In [ ]:

```
devlop an ai based system that predicts where crop is healthy or not using agricultur
```

In [49]:

```
import pandas as pd
df = pd.read_csv("../admin1/Downloads/Crop_recommendation (1).csv")
df
```

Out[49]:

	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
...	...	...	...	...	...	...	...	...
2195	107	34	32	26.774637	66.413269	6.780064	177.774507	coffee
2196	99	15	27	27.417112	56.636362	6.086922	127.924610	coffee
2197	118	33	30	24.131797	67.225123	6.362608	173.322839	coffee
2198	117	32	34	26.272418	52.127394	6.758793	127.175293	coffee
2199	104	18	30	23.603016	60.396475	6.779833	140.937041	coffee

2200 rows × 8 columns

In [13]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

df = pd.read_csv("../admin1/Downloads/Crop_recommendation (1).csv")
X = df[['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall']]
y = df['label']

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

model = RandomForestClassifier(n_estimators=100)
model.fit(X_train, y_train)

acc = accuracy_score(y_test, model.predict(X_test))
print(f"Accuracy: {acc:.2%}")

current_crop = "rice"
test_data = pd.DataFrame([[90, 42, 43, 20, 82, 6, 202]], columns=X.columns)
prediction = model.predict(test_data)[0]

is_healthy = prediction.lower() == current_crop.lower()
result_status = "Healthy" if is_healthy else "Unhealthy"
recommendation = f"Optimal for {current_crop}" if is_healthy else f"Better suit {pred

print(f"{result_status}: {recommendation}")
```

```
Accuracy: 99.32%
Healthy: Optimal for rice
```

In [75]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

```
df = pd.read_csv("../admin1/Downloads/Crop_recommendation (1).csv")
X = df[['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall']]
y = df['label']

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

model = LogisticRegression()
model.fit(X_train, y_train)

acc = accuracy_score(y_test, model.predict(X_test))
print(f"Accuracy: {acc:.2%}")

current_crop = "rice"
test_data = pd.DataFrame([[90, 42, 43, 20, 82, 6, 202]], columns=X.columns)
prediction = model.predict(test_data)[0]

is_healthy = prediction.lower() == current_crop.lower()
result_status = "Healthy" if is_healthy else "Unhealthy"
recommendation = f"Optimal for {current_crop}" if is_healthy else f"Better suit {pred

print(f"{result_status}: {recommendation}")
```

```
Accuracy: 94.55%
Healthy: Optimal for rice
```

```
/home/admin1/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:4
60: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
In [ ]:
```