Practice Questions

All the programming questions use **MNIST** dataset. Store all the samples in X and labels in y

```
# Common imports
import numpy as np
from pprint import pprint
from tempfile import mkdtemp
from shutil import rmtree

# to make this notebook's output stable across runs
np.random.seed(42)

#sklearn specific imports
# Dataset fetching
from sklearn.datasets import fetch_openml

# Feature scaling
from sklearn.preprocessing import StandardScaler, MinMaxScaler

# Pipeline utility
from sklearn.pipeline import make_pipeline

# Classifiers: dummy, logistic regression (SGD and LogisticRegression)
# and least square classification
from sklearn.dummy import DummyClassifier
from sklearn.linear_model import SGDClassifier,RidgeClassifier, LogisticRegression,Logisti

# Model selection
from sklearn.model_selection import cross_validate,RandomizedSearchCV,GridSearchCV,cross_v
from sklearn.model_selection import learning_curve,train_test_split


# Evaluation metrics
from sklearn.metrics import log_loss
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import precision_score, recall_score, classification_report
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import auc,roc_curve,roc_auc_score

# scipy
from scipy.stats import loguniform

# To plot pretty figures
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# global settings
mpl.rc('axes', labelsize=14)
mpl.rc('xtick', labelsize=12)
mpl.rc('ytick', labelsize=12)
mpl.rc('figure',figsize=(8,6))

# Ignore all warnings (convergence..) by sklearn
def warn(*args, **kwargs):
    pass
import warnings
warnings.warn = warn


X_pd,y_pd= fetch_openml('mnist_784',version=1,return_X_y=True)



# convert to numpy array
X = X_pd.to_numpy()
y = y_pd.to_numpy().astype(np.int32)
```

Question Group: 22T1_MLP_W5_PP

Total # of questiosn: 6

Que 1.

[1 point] Split the dataset in the following ratio.

1. Training : Take the first 70% of samples from X and store them in `x_train`
2. Testing: Take the remaining 30% of samples from X and store them in `x_test`
3. Store the respective labels in `y_train,y_test` respectively.

The last training sample is of digit __?

Type: NAT

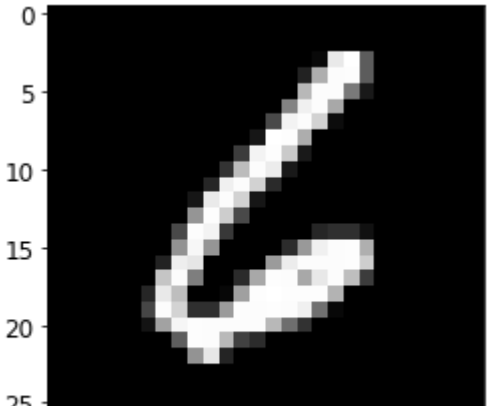
Ans : 6

Solution

```
x_train,x_test,y_train,y_test = X[:49000],X[49000:],y[:49000],y[49000:]


print('The last sample is of digit:',y_train[-1])
plt.imshow(x_train[-1,:].reshape(28,28),cmap='gray')
```

```
The last sample is of digit: 6
<matplotlib.image.AxesImage at 0x7f1d3256d890>
```



Common data

Build a classifier that differentiate digit 6 from digit 9.

- Steps to be followed
    1. Collect all digit-6 (Positive class) and digit-9 (Negative class) images from `x_train` and stack them properly as a single datamatri `x_train_69` .
    2. Keep all digit-6 images from index 0 to i followed by digit-9 images from index i+1 to n (i denotes the end index of digit-6 images)
    3. Similarly, collect the respective labels and store it in a variable `y_train_69`
    4. Set the label values to 1 for positive classes and 0 for negative classes.
    5. Load `from sklearn.utils import shuffle`
    6. Shuffle the datamatrix and labels. (Set `random_state` value to 1729).
    7. Create `x_test_69` and `y_test_69` by repeating the steps from 1 to 6 with required modifications

Que 2.

[1 point] What is the sum of all the labels in the vector `y_train_69` . [NAT]

Ans:4855

[1 point] What is the sum of all the labels in the vector `y_test_69` . [NAT]
Ans:2021

Solution

Train set

```
# get the index of dig6 and dig_9
dig_6_idx = np.where(y_train ==6)[0]
dig_9_idx = np.where(y_train ==9)[0]
index = np.concatenate((dig_6_idx,dig_9_idx),axis=0)
```

```
# get all samples from the index array
x_train_69 = x_train[index,:]
# create the label vector
y_train_69 = np.concatenate((np.ones(len(dig_6_idx)),np.zeros(len(dig_9_idx))))

from sklearn.utils import shuffle
x_train_69,y_train_69 = shuffle(x_train_69,y_train_69,random_state=1729)
```

Test set

```
# get the index of dig6 and dig_9
dig_6_idx = np.where(y_test ==6)[0]
dig_9_idx = np.where(y_test ==9)[0]
index = np.concatenate((dig_6_idx,dig_9_idx),axis=0)
# get all samples from the index array
x_test_69 = x_test[index,:]
# create the label vector
y_test_69 = np.concatenate((np.ones(len(dig_6_idx)),np.zeros(len(dig_9_idx))))

from sklearn.utils import shuffle
x_test_69,y_test_69 = shuffle(x_test_69,y_test_69,random_state=1729)


print('The sum of label vectors:',np.sum(y_train_69))
print('The sum of label vectors:',np.sum(y_test_69))
```

```
The sum of label vectors: 4855.0
The sum of label vectors: 2021.0
```

```
print('The sum of label vectors:',np.count_nonzero(y_train==6))
```

```
The sum of label vectors: 4855
```

Que 3.

[2 point] Apply `StandardScaler` to all the training samples in `x_train_69` and store the result in another variable (say, `x_train_69Tf`).

```
* What is the mean of the zeroth sample?
* What is the mean of zeroth feature?
* What is the standard deviation of the zeroth sample?
* What is the standard deviation of the zeroth feature?
```

Pack the answers (in order) in a tuple

Options:

1. (0,0,1,1)
2. (0.081,0,0.73,1)
3. (0.081,0,0.73,0)
4. (0,0.081,1.09,1)

Ans: 3

Solution:

```
scaler = StandardScaler()
x_train_69Tf = scaler.fit_transform(x_train_69)


print('Mean of 0th sample:',np.mean(x_train_69Tf[0,:]))
print('Mean of 0th sample:',np.mean(x_train_69Tf[:,0]))
print('Std of the 0th sample:',np.std(x_train_69Tf[0,:]))
print('Std of the 0th sample:',np.std(x_train_69Tf[:,0]))
```

```
Mean of 0th sample: 0.08128379559427823
Mean of 0th sample: 0.0
Std of the 0th sample: 0.7358823226037738
Std of the 0th sample: 0.0
```

Que 4.

[6 point]Train the LogisticRegression model using SGDClassifier() with the following common settings.

1. No Regularization
2. random_state : 10
3. Iteration : 10

Capture the loss for each iteration and plot the iteration vs loss curve. For which of the following settings, the iteration vs loss curve decreased monotonically?

```
A. Set Learning rate : 0.01 and plot the curve and fit the model with `x_train_69 `

B.Set learning rate to 0.000001 and fit the model with `x_train_69 ` .

C.Keep the learning rate as 0.01. Scale the samples using StandardScaler() and fit the model with

D.Use the "invscaling" stratagey for the learning rate with power_t = 1. Fit the model with x_trai
```

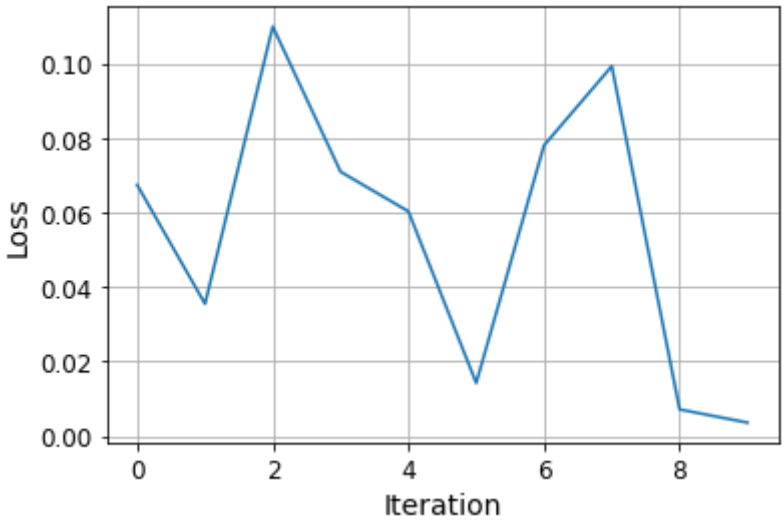
Answer: B,C,D

[0 point] How do you explain all these observations?

Solution:

1.

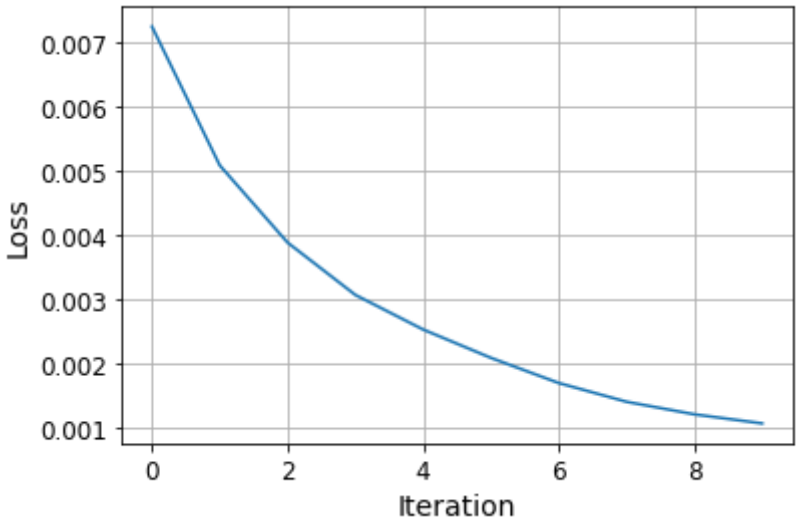
```
estimator = SGDClassifier(loss='log',
                          penalty='l2',
                          max_iter=1,
                          warm_start=True,
                          eta0=0.01,
                          alpha=0,
                          learning_rate='constant',
                          random_state=10)
pipe_sgd= make_pipeline(estimator)
Loss=[]
iterations= 10
for i in range(iterations):
  pipe_sgd.fit(x_train_69,y_train_69)
  y_pred = pipe_sgd.predict_proba(x_train_69)
  Loss.append(log_loss(y_train_69,y_pred))
plt.figure()
plt.plot(np.arange(iterations),Loss)
plt.grid(True)
plt.xlabel('Iteration')
plt.ylabel('Loss')
plt.show()
```



2.

```
estimator = SGDClassifier(loss='log',
                          penalty='l2',
                          max_iter=1,
                          warm_start=True,
                          eta0=0.000001,
                          alpha=0,
                          learning_rate='constant',
```

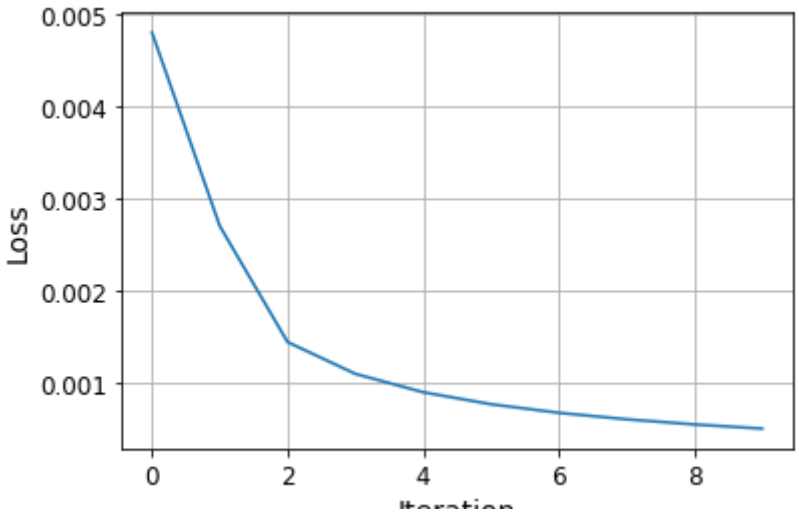
```
                        random_state=10)
pipe_sgd= make_pipeline(estimator)
Loss=[]
iterations= 10
for i in range(iterations):
  pipe_sgd.fit(x_train_69,y_train_69)
  y_pred = pipe_sgd.predict_proba(x_train_69)
  Loss.append(log_loss(y_train_69,y_pred))
plt.figure()
plt.plot(np.arange(iterations),Loss)
plt.grid(True)
plt.xlabel('Iteration')
plt.ylabel('Loss')
plt.show()
```



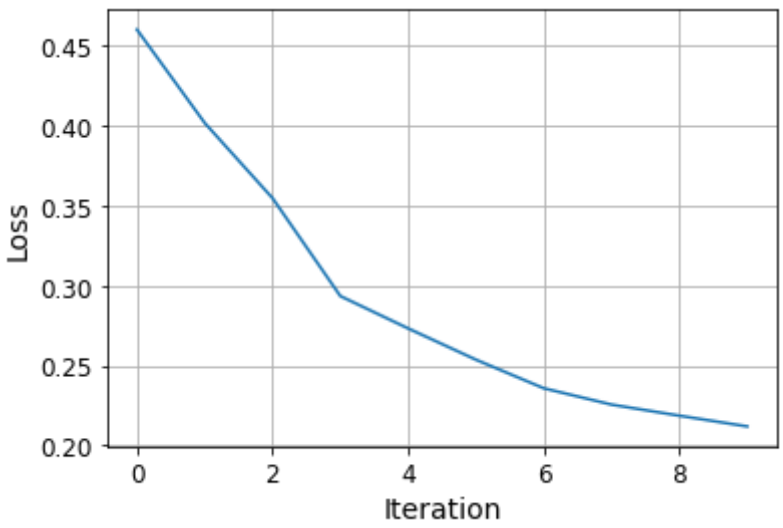
3.

```
estimator = SGDClassifier(loss='log',
                        penalty='l2',
                        max_iter=1,
                        warm_start=True,
                        eta0=0.01,
                        alpha=0,
                        learning_rate='constant',
                        random_state=10)
pipe_sgd= make_pipeline(StandardScaler(),estimator)
Loss=[]
iterations= 10
for i in range(iterations):
  pipe_sgd.fit(x_train_69,y_train_69)
  y_pred = pipe_sgd.predict_proba(x_train_69)
  Loss.append(log_loss(y_train_69,y_pred))
plt.figure()
plt.plot(np.arange(iterations),Loss)
plt.grid(True)
plt.xlabel('Iteration')
```

```
plt.ylabel('Loss')
plt.show()
```




```
estimator = SGDClassifier(loss='log',
                          penalty='l2',
                          max_iter=1,
                          warm_start=True,
                          eta0=0.01,
                          alpha=0,
                          power_t =1.0,
                          learning_rate='invscaling',
                          random_state=10)
pipe_sgd= make_pipeline(estimator)
Loss=[]
iterations= 10
for i in range(iterations):
  pipe_sgd.fit(x_train_69,y_train_69)
  y_pred = pipe_sgd.predict_proba(x_train_69)
  Loss.append(log_loss(y_train_69,y_pred))
plt.figure()
plt.plot(np.arange(iterations),Loss)
plt.grid(True)
plt.xlabel('Iteration')
plt.ylabel('Loss')
plt.show()
```



Colab paid products - Cancel contracts here