Step 1. Import Libraries :

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report
from sklearn import metrics, tree
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
import pickle
import warnings
```

Step 2. Load Dataset :

```
PATH = 'D:\comp_studies\Research project\Part 1 data\Crop_recommendation.csv'
df = pd.read_csv(PATH)
```

Step 3. Preprocess Data :

```
le = LabelEncoder()
df['label'] = le.fit_transform(df['label'])
```

Step 4. Separate Features and Target :

```
features = df[['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall']]
target = df['label']
labels = df['label']
```

Step 5. Split Data :

```
from sklearn.model_selection import train_test_split
Xtrain, Xtest, Ytrain, Ytest = train_test_split(features, target, test_size=0.2,
random_state=2)
```

Step 6. Train and Evaluate Models :

a) Random Forest :

```
rf_filename = 'random_forest_model.pkl'
try:
    RF = load_model(rf_filename)
except FileNotFoundError:
        RF = RandomForestClassifier(n_estimators=20, random_state=0)
        RF.fit(Xtrain, Ytrain)
        save_model(RF, rf_filename)
predicted_values = RF.predict(Xtest)
x = metrics.accuracy_score(Ytest, predicted_values)
print("Random Forest Classifier's Accuracy is: ", x*100,"%")
```

b) Gaussian Naive Bayes :

```
nb_filename = 'naive_bayes_model.pkl'
try:
        NaiveBayes = load_model(nb_filename)
except FileNotFoundError:
        NaiveBayes = GaussianNB()
        NaiveBayes.fit(Xtrain, Ytrain)
        save_model(NaiveBayes, nb_filename)
predicted_values = NaiveBayes.predict(Xtest)
x = metrics.accuracy_score(Ytest, predicted_values)
print("Naive Bayes's Accuracy is: ", x*100,"%")
```

c) XGBoost :

```
xgb_filename = 'xgboost_model.pkl'
try:
        XGB = load_model(xgb_filename)
except FileNotFoundError:
        from xgboost import XGBClassifier
        XGB = XGBClassifier()
        XGB.fit(Xtrain, Ytrain)
        save_model(XGB, xgb_filename)
predicted_values = XGB.predict(Xtest)
x = metrics.accuracy_score(Ytest, predicted_values)
print("XGBoost's Accuracy is: ", x*100," %")
```

Step 7. Predict Crop Recommendation :

```
l = []
n = int(input("Enter the value of nitrogen content: "))
p = int(input("Enter the value of phosphorus content: "))
k = int(input("Enter the value of potassium content: "))
temperature = float(input("Enter the value of temperature: "))
humidity = float(input("Enter the value of humidity: "))
```

```
ph = float(input("Enter the value of pH: "))
rain = float(input("Enter the value of Rainfall: "))
data = np.array([[n, p, k, temperature, humidity, ph, rain]])
prediction = XGB.predict(data)
l.append(prediction[0])
print("The crop that should be planted is: ", le.inverse_transform([l[0]])[0])
```