In [1]:
```
import numpy as np
import tensorflow as tf
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
from matplotlib import pyplot as plt

(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_da
print(np.shape(x_train))

def vectorize(_image):
    return np.reshape(_image, (-1,1))

vec_x_train = np.squeeze(np.array([vectorize(m) for m in x_train]))
vec_x_test = np.squeeze(np.array([vectorize(m) for m in x_test]))

# generate an instance of the logistic regression class model with mul
model = LogisticRegression(solver='saga', tol=0.01, multi_class='multi
```

```
(60000, 28, 28)
```

In [2]:
```
# use model.fit(some arguments) to fit the model to the data
model.fit(vec_x_train, y_train)
```

Out[2]:
```
LogisticRegression(multi_class='multinomial', solver='saga', tol=0.01
)
```

In [4]:
```
from sklearn.metrics import classification_report

### compute the accuracy and print a classification report
# use model.predict(some arguments) to predict the class of the test d
y_head_test = model.predict(vec_x_test)

# use classification_report(some arguments) to print accuracy
print(classification_report(y_test, y_head_test))
```

```
              precision    recall  f1-score   support

           0       0.95      0.98      0.97       980
           1       0.96      0.98      0.97      1135
           2       0.93      0.90      0.91      1032
           3       0.90      0.91      0.91      1010
           4       0.93      0.93      0.93       982
           5       0.91      0.86      0.88       892
           6       0.95      0.95      0.95       958
           7       0.93      0.92      0.93      1028
           8       0.88      0.88      0.88       974
           9       0.91      0.91      0.91      1009

    accuracy                           0.93     10000
   macro avg       0.92      0.92      0.92     10000
weighted avg       0.93      0.93      0.93     10000
```

In [5]:
```
from sklearn.preprocessing import OneHotEncoder

# convert the 10 classes to one hot encoding
one_hot = OneHotEncoder()
Y_train = one_hot.fit_transform(y_train.reshape(-1,1)).toarray()
Y_test = one_hot.fit_transform(y_test.reshape(-1,1)).toarray()
print(np.shape(Y_train))
```

```
(60000, 10)
```

In [30]:
```
import keras
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metri

# use model.fit(some arguments) to fit the model to the data, specify
model.fit(vec_x_train, Y_train, batch_size = 5, epochs = 10)
```

```
Epoch 1/10
60000/60000 [==============================] - 15s 255us/step - loss:
14.3045 - accuracy: 0.8502
Epoch 2/10
60000/60000 [==============================] - 15s 252us/step - loss:
13.3905 - accuracy: 0.8782
Epoch 3/10
60000/60000 [==============================] - 15s 256us/step - loss:
13.4908 - accuracy: 0.8827
Epoch 4/10
60000/60000 [==============================] - 16s 267us/step - loss:
13.2772 - accuracy: 0.8834
Epoch 5/10
60000/60000 [==============================] - 17s 287us/step - loss:
13.3067 - accuracy: 0.8868
Epoch 6/10
60000/60000 [==============================] - 18s 308us/step - loss:
13.2110 - accuracy: 0.8868
Epoch 7/10
60000/60000 [==============================] - 19s 317us/step - loss:
13.3363 - accuracy: 0.8880
Epoch 8/10
60000/60000 [==============================] - 20s 339us/step - loss:
13.3057 - accuracy: 0.8898
Epoch 9/10
60000/60000 [==============================] - 17s 277us/step - loss:
13.3966 - accuracy: 0.8881
Epoch 10/10
60000/60000 [==============================] - 15s 258us/step - loss:
13.3410 - accuracy: 0.8889
```

Out[30]: <keras.callbacks.callbacks.History at 0x7fbe5c951350>

In [32]:
```
from sklearn.metrics import classification_report

### compute the accuracy and print a classification report
# use model.predict(some arguments) to predict the class of the test d
# use classification_report(some arguments) to print accuracy
Y_head_test = model.predict_classes(vec_x_test)
print(classification_report(y_test, Y_head_test))
```

```
              precision    recall  f1-score   support

           0       0.95      0.97      0.96       980
           1       0.98      0.90      0.94      1135
           2       0.78      0.93      0.85      1032
           3       0.75      0.93      0.83      1010
           4       0.95      0.82      0.88       982
           5       0.97      0.63      0.76       892
           6       0.93      0.94      0.93       958
           7       0.94      0.89      0.91      1028
           8       0.85      0.81      0.83       974
           9       0.80      0.92      0.86      1009

    accuracy                           0.88     10000
   macro avg       0.89      0.87      0.87     10000
weighted avg       0.89      0.88      0.88     10000
```

```
Logistic regression model is more accurate and runs faster.
```