```
#Importing Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

```
#Reading the dataset
crop=pd.read_csv("/Crop_recommendation.csv")
crop
```

| | 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 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **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 |

2200 rows × 8 columns

```
crop.head()
```

| | 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 |

```
crop.tail()
```

| | 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 |

```
crop.isna().sum()
```

```
N              0
P              0
K              0
temperature    0
humidity       0
ph             0
rainfall       0
```

```
label          0
dtype: int64
```

```
print("The size of the dataset:",crop.size)
print("Shape of the dataset:",crop.shape)
```

```
The size of the dataset: 17600
Shape of the dataset: (2200, 8)
```

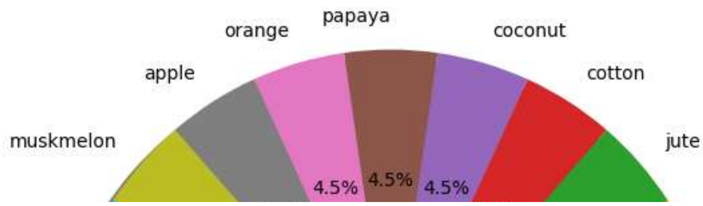
```
print("Number of columns in the dataset:\n",crop.columns)
```

```
Number of columns in the dataset:
 Index(['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall', 'label'], dtype='object')
```

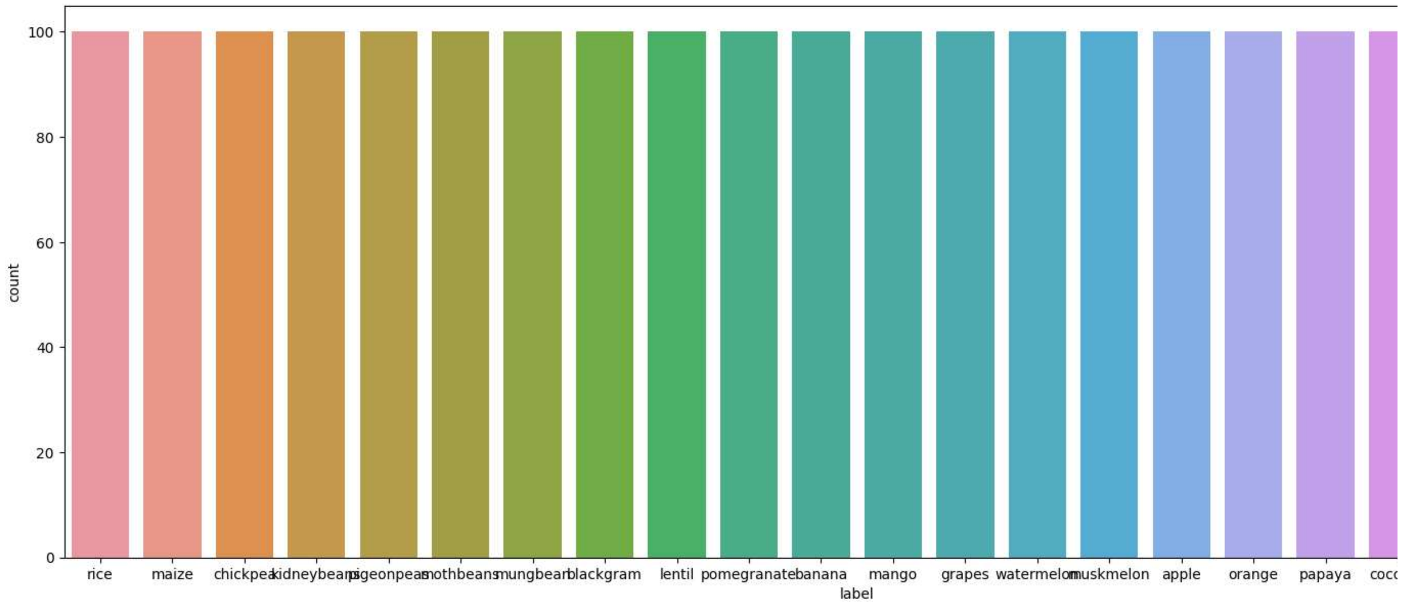
```
print("Value count of the labels:\n",crop['label'].value_counts())
```

```
Value count of the labels:
 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
```

```
#Data visualization
#Pie chart for label column
data_viz_df = crop.copy()
data_viz_df.head()
label_name = data_viz_df['label'].value_counts().index
val = data_viz_df['label'].value_counts().values
plt.figure(figsize = (8,8))
plt.pie(x = val , labels  = label_name , shadow = True , autopct = '%1.1f%%')
plt.show()
```




```
#Bar graph for Label column
plt.figure(figsize = (20 , 7))
sns.countplot(x = 'label' , data = data_viz_df)
plt.show()
```




```
#Label Encoding
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
crop['label'] = le.fit_transform(crop['label'])
crop.head()
```

| | N | P | K | temperature | humidity | ph | rainfall | label |
|---|---|---|---|---|---|---|---|---|
| **0** | 90 | 42 | 43 | 20.879744 | 82.002744 | 6.502985 | 202.935536 | 20 |
| **1** | 85 | 58 | 41 | 21.770462 | 80.319644 | 7.038096 | 226.655537 | 20 |
| **2** | 60 | 55 | 44 | 23.004459 | 82.320763 | 7.840207 | 263.964248 | 20 |
| **3** | 74 | 35 | 40 | 26.491096 | 80.158363 | 6.980401 | 242.864034 | 20 |
| **4** | 78 | 42 | 42 | 20.130175 | 81.604873 | 7.628473 | 262.717340 | 20 |

```
x=crop.drop('label',axis=1)
x
```

| | 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 |
| **2** | 60 | 55 | 44 | 23.004459 | 82.320763 | 7.840207 | 263.964248 |
| **3** | 74 | 35 | 40 | 26.491096 | 80.158363 | 6.980401 | 242.864034 |

```
y=crop['label']
y
```

```
0       20
1       20
2       20
3       20
4       20
        ..
2195     5
2196     5
2197     5
2198     5
2199     5
Name: label, Length: 2200, dtype: int64
```

```
#Train_Test_Split
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.33,random_state=0)
print("xtrain shape:",xtrain.shape)
print("xtest shape:",xtest.shape)
print("ytrain shape:",ytrain.shape)
print("ytest shape:",ytest.shape)
```

```
xtrain shape: (1474, 7)
xtest shape: (726, 7)
ytrain shape: (1474,)
ytest shape: (726,)
```

```
#LogisticRegression()
from sklearn.linear_model import LogisticRegression
LG = LogisticRegression()
LG.fit(xtrain,ytrain)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
#To check intercept and co-efficient
print('Intercept:',LG.intercept_)
print('Co-efficient:',LG.coef_)
```

```
Intercept: [-0.00780263 -0.01053188 -0.01657645  0.00332404 -0.00463209  0.00217817
 -0.01441515 -0.00392094 -0.00192072  0.01139565  0.00604579  0.01981117
  0.02530411  0.04437446 -0.00247394  0.00067193  0.00319134 -0.04082406
 -0.00357376  0.00243646 -0.0123666   0.00030511]
Co-efficient: [[-4.95073484e-01  2.37818458e-01  8.81830732e-01 -2.02632762e-01
  -2.15887867e-01 -5.59788206e-02 -2.65260000e-01]
 [ 4.05127140e-01  1.36164301e-01  1.71382690e-01 -1.54910037e-01
  -2.87219947e-01 -8.73894529e-02  6.77969943e-02]
 [-1.22341138e-01  4.17528443e-01 -4.32713598e-01  1.84071457e-01
   6.03238407e-02 -6.94249117e-02  6.43785233e-02]
 [ 7.08325343e-02  2.80986905e-01  4.72444119e-01 -2.58743623e-03
  -4.93916277e-01  3.47613283e-02  7.42472131e-02]
 [-3.39645974e-01 -5.62047213e-01 -1.45093902e-01 -4.47995478e-02
   2.49809393e-01 -5.68238347e-02  3.57349018e-01]
 [ 6.51856484e-01 -4.63425638e-01 -1.74397966e-01  7.00590637e-02
  -4.16995783e-01  2.76565987e-02  2.26341134e-01]
 [ 6.07064022e-01 -1.34120586e-01 -5.74851007e-01 -1.85494873e-01
   6.30183028e-02 -2.64960924e-02 -1.90339861e-02]
 [-4.88437684e-01  3.87346469e-01  1.04059784e+00 -1.27822994e-01
  -2.31305090e-01 -2.88194601e-02 -8.50592446e-01]
 [ 1.87619254e-01 -1.72504463e-01  8.65556339e-03 -1.27720022e-01
  -8.28986040e-02  6.62274965e-02  2.66109022e-01]
 [-1.94869829e-01  6.75571604e-01 -6.24381826e-02  2.12318514e-02
  -7.24283208e-01  6.29970737e-02  1.97017331e-01]
 [-2.03451192e-01  5.39442203e-01 -3.12328384e-01 -1.59233261e-01
   2.65899943e-01  5.12321440e-02 -1.67062430e-01]
 [ 4.51779041e-01 -4.11379427e-02 -3.08784109e-01  5.16554953e-02
  -7.66516820e-02  4.94236731e-02  9.55319619e-02]
```

```
[-2.92335907e-01 -1.84513905e-01  3.09791378e-01  5.71415316e-01
 -8.67931805e-02  9.61871208e-02  1.93947179e-01]
[-2.00695183e-01  1.74360039e-01 -7.08670458e-02  8.14513299e-01
 -2.31985140e-02  3.17598068e-01 -5.12918666e-02]
[-2.48551961e-01  2.41566023e-02 -4.41589441e-01  4.31574548e-03
  5.82246788e-01 -1.62197926e-02 -7.35054258e-02]
[ 5.58967400e-01 -3.63880997e-01  1.26053947e-01  7.76629075e-02
  3.39145879e-01  1.00746460e-03 -8.97768988e-01]
[-3.13038618e-01 -3.93235362e-01 -5.19932385e-01 -1.69364564e-02
  5.04927147e-01  4.25652560e-02  1.96768632e-01]
[-2.54664016e-01  1.16083229e-01  4.07403242e-01 -2.03504842e-01
  1.11101751e-01 -2.34937289e-01  1.65237761e-01]
[-2.47777779e-01  3.75688193e-01 -5.17238715e-01  8.52964584e-02
 -4.56960960e-02 -5.08523340e-02  2.60099344e-01]
[-3.01581873e-01 -3.96251359e-01  2.55926764e-01 -1.43333131e-01
  4.29057822e-01  2.21824551e-02  1.02038234e-01]
[ 1.91756984e-01 -2.23053944e-01 -2.53887706e-01 -4.75647887e-01
 -2.32198055e-02 -1.46840662e-01  3.54096485e-01]
[ 5.77461776e-01 -4.30975036e-01  1.40036163e-01 -3.55983415e-02
  1.02535187e-01  1.94397099e-03 -2.96443689e-01]]
```

```
#Predictions
y_pred=LG.predict(xtest)
y_pred
```

```
array([21, 21,  7,  3,  2,  8, 13,  9, 15,  1, 13,  5, 10, 14, 12,  0,  5,
       10,  5, 12,  4,  2,  9,  8,  6,  5, 10, 16, 13,  9, 19, 20, 11, 15,
        4,  6, 12, 12, 21, 13, 11,  2, 18, 21, 18, 14,  9,  9,  6, 14, 13,
        2,  0, 15, 18,  1, 17, 12, 10,  6, 16, 14, 21, 20, 15,  0,  7,  5,
        0, 16,  4, 19,  9, 11,  7, 13,  3, 11,  8, 12, 20, 13, 21, 21, 15,
        6, 11, 10, 13, 17,  2,  8, 14,  7, 14, 11,  5,  8, 10,  3, 16,  8,
       14,  1,  1, 20, 21,  5, 18, 15, 15, 12,  5,  7, 16, 19, 14, 10, 11,
        8, 19, 10, 16,  3,  3,  2, 19, 16,  3, 12, 13,  2, 15, 14,  6, 14,
        4, 19, 16,  2, 10,  7,  0,  5,  3,  0,  8, 12, 21, 17, 16,  4, 13,
        1, 19,  3, 21, 11,  0,  8, 10, 18,  8,  9,  9, 15, 20, 15,  1, 16,
        9,  0, 13,  4,  6, 14,  9, 19, 17, 16, 20, 17, 17,  9,  9,  1,  4,
       18, 20, 17, 11,  8, 13, 20, 11,  5, 18,  4,  3, 12,  4, 19,  6, 13,
       18, 16, 15, 11, 18,  1,  3,  2, 18, 16, 13, 14, 12, 17, 15, 19,  8,
       20,  2, 17,  2,  5, 11,  5, 16, 20, 13, 14, 16,  9, 19,  4, 12, 14,
        6, 20,  3, 14,  0, 18, 13, 20, 21,  2, 19, 16, 11,  7,  3, 18,  8,
       17, 19,  5, 12, 13,  8, 21, 19, 20,  7,  4,  8, 10,  3,  5,  5, 17,
       19, 11, 20,  3, 18, 16, 19, 18,  4,  9, 19, 15, 13, 12, 10,  1,  2,
       12,  9, 12,  6, 14,  4,  7,  7, 18, 17, 20, 20,  3, 15,  5, 21,  8,
        8, 13,  7, 15,  2, 13, 13,  3,  2, 12,  1, 12, 19,  8, 16, 15,  3,
       10,  6, 17,  7,  9, 10,  0, 20, 15,  0, 17,  2,  8,  3, 13, 10,  7,
       20,  9, 15, 12,  7, 17, 20,  5, 15, 13,  1, 17, 16,  9, 21, 18,  0,
       21, 21, 18,  9,  2,  9,  8,  4,  6,  9, 16,  6, 18, 19,  6,  6,  0,
        6,  0, 16, 11,  7,  1,  0, 13, 20,  9,  1, 20, 10,  3, 19,  1,  3,
       15, 19,  0, 10, 15, 16,  2, 15, 13, 12,  3, 19, 12,  3,  4, 15,  1,
       18, 17,  8,  2,  6, 20,  1,  4, 20,  2, 11, 16, 21, 20,  0,  7, 18,
        7,  3, 12,  8, 19, 11, 12,  7,  1, 14, 18,  1,  6,  2,  0,  0,  8,
        8, 21,  3,  1, 19,  1,  9,  7, 11,  5,  6,  8,  7,  5, 14,  2,  8,
       16, 18, 18, 15,  2, 21, 14, 21, 17, 14, 14, 14, 19, 16, 13,  0,  5,
        4, 11,  4,  7,  7,  3,  3, 12,  9, 12, 16, 14, 17, 18,  2, 17, 15,
        2,  1, 20,  5,  6,  7,  8,  3, 15,  1,  7, 21, 15,  9,  8, 18,  6,
       21, 19,  5,  4, 11, 20, 14,  9, 21, 14,  0,  0, 21,  1, 18, 14,  0,
       14,  6, 20, 17,  6, 17,  3,  0, 19, 13, 20,  2, 12, 16,  8,  1, 13,
        5,  6, 12,  5,  4, 19,  6,  7,  2,  3,  8,  3, 17, 16,  6,  1,  2,
       15, 17,  0, 16, 19, 11, 18, 17, 12, 19, 17,  7, 20,  6,  8, 13, 10,
       13,  9,  1, 13,  0, 17, 21,  4,  3, 10,  9, 13,  7,  7, 16, 20,  2,
        1,  6,  6, 13, 20, 20,  4, 13,  6,  5, 17,  5, 14, 10, 16, 19,  3,
       10,  6, 12, 16,  5, 20, 17, 17,  4, 20,  6, 13,  4, 20,  7,  0,  1,
        4,  1, 11, 12, 17, 17, 20,  8, 15,  6, 10,  9,  2,  5, 20, 16,  4,
        1,  2,  0, 11, 12,  3,  4, 15,  5, 19, 16,  7, 17,  3,  8, 21, 16,
        9, 16, 16, 10, 11, 12,  9, 19,  4, 13, 11, 10, 14, 20,  9, 16, 10,
        5, 14, 15,  4,  7,  4, 19, 18,  4, 10, 17,  1,  3, 13, 17, 16, 10,
       19,  2, 20, 16, 20,  3,  2, 18,  5,  3,  7,  4,  3,  7,  5, 19, 19,
        1,  3,  2, 18, 13,  0, 19,  0, 13,  0, 21, 18])
```

```
#Evaluating the model
train_accuracy=LG.score(xtrain,ytrain)
print('Train_accuracy(R_Squared):',train_accuracy)
test_accuracy=LG.score(xtest,ytest)
print('Test_accuracy(R_Squared):',test_accuracy)
```

```
Train_accuracy(R_Squared): 0.9721845318860244
Test_accuracy(R_Squared): 0.9435261707988981
```

```
import math
from sklearn.metrics import mean_squared_error,mean_absolute_error
```

```
print('Mean Absolute Error:',mean_absolute_error(ytest,y_pred))
print('Mean Squared Error:',mean_squared_error(ytest,y_pred))
print('Root Mean Squared Error:',math.sqrt(mean_squared_error(ytest,y_pred)))
```

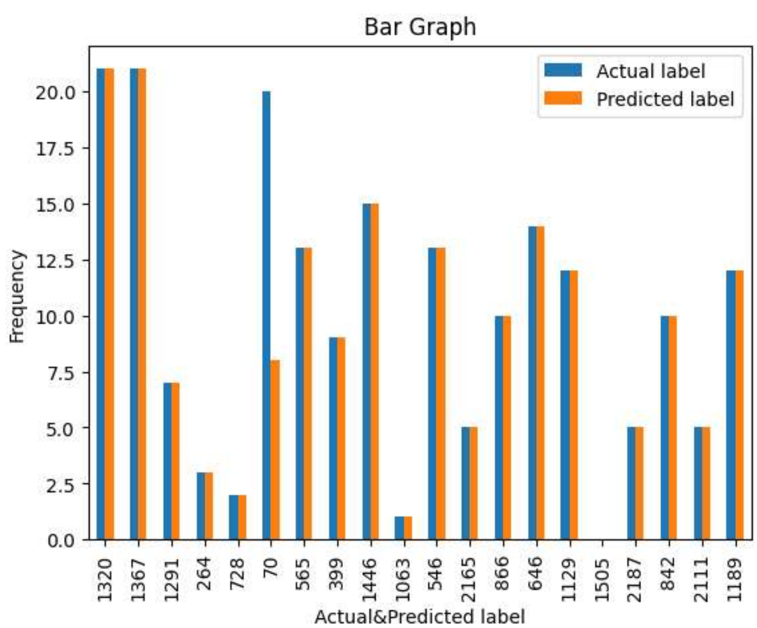
```
Mean Absolute Error: 0.47107438016528924
Mean Squared Error: 4.597796143250688
Root Mean Squared Error: 2.1442472206466046
```

```
#To check the actual label,predicted label
dfr=pd.DataFrame({'Actual label':ytest,'Predicted label':y_pred})
dfr
```

| | Actual label | Predicted label |
|---|---|---|
| **1320** | 21 | 21 |
| **1367** | 21 | 21 |
| **1291** | 7 | 7 |
| **264** | 3 | 3 |
| **728** | 2 | 2 |
| **...** | ... | ... |
| **1523** | 0 | 0 |
| **731** | 2 | 13 |
| **1545** | 0 | 0 |
| **1358** | 21 | 21 |
| **383** | 9 | 18 |

726 rows × 2 columns

```
#Plotting the Bar graph for above actual,predicted label
graph=dfr.head(20)
graph.plot(kind='bar')
plt.title('Bar Graph')
plt.xlabel('Actual&Predicted label')
plt.ylabel('Frequency')
plt.show()
```




```
#RandomForestClassifier()
from sklearn.ensemble import RandomForestRegressor
RF = RandomForestRegressor()
RF.fit(xtrain,ytrain)
```

```
▾ RandomForestRegressor
RandomForestRegressor()
```

```
#Evaluating model
train_accuracy = RF.score(xtrain,ytrain)
print("Train_accuracy(R_Squared):",train_accuracy)
test_accuracy = RF.score(xtest,ytest)
print("Test_accuracy(R_Squared):",test_accuracy)
```

```
Train_accuracy(R_Squared): 0.994821238022106
Test_accuracy(R_Squared): 0.9715513770838728
```

```
#Comparision between Linear and RandomForestRegression using boxplot
logistic_regression_accuracy=0.9435261707988981
random_forest_accuracy=0.9715513770838728
accuracy_scores=[logistic_regression_accuracy,random_forest_accuracy]
model_names=['LogisticRegression','RandomForestRegression']
plt.bar(model_names,accuracy_scores)
plt.xlabel('Models')
plt.ylabel('Test_Accuracy')
plt.title('Comparision of Test_Accuarcy:LogisticRegression v/s RandomForestRegression')
plt.show()
```