```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from matplotlib import pyplot as plt

data_crop=pd.read_csv("/content/drive/MyDrive/crop growth1.csv")

data_crop.shape
```

```
(10, 6)
```

```
data_crop.head()
```


```
{"summary":"{\n  \"name\": \"data_crop\",\n  \"rows\": 10,\n
\"fields\": [\n    {\n      \"column\": \"Soil_Type\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 10,\n        \"samples\": [\n          \"Sandy
Clay\",\n          \" Sandy \",\n          \"Loamy Sand\"\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Ph\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.6342099196813483,\n
\"min\": 5.5,\n        \"max\": 7.5,\n        \"num_unique_values\":
10,\n        \"samples\": [\n          7.5,\n          7.2,\n
7.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Rainfall (mm)\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 186,\n        \"min\": 150,\n
\"max\": 700,\n        \"num_unique_values\": 9,\n        \"samples\":
[\n          650,\n          300,\n          200\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Temperature\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
7,\n        \"min\": 18,\n        \"max\": 40,\n
\"num_unique_values\": 10,\n        \"samples\": [\n          38,\n
35,\n          40\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Humidity\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 15,\n        \"min\": 40,\n
\"max\": 85,\n        \"num_unique_values\": 10,\n        \"samples\":
[\n          45,\n          50,\n          40\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Crop_Suitable\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n          \"no\",\n
\"yes\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"data_crop"}
```

```
data = data_crop.drop(["Soil_Type","Rainfall
(mm)","Temperature"],axis=1)
data.head(10)
```


```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 10,\n  \"fields\":
[\n    {\n      \"column\": \"Ph\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.6342099196813483,\n
\"min\": 5.5,\n        \"max\": 7.5,\n        \"num_unique_values\":
10,\n        \"samples\": [\n          7.5,\n          7.2,\n
7.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Humidity\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 15,\n        \"min\": 40,\n
\"max\": 85,\n        \"num_unique_values\": 10,\n        \"samples\":
[\n          45,\n          50,\n          40\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Crop_Suitable\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n          \"no\",\n
\"yes\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"data"}
```


```
x = data[["Ph", "Humidity"]]  # Pass column names as a list
y = data["Crop_Suitable"]
```

```
data.isnull().sum()
```

```
Ph                0
Humidity          0
Crop_Suitable     0
dtype: int64
```

```
k=3
knn=KNeighborsClassifier(n_neighbors=k)
knn.fit(x,y)
```

```
KNeighborsClassifier(n_neighbors=3)
```

```
new_data = np.array([[1, 35]])
prediction = knn.predict(new_data)

# Extract the pH value from new_data
ph_value = new_data[0][0]

# Check if the prediction indicates the crop can be grown
if ph_value > 5:
    print('Crop can be grown')
else:
    print('Crop cannot be grown')
```

```
Crop cannot be grown

/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439:
UserWarning: X does not have valid feature names, but
KNeighborsClassifier was fitted with feature names
  warnings.warn(
```

```
from sklearn.linear_model import LinearRegression
LR = LinearRegression()

from sklearn.model_selection import train_test_split as ttp
from sklearn.metrics import classification_report

# Assuming 'y' is your target variable and it contains strings like
'yes' and 'no'
# Convert 'yes' to 1 and 'no' to 0
y_numeric = y.replace({'yes': 1, 'no': 0})

# Now fit the model with the numeric target variable
LR.fit(x, y_numeric)
```

```
LinearRegression()
```

```
new_data = np.array([[6, 35]])
prediction = LR.predict(new_data)[0]

ph_value = new_data[0][0]

# Check if the prediction indicates the crop can be grown
if ph_value > 5:
    print('Crop can be grown')
else:
    print('Crop cannot be grown')
```

```
Crop can be grown

/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439:
UserWarning: X does not have valid feature names, but LinearRegression
was fitted with feature names
  warnings.warn(
```

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(x, y)
```

```
LogisticRegression()
```

```
y_pred = model.predict(x)
accuracy = accuracy_score(y, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.9
```

```
new_data = np.array([[1, 35]])
prediction = model.predict(new_data)

# Extract the pH value from new_data
ph_value = new_data[0][0]

# Check if the prediction indicates the crop can be grown
if ph_value > 5:
    print('Crop can be grown')
else:
    print('Crop cannot be grown')
```

```
Crop cannot be grown
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439:
UserWarning: X does not have valid feature names, but
RandomForestClassifier was fitted with feature names
  warnings.warn(
```

```
from sklearn.tree import DecisionTreeClassifier

dt_regressor = DecisionTreeClassifier(random_state=42)

dt_regressor.fit(x, y)
```

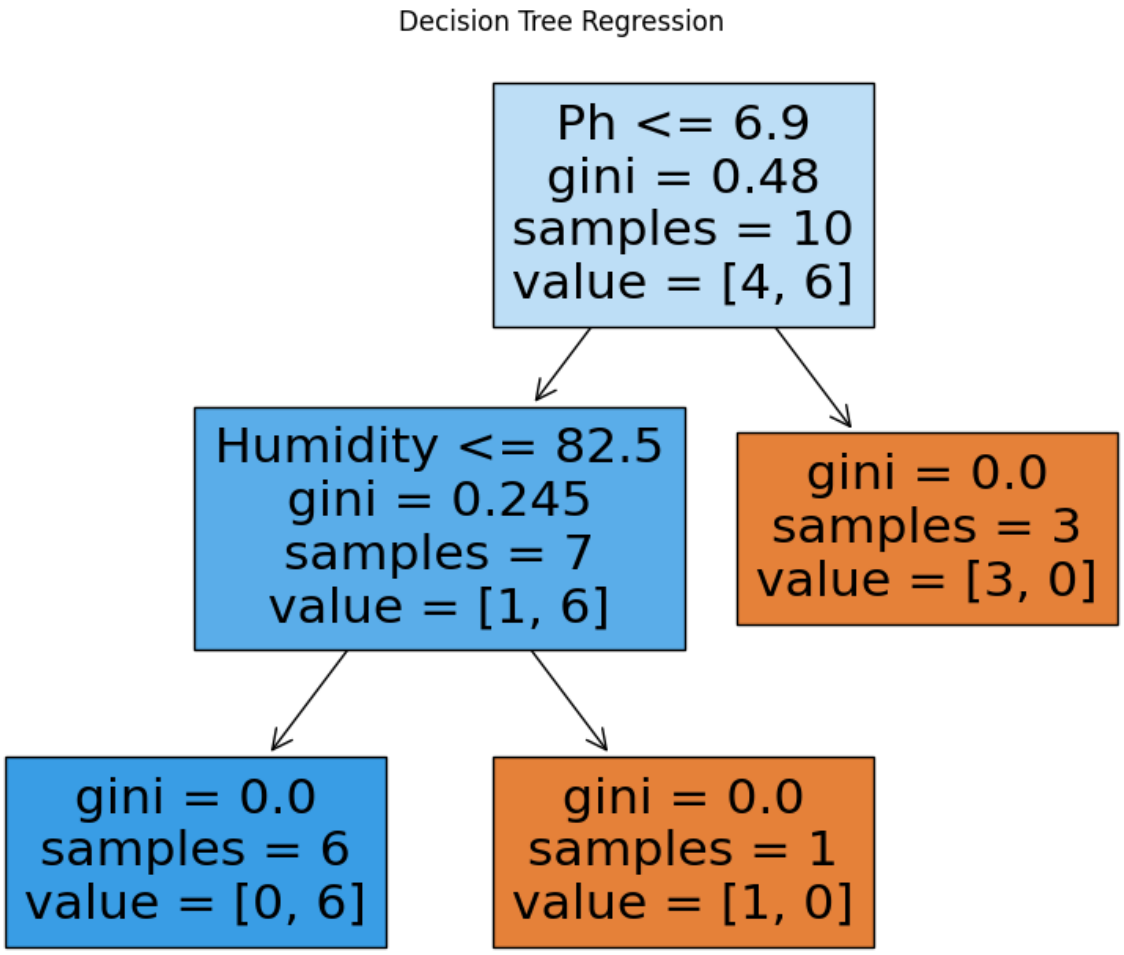
```
DecisionTreeClassifier(random_state=42)
```

```
y_pred = dt_regressor.predict(x)

y_pred = model.predict(x)
accuracy = accuracy_score(y, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 1.0
```

```
from sklearn.tree import plot_tree
plt.figure(figsize=(10, 8))
plot_tree(dt_regressor, feature_names=x.columns, filled=True)
plt.title("Decision Tree Regression")
plt.show()
```




```
from sklearn.ensemble import RandomForestRegressor

rf_regressor = RandomForestRegressor(n_estimators=100,
random_state=42)

y_numeric = y.replace({'yes': 1, 'no': 0})

# Now fit the model with the numeric target variable
rf_regressor.fit(x, y_numeric)
```

```
RandomForestRegressor(random_state=42)
```

```
y_pred = model.predict(x)
accuracy = accuracy_score(y, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 1.0
```

```
RandomForestRegressor(random_state=42)
# Predict on the test set
y_pred = rf_regressor.predict(x)

feature_importances = rf_regressor.feature_importances_
features = x.columns

plt.figure(figsize=(8, 6))
plt.bar(features, feature_importances, color='skyblue')
plt.xlabel('Features')
plt.ylabel('Importance')
plt.title('Feature Importance in Random Forest Model')
plt.show()
```