Name of Project

CROP AND FERETILISER RECOMMENDATION USING ML

Student Name: AAYUSH RAJ
AICTE Student_Id: **STU6447ad55804fe1682419029**

Learning Objectives

PYTHON
MACHINE LEARNING:

- DECISION TREE CLASSIFIER
- RECOMMENDATION:
  a. CROP RECOMMENDATION
  b. FERTILISER RECOMMENDATION

Source : www.freepik.com/

Tools and Technology used

PROGRAMMING LANGUAGE:

- PYTHON

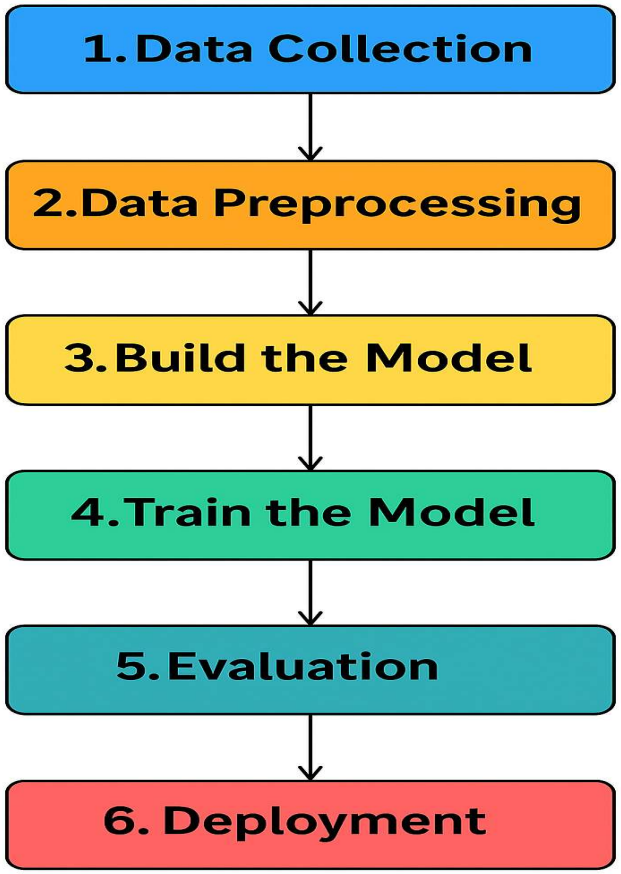
ENVIROMENT:

- VS CODE

MACHINE LEARNING LIBRARY:

- NUMPY
- PANDAS
- MATPLOTLIB
- SEABORN
- SKLEARN
- SCIKIT-LEARN

Methodology



Problem Statement:

Indian farmers often face challenges in selecting the right crop and fertilizer due to limited access to expert agronomic advice and inconsistent environmental conditions. This leads to suboptimal yields, resource wastage, and increased input costs. The objective of this project is to develop a machine learning-based system that analyzes soil nutrients (N, P, K), pH levels, and rainfall data to recommend the most suitable crop and corresponding fertilizer. The solution aims to empower farmers with data-driven decisions for sustainable and efficient farming.

Solution:

This project integrates **Green Skills with Artificial Intelligence (AI)** to promote sustainable agriculture through precision farming. By leveraging **machine learning**, the system empowers farmers to make environmentally responsible and data-driven decisions for crop and fertilizer selection.

Using the **Decision Tree Classifier**, a supervised ML algorithm, I built a robust model trained on key soil parameters such as Nitrogen (N), Phosphorus (P), Potassium (K), pH, and rainfall.

The model achieved:

- ✅ **98.49% accuracy in crop prediction**
- ✅ **100% accuracy in fertilizer recommendation**

These high-accuracy predictions ensure:

- **Reduced chemical overuse** through precise fertiliser application
- **Optimized land use and yield**, aligning with eco-friendly farming practices
- **Lower input costs** and **better returns for farmers**

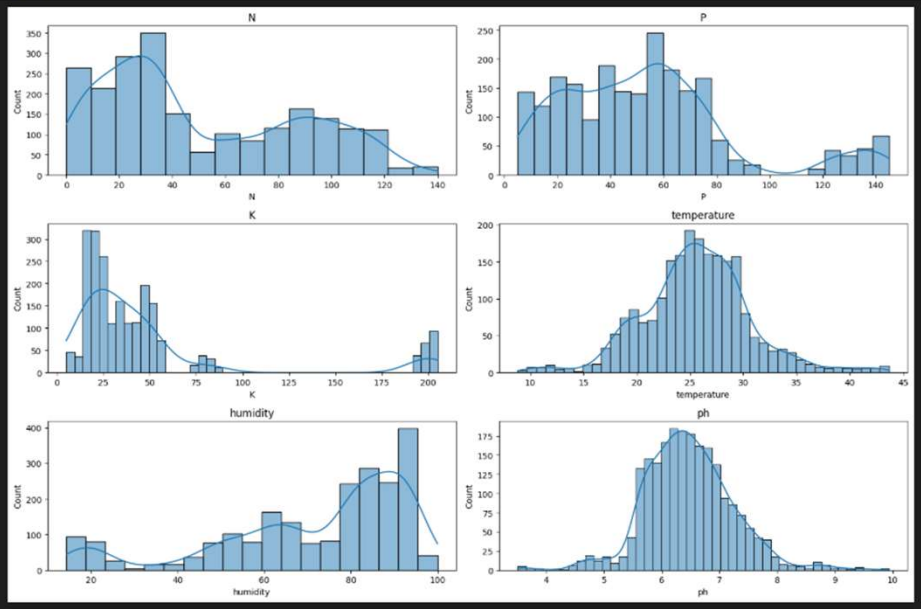
❖ **GitHub Repo:** SHELL-EDUNET-SKILLS4FUTURE-INTERNSHIP-2025.git

Screenshot of Output:

- **For Crop Recommendation:**

```
# visualize the distribution of each feature
fig, ax = plt.subplots(3, 2, figsize=(15, 10))
for i, subplot in zip(features, ax.flatten()):
    sns.histplot(crop[i], ax=subplot, kde=True)
    subplot.set_title(i)
plt.tight_layout()
plt.show()
```

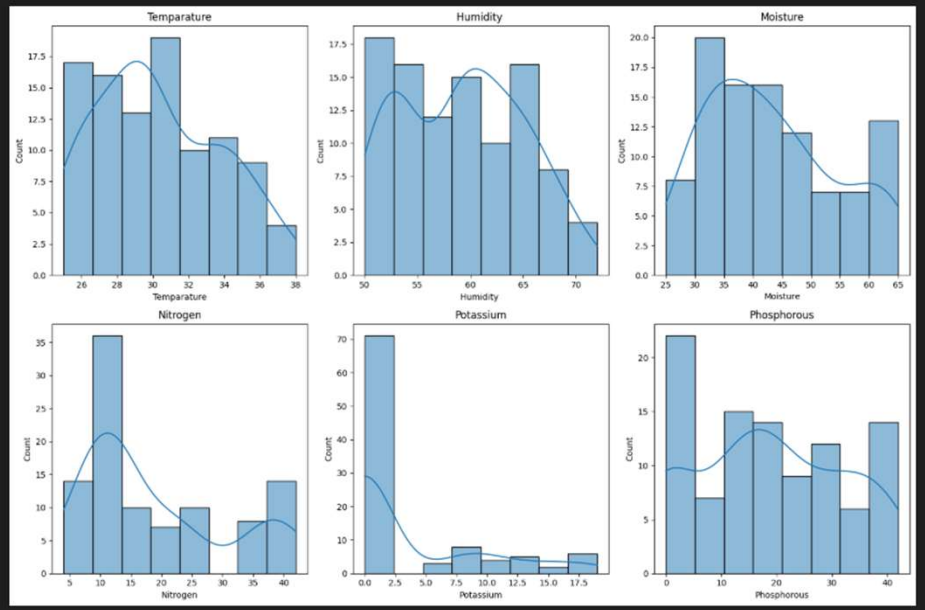
[11] ✓ 1.8s Python



- **For Fertilizer Recommendation:**

```
# visualize the distribution of each feature
fig, ax = plt.subplots(2, 3, figsize=(15, 10))
for i, subplot in zip(features, ax.flatten()):
    sns.histplot(fertilizer[i], ax=subplot, kde=True)
    subplot.set_title(i)
plt.tight_layout()
plt.show()
```

[13] Python

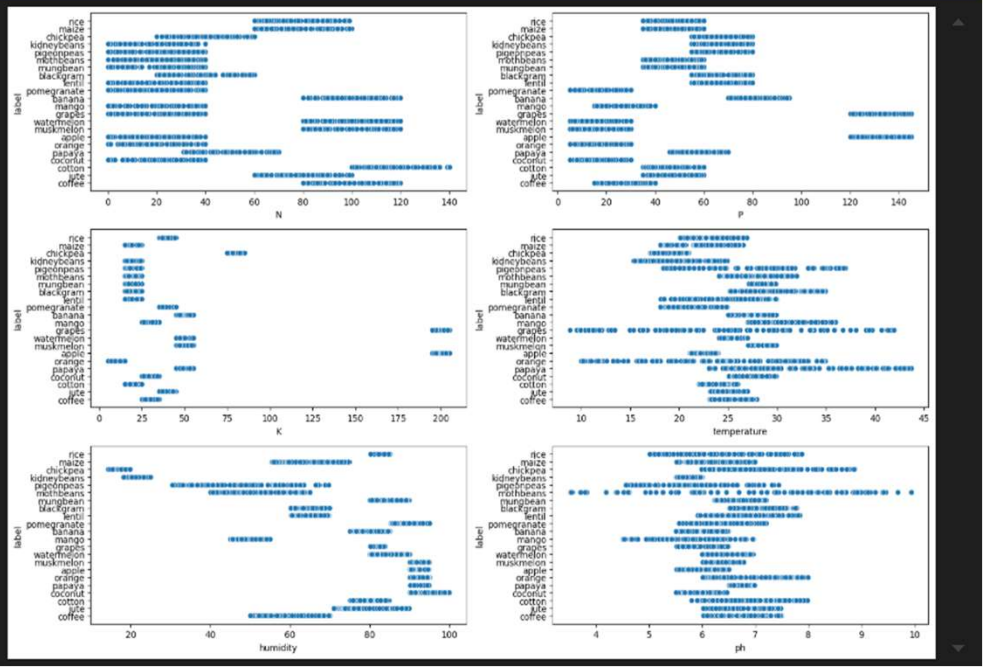


Screenshot of Output:

- **For Crop Recommendation:**

```
# plot scatter plot of each feature against the target
fig, ax = plt.subplots(3, 2, figsize=(15, 10))
for i, subplot in zip(features, ax.flatten()):
    sns.scatterplot(x=i, y='label', data=crop, ax=subplot)
plt.tight_layout()
plt.show()
```

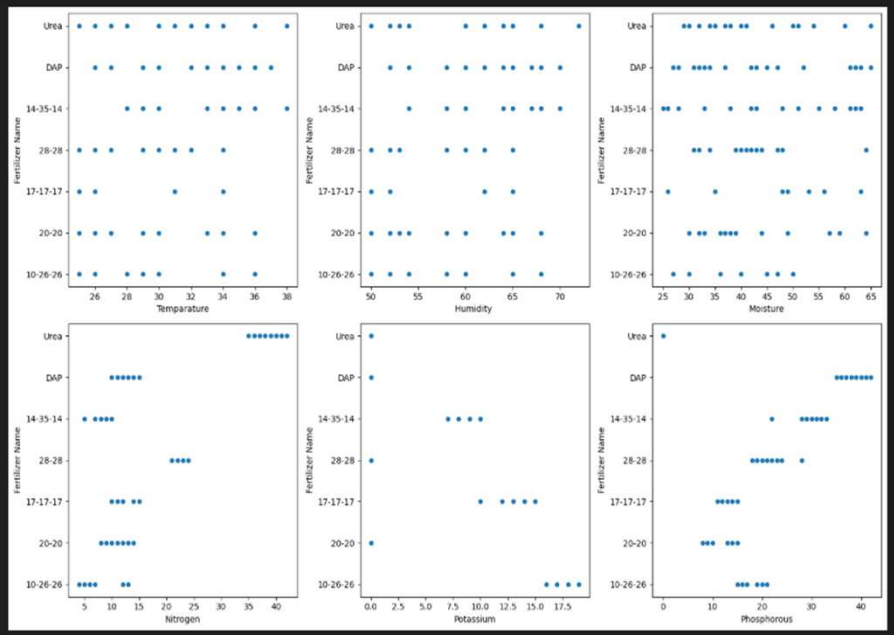
[79] ✓ 1.1s Python



- **For Fertilizer Recommendation:**

```
# plot scatter plot of each feature against the target
fig, ax = plt.subplots(2, 3, figsize=(15, 10))
for i, subplot in zip(features, ax.flatten()):
    sns.scatterplot(x=i, y='Fertilizer Name', data=fertilizer, ax=subplot)
plt.tight_layout()
plt.show()
```

[14] Python

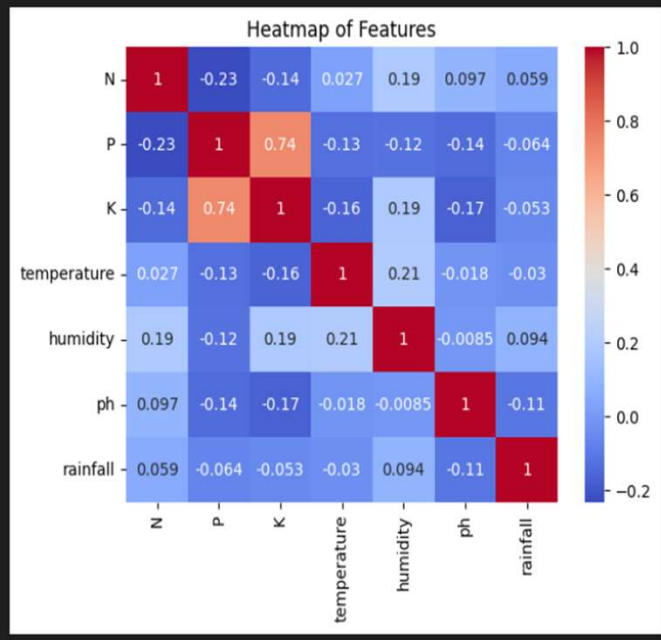


Screenshot of Output:

- **For Crop Recommendation:**

```
# Visualize the correlation matrix
import seaborn as sns
sns.heatmap(corr,annot=True,cbar=True,cmap='coolwarm')
plt.title("Heatmap of Features")
plt.show()
```

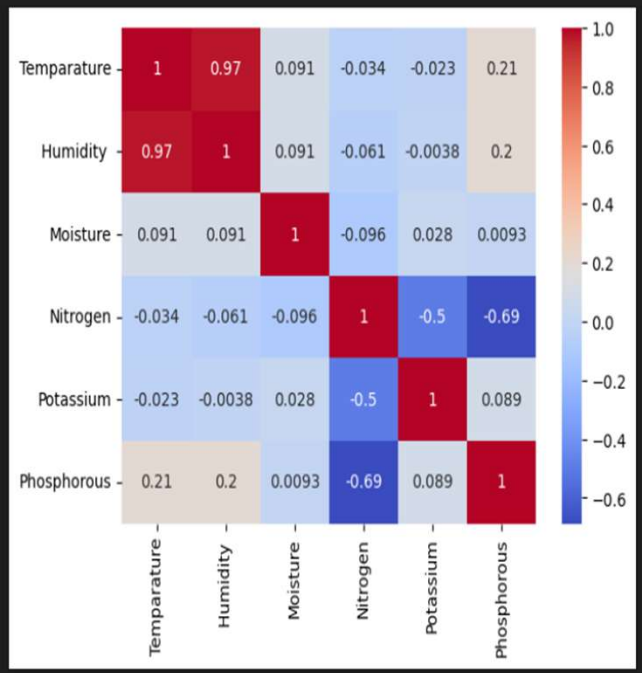
Python



- **For Fertilizer Recommendation:**

```
sns.heatmap(corr,annot=True,cbar=True,cmap='coolwarm')
plt.show()
```

[9] Python



Screenshot of Output:

- **For Crop Recommendation:**

Training Models

```
# Let's Train a Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
```
[58] Python

```
# intialize the DecisionTree Classifier
DT = DecisionTreeClassifier()
# Train the model
DT.fit(x_train,y_train)
```
[59] Python

```
DecisionTreeClassifier()
```

```
# evaluate the model on test dataset
y_pred = DT.predict(x_test)
print(f"Accuracy score of Decision Tree Classifier is: {accuracy_score(y_test,y_pred)}")
```
[60] Python

```
Accuracy score of Decision Tree Classifier is: 0.9886363636363636
```

```
# evaluate the model on train dataset
y_pred_train = DT.predict(x_train)
print(f"Accuracy score of Decision Tree Classifier is: {accuracy_score(y_train,
y_pred_train)}")
```
[61] Python

```
Accuracy score of Decision Tree Classifier is: 1.0
```

- **For Fertilizer Recommendation:**

Training Models

```
# initialize the DecisionTreeClassifier
model = DecisionTreeClassifier()
```
[25] Python

```
# train the model
model.fit(x_train, y_train)
```
[26] Python

```
DecisionTreeClassifier()
```

```
# evaluate the model on the test set and print the accuracy
accuracy = model.score(x_test, y_test)
print(f"The accuracy of the model is: {accuracy*100:.2f}%")
```
[27] Python

```
The accuracy of the model is: 100.00%
```

```
# evaluate the model on the training set and print the accuracy
accuracy = model.score(x_train, y_train)
print(f"The accuracy of the model on the training set is: {accuracy*100:.2f}%")
```
[28] Python

```
The accuracy of the model on the training set is: 100.00%
```

Conclusion:

This project stands as a testament to the transformative potential of merging **machine learning** with **eco-conscious agricultural practices**. By building an intelligent system that deciphers complex soil and environmental data, I was able to guide farmers toward informed choices—minimizing guesswork and maximizing productivity.

More than just a technical achievement, this initiative fosters **climate-smart farming** by encouraging responsible resource use and reducing environmental impact. It empowers farmers with practical, science-backed insights while aligning with broader sustainability goals.

Future Scope:

- **Regional Customization**: Train models for specific climates and soil types.
- **IoT Integration**: Use real-time data from sensors for smarter predictions.
- **Mobile & Multilingual Access**: Develop apps with voice and language support.
- **Smart Fertiliser Scheduling**: Suggest timing and quantity, not just type.
- **Sustainability Alerts**: Notify farmers of weather risks and eco-friendly tips.
- **Govt. Scheme Integration**: Connect with agri-portals and subsidy programs.
- **Advanced ML Models**: Explore deep learning for improved accuracy.