19. Data Science – ML – Logistic Regression – Multi class classification

Contents

19. Data Science – ML – Logistic Regression – Multi class classification

1. Logistic Regression

- ✓ Logistic regression comes under supervised Learning.
- ✓ It is a technique that is used to solve for classification problems.
- ✓ It is used for predicting the categorical dependent variable using a given set of independent variables.

2. Types of logistic regression

- ✓ Binary classification
  - o This is having two classes

- ✓ Multiclass classification
  - o This is having more than two classes

3. Binary classification examples

- ✓ In binary classification, there can be only two possible types of the dependent variables, such as,
  - o 0 or 1
  - o Pass or Fail
  - o Yes or No etc.

4. Multiclass classification examples

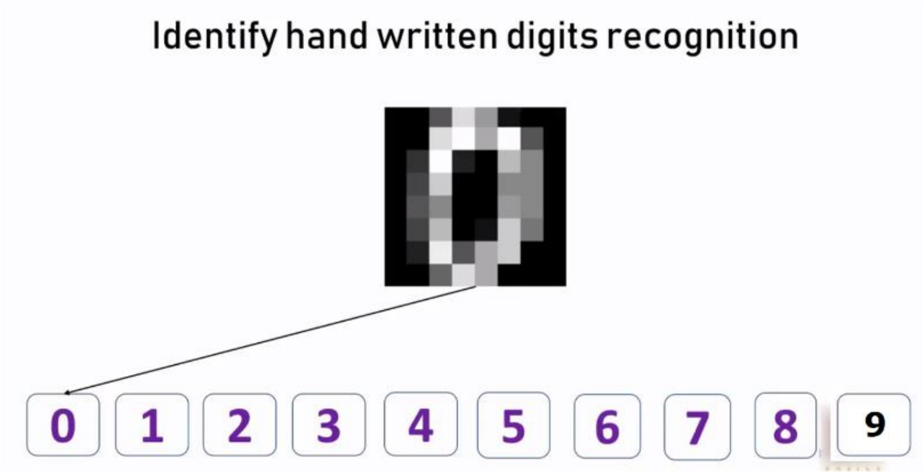
- ✓ In multiclass classification, there can be 3 or more possible unordered types of the dependent variable, such as,
  - o Ok, good, best
  - o Cat, dot, sheep etc

5. Practical example

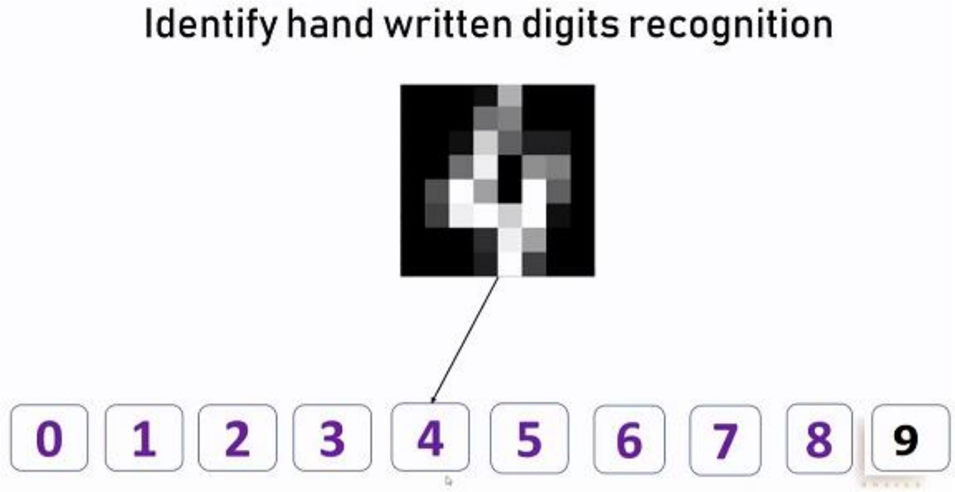
- ✓ Identify hand written digits
  - If image having ZERO number then output should be Zero.
  - If image having ONE number then output should be ONE.
  - If image having TWO number then output should be TWO.
  - ....
  - If image having NINE number then output should be NINE.

6. Problem

- ✓ Recognising the number.



Identify hand written digits recognition



Identify hand written digits recognition

7. Load dataset

- ✓ Sklearn library comes up with built-in datasets.
- ✓ One of the dataset names is called as digits dataset.

Program Name
Loading digits dataset
demo1.py

```
from sklearn.datasets import load_digits

# Loading the dataset
digits = load_digits()

print(len(digits.data))
```

Output

```
1797
```

Program Name
Loading digits dataset and checking attributes
demo2.py

```
from sklearn.datasets import load_digits

# Loading the dataset
digits = load_digits()

print(dir(digits))
```

Output

```
['DESCR', 'data', 'feature_names', 'frame', 'images', 'target',
'target_names']
```

Program Name

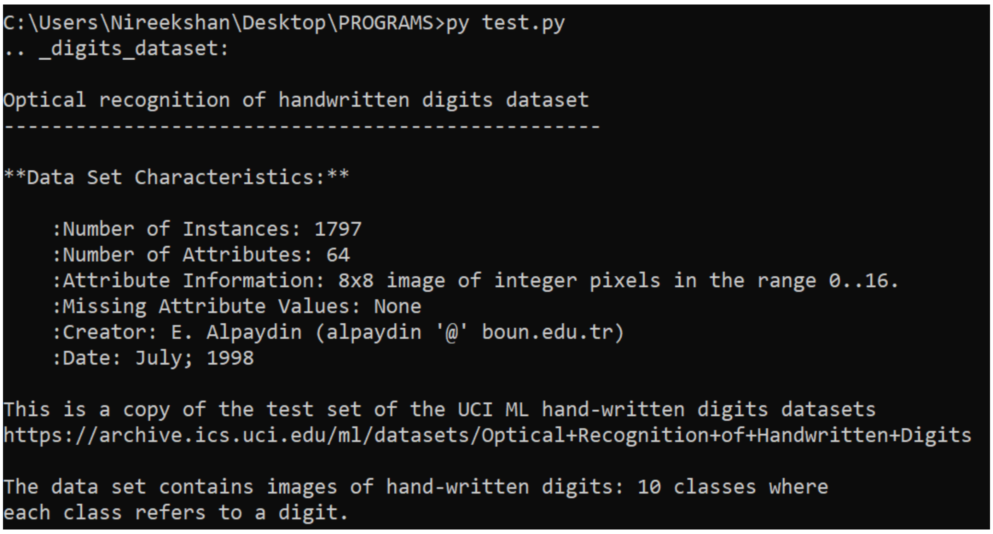
Loading digits dataset and accessing DESCR attribute
demo3.py

```
from sklearn.datasets import load_digits

# Loading the dataset
digits = load_digits()

print(digits.DESCR)
```

Output



```
C:\Users\Nireekshan\Desktop\PROGRAMS>py test.py
.. _digits_dataset:

Optical recognition of handwritten digits dataset
--------------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 1797
    :Number of Attributes: 64
    :Attribute Information: 8x8 image of integer pixels in the range 0..16.
    :Missing Attribute Values: None
    :Creator: E. Alpaydin (alpaydin '@' boun.edu.tr)
    :Date: July; 1998

This is a copy of the test set of the UCI ML hand-written digits datasets
https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits

The data set contains images of hand-written digits: 10 classes where
each class refers to a digit.
```

Program Name

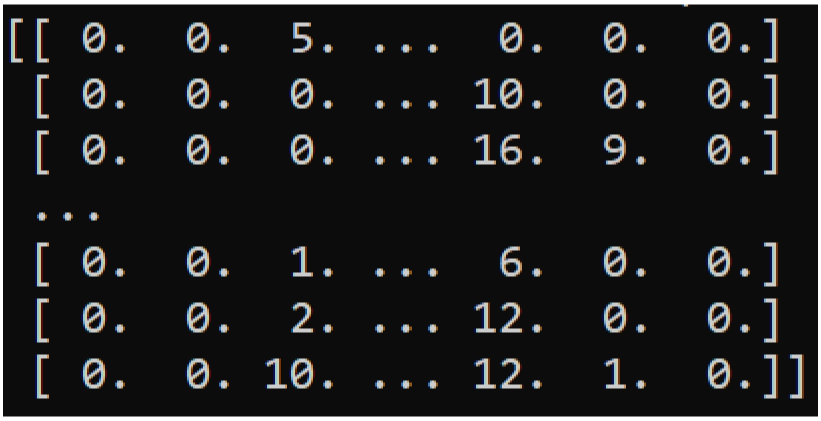
Loading digits dataset and accessing data attribute
demo4.py

```
from sklearn.datasets import load_digits

# Loading the dataset
digits = load_digits()

print(digits.data)
```

Output



```
[[ 0.   0.   5. ...   0.   0.   0.]
 [ 0.   0.   0. ...  10.   0.   0.]
 [ 0.   0.   0. ...  16.   9.   0.]
 ...
 [ 0.   0.   1. ...   6.   0.   0.]
 [ 0.   0.   2. ...  12.   0.   0.]
 [ 0.   0.  10. ...  12.   1.   0.]]
```

Program Name

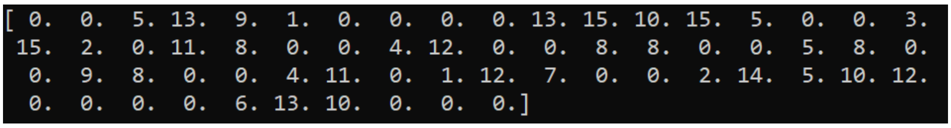
Loading digits dataset, checking the first value
demo5.py

```
from sklearn.datasets import load_digits

# Loading the dataset
digits = load_digits()

print(digits.data[0])
```

Output



```
[ 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.]
```

Program Name

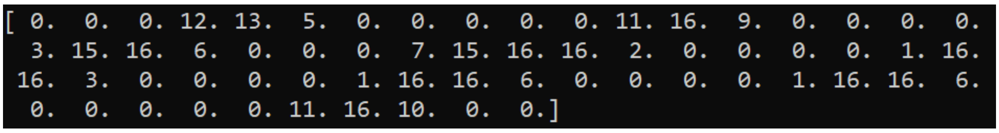
Loading digits dataset, checking the second value
demo6.py

```
from sklearn.datasets import load_digits

# Loading the dataset
digits = load_digits()

print(digits.data[1])
```

Output



```
[ 0.  0.  0. 12. 13.  5.  0.  0.  0.  0.  0. 11. 16.  9.  0.  0.  0.  0.
  3. 15. 16.  6.  0.  0.  0.  7. 15. 16. 16.  2.  0.  0.  0.  0.  1. 16.
 16.  3.  0.  0.  0.  0.  1. 16. 16.  6.  0.  0.  0.  0.  1. 16. 16.  6.
  0.  0.  0.  0.  0. 11. 16. 10.  0.  0.]
```

Program Name

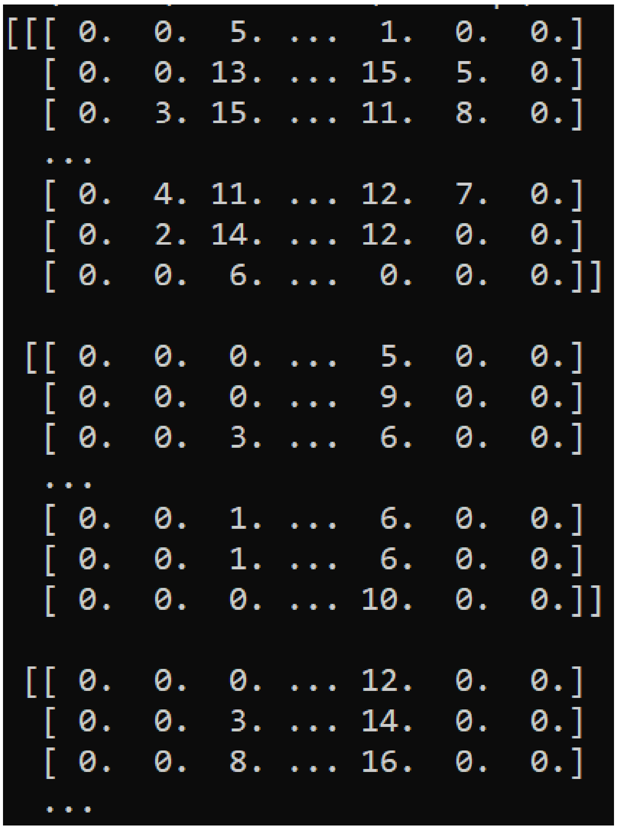
Loading digits dataset and accessing images attribute
demo7.py

```
from sklearn.datasets import load_digits

# Loading the dataset
digits = load_digits()

print(digits.images)
```

Output



```
[[[ 0.  0.  5. ...  1.  0.  0.]
  [ 0.  0. 13. ... 15.  5.  0.]
  [ 0.  3. 15. ... 11.  8.  0.]
  ...
  [ 0.  4. 11. ... 12.  7.  0.]
  [ 0.  2. 14. ... 12.  0.  0.]
  [ 0.  0.  6. ...  0.  0.  0.]]

 [[ 0.  0.  0. ...  5.  0.  0.]
  [ 0.  0.  0. ...  9.  0.  0.]
  [ 0.  0.  3. ...  6.  0.  0.]
  ...
  [ 0.  0.  1. ...  6.  0.  0.]
  [ 0.  0.  1. ...  6.  0.  0.]
  [ 0.  0.  0. ... 10.  0.  0.]]

 [[ 0.  0.  0. ... 12.  0.  0.]
  [ 0.  0.  3. ... 14.  0.  0.]
  [ 0.  0.  8. ... 16.  0.  0.]
  ...
```

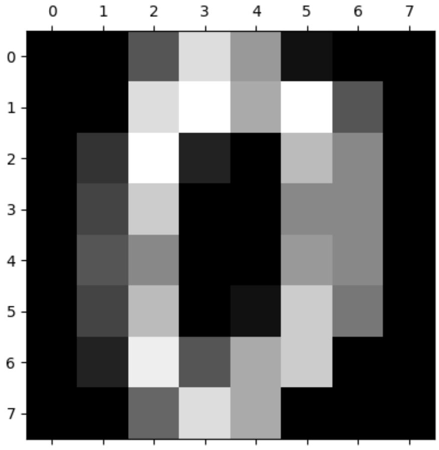
Program Name Loading digits dataset, displaying image
demo8.py

```
from sklearn.datasets import load_digits
import matplotlib.pyplot as plt

# Loading the dataset
digits = load_digits()

plt.matshow(digits.images[0])
plt.gray()
plt.show()
```

Output



Program Name

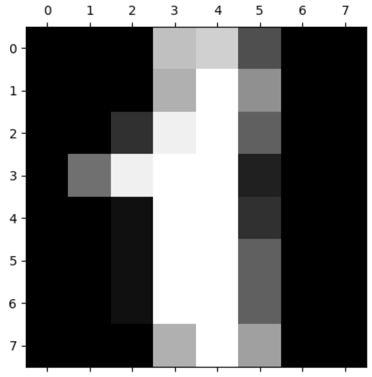
Loading digits dataset, displaying image
demo9.py

```
from sklearn.datasets import load_digits
import matplotlib.pyplot as plt

# Loading the dataset
digits = load_digits()

plt.matshow(digits.images[1])
plt.gray()
plt.show()
```

Output



Program Name Loading digits dataset, displaying image
demo10.py

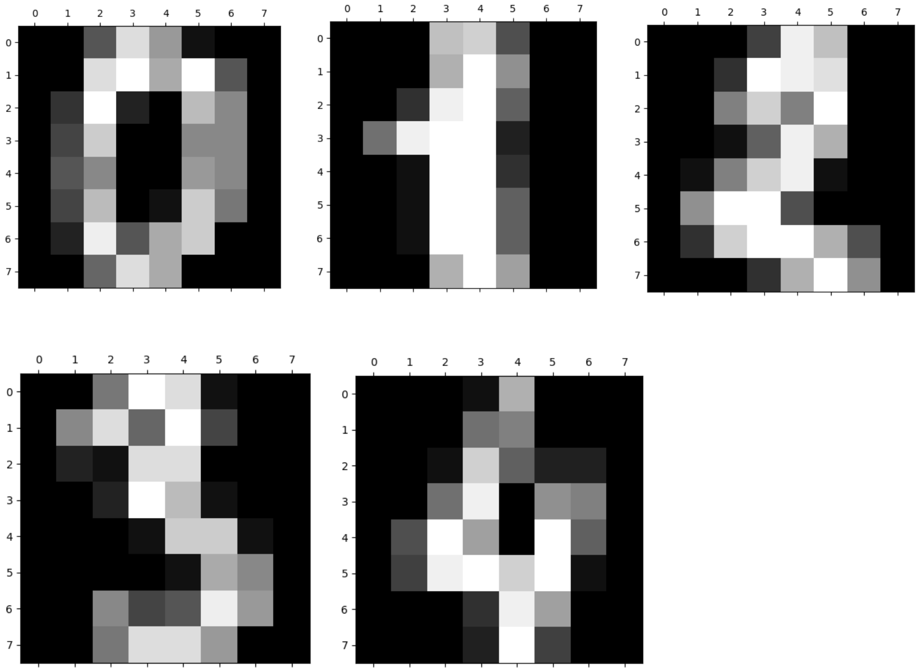
```
from sklearn.datasets import load_digits
import matplotlib.pyplot as plt

# Loading the dataset
digits = load_digits()

plt.gray()

for i in range(5):
        plt.matshow(digits.images[i])
        plt.show()
```

Output



Program Name

Loading digits dataset and accessing target attribute
demo11.py

```
from sklearn.datasets import load_digits

# Loading the dataset
digits = load_digits()

print(digits.target)
```

Output

```
[0 1 2 ... 8 9 8]
```

Program Name

Loading digits dataset and accessing target attribute
demo12.py

```
from sklearn.datasets import load_digits

# Loading the dataset
digits = load_digits()

print(digits.target[0:5])
```

Output

```
[0 1 2 3 4]
```

Program Name

Splitting the dataset
demo13.py

```
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split

# Loading the dataset
digits = load_digits()

print("Splitting the dataset")
X_train, X_test, y_train, y_test = train_test_split(digits.data,
digits.target, test_size = 0.2)
```

Output

```
Splitting the dataset
```

Program Name

Model creation
demo14.py

```
from sklearn.datasets import load_digits
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

# Loading the dataset
digits = load_digits()

X_train, X_test, y_train, y_test = train_test_split(digits.data,
digits.target, test_size=0.2)

print("Model creation")
model = LogisticRegression(solver = 'lbfgs', max_iter = 3000)

model.fit(X_train, y_train)
```

Output

```
Model creation
```

Note

- ✓ lbfgs stand for: "Limited-memory Broyden–Fletcher–Goldfarb–Shanno Algorithm".
- ✓ It is one of the solvers' algorithms provided by Scikit-Learn Library.

Program Name

Model score
demo15.py

```
from sklearn.datasets import load_digits
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

# Loading the dataset
digits = load_digits()

X_train, X_test, y_train, y_test = train_test_split(digits.data,
digits.target, test_size=0.2)

model = LogisticRegression(solver = 'lbfgs', max_iter = 3000)

model.fit(X_train, y_train)
print(model.score(X_test, y_test))
```

Output

```
0.9611111111111111
```

Program Name

Model prediction
demo16.py

```
from sklearn.datasets import load_digits
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

# Loading the dataset
digits = load_digits()

X_train, X_test, y_train, y_test = train_test_split(digits.data,
digits.target, test_size = 0.2)

model = LogisticRegression(solver = 'lbfgs', max_iter = 3000)

model.fit(X_train, y_train)

print(model.predict([digits.data[6]]))
```

Output

```
[6]
```

Program Name

Model prediction
demo17.py

```
from sklearn.datasets import load_digits
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

# Loading the dataset
digits = load_digits()

X_train, X_test, y_train, y_test = train_test_split(digits.data,
digits.target, test_size=0.2)

model = LogisticRegression(solver = 'lbfgs', max_iter = 3000)

model.fit(X_train, y_train)

print(model.predict([digits.data[9]]))
```

Output

```
[9]
```

Program Name

Model prediction
demo18.py

```
from sklearn.datasets import load_digits
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

# Loading the dataset
digits = load_digits()

X_train, X_test, y_train, y_test = train_test_split(digits.data,
digits.target, test_size=0.2)

model = LogisticRegression(solver = 'lbfgs', max_iter = 3000)

model.fit(X_train, y_train)

print(model.predict([digits.data[20]]))
```

Output

```
[0]
```

Program Name

Model prediction
demo19.py

```
from sklearn.datasets import load_digits
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

# Loading the dataset
digits = load_digits()

X_train, X_test, y_train, y_test = train_test_split(digits.data,
digits.target, test_size=0.2)

model = LogisticRegression(solver = 'lbfgs', max_iter = 3000)

model.fit(X_train, y_train)

print(model.predict(digits.data[0:5]))
```

Output

```
[0 1 2 3 4]
```