- NumPy : used to perform a wide variety of mathematical operations on arrays
- Pandas : open source Python package that is most widely used for data science/data analysis and machine learning tasks
- Seaborn : data visualization library for statistical graphics plotting in Python.
- Pickle : primarily used in serializing and deserializing a Python object structure
- MatplotLib : plotting library used for 2D graphics in python programming language
- Sklearn : contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction

```
import numpy as np
import pandas as pd
import seaborn as sns
import pickle
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.ensemble import ExtraTreesRegressor

import warnings
warnings.filterwarnings('ignore')


from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call dr
```

dataset_path is path from **Google Drive** to **dataset of crop_recommendation**

```
# Path needs to be changed as per location of dataset in mounted drive

dataset_path = '/content/drive/MyDrive/Cloud/Crop_recommendation.csv'
df = pd.read_csv(dataset_path)
```

1. Shows Dataset of given size (default = 5)

2. Shape of entire dataset (Rows,Columns)
3. All features available in dataset
4. Different Types of Crops that can be recommended

```
sample_size = 5
print("\nDataset Sample: ")
print(df.head(sample_size))
print("\nShape of Dataset: ")
print(df.shape)
print("\nFeatures of Dataset: ")
print(df.columns)
print("\nDifferent Crop Labels: ")
print(df['label'].unique())
```

```
Dataset Sample:
    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

Shape of Dataset:
(2200, 8)

Features of Dataset:
Index(['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall', 'label'], dty

Different Crop Labels:
['rice' 'maize' 'chickpea' 'kidneybeans' 'pigeonpeas' 'mothbeans'
 'mungbean' 'blackgram' 'lentil' 'pomegranate' 'banana' 'mango' 'grapes'
 'watermelon' 'muskmelon' 'apple' 'orange' 'papaya' 'coconut' 'cotton'
 'jute' 'coffee']
```

Number of null fields for each features

```
df.isnull().sum()
```

```
N              0
P              0
K              0
temperature    0
humidity       0
ph             0
rainfall       0
label          0
dtype: int64
```

Dataset description summary

```
df.describe()
```

	N	P	K	temperature	humidity	ph
count	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000
mean	50.551818	53.362727	48.149091	25.616244	71.481779	6.469480
std	36.917334	32.985883	50.647931	5.063749	22.263812	0.773938
min	0.000000	5.000000	5.000000	8.825675	14.258040	3.504752
25%	21.000000	28.000000	20.000000	22.769375	60.261953	5.971693
50%	37.000000	51.000000	32.000000	25.598693	80.473146	6.425045
75%	84.250000	68.000000	49.000000	28.561654	89.948771	6.923643
max	140.000000	145.000000	205.000000	43.675493	99.981876	9.935091

Encoding Labels in dataset

```
# label_crop_code = df["label"].astype('category').tolist()
# replace_map_comp = {'label' : {k: v for k,v in zip(label_crop_code,list(range(1,len
# print("Encoded Crop Code: \n")
# print(replace_map_comp)
final_data_set = df.copy()
# final_data_set.replace(replace_map_comp, inplace=True)
```

Sample of Processed Dataset

```
print("Processed Dataset Sample: ")
final_data_set.head(5)
```

```
Processed Dataset Sample:
```

	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

Matrix of Correlation between different features and label of dataset

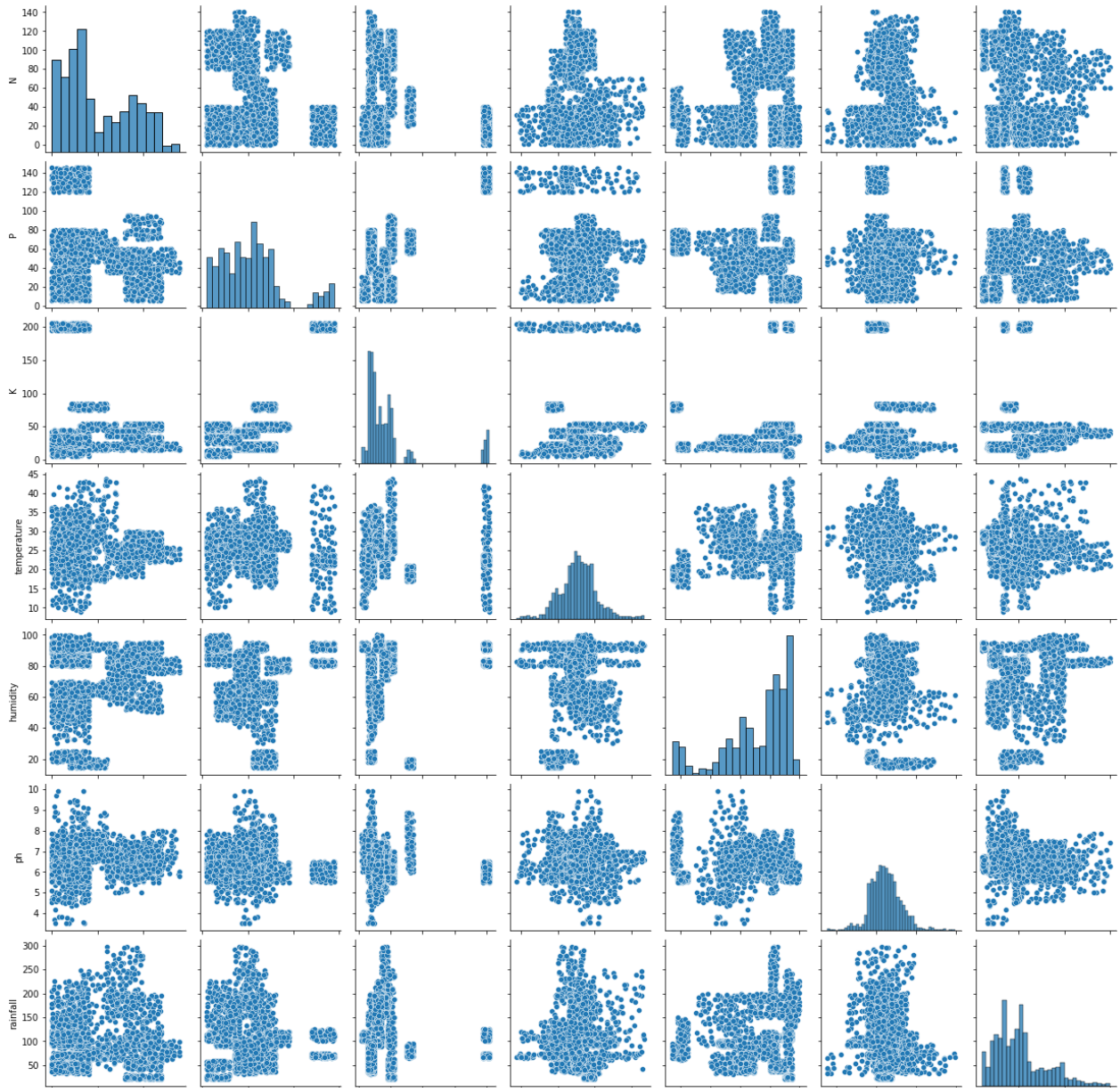
```
final_data_set.corr()
```

	N	P	K	temperature	humidity	ph	rai
N	1.000000	-0.231460	-0.140512	0.026504	0.190688	0.096683	0.0
P	-0.231460	1.000000	0.736232	-0.127541	-0.118734	-0.138019	-0.0
K	-0.140512	0.736232	1.000000	-0.160387	0.190859	-0.169503	-0.0
temperature	0.026504	-0.127541	-0.160387	1.000000	0.205320	-0.017795	-0.0
humidity	0.190688	-0.118734	0.190859	0.205320	1.000000	-0.008483	0.0
ph	0.096683	-0.138019	-0.169503	-0.017795	-0.008483	1.000000	-0.1
rainfall	0.059020	-0.063839	-0.053461	-0.030084	0.094423	-0.109069	1.0

Plot pairwise relationships in a dataset.

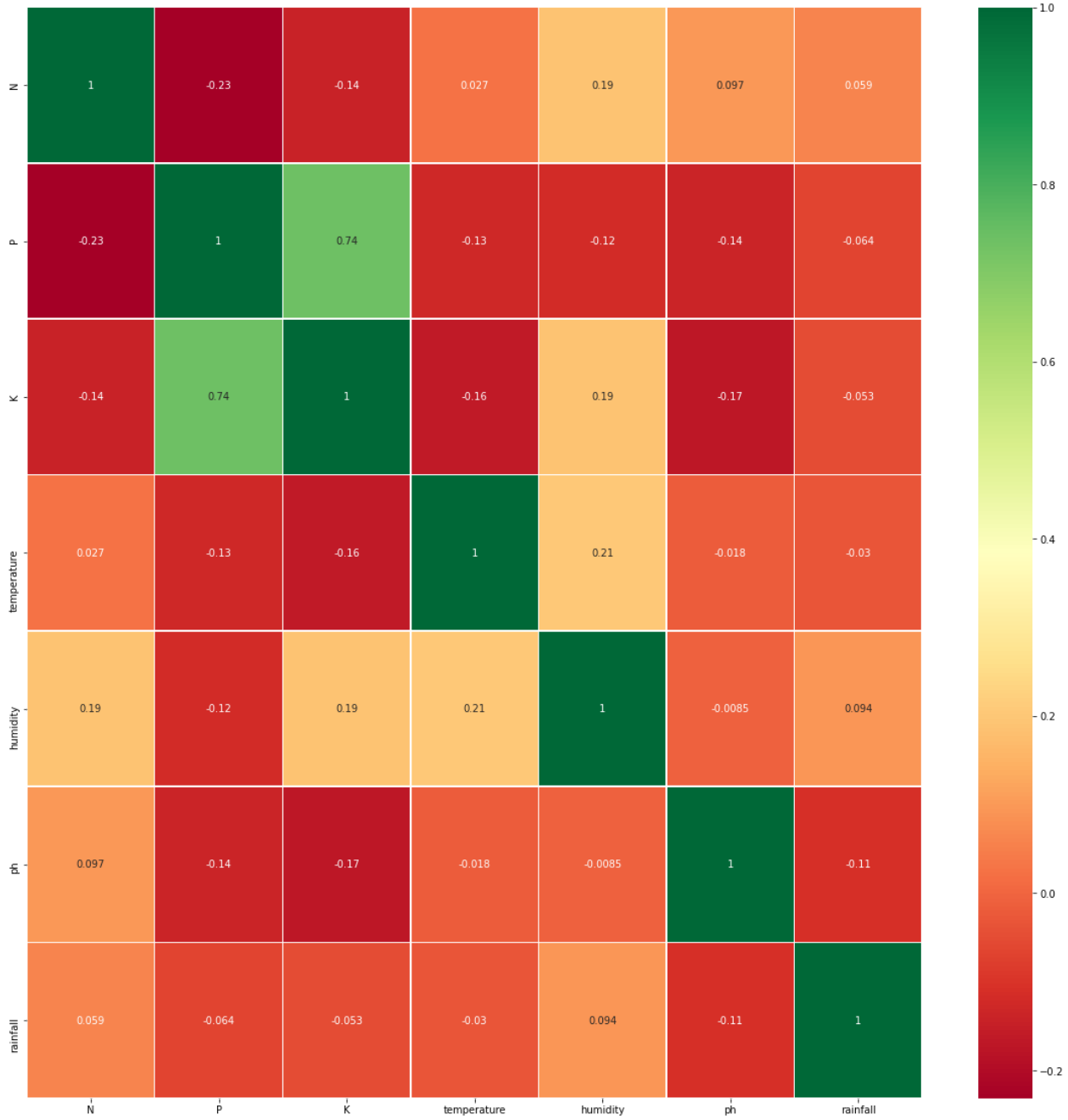
```
sns.pairplot(final_data_set)
```

```
<seaborn.axisgrid.PairGrid at 0x7f06a56d68d0>
```



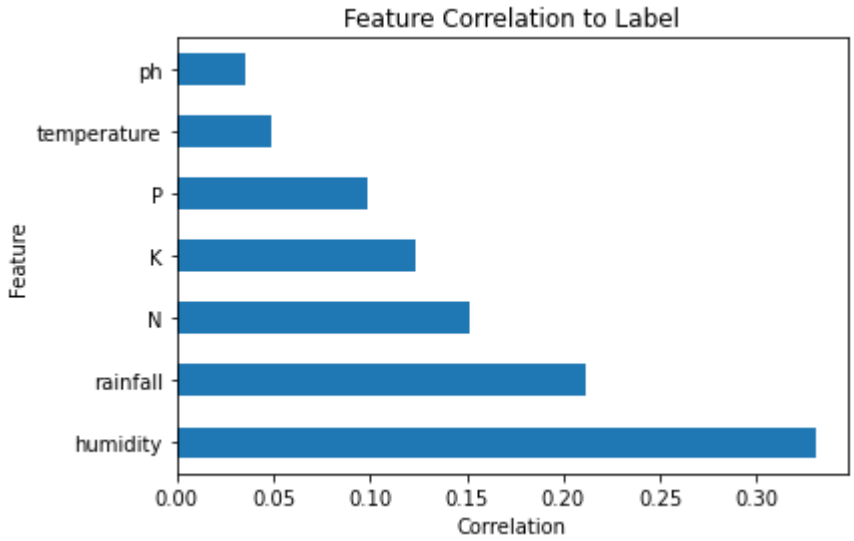
Plot correlation rectangular data as a color-encoded matrix

```
corrmat = final_data_set.corr()
top_corr_features = corrmat.index
plt.figure(figsize = (20,20))
g=sns.heatmap(df[top_corr_features].corr(),annot=True,cmap="RdYlGn", linewidths=.5)
```


N
P
K
temperature
humidity
ph
rainfall
1 -0.23 -0.14 0.027 0.19 0.097 0.059
-0.23 1 0.74 -0.13 -0.12 -0.14 -0.064
-0.14 0.74 1 -0.16 0.19 -0.17 -0.053
0.027 -0.13 -0.16 1 0.21 -0.018 -0.03
0.19 -0.12 0.19 0.21 1 -0.0085 0.094
0.097 -0.14 -0.17 -0.018 -0.0085 1 -0.11
0.059 -0.064 -0.053 -0.03 0.094 -0.11 1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

Seperate Features and label

```
X = final_data_set[['N', 'P','K','temperature', 'humidity', 'ph', 'rainfall']]
y = final_data_set['label']


label_crop_code = df["label"].astype('category').tolist()
replace_map_comp = {'label' : {k: v for k,v in zip(label_crop_code,list(range(1,len(l
correlation_data_set = df.copy()
correlation_data_set.replace(replace_map_comp, inplace=True)

corr_X = correlation_data_set[['N', 'P','K','temperature', 'humidity', 'ph', 'rainfal
corr_y = correlation_data_set['label']

model = ExtraTreesRegressor()
model.fit(corr_X,corr_y)

feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(len(corr_X)).plot(kind='barh')
plt.title('Feature Correlation to Label')
plt.xlabel('Correlation')
plt.ylabel('Feature')
plt.show()
```



Feature Sample Dataset

```
print("Feature Dataset Sample: ")
X.head(5)
```

Feature Dataset Sample:

	N	P	K	temperature	humidity	ph	rainfall
0	90	42	43	20.879744	82.002744	6.502985	202.935536
1	85	58	41	21.770462	80.319644	7.038096	226.655537

Spliting available data as Train and Test

3 74 35 40 26.491096 80.158363 6.980401 242.864034

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2)


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

Decision Tree

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

DecisionTree.fit(X,y)

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

print(classification_report(y_test,predicted_values))
```

```
DecisionTrees's Accuracy is:  93.63636363636364
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        17
      banana       1.00      1.00      1.00        21
   blackgram       0.72      1.00      0.84        21
    chickpea       1.00      1.00      1.00        18
     coconut       1.00      1.00      1.00        15
      coffee       1.00      0.96      0.98        26
      cotton       1.00      1.00      1.00        25
      grapes       1.00      1.00      1.00        21
        jute       1.00      0.25      0.40        20
 kidneybeans       1.00      0.89      0.94        19
      lentil       0.93      1.00      0.96        26
       maize       0.86      0.95      0.90        19
       mango       1.00      1.00      1.00        19
   mothbeans       1.00      0.50      0.67        18
    mungbean       1.00      1.00      1.00        23
   muskmelon       1.00      1.00      1.00        21
      orange       1.00      1.00      1.00        14
```

```
              papaya       1.00      1.00      1.00        16
           pigeonpeas       1.00      1.00      1.00        21
          pomegranate       1.00      1.00      1.00        17
                 rice       0.53      1.00      0.69        17
           watermelon       1.00      1.00      1.00        26

            accuracy                           0.94       440
           macro avg       0.96      0.93      0.93       440
        weighted avg       0.96      0.94      0.93       440
```

```
score = cross_val_score(DecisionTree,  X, y,cv=5)
print(score)
```

```
     [0.93636364 0.90909091 0.91818182 0.87045455 0.93636364]
```

```
# Dump the trained Naive Bayes classifier with Pickle
DT_pkl_filename = '/content/drive/MyDrive/Cloud/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()
```

Naive Bayes

```
NaiveBayes = GaussianNB()

NaiveBayes.fit(X_train,y_train)

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

print(classification_report(y_test,predicted_values))
```

```
     Naive Bayes's Accuracy is:  0.9954545454545455
                   precision    recall  f1-score   support

                apple       1.00      1.00      1.00        17
               banana       1.00      1.00      1.00        21
            blackgram       1.00      1.00      1.00        21
             chickpea       1.00      1.00      1.00        18
              coconut       1.00      1.00      1.00        15
               coffee       1.00      1.00      1.00        26
               cotton       1.00      1.00      1.00        25
               grapes       1.00      1.00      1.00        21
                 jute       1.00      0.95      0.97        20
```

```
      kidneybeans       1.00      1.00      1.00        19
           lentil       1.00      0.96      0.98        26
            maize       1.00      1.00      1.00        19
            mango       1.00      1.00      1.00        19
        mothbeans       0.95      1.00      0.97        18
         mungbean       1.00      1.00      1.00        23
        muskmelon       1.00      1.00      1.00        21
           orange       1.00      1.00      1.00        14
           papaya       1.00      1.00      1.00        16
       pigeonpeas       1.00      1.00      1.00        21
      pomegranate       1.00      1.00      1.00        17
             rice       0.94      1.00      0.97        17
       watermelon       1.00      1.00      1.00        26

         accuracy                           1.00       440
        macro avg       1.00      1.00      1.00       440
     weighted avg       1.00      1.00      1.00       440
```

```
score = cross_val_score(NaiveBayes,X,y,cv=5)
score
```

```
array([0.99772727, 0.99545455, 0.99545455, 0.99545455, 0.99090909])
```

```
# Dump the trained Naive Bayes classifier with Pickle
NB_pkl_filename = '/content/drive/MyDrive/Cloud/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)
# Close the pickle instances
NB_Model_pkl.close()
```

SVM

```
SVM = SVC(gamma='auto')

SVM.fit(X_train,y_train)

predicted_values = SVM.predict(X_test)

x = metrics.accuracy_score(y_test, predicted_values)
acc.append(x)
model.append('SVM')
print("SVM's Accuracy is: ", x)

print(classification_report(y_test,predicted_values))
```

```
SVM's Accuracy is:  0.12045454545454545
              precision    recall  f1-score   support
```

```
              apple       1.00      0.22      0.36        18
             banana       1.00      0.09      0.16        23
          blackgram       1.00      0.04      0.08        23
           chickpea       1.00      0.11      0.20        18
            coconut       1.00      0.06      0.12        16
             coffee       0.00      0.00      0.00        21
             cotton       1.00      0.08      0.15        25
             grapes       0.00      0.00      0.00        26
               jute       1.00      0.20      0.33        15
        kidneybeans       0.03      1.00      0.07        14
             lentil       0.00      0.00      0.00        29
              maize       0.00      0.00      0.00        17
              mango       1.00      0.20      0.33        20
          mothbeans       0.00      0.00      0.00        19
           mungbean       1.00      0.26      0.42        19
          muskmelon       1.00      0.29      0.44        21
             orange       1.00      0.07      0.12        15
             papaya       0.00      0.00      0.00        25
         pigeonpeas       0.00      0.00      0.00        23
        pomegranate       1.00      0.43      0.60        14
               rice       0.00      0.00      0.00        19
         watermelon       1.00      0.10      0.18        20

           accuracy                           0.12       440
          macro avg       0.59      0.14      0.16       440
       weighted avg       0.56      0.12      0.15       440
```

```
# Cross validation score (SVM)
score = cross_val_score(SVM,X,y,cv=5)
score
```

```
array([0.27727273, 0.28863636, 0.29090909, 0.275     , 0.26818182])
```

```
# Dump the trained Naive Bayes classifier with Pickle
SVM_pkl_filename = '/content/drive/MyDrive/Cloud/Models/SVM.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

```
LogReg = LogisticRegression(random_state=2)

LogReg.fit(X_train,y_train)

predicted_values = LogReg.predict(X_test)

x = metrics.accuracy_score(y_test, predicted_values)
```

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

print(classification_report(y_test,predicted_values))
```

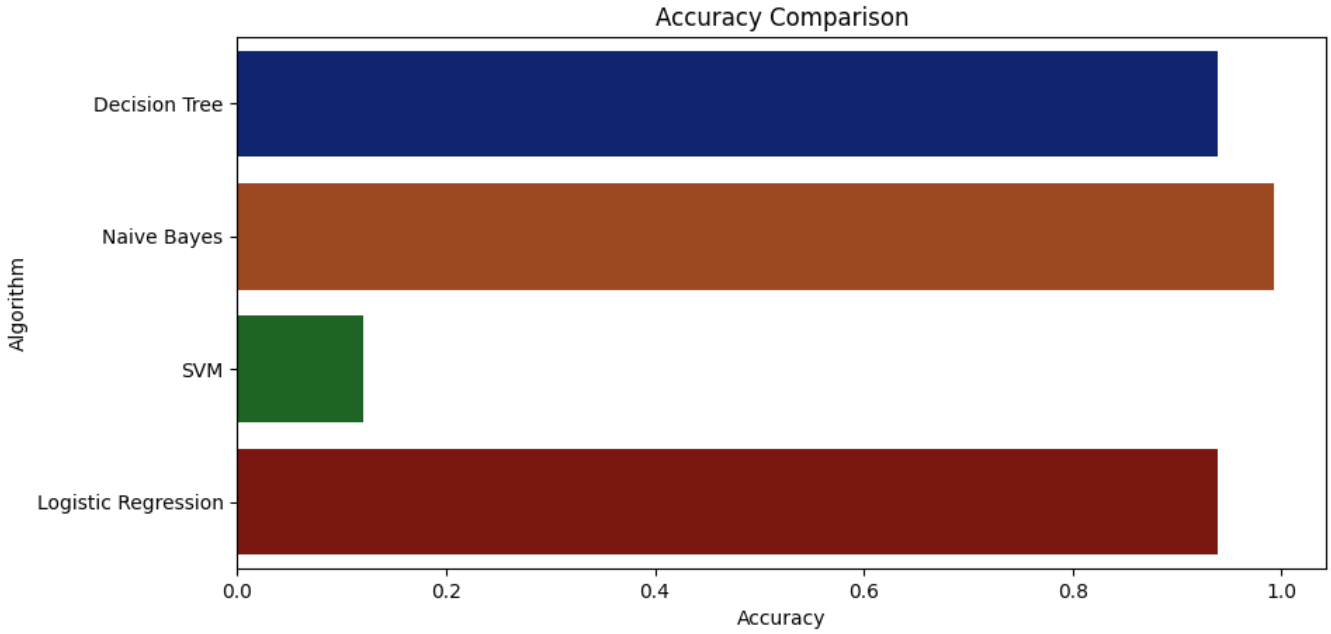
```
Logistic Regression's Accuracy is:  0.9386363636363636
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        18
      banana       0.96      1.00      0.98        23
   blackgram       0.88      0.91      0.89        23
    chickpea       1.00      1.00      1.00        18
     coconut       0.84      1.00      0.91        16
      coffee       0.95      1.00      0.98        21
      cotton       0.96      0.96      0.96        25
      grapes       1.00      1.00      1.00        26
        jute       0.57      0.53      0.55        15
 kidneybeans       1.00      1.00      1.00        14
      lentil       0.90      0.93      0.92        29
       maize       0.93      0.82      0.87        17
       mango       1.00      1.00      1.00        20
   mothbeans       0.94      0.84      0.89        19
    mungbean       1.00      1.00      1.00        19
   muskmelon       1.00      1.00      1.00        21
      orange       1.00      1.00      1.00        15
      papaya       1.00      0.88      0.94        25
  pigeonpeas       1.00      1.00      1.00        23
 pomegranate       1.00      1.00      1.00        14
        rice       0.65      0.68      0.67        19
  watermelon       1.00      1.00      1.00        20

    accuracy                           0.94       440
   macro avg       0.94      0.93      0.93       440
weighted avg       0.94      0.94      0.94       440
```

```
# Cross validation score (Logistic Regression)
score = cross_val_score(LogReg,X,y,cv=5)
score
```

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

```
plt.figure(figsize=[10,5],dpi = 100)
plt.title('Accuracy Comparison')
plt.xlabel('Accuracy')
plt.ylabel('Algorithm')
sns.barplot(x = acc,y = model,palette='dark')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f64cb38a390>
```




```
data = np.array([[104,18, 30, 23.603016, 60.3, 6.7, 140.91]])
prediction = NaiveBayes.predict(data)
print(prediction)
```

```
['coffee']
```