```
import numpy as np
import pandas as pd
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam
```

```
dataset = pd.read_csv('crop_data.csv')  # Replace with your dataset path

# Split dataset into features (X) and labels (y)
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

```
dataset.shape
```

```
dataset.info()
```

```
dataset.isnull().sum()
```

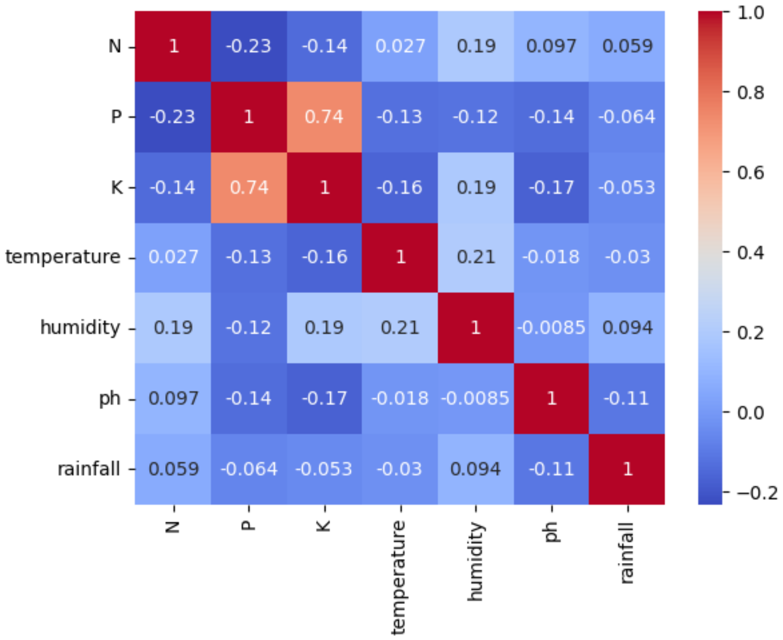
```
dataset.describe()
```

	N	P	K	temperature	humidity	ph	
count	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2
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	

```
corr = dataset.corr()
corr

import seaborn as sns
sns.heatmap(corr,annot=True,cbar=True, cmap='coolwarm')
```

```
<ipython-input-30-142c1a91294e>:1: FutureWarning: The default value of numeri
  corr = dataset.corr()
<Axes: >
```



```
dataset['label'].value_counts()
import matplotlib.pyplot as plt
sns.distplot(dataset['N'])
plt.show()
```

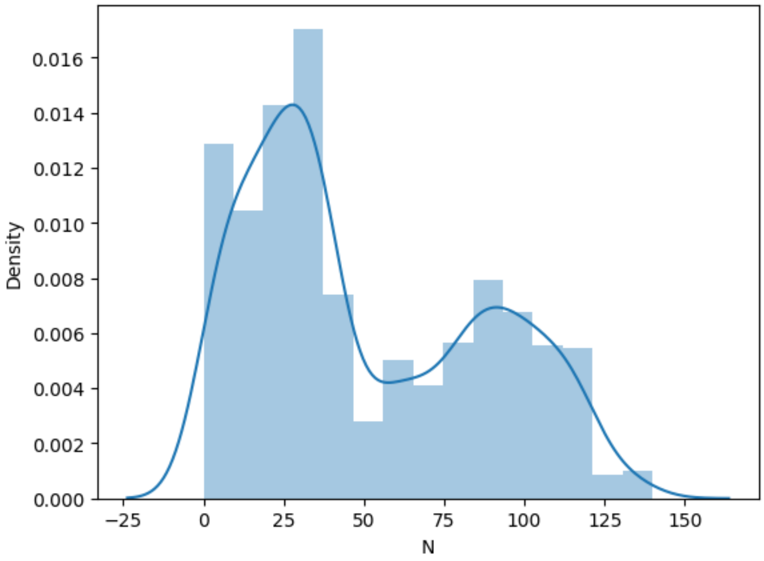
```
<ipython-input-31-166935e5c562>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

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



```
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
}
dataset['crop_num']=dataset['label'].map(crop_dict)
```

```
dataset['crop_num'].value_counts()
```

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

```
X
```

```
# Encode labels to numerical values
le = LabelEncoder()
y = le.fit_transform(y)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Normalize the input features (optional but recommended)
scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)


# Build the deep learning model
from tensorflow.keras.models import Sequential
model = Sequential()


# Input layer
model.add(Dense(units=64, activation='relu', input_dim=X_train.shape[1]))
model.add(Dropout(0.2))



# Hidden layers
model.add(Dense(units=128, activation='relu'))
model.add(Dropout(0.2))



# Output layer
model.add(Dense(units=len(np.unique(y)), activation='sigmoid'))

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.001), loss='sparse_categorical_crossentropy', metrics=['accuracy'])


# Train the model
history = model.fit(X_train, y_train, epochs=50, batch_size=32, validation_data=(X_test, y_test))


# Evaluate the model
test_loss, test_accuracy = model.evaluate(X_test, y_test)
print(f"Test Loss: {test_loss}, Test Accuracy: {test_accuracy}")
```

```
14/14 [==============================] - 0s 2ms/step - loss: 0.0637 - accuracy: 0.9795
Test Loss: 0.0636519268155098, Test Accuracy: 0.9795454740524292
```

```
# Make predictions on new data (adjust these values accordingly)
new_data = np.array([[58,53,45,38.79746068,41.82913698,12.2527,277]])
new_data = scaler.transform(new_data)
predictions = model.predict(new_data)
```

```
1/1 [==============================] - 0s 23ms/step
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but Stand
  warnings.warn(
```

```
# Decode the predicted labels
predicted_category = le.inverse_transform([np.argmax(predictions)])

# Define categories for output
categories = le.classes_


# Output the prediction
if test_accuracy > 0.7:  # Adjust this threshold as needed
    print(f'Predicted category: {categories[predicted_category[0]]}')
else:
    print('Cannot predict')

print(f'Accuracy on the test set: {test_accuracy:.2%}')
```

```
Predicted category: 18
Accuracy on the test set: 97.27%
```

```
def validate_input(N, P, K, temperature, humidity, pH, rainfall):
    if not (0 <= N <= 100 and 0 <= P <= 100 and 0 <= K <= 100):
        raise ValueError("Nutrient levels (N, P, K) should be between 0 and 100.")
    if not (-10 <= temperature <= 50):
        raise ValueError("Temperature should be between -10°C and 50°C.")
    if not (0 <= humidity <= 100):
        raise ValueError("Humidity should be between 0% and 100%.")
    if not (0 <= pH <= 14):
        raise ValueError("pH should be between 0 and 14.")
    if not (0 <= rainfall <= 1000):
        raise ValueError("Rainfall should be between 0 mm and 1000 mm.")
```

```
try:
    # Replace 'new_data' with your own input data
    new_data = [58,53,45,38.79746068,41.82913698,12.2527,277]
    validate_input(*new_data)

    scaled_new_data = scaler.transform([new_data])
    predictions = model.predict(scaled_new_data)

    # Decode the predicted label
    predicted_crop = le.inverse_transform([np.argmax(predictions)])

    crop_dict = {1: "Rice", 2: "Maize", 3: "Jute", 4: "Cotton", 5: "Coconut", 6: "Papaya", 7: "Orange",
                 8: "Apple", 9: "Muskmelon", 10: "Watermelon", 11: "Grapes", 12: "Mango", 13: "Banana",
                 14: "Pomegranate", 15: "Lentil", 16: "Blackgram", 17: "Mungbean", 18: "Mothbeans",
                 19: "Pigeonpeas", 20: "Kidneybeans", 21: "Chickpea", 22: "Coffee"}
    if predicted_crop[0] in crop_dict:
      dataset = crop_dict[predicted_crop[0]]
      print("{} is a best crop to be cultivated ".format(dataset))
    else:
      print("Sorry are not able to recommend a proper crop for this environment")
except ValueError as e:
    print(f"Input validation error: {e}")
```

```
1/1 [==============================] - 0s 37ms/step
Mungbean is a best crop to be cultivated
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but Stand
  warnings.warn(
```