Agricuture Production in India from 2001-2014

Data Analysis used ML

In [27]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn import svm
```

In [28]:

```
data = pd.read_csv("C:/Users/pc/Desktop/Agriculture/datasets1.csv")
data.head(10)
```

Out[28]:

	Crop	State	Cost of Cultivation (`/Hectare) A2+FL	Cost of Cultivation (`/Hectare) C2	Cost of Production (`/Quintal) C2	Yield (Quintal/ Hectare)
0	ARHAR	Uttar Pradesh	9794.05	23076.74	1941.55	9.83
1	ARHAR	Karnataka	10593.15	16528.68	2172.46	7.47
2	ARHAR	Gujarat	13468.82	19551.90	1898.30	9.59
3	ARHAR	Andhra Pradesh	17051.66	24171.65	3670.54	6.42
4	ARHAR	Maharashtra	17130.55	25270.26	2775.80	8.72
5	COTTON	Maharashtra	23711.44	33116.82	2539.47	12.69
6	COTTON	Punjab	29047.10	50828.83	2003.76	24.39
7	COTTON	Andhra Pradesh	29140.77	44756.72	2509.99	17.83
8	COTTON	Gujarat	29616.09	42070.44	2179.26	19.05
9	COTTON	Haryana	29918.97	44018.18	2127.35	19.90

In [29]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49 entries, 0 to 48
Data columns (total 6 columns):
 #   Column                                Non-Null Count  Dtype  
---  ------                                --------------  -----  
 0   Crop                                  49 non-null     object 
 1   State                                 49 non-null     object 
 2   Cost of Cultivation (`/Hectare) A2+FL  49 non-null     float64
 3   Cost of Cultivation (`/Hectare) C2     49 non-null     float64
 4   Cost of Production (`/Quintal) C2      49 non-null     float64
 5   Yield (Quintal/ Hectare)              49 non-null     float64
dtypes: float64(4), object(2)
memory usage: 2.4+ KB
```

In [30]:

```
data.describe()
```

Out[30]:

	Cost of Cultivation (`/Hectare) A2+FL	Cost of Cultivation (`/Hectare) C2	Cost of Production (`/Quintal) C2	Yield (Quintal/ Hectare)
count	49.000000	49.000000	49.000000	49.000000
mean	20363.537347	31364.666735	1620.537755	98.086735
std	13561.435306	20095.783569	1104.990472	245.293123
min	5483.540000	7868.640000	85.790000	1.320000
25%	12774.410000	19259.840000	732.620000	9.590000
50%	17022.000000	25909.050000	1595.560000	13.700000
75%	24731.060000	35423.480000	2228.970000	36.610000
max	66335.060000	91442.630000	5777.480000	1015.450000

In [31]:

```
data.corr()
```

Out[31]:

	Cost of Cultivation (`/Hectare) A2+FL	Cost of Cultivation (`/Hectare) C2	Cost of Production (`/Quintal) C2	Yield (Quintal/ Hectare)
Cost of Cultivation (`/Hectare) A2+FL	1.000000	0.981225	-0.434422	0.863400
Cost of Cultivation (`/Hectare) C2	0.981225	1.000000	-0.497092	0.866424
Cost of Production (`/Quintal) C2	-0.434422	-0.497092	1.000000	-0.487272
Yield (Quintal/ Hectare)	0.863400	0.866424	-0.487272	1.000000

In [32]:

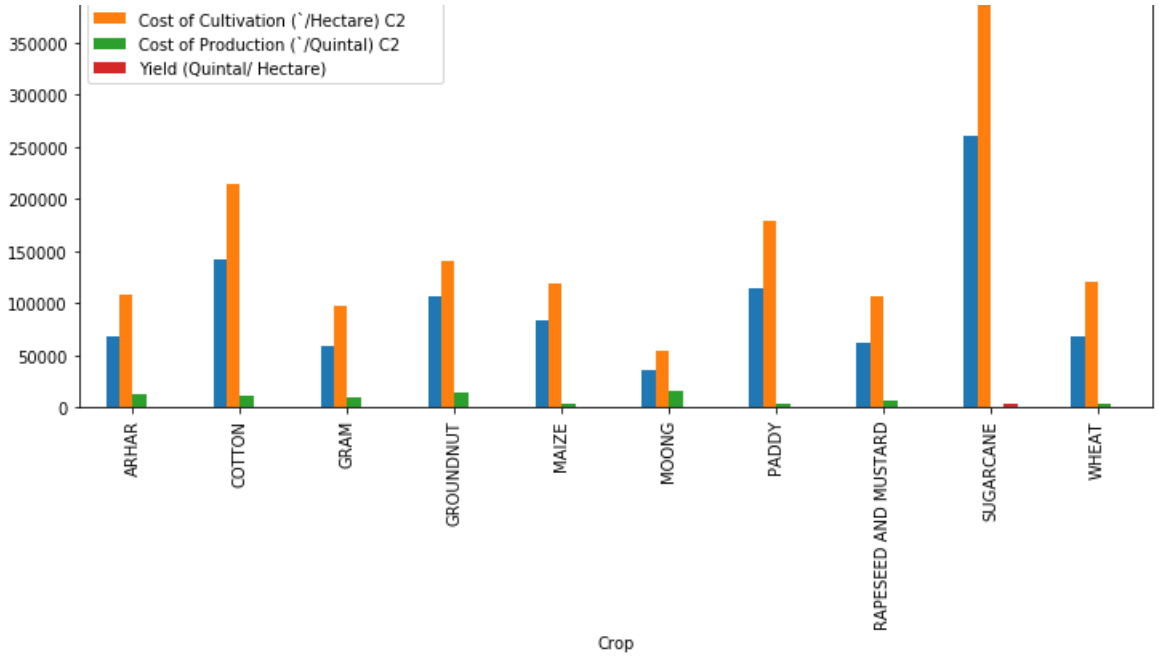
```
data.groupby('Crop').sum()
```

Out[32]:

Crop	Cost of Cultivation (`/Hectare) A2+FL	Cost of Cultivation (`/Hectare) C2	Cost of Production (`/Quintal) C2	Yield (Quintal/ Hectare)
ARHAR	68038.23	108599.23	12458.65	42.03
COTTON	141434.37	214790.99	11359.83	93.86
GRAM	58597.55	96543.87	8963.00	52.79
GROUNDNUT	106413.91	140940.38	13523.19	51.44
MAIZE	83050.75	119186.49	3872.83	153.99
MOONG	35593.35	53881.98	14950.50	20.98
PADDY	114050.70	178841.11	3638.67	231.48
RAPESEED AND MUSTARD	61302.45	106117.16	7077.97	71.60
SUGARCANE	260823.58	398275.13	493.24	3952.48
WHEAT	68508.44	119692.33	3068.47	135.60

In [33]:

```
data.groupby('Crop').sum().plot(kind='bar', figsize=(12,5));
```

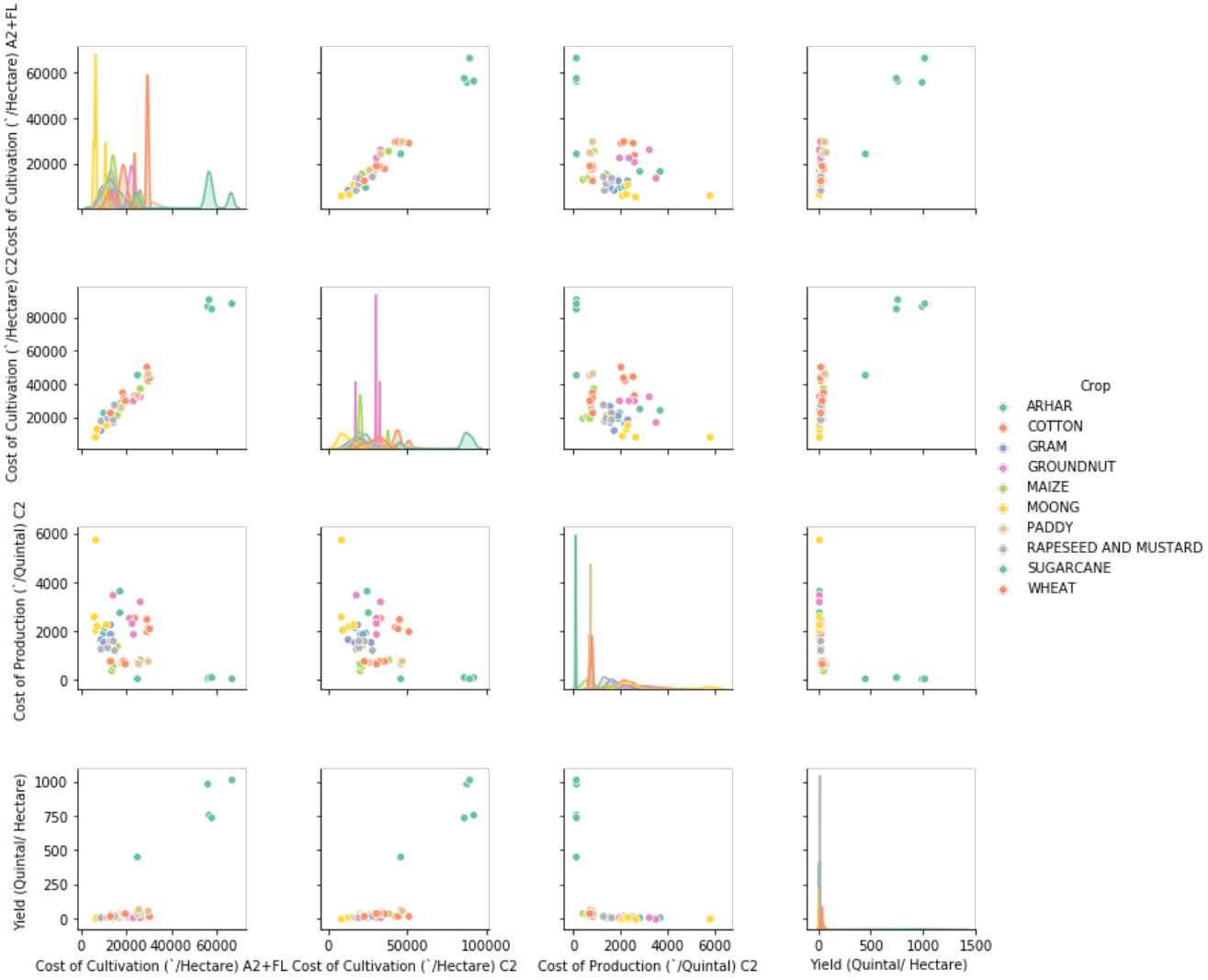


In [34]:

```
sns.pairplot(data=data, hue='Crop', palette='Set2')
```

Out[34]:

```
<seaborn.axisgrid.PairGrid at 0x5894ed58c8>
```



In [35]:

```
print(data.groupby('Crop').size())
```

```
Crop
ARHAR                   5
COTTON                  5
GRAM                    5
GROUNDNUT               5
MAIZE                   5
MOONG                   5
PADDY                   5
RAPESEED AND MUSTARD    5
SUGARCANE               5
WHEAT                   4
dtype: int64
```

In [36]:

```
print(data.groupby('State').size())
```

```
State
Andhra Pradesh    8
Bihar             1
Gujarat           4
Haryana           2
Karnataka         5
Madhya Pradesh    3
Maharashtra       6
Orissa            2
Punjab            3
Rajasthan         5
Tamil Nadu        2
Uttar Pradesh     7
West Bengal       1
dtype: int64
```

In [11]:

```
X = data[['Cost of Cultivation (`/Hectare) A2+FL','Cost of Cultivation (`/Hectare) C2','Cost of Pr
oduction (`/Quintal) C2','Yield (Quintal/ Hectare) ']]
y = data['Crop']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=0)
X, val = train_test_split(X, test_size=0.2)
print(len(val), 'validation examples')
print(len(X), 'train examples')
print(len(y), 'test examples')
```

```
10 validation examples
39 train examples
49 test examples
```

In [12]:

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

```
[[-0.77366872 -0.72101221 -0.08120948 -0.36666819]
 [-0.24126739 -0.2948391  -0.81133114 -0.28265292]
 [ 0.29131015  0.61931295 -0.85074672 -0.1498154 ]
 [-1.00699702 -1.14570671  3.60520259 -0.40072228]
 [-0.12741955 -0.10772275 -0.83877722 -0.26454389]
 [-0.51941111 -0.59467114  0.22095916 -0.3693257 ]
 [-0.2088321   0.1539046  -0.73302343 -0.25452128]
 [ 0.13844256 -0.09649335  0.2389483  -0.35467143]
 [-0.50704278 -0.70021123  1.60435169 -0.38785234]
 [-0.99010123 -0.893816    0.50943966 -0.38333457]
 [-0.4533209  -0.5824843  -0.86042177 -0.3162894 ]
```

```
 [ 0.01895862 -0.08139315  0.79379364 -0.36029017]
 [ 2.40732125  2.57544001 -1.35965085  3.33835583]
 [-0.51628226 -0.58024823 -1.08231105 -0.24267638]
 [ 2.47431889  2.79602195 -1.33069549  2.47166601]
 [ 0.60081937  0.4674054   0.46607205 -0.33341136]
 [-0.55291062 -0.6358255   0.55193485 -0.37517222]
 [-1.07339664 -1.12691919  0.84546661 -0.39430629]
 [ 0.1911793   0.04511205  0.7803236  -0.35755673]
 [ 0.10767992 -0.08332454  0.62200698 -0.3602522 ]
 [-0.85119345 -0.68173547 -0.31880218 -0.35660762]
 [-0.45328483 -0.25460586 -0.07501536 -0.34237096]
 [-0.50055336 -0.61676065 -0.03020827 -0.36165689]
 [-0.71891314 -0.73725992  0.46013965 -0.37737416]
 [-0.56351888 -0.49720421  0.20733208 -0.36423847]
 [ 2.5473328   2.52998188 -1.34130402  2.41885749]
 [ 0.32824168  0.26607932 -0.70180853 -0.24370142]
 [ 0.56134521  0.88049082  0.31296373 -0.31313835]
 [-0.36910071 -0.52424458 -0.22479106 -0.35372232]
 [-0.66392931 -0.60844602 -0.26498308 -0.35432975]
 [-0.49693125 -0.54186384 -0.92766729 -0.28766422]
 [ 0.35540718  0.02467282  1.36299017 -0.37031277]
 [ 0.62183193  0.55926966  0.42078514 -0.33018438]
 [-0.43293759 -0.21944706 -0.34364848 -0.33003252]
 [-0.58909359 -0.4561088  -0.72826879 -0.31617551]
 [-0.27084856 -0.37678277  1.76708271 -0.38136042]
 [-0.77435138 -0.42842369  0.25869098 -0.36841455]
 [-0.13711136 -0.01214948 -0.76352296 -0.27289606]
 [ 3.14822889  2.68200828 -1.36029643  3.44936381]]
[[-0.70589754 -0.79184014  0.53759239 -0.38029742]
 [-0.2653755  -0.32496733  0.98650074 -0.3726286 ]
 [-1.02339817 -1.08453666  0.369592   -0.39035799]
 [-0.27290625 -0.18940513 -0.79599414 -0.26674583]
 [ 0.26191629  0.04177751 -0.79718934 -0.25752047]
 [-0.56758639 -0.45278135 -0.04315487 -0.35421586]
 [ 0.60420144  0.67397476 -0.74602237 -0.19313281]
 [ 0.24854479  0.61687312 -1.353448    1.29845143]
 [ 0.56784365  0.59410256  0.75460489 -0.33804301]
 [-0.86047177 -0.92204256  0.04068393 -0.37980388]]
```

In [13]:

```
lda = LinearDiscriminantAnalysis()
lda.fit(X_train, y_train)
lda_prediction = lda.predict(X_test)
print('Accuracy of LDA classifier on training set: {:.2f}'
     .format(lda.score(X_train, y_train)))
print('Accuracy of LDA classifier on test set: {:.2f}'
     .format(lda.score(X_test, y_test)))
print(accuracy_score(lda_prediction, y_test))
print(confusion_matrix(lda_prediction, y_test))
print(classification_report(lda_prediction, y_test))
```

```
Accuracy of LDA classifier on training set: 0.64
Accuracy of LDA classifier on test set: 0.10
0.1
[[0 0 0 0 0 1 0 0 0 0]
 [0 1 0 0 0 0 1 0 0 0]
 [0 0 0 0 0 1 0 1 0 0]
 [1 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0 0]
 [0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 1 0]
 [0 0 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0 0]]
                      precision    recall  f1-score   support

              ARHAR       0.00      0.00      0.00         1
             COTTON       1.00      0.50      0.67         2
               GRAM       0.00      0.00      0.00         2
         GROUNDNUT       0.00      0.00      0.00         1
             MAIZE       0.00      0.00      0.00         1
             MOONG       0.00      0.00      0.00         0
             PADDY       0.00      0.00      0.00         1
RAPESEED AND MUSTARD       0.00      0.00      0.00         1
```

```
            SUGARCANE       0.00      0.00      0.00         0
                WHEAT       0.00      0.00      0.00         1

             accuracy                           0.10        10
            macro avg       0.10      0.05      0.07        10
         weighted avg       0.20      0.10      0.13        10
```

```
E:\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1272: UndefinedMetricWarning: Pr
ecision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
E:\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1272: UndefinedMetricWarning: Re
call and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use `zero_di
vision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [14]:

```
SVC_model = svm.SVC(kernel='rbf')
SVC_model.fit(X_train, y_train)
print('Accuracy of SVC_model on training set: {:.2f}'
     .format(SVC_model.score(X_train, y_train)))
print('Accuracy of SVC on test set: {:.2f}'
     .format(SVC_model.score(X_test, y_test)))
SVC_prediction = SVC_model.predict(X_test)
print(accuracy_score(SVC_prediction, y_test))
print(confusion_matrix(SVC_prediction, y_test))
print(classification_report(SVC_prediction, y_test))
```

```
Accuracy of SVC_model on training set: 0.54
Accuracy of SVC on test set: 0.10
0.1
[[0 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 2 0 1 0 0]
 [1 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 3 0 1 0]
 [0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0]
 [0 0 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0]]
                      precision    recall  f1-score   support

               ARHAR       0.00      0.00      0.00         0
              COTTON       1.00      1.00      1.00         1
                GRAM       0.00      0.00      0.00         3
           GROUNDNUT       0.00      0.00      0.00         1
               MAIZE       0.00      0.00      0.00         4
               MOONG       0.00      0.00      0.00         0
               PADDY       0.00      0.00      0.00         0
RAPESEED AND MUSTARD       0.00      0.00      0.00         1
           SUGARCANE       0.00      0.00      0.00         0

            accuracy                           0.10        10
           macro avg       0.11      0.11      0.11        10
        weighted avg       0.10      0.10      0.10        10
```

```
E:\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1272: UndefinedMetricWarning: Pr
ecision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
E:\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1272: UndefinedMetricWarning: Re
call and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use `zero_di
vision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [15]:

```
svm = SVC()
svm.fit(X_train, y_train)
print('Accuracy of svm on training set: {:.2f}'
```

```
     .format(svm.score(X_train, y_train)))
print('Accuracy of svm on test set: {:.2f}'
     .format(svm.score(X_test, y_test)))
svm_prediction = svm.predict(X_test)
print(accuracy_score(svm_prediction, y_test))
print(confusion_matrix(svm_prediction, y_test))
print(classification_report(svm_prediction, y_test))
```

```
Accuracy of svm on training set: 0.54
Accuracy of svm on test set: 0.10
0.1
[[0 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 2 0 1 0 0]
 [1 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 3 0 1 0]
 [0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0]
 [0 0 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0]]
                      precision    recall  f1-score   support

              ARHAR       0.00      0.00      0.00         0
             COTTON       1.00      1.00      1.00         1
               GRAM       0.00      0.00      0.00         3
         GROUNDNUT       0.00      0.00      0.00         1
             MAIZE       0.00      0.00      0.00         4
             MOONG       0.00      0.00      0.00         0
             PADDY       0.00      0.00      0.00         0
RAPESEED AND MUSTARD       0.00      0.00      0.00         1
         SUGARCANE       0.00      0.00      0.00         0

          accuracy                           0.10        10
         macro avg       0.11      0.11      0.11        10
      weighted avg       0.10      0.10      0.10        10
```

```
E:\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1272: UndefinedMetricWarning: Pr
ecision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
E:\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1272: UndefinedMetricWarning: Re
call and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use `zero_di
vision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [16]:

```
print(SVC_prediction)
```

```
['GRAM' 'GROUNDNUT' 'GRAM' 'MAIZE' 'MAIZE' 'GRAM' 'MAIZE' 'MAIZE' 'COTTON'
 'RAPESEED AND MUSTARD']
```

In [17]:

```
KNN_model = KNeighborsClassifier(n_neighbors=10)
KNN_model.fit(X_train, y_train)
print('Accuracy of K-NN classifier on training set: {:.2f}'
     .format(KNN_model.score(X_train, y_train)))
print('Accuracy of K-NN classifier on test set: {:.2f}'
     .format(KNN_model.score(X_test, y_test)))
KNN_prediction = KNN_model.predict(X_test)
print(accuracy_score(KNN_prediction, y_test))
print(confusion_matrix(KNN_prediction, y_test))
print(classification_report(KNN_prediction, y_test))
```

```
Accuracy of K-NN classifier on training set: 0.59
Accuracy of K-NN classifier on test set: 0.20
0.2
[[0 0 1 0 0 2 0 0 0 0]
 [0 1 0 0 0 0 1 0 0 0]
 [0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0]
```

```
[0 0 0 0 0 0 1 0 1 0]
[0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 1 0 0]
[0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 1 0 0 0]]
                     precision    recall  f1-score   support

             ARHAR       0.00      0.00      0.00         3
            COTTON       1.00      0.50      0.67         2
              GRAM       0.00      0.00      0.00         0
         GROUNDNUT       0.00      0.00      0.00         1
             MAIZE       0.00      0.00      0.00         2
             MOONG       0.00      0.00      0.00         0
             PADDY       0.00      0.00      0.00         0
RAPESEED AND MUSTARD       1.00      1.00      1.00         1
         SUGARCANE       0.00      0.00      0.00         0
             WHEAT       0.00      0.00      0.00         1

          accuracy                           0.20        10
         macro avg       0.20      0.15      0.17        10
      weighted avg       0.30      0.20      0.23        10
```

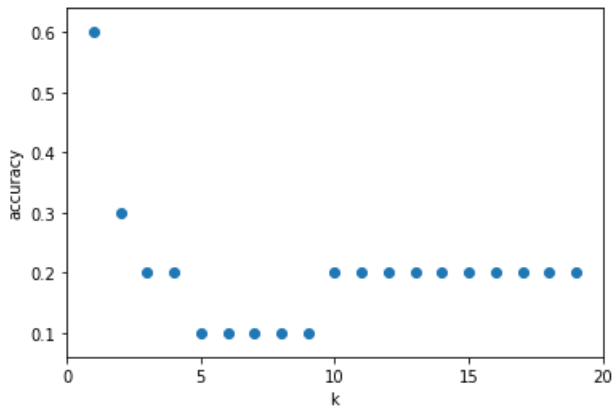
```
E:\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1272: UndefinedMetricWarning: Pr
ecision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
E:\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1272: UndefinedMetricWarning: Re
call and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use `zero_di
vision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [21]:

```
k_range = range(1, 20)
scores = []
for k in k_range:
    KNN = KNeighborsClassifier(n_neighbors = k)
    KNN.fit(X_train, y_train)
    scores.append(KNN.score(X_test, y_test))
plt.figure()
plt.xlabel('k')
plt.ylabel('accuracy','O')
plt.scatter(k_range, scores)
plt.xticks([0,5,10,15,20])
```

Out[21]:

```
([<matplotlib.axis.XTick at 0x5894cff0c8>,
  <matplotlib.axis.XTick at 0x5894d57388>,
  <matplotlib.axis.XTick at 0x5894d53448>,
  <matplotlib.axis.XTick at 0x5894d9e908>,
  <matplotlib.axis.XTick at 0x5894d9ee48>],
 <a list of 5 Text xticklabel objects>)
```



In [22]:

In [22]:

```
print(KNN_prediction)
```

```
['ARHAR' 'GROUNDNUT' 'ARHAR' 'MAIZE' 'WHEAT' 'RAPESEED AND MUSTARD'
 'COTTON' 'MAIZE' 'COTTON' 'ARHAR']
```

In [23]:

```
NB_model = GaussianNB()
NB_model.fit(X_train,y_train)
print('Accuracy of GaussianNB on training set: {:.2f}'
     .format(NB_model.score(X_train, y_train)))
print('Accuracy of GaussianNB on test set: {:.2f}'
     .format(NB_model.score(X_test, y_test)))
NB_prediction = NB_model.predict(X_test)
print(accuracy_score(NB_prediction, y_test))
print(confusion_matrix(NB_prediction, y_test))
print(classification_report(NB_prediction, y_test))
```

```
Accuracy of GaussianNB on training set: 0.72
Accuracy of GaussianNB on test set: 0.50
0.5
[[0 0 1 0 1 0 0 0 0]
 [0 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0]
 [1 0 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 0 0]
 [0 0 0 0 0 2 0 0 0]
 [0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0]
 [0 0 0 0 0 1 0 0 0]]
                      precision    recall  f1-score   support

               ARHAR       0.00      0.00      0.00         2
              COTTON       1.00      1.00      1.00         1
                GRAM       0.00      0.00      0.00         1
           GROUNDNUT       0.00      0.00      0.00         1
               MOONG       0.50      1.00      0.67         1
               PADDY       0.67      1.00      0.80         2
RAPESEED AND MUSTARD       0.00      0.00      0.00         0
           SUGARCANE       1.00      1.00      1.00         1
               WHEAT       0.00      0.00      0.00         1

            accuracy                           0.50        10
           macro avg       0.35      0.44      0.39        10
        weighted avg       0.38      0.50      0.43        10
```

```
E:\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1272: UndefinedMetricWarning: Pr
ecision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
E:\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1272: UndefinedMetricWarning: Re
call and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use `zero_di
vision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [24]:

```
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
print(clf)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

In [25]:

```
print('Accuracy of Decision Tree classifier on training set: {:.2f}'
     .format(clf.score(X_train, y_train)))
print('Accuracy of Decision Tree classifier on test set: {:.2f}'
     .format(clf.score(X_test, y_test)))
clf_prediction = clf.predict(X_test)
print(accuracy_score(clf_prediction, y_test))
print(confusion_matrix(clf_prediction, y_test))
print(classification_report(clf_prediction, y_test))
```

```
Accuracy of Decision Tree classifier on training set: 1.00
Accuracy of Decision Tree classifier on test set: 0.50
0.5
[[0 0 0 0 1 0 0 0 0]
 [0 1 0 0 0 0 0 0 0]
 [1 0 1 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0]
 [0 0 0 0 1 0 0 0 0]
 [0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 1 0]
 [0 0 0 0 0 2 0 0 0]]
                      precision    recall  f1-score   support

              ARHAR       0.00      0.00      0.00         1
             COTTON       1.00      1.00      1.00         1
               GRAM       1.00      0.50      0.67         2
              MAIZE       0.00      0.00      0.00         1
              MOONG       0.50      1.00      0.67         1
              PADDY       0.00      0.00      0.00         0
RAPESEED AND MUSTARD       1.00      1.00      1.00         1
          SUGARCANE       1.00      1.00      1.00         1
              WHEAT       0.00      0.00      0.00         2

           accuracy                           0.50        10
          macro avg       0.50      0.50      0.48        10
       weighted avg       0.55      0.50      0.50        10
```

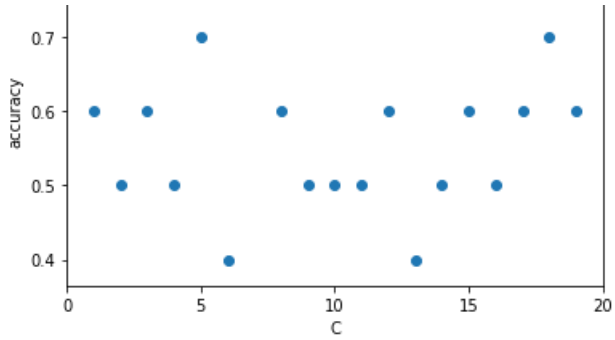
```
E:\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1272: UndefinedMetricWarning: Pr
ecision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
E:\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1272: UndefinedMetricWarning: Re
call and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use `zero_di
vision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [26]:

```
C_range = range(1, 20)
scores = []
for C in C_range:
    clf = DecisionTreeClassifier()
    clf.fit(X_train, y_train)
    scores.append(clf.score(X_test, y_test))
plt.figure()
plt.xlabel('C')
plt.ylabel('accuracy')
plt.scatter(C_range, scores)
plt.xticks([0,5,10,15,20])
```

Out[26]:

```
([<matplotlib.axis.XTick at 0x5894dd9388>,
  <matplotlib.axis.XTick at 0x5894d120c8>,
  <matplotlib.axis.XTick at 0x5894d09048>,
  <matplotlib.axis.XTick at 0x5894e026c8>,
  <matplotlib.axis.XTick at 0x5894e02d48>],
 <a list of 5 Text xticklabel objects>)
```



In [56]:

```
print(clf_prediction)
```

```
['ARHAR' 'ARHAR' 'MOONG' 'PADDY' 'MAIZE' 'GRAM' 'PADDY' 'PADDY' 'COTTON'
 'GRAM']
```

In [ ]:

```

```