Mnist database : The MNIST handwritten digit classification problem is a standard dataset used in computer vision and deep learning. Although the dataset is effectively solved, it can be used as the basis for learning and practicing how to develop, evaluate, and use convolutional deep learning neural networks for image classification from scratch. This includes how to develop a robust test harness for estimating the performance of the model, how to explore improvements to the model, and how to save the model and later load it to make predictions on new data. In this tutorial, you will discover how to develop a convolutional neural network for handwritten digit classification from scratch

Sklearn : Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

Numpy : NumPy is the fundamental package for scientific computing in Python

Pandas : Pandas is an open source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named Numpy, which provides support for multi-dimensional arrays. As one of the most popular data wrangling packages, Pandas works well with many other data science modules inside the Python ecosystem, and is typically included in every Python distribution, from those that come with your operating system to commercial vendor distributions like ActiveState's ActivePython.

seaborn : Seaborn is a data visualization library built on top of matplotlib and closely integrated with pandas data structures in Python.

matplotlib : Matplotlib is a plotting library available for the Python programming language as a component of NumPy, a big data numerical handling resource. Matplotlib uses an object oriented API to embed plots in Python applications.

Importing librarys

In [3]:

```
import numpy as np
import pandas as pd
import seaborn as sn
import matplotlib.pyplot as plt
%matplotlib inline
```

Loading data

In [ ]:

```
from sklearn.datasets import fetch_openml
mnist = fetch_openml('mnist_784')
```

creating dataframe of mnist data

In [29]:

```
dir(mnist)
```

Out[29]:

```
['DESCR',
 'categories',
 'data',
 'details',
 'feature_names',
 'frame',
 'target',
 'target_names',
 'url']
```

In [5]:

```
import pandas as pd
pixels = pd.DataFrame(mnist.data)
labels = pd.DataFrame(mnist.target)
```

printing first value of data

In [6]:

```
pixels.loc[0].values
```

Out[6]:

```
array([  0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   3.,  18.,
        18.,  18., 126., 136., 175.,  26., 166., 255., 247., 127.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
        30.,  36.,  94., 154., 170., 253., 253., 253., 253., 253., 225.,
       172., 253., 242., 195.,  64.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,  49., 238., 253., 253., 253., 253.,
       253., 253., 253., 253., 251.,  93.,  82.,  82.,  56.,  39.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
        18., 219., 253., 253., 253., 253., 253., 198., 182., 247., 241.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,  80., 156., 107., 253.,
       253., 205.,  11.,   0.,  43., 154.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,  14.,   1., 154., 253.,  90.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
       139., 253., 190.,   2.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,  11., 190., 253.,  70.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,  35., 241., 225., 160., 108.,   1.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,  81., 240.,
       253., 253., 119.,  25.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,  45., 186., 253., 253., 150.,  27.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,  16.,  93., 252., 253., 187.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0., 249., 253.,
       249.,  64.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,  46., 130., 183., 253., 253., 207.,   2.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,  39., 148., 229., 253., 253., 253.,
       250., 182.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,  24., 114.,
       221., 253., 253., 253., 253., 201.,  78.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,  23.,  66., 213., 253., 253., 253., 253., 198.,  81.,
         2.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,    0.,    0.,    0.,    0.,    0.,   18.,  171.,  219.,  253.,  253.,
       253.,  253.,  195.,   80.,    9.,    0.,    0.,    0.,    0.,    0.,    0.,
         0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,   55.,
       172.,  226.,  253.,  253.,  253.,  253.,  244.,  133.,   11.,    0.,    0.,
         0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
         0.,    0.,    0.,    0.,    0.,  136.,  253.,  253.,  253.,  212.,  135.,
       132.,   16.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
         0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
         0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
         0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
         0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
         0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
         0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
         0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
         0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
         0.,    0.,    0.])
```

In [7]:

```
labels.loc[0].values
```

Out[7]:

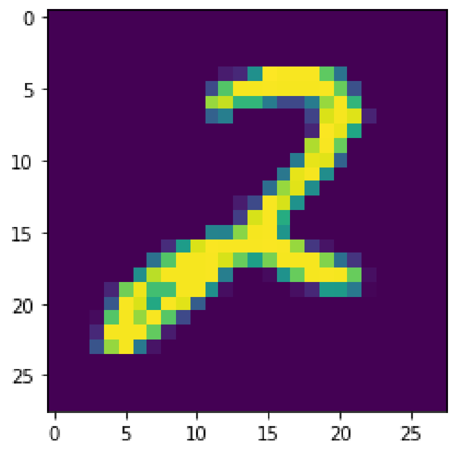
```
['5']
Categories (10, object): ['0', '1', '2', '3', ..., '6', '7', '8', '9']
```

In [37]:

```
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,
           interpolation='nearest')
plt.show()
```



In [8]:

```
len(pixels)
```

Out[8]:

```
70000
```

In [9]:

```
len(labels)
```

Out[9]:

```
70000
```

In [10]:

```
labels.head()
```

Out[10]:

	class
0	5
1	0
2	4
3	1
4	9

In [11]:

```
pixels.head()
```

Out[11]:

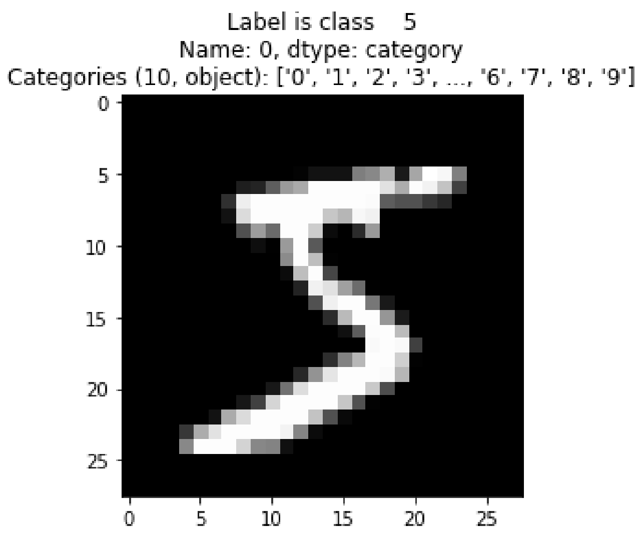
	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	pixel10	...	pixel775	pixel776	pixel777	pixel778	pixel77
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	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	0.0	...	0.0	0.0	0.0	0.0	0.

5 rows × 784 columns

printing data and target value

In [12]:

```
label = labels.loc[0]
pixel = pixels.loc[0]
pixel = np.array(pixel, dtype='uint8')
pixel = pixel.reshape((28,28))
plt.title('Label is {label}'.format(label=label))
plt.imshow(pixel, cmap='gray')
plt.show()
```



splitting the data into training data and testing data

In [13]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(mnist.data, mnist.target, test_size=
1/7.0)
```

Using RandomForestClassifier to train the model

In [14]:

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(X_train, y_train)
```

Out[14]:

```
▾ RandomForestClassifier
RandomForestClassifier()
```

measuring the score and accuracy of model

In [16]:

```
model.score(X_test, y_test)
```

Out[16]:

```
0.9697
```

In [17]:

```
from sklearn.metrics import accuracy_score
accuracy_score(y_pred, y_test)
```

Out[17]:

```
0.9697
```

In [25]:

```
model.predict([pixels.loc[3].values]) #predicting the value of data at index 3
```

```
C:\Users\acer\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not ha
ve valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```

Out[25]:

```
array(['1'], dtype=object)
```

calculating cofusion matrix

In [19]:

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

In [21]:

```
from sklearn.metrics import confusion_matrix
```

In [22]:

```
cm = confusion_matrix(y_test, y_pred)
cm
```

Out[22]:

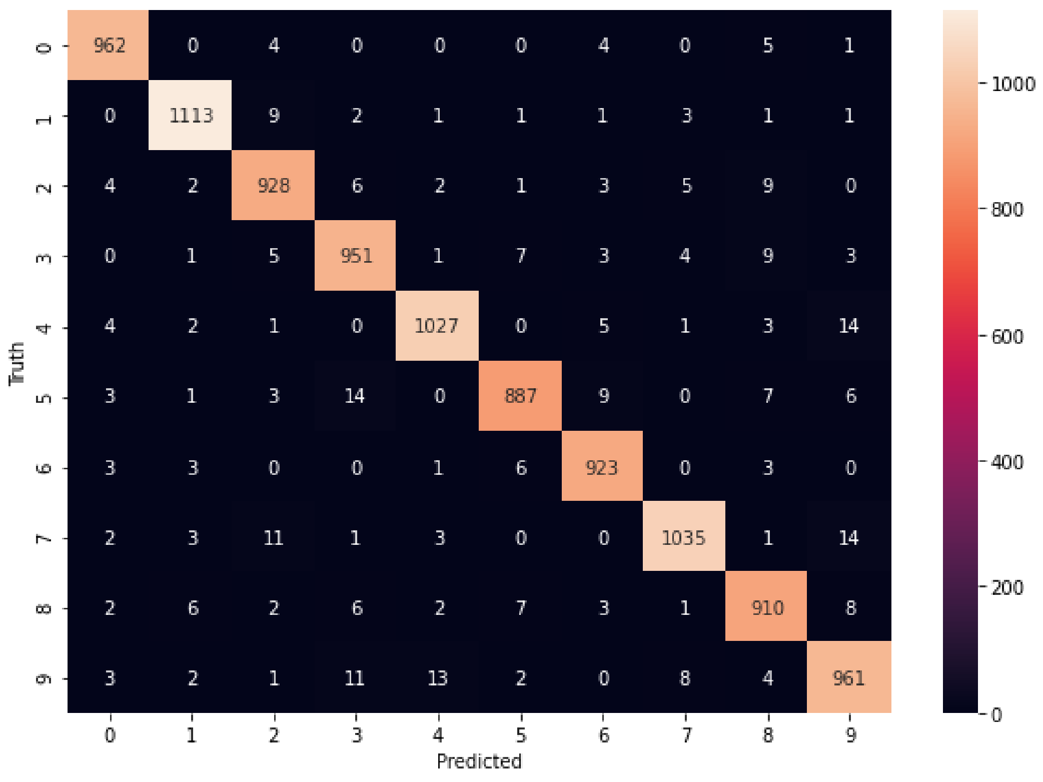
```
array([[ 962,    0,    4,    0,    0,    0,    4,    0,    5,    1],
       [   0, 1113,    9,    2,    1,    1,    1,    3,    1,    1],
       [   4,    2,  928,    6,    2,    1,    3,    5,    9,    0],
       [   0,    1,    5,  951,    1,    7,    3,    4,    9,    3],
       [   4,    2,    1,    0, 1027,    0,    5,    1,    3,   14],
       [   3,    1,    3,   14,    0,  887,    9,    0,    7,    6],
       [    3,     3,     0,     0,     1,     6,   923,     0,     3,     0],
       [    2,     3,    11,     1,     3,     0,     0,  1035,     1,    14],
       [    2,     6,     2,     6,     2,     7,     3,     1,   910,     8],
       [    3,     2,     1,    11,    13,     2,     0,     8,     4,   961]],
      dtype=int64)
```

In [23]:

```
plt.figure(figsize=(10,7))
sn.heatmap(cm,annot=True, fmt="d")
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Out[23]:

```
Text(69.0, 0.5, 'Truth')
```


Truth
Predicted