

ML-MAJOR-MAY

MAJOR PROJECT REPORT ON

**“DIGIT CLASSIFICATION USING SVM
ALGORITHM ”**

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING

For the academic year 2021-2022

SUBMITTED BY:

Chetan Chirag KH

Don Bosco Institute Of Technology

1. Problem Statement

Design a project from the MNIST dataset to identify digit classification using the SVM algorithm.

2. Aim Of the Project

We will develop a model using Support Vector Machine which should correctly classify the handwritten digits from 0-9 based on the pixel values given as features. Thus, this is a 10-class classification problem.

3. List of Libraries used in my project

- I. numpy
- II. pandas
- III. matplotlib.pyplot
- IV. seaborn

The platform I have used for my machine learning project is Google Colaboratory and have used list of libraries which are mentioned above and I have also imported sklearn for the creating my method and have use SVM algorithm.

4. Explanation

For this problem, we use the MNIST data which is a large database of handwritten digits. The 'pixel values' of each digit (image) comprise the features, and the actual number between 0-9 is the label.

Since each image is of 28 x 28 pixels, and each pixel forms a feature, there are 784 features. MNIST digit recognition is a well-studied problem in the ML community, and people have trained numerous models (Neural Networks, SVMs, boosted trees etc.) achieving error rates as low as 0.23% (i.e. accuracy = 99.77%, with a convolutional neural network).

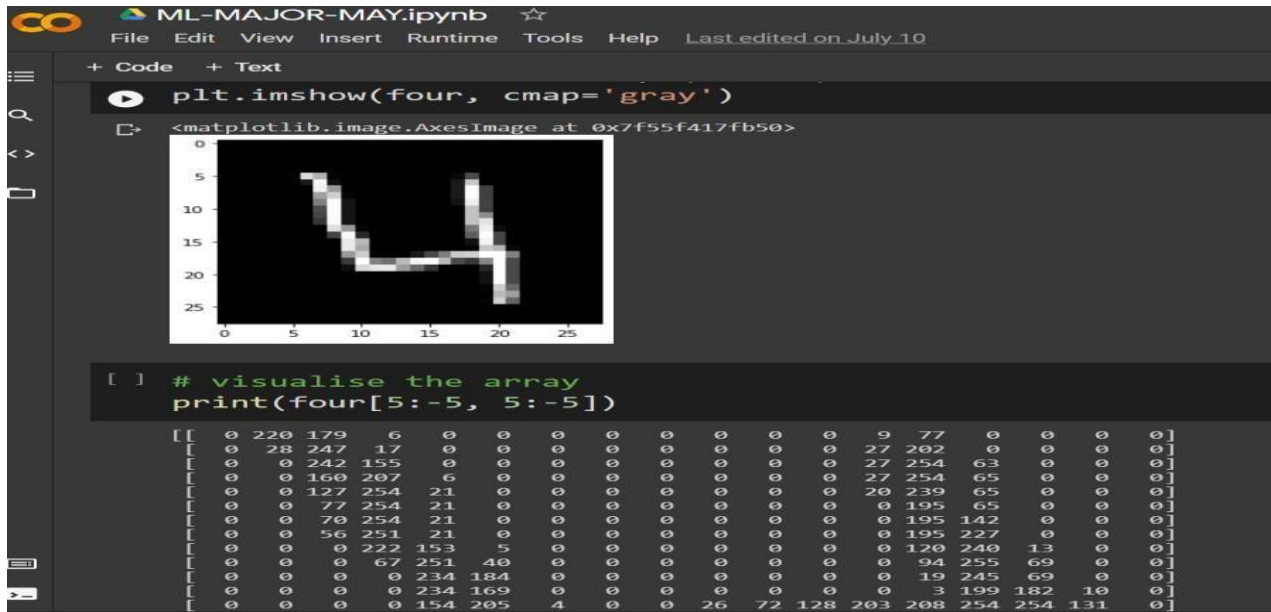
Before the popularity of neural networks, though, models such as SVMs and boosted trees were the state-of-the-art in such problems.

We'll first explore the dataset a bit, prepare it (scale etc.) and then experiment with linear and non-linear SVMs with various hyperparameters.

5. Conclusion

The final accuracy on test data is approx. 92%. Note that this can be significantly increased by using the entire training data of 42,000 images (we have used just 10% of that !). Hence I have to conclude that our model predicted the correct values.

ScreenShots



ML-MAJOR-MAY.ipynb

File Edit View Insert Runtime Tools Help Last edited on July 10

+ Code + Text

```
pixel780 0
pixel781 0
pixel782 0
pixel783 0
length: 785, dtype: int64
```

```
# average values/distributions of features
description = digits.describe()
description
```

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	pixel10	pixel11	pixel12	pixel13	pixel14	pixel15	pixel16
count	42000.000000	42000.0	42000.0	42000.0	42000.0	42000.0	42000.0	42000.0	42000.0	42000.0	42000.0	42000.0	42000.0	42000.000000	42000.000000	42000.000000	42000.000000	42000.0
mean	4.456643	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.003000	0.011190	0.005143	0.000214	0.0
std	2.887730	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.56812	1.626927	1.053972	0.043916	0.0
min	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.0
25%	2.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.0
50%	4.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.0
75%	7.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.0
max	9.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	116.000000	254.000000	216.000000	9.000000	0.0

8 rows x 785 columns

Data Preparation for Model Building

ML-MAJOR-MAY.ipynb

File Edit View Insert Runtime Tools Help Last edited on July 10

+ Code + Text

```
# evaluation: CM
confusion = metrics.confusion_matrix(y_true = y_test, y_pred = predictions)

# measure accuracy
test_accuracy = metrics.accuracy_score(y_true=y_test, y_pred=predictions)

print(test_accuracy, "\n")
print(confusion)
```

```
0.924973544973545
```

```
[[3587  0 10 10  5 15 50 12 25  1]
 [  0 4100 14 16  5  3  6 18 10  5]
 [ 24 23 3407  6 44  5 36 123 54  0]
 [  4 21  86 3502  5 89 11 73 76 33]
 [  3 11 36  7 3450 13 23 43  6 110]
 [ 20 29 14 114 18 3020 79 53 36 35]
 [ 31 12 11  1 14 34 3521 44 25  0]
 [  4 20 27  0 36  7 1 3739 7 97]
 [ 14 59 32 80 22 97 25 44 3251 41]
 [ 23 13 13 50 98  7  0 176 19 3379]]
```

Conclusion

The final accuracy on test data is approx. 92%. Note that this can be significantly increased by using the entire training data of 42,000 images (we have used just 10% of that!).