In [2]:
```
#importing libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
#importing the dataset
data = pd.read_csv('f2.csv')
data.head()
```

Out[3]:

	Temparature	Humidity	Moisture	Soil_Type	Crop_Type	Nitrogen	Potassium	Phosphorous
0	20	83	26	Clayey	rice	90	49	36
1	25	84	32	Loamy	rice	66	59	36
2	33	64	50	Loamy	Wheat	41	0	0
3	34	65	54	Loamy	Wheat	38	0	0
4	38	72	51	Loamy	Wheat	39	0	0

In [4]:
```
# Define the number of augmented samples to create for each original sampl
# increase dataset from 100 rows to 500


# num_augmented_samples = 3

# augmented_data = []

# for index, row in dataset.iterrows():
#     original_sample = row.to_dict()
#     augmented_samples = []

#     for _ in range(num_augmented_samples):
#         augmented_sample = original_sample.copy()
#         augmented_sample['Temparature'] += np.random.normal(0, 0.1)
#         augmented_sample['Humidity'] += np.random.normal(0, 0.1)
#         augmented_sample['Moisture'] += np.random.normal(0, 0.1)
#         augmented_sample['Nitrogen'] += np.random.normal(0, 0.1)
#         augmented_sample['Potassium'] += np.random.normal(0, 0.1)
#         augmented_sample['Phosphorous'] += np.random.normal(0, 0.1)
#         # Add more transformations for other columns as needed
#         augmented_samples.append(augmented_sample)


#     augmented_data.extend(augmented_samples)

# # Convert the augmented data to a DataFrame
# augmented_dataset = pd.DataFrame(augmented_data)

# # Concatenate the original dataset
# data = pd.concat([dataset, augmented_dataset], ignore_index=True)
```

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 552 entries, 0 to 551
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Temparature  552 non-null    int64 
 1   Humidity     552 non-null    int64 
 2   Moisture     552 non-null    int64 
 3   Soil_Type    552 non-null    object
 4   Crop_Type    552 non-null    object
 5   Nitrogen     552 non-null    int64 
 6   Potassium    552 non-null    int64 
 7   Phosphorous  552 non-null    int64 
 8   Fertilizer   552 non-null    object
dtypes: int64(6), object(3)
memory usage: 38.9+ KB
```

In [6]:
```
#changing the column names
data.rename(columns={'Humidity ':'Humidity','Soil Type':'Soil_Type','Crop
```

In [7]:
```
#checking unique values
data.nunique()
```

Out[7]:
```
Temparature    21
Humidity       34
Moisture       41
Soil_Type       5
Crop_Type      17
Nitrogen       47
Potassium      31
Phosphorous    36
Fertilizer     14
dtype: int64
```

In [8]:
```
#checking for null values
data.isna().sum()
```

Out[8]:
```
Temparature    0
Humidity       0
Moisture       0
Soil_Type      0
Crop_Type      0
Nitrogen       0
Potassium      0
Phosphorous    0
Fertilizer     0
dtype: int64
```

In [9]:
```
data['Fertilizer'].unique()
```

Out[9]:
```
array(['Urea', 'TSP', 'Superphosphate', 'Potassium sulfate.',
       'Potassium chloride', 'DAP', '28-28', '20-20', '17-17-17',
       '15-15-15', '14-35-14', '14-14-14', '10-26-26', '10-10-10'],
      dtype=object)
```

In [10]:
```
data['Crop_Type'].unique()
```

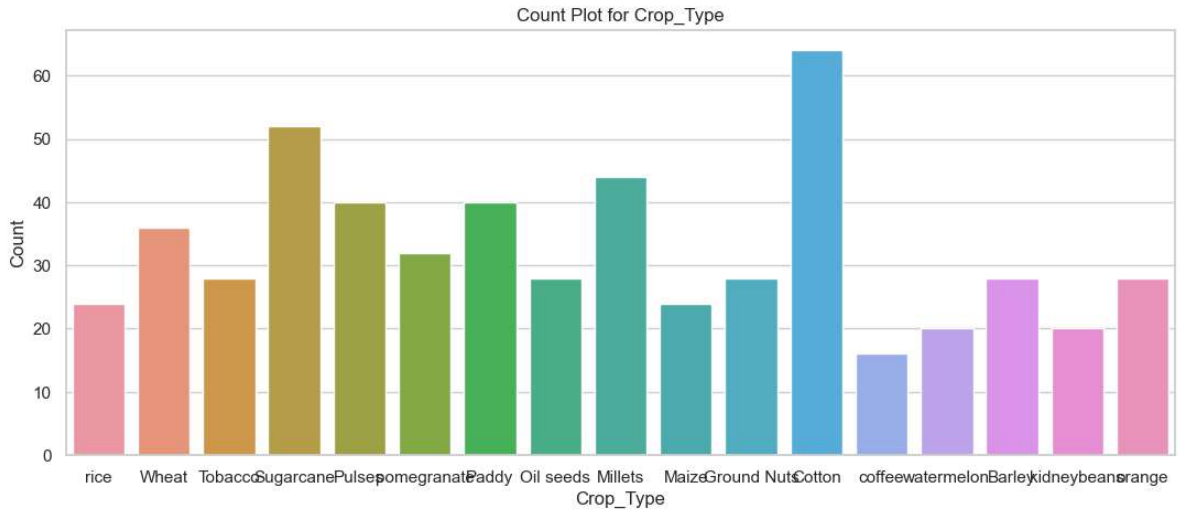
Out[10]:
```
array(['rice', 'Wheat', 'Tobacco', 'Sugarcane', 'Pulses', 'pomegranate',
       'Paddy', 'Oil seeds', 'Millets', 'Maize', 'Ground Nuts', 'Cotton',
       'coffee', 'watermelon', 'Barley', 'kidneybeans', 'orange'],
      dtype=object)
```

In [11]:
```
#statistical parameters
data.describe(include='all')
```

Out[11]:

	Temparature	Humidity	Moisture	Soil_Type	Crop_Type	Nitrogen	Potassium	Ph
count	552.000000	552.000000	552.000000	552	552	552.000000	552.000000	
unique	NaN	NaN	NaN	5	17	NaN	NaN	
top	NaN	NaN	NaN	Loamy	Cotton	NaN	NaN	
freq	NaN	NaN	NaN	192	64	NaN	NaN	
mean	28.630435	64.557971	42.840580	NaN	NaN	28.521739	10.144928	
std	5.088082	11.880236	11.507275	NaN	NaN	29.121989	13.456956	
min	0.000000	50.000000	25.000000	NaN	NaN	0.000000	0.000000	
25%	26.000000	54.000000	33.000000	NaN	NaN	10.000000	0.000000	
50%	29.000000	62.000000	41.000000	NaN	NaN	15.000000	0.000000	
75%	32.000000	68.000000	51.000000	NaN	NaN	37.000000	18.000000	
max	38.000000	95.000000	65.000000	NaN	NaN	126.000000	59.000000	

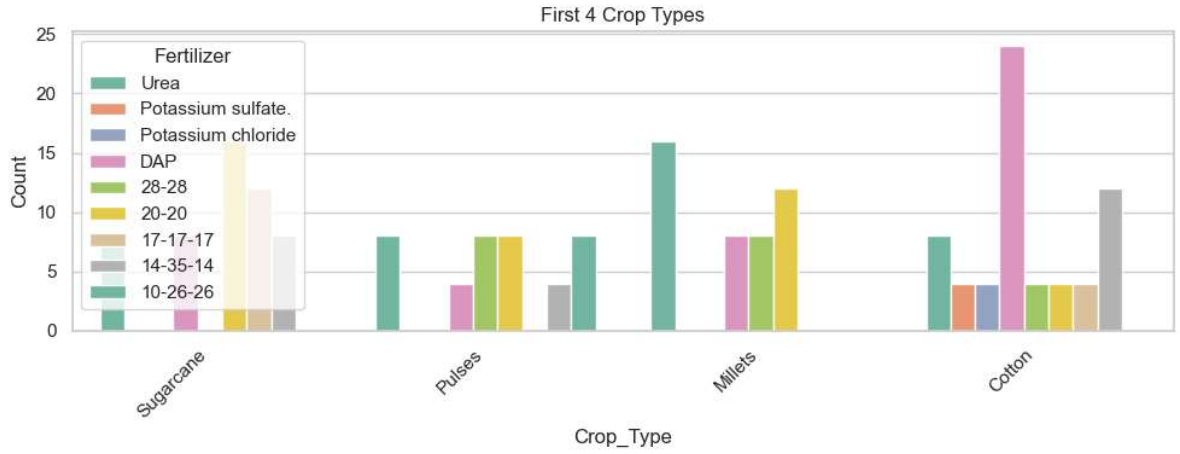
In [12]:
```
plt.figure(figsize=(13, 5))
sns.set(style="whitegrid")
sns.countplot(data=data, x='Crop_Type')
plt.title('Count Plot for Crop_Type')
plt.xlabel('Crop_Type')
plt.ylabel('Count')
plt.show()
```



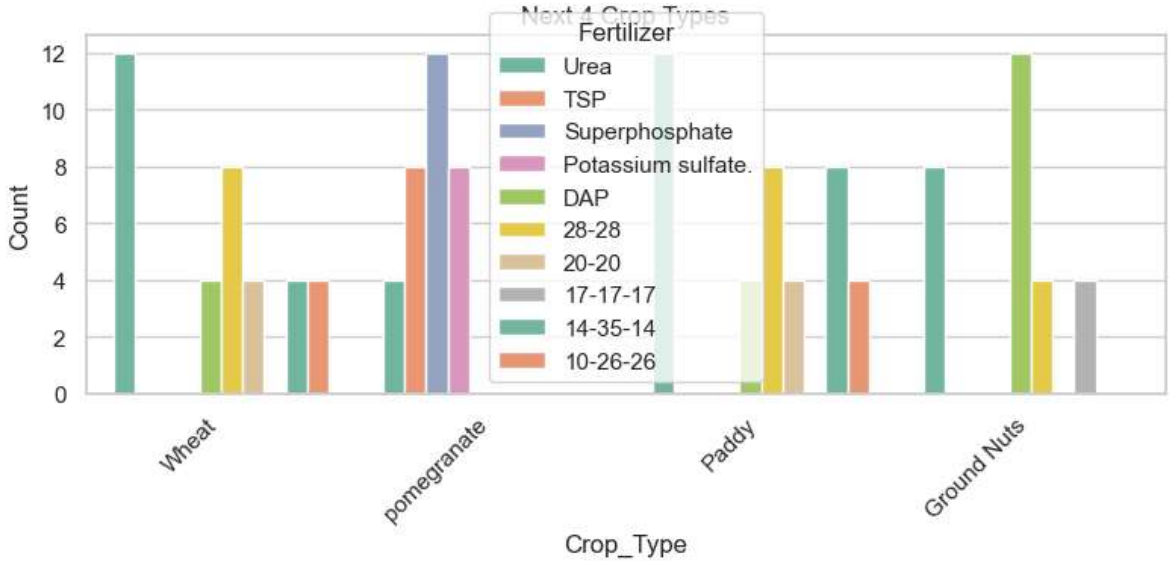
In [13]:
```
#The plot that shows the count (frequency) of each unique crop type in the
#The x-axis represents the different crop types.
#The y-axis represents the count (the number of occurrences) of each crop
```

In [14]:

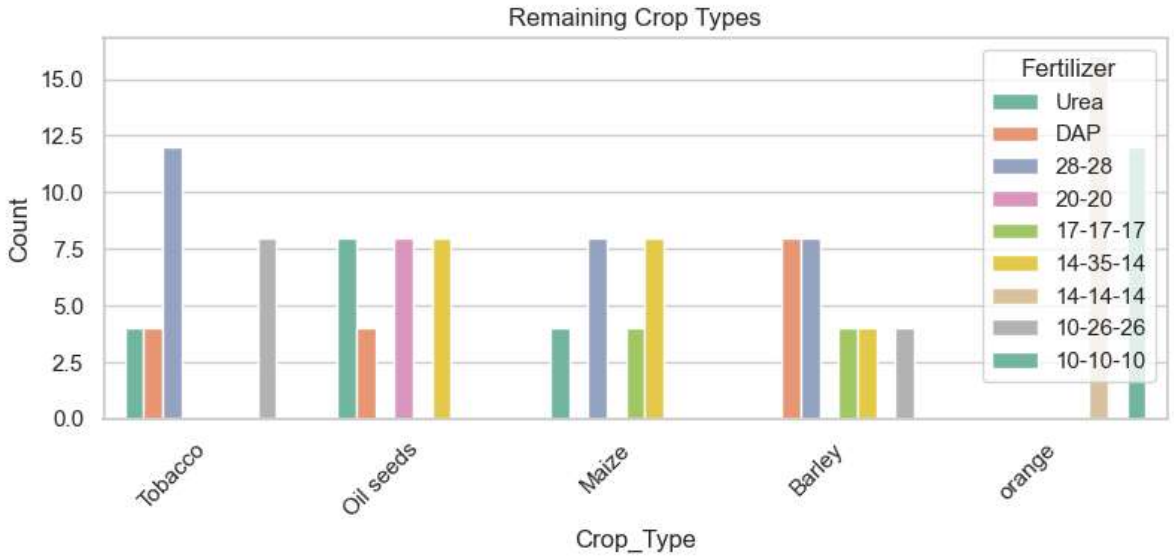
```
#first 4 crop types
part1_data = data[data['Crop_Type'].isin(data['Crop_Type'].value_counts()

# Create the first countplot
plt.figure(figsize=(10, 4))
sns.set(style="whitegrid")
sns.countplot(data=part1_data, x='Crop_Type', hue='Fertilizer', width=0.8
plt.title('First 4 Crop Types')
plt.xlabel('Crop_Type')
plt.ylabel('Count')
plt.legend(title='Fertilizer')
plt.xticks(rotation=45, horizontalalignment='right')
plt.tight_layout()
plt.show()
```



In [15]:
```
# Split the data into three parts: next 4 crop types
part2_data = data[data['Crop_Type'].isin(data['Crop_Type'].value_counts()

# Create the second countplot
plt.figure(figsize=(8, 4))
sns.set(style="whitegrid")
sns.countplot(data=part2_data, x='Crop_Type', hue='Fertilizer', width=0.8
plt.title('Next 4 Crop Types')
plt.xlabel('Crop_Type')
plt.ylabel('Count')
plt.legend(title='Fertilizer')
plt.xticks(rotation=45, horizontalalignment='right')
plt.tight_layout()
plt.show()

```



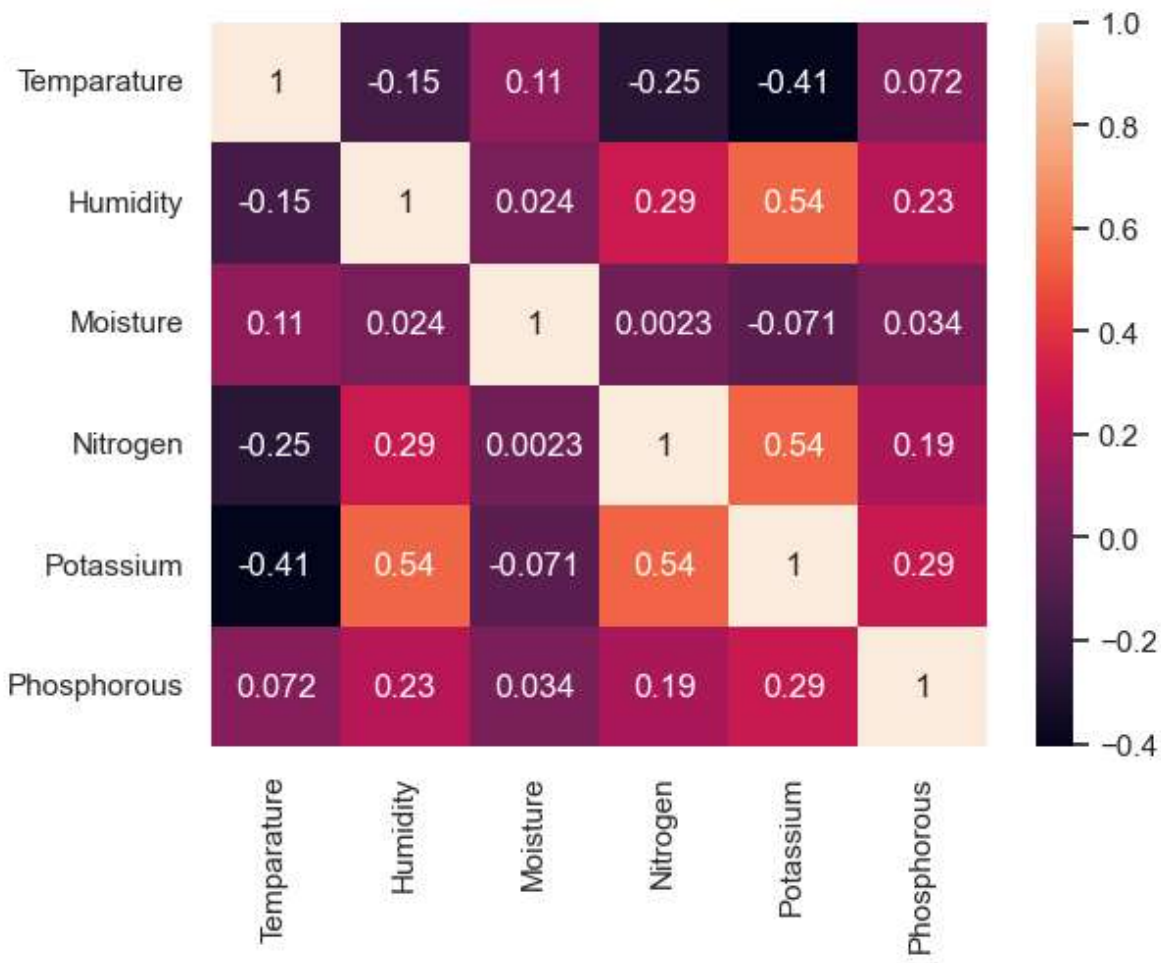
In [16]:
```
# Split the data into three parts: remaining crop types
part3_data = data[data['Crop_Type'].isin(data['Crop_Type'].value_counts()

# Create the third countplot
plt.figure(figsize=(8, 4))
sns.set(style="whitegrid")
sns.countplot(data=part3_data, x='Crop_Type', hue='Fertilizer', width=0.8
plt.title('Remaining Crop Types')
plt.xlabel('Crop_Type')
plt.ylabel('Count')
plt.legend(title='Fertilizer')
plt.xticks(rotation=45, horizontalalignment='right')
plt.tight_layout()
plt.show()

```



In [17]:
```
#  this plot is provides insights into how different crop types are distr
# The x-axis represents the different crop types.
# The y-axis represents the count (the number of occurrences) of each crop

```

In [18]:
```
#Heatmap for Correlation Analysis
sns.heatmap(data.corr(),annot=True)
```

Out[18]: <Axes: >



In [19]:
```
#here is no such correlation between any of variables..
```

In [20]:
```
#encoding the labels for categorical variables
from sklearn.preprocessing import LabelEncoder
#it  transforming non-numeric data into a numeric format
```

In [21]:
```
#encoding Soil Type variable
encode_soil = LabelEncoder()

#fitting the label encoder
data.Soil_Type = encode_soil.fit_transform(data.Soil_Type)

#creating the DataFrame
Soil_Type = pd.DataFrame(zip(encode_soil.classes_,encode_soil.transform(e
Soil_Type = Soil_Type.set_index('Original')
Soil_Type
```

Out[21]:

Original	Encoded
Black	0
Clayey	1
Loamy	2
Red	3
Sandy	4

In [22]:
```
#encoding Crop Type variable
encode_crop = LabelEncoder()

#fitting the label encoder
data.Crop_Type = encode_crop.fit_transform(data.Crop_Type)

#creating the DataFrame
Crop_Type = pd.DataFrame(zip(encode_crop.classes_,encode_crop.transform(e
Crop_Type = Crop_Type.set_index('Original')
Crop_Type
```

Out[22]:

Original	Encoded
Barley	0
Cotton	1
Ground Nuts	2
Maize	3
Millets	4
Oil seeds	5
Paddy	6
Pulses	7
Sugarcane	8
Tobacco	9
Wheat	10
coffee	11
kidneybeans	12
orange	13
pomegranate	14
rice	15
watermelon	16

In [23]:
```
#encoding Fertilizer variable
encode_ferti = LabelEncoder()

#fitting the label encoder
data.Fertilizer = encode_ferti.fit_transform(data.Fertilizer)

#creating the DataFrame
Fertilizer = pd.DataFrame(zip(encode_ferti.classes_,encode_ferti.transfor
Fertilizer = Fertilizer.set_index('Original')
Fertilizer
```

Out[23]:

Original	Encoded
10-10-10	0
10-26-26	1
14-14-14	2
14-35-14	3
15-15-15	4
17-17-17	5
20-20	6
28-28	7
DAP	8
Potassium chloride	9
Potassium sulfate.	10
Superphosphate	11
TSP	12
Urea	13

In [24]:
```
#splitting the data into train and test
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(data.drop('Fertilizer
print('Shape of Splitting :')
print('x_train = {}, y_train = {}, x_test = {}, y_test = {}'.format(x_tra
```

```
Shape of Splitting :
x_train = (441, 8), y_train = (441,), x_test = (111, 8), y_test = (111,)
```

In [25]:
```
# here 20% of the data should be used for testing (evaluation), and the re
#x_train and x_test = contain the features (independent variables) used fo
#y_train and y_test = contains the labels(dependent variable) used for tra
```

In [26]:
```
1 x_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 441 entries, 213 to 37
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Temparature  441 non-null    int64
 1   Humidity     441 non-null    int64
 2   Moisture     441 non-null    int64
 3   Soil_Type    441 non-null    int32
 4   Crop_Type    441 non-null    int32
 5   Nitrogen     441 non-null    int64
 6   Potassium    441 non-null    int64
 7   Phosphorous  441 non-null    int64
dtypes: int32(2), int64(6)
memory usage: 27.6 KB
```

In [27]:
```
1 acc = [] # TEST
2 model = []
3 acc1=[] # TRIAN
```

Logistic regression model

In [29]:

```
#importing libraries
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report
from sklearn import metrics
from sklearn import tree
import warnings
warnings.filterwarnings('ignore')

ds = DecisionTreeClassifier(criterion="entropy",random_state=2,max_depth=!
ds.fit(x_train,y_train)

predicted_values = ds.predict(x_test)
x = metrics.accuracy_score(y_test, predicted_values)
acc.append(x)
model.append('Decision Tree')

predicted_values = ds.predict(x_train)
y = metrics.accuracy_score(y_train, predicted_values)
acc1.append(y)

print("DecisionTrees's Accuracy is: ", x*100, y*100)

print(classification_report(y_test,predicted_values))
```

```
DecisionTrees's Accuracy is:  90.09009009009009 93.42403628117914
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
e:\DATA_SCIENCE_PRACTISE_FILES\Project\college projects\New folder (2)\Fertil
izer-Prediction-main\finalferti.ipynb Cell 28 line 2
     <a href='vscode-notebook-cell:/e%3A/DATA_SCIENCE_PRACTISE_FILES/Project/
college%20projects/New%20folder%20%282%29/Fertilizer-Prediction-main/finalfer
ti.ipynb#X40sZmlsZQ%3D%3D?line=18'>19</a> acc1.append(y)
     <a href='vscode-notebook-cell:/e%3A/DATA_SCIENCE_PRACTISE_FILES/Project/
college%20projects/New%20folder%20%282%29/Fertilizer-Prediction-main/finalfer
ti.ipynb#X40sZmlsZQ%3D%3D?line=20'>21</a> print("DecisionTrees's Accuracy is:
", x*100, y*100)
---> <a href='vscode-notebook-cell:/e%3A/DATA_SCIENCE_PRACTISE_FILES/Project/
college%20projects/New%20folder%20%282%29/Fertilizer-Prediction-main/finalfer
ti.ipynb#X40sZmlsZQ%3D%3D?line=22'>23</a> print(classification_report(y_test,
predicted_values))

File c:\Users\msi1\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:2310, in classification_report(y_true, y_pred, labels, target_names, sam
ple_weight, digits, output_dict, zero_division)
   2195 def classification_report(
   2196     y_true,
   2197     y_pred,
   (...)
   2204     zero_division="warn",
   2205 ):
   2206     """Build a text report showing the main classification metrics.
   2207
   2208     Read more in the :ref:`User Guide <classification_report>`.
   (...)
   2307     <BLANKLINE>
   2308     """
-> 2310     y_type, y_true, y_pred = _check_targets(y_true, y_pred)
   2312     if labels is None:
   2313         labels = unique_labels(y_true, y_pred)

File c:\Users\msi1\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:86, in _check_targets(y_true, y_pred)
     59 def _check_targets(y_true, y_pred):
     60     """Check that y_true and y_pred belong to the same classification
task.
     61
     62     This converts multiclass or binary types to a common shape, and r
aises a
   (...)
     84     y_pred : array or indicator matrix
     85     """
---> 86     check_consistent_length(y_true, y_pred)
     87     type_true = type_of_target(y_true, input_name="y_true")
     88     type_pred = type_of_target(y_pred, input_name="y_pred")

File c:\Users\msi1\anaconda3\lib\site-packages\sklearn\utils\validation.py:39
7, in check_consistent_length(*arrays)
    395 uniques = np.unique(lengths)
    396 if len(uniques) > 1:
--> 397     raise ValueError(
    398         "Found input variables with inconsistent numbers of samples:
%r"
```

```
    399         % [int(l) for l in lengths]
    400     )
```

```
ValueError: Found input variables with inconsistent numbers of samples: [111,
441]
```

In [30]:

```
from sklearn.naive_bayes import GaussianNB

NaiveBayes = GaussianNB()

NaiveBayes.fit(x_train,y_train)

predicted_values = NaiveBayes.predict(x_test)
x = metrics.accuracy_score(y_test, predicted_values)
acc.append(x)

predicted_values = NaiveBayes.predict(x_train)
y = metrics.accuracy_score(y_train, predicted_values)
acc1.append(y)

model.append('Naive Bayes')
print("Naive Bayes's Accuracy is: ", x,y)

print(classification_report(y_test,predicted_values))
```

```
Naive Bayes's Accuracy is:  0.9459459459459459 0.9138321995464853
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
e:\DATA_SCIENCE_PRACTISE_FILES\Project\college projects\New folder (2)\Fertil
izer-Prediction-main\finalferti.ipynb Cell 29 line 1
     <a href='vscode-notebook-cell:/e%3A/DATA_SCIENCE_PRACTISE_FILES/Project/
college%20projects/New%20folder%20%282%29/Fertilizer-Prediction-main/finalfer
ti.ipynb#X41sZmlsZQ%3D%3D?line=14'>15</a> model.append('Naive Bayes')
     <a href='vscode-notebook-cell:/e%3A/DATA_SCIENCE_PRACTISE_FILES/Project/
college%20projects/New%20folder%20%282%29/Fertilizer-Prediction-main/finalfer
ti.ipynb#X41sZmlsZQ%3D%3D?line=15'>16</a> print("Naive Bayes's Accuracy is:
", x,y)
---> <a href='vscode-notebook-cell:/e%3A/DATA_SCIENCE_PRACTISE_FILES/Project/
college%20projects/New%20folder%20%282%29/Fertilizer-Prediction-main/finalfer
ti.ipynb#X41sZmlsZQ%3D%3D?line=17'>18</a> print(classification_report(y_test,
predicted_values))

File c:\Users\msi1\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:2310, in classification_report(y_true, y_pred, labels, target_names, sam
ple_weight, digits, output_dict, zero_division)
   2195 def classification_report(
   2196     y_true,
   2197     y_pred,
   (...)
   2204     zero_division="warn",
   2205 ):
   2206     """Build a text report showing the main classification metrics.
   2207
   2208     Read more in the :ref:`User Guide <classification_report>`.
   (...)
   2307     <BLANKLINE>
   2308     """
-> 2310     y_type, y_true, y_pred = _check_targets(y_true, y_pred)
   2312     if labels is None:
   2313         labels = unique_labels(y_true, y_pred)

File c:\Users\msi1\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:86, in _check_targets(y_true, y_pred)
     59 def _check_targets(y_true, y_pred):
     60     """Check that y_true and y_pred belong to the same classification
task.
     61
     62     This converts multiclass or binary types to a common shape, and r
aises a
   (...)
     84     y_pred : array or indicator matrix
     85     """
---> 86     check_consistent_length(y_true, y_pred)
     87     type_true = type_of_target(y_true, input_name="y_true")
     88     type_pred = type_of_target(y_pred, input_name="y_pred")

File c:\Users\msi1\anaconda3\lib\site-packages\sklearn\utils\validation.py:39
7, in check_consistent_length(*arrays)
    395 uniques = np.unique(lengths)
    396 if len(uniques) > 1:
--> 397     raise ValueError(
    398         "Found input variables with inconsistent numbers of samples:
%r"
```

```
    399         % [int(l) for l in lengths]
    400     )
```

```
ValueError: Found input variables with inconsistent numbers of samples: [111,
441]
```

In [31]:

```
from sklearn.svm import SVC
# data normalization with sklearn
from sklearn.preprocessing import MinMaxScaler
# fit scaler on training data
norm = MinMaxScaler().fit(x_train)
X_train_norm = norm.transform(x_train)
# transform testing dataabs
X_test_norm = norm.transform(x_test)
SVM = SVC(kernel='poly', degree=3, C=1)
SVM.fit(X_train_norm,y_train)

predicted_values = SVM.predict(X_test_norm)
x = metrics.accuracy_score(y_test, predicted_values)
acc.append(x)

predicted_values = SVM.predict(X_train_norm)
y = metrics.accuracy_score(y_train, predicted_values)
acc1.append(y)

model.append('SVM')
print("SVM's Accuracy is: ", x,y)

print(classification_report(y_test,predicted_values))
```

```
SVM's Accuracy is:  0.990990990990991 0.9931972789115646
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
e:\DATA_SCIENCE_PRACTISE_FILES\Project\college projects\New folder (2)\Fertil
izer-Prediction-main\finalferti.ipynb Cell 30 line 2
     <a href='vscode-notebook-cell:/e%3A/DATA_SCIENCE_PRACTISE_FILES/Project/
college%20projects/New%20folder%20%282%29/Fertilizer-Prediction-main/finalfer
ti.ipynb#X42sZmlsZQ%3D%3D?line=19'>20</a> model.append('SVM')
     <a href='vscode-notebook-cell:/e%3A/DATA_SCIENCE_PRACTISE_FILES/Project/
college%20projects/New%20folder%20%282%29/Fertilizer-Prediction-main/finalfer
ti.ipynb#X42sZmlsZQ%3D%3D?line=20'>21</a> print("SVM's Accuracy is: ", x,y)
---> <a href='vscode-notebook-cell:/e%3A/DATA_SCIENCE_PRACTISE_FILES/Project/
college%20projects/New%20folder%20%282%29/Fertilizer-Prediction-main/finalfer
ti.ipynb#X42sZmlsZQ%3D%3D?line=22'>23</a> print(classification_report(y_test,
predicted_values))

File c:\Users\msi1\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:2310, in classification_report(y_true, y_pred, labels, target_names, sam
ple_weight, digits, output_dict, zero_division)
   2195 def classification_report(
   2196     y_true,
   2197     y_pred,
   (...)
   2204     zero_division="warn",
   2205 ):
   2206     """Build a text report showing the main classification metrics.
   2207
   2208     Read more in the :ref:`User Guide <classification_report>`.
   (...)
   2307     <BLANKLINE>
   2308     """
-> 2310     y_type, y_true, y_pred = _check_targets(y_true, y_pred)
   2312     if labels is None:
   2313         labels = unique_labels(y_true, y_pred)

File c:\Users\msi1\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:86, in _check_targets(y_true, y_pred)
     59 def _check_targets(y_true, y_pred):
     60     """Check that y_true and y_pred belong to the same classification
task.
     61
     62     This converts multiclass or binary types to a common shape, and r
aises a
   (...)
     84     y_pred : array or indicator matrix
     85     """
---> 86     check_consistent_length(y_true, y_pred)
     87     type_true = type_of_target(y_true, input_name="y_true")
     88     type_pred = type_of_target(y_pred, input_name="y_pred")

File c:\Users\msi1\anaconda3\lib\site-packages\sklearn\utils\validation.py:39
7, in check_consistent_length(*arrays)
    395 uniques = np.unique(lengths)
    396 if len(uniques) > 1:
--> 397     raise ValueError(
    398         "Found input variables with inconsistent numbers of samples:
%r"
    399         % [int(l) for l in lengths]
```

```
    400     )
```

```
ValueError: Found input variables with inconsistent numbers of samples: [111,
441]
```

In [32]:

```
from sklearn.linear_model import LogisticRegression

LogReg = LogisticRegression(random_state=2)

LogReg.fit(x_train,y_train)

predicted_values = LogReg.predict(x_test)
x = metrics.accuracy_score(y_test, predicted_values)
acc.append(x)

predicted_values = LogReg.predict(x_train)
y = metrics.accuracy_score(y_train, predicted_values)
acc1.append(y)

model.append('Logistic Regression')
print("Logistic Regression's Accuracy is: ", x,y)

print(classification_report(y_test,predicted_values))
```

```
Logistic Regression's Accuracy is:  0.7837837837837838 0.909297052154195
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
e:\DATA_SCIENCE_PRACTISE_FILES\Project\college projects\New folder (2)\Fertil
izer-Prediction-main\finalferti.ipynb Cell 31 line 1
     <a href='vscode-notebook-cell:/e%3A/DATA_SCIENCE_PRACTISE_FILES/Project/
college%20projects/New%20folder%20%282%29/Fertilizer-Prediction-main/finalfer
ti.ipynb#X43sZmlsZQ%3D%3D?line=14'>15</a> model.append('Logistic Regression')
     <a href='vscode-notebook-cell:/e%3A/DATA_SCIENCE_PRACTISE_FILES/Project/
college%20projects/New%20folder%20%282%29/Fertilizer-Prediction-main/finalfer
ti.ipynb#X43sZmlsZQ%3D%3D?line=15'>16</a> print("Logistic Regression's Accura
cy is: ", x,y)
---> <a href='vscode-notebook-cell:/e%3A/DATA_SCIENCE_PRACTISE_FILES/Project/
college%20projects/New%20folder%20%282%29/Fertilizer-Prediction-main/finalfer
ti.ipynb#X43sZmlsZQ%3D%3D?line=17'>18</a> print(classification_report(y_test,
predicted_values))

File c:\Users\msi1\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:2310, in classification_report(y_true, y_pred, labels, target_names, sam
ple_weight, digits, output_dict, zero_division)
   2195 def classification_report(
   2196     y_true,
   2197     y_pred,
   (...)
   2204     zero_division="warn",
   2205 ):
   2206     """Build a text report showing the main classification metrics.
   2207
   2208     Read more in the :ref:`User Guide <classification_report>`.
   (...)
   2307     <BLANKLINE>
   2308     """
-> 2310     y_type, y_true, y_pred = _check_targets(y_true, y_pred)
   2312     if labels is None:
   2313         labels = unique_labels(y_true, y_pred)

File c:\Users\msi1\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:86, in _check_targets(y_true, y_pred)
     59 def _check_targets(y_true, y_pred):
     60     """Check that y_true and y_pred belong to the same classification
task.
     61
     62     This converts multiclass or binary types to a common shape, and r
aises a
   (...)
     84     y_pred : array or indicator matrix
     85     """
---> 86     check_consistent_length(y_true, y_pred)
     87     type_true = type_of_target(y_true, input_name="y_true")
     88     type_pred = type_of_target(y_pred, input_name="y_pred")

File c:\Users\msi1\anaconda3\lib\site-packages\sklearn\utils\validation.py:39
7, in check_consistent_length(*arrays)
    395 uniques = np.unique(lengths)
    396 if len(uniques) > 1:
--> 397     raise ValueError(
    398         "Found input variables with inconsistent numbers of samples:
%r"
```

```
    399         % [int(l) for l in lengths]
    400     )
```

```
ValueError: Found input variables with inconsistent numbers of samples: [111,
441]
```

In [33]:

```
from sklearn.ensemble import RandomForestClassifier

RF = RandomForestClassifier(n_estimators=20, random_state=0)
RF.fit(x_train,y_train)

predicted_values = RF.predict(x_test)
x = metrics.accuracy_score(y_test, predicted_values)
acc.append(x)

predicted_values = RF.predict(x_train)
y = metrics.accuracy_score(y_train, predicted_values)
acc1.append(y)

model.append('RF')
print("RF's Accuracy is: ", x,y)

print(classification_report(y_test,predicted_values))
```

```
RF's Accuracy is:  1.0 1.0
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
e:\DATA_SCIENCE_PRACTISE_FILES\Project\college projects\New folder (2)\Fertil
izer-Prediction-main\finalferti.ipynb Cell 32 line 1
     <a href='vscode-notebook-cell:/e%3A/DATA_SCIENCE_PRACTISE_FILES/Project/
college%20projects/New%20folder%20%282%29/Fertilizer-Prediction-main/finalfer
ti.ipynb#X44sZmlsZQ%3D%3D?line=13'>14</a> model.append('RF')
     <a href='vscode-notebook-cell:/e%3A/DATA_SCIENCE_PRACTISE_FILES/Project/
college%20projects/New%20folder%20%282%29/Fertilizer-Prediction-main/finalfer
ti.ipynb#X44sZmlsZQ%3D%3D?line=14'>15</a> print("RF's Accuracy is: ", x,y)
---> <a href='vscode-notebook-cell:/e%3A/DATA_SCIENCE_PRACTISE_FILES/Project/
college%20projects/New%20folder%20%282%29/Fertilizer-Prediction-main/finalfer
ti.ipynb#X44sZmlsZQ%3D%3D?line=16'>17</a> print(classification_report(y_test,
predicted_values))

File c:\Users\msi1\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:2310, in classification_report(y_true, y_pred, labels, target_names, sam
ple_weight, digits, output_dict, zero_division)
   2195 def classification_report(
   2196     y_true,
   2197     y_pred,
   (...)
   2204     zero_division="warn",
   2205 ):
   2206     """Build a text report showing the main classification metrics.
   2207
   2208     Read more in the :ref:`User Guide <classification_report>`.
   (...)
   2307     <BLANKLINE>
   2308     """
-> 2310     y_type, y_true, y_pred = _check_targets(y_true, y_pred)
   2312     if labels is None:
   2313         labels = unique_labels(y_true, y_pred)

File c:\Users\msi1\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:86, in _check_targets(y_true, y_pred)
     59 def _check_targets(y_true, y_pred):
     60     """Check that y_true and y_pred belong to the same classification
task.
     61
     62     This converts multiclass or binary types to a common shape, and r
aises a
   (...)
     84     y_pred : array or indicator matrix
     85     """
---> 86     check_consistent_length(y_true, y_pred)
     87     type_true = type_of_target(y_true, input_name="y_true")
     88     type_pred = type_of_target(y_pred, input_name="y_pred")

File c:\Users\msi1\anaconda3\lib\site-packages\sklearn\utils\validation.py:39
7, in check_consistent_length(*arrays)
    395 uniques = np.unique(lengths)
    396 if len(uniques) > 1:
--> 397     raise ValueError(
    398         "Found input variables with inconsistent numbers of samples:
%r"
    399         % [int(l) for l in lengths]
```

```
    400     )
```

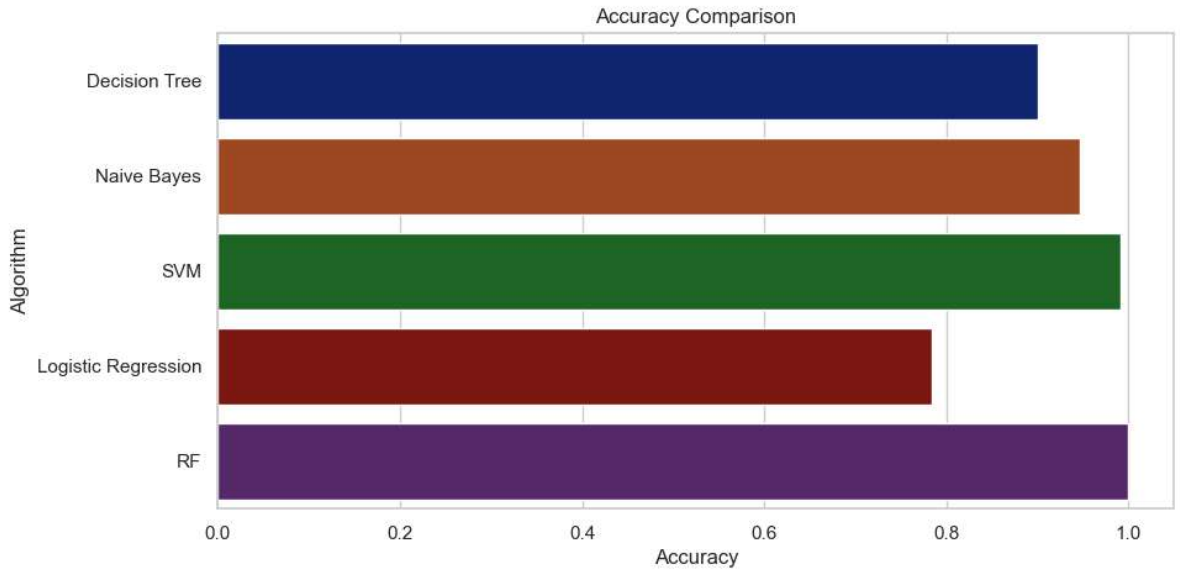
```
ValueError: Found input variables with inconsistent numbers of samples: [111,
441]
```

In [ ]:
```
from sklearn.model_selection import cross_val_score

score = cross_val_score(RF,data,data.Fertilizer,cv=5)
print("Cross-validation score of RF is:",score)
score = cross_val_score(LogReg,data,data.Fertilizer,cv=5)
print("Cross-validation score of LogReg is:",score)
score = cross_val_score(SVM,data,data.Fertilizer,cv=5)
print("Cross-validation score of SVM is:",score)
score = cross_val_score(NaiveBayes,data,data.Fertilizer,cv=5)
print("Cross-validation score of NaiveBayes is:",score)
score = cross_val_score(ds, data, data.Fertilizer,cv=5)
print("Cross-validation score of ds is:",score)
```

```
Cross-validation score of RF is: [1. 1. 1. 1. 1.]
Cross-validation score of LogReg is: [0.97297297 0.94594595 0.91818182 0.8818
1818 0.87272727]
Cross-validation score of SVM is: [0.96396396 0.92792793 0.91818182 0.8818181
8 0.98181818]
Cross-validation score of NaiveBayes is: [1. 1. 1. 1. 1.]
Cross-validation score of ds is: [1. 1. 1. 1. 1.]
```

In [ ]:
```
plt.figure(figsize=[10,5],dpi = 100)
plt.title('Accuracy Comparison')
plt.xlabel('Accuracy')
plt.ylabel('Algorithm')
sns.barplot(x = acc,y = model,palette='dark')
```

Out[35]:
```
<Axes: title={'center': 'Accuracy Comparison'}, xlabel='Accuracy', ylabel='Al
gorithm'>
```



In [ ]:
```
#pickling the file
import pickle
pickle_out = open('classifier.pkl','wb')
pickle.dump(RF,pickle_out)
pickle_out.close()
```

In [ ]:
```
model = pickle.load(open('classifier.pkl','rb'))
model.predict([[34,67,62,0,1,7,0,30]])
```

Out[37]:
```
array([6])
```

In [ ]:
```
model = pickle.load(open('classifier.pkl','rb'))
model.predict([[25,78,43,4,1,22,26,38]])
```

Out[38]:
```
array([10])
```

In [ ]:
```
#pickling the file
import pickle
pickle_out = open('fertilizer.pkl','wb')
pickle.dump(encode_ferti,pickle_out)
pickle_out.close()
```

In [ ]:
```
ferti = pickle.load(open('fertilizer.pkl','rb'))
ferti.classes_[1]
```

Out[41]:
```
'10-26-26'
```

In [ ]:
```

```

In [ ]:
```

```