Experiment 10

Aim: Construct a Crop Recommendation System based on the Weather and Soil Content Data using Random Forrest Method.

Concept to be Applied: Machine Learning is well equipped when it comes to analyzing data regarding soil conditions, including moisture level, temperature, and chemical makeup, all of which have an impact upon crop growth and livestock well-being.

Today in agriculture, this can allow crops to be grown at much higher precision, enabling farmers to treat plants and animals almost individually, which in turn significantly increases the effectiveness of farmers' decisions.

Using this can develop means to even predict harvest yields and evaluate crop quality for individual plant species to detect crop disease and weed infestations which were previously impossible.

Dataset Description: This dataset was build by augmenting datasets of rainfall, climate and fertilizer data available for India. Following is the description of the dataset.

- N - ratio of Nitrogen content in soil
- P - ratio of Phosphorous content in soil
- K - ratio of Potassium content in soil
- temperature - temperature in degree Celsius
- humidity - relative humidity in %
- ph - ph value of the soil
- rainfall - rainfall in mm

Procedure/Steps (With Python Code):

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

df=pd.read_csv('crop_recommendation.csv')
df.head()
```

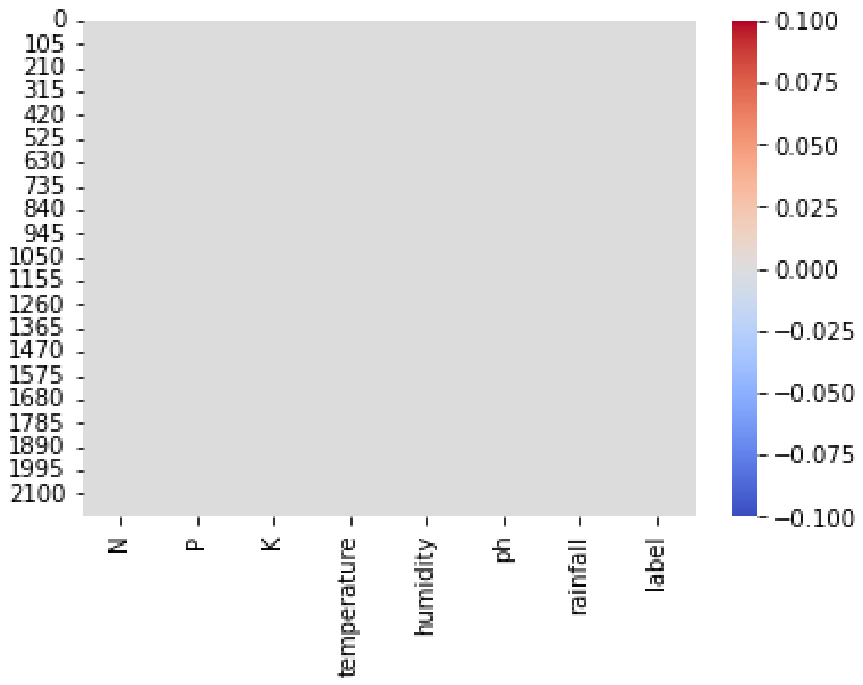
	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473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

df.describe()

	N	P	K	temperature	humidity	ph	rainfall
count	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000
mean	50.551818	53.362727	48.149091	25.616244	71.481779	6.469480	103.463655
std	36.917334	32.985883	50.647931	5.063749	22.263812	0.773938	54.958389
min	0.000000	5.000000	5.000000	8.825675	14.258040	3.504752	20.211267
25%	21.000000	28.000000	20.000000	22.769375	60.261953	5.971693	64.551686
50%	37.000000	51.000000	32.000000	25.598693	80.473146	6.425045	94.867624
75%	84.250000	68.000000	49.000000	28.561654	89.948771	6.923643	124.267508
max	140.000000	145.000000	205.000000	43.675493	99.981876	9.935091	298.560117

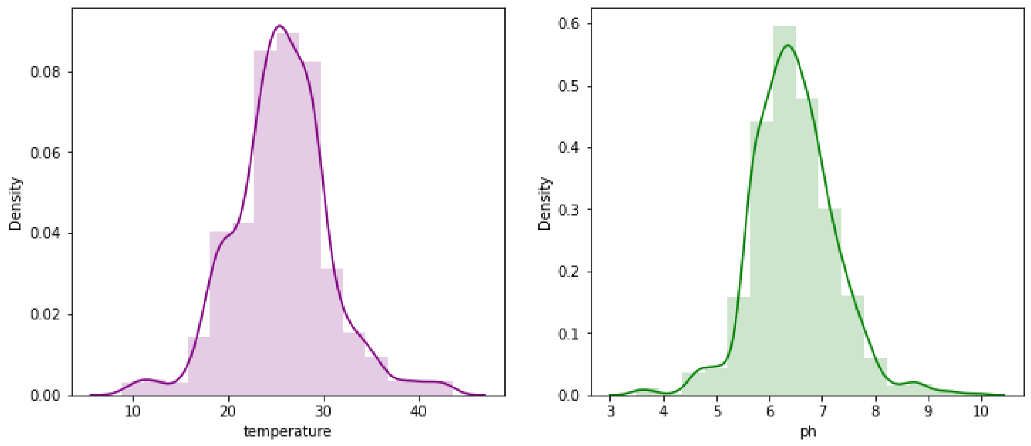
Heatmap to check null/missing values

```
sns.heatmap(df.isnull(),cmap="coolwarm")
plt.show()
```



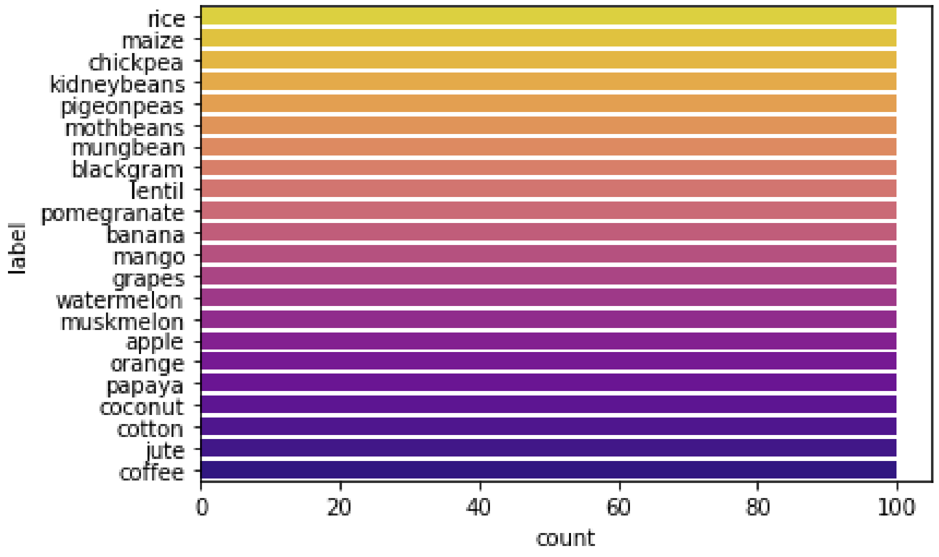
Let's have a closer look at the distribution of temperature and ph.
It is symmetrical and bell shaped, showing that trials will usually give a result near the average, but will occasionally deviate by large amounts. It's also fascinating how these two really resemble each other!

```
plt.figure(figsize=(12,5))
plt.subplot(1, 2, 1)
sns.distplot(df['temperature'],color="purple",bins=15,hist_kws={'alpha':0.2})
plt.subplot(1, 2, 2)
sns.distplot(df['ph'],color="green",bins=15,hist_kws={'alpha':0.2})
```



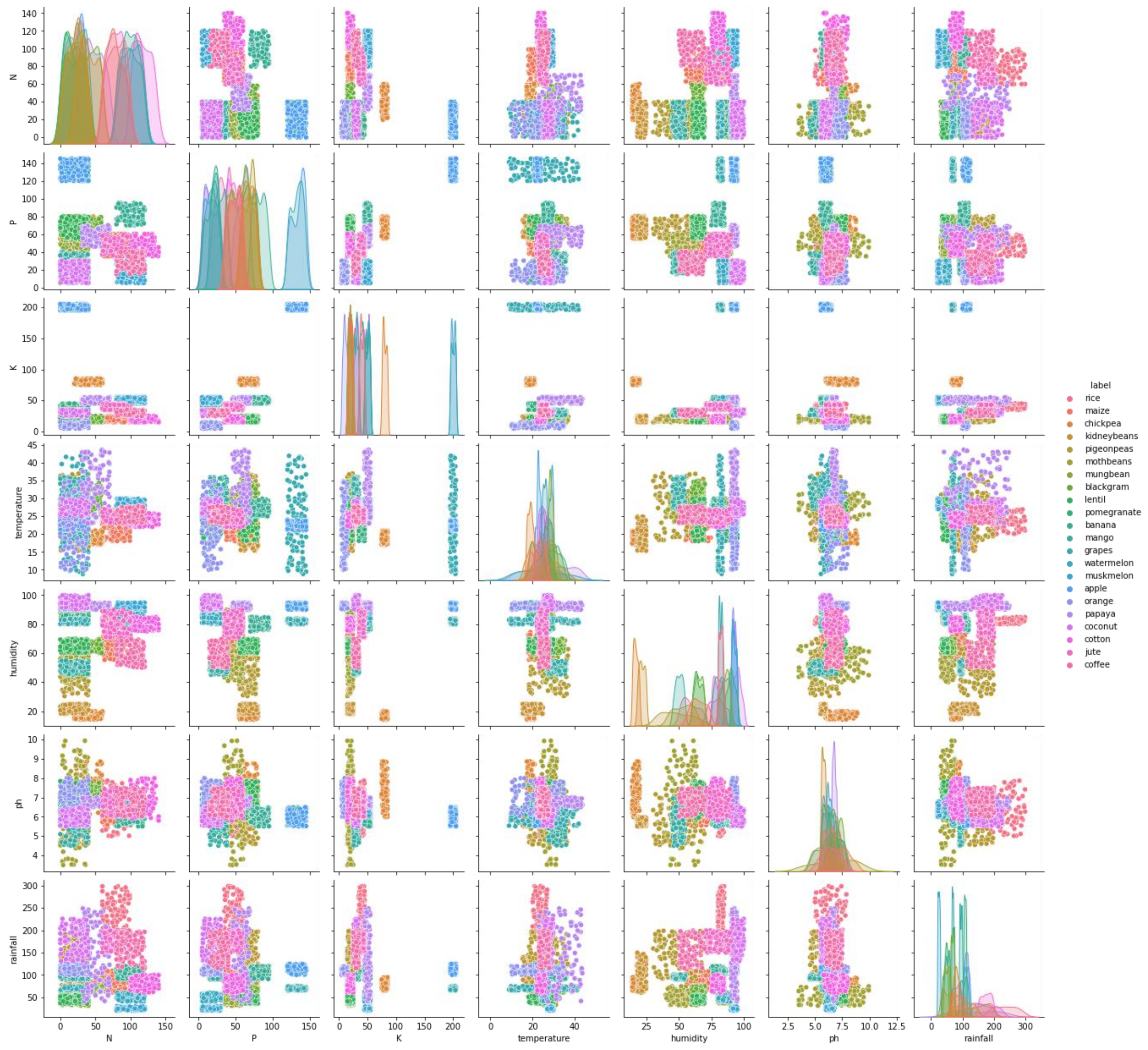
A quick check if the dataset is balanced or not.
If found imbalanced, we would have to downsample some targets which are more in quantity but so far everything looks good!

```
sns.countplot(y='label',data=df, palette="plasma_r")
```



A very important plot to visualize the diagonal distribution between two features for all the combinations! It is great to visualize how classes differ from each other in a particular space.

sns.pairplot(df, hue = 'label')

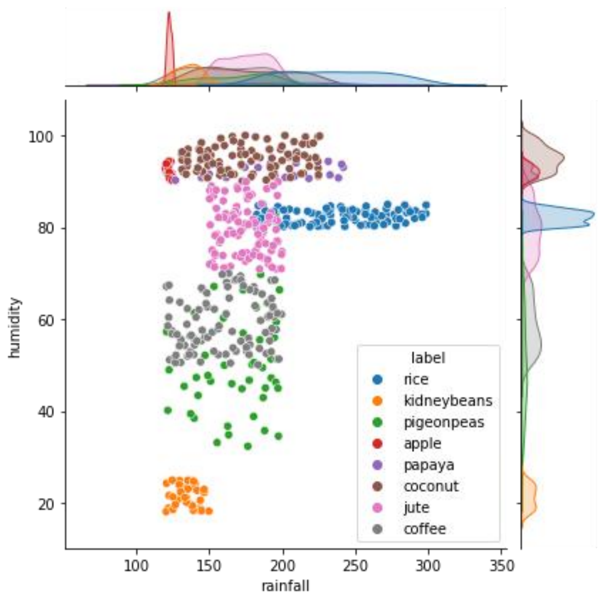


During rainy season, average rainfall is high (average 120 mm) and temperature is mildly chill (less than 30'C).

Rain affects soil moisture which affects ph of the soil. Here are the crops which are likely to be planted during this season.

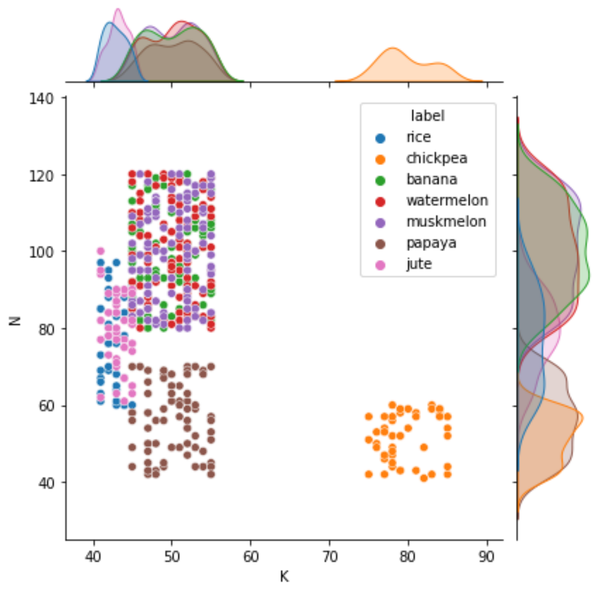
- **Rice needs heavy rainfall (>200 mm) and a humidity above 80%. No wonder major rice production in India comes from East Coasts which has average of 220 mm rainfall every year!**
- **Coconut is a tropical crop and needs high humidity therefore explaining massive exports from coastal areas around the country.**

```
sns.jointplot(x="rainfall",y="humidity",data=df[(df['temperature']<30) &
(df['rainfall']>120)],hue="label")
```



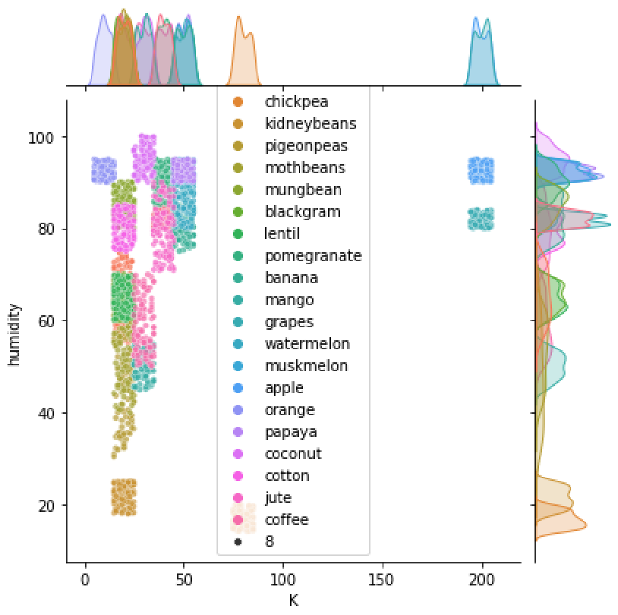
This graph correlates with average potassium (K) and average nitrogen (N) value (both>50).These soil ingredients direcly affects nutrition value of the food. Fruits which have high nutrients typically has consistent potassium values.

```
sns.jointplot(x="K",y="N",data=df[(df['N']>40)&(df['K']>40)],hue="label")
```



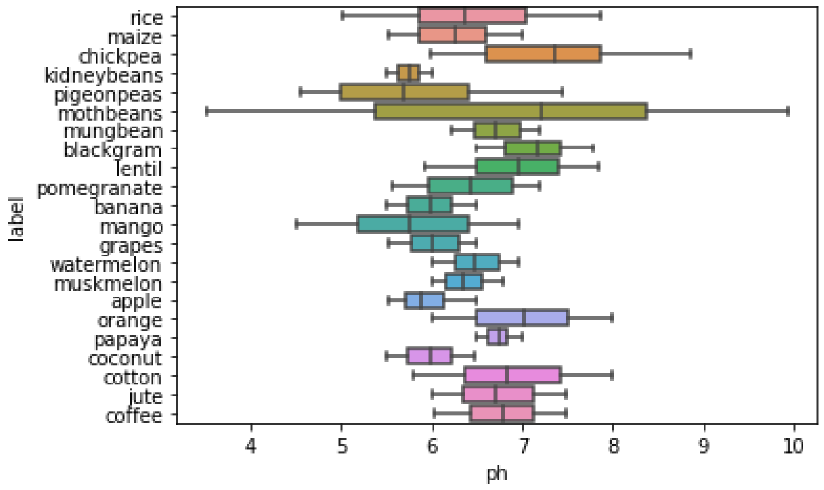
Let's try to plot a specfic case of pairplot between `humidity` and `K` (potassium levels in the soil.). sns.jointplot() can be used for bivariate analysis to plot between humidity and K levels based on Label type. It further generates frequency distribution of classes with respect to features.

sns.jointplot(x="K",y="humidity",data=df,hue='label',size=8,s=30,alpha=0.7)



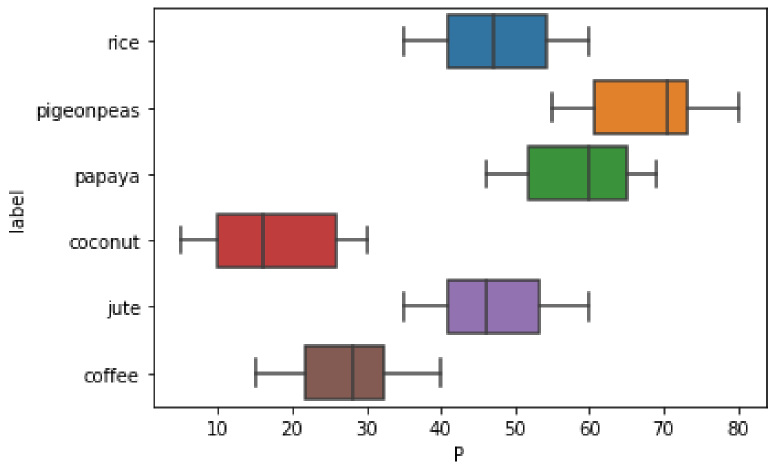
We can see ph values are critical when it comes to soil. A stability between 6 and 7 is preffered.

sns.boxplot(y='label',x='ph',data=df)



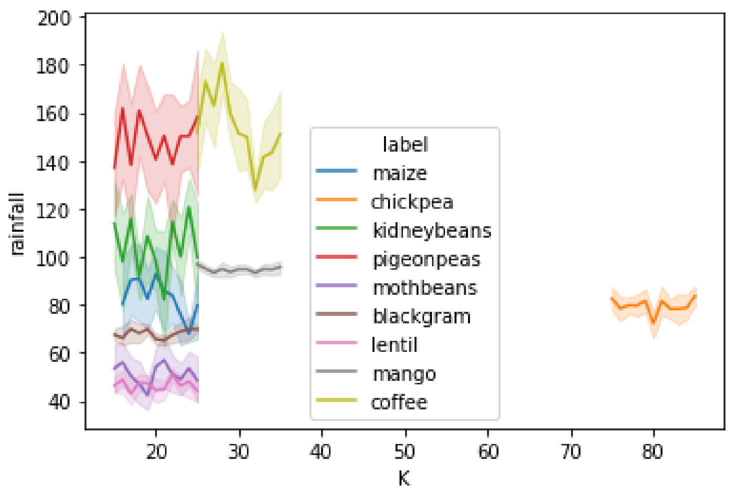
Another interesting analysis where Phosphorous levels are quite differentiable when it rains heavily (above 150 mm).

```
sns.boxplot(y='label',x='P',data=df[df['rainfall']>150])
```



Further analyzing phosphorous levels. When humidity is less than 65, almost same phosphor levels(approx 14 to 25) are required for 6 crops which could be grown just based on the amount of rain expected over the next few weeks.

```
sns.lineplot(data = df[(df['humidity']<65)], x = "K", y = "rainfall",hue="label")
```



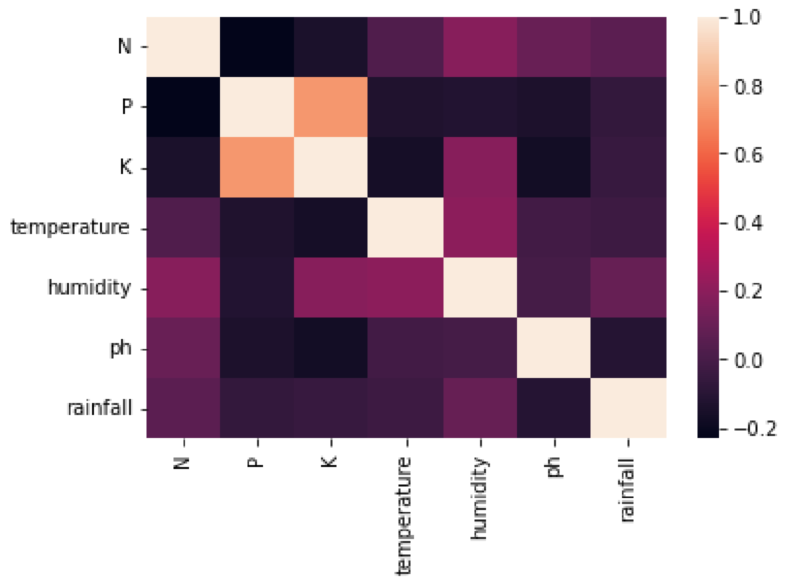
Let's make the data ready for machine learning model

```
c=df.label.astype('category')
targets = dict(enumerate(c.cat.categories))
df['target']=c.cat.codes
```

```
y=df.target
X=df[['N','P','K','temperature','humidity','ph','rainfall']]
```

<span style="color:red">**Correlation visualization between features. We can see how Phosphorous levels and Potassium levels are highly correlated.**</span>

```
sns.heatmap(X.corr())
```



Feature scaling is required before creating training data and feeding it to the model.

As we saw earlier, two of our features (temperature and ph) are gaussian distributed, therefore scaling them between 0 and 1 with MinMaxScaler.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler

X_train, X_test, y_train, y_test = train_test_split(X, y,random_state=1)

scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)

# we must apply the scaling to the test set as well that we are computing for the training set
X_test_scaled = scaler.transform(X_test)
```

Building Random Forrest Classifier and training the model: max depth and n_estimator are important to fine tune otherwise trees will be densely graphed which will be a classic case of overfitting. max_depth=4 and n_estimators=10 gives pretty much satisfying results by making sure model is able to generalize well.

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(max_depth=4,n_estimators=100,random_state=42).fit(X_train, y_train)

print('RF Accuracy on training set: {:.2f}'.format(clf.score(X_train, y_train)))
print('RF Accuracy on test set: {:.2f}'.format(clf.score(X_test, y_test)))
```

Input:

https://drive.google.com/file/d/1UG3wOI775QhfgsiUGu4ZekrwIATPuFy6/view?usp=drive_link

Use yellowbrick for classification report as they are great for visualizing in a tabular format

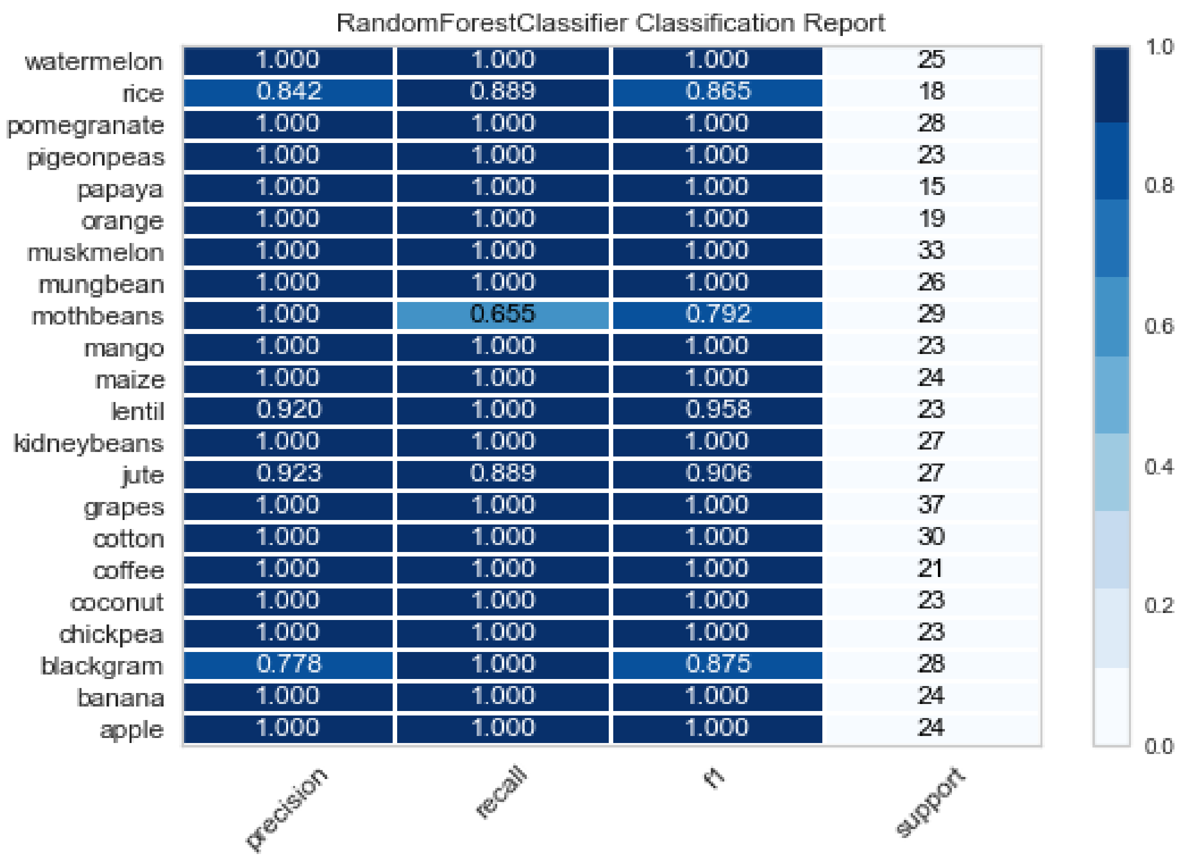
```
from yellowbrick.classifier import ClassificationReport
classes=list(targets.values())
visualizer = ClassificationReport(clf, classes=classes, support=True,cmap="Blues")

visualizer.fit(X_train, y_train)  # Fit the visualizer and the model
visualizer.score(X_test, y_test)  # Evaluate the model on the test data
visualizer.show()
```

Recommend crop for the input ***[19, 72, 15, 28.83600962, 69.76112921, 6.890760124, 44.08562546].***

Output:

Recommendation: **'Lentil'**.


RandomForestClassifier Classification Report

	precision	recall	f1	support
watermelon	1.000	1.000	1.000	25
rice	0.842	0.889	0.865	18
pomegranate	1.000	1.000	1.000	28
pigeonpeas	1.000	1.000	1.000	23
papaya	1.000	1.000	1.000	15
orange	1.000	1.000	1.000	19
muskmelon	1.000	1.000	1.000	33
mungbean	1.000	1.000	1.000	26
mothbeans	1.000	0.655	0.792	29
mango	1.000	1.000	1.000	23
maize	1.000	1.000	1.000	24
lentil	0.920	1.000	0.958	23
kidneybeans	1.000	1.000	1.000	27
jute	0.923	0.889	0.906	27
grapes	1.000	1.000	1.000	37
cotton	1.000	1.000	1.000	30
coffee	1.000	1.000	1.000	21
coconut	1.000	1.000	1.000	23
chickpea	1.000	1.000	1.000	23
blackgram	0.778	1.000	0.875	28
banana	1.000	1.000	1.000	24
apple	1.000	1.000	1.000	24