```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score, roc_curve, auc, f1_score

# Loading dataset
df = pd.read_csv('crop.csv')

df.isnull().sum()

le = LabelEncoder()

df['State'] = le.fit_transform(df['State'])
df['District'] = le.fit_transform(df['District'])
df['Crop'] = le.fit_transform(df['Crop'])
df['Season'] = le.fit_transform(df['Season'])

X = df.drop(['Production'], axis=1)
y = df['Production']

# Drop rows with NaN values in y
df.dropna(subset=['Production'], inplace=True)
X = df.drop(['Production'], axis=1)
y = df['Production']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.002,
random_state=20)

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Decision Tree
dt = DecisionTreeRegressor()
dt.fit(X_train, y_train)
y_pred_dt = dt.predict(X_test)

# Evaluation
print("Decision Tree Regressor:")
print("Accuracy:", accuracy_score(y_test, y_pred_dt))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_dt))
print("Classification Report:\n", classification_report(y_test, y_pred_dt))
print("F1 Score:", f1_score(y_test, y_pred_dt, average='macro'))

# KNN
knn = KNeighborsClassifier(n_neighbors=3)
```

```
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)

# Evaluation
print("KNN:")
print("Accuracy:", accuracy_score(y_test, y_pred_knn))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_knn))
print("Classification Report:\n", classification_report(y_test, y_pred_knn))
print("F1 Score:", f1_score(y_test, y_pred_knn, average='macro'))
```