

## ▼ Support Vector Machine (SVM)

Goal : train SVM classifier on the raw pixel intensities and then classify unknown digits.

Step 1 – Structuring initial dataset

Step 2 – Splitting the dataset

Step 3 – Extracting features

Step 4 – Training classification model

Step 5 – Evaluating classifier

```
1 # import the necessary packages
2 from __future__ import print_function
3 from sklearn.neighbors import KNeighborsClassifier
4 from sklearn.metrics import classification_report
5 from sklearn import datasets
6 from skimage import exposure
7 import numpy as np
8 import imutils
9 import cv2
10 import sklearn
11
12 from sklearn.model_selection import train_test_split
13
14 # load the MNIST digits dataset
15 mnist = datasets.load_digits()
16
17 # take the MNIST data and construct the training and testing split, using 75% of the
18 # data for training and 25% for testing
19 (trainData, testData, trainLabels, testLabels) = train_test_split(np.array(mnist.data),
20     mnist.target, test_size=0.25, random_state=42)
21
22 # now, let's take 10% of the training data and use that for validation
23 (trainData, valData, trainLabels, valLabels) = train_test_split(trainData, trainLabels,
24     test_size=0.1, random_state=84)
25
26 # show the sizes of each data split
27 print("training data points: {}".format(len(trainLabels)))
28 print("validation data points: {}".format(len(valLabels)))
29 print("testing data points: {}".format(len(testLabels)))

```

training data points: 1212  
validation data points: 135  
testing data points: 450

## ▼ Imp Parameters

1) Regularization parameter (C)

2) kernel

```
1 from sklearn.svm import SVC
2 model = SVC(C=0.5, kernel='linear')
3 model.fit(trainData, trainLabels)
4 score = model.score(valData, valLabels)
5 print (score*100)

```

98.51851851851852

```
1 # test data
2 predictions = model.predict(testData)
3
4 # show a final classification report demonstrating the accuracy of the classifier
5 # for each of the digits
6 print("EVALUATION ON TESTING DATA")
7 report = (classification_report(testLabels, predictions, output_dict=True)) # made the output of report as dictionary
8 print (report)

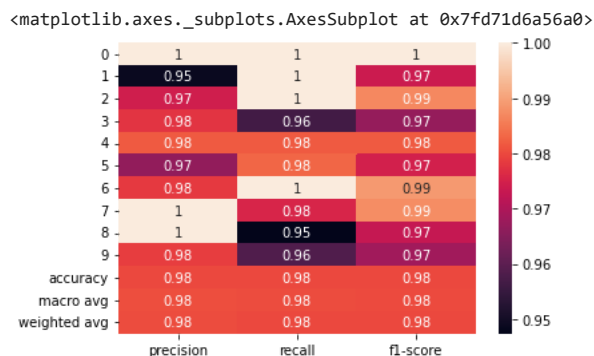
```

EVALUATION ON TESTING DATA

```
{'0': {'precision': 1.0, 'recall': 1.0, 'f1-score': 1.0, 'support': 43}, '1': {'precision': 0.9487179487179487, 'recall': 1.0, 'f1-score': 0.9717391304347826, 'support': 43}, '2': {'precision': 0.9904761904761905, 'recall': 0.9767441860465116, 'f1-score': 0.983541887804878, 'support': 43}, '3': {'precision': 0.9797300613496933, 'recall': 0.967741935483871, 'f1-score': 0.9737142857142857, 'support': 43}, '4': {'precision': 0.9803921568627451, 'recall': 0.9803921568627451, 'f1-score': 0.9803921568627451, 'support': 43}, '5': {'precision': 0.9705882352941176, 'recall': 0.9803921568627451, 'f1-score': 0.9754088061855671, 'support': 43}, '6': {'precision': 0.9803921568627451, 'recall': 0.9904761904761905, 'f1-score': 0.9854166666666667, 'support': 43}, '7': {'precision': 0.9803921568627451, 'recall': 0.9803921568627451, 'f1-score': 0.9803921568627451, 'support': 43}, '8': {'precision': 0.951915014739232, 'recall': 0.9767441860465116, 'f1-score': 0.9642857142857143, 'support': 43}, '9': {'precision': 0.967741935483871, 'recall': 0.967741935483871, 'f1-score': 0.967741935483871, 'support': 43}, 'accuracy': 0.9803921568627451, 'macro avg': 0.9803921568627451, 'weighted avg': 0.9803921568627451}
```

## ▼ classification report

```
1 import seaborn as sns
2 import pandas as pd
3 sns.heatmap(pd.DataFrame(report).iloc[:-1, :].T, annot=True)
```



```
1 from google.colab.patches import cv2_imshow
2 for i in list(map(int, np.random.randint(0, high=len(testLabels), size=(5,)))):
3     # grab the image and classify it
4     image = testData[i]
5
6     prediction = model.predict(image.reshape(1, -1))[0]
7
8     # convert the image for a 64-dim array to an 8 x 8 image compatible with OpenCV,
9     # then resize it to 32 x 32 pixels so we can see it better
10    image = image.reshape((8, 8)).astype("uint8")
11    image = exposure.rescale_intensity(image, out_range=(0, 255))
12    image = imutils.resize(image, width=32, inter=cv2.INTER_CUBIC)
13
14    # show the prediction
15    print("I think that digit is: {}".format(prediction))
16    cv2_imshow(image)
17
```

I think that digit is: 5



I think that digit is: 1



I think that digit is: 8



I think that digit is: 3



I think that digit is: 3



✓ 0s completed at 3:04 PM

● ✕