```
Start coding or generate with AI.
```

Title of Project

Handwritten Digit Prediction - Classification Analysis

Objective

The objective of this project is to build a machine learning model that can accurately predict handwritten digits.

Data Source

The dataset used for this project is the "digits" dataset from scikit-learn's built-in datasets. It contains images of handwritten digits, where each image is an 8x8 pixel matrix, and the target variable represents the actual digit value (0-9).

Import Library

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt
```

Import Data

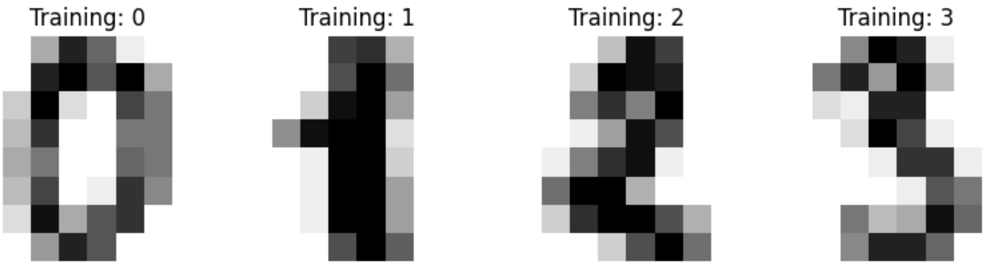
```
from sklearn.datasets import load_digits
```

Data Preprocessing

Flatten Image

```
df = load_digits()

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



```
df.images.shape
```

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

```
df.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.]])
```

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

```
(8, 8)
```

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

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

```
(64,)
```

```
data.shape
```

```
(1797, 64)
```

Scaling Data

```
data.min()
```

```
0.0
```

```
data.max()
```

```
1.0
```

```
data = data/16
```

```
data.min()
```

```
0.0
```

```
data.max()
```

```
0.0625
```

```
data[0]
```

```
array([0.        , 0.        , 0.01953125, 0.05078125, 0.03515625,
       0.00390625, 0.        , 0.        , 0.        , 0.        ,
       0.05078125, 0.05859375, 0.0390625 , 0.05859375, 0.01953125,
       0.        , 0.        , 0.01171875, 0.05859375, 0.0078125 ,
       0.        , 0.04296875, 0.03125   , 0.        , 0.        ,
       0.015625  , 0.046875  , 0.        , 0.        , 0.03125   ,
       0.03125   , 0.        , 0.        , 0.01953125, 0.03125   ,
       0.        , 0.        , 0.03515625, 0.03125   , 0.        ,
       0.        , 0.015625  , 0.04296875, 0.        , 0.00390625,
       0.046875  , 0.02734375, 0.        , 0.        , 0.0078125 ,
       0.0546875 , 0.01953125, 0.0390625 , 0.046875  , 0.        ,
       0.        , 0.        , 0.        , 0.0234375 , 0.05078125,
       0.0390625 , 0.        , 0.        , 0.        ])
```

Train test split Data

```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(data, df.target, test_size = 0.3)
xtrain.shape, xtest.shape, ytrain.shape, ytest.shape
```

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

Random Forest model

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
```

```
rf.fit(xtrain, ytrain)
```

```
▾ RandomForestClassifier ⓘ ?
RandomForestClassifier()
```

Predict Test Data

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

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

Model Accuracy

```
from sklearn.metrics import confusion_matrix, classification_report

confusion_matrix(ytest, y_pred)
```

```
array([[63,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 48,  0,  0,  0,  1,  0,  0,  0,  0],
       [ 0,  1, 47,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0, 61,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0, 52,  0,  0,  1,  0,  0],
       [ 0,  0,  0,  0,  0, 52,  0,  0,  0,  1],
       [ 0,  0,  0,  0,  0,  0, 53,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 51,  0,  0],
       [ 0,  3,  0,  0,  0,  0,  0,  0, 51,  0],
       [ 0,  0,  0,  1,  0,  1,  0,  1,  0, 52]])
```

Model Evaluation

```
print(classification_report(ytest, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        63
           1       0.92      0.98      0.95        49
           2       1.00      0.98      0.99        48
           3       0.98      1.00      0.99        61
           4       1.00      0.98      0.99        53
           5       0.96      0.98      0.97        53
           6       1.00      1.00      1.00        53
           7       0.96      1.00      0.98        51
           8       1.00      0.94      0.97        54
           9       0.98      0.95      0.96        55

    accuracy                           0.98       540
   macro avg       0.98      0.98      0.98       540
weighted avg       0.98      0.98      0.98       540
```

Prediction

```
from PIL import Image
import numpy as np

# Load the image and convert it to grayscale
image = Image.open('download.png').convert('L')

# Resize the image to 8x8 pixels (the same size as the dataset images)
image = image.resize((8, 8))
```

```
# Convert the image to a numpy array
new_image = np.array(image)

# Flatten the 8x8 image into a 1D array for prediction
new_image_flattened = new_image.flatten().reshape(1, -1)

# Scale the feature variables using the previously defined scaler
new_image_scaled = scaler.transform(new_image_flattened)

# Make a prediction using the model
predicted_digit = model.predict(new_image_scaled)

# Print the predicted digit
print("Predicted Digit:", predicted_digit[0])
```

```
Predicted Digit: 4
```