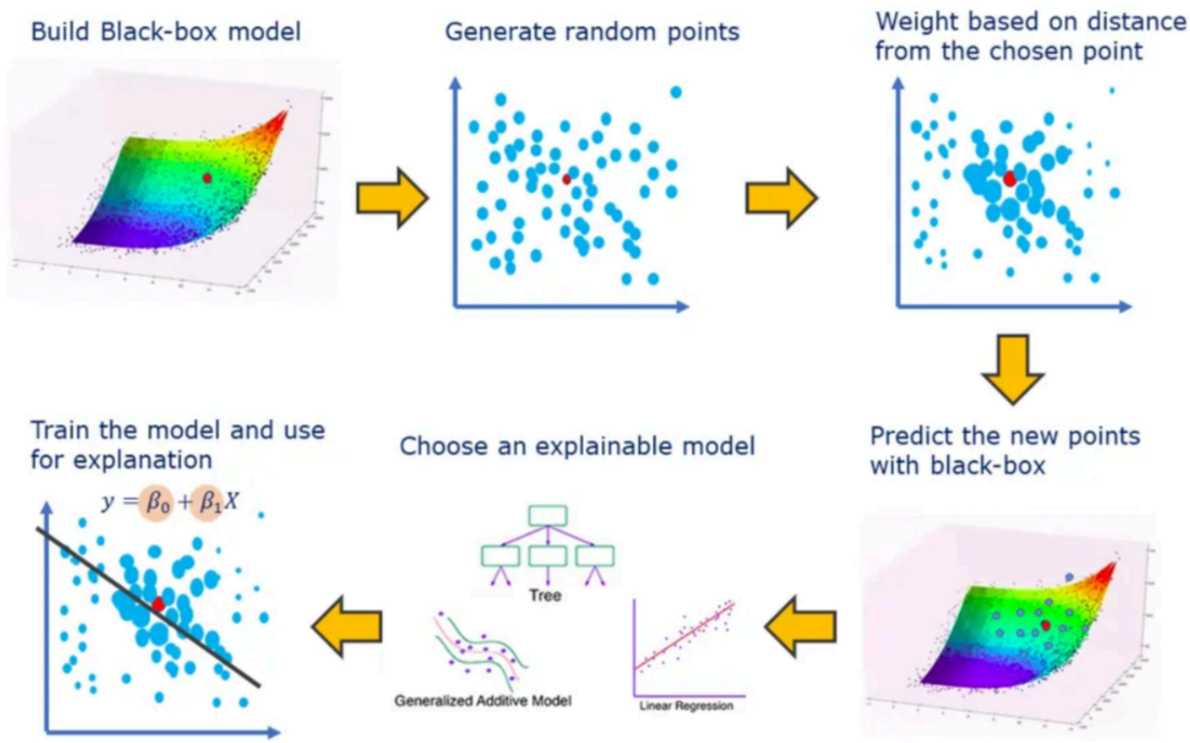
2.2 Why Do We Need LIME?

Many ML models, especially ensemble models like **Random Forest** and deep learning models, are considered **black-box models** because their decision-making process is complex. LIME helps explain how individual predictions are made by these models.

2.3 How LIME Works?

LIME provides an explanation by following these steps:

1. **Select an Instance:**
    - Pick one data point (test sample) whose prediction you want to explain.
2. **Generate Perturbed Data:**
    - LIME creates many small **perturbed** variations of the selected instance by making slight changes to the feature values.
3. **Get Predictions from the Black-Box Model:**
    - Each perturbed sample is fed into the original ML model to see how the model's prediction changes.
4. **Train an Interpretable Model:**
    - LIME trains a **simple, interpretable model** (like linear regression) on the perturbed samples.
    - This simple model approximates how the complex model behaves for this specific instance.
5. **Feature Importance Calculation:**
    - LIME assigns importance scores to each feature, showing how much each one contributed to the prediction.



Steps of the LIME algorithm.

7. Label Encoding for LIME

```python
le = LabelEncoder()
y_train_encoded = le.fit_transform(y_train)
y_test_encoded = le.transform(y_test)
```

- Converts categorical crop labels into numerical values for LIME explainability.

8. Create LIME Explainer

```python
explainer = lime.lime_tabular.LimeTabularExplainer(
    training_data=X_train.values,
    feature_names=X.columns,
    class_names=le.classes_,
```

```
    mode="classification"
)
```

- LIME explainer is initialized with training data.

9. Select a Random Test Instance & Make Prediction

```python
i = random.randint(0, 430)
instance = X_test.iloc[i].values.reshape(1, -1)
predicted_value = rf_model.predict(instance)[0]
predicted_class_index = le.transform([predicted_value])[0]
```

- Selects a random test sample and predicts its crop label.

10. Explain the Prediction using LIME

```python
exp = explainer.explain_instance(
    data_row=instance.flatten(),
    predict_fn=rf_model.predict_proba,
    labels=[predicted_class_index]
)
exp.show_in_notebook()
```

- Generates an **interpretable explanation** of the model's prediction for the selected instance using LIME.

Outputs:

Observation: The model predicts pigeon peas with 54% probability, but confidence is moderate. Other possible crops include blackgram (19%) and mothbeans (12%).

Key influencing features for pigeon peas: low potassium (≤ 20.00), nitrogen (20-37), moderate rainfall (64.09 - 94.30), and humidity (≤ 60.28).

The prediction is less certain compared to the previous case, suggesting overlapping feature conditions for multiple crops.

Observation The model predicts **pomegranate with 87% confidence**, with **coconut (6%)** as the next likely option.

Key influencing factors: **low phosphorus (≤ 28), potassium (31-48), moderate nitrogen (≤ 37), high rainfall (94.30 - 103.88), and humidity (> 89.90)**.

The **high pH (7.01)** slightly deviates from the influencing range but does not significantly impact the prediction.

The model shows **high confidence**, indicating strong feature alignment with pomegranate-growing