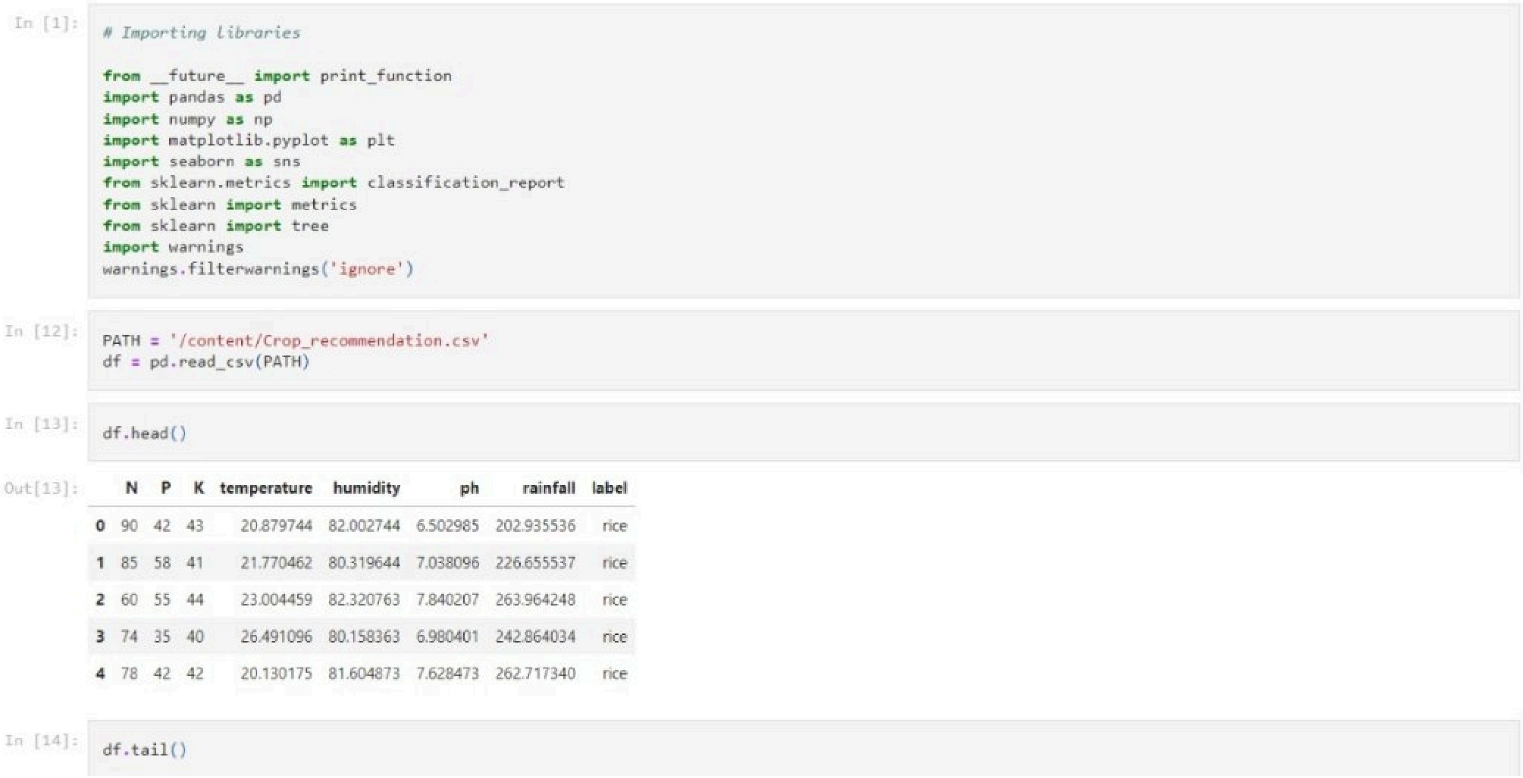


In [1]:

```
# 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 [12]:

```
PATH = '/content/Crop_recommendation.csv'
df = pd.read_csv(PATH)
```

In [13]:

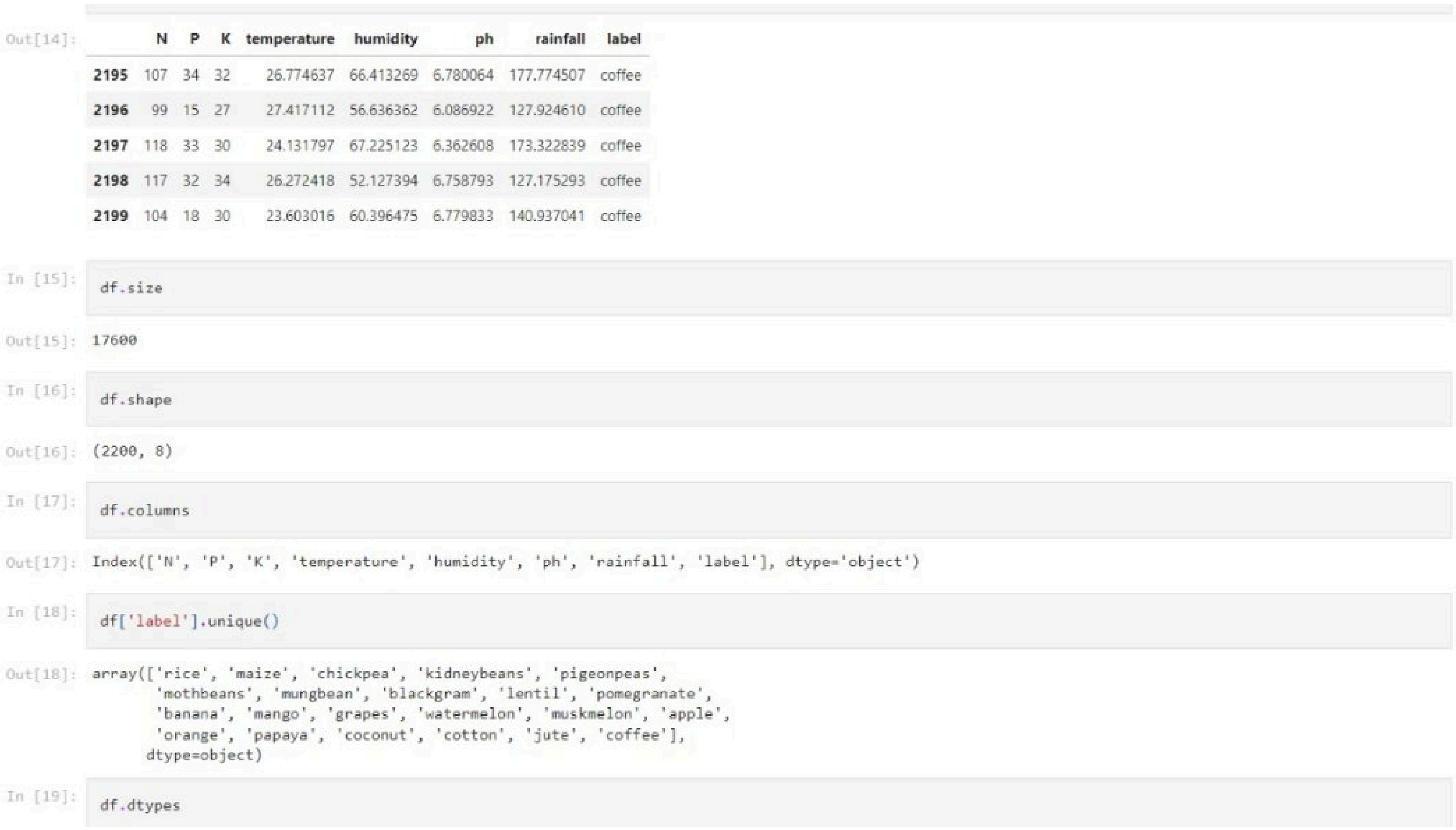
```
df.head()
```

Out[13]:

```
    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 [14]:

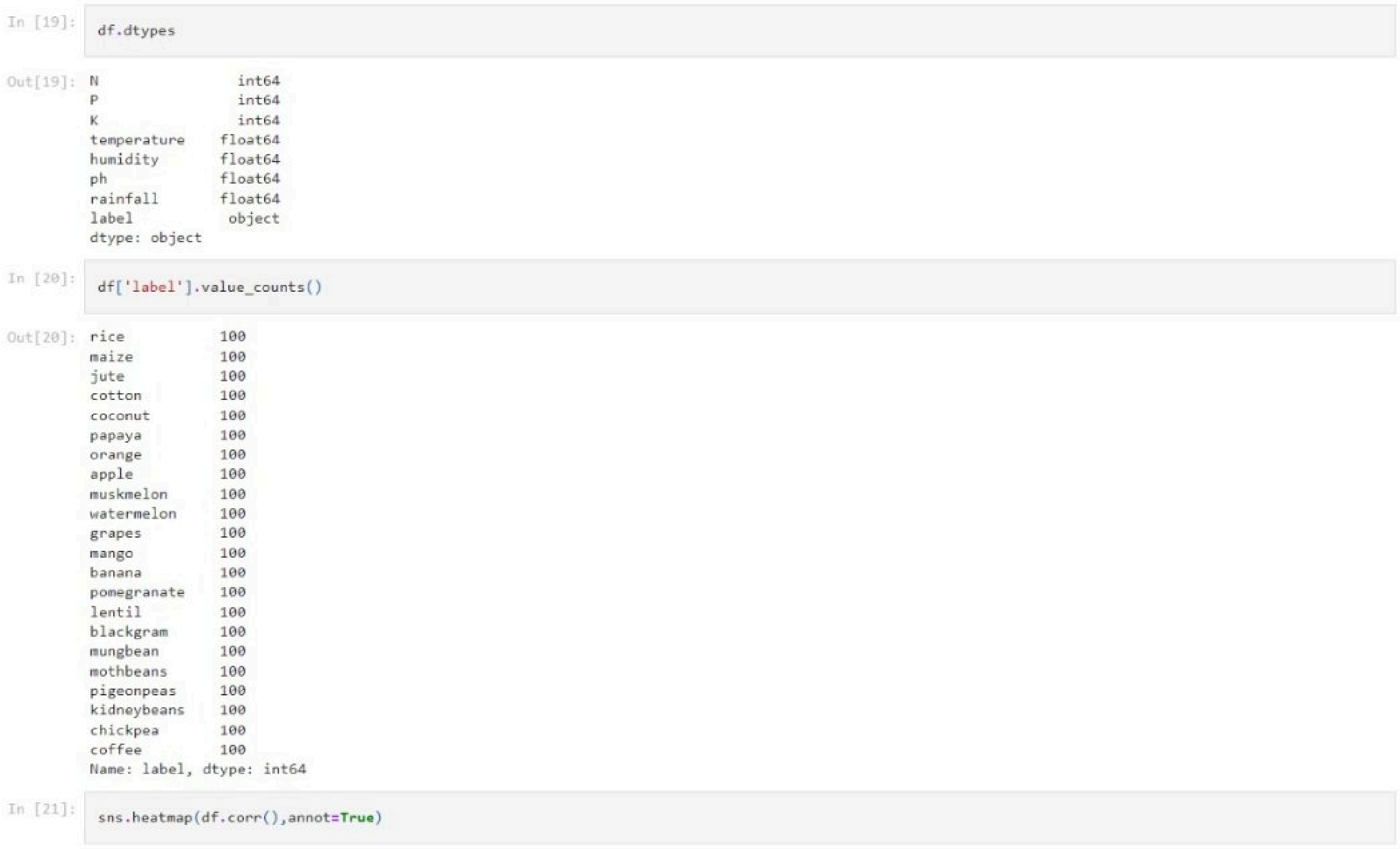
```
df.tail()
```



Out[14]:

|  | 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 [15]: df.size

Out[15]: 17600

In [16]: df.shape

Out[16]: (2200, 8)

In [17]: df.columns

Out[17]: Index(['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall', 'label'], dtype='object')

In [18]: df['label'].unique()

Out[18]: 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 [19]: df.dtypes
```



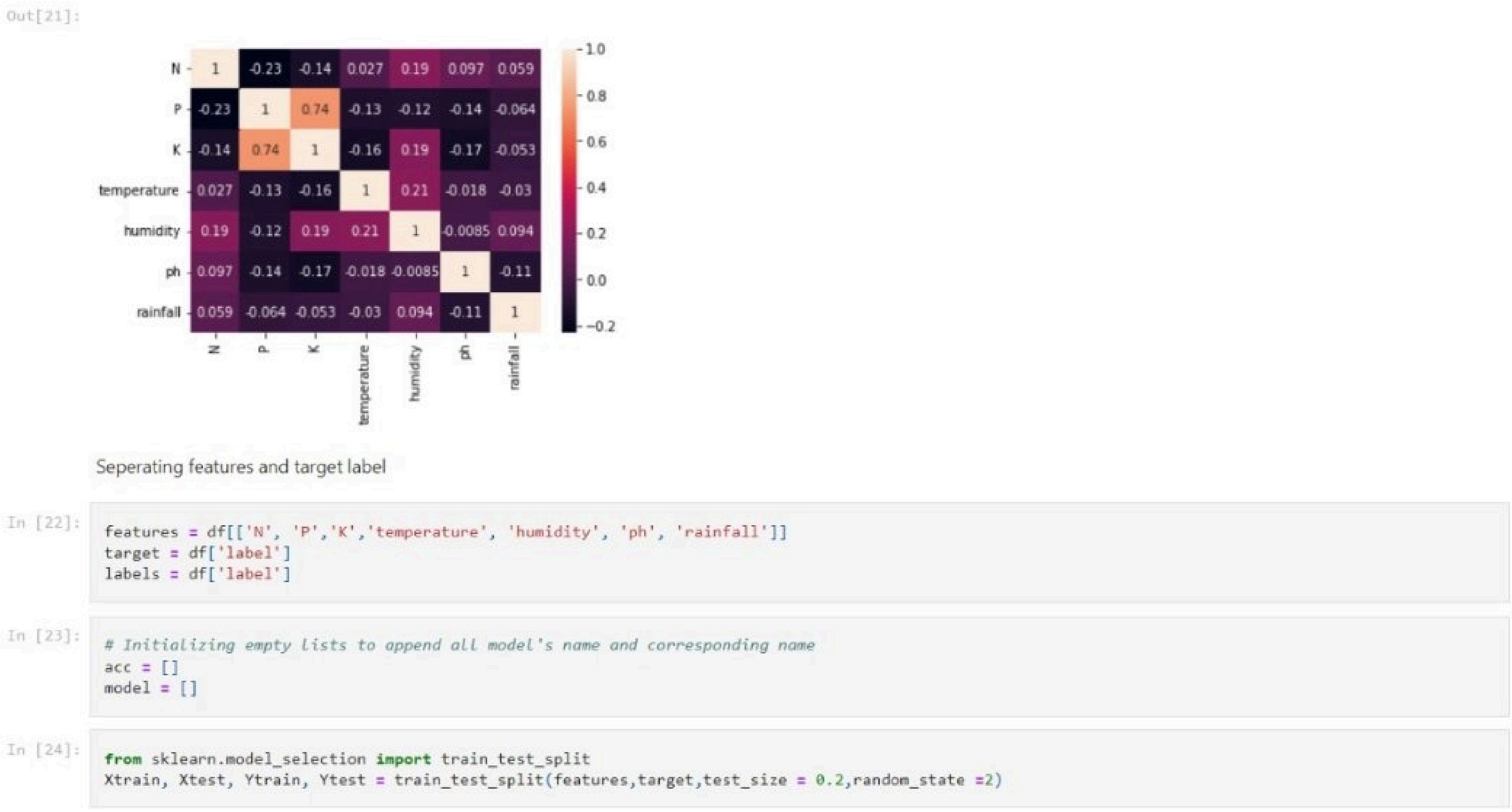
```
In [19]: df.dtypes
```

```
Out[19]: N                int64
         P                int64
         K                int64
         temperature    float64
         humidity       float64
         ph             float64
         rainfall       float64
         label           object
         dtype: object
```

```
In [20]: df['label'].value_counts()
```

```
Out[20]: rice           100
         maize          100
         jute           100
         cotton         100
         coconut        100
         papaya         100
         orange         100
         apple          100
         muskmelon      100
         watermelon     100
         grapes         100
         mango          100
         banana         100
         pomegranate    100
         lentil         100
         blackgram      100
         mungbean       100
         mothbeans      100
         pigeonpeas     100
         kidneybeans    100
         chickpea       100
         coffee         100
         Name: label, dtype: int64
```

```
In [21]: sns.heatmap(df.corr(),annot=True)
```



Out[21]:

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

Seperating features and target label

In [22]:

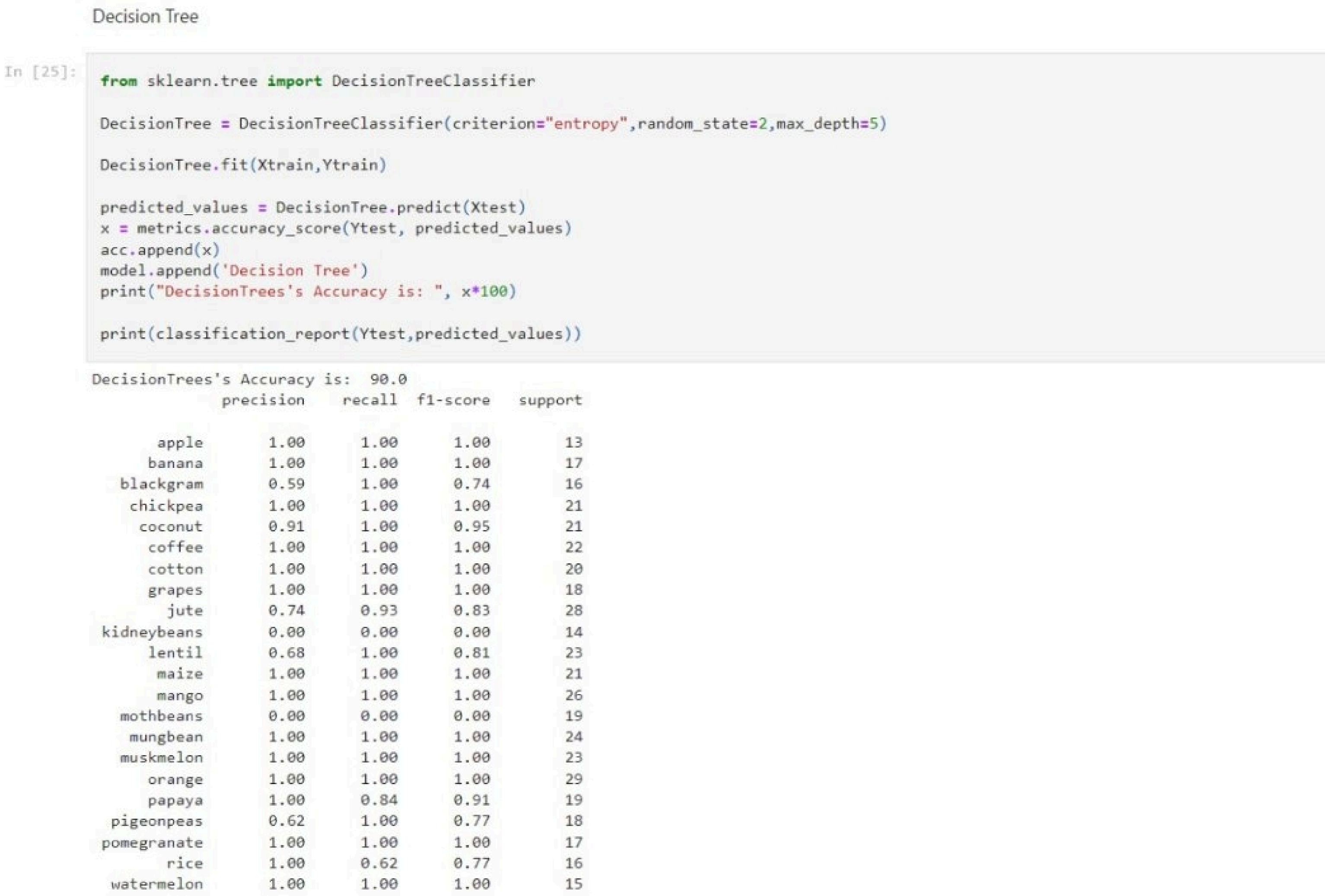
```
features = df[['N', 'P','K','temperature', 'humidity', 'ph', 'rainfall']]
target = df['label']
labels = df['label']
```

In [23]:

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

In [24]:

```
from sklearn.model_selection import train_test_split
Xtrain, Xtest, Ytrain, Ytest = train_test_split(features,target,test_size = 0.2,random_state =2)
```

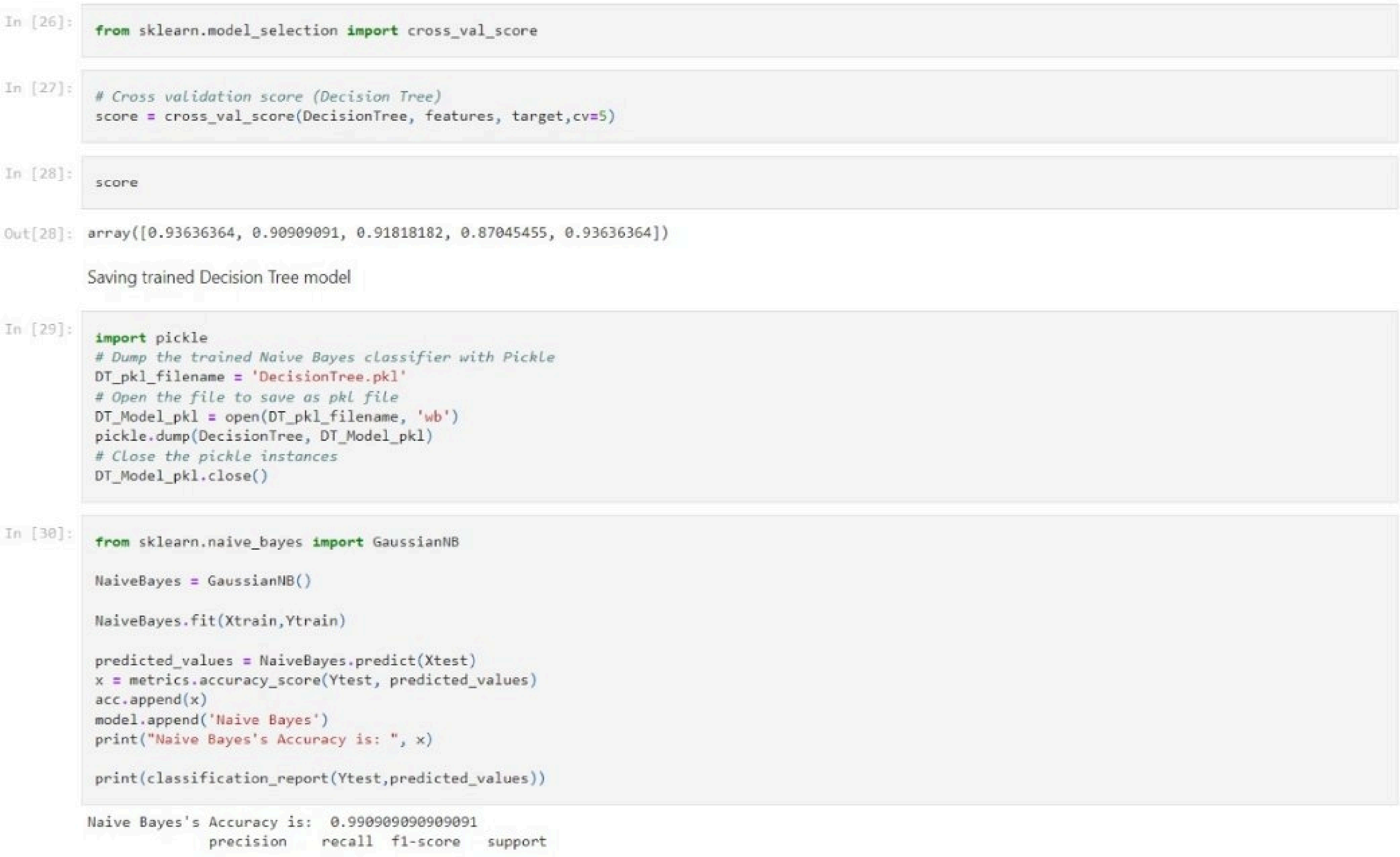


Decision Tree

```
In [25]:
```

```python
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
```



```
In [26]:  from sklearn.model_selection import cross_val_score
```

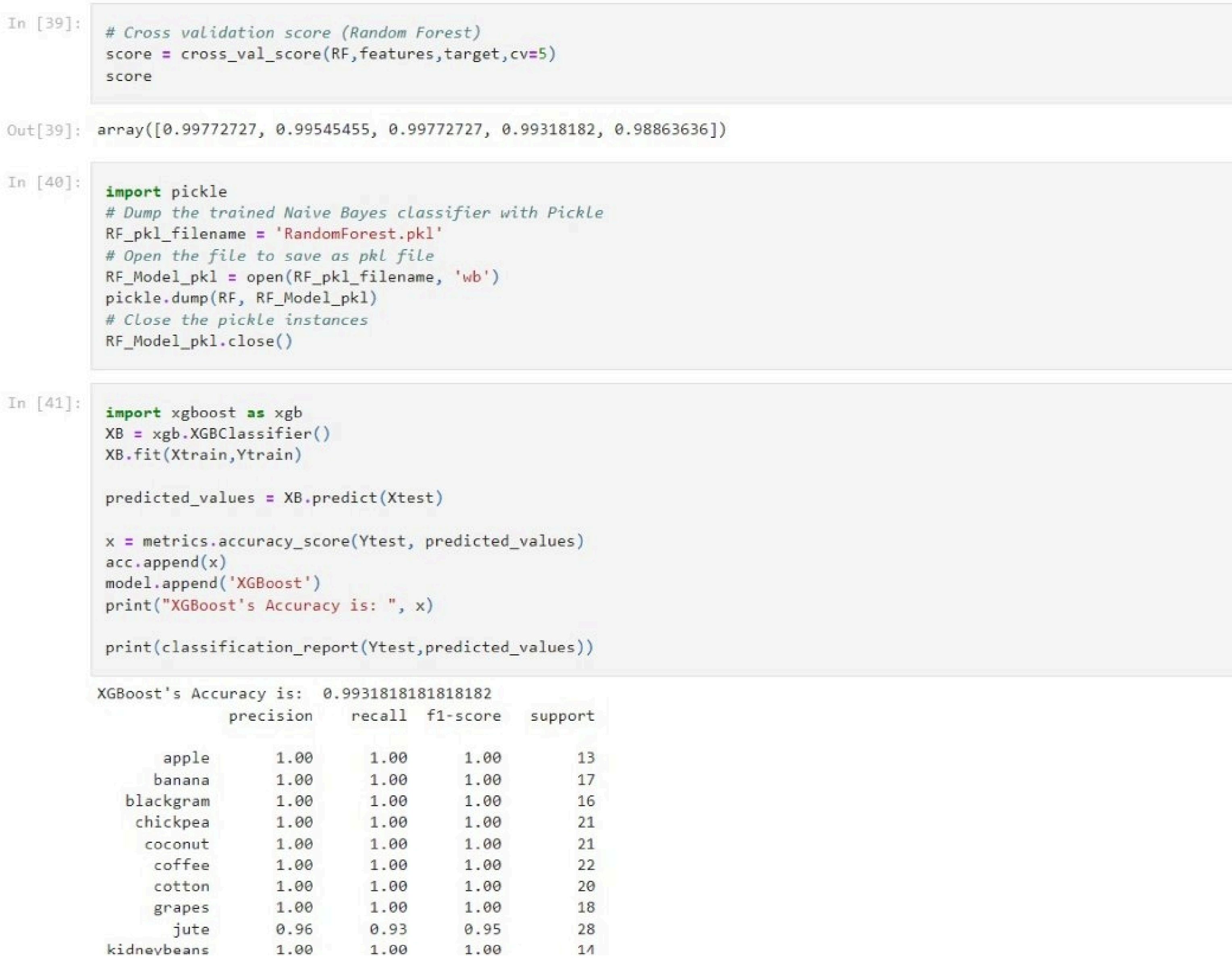
```
In [27]:  # Cross validation score (Decision Tree)
          score = cross_val_score(DecisionTree, features, target,cv=5)
```

```
In [28]:  score

Out[28]: array([0.93636364, 0.90909091, 0.91818182, 0.87045455, 0.93636364])
```

Saving trained Decision Tree model

```
In [29]:  import pickle
          # Dump the trained Naive Bayes classifier with Pickle
          DT_pkl_filename = '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()
```

```
In [30]:  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
```



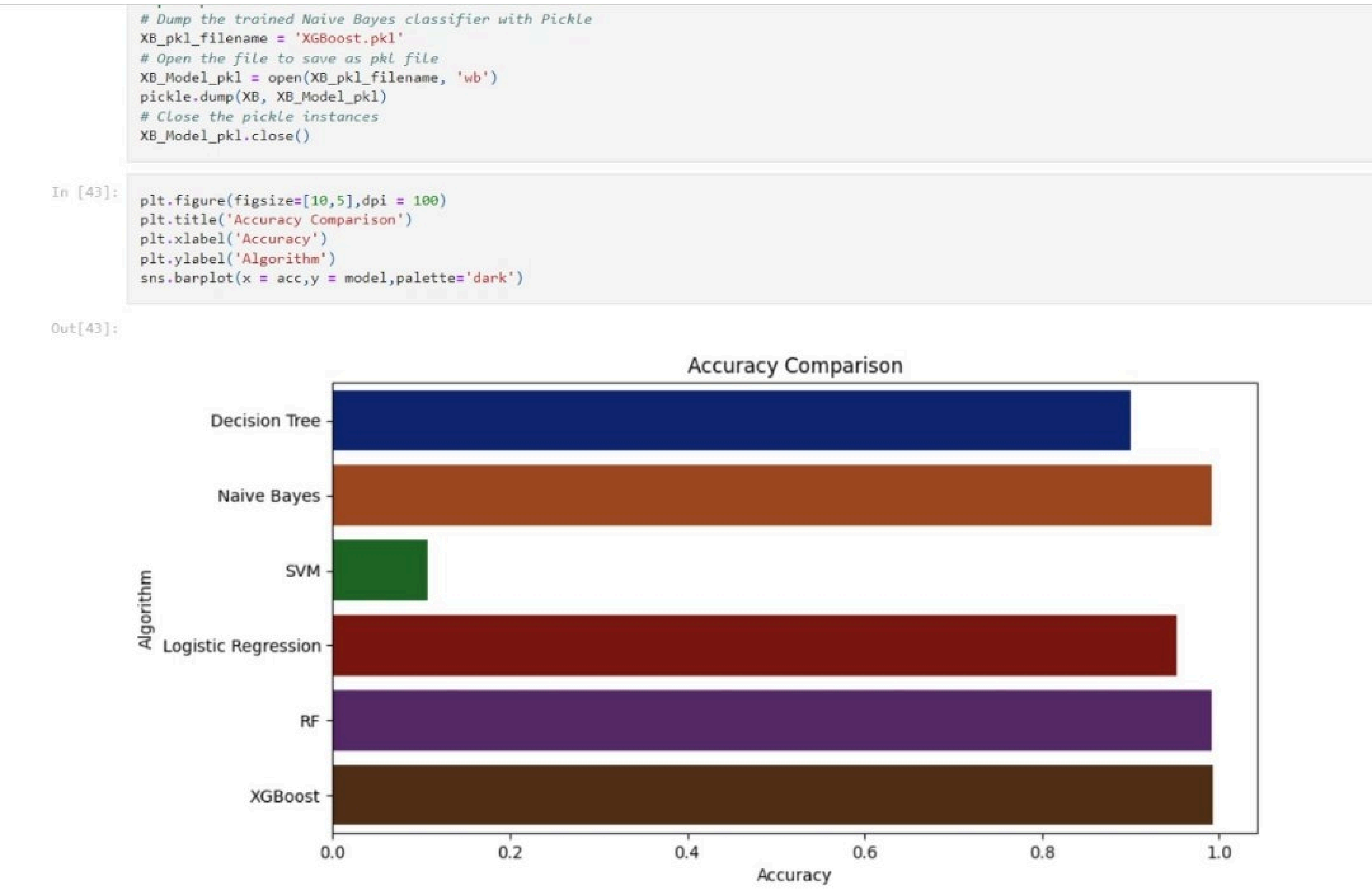
```
In [39]:
# Cross validation score (Random Forest)
score = cross_val_score(RF,features,target,cv=5)
score
```

```
Out[39]: array([0.99772727, 0.99545455, 0.99772727, 0.99318182, 0.98863636])
```

```
In [40]:
import pickle
# Dump the trained Naive Bayes classifier with Pickle
RF_pkl_filename = '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()
```

```
In [41]:
import xgboost as xgb
XB = xgb.XGBClassifier()
XB.fit(Xtrain,Ytrain)

predicted_values = XB.predict(Xtest)

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

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

```
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       1.00      1.00      1.00        22
      cotton       1.00      1.00      1.00        20
      grapes       1.00      1.00      1.00        18
        jute       0.96      0.93      0.95        28
 kidneybeans       1.00      1.00      1.00        14
```

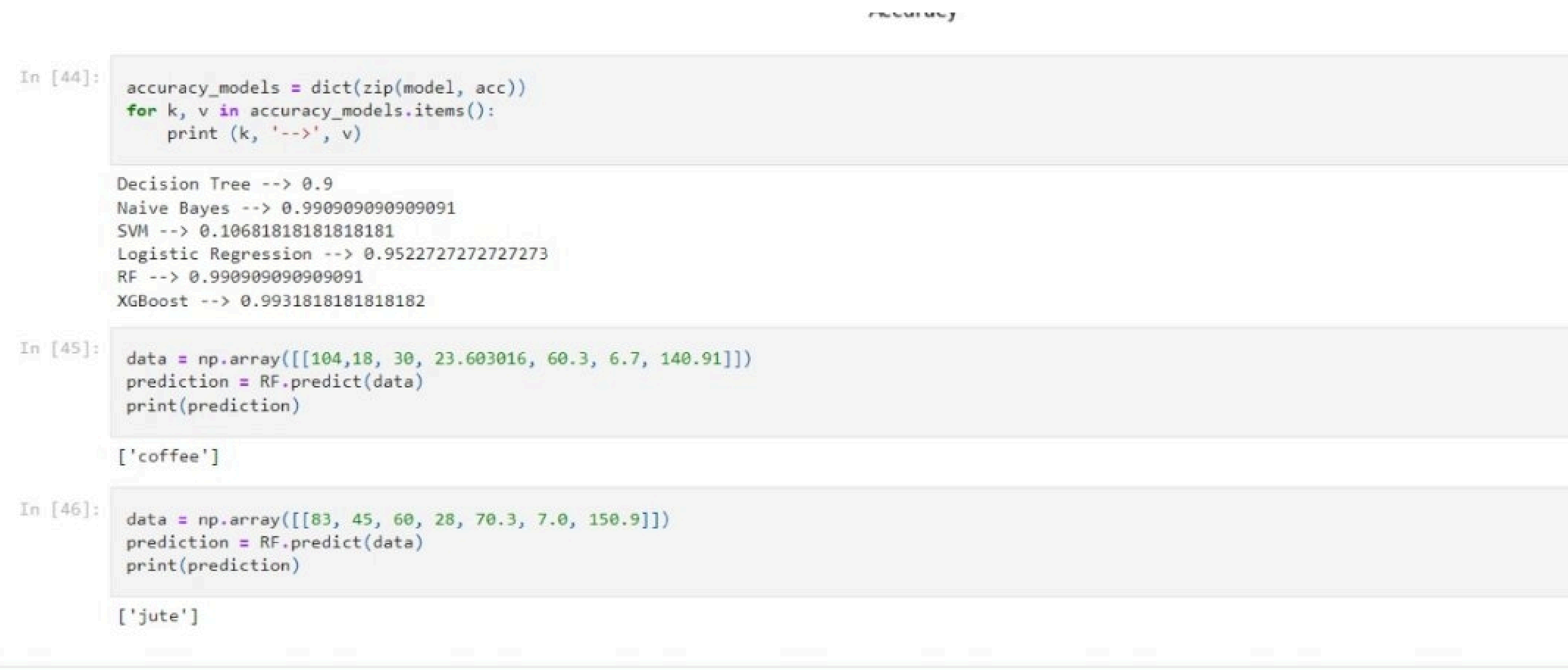


```
# Dump the trained Naive Bayes classifier with Pickle
XB_pkl_filename = 'XGBoost.pkl'
# Open the file to save as pkl file
XB_Model_pkl = open(XB_pkl_filename, 'wb')
pickle.dump(XB, XB_Model_pkl)
# Close the pickle instances
XB_Model_pkl.close()
```

In [43]:

```
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')
```

Out[43]:



Accuracy

In [44]:
```
accuracy_models = dict(zip(model, acc))
for k, v in accuracy_models.items():
    print (k, '-->', v)
```

```
Decision Tree --> 0.9
Naive Bayes --> 0.990909090909091
SVM --> 0.10681818181818181
Logistic Regression --> 0.9522727272727273
RF --> 0.990909090909091
XGBoost --> 0.9931818181818182
```

In [45]:
```
data = np.array([[104,18, 30, 23.603016, 60.3, 6.7, 140.91]])
prediction = RF.predict(data)
print(prediction)
```

```
['coffee']
```

In [46]:
```
data = np.array([[83, 45, 60, 28, 70.3, 7.0, 150.9]])
prediction = RF.predict(data)
print(prediction)
```

```
['jute']
```