I am implementing the NUMBER RECOGNITION MODEL using the load_digits dataset From the sklearn

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.datasets import load_digits

digits = load_digits()                                  #imported the load_digits data to digits(variable)
print(digits.data[0])               #gives the 1-d array of that particular digit matrix
```

```
[ 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.]
```

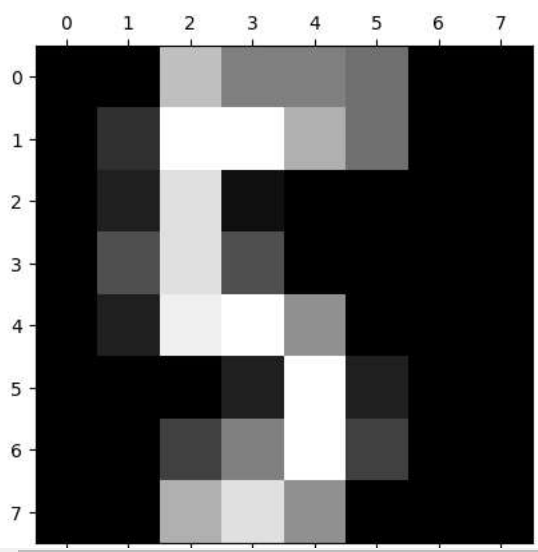
```
print(f"Size of the data(X) :  {digits.data.shape}")
print(f"Size of the Target (Y) : {digits.target.shape}")
```

```
Size of the data(X) :  (1797, 64)
Size of the Target (Y) : (1797,)
```

To get the image of the digits we can use the matplotlib

```
plt.gray()
plt.matshow(digits.images[25])          # prints the image of number which is at position at 25
plt.show()
```

```
<matplotlib.image.AxesImage at 0x79f67906bc10>
<Figure size 640x480 with 0 Axes>
```



Splitting the dataset into train_data and test_data

```
from sklearn.model_selection import train_test_split

x = digits.data
y = digits.target
x_train , x_test , y_train , y_test = train_test_split(x ,y , test_size = 0.2 , random_state = 42)
```

Importing the LogisticRegression and training the model

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression(multi_class = 'multinomial' , solver = 'lbfgs' , max_iter = 200)
model.fit(x_train  , y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:1247: FutureWarning: 'multi_class' was deprecated in version 1
  warnings.warn(
```

```
▾ LogisticRegression
LogisticRegression(max_iter=200, multi_class='multinomial')
```

Testing the data using the model

```
predicted_result = model.predict(x_test)
```

Calculating the Score of the Model

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

print(f"Accuracy Score of Testing data : {accuracy_score(y_test , predicted_result)*100}")
print(f"Confusion Matrix : {confusion_matrix(y_test , predicted_result)}")
print(f"Classification Report : {classification_report(y_test , predicted_result)}")
```

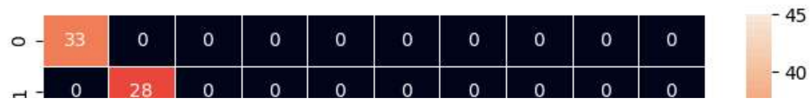
```
Accuracy Score of Testing data : 97.5
Confusion Matrix : [[33  0  0  0  0  0  0  0  0  0]
 [ 0 28  0  0  0  0  0  0  0  0]
 [ 0  0 33  0  0  0  0  0  0  0]
 [ 0  0  0 33  0  1  0  0  0  0]
 [ 0  1  0  0 45  0  0  0  0  0]
 [ 0  0  0  0  0 45  1  0  0  1]
 [ 0  0  0  0  0  1 34  0  0  0]
 [ 0  0  0  0  0  1  0 33  0  0]
 [ 0  0  0  0  0  1  0  0 29  0]
 [ 0  0  0  1  0  0  0  0  1 38]]
Classification Report :               precision    recall  f1-score   support

           0       1.00      1.00      1.00        33
           1       0.97      1.00      0.98        28
           2       1.00      1.00      1.00        33
           3       0.97      0.97      0.97        34
           4       1.00      0.98      0.99        46
           5       0.92      0.96      0.94        47
           6       0.97      0.97      0.97        35
           7       1.00      0.97      0.99        34
           8       0.97      0.97      0.97        30
           9       0.97      0.95      0.96        40

    accuracy                           0.97       360
   macro avg       0.98      0.98      0.98       360
weighted avg       0.98      0.97      0.98       360
```

Visualizing the confusion matrix using the seaborn

```
cn = confusion_matrix(y_test , predicted_result)
plt.figure(figsize = (8,5))
sns.heatmap(cn , annot = True , linewidth = 0.5)
plt.show()
```



printing the intercepts and coef of the of model each individual digit has its own intercept and coef(weights)

```
1 print(model.intercept_.shape)
2 print(model.coef_.shape)
```

```
(10,)
(10, 64)
```

Printing the intercept and weights of digit '0'.... In this way we can print intercepts and weights of number(0-9)


```
1 print(model.intercept_[0])
2 print(model.coef_[0])
```

```
0.005526295878635729
[ 0.00000000e+00 -5.15350463e-03 -1.40862905e-02  8.44001883e-02
  2.70552351e-02 -4.58767174e-02 -1.46596794e-01 -2.26156928e-02
 -2.57412474e-05 -7.46892821e-02 -2.22591127e-04  1.82950843e-01
  4.58725220e-04  7.96568438e-02 -5.51853265e-02 -2.07434750e-02
 -7.77469056e-04  5.06073047e-03  9.39548399e-02 -3.26803575e-02
 -3.54010758e-01  1.56936268e-01  5.09337964e-02 -5.36142765e-03
 -3.87306374e-04  1.45157802e-01  1.31507663e-01 -7.87094415e-02
 -4.02076554e-01  5.62993512e-02  5.56172804e-02 -1.09226900e-04
  0.00000000e+00  1.80701961e-01  4.81080008e-02 -1.20399999e-01
 -3.62043425e-01  5.11453634e-02  5.88826590e-02  0.00000000e+00
 -2.40709560e-04 -5.12007528e-02  2.61564221e-01 -1.98276068e-01
 -1.53055812e-01  9.08531843e-02  2.80404960e-02 -1.32430108e-04
 -8.31464092e-04 -1.38501536e-01  1.25362591e-01 -9.53550909e-02
  1.79252586e-01  1.04022589e-02 -1.26172454e-02 -4.33327778e-03
 -3.06386151e-06 -5.87699279e-03 -6.86919749e-02  2.12758661e-01
 -3.92647187e-02 -4.40627541e-02 -1.00284457e-02 -1.01487278e-02]
```