Team ID: PNT2022TMID07706

# FERTILIZER RECOMMENDATIONSYSTEMFOR DISEASE PREDECTION.

## Crop recommendation model

In [1]:
```
# Importing libraries

from __future__ import print_function
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
from sklearn import tree
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
df = pd.read_csv('../Data-processed/crop-recommendation.csv')
```

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

Out[3]:

| | 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 |

In [4]:
```
df.tail()
```

Out[4]:

| | N | P | K | temperature | humidity | ph | rainfall | label |
|---|---|---|---|---|---|---|---|---|
| 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 |

In [5]:
```
df.size
```

Out[5]:
```
17600
```

In [6]:
```
df.shape
```

Out[6]:
```
(2200, 8)
```

In [7]:
```
df.columns
```

Out[7]:
```
Index(['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall', 'label'], dtype='object')
```

In [8]:
```
df['label'].unique()
```

Out[8]:
```
array(['rice', 'maize', 'chickpea', 'kidneybeans', 'pigeonpeas',
       'mothbeans', 'mungbean', 'blackgram', 'lentil', 'pomegranate',
       'banana', 'mango', 'grapes', 'watermelon', 'muskmelon', 'apple',
       'orange', 'papaya', 'coconut', 'cotton', 'jute', 'coffee'],
      dtype=object)
```

In [9]:
```
df.dtypes
```

Out[9]:
```
N               int64
P               int64
K               int64
temperature   float64
```

```
temperature    float64
humidity       float64
ph             float64
rainfall       float64
label           object
dtype: object
```

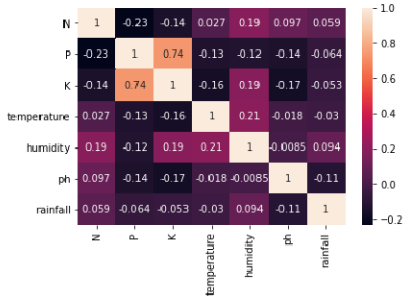
In [10]:
```
df['label'].value_counts()
```

Out[10]:
```
muskmelon      100
kidneybeans    100
papaya         100
pigeonpeas     100
blackgram      100
cotton         100
mothbeans      100
mungbean       100
watermelon     100
orange         100
mango          100
banana         100
rice           100
pomegranate    100
chickpea       100
apple          100
jute           100
grapes         100
lentil         100
coffee         100
maize          100
coconut        100
Name: label, dtype: int64
```

In [11]:
```
sns.heatmap(df.corr(),annot=True)
```

Out[11]:

Out[11]:




## Seperating features and target label

In [12]:
```
features = df[['N', 'P','K','temperature', 'humidity', 'ph', 'rainfall']]
target = df['label']
#features = df[['temperature', 'humidity', 'ph', 'rainfall']]
labels = df['label']
```

In [13]:
```
# Initialzing empty lists to append all model's name and corresponding name
acc = []
model = []
```

In [14]:
```
# Splitting into train and test 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)
```

## Decision Tree

In [15]:
```
from sklearn.tree import DecisionTreeClassifier

DecisionTree = DecisionTreeClassifier(criterion="entropy",random_state=2,max_depth=5)

DecisionTree.fit(Xtrain,Ytrain)

predicted_values = DecisionTree.predict(Xtest)
x = metrics.accuracy_score(Ytest, predicted_values)
acc.append(x)
model.append('Decision Tree')
print("DecisionTrees's Accuracy is: ", x*100)

print(classification_report(Ytest,predicted_values))
```

```
DecisionTrees's Accuracy is:  90.0
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        13
      banana       1.00      1.00      1.00        17
   blackgram       0.59      1.00      0.74        16
    chickpea       1.00      1.00      1.00        21
     coconut       0.91      1.00      0.95        21
      coffee       1.00      1.00      1.00        22
      cotton       1.00      1.00      1.00        20
      grapes       1.00      1.00      1.00        18
        jute       0.74      0.93      0.83        28
 kidneybeans       0.00      0.00      0.00        14
      lentil       0.68      1.00      0.81        23

       maize       1.00      1.00      1.00        21
       mango       1.00      1.00      1.00        26
   mothbeans       0.00      0.00      0.00        19
    mungbean       1.00      1.00      1.00        24
   muskmelon       1.00      1.00      1.00        23
      orange       1.00      1.00      1.00        29
      papaya       1.00      0.84      0.91        19
  pigeonpeas       0.62      1.00      0.77        18
 pomegranate       1.00      1.00      1.00        17
        rice       1.00      0.62      0.77        16
  watermelon       1.00      1.00      1.00        15

    accuracy                           0.90       440
   macro avg       0.84      0.88      0.85       440
weighted avg       0.86      0.90      0.87       440
```

In [16]:
```
from sklearn.model_selection import cross_val_score
```

In [17]:
```
# Cross validation score (Decision Tree)
score = cross_val_score(DecisionTree, features, target,cv=5)
```

In [18]:
```
score
```

Out[18]:
```
array([0.93636364, 0.90909091, 0.91818182, 0.87045455, 0.93636364])
```

### Saving trained Decision Tree model

In [19]:
```
import pickle
# Dump the trained Naive Bayes classifier with Pickle
DT_pkl_filename = '../models/DecisionTree.pkl'
# Open the file to save as pkl file
DT_Model_pkl = open(DT_pkl_filename, 'wb')
pickle.dump(DecisionTree, DT_Model_pkl)
# Close the pickle instances
DT_Model_pkl.close()
```

## Guassian Naive Bayes

In [20]:
```
from sklearn.naive_bayes import GaussianNB

NaiveBayes = GaussianNB()

NaiveBayes.fit(Xtrain,Ytrain)

predicted_values = NaiveBayes.predict(Xtest)
x = metrics.accuracy_score(Ytest, predicted_values)
acc.append(x)
model.append('Naive Bayes')
print("Naive Bayes's Accuracy is: ", x)

print(classification_report(Ytest,predicted_values))
```

```
Naive Bayes's Accuracy is:  0.990909090909091
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        13
      banana       1.00      1.00      1.00        17
   blackgram       1.00      1.00      1.00        16
      coffee       1.00      1.00      1.00        22
      cotton       1.00      1.00      1.00        20
      grapes       1.00      1.00      1.00        18
        jute       0.88      1.00      0.93        28
 kidneybeans       1.00      1.00      1.00        14
      lentil       1.00      1.00      1.00        23
       maize       1.00      1.00      1.00        21
       mango       1.00      1.00      1.00        26
   mothbeans       1.00      1.00      1.00        19
    mungbean       1.00      1.00      1.00        24
   muskmelon       1.00      1.00      1.00        23
      orange       1.00      1.00      1.00        29
      papaya       1.00      1.00      1.00        19
  pigeonpeas       1.00      1.00      1.00        18
 pomegranate       1.00      1.00      1.00        17
        rice       1.00      0.75      0.86        16
  watermelon       1.00      1.00      1.00        15

    accuracy                           0.99       440
   macro avg       0.99      0.99      0.99       440
weighted avg       0.99      0.99      0.99       440
```

In [21]:
```
# Cross validation score (NaiveBayes)
score = cross_val_score(NaiveBayes,features,target,cv=5)
score
```

Out[21]:
```
array([0.99772727, 0.99545455, 0.99545455, 0.99545455, 0.99090909])
```

### Saving trained Guassian Naive Bayes model

In [23]:
```
import pickle
# Dump the trained Naive Bayes classifier with Pickle
NB_pkl_filename = '../models/NBClassifier.pkl'
# Open the file to save as pkl file
NB_Model_pkl = open(NB_pkl_filename, 'wb')
pickle.dump(NaiveBayes, NB_Model_pkl)
```

## Support Vector Machine (SVM)

In [24]:
```
from sklearn.svm import SVC
# data normalization with sklearn
from sklearn.preprocessing import MinMaxScaler
# fit scaler on training data
norm = MinMaxScaler().fit(Xtrain)
X_train_norm = norm.transform(Xtrain)
# transform testing dataabs
X_test_norm = norm.transform(Xtest)
SVM = SVC(kernel='poly', degree=3, C=1)
SVM.fit(X_train_norm,Ytrain)
predicted_values = SVM.predict(X_test_norm)
x = metrics.accuracy_score(Ytest, predicted_values)
acc.append(x)
model.append('SVM')
print("SVM's Accuracy is: ", x)

print(classification_report(Ytest,predicted_values))
```

```
SVM's Accuracy is:  0.9795454545454545
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        13
      banana       1.00      1.00      1.00        17
   blackgram       1.00      1.00      1.00        16
    chickpea       1.00      1.00      1.00        21
     coconut       1.00      1.00      1.00        21
      coffee       1.00      0.95      0.98        22
      cotton       0.95      1.00      0.98        20
      grapes       1.00      1.00      1.00        18
        jute       0.83      0.89      0.86        28
 kidneybeans       1.00      1.00      1.00        14
      lentil       1.00      1.00      1.00        23
```

```
 kidneybeans       1.00      1.00      1.00        14
      lentil       1.00      1.00      1.00        23
       maize       1.00      0.95      0.98        21
       mango       1.00      1.00      1.00        26
   mothbeans       1.00      1.00      1.00        19
    mungbean       1.00      1.00      1.00        24
   muskmelon       1.00      1.00      1.00        23
      orange       1.00      1.00      1.00        29
      papaya       1.00      1.00      1.00        19
  pigeonpeas       1.00      1.00      1.00        18
 pomegranate       1.00      1.00      1.00        17
        rice       0.80      0.75      0.77        16
  watermelon       1.00      1.00      1.00        15

    accuracy                           0.98       440
   macro avg       0.98      0.98      0.98       440
weighted avg       0.98      0.98      0.98       440
```

In [37]:
```
# Cross validation score (SVM)
score = cross_val_score(SVM,features,target,cv=5)
score
```

Out[37]:
```
array([0.97954545, 0.975     , 0.98863636, 0.98863636, 0.98181818])
```

In [27]:
```
#Saving trained SVM model
```

In [28]:
```
import pickle
# Dump the trained SVM classifier with Pickle
SVM_pkl_filename = '../models/SVMClassifier.pkl'
# Open the file to save as pkl file
SVM_Model_pkl = open(SVM_pkl_filename, 'wb')
pickle.dump(SVM, SVM_Model_pkl)
```

```
SVM_Model_pkl = open(SVM_pkl_filename, 'wb')
pickle.dump(SVM, SVM_Model_pkl)
# Close the pickle instances
SVM_Model_pkl.close()
```

# Logistic Regression

In [29]:

```
from sklearn.linear_model import LogisticRegression

LogReg = LogisticRegression(random_state=2)

LogReg.fit(Xtrain,Ytrain)

predicted_values = LogReg.predict(Xtest)

x = metrics.accuracy_score(Ytest, predicted_values)
acc.append(x)
model.append('Logistic Regression')
print("Logistic Regression's Accuracy is: ", x)

print(classification_report(Ytest,predicted_values))
```

```
Logistic Regression's Accuracy is:  0.9522727272727273
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        13
      banana       1.00      1.00      1.00        17
   blackgram       0.86      0.75      0.80        16
    chickpea       1.00      1.00      1.00        21
     coconut       1.00      1.00      1.00        21
      coffee       1.00      1.00      1.00        22
      cotton       0.86      0.90      0.88        20
```

```
 kidneybeans       1.00      1.00      1.00        14
      lentil       0.88      1.00      0.94        23
       maize       0.90      0.86      0.88        21
       mango       0.96      1.00      0.98        26
   mothbeans       0.84      0.84      0.84        19
    mungbean       1.00      0.96      0.98        24
   muskmelon       1.00      1.00      1.00        23
      orange       1.00      1.00      1.00        29
      papaya       1.00      0.95      0.97        19
  pigeonpeas       1.00      1.00      1.00        18
 pomegranate       1.00      1.00      1.00        17
        rice       0.85      0.69      0.76        16
  watermelon       1.00      1.00      1.00        15

    accuracy                           0.95       440
   macro avg       0.95      0.95      0.95       440
weighted avg       0.95      0.95      0.95       440
```

In [30]:

```
# Cross validation score (Logistic Regression)
score = cross_val_score(LogReg,features,target,cv=5)
score
```

Out[30]:

```
array([0.95      , 0.96590909, 0.94772727, 0.96590909, 0.94318182])
```

## Saving trained Logistic Regression model

In [35]:

```
import pickle
# Dump the trained Naive Bayes classifier with Pickle
LR_pkl_filename = '../models/LogisticRegression.pkl'
# Open the file to save as pkl file
LR_Model_pkl = open(DT_pkl_filename, 'wb')
pickle.dump(LogReg, LR_Model_pkl)
```

```
# Open the file to save as pkl file
SVM_Model_pkl = open(SVM_pkl_filename, 'wb')
pickle.dump(SVM, SVM_Model_pkl)
# Close the pickle instances
SVM_Model_pkl.close()
```

## Logistic Regression

In [29]:
```
from sklearn.linear_model import LogisticRegression

LogReg = LogisticRegression(random_state=2)

LogReg.fit(Xtrain,Ytrain)

predicted_values = LogReg.predict(Xtest)

x = metrics.accuracy_score(Ytest, predicted_values)
acc.append(x)
model.append('Logistic Regression')
print("Logistic Regression's Accuracy is: ", x)

print(classification_report(Ytest,predicted_values))
```

```
Logistic Regression's Accuracy is:  0.9522727272727273
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        13
      banana       1.00      1.00      1.00        17
   blackgram       0.86      0.75      0.80        16
    chickpea       1.00      1.00      1.00        21
     coconut       1.00      1.00      1.00        21
      coffee       1.00      1.00      1.00        22
      cotton       0.86      0.90      0.88        20
      grapes       1.00      1.00      1.00        18
  kidneybeans       1.00      1.00      1.00        14
       lentil       0.88      1.00      0.94        23
        maize       0.90      0.86      0.88        21
        mango       0.96      1.00      0.98        26
    mothbeans       0.84      0.84      0.84        19
     mungbean       1.00      0.96      0.98        24
    muskmelon       1.00      1.00      1.00        23
       orange       1.00      1.00      1.00        29
       papaya       1.00      0.95      0.97        19
   pigeonpeas       1.00      1.00      1.00        18
  pomegranate       1.00      1.00      1.00        17
         rice       0.85      0.69      0.76        16
   watermelon       1.00      1.00      1.00        15

     accuracy                           0.95       440
    macro avg       0.95      0.95      0.95       440
 weighted avg       0.95      0.95      0.95       440
```

In [30]:
```
# Cross validation score (Logistic Regression)
score = cross_val_score(LogReg,features,target,cv=5)
score
```

Out[30]:
```
array([0.95      , 0.96590909, 0.94772727, 0.96590909, 0.94318182])
```

### Saving trained Logistic Regression model

In [35]:
```
import pickle
# Dump the trained Naive Bayes classifier with Pickle
LR_pkl_filename = '../models/LogisticRegression.pkl'
# Open the file to save as pkl file
LR_Model_pkl = open(DT_pkl_filename, 'wb')
pickle.dump(LogReg, LR_Model_pkl)
# Close the pickle instances
```

```
RF's Accuracy is:  0.990909090909091
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        13
      banana       1.00      1.00      1.00        17
   blackgram       0.94      1.00      0.97        16
    chickpea       1.00      1.00      1.00        21
     coconut       1.00      1.00      1.00        21
      coffee       1.00      1.00      1.00        22
      cotton       1.00      1.00      1.00        20
      grapes       1.00      1.00      1.00        18
        jute       0.90      1.00      0.95        28
 kidneybeans       1.00      1.00      1.00        14
      lentil       1.00      1.00      1.00        23
       maize       1.00      1.00      1.00        21
       mango       1.00      1.00      1.00        26
   mothbeans       1.00      0.95      0.97        19
    mungbean       1.00      1.00      1.00        24
   muskmelon       1.00      1.00      1.00        23
      orange       1.00      1.00      1.00        29
      papaya       1.00      1.00      1.00        19
  pigeonpeas       1.00      1.00      1.00        18
 pomegranate       1.00      1.00      1.00        17
        rice       1.00      0.81      0.90        16
  watermelon       1.00      1.00      1.00        15

    accuracy                           0.99       440
   macro avg       0.99      0.99      0.99       440
weighted avg       0.99      0.99      0.99       440
```

In [37]:

```
# Cross validation score (Random Forest)
score = cross_val_score(RF,features,target,cv=5)
score
```

Out[37]:

```
array([0.99772727, 0.99545455, 0.99772727, 0.99318182, 0.98863636])
```

## Saving trained Random Forest model

In [38]:

```
import pickle
# Dump the trained Naive Bayes classifier with Pickle
RF_pkl_filename = '../models/RandomForest.pkl'
# Open the file to save as pkl file
RF_Model_pkl = open(RF_pkl_filename, 'wb')
pickle.dump(RF, RF_Model_pkl)
# Close the pickle instances
RF_Model_pkl.close()
```

# XGBoost

In [39]:

```
import xgboost as xgb
XB = xgb.XGBClassifier()
XB.fit(Xtrain,Ytrain)

predicted_values = XB.predict(Xtest)
```

```
print("XGBoost's Accuracy is: ", x)

print(classification_report(Ytest,predicted_values))
```

```
[14:16:03] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGBoost
1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Exp
licitly set eval_metric if you'd like to restore the old behavior.
XGBoost's Accuracy is:  0.9931818181818182
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        13
      banana       1.00      1.00      1.00        17
   blackgram       1.00      1.00      1.00        16
    chickpea       1.00      1.00      1.00        21
     coconut       1.00      1.00      1.00        21
      coffee       0.96      1.00      0.98        22
      cotton       1.00      1.00      1.00        20
      grapes       1.00      1.00      1.00        18
        jute       1.00      0.93      0.96        28
 kidneybeans       1.00      1.00      1.00        14
      lentil       0.96      1.00      0.98        23
       maize       1.00      1.00      1.00        21
       mango       1.00      1.00      1.00        26
   mothbeans       1.00      0.95      0.97        19
    mungbean       1.00      1.00      1.00        24
   muskmelon       1.00      1.00      1.00        23
      orange       1.00      1.00      1.00        29
      papaya       1.00      1.00      1.00        19
  pigeonpeas       1.00      1.00      1.00        18
 pomegranate       1.00      1.00      1.00        17
        rice       0.94      1.00      0.97        16
  watermelon       1.00      1.00      1.00        15
```

In [46]:
```
# Cross validation score (XGBoost)
score = cross_val_score(XB,features,target,cv=5)
score
```

```
[08:54:44] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGBoost
1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Exp
licitly set eval_metric if you'd like to restore the old behavior.
[08:54:45] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGBoost
1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Exp
licitly set eval_metric if you'd like to restore the old behavior.
[08:54:46] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGBoost
1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Exp
licitly set eval_metric if you'd like to restore the old behavior.
[08:54:47] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGBoost
1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Exp
licitly set eval_metric if you'd like to restore the old behavior.
[08:54:48] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGBoost
1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Exp
licitly set eval_metric if you'd like to restore the old behavior.
```

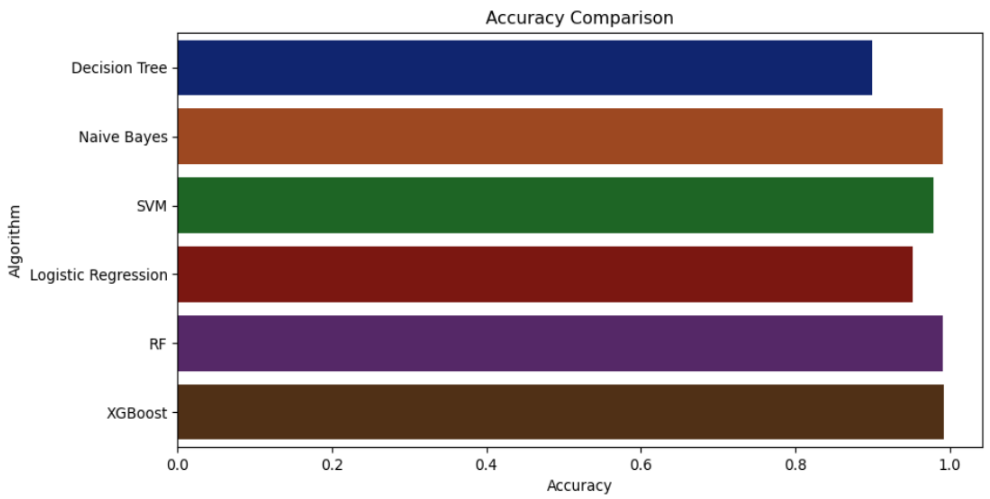
Out[46]:
```
array([0.99318182, 0.99318182, 0.99318182, 0.99090909, 0.99090909])
```

## Saving trained XGBoost model

In [40]:
```
import pickle
# Dump the trained Naive Bayes classifier with Pickle
XB_pkl_filename = '../models/XGBoost.pkl'
# Open the file to save as pkl file
XB_Model_pkl = open(XB_pkl_filename, 'wb')
pickle.dump(XB, XB_Model_pkl)
```

```
sns.barplot(x = acc,y = model,palette='dark')
```

Out[41]:




In [42]: