Crop Dataset - Random Forest Classification

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

from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import accuracy_score, classification_report

df=pd.read_csv("C:/Users/user/Downloads/Crop_recommendation.csv")

df
```

```
         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 x 8 columns]
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2200 entries, 0 to 2199
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
```

```
 0   N            2200 non-null   int64
 1   P            2200 non-null   int64
 2   K            2200 non-null   int64
 3   temperature  2200 non-null   float64
 4   humidity     2200 non-null   float64
 5   ph           2200 non-null   float64
 6   rainfall     2200 non-null   float64
 7   label        2200 non-null   object
dtypes: float64(4), int64(3), object(1)
memory usage: 137.6+ KB
```

```
df.isnull().sum()
```

```
N              0
P              0
K              0
temperature    0
humidity       0
ph             0
rainfall       0
label          0
dtype: int64
```

```
df.describe()
```

```
                 N            P            K  temperature     humidity
\
count  2200.000000  2200.000000  2200.000000  2200.000000  2200.000000

mean     50.551818    53.362727    48.149091    25.616244    71.481779

std      36.917334    32.985883    50.647931     5.063749    22.263812

min       0.000000     5.000000     5.000000     8.825675    14.258040

25%      21.000000    28.000000    20.000000    22.769375    60.261953

50%      37.000000    51.000000    32.000000    25.598693    80.473146

75%      84.250000    68.000000    49.000000    28.561654    89.948771

max     140.000000   145.000000   205.000000    43.675493    99.981876


                ph     rainfall
count  2200.000000  2200.000000
mean      6.469480   103.463655
std       0.773938    54.958389
min       3.504752    20.211267
25%       5.971693    64.551686
50%       6.425045    94.867624
```

```
75%       6.923643   124.267508
max       9.935091   298.560117
```

```
df = df.dropna()
```

```
df
```

```
         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 x 8 columns]
```

```
df.duplicated().sum()
```

```
0
```

```
from sklearn.preprocessing import LabelEncoder
```

```
label_encoder = LabelEncoder()
```

```
df.columns
```

```
Index(['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall',
'label'], dtype='object')
```

```
df['label'] = label_encoder.fit_transform(df['label'])
```

```
df['label']
```

```
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: int32
```

```
df.isnull().sum()
```

```
N              0
P              0
K              0
temperature    0
humidity       0
ph             0
rainfall       0
label          0
dtype: int64
```

```
correlation_matrix = df.corr()
```

```
correlation_matrix
```

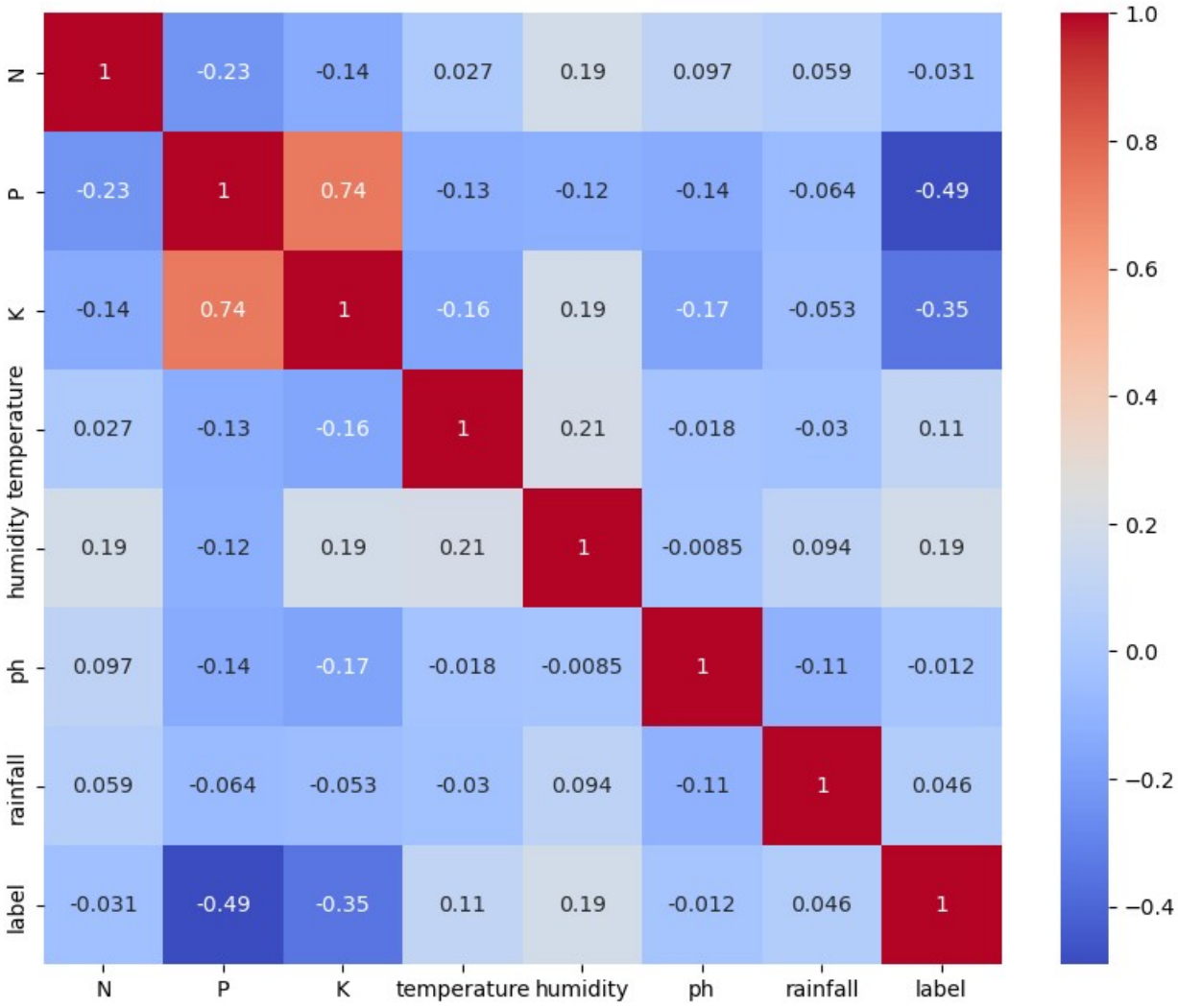
```
                    N         P         K  temperature  humidity
ph  \
N            1.000000 -0.231460 -0.140512     0.026504  0.190688
0.096683
P           -0.231460  1.000000  0.736232    -0.127541 -0.118734 -
0.138019
K           -0.140512  0.736232  1.000000    -0.160387  0.190859 -
0.169503
temperature  0.026504 -0.127541 -0.160387     1.000000  0.205320 -
0.017795
humidity     0.190688 -0.118734  0.190859     0.205320  1.000000 -
0.008483
ph           0.096683 -0.138019 -0.169503    -0.017795 -0.008483
1.000000
rainfall     0.059020 -0.063839 -0.053461    -0.030084  0.094423 -
0.109069
label       -0.031130 -0.491006 -0.346417     0.113606  0.193911 -
0.012253

             rainfall     label
N            0.059020 -0.031130
```

```
P           -0.063839 -0.491006
K           -0.053461 -0.346417
temperature -0.030084  0.113606
humidity     0.094423  0.193911
ph          -0.109069 -0.012253
rainfall     1.000000  0.045611
label        0.045611  1.000000
```

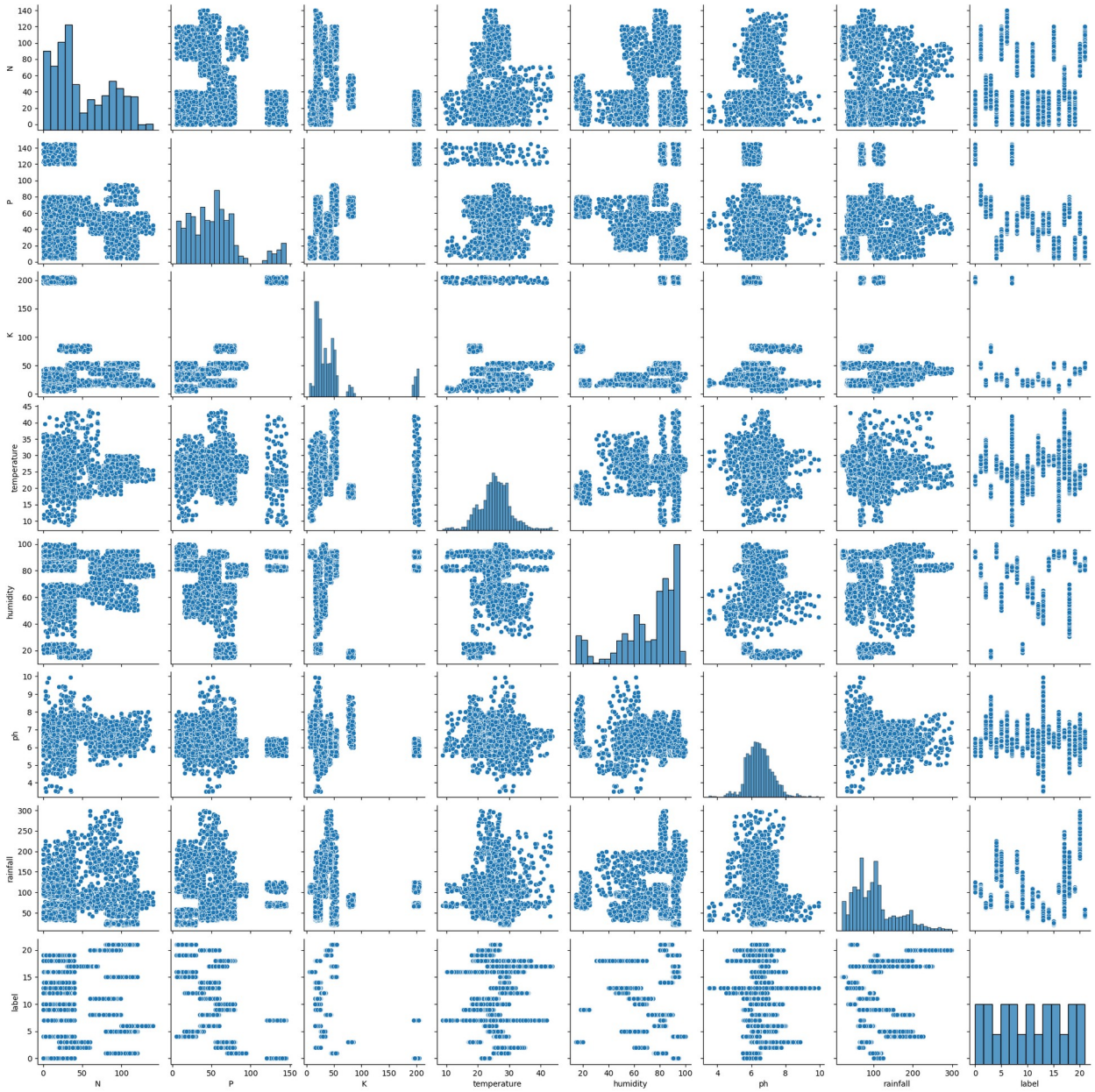
```
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
```

```
<Axes: >
```




```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x1b9fb199b90>
```




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

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
4	78	42	42	20.130175	81.604873	7.628473	262.717340
...	...	..	..	...	...	...	...
2195	107	34	32	26.774637	66.413269	6.780064	177.774507

```
2196   99  15  27    27.417112  56.636362  6.086922  127.924610
2197  118  33  30    24.131797  67.225123  6.362608  173.322839
2198  117  32  34    26.272418  52.127394  6.758793  127.175293
2199  104  18  30    23.603016  60.396475  6.779833  140.937041

[2200 rows x 7 columns]
```

```
y = df['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: int32
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((1760, 7), (440, 7), (1760,), (440,))
```

```
le = LabelEncoder()
y_encoded = le.fit_transform(y)

from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()

param_grid = {
    'n_estimators': [50, 100],
    'max_depth': [5, 10],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2],}

grid_search = GridSearchCV(estimator=rfc, param_grid=param_grid, cv=3,
verbose=2)

grid_search.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 16 candidates, totalling 48 fits
[CV] END max_depth=5, min_samples_leaf=1, min_samples_split=2,
n_estimators=50; total time=   0.5s
[CV] END max_depth=5, min_samples_leaf=1, min_samples_split=2,
```

```
n_estimators=50; total time=   0.2s
[CV] END max_depth=5, min_samples_leaf=1, min_samples_split=2,
n_estimators=50; total time=   0.2s
[CV] END max_depth=5, min_samples_leaf=1, min_samples_split=2,
n_estimators=100; total time=   0.6s
[CV] END max_depth=5, min_samples_leaf=1, min_samples_split=2,
n_estimators=100; total time=   0.8s
[CV] END max_depth=5, min_samples_leaf=1, min_samples_split=2,
n_estimators=100; total time=   0.8s
[CV] END max_depth=5, min_samples_leaf=1, min_samples_split=5,
n_estimators=50; total time=   0.3s
[CV] END max_depth=5, min_samples_leaf=1, min_samples_split=5,
n_estimators=50; total time=   0.2s
[CV] END max_depth=5, min_samples_leaf=1, min_samples_split=5,
n_estimators=50; total time=   0.2s
[CV] END max_depth=5, min_samples_leaf=1, min_samples_split=5,
n_estimators=100; total time=   0.7s
[CV] END max_depth=5, min_samples_leaf=1, min_samples_split=5,
n_estimators=100; total time=   0.5s
[CV] END max_depth=5, min_samples_leaf=1, min_samples_split=5,
n_estimators=100; total time=   0.6s
[CV] END max_depth=5, min_samples_leaf=2, min_samples_split=2,
n_estimators=50; total time=   0.2s
[CV] END max_depth=5, min_samples_leaf=2, min_samples_split=2,
n_estimators=50; total time=   0.2s
[CV] END max_depth=5, min_samples_leaf=2, min_samples_split=2,
n_estimators=50; total time=   0.2s
[CV] END max_depth=5, min_samples_leaf=2, min_samples_split=2,
n_estimators=100; total time=   0.5s
[CV] END max_depth=5, min_samples_leaf=2, min_samples_split=2,
n_estimators=100; total time=   0.5s
[CV] END max_depth=5, min_samples_leaf=2, min_samples_split=2,
n_estimators=100; total time=   0.8s
[CV] END max_depth=5, min_samples_leaf=2, min_samples_split=5,
n_estimators=50; total time=   0.7s
[CV] END max_depth=5, min_samples_leaf=2, min_samples_split=5,
n_estimators=50; total time=   0.9s
[CV] END max_depth=5, min_samples_leaf=2, min_samples_split=5,
n_estimators=50; total time=   0.3s
[CV] END max_depth=5, min_samples_leaf=2, min_samples_split=5,
n_estimators=100; total time=   0.7s
[CV] END max_depth=5, min_samples_leaf=2, min_samples_split=5,
n_estimators=100; total time=   0.6s
[CV] END max_depth=5, min_samples_leaf=2, min_samples_split=5,
n_estimators=100; total time=   0.6s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2,
n_estimators=50; total time=   0.3s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2,
n_estimators=50; total time=   0.3s
```

```
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2,
n_estimators=50; total time=   0.3s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2,
n_estimators=100; total time=   0.6s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2,
n_estimators=100; total time=   0.7s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2,
n_estimators=100; total time=   0.7s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=5,
n_estimators=50; total time=   0.4s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=5,
n_estimators=50; total time=   0.3s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=5,
n_estimators=50; total time=   0.3s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=5,
n_estimators=100; total time=   0.7s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=5,
n_estimators=100; total time=   0.7s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=5,
n_estimators=100; total time=   0.7s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2,
n_estimators=50; total time=   0.3s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2,
n_estimators=50; total time=   0.5s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2,
n_estimators=50; total time=   0.3s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2,
n_estimators=100; total time=   0.7s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2,
n_estimators=100; total time=   0.7s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2,
n_estimators=100; total time=   0.8s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=5,
n_estimators=50; total time=   0.3s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=5,
n_estimators=50; total time=   0.3s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=5,
n_estimators=50; total time=   0.4s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=5,
n_estimators=100; total time=   0.9s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=5,
n_estimators=100; total time=   0.8s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=5,
n_estimators=100; total time=   0.7s

GridSearchCV(cv=3, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [5, 10], 'min_samples_leaf': [1,
2],
                         'min_samples_split': [2, 5],
```

```
                           'n_estimators': [50, 100]},
             verbose=2)

best_model = grid_search.best_estimator_

best_model

RandomForestClassifier(max_depth=10, min_samples_split=5)

y_pred = best_model.predict(X_test)

y_pred

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

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

```
accuracy = accuracy_score(y_test, y_pred)
```

```
accuracy
```

```
0.9931818181818182
```

```
report = classification_report(y_test, y_pred)
```

```
print('Classification Report:')
```

```
Classification Report:
```

```
print(report)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        23
           1       1.00      1.00      1.00        21
           2       1.00      1.00      1.00        20
           3       1.00      1.00      1.00        26
           4       1.00      1.00      1.00        27
           5       1.00      1.00      1.00        17
           6       1.00      1.00      1.00        17
           7       1.00      1.00      1.00        14
           8       0.92      1.00      0.96        23
           9       1.00      1.00      1.00        20
          10       0.92      1.00      0.96        11
          11       1.00      1.00      1.00        21
          12       1.00      1.00      1.00        19
          13       1.00      0.96      0.98        24
          14       1.00      1.00      1.00        19
          15       1.00      1.00      1.00        17
          16       1.00      1.00      1.00        14
          17       1.00      1.00      1.00        23
          18       1.00      1.00      1.00        23
          19       1.00      1.00      1.00        23
          20       1.00      0.89      0.94        19
```

```
          21       1.00      1.00      1.00        19

    accuracy                           0.99       440
   macro avg       0.99      0.99      0.99       440
weighted avg       0.99      0.99      0.99       440


r2_score(y_test,y_pred)

0.984051571953857
```

KNN Classification

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()

param_grid = {
    'n_neighbors': [3, 5, 7, 10],
    'weights': ['uniform', 'distance'],}

grid_search = GridSearchCV(estimator=knn, param_grid=param_grid, cv=2,
verbose=2)

grid_search.fit(X_train, y_train)

Fitting 2 folds for each of 8 candidates, totalling 16 fits
[CV] END ....................n_neighbors=3, weights=uniform; total
time=   0.1s
[CV] END ....................n_neighbors=3, weights=uniform; total
time=   0.1s
[CV] END ....................n_neighbors=3, weights=distance; total
time=   0.0s
[CV] END ....................n_neighbors=3, weights=distance; total
time=   0.0s
[CV] END ....................n_neighbors=5, weights=uniform; total
time=   0.0s
[CV] END ....................n_neighbors=5, weights=uniform; total
time=   0.1s
[CV] END ....................n_neighbors=5, weights=distance; total
time=   0.0s
[CV] END ....................n_neighbors=5, weights=distance; total
time=   0.0s
[CV] END ....................n_neighbors=7, weights=uniform; total
time=   0.0s
[CV] END ....................n_neighbors=7, weights=uniform; total
time=   0.0s
[CV] END ....................n_neighbors=7, weights=distance; total
```

```
time=   0.0s
[CV] END ..................n_neighbors=7, weights=distance; total
time=   0.0s
[CV] END ..................n_neighbors=10, weights=uniform; total
time=   0.1s
[CV] END ..................n_neighbors=10, weights=uniform; total
time=   0.1s
[CV] END ..................n_neighbors=10, weights=distance; total
time=   0.0s
[CV] END ..................n_neighbors=10, weights=distance; total
time=   0.0s
```

```
GridSearchCV(cv=2, estimator=KNeighborsClassifier(),
             param_grid={'n_neighbors': [3, 5, 7, 10],
                         'weights': ['uniform', 'distance']},
             verbose=2)
```

```
best_est = grid_search.best_estimator_
```

```
best_est
```

```
KNeighborsClassifier(n_neighbors=3, weights='distance')
```

```
acc = accuracy_score(y_test, y_pred)
```

```
acc
```

```
0.9931818181818182
```

```
rep = classification_report(y_test, y_pred)
```

```
print(rep)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        23
           1       1.00      1.00      1.00        21
           2       1.00      1.00      1.00        20
           3       1.00      1.00      1.00        26
           4       1.00      1.00      1.00        27
           5       1.00      1.00      1.00        17
           6       1.00      1.00      1.00        17
           7       1.00      1.00      1.00        14
           8       0.92      1.00      0.96        23
           9       1.00      1.00      1.00        20
          10       0.92      1.00      0.96        11
          11       1.00      1.00      1.00        21
          12       1.00      1.00      1.00        19
          13       1.00      0.96      0.98        24
          14       1.00      1.00      1.00        19
          15       1.00      1.00      1.00        17
          16       1.00      1.00      1.00        14
```

```
          17       1.00      1.00      1.00        23
          18       1.00      1.00      1.00        23
          19       1.00      1.00      1.00        23
          20       1.00      0.89      0.94        19
          21       1.00      1.00      1.00        19

    accuracy                           0.99       440
   macro avg       0.99      0.99      0.99       440
weighted avg       0.99      0.99      0.99       440
```

```
r2_score(y_test,y_pred)
```

```
0.984051571953857
```