github link : https://github.com/Yashasvee-second/MLlab-ex4-spam-classification

```
!git clone https://github.com/Ojus999/Machine-Learning-Sem-6.git
```

```
fatal: destination path 'Machine-Learning-Sem-6' already exists and is not an empty directory.
```

```
# importing libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets, svm, metrics
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import cv2
```

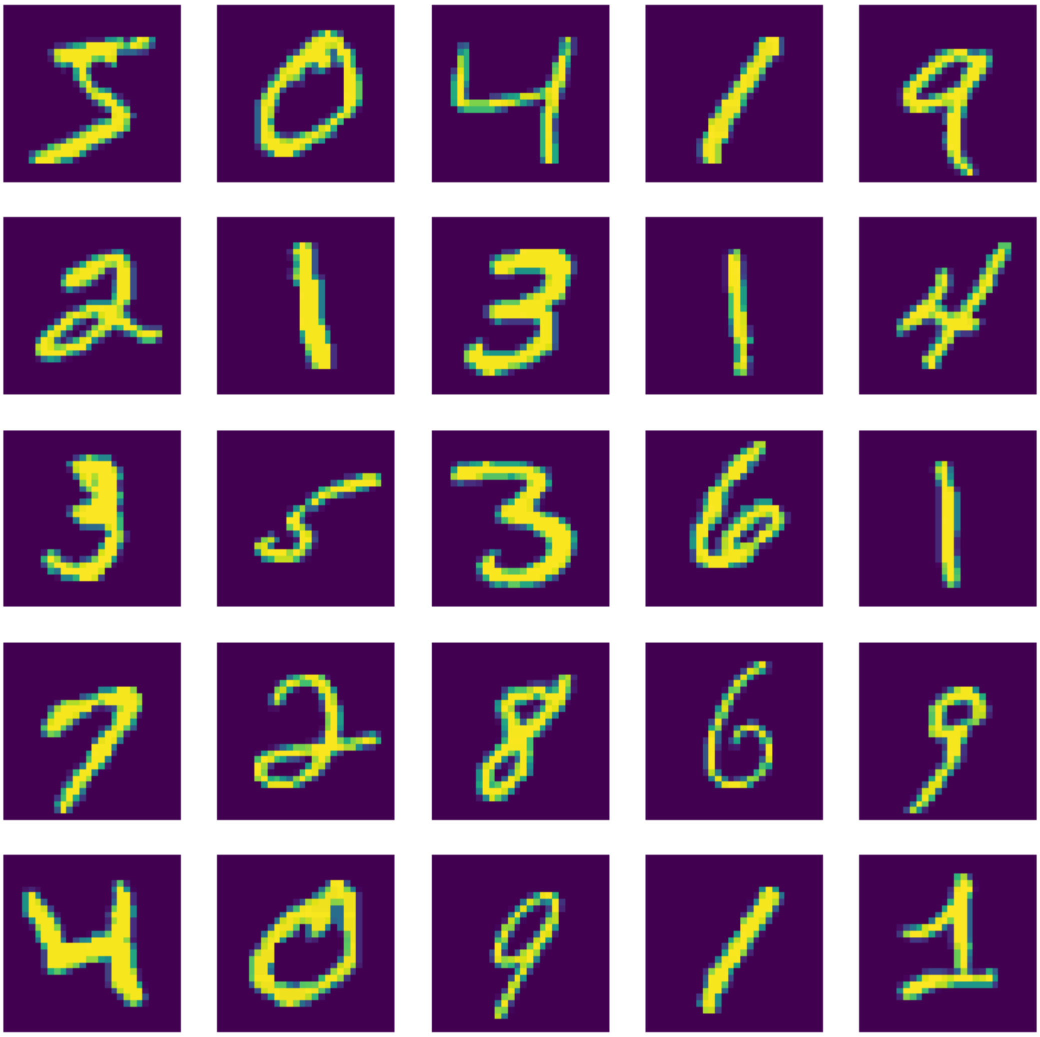
```
# funcs to help with loading data and or
def load_mnist_images(path):
    with open(path, 'rb') as f:
        data = np.frombuffer(f.read(), dtype=np.uint8, offset=16)
    return data.reshape(-1, 28*28)

def load_mnist_labels(path):
    with open(path, 'rb') as f:
        data = np.frombuffer(f.read(), dtype=np.uint8, offset=8)
    return data

X_train = load_mnist_images('/content/Machine-Learning-Sem-6/Ex 4/mnist/train-images-idx3-ubyte/train-images.idx3-ubyte')
y_train = load_mnist_labels('/content/Machine-Learning-Sem-6/Ex 4/mnist/train-labels-idx1-ubyte/train-labels.idx1-ubyte')
X_test = load_mnist_images('/content/Machine-Learning-Sem-6/Ex 4/mnist/t10k-images-idx3-ubyte/t10k-images.idx3-ubyte')
y_test = load_mnist_labels('/content/Machine-Learning-Sem-6/Ex 4/mnist/t10k-labels-idx1-ubyte/t10k-labels.idx1-ubyte')
```

```
X_train = X_train / 255.0
X_test = X_test / 255.0
```

```
# Visualization of some samples from the dataset
plt.figure(figsize=(10, 10))
for i in range(25):
    plt.subplot(5, 5, i+1)
    plt.imshow(X_train[i].reshape(28, 28))
    plt.axis('off')
plt.show()
```



```
#Split the data into training, testing, and validation sets
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2, random_state=42)
```

```
# Train the model
# C is a regularisation param, controls tradeoff between maximising margin and minimising training error
# gamma controls shape of decision boundary. Higher gamma means more smooth , accurate boundary with curves
svm_model = svm.SVC(kernel='linear', C=10, gamma="scale")
svm_model.fit(X_train, y_train)
```

```
▾          SVC
SVC(C=10, kernel='linear')
```

```
#Test the model
y_pred = svm_model.predict(X_test)
```

```
#Measure the performance of the trained model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

print("Accuracy:", accuracy)
print("Confusion Matrix:\n", conf_matrix)
print("Classification Report:\n", classification_rep)
```

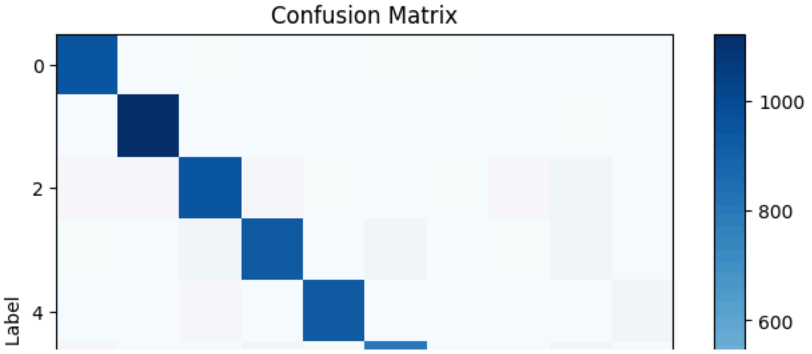
```
Accuracy: 0.9299
Confusion Matrix:
 [[ 952    0    6    1    1    8    7    2    2    1]
 [   0 1121    2    3    0    1    2    1    5    0]
 [   9    9  955   13    6    4    8   10   18    0]
 [   8    2   16  940    2   15    3    7   14    3]
 [   2    1   13    0  931    2    4    3    4   22]
 [  12    4    6   39    5  792    8    1   21    4]
 [  11    2   17    1    7   23  895    0    2    0]
 [   1    6   21   16   14    1    0  946    5   18]
 [   8    8   13   26    7   23    7    3  865   14]
 [   5    7    2   13   35    8    0   26   11  902]]
Classification Report:
```

```
              precision    recall  f1-score   support

           0       0.94      0.97      0.96       980
           1       0.97      0.99      0.98      1135
           2       0.91      0.93      0.92      1032
           3       0.89      0.93      0.91      1010
           4       0.92      0.95      0.94       982
           5       0.90      0.89      0.90       892
           6       0.96      0.93      0.95       958
           7       0.95      0.92      0.93      1028
           8       0.91      0.89      0.90       974
           9       0.94      0.89      0.91      1009

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

```
# Visualize confusion matrix
plt.figure(figsize=(8, 6))
plt.imshow(conf_matrix, cmap='Blues')
plt.colorbar()
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```


Confusion Matrix
Label