## IMPORTING LIBRARIES

```
import numpy as np
import pandas as pd
```

## IMPORTING DATA

```
crop = pd.read_csv('/content/Crop
crop.head()
```

|   | N | P | K | temperature |   |
|---|---|---|---|---|---|
| **0** | 90 | 42 | 43 | 20.879744 | 8 |
| **1** | 85 | 58 | 41 | 21.770462 | 8 |
| **2** | 60 | 55 | 44 | 23.004459 | 8 |
| **3** | 74 | 35 | 40 | 26.491096 | 8 |
| **4** | 78 | 42 | 42 | 20.130175 | 8 |

Next steps: code crop reco

```
crop.shape
```

```
(2200, 8)
```

```
crop.info()
```

```
<class 'pandas.core.frame.Dat
RangeIndex: 2200 entries, 0 t
Data columns (total 8 columns
 #   Column       Non-Null Co
---  ------       -----------
 0   N            2200 non-nu
 1   P            2200 non-nu
 2   K            2200 non-nu
 3   temperature  2200 non-nu
 4   humidity     2200 non-nu
 5   ph           2200 non-nu
 6   rainfall     2200 non-nu
 7   label        2200 non-nu
dtypes: float64(4), int64(3),
memory usage: 137.6+ KB
```

NameError TypeError AttributeError ✕ •••

You

Please explain the error:

```
AttributeError: 'numpy.ndarray' object has no attribute '
```

Gemini

# Hypothesis

The error message `AttributeError: 'numpy.ndarray' object has no attribute 'label'` indicates that you are trying to access an attribute named 'label' on a NumPy array, but NumPy arrays do not have such an attribute.

In this case, `crop.label.unique()` returns a NumPy array containing the unique values of the 'label' column. Then, you are trying to access the 'label' attribute of this array using `.label`. Since NumPy arrays do not have a 'label' attribute, the error is raised.

# Suggested Changes

```
crop['label'].unique()
```

Use code with caution

**Explanation:**

1. **`crop['label']`:** This selects the 'label' column from the 'crop' DataFrame.
2. **`.unique()`:** This method is applied to the 'label' column (which is a Pandas Series) and returns a NumPy array containing the unique values in that column.
3. By simply removing `.label`, you are now correctly accessing the unique values of the 'label' column as a NumPy array, avoiding the `AttributeError`.

▶ Sources

Rate this answer

```
crop.isnull().sum()
```

| | 0 |
|---|---|
| **N** | 0 |
| **P** | 0 |
| **K** | 0 |
| **temperature** | 0 |
| **humidity** | 0 |
| **ph** | 0 |
| **rainfall** | 0 |
| **label** | 0 |

**dtype:** int64

```
crop.duplicated().sum()
```

```
0
```

```
crop.describe()
```

| | N | |
|---|---|---|
| **count** | 2200.000000 | 2200.00000 |
| **mean** | 50.551818 | 53.3627 |
| **std** | 36.917334 | 32.9858 |
| **min** | 0.000000 | 5.0000 |
| **25%** | 21.000000 | 28.0000 |
| **50%** | 37.000000 | 51.0000 |
| **75%** | 84.250000 | 68.0000 |
| **max** | 140.000000 | 145.0000 |

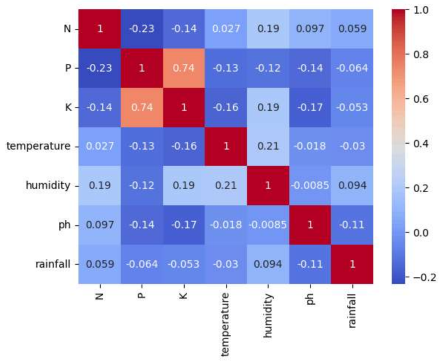
## EXPLORING DATA

```
# Select only numerical features
numerical_features = crop.select_
corr = crop[numerical_features].c
corr
```

| | N | |
|---|---|---|
| **N** | 1.000000 | -0.2314 |
| **P** | -0.231460 | 1.0000 |
| **K** | -0.140512 | 0.7362 |
| **temperature** | 0.026504 | -0.1275 |
| **humidity** | 0.190688 | -0.1187 |
| **ph** | 0.096683 | -0.1380 |
| **rainfall** | 0.059020 | -0.0638 |

Next steps: code corr reco

```
import seaborn as sns
import matplotlib.pyplot as plt


sns.heatmap(corr, annot=True, cba
<Axes: >
```



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

```
crop['label'].value_counts()
```

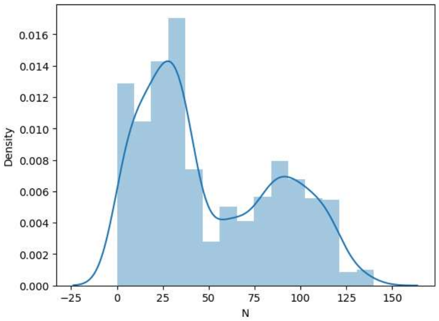
```
                count
        label
         rice     100
        maize     100
         jute     100
       cotton     100
      coconut     100
       papaya     100
       orange     100
        apple     100
    muskmelon     100
   watermelon     100
       grapes     100
        mango     100
       banana     100
  pomegranate     100
       lentil     100
    blackgram     100
     mungbean     100
    mothbeans     100
   pigeonpeas     100
  kidneybeans     100
     chickpea     100
       coffee     100

dtype: int64
```

```
crop['label'].unique().size
```

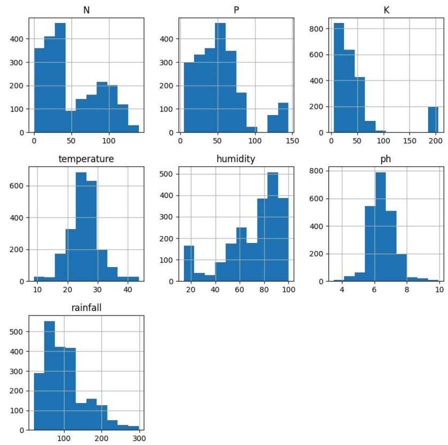
```
22
```

```
sns.distplot(crop['N'])
plt.show()
```

```
<ipython-input-232-b5d6dedccd

`distplot` is a deprecated fu

Please adapt your code to use
similar flexibility) or `hist

For a guide to updating your
https://gist.github.com/mwask

  sns.distplot(crop['N'])
```



```
crop.hist(figsize=(10,10))
plt.show()
```



```
crop.label.unique()
```

```
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)
```

## ENCODING

```
crop_dict = {
 'rice':    1,
'maize': 2,
'jute': 3,
'cotton': 4,
'coconut': 5,
'papaya': 6,
'orange': 7,
'apple': 8,
'muskmelon': 9,
'watermelon': 10,
'grapes': 11,
'mango': 12,
'banana':   13,
'pomegranate': 14,
'lentil':   15,
'blackgram': 16,
'mungbean': 17,
'mothbeans': 18,
'pigeonpeas':   19,
'kidneybeans': 20,
'chickpea': 21,
'coffee': 22
}
crop['crop_num'] =crop['label'].m
```

```
crop['label'].value_counts() # Ch
```

| label | count |
|---|---|
| **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 |

**dtype:** int64

```
X=crop.drop('label', axis=1, inpl
```

```
X.head()
```

|   | N | P | K | temperature |
|---|---|---|---|---|
| **0** | 90 | 42 | 43 | 20.879744 |
| **1** | 85 | 58 | 41 | 21.770462 |
| **2** | 60 | 55 | 44 | 23.004459 |
| **3** | 74 | 35 | 40 | 26.491096 |
| **4** | 78 | 42 | 42 | 20.130175 |

Next steps: code X recomm

## Train Test Split

```
X = crop.drop('crop_num', axis=1)
y = crop['crop_num']
```

```
X.shape
```

```
(2200, 7)
```

```
y.shape
```

```
(2200,)
```

## TRAIN TEST SPLIT

```
from sklearn import preprocessing
from sklearn.model_selection impo
```

```
X_train ,X_test , y_train, y_test
```

```
X_train.shape
X_test.shape
```

```
(440, 7)
```

## Scale the features using MinMaxScalar

```
from sklearn.preprocessing import
ms = MinMaxScaler()
```

```
ms.fit(X_train)
X_train = ms.fit_transform(X_trai
X_test = ms.transform(X_test)
```

```
X_train
```

```
array([[0.12142857,
0.07857143, 0.045     , ...,
0.9089898 , 0.48532225,
        0.29685161],
       [0.26428571,
0.52857143, 0.07      , ...,
0.64257946, 0.56594073,
        0.17630752],
       [0.05      ,
0.48571429, 0.1       , ...,
0.57005802, 0.58835229,
        0.08931844],
       ...,
       [0.07857143,
0.22142857, 0.13      , ...,
0.43760347, 0.46198144,
        0.28719815],
       [0.07857143, 0.85
, 0.995     , ...,
0.76763665, 0.44420505,
        0.18346657],
       [0.22857143,
0.52142857, 0.085     , ...,
0.56099735, 0.54465022,
        0.11879596]])
```

### Standarization

```
from sklearn.preprocessing import
sc = StandardScaler()
sc.fit(X_train)
X_train = sc.fit_transform(X_trai
X_test = sc.transform(X_test)
```

```
X_train
```

```
array([[-9.03426596e-01,
-1.12616170e+00,
-6.68506601e-01, ...,
         9.36586183e-01,
1.93473784e-01,
5.14970176e-03],
       [-3.67051340e-01,
7.70358846e-01,
-5.70589522e-01, ...,
        -1.00470485e-01,
8.63917548e-01,
-6.05290566e-01],
       [-1.17161422e+00,
5.89737842e-01,
-4.53089028e-01, ...,
        -3.82774991e-01,
1.05029771e+00,
-1.04580687e+00],
       ...,
       [-1.06433917e+00,
-5.24091685e-01,
-3.35588533e-01, ...,
        -8.98381379e-01,
-6.34357580e-04,
-4.37358211e-02],
       [-1.06433917e+00,
2.12501638e+00,
3.05234239e+00, ...,
         3.86340190e-01,
-1.48467347e-01,
-5.69036842e-01],
       [-5.01145154e-01,
7.40255346e-01,
-5.11839275e-01, ...,
        -4.18045489e-01,
6.86860180e-01,
-8.96531475e-01]])
```

## Training Model

```
from sklearn.linear_model import
from sklearn.neighbors import KNe
from sklearn.tree import Decision
from sklearn.ensemble import Rand
from sklearn.svm import SVC
from sklearn.naive_bayes import G
from sklearn.tree import ExtraTre
from sklearn.metrics import accur


from mmap import MADV_DOFORK
models = {
    'logustic Regression' : Logis
    'Naive Bayes' : GaussianNB(),
    'Decision Tree' : DecisionTre
    'Extra Tree' : ExtraTreeClass
    'Random Forest' : RandomFores
    'SVM' : SVC(),
```

Enter a prompt here

0 / 1000

Responses may display inaccurate or offensive information that doesn't represent Google's views. Learn more