به نام خدا

دانشگاه صنعتی شاهرود

Shahrood University of Technology

دانشکده مهندسی کامپیوتر و فناوری اطلاعات

عنوان تمرین: دسته بندی به کمک شبکه عصبی

استاد: دکتر مرضیه رحیمی

نویسنده: مریم درویشیان

تاریخ: 1401/12/25

1- مجموعه داده MNIST را معرفی کنید؟

پایگاه داده MNIST یک پایگاه داده بزرگ از ارقام دست نویس است که معمولاً برای آموزش سیستم های مختلف پردازش تصویر استفاده می‌شود. این پایگاه داده همچنین به طور گسترده‌ای برای آموزش و آزمایش در زمینه یادگیری ماشین استفاده می‌شود. این مجموعه با ادغام کردن داده‌های مجموعه‌های اصلی NIST ایجاد شده‌است. از آنجایی که مجموعه داده‌های آموزشی NIST از کارمندان اداره سرشماری آمریکا، و مجموعه داده آزمایشی از دانش آموزان دبیرستانی آمریکایی جمع آوری شده بودند، سازندگان این دیتابیس گمان می‌کردند که این مجموعه برای آزمایش‌های یادگیری ماشین مناسب نیست. علاوه بر این، تصاویر سیاه و سفید مجموعهNIST نرمال سازی شدند تا در فضای ۲۸X۲۸ پیکسل قرار بگیرند و همچنین عملیات هموار سازی که باعث پدید آمدن تصاویر طیف خاکستری شد. پایگاه داده MNIST شامل ۶۰ هزار تصویر آموزشی و ۱۰ هزار تصویر آزمایشی است. نیمی از مجموعه آموزشی و نیمی از مجموعه تست از مجموعه داده آموزشی NIST گرفته شده است. سازندگان اصلی پایگاه داده، فهرستی از برخی روش های آزمایش شده بر روی آن را ساخته‌اند. در مقاله اصلی خود، آنها توسط روش ماشین بردار پشتیبان، به نرخ خطای ۰.۸٪ دست یافته‌اند. مجموعه داده گسترش یافته‌ای شبیه به MNIST به نام EMNIST نیز در سال ۲۰۱۷ منتشر شد، که شامل ۲۴۰ هزار تصویر آموزشی و ۴۰ هزار تصویر آزمایشی از ارقام و کاراکترهای دست نویس است.

2- چند نرون در لایه خروجی وجود دارد؟10 نرون

3- مراحل آموزش و تست را انجام دهید و دقت هر بخش را گزارش کنیدتحلیل خود را از دقتهای بدست آمده بنویسید

```
import pandas as pd
import sklearn.model_selection as ms
import sklearn.datasets as dt
import sklearn.neural_network as nn
from sklearn.metrics import confusion_matrix,precision_score,recall_score,f1_score

mnist=dt.load_digits()
mnist.keys()
pd.DataFrame(mnist.data).head()
X=mnist.data
Y=mnist.target
xtr,xte,ytr,yte=ms.train_test_split(X,Y,train_size=0.8,random_state=1)
MLP=nn.MLPClassifier(hidden_layer_sizes=(50,),activation='logistic',alpha=0.1,solver='sgd',max_iter=1000,verbose=10,random_state=1,learning_rate_init=0.1,learn
MLP.fit(xtr,ytr)
trACC=MLP.score(xtr,ytr)
teACC=MLP.score(xte,yte)
print('Train accuracy:', trACC)
print('Test accuracy:', teACC)
```

```
Iteration 4, loss = 0.29394593
Iteration 5, loss = 0.21031134
Iteration 6, loss = 0.17682421
Iteration 7, loss = 0.14287186
Iteration 8, loss = 0.12820629
Iteration 9, loss = 0.11229184
Iteration 10, loss = 0.10492172
Iteration 11, loss = 0.09847796
Iteration 12, loss = 0.09310076
Iteration 13, loss = 0.09312003
Iteration 14, loss = 0.09220696
Iteration 15, loss = 0.08599993
Iteration 16, loss = 0.08230878
Iteration 17, loss = 0.08079831
Iteration 18, loss = 0.07795512
Iteration 19, loss = 0.07683380
Iteration 20, loss = 0.07591750
Iteration 21, loss = 0.07357728
Iteration 22, loss = 0.07469043
Iteration 23, loss = 0.07231482
Iteration 24, loss = 0.07106737
Iteration 25, loss = 0.07122565
...
Iteration 255, loss = 0.05125513
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Learning rate too small. Stopping.
Train accuracy: 1.0
Test accuracy: 0.9861111111111112
```

داده های Train با MLP classifier با دقت صد درصد اموزش می بینند و با دقت 0.9861 داده های تست را به درستی تشخیص و طبقه میکند.همین طور مقادیر f1 score ,recall , Cofusion matrix نیز براورد شده است.

```python
y_pred_train=MLP.predict(xtr)
y_pred_test=MLP.predict(xte)

confusion_matrix_train=confusion_matrix(ytr,y_pred_train)
confusion_matrix_test=confusion_matrix(yte,y_pred_test)

print('Cofusion matrix train:',confusion_matrix_train)
print('Cofusion matrix test:',confusion_matrix_test)
```

✓ 5.5s Python

```
Cofusion matrix train: [[135   0   0   0   0   0   0   0   0   0]
 [  0 147   0   0   0   0   0   0   0   0]
 [  0   0 141   0   0   0   0   0   0   0]
 [  0   0   0 142   0   0   0   0   0   0]
 [  0   0   0   0 143   0   0   0   0   0]
 [  0   0   0   0   0 152   0   0   0   0]
 [  0   0   0   0   0   0 144   0   0   0]
 [  0   0   0   0   0   0   0 142   0   0]
 [  0   0   0   0   0   0   0   0 145   0]
 [  0   0   0   0   0   0   0   0   0 146]]
Cofusion matrix test: [[42  0  0  0  1  0  0  0  0  0]
 [ 0 35  0  0  0  0  0  0  0  0]
 [ 0  0 36  0  0  0  0  0  0  0]
 [ 0  0  0 41  0  0  0  0  0  0]
 [ 0  0  0  0 38  0  0  0  0  0]
 [ 0  0  0  0  0 30  0  0  0  0]
 [ 0  0  0  0  0  0 37  0  0  0]
 [ 0  0  0  0  0  0  0 36  0  1]
 [ 0  0  0  0  0  1  0  0 27  1]
 [ 0  0  0  0  0  1  0  0  0 33]]
```

```python
recall_train=recall_score(ytr,y_pred_train,average=None)
recall_test=recall_score(yte,y_pred_test,average=None)

print('Train recall:',recall_train)
print('Test recall:',recall_test)
```

✓ 1.5s Pytho

```
Train recall: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
Test recall: [0.97674419 1.         1.         1.         1.         1.
 1.         0.97297297 0.93103448 0.97058824]
```

```
f1_score_train=f1_score(ytr,y_pred_train,average=None)
f1_score_test=f1_score(yte,y_pred_test,average=None)

print('Train f1score:',f1_score_train)
print('Test f1score:',f1_score_test)
```

✓ 1.4s

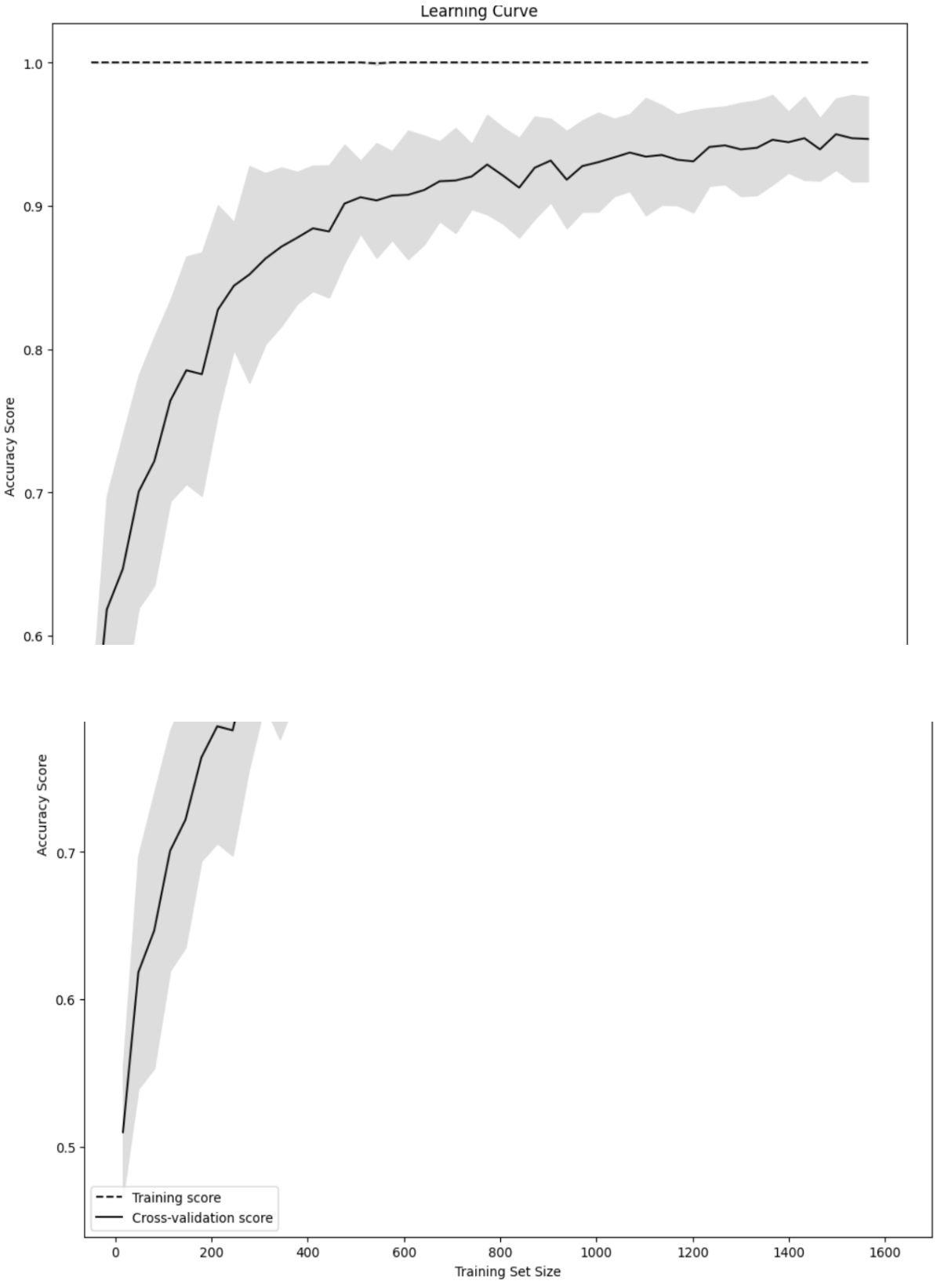
```
Train f1score: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
Test f1score: [0.98823529 1.         1.         1.         0.98701299 0.96774194
 1.         0.98630137 0.96428571 0.95652174]
```

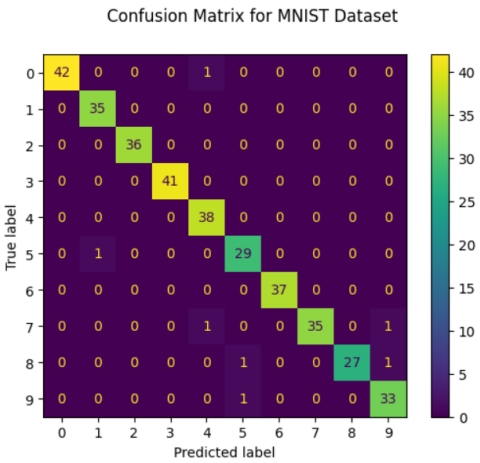
4-تحلیل خود را از دقتهای بدست آمده بنویسید.

داده های Train با MLP classifier با دقت صد درصد اموزش می بینند و با دقت 0.9861 داده های تست را به درستی تشخیص و طبقه میکند.همین طور مقادیر f1 score ,recall , Cofusion matrix نیز براورد شده است.مشخص میشود که هایپر پارامترها به خوبی تنظیم شده است و مدل به خوبی میتواند با داده های اموزش یادبگیرد و با دقت بالا داده های تست را ارزیابی و طبقه بندی نماید.

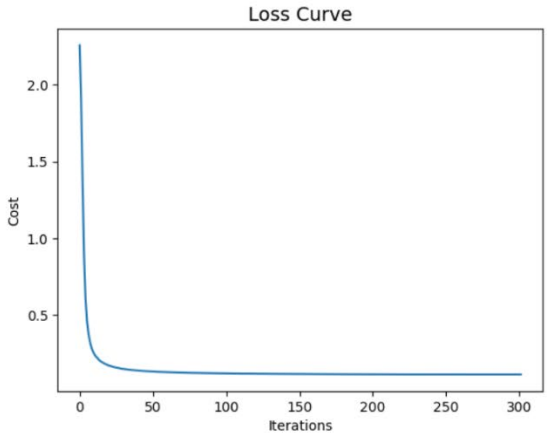
5- (نمودار روند همگرایی مدل)تغییرات خطای آموزش طی تکرارهای متوالی الگوریتم را رسم نمایید

```
X=mnist.data
Y=mnist.target
xtr,xte,ytr,yte=ms.train_test_split(X,Y,train_size=0.8,random_state=1)
MLP=nn.MLPClassifier(hidden_layer_sizes=(50,),activation='logistic',alpha=0.1,solver='sgd',max_iter=1000,verbose=10,random_state=1,learning_rate_init=0.1,learn
MLP.fit(xtr,ytr)
trACC=MLP.score(xtr,ytr)
teACC=MLP.score(xte,yte)
print('Train accuracy:', trACC)
print('Test accuracy:', teACC)

train_sizes, train_scores, test_scores = learning_curve(MLPClassifier(), X, Y, cv=10, scoring='accuracy', n_jobs=-1, train_sizes=np.linspace(0.01, 1.0, 50))

train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)

test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)


plt.subplots(1, figsize=(10,10))
plt.plot(train_sizes, train_mean, '--', color="#111111",  label="Training score")
plt.plot(train_sizes, test_mean, color="#111111", label="Cross-validation score")

plt.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, color="#DDDDDD")
plt.fill_between(train_sizes, test_mean - test_std, test_mean + test_std, color="#DDDDDD")

plt.title("Learning Curve")
plt.xlabel("Training Set Size"), plt.ylabel("Accuracy Score"), plt.legend(loc="best")
plt.tight_layout()
plt.show()
```


Learning Curve
Accuracy Score
1.0
0.9
0.8
0.7
0.6
Accuracy Score
0.7
0.6
0.5
Training score
Cross-validation score
0
200
400
600
800
1000
1200
1400
1600
Training Set Size

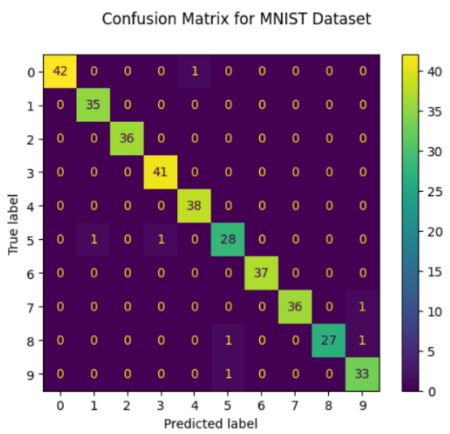



```
[2.2581933653973953, 1.8941289532284298, 1.3648635887181209, 0.8887745589567048, 0.607336431915613, 0.46028140992804206, 0.3779326021155344, 0.3256465881138954, 0
```



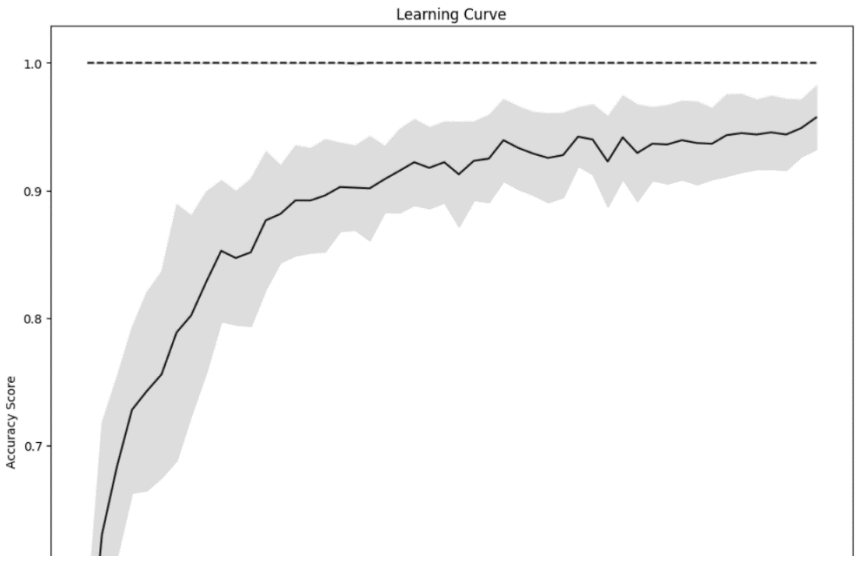
6- بـه جـای هـزار بـار تکـرار الگـوریتم بـا اجـرای صـد بـار الگـوریتم آیـا مـدل همگـرا میشـود؟ بـا رسـم مجـدد نمـودار همگرایی بررسی کنید؟ خیر

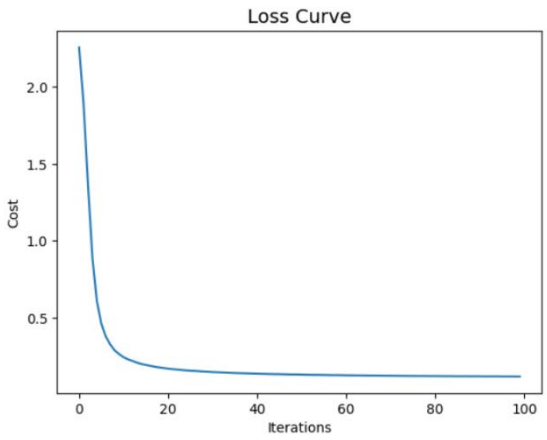
```
Iteration 99, loss = 0.05454938
Iteration 100, loss = 0.05463692
Train accuracy: 1.0
Test accuracy: 0.9833333333333333

\neural_network\_multilayer_perceptron.py:702: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (100) reached and the optimization hasn't converged yet.
```



[2.2581933653973953, 1.8941289532284298, 1.3648635887181209, 0.8887745589567048, 0.607336431915613, 0.460281409928042O6, 0.3779326021155344, 0.3256465881138954, 0

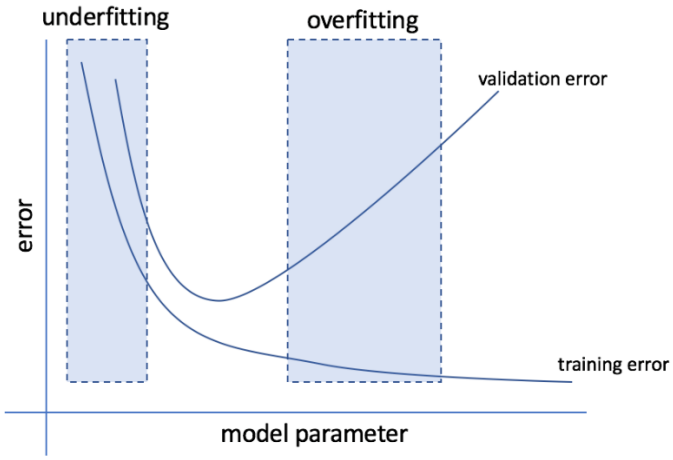
همانطور که در نمودار یادگیری می بینیم مدل با صد بار تکرار همگرا نشده است.





7-دقت مدل با 100 بار تکرار برای آموزش و تست چقدر است؟ تحلیل کنید.

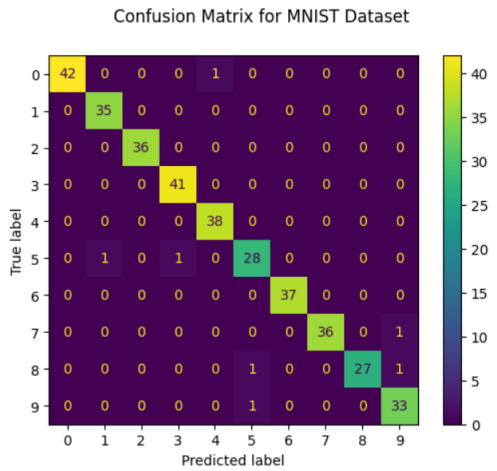
دقت بعد از صد بار iteration همانطور که از خزوجی مشخص است برای داده های اموزش 0.8135 درصد و برای داده های تست 0.8222 درصد میباشد.مقادیر Recall و F1-score نیز برای صد بار تکرار براورد دقیق شده است. دقت به عنوان کسری از مثال‌های مرتبط (مثبت‌های واقعی) در بین همه نمونه‌هایی که پیش‌بینی شده بود به یک کلاس خاص تعلق دارند، تعریف می‌شود. Recall به عنوان کسری از مثال‌ها تعریف می‌شود که پیش‌بینی می‌شد با توجه به تمام مثال‌هایی که واقعاً به کلاس تعلق دارند، متعلق به یک کلاس باشند. شکل زیر میزان خطا در طی تکرارهای متوالی و نحوه تحلیل نمودار را نشان میدهد.



```python
from sklearn.metrics import accuracy_score, confusion_matrix, plot_confusion_matrix,precision_score,recall_score,f1_score
from sklearn.model_selection import learning_curve
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler

mnist=dt.load_digits()
mnist.keys()
pd.DataFrame(mnist.data).head()
X=mnist.data
Y=mnist.target
xtr,xte,ytr,yte=ms.train_test_split(X,Y,train_size=0.8,random_state=1)

sc=StandardScaler()
scaler = sc.fit(xtr)
trainX_scaled = scaler.transform(xtr)
testX_scaled = scaler.transform(xte)

MLP=nn.MLPClassifier(hidden_layer_sizes=(50,),activation='logistic',alpha=0.1,solver='sgd',max_iter=100,verbose=10,random_state=1,learning_rate_init=0.1,learni
MLP.fit(trainX_scaled,ytr)

y_pred = MLP.predict(testX_scaled)

print('Accuracy: {:.2f}'.format(accuracy_score(yte, y_pred)))
fig = plot_confusion_matrix(MLP, testX_scaled, yte, display_labels=MLP.classes_)
fig.figure_.suptitle("Confusion Matrix for MNIST Dataset")
plt.show()

trACC=MLP.score(xtr,ytr)
teACC=MLP.score(xte,yte)

print('Train accuracy:', trACC)
print('Test accuracy:', teACC)
```

1.3s Python

</>




```
Train accuracy: 0.813500347947112
Test accuracy: 0.8222222222222222
```

```
Train accuracy: 0.813500347947112
Test accuracy: 0.8222222222222222
```

```
recall_train=recall_score(ytr,y_pred_train,average=None)
recall_test=recall_score(yte,y_pred_test,average=None)

print('Train recall:',recall_train)
print('Test recall:',recall_test)
```

[18] ✓ 0.0s Python

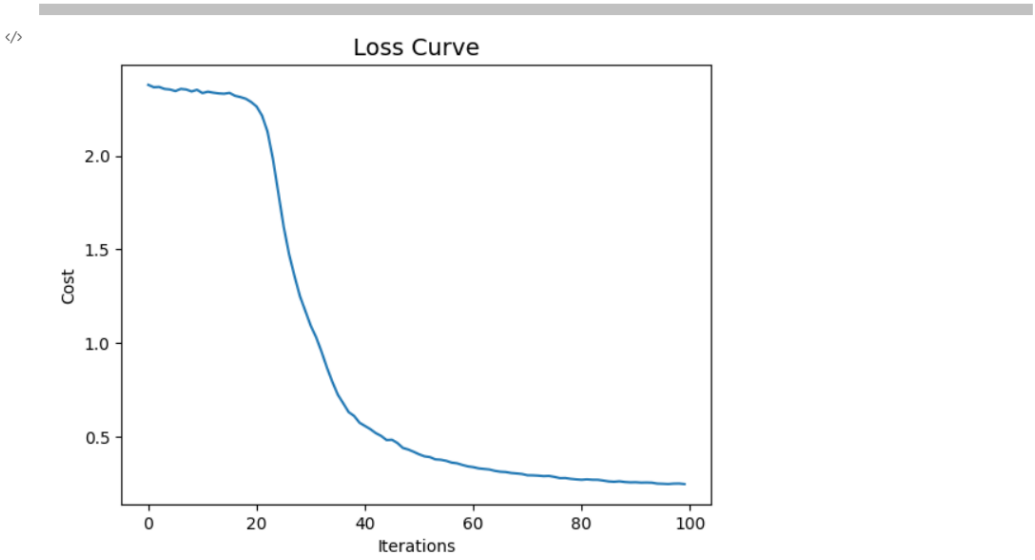
```
Train recall: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
Test recall: [0.97674419 1.         1.         1.         1.
 1.         0.97297297 0.93103448 0.97058824]
```

```
f1_score_train=f1_score(ytr,y_pred_train,average=None)
f1_score_test=f1_score(yte,y_pred_test,average=None)

print('Train f1score:',f1_score_train)
print('Test f1score:',f1_score_test)
```

[20] ✓ 0.0s Python

```
Train f1score: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
Test f1score: [0.98823529 1.         1.         1.         0.98701299 0.96774194
 1.         0.98630137 0.96428571 0.95652174]
```

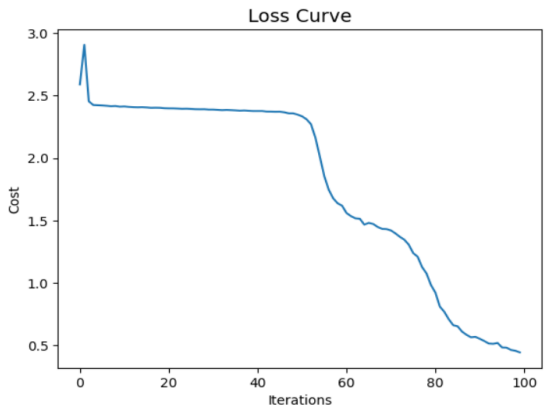
8-حال تعداد نرونهای لایه مخفی را تغییر میدهیم. به ترتیب برای مقادیر 100و150 و 200 نمودار تغییرات خطای آموزش و اعتبارسنجی را به ازای تعداد نرونهای مختلف رسم نمایید. آیا افزایش تعداد نرون تاثیر بهینه داشته است؟ خیر افزایش تعداد نرون در صد بار تکرار باعث افزایش و سپس کاهش خطا شده است. نمودار خطا با 50 نرون میزان خطای کمتری را در روند اموزش داشته است.



```
X=mnist.data
Y=mnist.target
xtr,xte,ytr,yte=ms.train_test_split(X,Y,train_size=0.8,random_state=1)

sc=StandardScaler()
scaler = sc.fit(xtr)
trainX_scaled = scaler.transform(xtr)
testX_scaled = scaler.transform(xte)

MLP=nn.MLPClassifier(hidden_layer_sizes=(500,750,400),activation='logistic',alpha=0.1,solver='sgd',max_iter=100,verbose=10,random_state=1,learning_rate_init=
MLP.fit(trainX_scaled,ytr)

y_pred = MLP.predict(testX_scaled)

trACC=MLP.score(xtr,ytr)
teACC=MLP.score(xte,yte)

print('Train accuracy:', trACC)
print('Test accuracy:', teACC)
trACC=MLP.score(xtr,ytr)
teACC=MLP.score(xte,yte)
loss_values = MLP.loss_curve_
print (loss_values)
plt.plot(loss_values)
plt.title("Loss Curve", fontsize=14)
plt.xlabel('Iterations')
plt.ylabel('Cost')
plt.show()
```

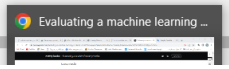
55.0s Pytho

```
Train accuracy: 0.7077244258872651
Test accuracy: 0.725
[2.5897693544347495, 2.9071003428536697, 2.4550589383565593, 2.425018124286918, 2.4232409047524297, 2.421460630479477, 2.4188163177245863, 2.4154048372641306, 2.4
```



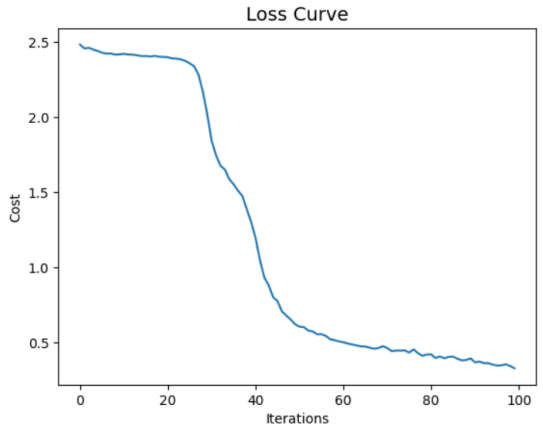
همین طور که مشاهده می کنید نمودار خطا با افزایش میزان نرونها در هر لایه افزایش پیدا کرده است.بطوریکه دقت برای مرحله اموزش 0.7077 درصد و برای مرحله تست به 0.725 درصد رسیده و کاهش پیدا کرده است.

```python
sc=StandardScaler()
scaler = sc.fit(xtr)
trainX_scaled = scaler.transform(xtr)
testX_scaled = scaler.transform(xte)

MLP=nn.MLPClassifier(hidden_layer_sizes=(1000,750,200),activation='logistic',alpha=0.1,solver='sgd',max_iter=100,verbose=10,random_state=1,learning_rate_init=0
MLP.fit(trainX_scaled,ytr)

y_pred = MLP.predict(testX_scaled)

trACC=MLP.score(xtr,ytr)
teACC=MLP.score(xte,yte)

print('Train accuracy:', trACC)
print('Test accuracy:', teACC)
trACC=MLP.score(xtr,ytr)
teACC=MLP.score(xte,yte)
loss_values = MLP.loss_curve_
print (loss_values)
plt.plot(loss_values)
plt.title("Loss Curve", fontsize=14)
plt.xlabel('Iterations')
plt.ylabel('Cost')
plt.show()
```



1m 16.5s Python

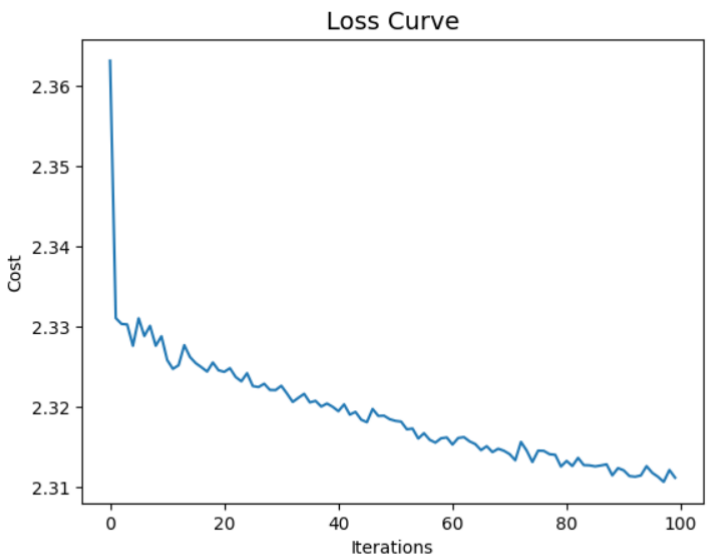
```
Train accuracy: 0.7835768963117606
Test accuracy: 0.7972222222222223
[2.4802446650658565, 2.4548184637941883, 2.4590469753022877, 2.447575126763922, 2.438112822275095, 2.4268996045114664, 2.4205108054834334, 2.42130132508052, 2.413
```



9-تعداد لایه مخفی را به جای یک لایه, دو لایه قرار دهید با همان تعداد 50 نرون.دقت تست افزایش می یابد؟ یا کاهش؟ در حالت کلی افزایش لایه منجر به بهبود عملکرد میشود؟

```
mnist=dt.load_digits()
mnist.keys()
pd.DataFrame(mnist.data).head()
X=mnist.data
Y=mnist.target
xtr,xte,ytr,yte=ms.train_test_split(X,Y,train_size=0.8,random_state=1)

sc=StandardScaler()
scaler = sc.fit(xtr)
trainX_scaled = scaler.transform(xtr)
testX_scaled = scaler.transform(xte)

MLP=nn.MLPClassifier(hidden_layer_sizes=(50,50,40,10),activation='logistic',alpha=0.1,solver='sgd',max_iter=100,verbose=10,random_state=1,learning_rate_init=0.
MLP.fit(trainX_scaled,ytr)

y_pred = MLP.predict(testX_scaled)

trACC=MLP.score(xtr,ytr)
teACC=MLP.score(xte,yte)

print('Train accuracy:', trACC)
print('Test accuracy:', teACC)
trACC=MLP.score(xtr,ytr)
teACC=MLP.score(xte,yte)
loss_values = MLP.loss_curve_
print (loss_values)
plt.plot(loss_values)
plt.title("Loss Curve",fontsize=14)
```

دقت با افزایش تعداد لایه مخفی بسیار کاهش پیدا کرده است بطوریکه در مدل اولیه با یک لایه مخفی به صد درصد دقت در مرحله اموزش رسید درحالیکه در اینجا دقت 0.10 درصد و کاهش شدید داشته است.



نمودار بالا برای `hidden_layer_sizes=(50,100,20,40,10)` است , سه لایه مخفی دارد که این مورد هم میزان خطایی بالایی را نسبت به قبل دارد و دقت کاهش پیدا کرده است. بطور کلی با افزایش تعداد لایه ها در لایه مخفی با توجه به نوع مساله ممکن است میزان خطا افزایش یا کاهش پیدا کند. این به ماهیت مساله برمیگردد. در مورد اغلب مدلهایی که تعداد ویژگیها زیاد و نوعا مسایل پیچیده ای هستند معمولا افزایش لایه ها جواب بهتری را در اموزش و تست به ما میدهد.