DSE3900 - APR4

September 14, 2023

[ ]:

```
from sklearn.datasets import fetch_openml

X, y = fetch_openml('mnist_784', version =1, return_X_y=True, parser='auto')

X = X.values
y = y.astype(int).values

X = ((X / 255.)-.5)*2
```

[ ]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify= y,␣
 ↪test_size=0.2)
```

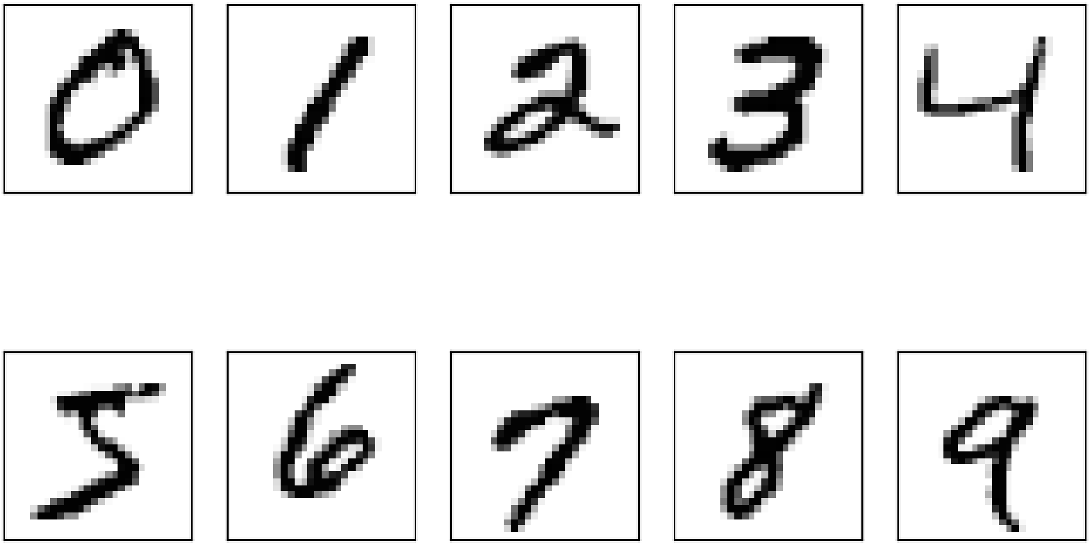
[ ]:

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots(nrows=2, ncols=5, sharex = True, sharey= True)
ax = ax.flatten()

for i in range(0, 10):
    img = X[y==i][2].reshape(28,28)
    ax[i].imshow(img, cmap = 'Greys')

ax[0].set_xticks([])
ax[0].set_yticks([])
plt.tight_layout()
plt.show()
```

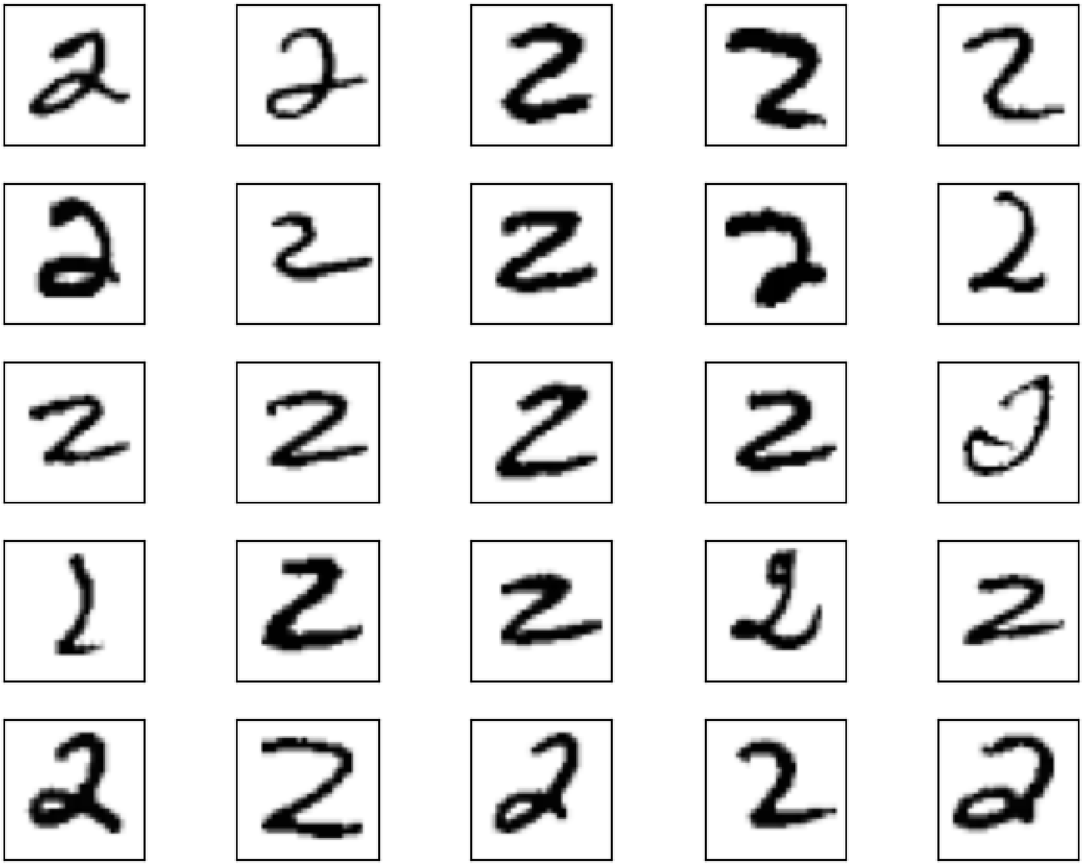


[ ]:

```
fig, ax = plt.subplots(nrows=5, ncols=5, sharex = True, sharey= True)
ax = ax.flatten()

for i in range(25):
    img = X[y==2][i].reshape(28,28)
    ax[i].imshow(img, cmap='Greys')

ax[0].set_xticks([])
ax[0].set_yticks([])
plt.tight_layout()
plt.show()
```



```
[ ]: from sklearn.neural_network import MLPClassifier as MLP
     from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,␣
      ↪accuracy_score
```

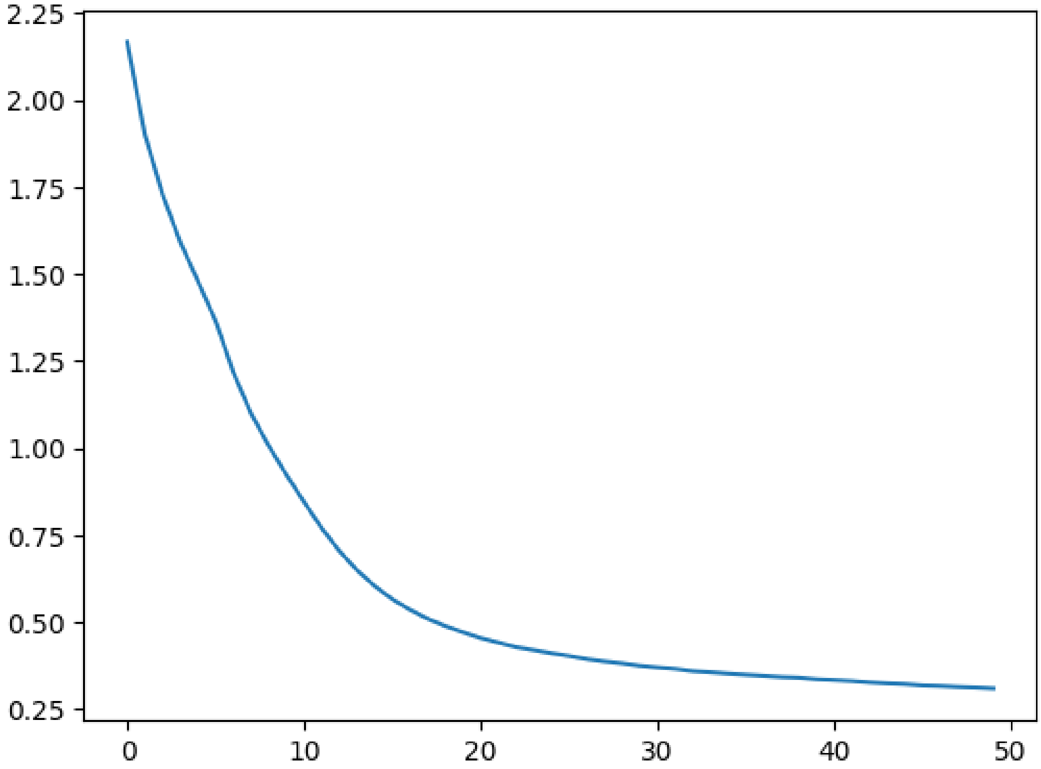
```
[ ]: mlp = MLP(hidden_layer_sizes=(50,20,10,5,), activation='logistic',max_iter=50)
     mlp.fit(X_train, y_train)
```

```
c:\Users\patri\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\neural_network\_multilayer_perceptron.py:686:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (50) reached and
the optimization hasn't converged yet.
  warnings.warn(
```

```
[ ]: MLPClassifier(activation='logistic', hidden_layer_sizes=(50, 20, 10, 5),
                   max_iter=50)
```

```
[ ]: plt.plot(mlp.loss_curve_)
```

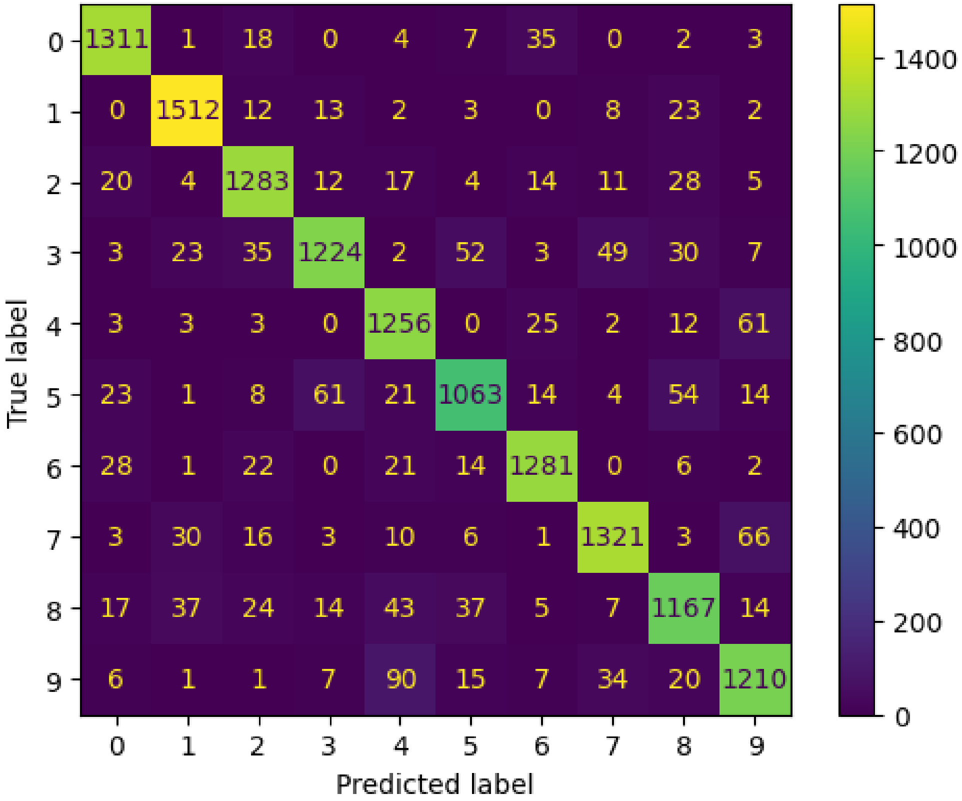
```
[ ]: [<matplotlib.lines.Line2D at 0x1d429eefe50>]
```



[ ]:
```
y_pred = mlp.predict(X_test)

ConfusionMatrixDisplay(confusion_matrix(y_test, y_pred)).plot()
print('Accuracy: ', accuracy_score(y_test, y_pred))
```

```
Accuracy:  0.902
```



[ ]:

```
import pandas as pd

for i in range (10):
    y_test_num = y_test[y_test == i]
    y_pred_num = y_pred[y_test == i]
    prcnt = len(y_test_num[y_test_num==y_pred_num])/len(y_test_num)
    print('percent of', i, 'correct:', round(prcnt,2))
print('Success rate of labeling %i was: %f' %(i,round(prcnt,2)))
```

```
percent of 0 correct: 0.96
percent of 1 correct: 0.97
percent of 2 correct: 0.9
percent of 3 correct: 0.89
percent of 4 correct: 0.94
percent of 5 correct: 0.88
percent of 6 correct: 0.95
percent of 7 correct: 0.93
percent of 8 correct: 0.88
percent of 9 correct: 0.91
Success rate of labeling 9 was: 0.910000
```

[ ]:

```
mlp = MLP(hidden_layer_sizes=(10,5), activation='logistic',max_iter=50,␣
 ↪verbose=True, tol=0.0001)
mlp.fit(X_train, y_train)
```

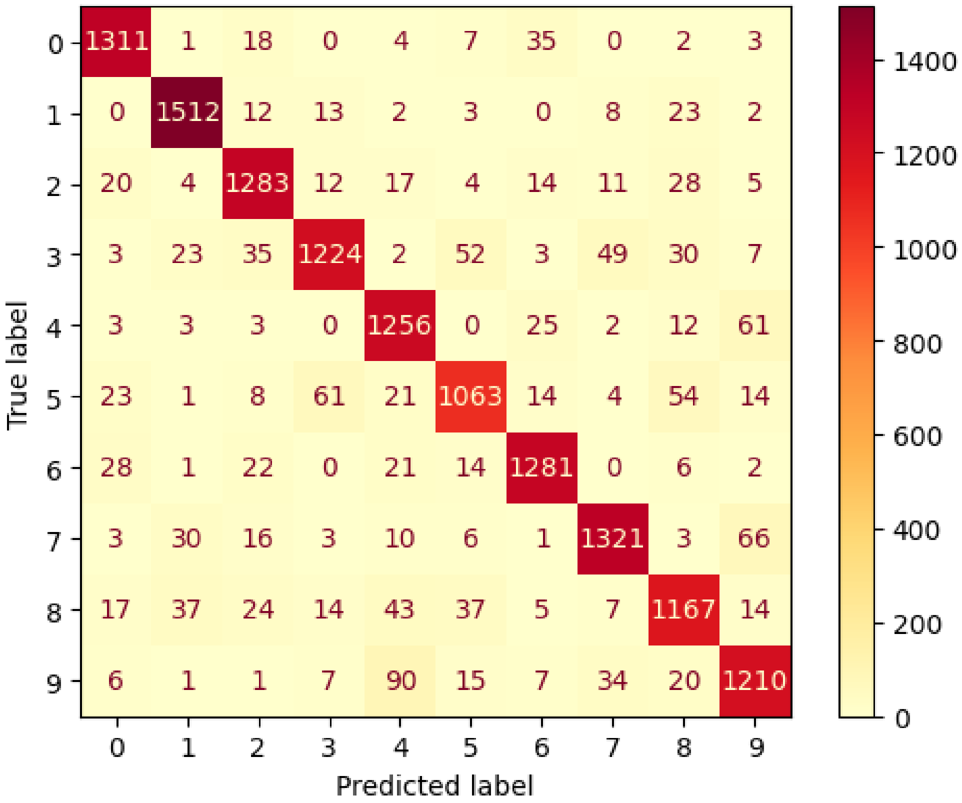
```
Iteration 1, loss = 2.16651402
Iteration 2, loss = 1.90077955
Iteration 3, loss = 1.72867699
Iteration 4, loss = 1.59210294
Iteration 5, loss = 1.47970946
Iteration 6, loss = 1.36575479
Iteration 7, loss = 1.21811033
Iteration 8, loss = 1.10045271
Iteration 9, loss = 1.00688417
Iteration 10, loss = 0.92415229
Iteration 11, loss = 0.84570911
Iteration 12, loss = 0.77074324
Iteration 13, loss = 0.70453800
Iteration 14, loss = 0.64964856
Iteration 15, loss = 0.60405658
Iteration 16, loss = 0.56608207
Iteration 17, loss = 0.53583074
Iteration 18, loss = 0.51003562
Iteration 19, loss = 0.48915934
Iteration 20, loss = 0.47111695
Iteration 21, loss = 0.45466965
Iteration 22, loss = 0.44168918
Iteration 23, loss = 0.42873927
Iteration 24, loss = 0.42011140
Iteration 25, loss = 0.41107088
Iteration 26, loss = 0.40328862
Iteration 27, loss = 0.39430946
Iteration 28, loss = 0.38759175
Iteration 29, loss = 0.38202518
Iteration 30, loss = 0.37518728
Iteration 31, loss = 0.37027572
Iteration 32, loss = 0.36642070
Iteration 33, loss = 0.35974843
Iteration 34, loss = 0.35659027
Iteration 35, loss = 0.35262458
Iteration 36, loss = 0.34937253
Iteration 37, loss = 0.34624457
Iteration 38, loss = 0.34286605
Iteration 39, loss = 0.34105485
Iteration 40, loss = 0.33631706
Iteration 41, loss = 0.33405854
Iteration 42, loss = 0.33134404
Iteration 43, loss = 0.32757413
Iteration 44, loss = 0.32466061
Iteration 45, loss = 0.32256313
Iteration 46, loss = 0.31867461
Iteration 47, loss = 0.31706853
Iteration 48, loss = 0.31454241
Iteration 49, loss = 0.31277983
Iteration 50, loss = 0.31029305
```

```
c:\Users\patri\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\neural_network\_multilayer_perceptron.py:686:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (50) reached and
the optimization hasn't converged yet.
  warnings.warn(
```

```
[ ]: MLPClassifier(activation='logistic', hidden_layer_sizes=(10, 5), max_iter=50,
                   verbose=True)
```

```
[ ]: def plotConfusionMatrix(y_test, y_pred):
         ConfusionMatrixDisplay(confusion_matrix(y_test, y_pred)).plot(cmap='YlOrRd')

     plotConfusionMatrix(y_test,y_pred)
     accuracy_score(y_test, y_pred)
```



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	1311	1	18	0	4	7	35	0	2	3
1	0	1512	12	13	2	3	0	8	23	2
2	20	4	1283	12	17	4	14	11	28	5
3	3	23	35	1224	2	52	3	49	30	7
4	3	3	3	0	1256	0	25	2	12	61
5	23	1	8	61	21	1063	14	4	54	14
6	28	1	22	0	21	14	1281	0	6	2
7	3	30	16	3	10	6	1	1321	3	66
8	17	37	24	14	43	37	5	7	1167	14
9	6	1	1	7	90	15	7	34	20	1210

[ ]:

```python
from sklearn.linear_model import LogisticRegression

lgr = LogisticRegression()
lgr.fit(X_train, y_train)
```

```
c:\Users\patri\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

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(
```

[ ]:

```
LogisticRegression()
```

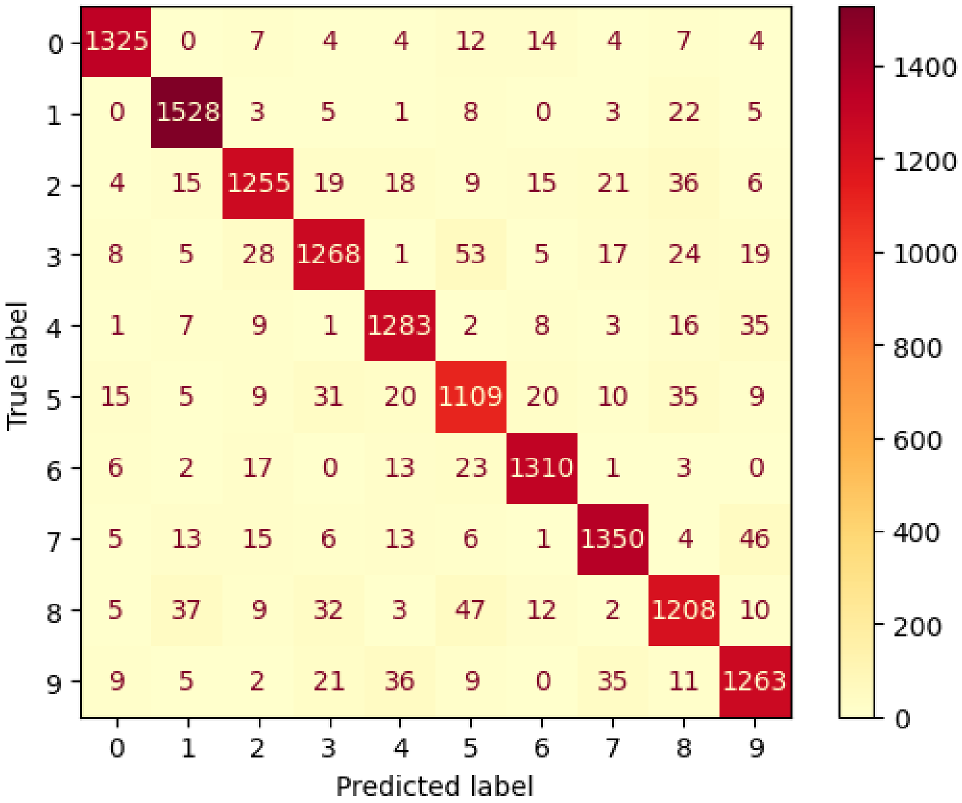
[ ]:

```python
y_pred = lgr.predict(X_test)

plotConfusionMatrix(y_test, y_pred)
accuracy_score(y_test, y_pred)
```

[ ]:

```
0.9213571428571429
```


True label
Predicted label
0 1 2 3 4 5 6 7 8 9
0 1325 0 7 4 4 12 14 4 7 4
1 0 1528 3 5 1 8 0 3 22 5
2 4 15 1255 19 18 9 15 21 36 6
3 8 5 28 1268 1 53 5 17 24 19
4 1 7 9 1 1283 2 8 3 16 35
5 15 5 9 31 20 1109 20 10 35 9
6 6 2 17 0 13 23 1310 1 3 0
7 5 13 15 6 13 6 1 1350 4 46
8 5 37 9 32 3 47 12 2 1208 10
9 9 5 2 21 36 9 0 35 11 1263
1400
1200
1000
800
600
400
200
0