DML MINI PROJECT

AIM: Handwritten digit recognition using mnist dataset

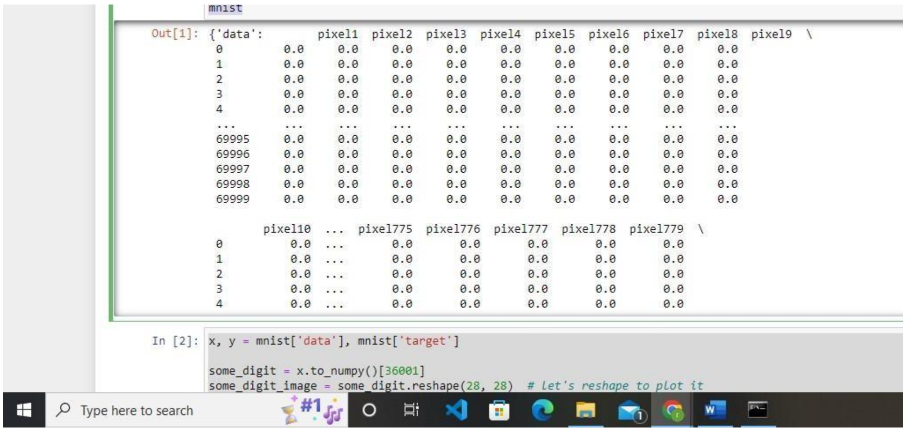
What is MNIST?

1. Set of 70,000 small images of digits handwritten by high school students and employees of the US causes Bureau.
2. All images are labeled with the respective digit they represent.
3. MNIST is the hello world of machine learning. Every time a data scientist or machine learning engineer makes a new algorithm for classification, they would always first check its performance on the MNIST dataset.
4. There are 70,000 images and each image has 28*28 = 784 features.
5. Each image is 28*28 pixels and each feature simply represents one-pixel intensity from 0 to 255. If the intensity is 0, it means that the pixel is white and if it is 255, it means it is black.

CODE:

```
from sklearn.datasets import

fetch_openml import matplotlib

import matplotlib.pyplot as plt

import numpy as np

from          sklearn.linear_model

              import   LogisticRegression   from

sklearn.model_selection import

              cross_val_score       mnist

              =       fetch_openml('mnist_784')

mnist
```

OUTPUT:



```
mnist
```

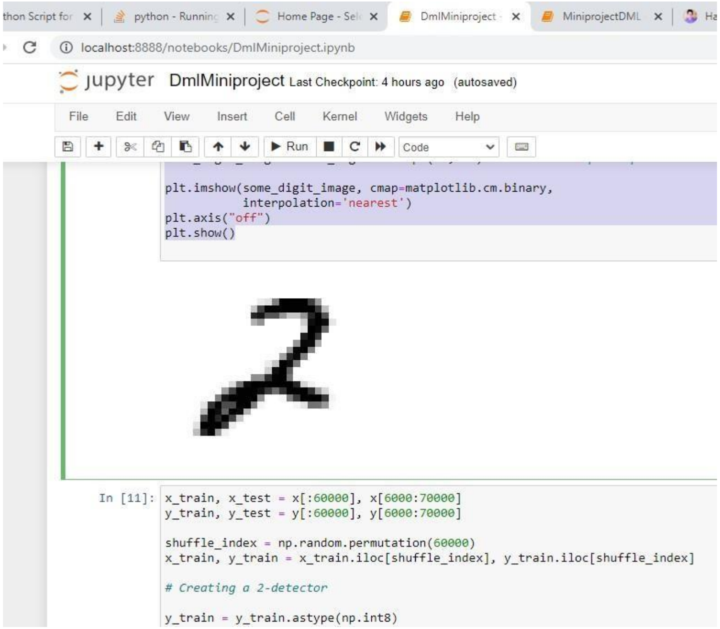
```
Out[1]: {'data':        pixel1  pixel2  pixel3  pixel4  pixel5  pixel6  pixel7  pixel8  pixel9  \
         0         0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
         1         0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
         2         0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
         3         0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
         4         0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
         ...       ...     ...     ...     ...     ...     ...     ...     ...     ...
         69995     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
         69996     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
         69997     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
         69998     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
         69999     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0

                pixel10  ...  pixel775  pixel776  pixel777  pixel778  pixel779  \
         0          0.0  ...       0.0       0.0       0.0       0.0       0.0
         1          0.0  ...       0.0       0.0       0.0       0.0       0.0
         2          0.0  ...       0.0       0.0       0.0       0.0       0.0
         3          0.0  ...       0.0       0.0       0.0       0.0       0.0
         4          0.0  ...       0.0       0.0       0.0       0.0       0.0
```

```
In [2]: x, y = mnist['data'], mnist['target']

        some_digit = x.to_numpy()[36001]
        some_digit_image = some_digit.reshape(28, 28)  # let's reshape to plot it
```

CODE:

```
x, y = mnist['data'], mnist['target']
some_digit =
x.to_numpy()[36001]
some_digit_image = some_digit.reshape(28, 28) # let's reshape to plot it
plt.imshow(some_digit_image, cmap=matplotlib.cm.binary,
interpolation='nearest') plt.axis("off") plt.show()
```

OUTPUT:



```
plt.imshow(some_digit_image, cmap=matplotlib.cm.binary,
           interpolation='nearest')
plt.axis("off")
plt.show()
```

In [11]:

```
x_train, x_test = x[:60000], x[6000:70000]
y_train, y_test = y[:60000], y[6000:70000]

shuffle_index = np.random.permutation(60000)
x_train, y_train = x_train.iloc[shuffle_index], y_train.iloc[shuffle_index]

# Creating a 2-detector

y_train = y_train.astype(np.int8)
```

CODE

```
x_train, x_test = x[:60000],
x[6000:70000]y_train, y_test =
y[:60000], y[6000:70000]
shuffle_index =
np.random.permutation(60000)
x_train, y_train = x_train.iloc[shuffle_index], y_train.iloc[shuffle_index]

# Creating a 2-
detectory_train =
y_train.astype(np.int
8)y_test =
y_test.astype(np.int8)

y_train_2 = (y_train

== 2)y_test_2 =

(y_test == 2)

clf =

LogisticRegression(tol=0.1)

clf.fit(x_train, y_train_2)
```

OUTPUT

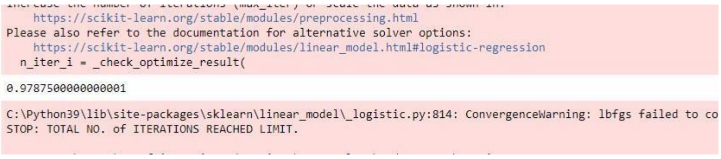
LogisticRegression(tol=0.1)

CODE:

```
a = cross_val_score(clf, x_train, y_train_2, cv=3,
scoring="accuracy")print(a.mean())
```

In [15]:

OUTPUT:



```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(

0.9787500000000001

C:\Python39\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to co
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```