crop yield prediction model

In [1]:
```
import pandas as pd
import numpy as np
import sklearn

yield_df = pd.read_csv("yield_df.csv")
yield_df.head()
```

Out[1]:

	Unnamed: 0	Area	Item	Year	hg/ha_yield	ha_area	average_rain_fall_mm_per_year	pestc_l
0	0	Albania	Apples	2002	70222.0	2250.0	1485.0	
1	1	Albania	Beans	2002	78261.0	690.0	1485.0	
2	2	Albania	Grapes	2002	159746.0	5202.0	1485.0	
3	3	Albania	Maize	2002	39460.0	50000.0	1485.0	
4	4	Albania	Oranges	2002	68000.0	500.0	1485.0	

In [2]:
```
yield_df.drop(columns = ['Unnamed: 0','Year'], inplace=True)
```

In [3]:
```
yield_df.head()
```

Out[3]:

	Area	Item	hg/ha_yield	ha_area	average_rain_fall_mm_per_year	pestc_kg/ha	avg_temp
0	Albania	Apples	70222.0	2250.0	1485.0	0.47	16.47
1	Albania	Beans	78261.0	690.0	1485.0	0.47	16.47
2	Albania	Grapes	159746.0	5202.0	1485.0	0.47	16.47
3	Albania	Maize	39460.0	50000.0	1485.0	0.47	16.47
4	Albania	Oranges	68000.0	500.0	1485.0	0.47	16.47

In [4]:
```
yield_df.shape
```

Out[4]:
```
(38320, 10)
```

In [5]:

```
yield_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38320 entries, 0 to 38319
Data columns (total 10 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   Area                           38320 non-null  object 
 1   Item                           38320 non-null  object 
 2   hg/ha_yield                    38320 non-null  float64
 3   ha_area                        38320 non-null  float64
 4   average_rain_fall_mm_per_year  38320 non-null  float64
 5   pestc_kg/ha                    38320 non-null  float64
 6   avg_temp                       38320 non-null  float64
 7   fertN_kg/ha                    38320 non-null  float64
 8   fertP_kg/ha                    38320 non-null  float64
 9   fertK_kg/ha                    38320 non-null  float64
dtypes: float64(8), object(2)
memory usage: 2.9+ MB
```

In [6]:

```
yield_df_onehot = pd.get_dummies(yield_df, columns=['Area',"Item"], prefix = ['Co
features=yield_df_onehot.loc[:, yield_df_onehot.columns != 'hg/ha_yield']
label=yield_df['hg/ha_yield']
features.head()
```

Out[6]:

	ha_area	average_rain_fall_mm_per_year	pestc_kg/ha	avg_temp	fertN_kg/ha	fertP_kg/ha	fertK_
0	2250.0	1485.0	0.47	16.47	54.67	26.02	
1	690.0	1485.0	0.47	16.47	54.67	26.02	
2	5202.0	1485.0	0.47	16.47	54.67	26.02	
3	50000.0	1485.0	0.47	16.47	54.67	26.02	
4	500.0	1485.0	0.47	16.47	54.67	26.02	

5 rows × 122 columns

In [7]:
```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
features=scaler.fit_transform(features)

features
```

Out[7]:
```
array([[4.94099356e-05, 4.63927532e-01, 3.16285330e-02, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [1.51523802e-05, 4.63927532e-01, 3.16285330e-02, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [1.14235771e-04, 4.63927532e-01, 3.16285330e-02, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       ...,
       [1.54027239e-04, 1.96053057e-01, 4.17227456e-02, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [2.76074172e-03, 1.96053057e-01, 4.17227456e-02, ...,
        1.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [4.77695257e-04, 1.96053057e-01, 4.17227456e-02, ...,
        0.00000000e+00, 0.00000000e+00, 1.00000000e+00]])
```

In [8]:
```
features.shape
```

Out[8]:
```
(38320, 122)
```

In [9]:
```
label.shape
```

Out[9]:
```
(38320,)
```

In [10]:
```
from sklearn.model_selection import train_test_split
train_data, test_data, train_labels, test_labels = train_test_split(features, lab
```

In [11]:
```
from sklearn.metrics import r2_score
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.tree import DecisionTreeRegressor

m1 = DecisionTreeRegressor()
decisiontree_model = m1.fit(train_data,train_labels)
y_pred = decisiontree_model.predict(test_data)
r2 = r2_score(test_labels,y_pred)
print('DecisionTreeRegressor ',r2)

m2 = RandomForestRegressor(n_estimators=200, max_depth=6, random_state=0)
randomforest_model = m2.fit(train_data,train_labels)
y_pred = randomforest_model.predict(test_data)
r2 = r2_score(test_labels,y_pred)
print('RandomForestRegressor ',r2)

m3 = GradientBoostingRegressor(n_estimators=200, max_depth=6, random_state=0)
gradientboost_model = m3.fit(train_data,train_labels)
y_pred = gradientboost_model.predict(test_data)
r2 = r2_score(test_labels,y_pred)
print('GradientBoostingRegressor ',r2)
```

```
DecisionTreeRegressor  0.9839319415584543
RandomForestRegressor  0.8819168653931477
GradientBoostingRegressor  0.9738480259438032
```

In [12]:
```
import pickle
filename = 'yield_model.sav'
pickle.dump(decisiontree_model, open(filename, 'wb'))
```

crop prediction model

In [24]:
```
data = pd.read_csv('cpdata.csv')
data.head()
```

Out[24]:

	Unnamed: 0	temperature	humidity	ph	rainfall	label
0	0	20.879744	82.002744	6.502985	202.935536	rice
1	1	21.770462	80.319644	7.038096	226.655537	rice
2	2	23.004459	82.320763	7.840207	263.964248	rice
3	3	26.491096	80.158363	6.980401	242.864034	rice
4	4	20.130175	81.604873	7.628473	262.717340	rice

In [25]:
```
data.shape
```

Out[25]:
```
(3100, 6)
```

In [26]:
```
data.drop(columns = ['Unnamed: 0'], inplace=True)
data.head()
```

Out[26]:

	temperature	humidity	ph	rainfall	label
0	20.879744	82.002744	6.502985	202.935536	rice
1	21.770462	80.319644	7.038096	226.655537	rice
2	23.004459	82.320763	7.840207	263.964248	rice
3	26.491096	80.158363	6.980401	242.864034	rice
4	20.130175	81.604873	7.628473	262.717340	rice

In [27]:
```
data.label.unique()
```

Out[27]:
```
array(['rice', 'wheat', 'Mung Bean', 'Tea', 'millet', 'maize', 'Lentils',
       'Jute', 'Coffee', 'Cotton', 'Groundnuts', 'Peas', 'Rubber',
       'Sugar cane', 'Tobacco', 'Kidney Beans', 'Moth Beans', 'Coconut',
       'Black gram', 'Adzuki Beans', 'Pigeon peas', 'Chick peas',
       'Bananas', 'grapes', 'Apples', 'Mangoes', 'muskmelon', 'Oranges',
       'Papayas', 'pomegranate', 'Watermelons'], dtype=object)
```

In [28]:
```
label = pd.get_dummies(data.label).iloc[: ,:]
data= pd.concat([data,label],axis=1)
data.drop('label', axis=1,inplace=True)
data.head()
```

Out[28]:

	temperature	humidity	ph	rainfall	Adzuki Beans	Apples	Bananas	Black gram	Chick peas	Coconut
0	20.879744	82.002744	6.502985	202.935536	0	0	0	0	0	0
1	21.770462	80.319644	7.038096	226.655537	0	0	0	0	0	0
2	23.004459	82.320763	7.840207	263.964248	0	0	0	0	0	0
3	26.491096	80.158363	6.980401	242.864034	0	0	0	0	0	0
4	20.130175	81.604873	7.628473	262.717340	0	0	0	0	0	0

5 rows × 35 columns

In [29]:
```
features = data.iloc[:, 0:4].values
labels = data.iloc[: ,4:].values
features
```

Out[29]:
```
array([[ 20.87974371,  82.00274423,   6.50298529, 202.9355362 ],
       [ 21.77046169,  80.31964408,   7.03809636, 226.6555374 ],
       [ 23.00445915,  82.3207629 ,   7.84020714, 263.9642476 ],
       ...,
       [ 25.3310446 ,  84.30533791,   6.90424171,  41.53218699],
       [ 26.89750174,  83.89241484,   6.46327108,  43.97193745],
       [ 26.98603693,  89.4138489 ,   6.26083896,  58.54876687]])
```

In [30]:

```
X_train,X_test,y_train,y_test = train_test_split(features, labels, test_size=0.15
```

In [31]:

```
from sklearn.preprocessing import StandardScaler
msc = StandardScaler()
X_train = msc.fit_transform(X_train)
X_test = msc.transform(X_test)
```

In [32]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

model = DecisionTreeClassifier()
dec_model = model.fit(X_train,y_train)
pred = dec_model.predict(X_test)
a = accuracy_score(y_test, pred)
print("DecisionTreeClassifier accuracy : ", a*100)
```

```
DecisionTreeClassifier accuracy :  88.6021505376344
```

In [33]:

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators=100, random_state=0)
rand_model = model.fit(X_train,y_train)
pred = rand_model.predict(X_test)
a = accuracy_score(y_test, pred)
print('RandomForestClassifier accuracy : ',a*100)
```

```
RandomForestClassifier accuracy :  91.39784946236558
```

In [34]:

```
import pickle
filename = 'crop_model.sav'
pickle.dump(rand_model, open(filename, 'wb'))
```