Digit_Predication_ds_project

Objective

Main objective of this digit predication project include:

1. Classification Accuracy
2. Model Generalization
3. Efficiency
4. Continous improvement
5. Interpretabilty

Import Library

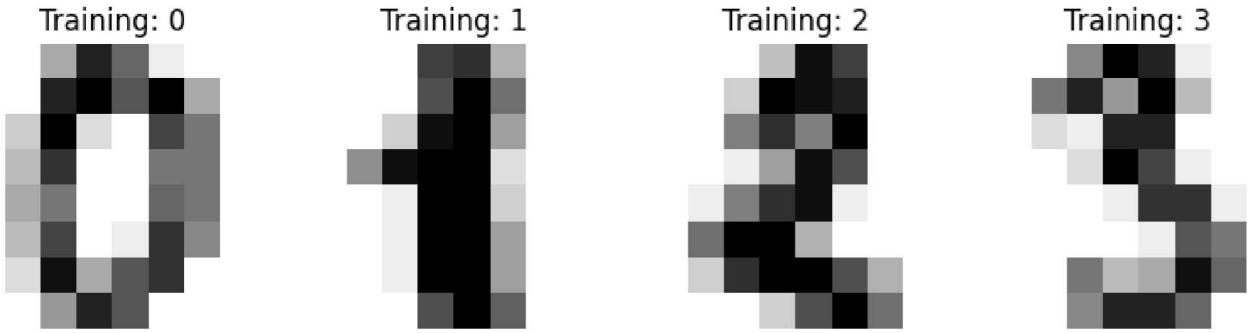
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Import data

```
from sklearn.datasets import load_digits

data = load_digits()

_, axes = plt.subplots(nrows=1, ncols=4, figsize=(10, 3))
for ax, image, label in zip(axes, data.images, data.target):
    ax.set_axis_off()
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation="nearest")
    ax.set_title("Training: %i" %label)
```



Double-click (or enter) to edit

Data Preprocessing

Flatten image

```
data.images.shape
```

```
(1797, 8, 8)
```

```
data.images[0]
```

```
array([[ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.],
       [ 0.,  0., 13., 15., 10., 15.,  5.,  0.],
       [ 0.,  3., 15.,  2.,  0., 11.,  8.,  0.],
       [ 0.,  4., 12.,  0.,  0.,  8.,  8.,  0.],
       [ 0.,  5.,  8.,  0.,  0.,  9.,  8.,  0.],
       [ 0.,  4., 11.,  0.,  1., 12.,  7.,  0.],
       [ 0.,  2., 14.,  5., 10., 12.,  0.,  0.],
       [ 0.,  0.,  6., 13., 10.,  0.,  0.,  0.]])
```

```
data.images[0].shape
```

```
(8, 8)
```

```
len(data.images)
```

```
1797
```

```
n_samples = len(data.images)
df = data.images.reshape(n_samples, -1)
```

```
df[0]
```

```
array([ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.,  0.,  0., 13., 15., 10.,
       15.,  5.,  0.,  0.,  3., 15.,  2.,  0., 11.,  8.,  0.,  0.,  4.,
       12.,  0.,  0.,  8.,  8.,  0.,  0.,  5.,  8.,  0.,  0.,  9.,  8.,
        0.,  0.,  4., 11.,  0.,  1., 12.,  7.,  0.,  0.,  2., 14.,  5.,
       10., 12.,  0.,  0.,  0.,  0.,  6., 13., 10.,  0.,  0.,  0.])
```

```
df[0].shape
```

```
(64,)
```

```
df.shape
```

```
(1797, 64)
```

Scaling Image Data

```
df.min()
```

```
0.0
```

```
df.max()
```

```
16.0
```

```
df = df/16
```

```
df.min()
```

```
0.0
```

```
df.max()
```

```
1.0
```

```
df[0]
```

```
array([0.    , 0.    , 0.3125, 0.8125, 0.5625, 0.0625, 0.    , 0.    ,
       0.    , 0.    , 0.8125, 0.9375, 0.625 , 0.9375, 0.3125, 0.    ,
       0.    , 0.1875, 0.9375, 0.125 , 0.    , 0.6875, 0.5   , 0.    ,
       0.    , 0.25  , 0.75  , 0.    , 0.    , 0.5   , 0.5   , 0.    ,
       0.    , 0.3125, 0.5   , 0.    , 0.    , 0.5625, 0.5   , 0.    ,
       0.    , 0.25  , 0.6875, 0.    , 0.0625, 0.75  , 0.4375, 0.    ,
       0.    , 0.125 , 0.875 , 0.3125, 0.625 , 0.75  , 0.    , 0.    ,
       0.    , 0.    , 0.375 , 0.8125, 0.625 , 0.    , 0.    , 0.    ])
```

Train Test Split Data

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(df, data.target, test_size=0.3)
```

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

```
((1257, 64), (540, 64), (1257,), (540,))
```

Random Forest Model

```
from sklearn.ensemble import RandomForestClassifier
```

```
rf = RandomForestClassifier()
```

```
rf.fit(X_train, y_train)
```

```
▸ RandomForestClassifier
```

Predict Test Data

```
y_pred = rf.predict(X_test)
```

```
y_pred
```

```
array([8, 7, 7, 8, 9, 2, 4, 0, 4, 8, 3, 3, 2, 2, 1, 5, 1, 1, 8, 1, 6, 6,
       7, 6, 5, 7, 7, 9, 9, 5, 8, 7, 7, 8, 3, 7, 5, 2, 5, 5, 4, 5, 1, 1,
       5, 4, 1, 7, 7, 9, 1, 4, 9, 7, 4, 2, 8, 5, 9, 3, 4, 3, 0, 2, 3, 2,
       2, 8, 3, 8, 4, 9, 6, 7, 5, 4, 7, 9, 6, 6, 9, 7, 7, 1, 2, 0, 4, 1,
       9, 6, 3, 9, 5, 3, 8, 8, 9, 8, 9, 3, 5, 6, 0, 8, 0, 4, 4, 8, 0, 8,
       3, 7, 6, 4, 2, 8, 0, 6, 3, 7, 0, 3, 3, 5, 6, 0, 9, 7, 2, 0, 5, 0,
       4, 1, 5, 8, 0, 2, 9, 8, 4, 3, 3, 4, 1, 8, 0, 2, 0, 8, 9, 6, 7, 5,
       9, 1, 5, 4, 7, 7, 5, 5, 1, 6, 5, 3, 7, 8, 1, 5, 6, 5, 0, 7, 6, 3,
       1, 1, 3, 8, 2, 5, 8, 1, 2, 1, 8, 3, 3, 1, 2, 9, 4, 0, 0, 9, 4, 2,
       1, 4, 7, 3, 7, 6, 1, 8, 7, 9, 8, 3, 5, 0, 0, 4, 1, 8, 7, 3, 3, 3,
       6, 1, 5, 4, 6, 7, 1, 7, 6, 9, 1, 1, 8, 4, 5, 8, 1, 6, 4, 2, 3, 4,
       9, 1, 1, 1, 9, 0, 2, 0, 9, 4, 8, 8, 9, 6, 1, 7, 6, 8, 4, 3, 3, 9,
       4, 4, 0, 9, 0, 2, 2, 3, 4, 6, 8, 2, 1, 5, 4, 5, 4, 6, 4, 3, 3, 3,
       4, 1, 3, 6, 2, 4, 7, 9, 7, 6, 7, 7, 4, 9, 4, 0, 8, 8, 5, 7, 7, 4,
       9, 8, 3, 9, 4, 2, 6, 4, 4, 1, 8, 2, 6, 6, 7, 7, 8, 9, 1, 6, 3, 8,
       8, 2, 4, 2, 9, 6, 2, 7, 7, 9, 0, 2, 9, 7, 3, 0, 6, 5, 0, 0, 8, 0,
       9, 6, 1, 3, 1, 6, 8, 3, 3, 9, 7, 7, 3, 5, 1, 7, 1, 6, 2, 7, 0, 5,
       5, 1, 6, 7, 7, 3, 0, 9, 7, 0, 4, 7, 9, 4, 8, 8, 1, 7, 5, 7, 3, 3,
       1, 3, 3, 5, 5, 2, 9, 7, 3, 0, 5, 9, 0, 5, 6, 1, 2, 2, 0, 0, 4, 4,
       2, 3, 7, 8, 8, 9, 9, 1, 3, 4, 7, 1, 2, 7, 7, 5, 9, 9, 6, 0, 2, 2,
       7, 7, 1, 6, 4, 3, 0, 4, 7, 6, 3, 8, 4, 0, 1, 8, 8, 1, 2, 1, 1, 0,
       8, 3, 9, 2, 0, 7, 3, 8, 3, 5, 5, 7, 5, 3, 3, 6, 9, 0, 7, 4, 4, 2,
       9, 2, 7, 8, 2, 5, 4, 5, 2, 3, 7, 9, 7, 1, 7, 4, 1, 8, 1, 0, 7, 4,
       9, 7, 5, 9, 9, 2, 4, 5, 2, 2, 6, 8, 7, 4, 6, 3, 6, 5, 2, 7, 3, 0,
       7, 5, 1, 5, 1, 0, 6, 5, 1, 6, 8, 2])
```

Model Accuracy

```
from sklearn.metrics import confusion_matrix, classification_report
```

```
confusion_matrix(y_test, y_pred)
```

```
array([[44,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 55,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0, 47,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0, 59,  0,  0,  0,  2,  0,  0],
       [ 0,  1,  0,  0, 57,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0, 48,  0,  0,  0,  0],
```

```
              [ 1,  0,  0,  0,  0,  0, 46,  0,  1,  0],
              [ 0,  0,  0,  0,  0,  0,  0, 65,  0,  1],
              [ 0,  1,  1,  0,  0,  1,  0,  0, 54,  0],
              [ 0,  0,  0,  0,  0,  1,  0,  4,  0, 51]])

print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.98	1.00	0.99	44
1	0.96	1.00	0.98	55
2	0.98	1.00	0.99	47
3	1.00	0.97	0.98	61
4	1.00	0.98	0.99	58
5	0.96	1.00	0.98	48
6	1.00	0.96	0.98	48
7	0.92	0.98	0.95	66
8	0.98	0.95	0.96	57
9	0.98	0.91	0.94	56
accuracy			0.97	540
macro avg	0.98	0.98	0.98	540
weighted avg	0.98	0.97	0.97	540