과목명	머신러닝프로그래밍				
평가 내용	2023년도 1학기 기말프로젝트 최종 보고서				
프로그램명	기후와 토양환경을 기반으로 한 작물 추천 프로그램				
학생 정보	팀장	학번	202004008	이름	김예은
	팀원	학번	202004037	이름	바리마무두
		학번	201904019	이름	유재현

1. 프로그램 필요성

현재 세계적으로 기후 변화, 인구 증가, 자원 부족 등의 문제로 인해 농업 분야에 있어 작물 선택은 매우 중요한 사항이다. 그렇기에 환경에 맞는 적절한 작물을 재배한다면 보다 나은 농업 생산량을 기대 할 수 있다.

머신러닝을 활용하여 각 작물이 잘 자라는 환경요인을 학습하고, 적절한 알고리즘을 사용하면 정밀하게 자신의 환경에 알 맞는 작물을 추천 받을 수 있다.

더 나아가 어떤 작물을 키우기 위해 요구되는 환경 조건이나 필요한 토양 영양 성분을 파악 할 수 있어 자신이 원하는 작물의 최적의 환경을 조성하는데 도움이 될 수 있다.

2. 프로그램 소개

제목 : 기후와 토양환경을 기반으로 한 작물 추천 프로그램

■ 목적 : 농업 생산량의 증진을 위해 각 환경요인에 맞는 적절한 작물을 추천함으로서 더 나은 선택을 할 수 있도록 도움을 준다.

■ 요약 : 알고리즘을 활용하여 종류별로 작물이 자라는 환경 데이터를 train 데이터와 test 데이터로 나눠 학습한다.

이후 학습된 모델을 바탕으로 작물 추천 프로그램을 실행하여 환경 요소 값을 입력하면 해당 환경에 맞는 적절한 작물이 추천된다.

■ 기대효과 :

1) 작물 선택에 대한 더 나은 정보로 인한 생산성, 수익성 향상

2) 농작물의 수요와 공급을 균형 있게 조절로 인한 농작물 시장의 안정성 향상

3) 적합한 작물을 선택함으로써 토양의 영양 소모를 최소화하고 작물의 생장에 필요한 자원을 효율적으로 활용

3. 데이터 분석

(1) 출처 : kaggle

(https://www.kaggle.com/code/theeyeschico/crop-analysis-and-prediction/notebook),

(2) 데이터 요약

-각 요소들은 식물이 자라는데 가장 많은 영향을 끼치는 요소들이다. 식물이 자라는데 필요한 토양 영양 성분인 K(칼륨), N(질소), P(인)은 서로 상호작용하여 식물의 생장과 발달에 영향을 미친다. 또한 강수량, 습도, 온도도 중요한 요인으로써 농업의 생산량에 중요한 영향을 끼친다.

인덱스	이름	데이터 타입	값	설명
0	N	연속형	정수형	질소(토양)
1	P	연속형	정수형	인(토양)
2	K	연속형	정수형	칼륨(토양)
3	temperature	연속형	실수형	평균 온도 (한국)
4	humidity	연속형	실수형	습도 (한국)
5	ph	연속형	실수형	산성도
6	rainfall	연속형	실수형	강수량 (한국)
7	label	범주형	문자형	작물명

4. 사용 알고리즘

- 이름: KNN, SVM, Decision Tree Classifier & Random Forest Classifier
- 출처:

https://www.kaggle.com/code/theeyeschico/crop-analysis-and-prediction/notebook

- 설명

1) knn 알고리즘 / 지도학습 (분류)

: 새로운 데이터 포인트에 대해 가장 가까운 이웃들(K개의 이웃)을 찾아 그들의 레이블이나 값을 기반으로 예측을 수행하는 알고리즘

2) 의사결정트리(Decision Trees) / 지도학습 (분류) & Random Forest (비지도 학습)

: 여러 개의 의사 결정 트리를 독립적으로 학습하고, 각각의 트리가 독립적으로 예측한 결과를 종합하여 최종 예측을 수행 하는 알고리즘

3. SVM(Support Vector Machine) / 지도학습 (분류) (프로그램 구현에 사용하진 않음)

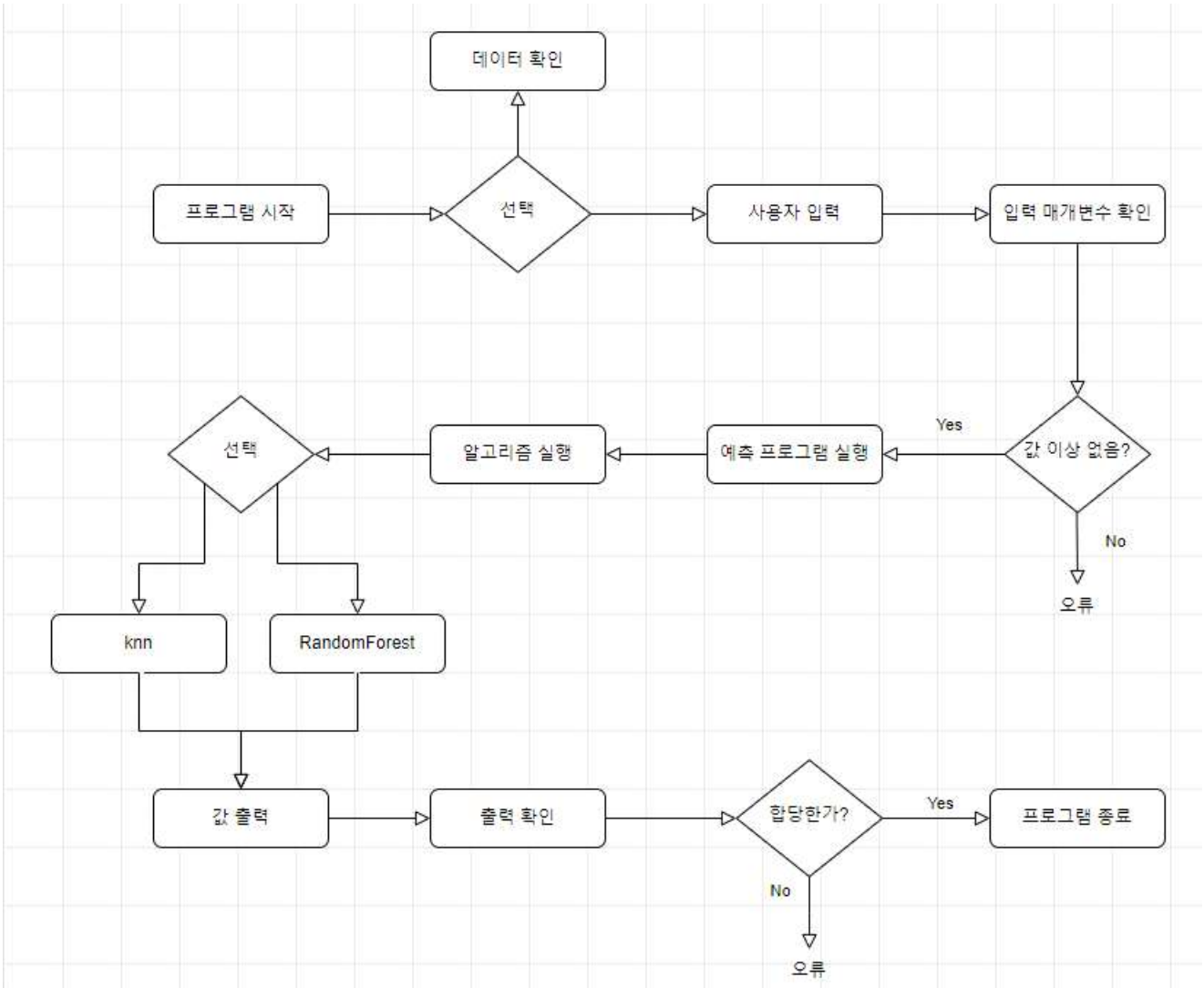
: 데이터를 분류하기 위해 결정 경계(Decision Boundary)를 찾는 기법으로, 클래스 간의 마진(Margin)을 최대화하는 초평면(Hyperplane)을 찾음. 이를 통해 데이터를 선형적으로 분리하거나 비선형 매핑을 통해 분류할 수 있음.

1) 선형, RBF, 다항 커널을 갖는 모델을 각각 학습하고 SVM의 정확도를 출력 - 각 커널 유형이 테스트 데이터에 대해 어떤 성능을 보이는지 확인 가능

2) 그리드 서치(Grid Search)를 사용하여 SVM(Support Vector Machine) 모델의 최적 매개변수를 탐색하는 작업 수행

3) 그리드 서치를 통해 얻은 최상의 평균 교차 검증 점수와 해당 매개 변수 조합을 출력 - 최적의 매개 변수로 설정된 SVM 모델의 성능과 설정 확인 가능

6. 순서도 및 구현



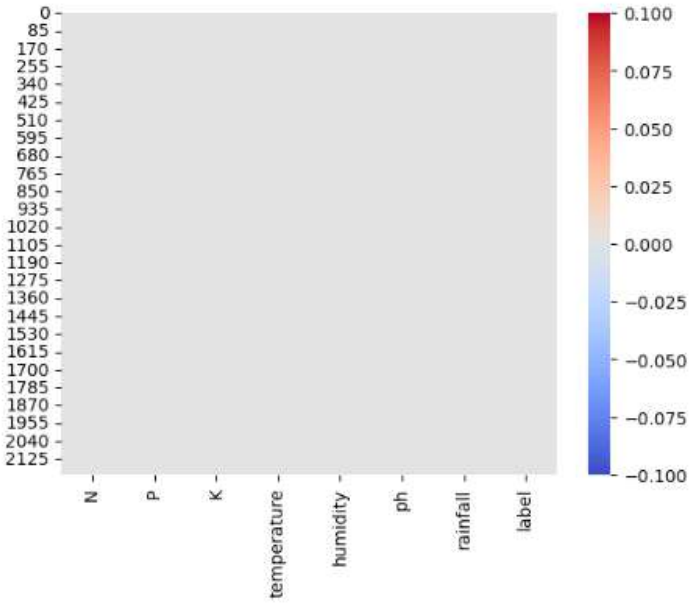
1) 결측값 확인

Seaborn 라이브러리의 sns.heatmap() 함수는 히트맵을 사용하여 DataFrame 내의 결측값을 시각화하는 데 사용된다. 이 그래프에서 짙은 회색은 결측값이 0인 것을 의미한다. 결측값이 0에 가까울수록 색상은 밝은 회색으로 변한다. (결측값 없음)

Exploratory Data Analysis

Heatmap to check null/missing values

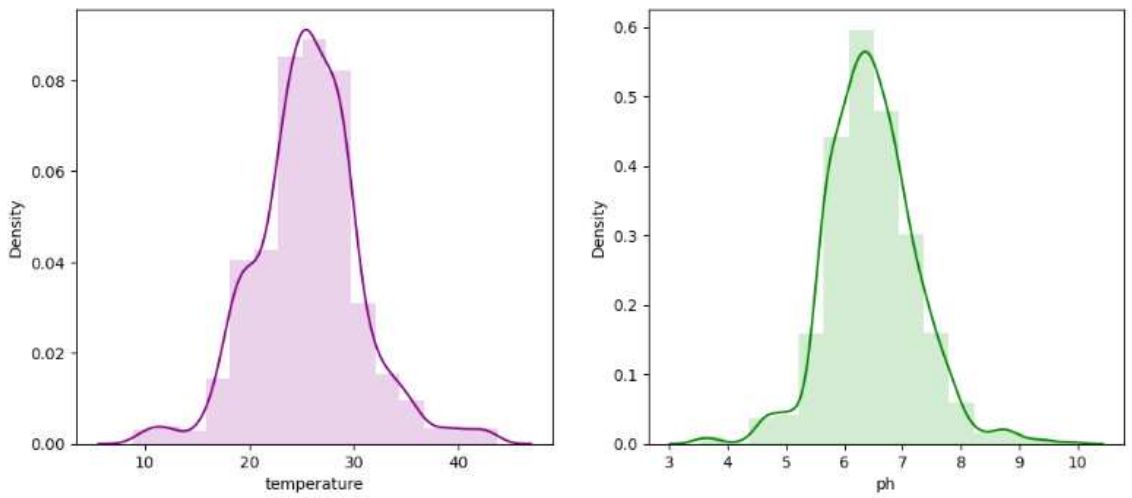
```
In [5]: sns.heatmap(df.isnull(),cmap="coolwarm")
        plt.show()
```



2) 데이터프레임 변수 분포 시각화

이 그래프는 "온도"와 "pH" 변수의 분포 특성을 시각화하고 데이터의 중심 경향과 분산에 대한 정보를 제공한다. (정규분포 모양 = 균일하다)

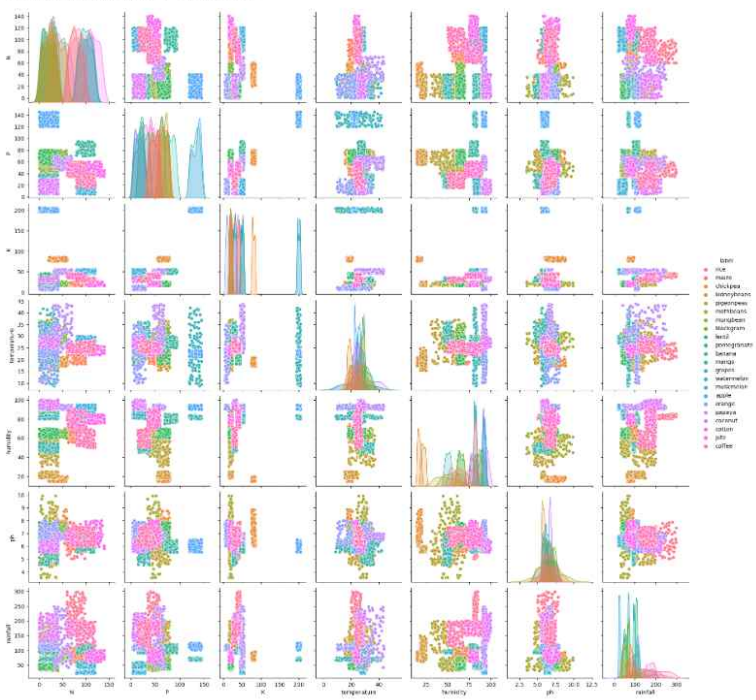
```
In [6]: plt.figure(figsize=(12,5))
        plt.subplot(1, 2, 1)
        # sns.distplot(df_setosa['sepal_length'], kde=True, color='green', bins=20, hist_kws={'alpha':0.3})
        sns.distplot(df['temperature'],color="purple",bins=15,hist_kws={'alpha':0.2})
        plt.subplot(1, 2, 2)
        sns.distplot(df['ph'],color="green",bins=15,hist_kws={'alpha':0.2})
Out [6]: <Axes: xlabel='ph', ylabel='Density'>
```



3) 데이터 시각화

데이터를 시각화하여 데이터의 분포나 각 요소의 상관관계를 시각적으로 알 수 있다.

```
In [8]: sns.pairplot(df, hue = 'label')
Out[8]: <seaborn.axisgrid.PairGrid at 0x208b13394c0>
```



4) 각 열들의 상관관계 정도

데이터 프레임에서 각 열들 간의 상관관계를 계산하여 연관성을 지표로 나타낸다. 이를 바탕으로 계산된 상관 계수를 히트맵으로 시각화한다.

DATA PRE-PROCESSING

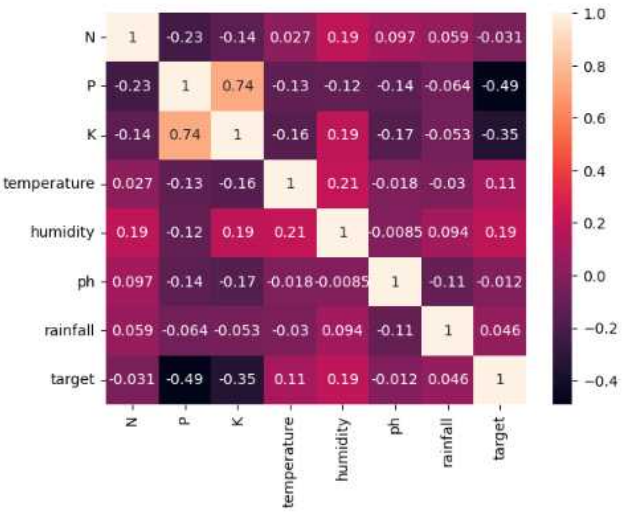
Let's make the data ready for machine learning model

```
In [15]: 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']]
```

Correlation visualization between features. We can see how Phosphorous levels and Potassium levels are highly correlated.

```
In [16]: sns.heatmap(df.corr(),annot=True)
Out[16]: <Axes: >
```



5) feature scaling

알고리즘을 사용하기 이전에 데이터를 train 데이터와 test 데이터를 7:3 비율로 나누고, KNN 알고리즘을 사용하기 위해 train 데이터와 test 데이터를 0~1 사이로 스케일링 해준다.

FEATURE SCALING

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.

In [17]:
```
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,test_size = 0.3,random_state =2)

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)
```

6) KNN 알고리즘 실행 (지도학습)

X_train_scaled, y_train 70% 와 X_test_scaled, y_test 30%를 나눈 것을 기반으로 모델 학습을 진행하고, 그 결과로 정확도를 출력한다.

MODEL SELECTION

KNN Classifier for Crop prediction.

In [19]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_train_scaled, y_train)
knn.score(X_test_scaled, y_test)
```

Out[19]:
```
0.9803030303030303
```

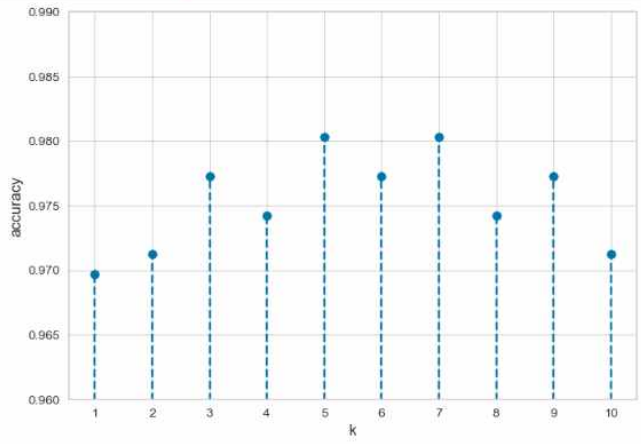
7) KNN 알고리즘의 이웃 값에 따른 정확도 시각화

KNN 알고리즘은 이웃값(K)을 설정하여 인접한 K개의 데이터를 기준으로 분류를 진행할 수 있다.

In [22]:
```
k_range = range(1,11)
scores = []

for k in k_range:
    knn = KNeighborsClassifier(n_neighbors = k)
    knn.fit(X_train_scaled, y_train)
    scores.append(knn.score(X_test_scaled, y_test))

plt.xlabel('k')
plt.ylabel('accuracy')
plt.scatter(k_range, scores)
plt.vlines(k_range,0, scores, linestyle="dashed")
plt.ylim(0.96,0.99)
plt.xticks([i for i in range(1,11)]);
```



8) 의사결정트리(Decision Trees) (지도학습) & Random Forest (비지도 학습)

(1) 의사 결정 트리를 이용하여 모델 학습에 대한 정확도 출력

(2) 의사 결정 트리를 이용하여 특성 중요도 시각화

Classifying using decision tree

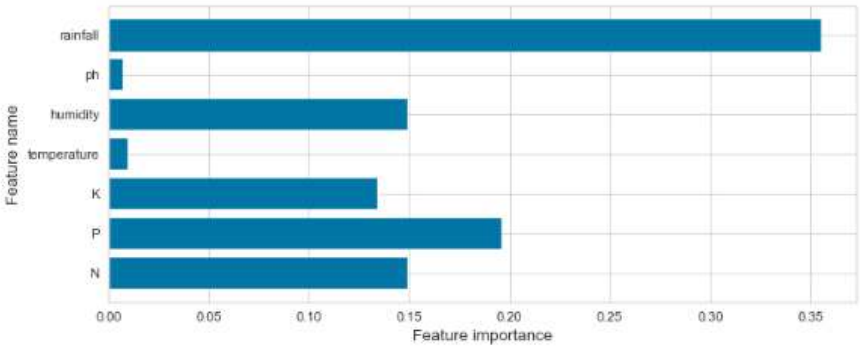
In [26]:
```
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier(random_state=42).fit(X_train, y_train)
clf.score(X_test,y_test)
```

Out [26]: 0.9833333333333333

Let's visualize the import features which are taken into consideration by decision trees.

In [27]:
```
plt.figure(figsize=(10,4), dpi=80)
c_features = len(X_train.columns)
plt.barh(range(c_features), clf.feature_importances_)
plt.xlabel("Feature importance")
plt.ylabel("Feature name")
plt.yticks(np.arange(c_features), X_train.columns)
plt.show()
```



(3) 의사 결정 트리를 기반으로 랜덤 포레스트 실행

- 과적합 방지를 위해 의사결정트리의 최대 깊이를 4로 제한하고 (max_depth=4), 100개의 의사 결정 트리를 사용 (n_estimators=100)
- 랜덤 포레스트를 이용하여 모델 학습을 진행
- 훈련 데이터의 예측 정확도 및 테스트 데이터의 정확도 출력

Classification using Random Forest.

In [46]:
```
'''
max depth and n_estimator are important to fine tune otherwise trees will be densely graphed which will be a classic case of overf
'''

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)))
```

```
RF Accuracy on training set: 0.95
RF Accuracy on test set: 0.94
```

9) 구현

data = np.array에 각 열에 해당하는 값(칼륨, 인, 질소, 온도, 습도, 산성도, 강수량)을 입력한다. 프로그램을 실행하면 prediction = (알고리즘 이름).predict(data)에서 모델 학습을 마친 알고리즘이 적절한 작물을 추천 해준다.

알고리즘은 KNN과 RandomForest 두 가지로 확인 가능하다.

crop list

In [31]:
```
targets
```

Out [31]:
```
{0: 'apple',
 1: 'banana',
 2: 'blackgram',
 3: 'chickpea',
 4: 'coconut',
 5: 'coffee',
 6: 'cotton',
 7: 'grapes',
 8: 'jute',
 9: 'kidneybeans',
 10: 'lentil',
 11: 'maize',
 12: 'mango',
 13: 'mothbeans',
 14: 'mungbean',
 15: 'muskmelon',
 16: 'orange',
 17: 'papaya',
 18: 'pigeonpeas',
 19: 'pomegranate',
 20: 'rice',
 21: 'watermelon'}
```

CROP PREDICTION

**K / P / N / Temperature / humidity / pH / Rainfall **

knn

In [60]:
```
data = np.array([[100, 40, 37, 25, 40, 6.9, 5]])
prediction = knn.predict(data) # knn알고리즘 적용
value = prediction[0]
targets[value]
```

Out [60]:
```
'banana'
```

Random Forest Classifier

In [43]:
```
data = np.array([[75, 41, 35, 24.97042599, 78.62697699, 6.856833064, 166.64152]])
prediction = clf.predict(data) # Random Forest 알고리즘 적용
value = prediction[0]
targets[value]
```

Out [43]:
```
'rice'
```

In [ ]:
```

```

7. 성능 평가 : 오차 행렬

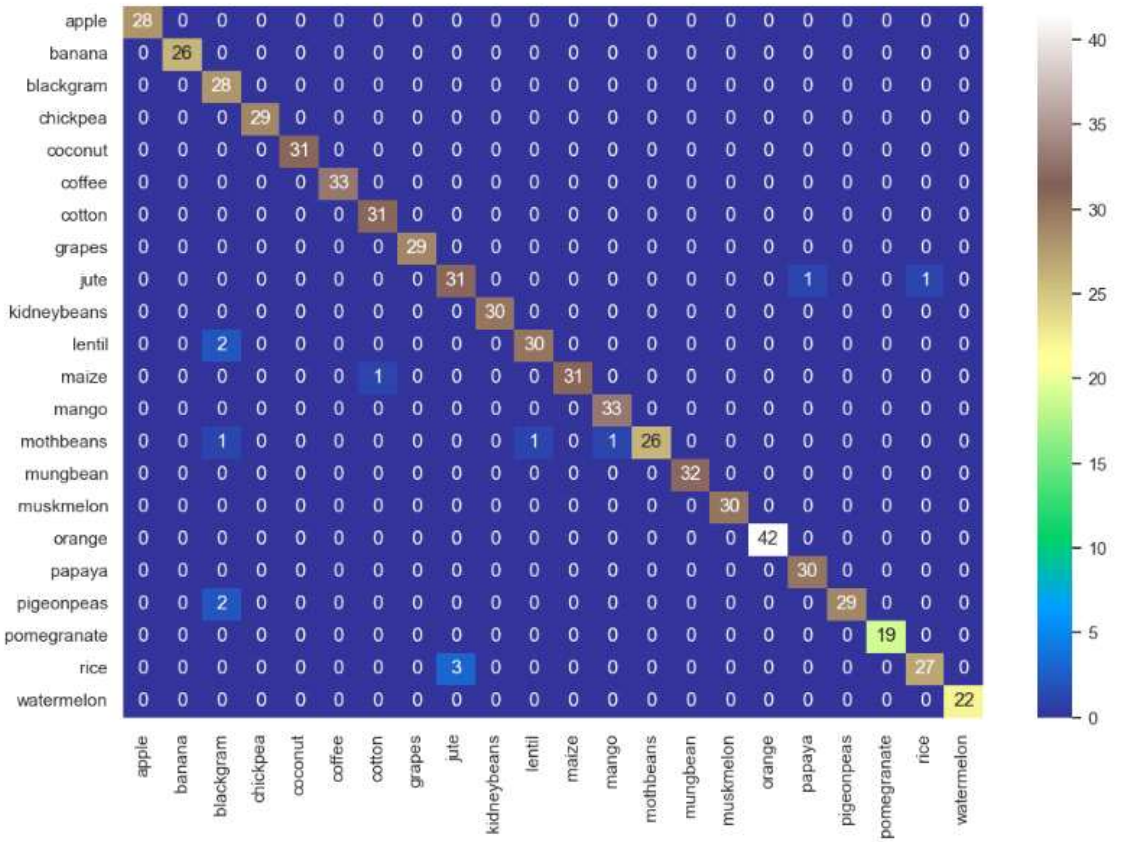
함수를 사용하여 분류 모델의 예측 결과와 실제 값 사이의 혼동 행렬 계산하고, 히트맵으로 이를 시각화 하여 나타낸다.

1) KNN 알고리즘

Confusion Matrix

In [20]:
```
from sklearn.metrics import confusion_matrix
mat=confusion_matrix(y_test,knn.predict(X_test_scaled))
df_cm = pd.DataFrame(mat, list(targets.values()), list(targets.values()))
sns.set(font_scale=1.0) # for label size
plt.figure(figsize = (12,8))
sns.heatmap(df_cm, annot=True, annot_kws={"size": 12},cmap="terrain")
```

Out [20]: <Axes: >

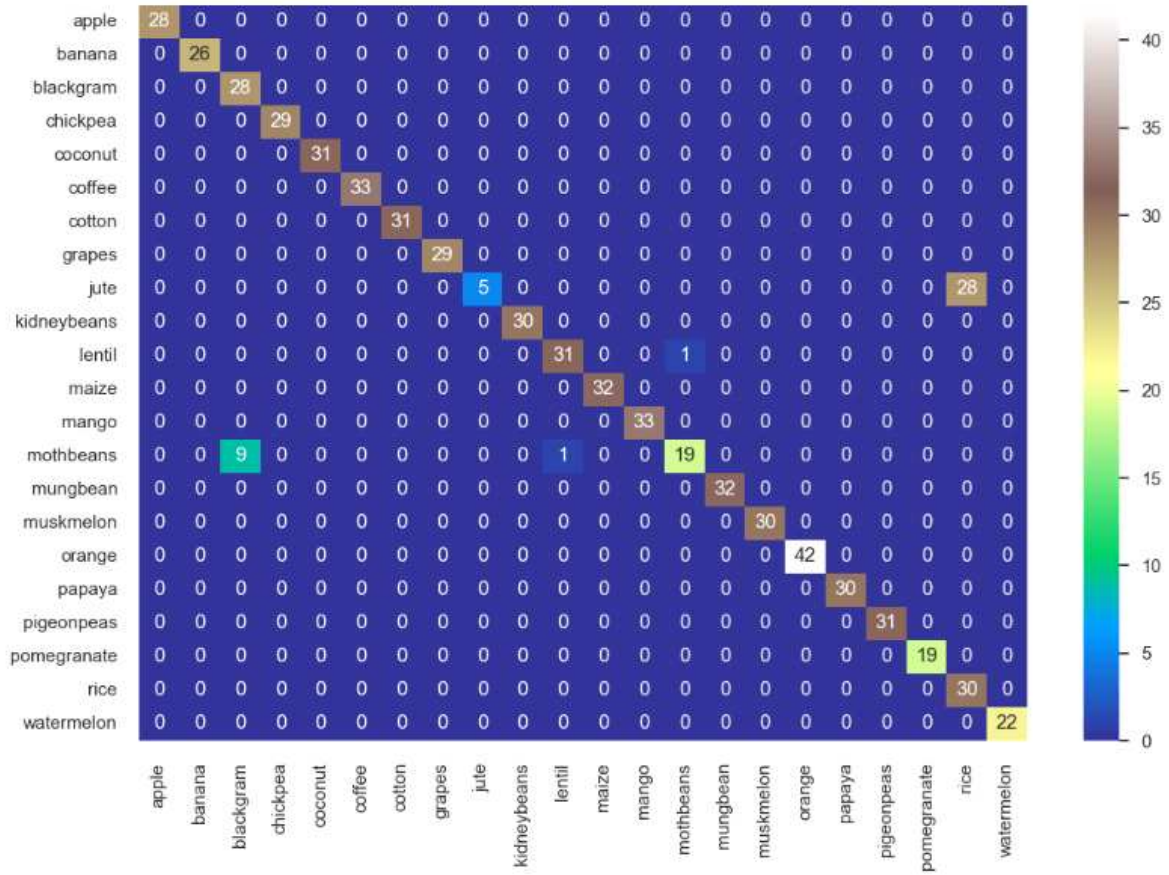


2) Randome Forest

RF Accuracy on test set: 0.994

In [47]:
```
from sklearn.metrics import confusion_matrix
mat=confusion_matrix(y_test,clf.predict(X_test))
df_cm = pd.DataFrame(mat, list(targets.values()), list(targets.values()))
sns.set(font_scale=1.0) # for label size
plt.figure(figsize = (12,8))
sns.heatmap(df_cm, annot=True, annot_kws={"size": 12},cmap="terrain")
```

Out[47]: <Axes: >



8. 성능 평가 : F1 스코어(F1 score)

yellowbrick 라이브러리의 ClassificationReport를 사용하여 분류 모델의 성능 보고서를 시각화한다. 각 작물에 대한 정확도, 민감도, 재현율, F1 스코어 시각화, support (훈련 데이터에서의 샘플 수)를 확인 할 수 있다.

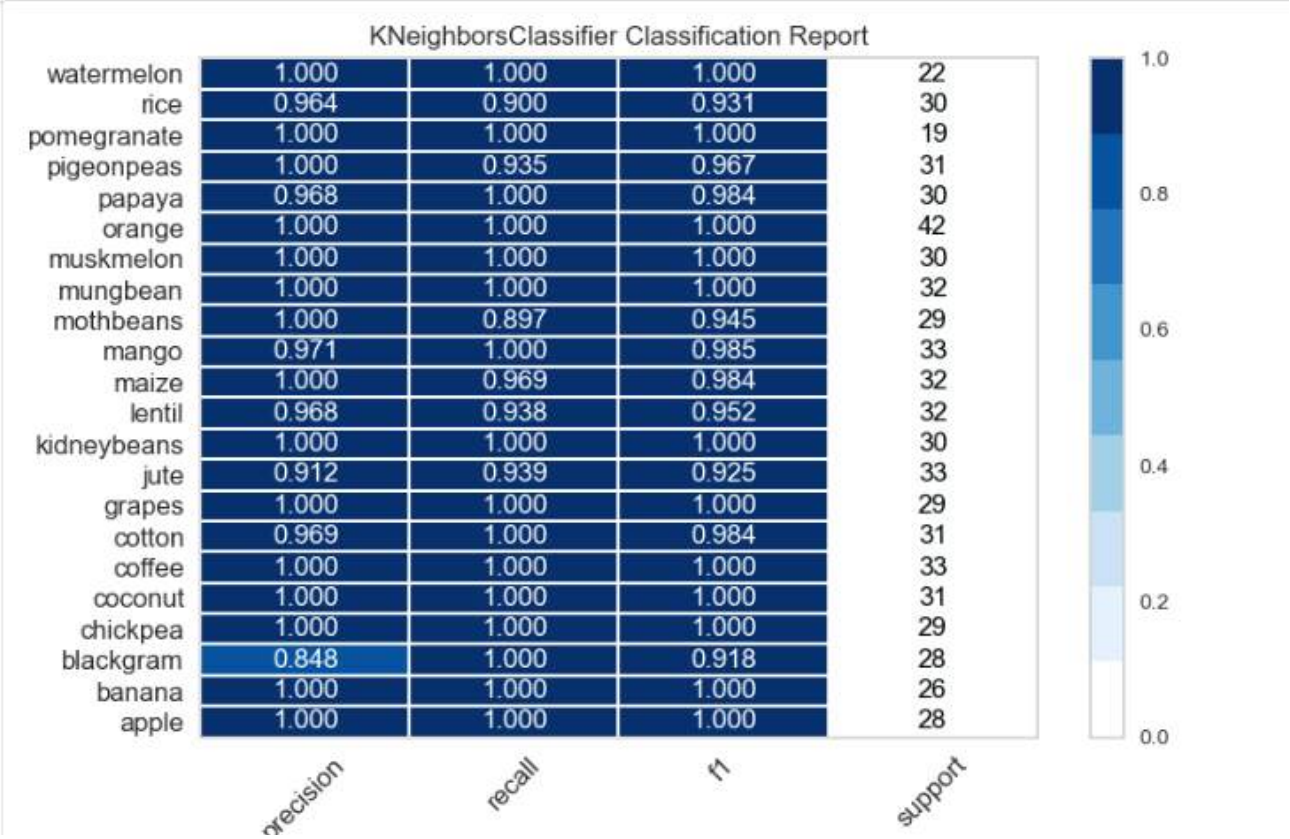
1) KNN

Classification report of KNN

Let's use yellowbrick for classification report as they are great for visualizing in a tabular format

```
In [21]: from yellowbrick.classifier import ClassificationReport
classes=list(targets.values())
visualizer = ClassificationReport(knn, classes=classes, support=True,cmap="Blues")

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



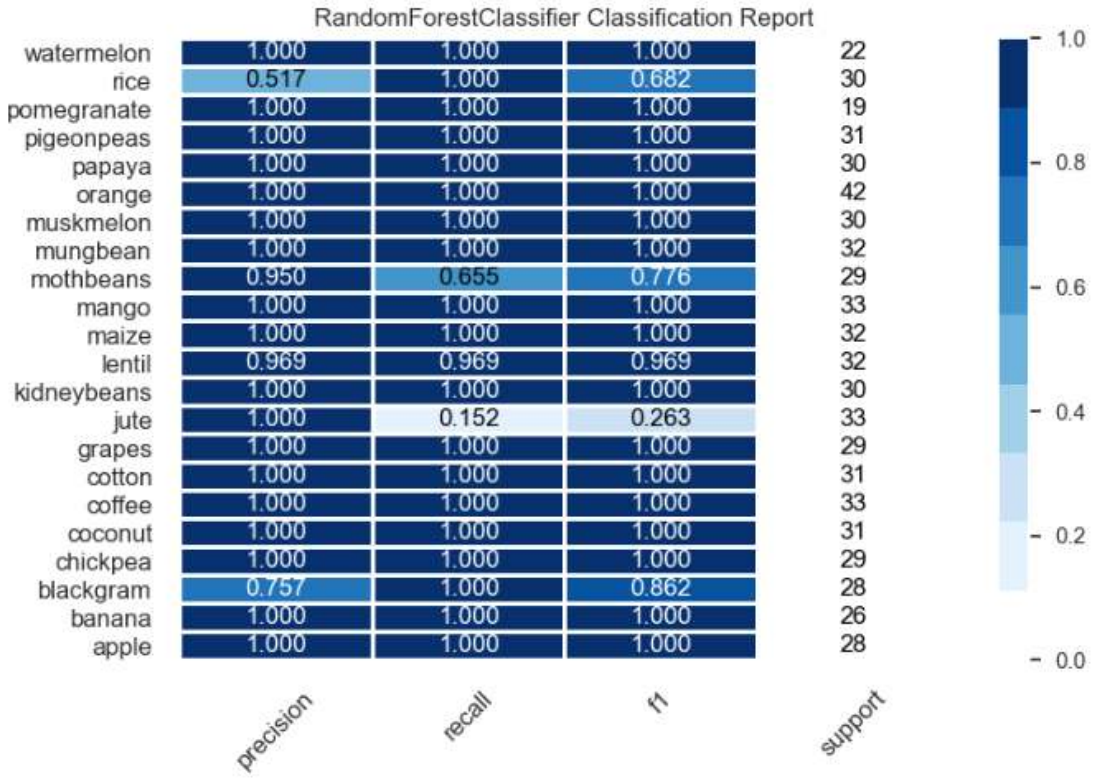
2) Random Forest

Classification report of Random Forest

Let's use yellowbrick for classification report as they are great for visualizing in a tabular format

```
In [30]: 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()
```



9. 결론

KNN, Random Forest 이 두 가지 알고리즘을 활용하여 작물 추천 프로그램을 구현하였다. 모두 정확도 0.98 이상의 성능을 보여주어 환경 요인에 대한 적절한 작물을 정확하게 추천해 주고 있다.

알고리즘으로 모델 학습을 진행하기에 앞서 적절한 스케일링과 선행 알고리즘이 요구되었다.

KNN 알고리즘에서 스케일링은 데이터 간의 거리를 정확하게 계산해야 하기 때문에 꼭 필요한 과정이다. 스케일이 크게 다른 특성이 있을 경우에 거리 계산에서 큰 스케일을 가진 특성이 다른 특성에 비해 더 큰 영향을 미칠 수 있어 예측에 편향이 발생할 수 있다. 그렇기에 train 데이터와 test 데이터를 표준화 하는 작업이 필요했다.

Random Forest 알고리즘의 경우에는 의사결정트리 알고리즘의 과대적합을 보완하고자 하여 나온 알고리즘이기 때문에 의사결정트리 알고리즘이 먼저 선행되어야 했다.

이러한 어려운 점이 있었지만 파이썬에서 제공하는 라이브러리와 함수가 있어서 어렵지 않게 구현할 수 있었다.

아쉬운 점은 Random Forest 알고리즘의 경우에 오차행렬을 보면 알 수 있듯이 작물 “jute”를 잘 구별해 내지 못하고 “rice”로 잘못 추천하는 경우가 있다. 이 부분에 대해서 좀 더 보완이 되어야 할 것 같다.