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

Donbosco Institute Of Technology

Under the guidance of

verzeo

learn here lead anywhere

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).

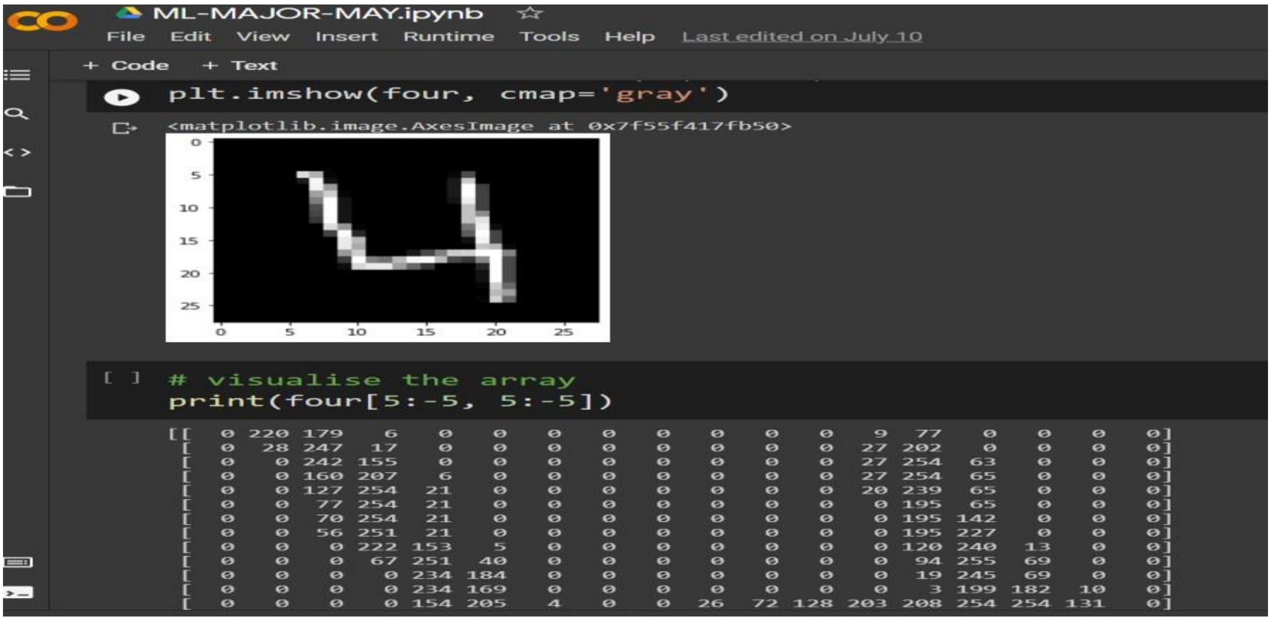
Before the popularity of neural networks, though, models such as SVMs and boosted trees were the state-of-the-art in such problems.

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

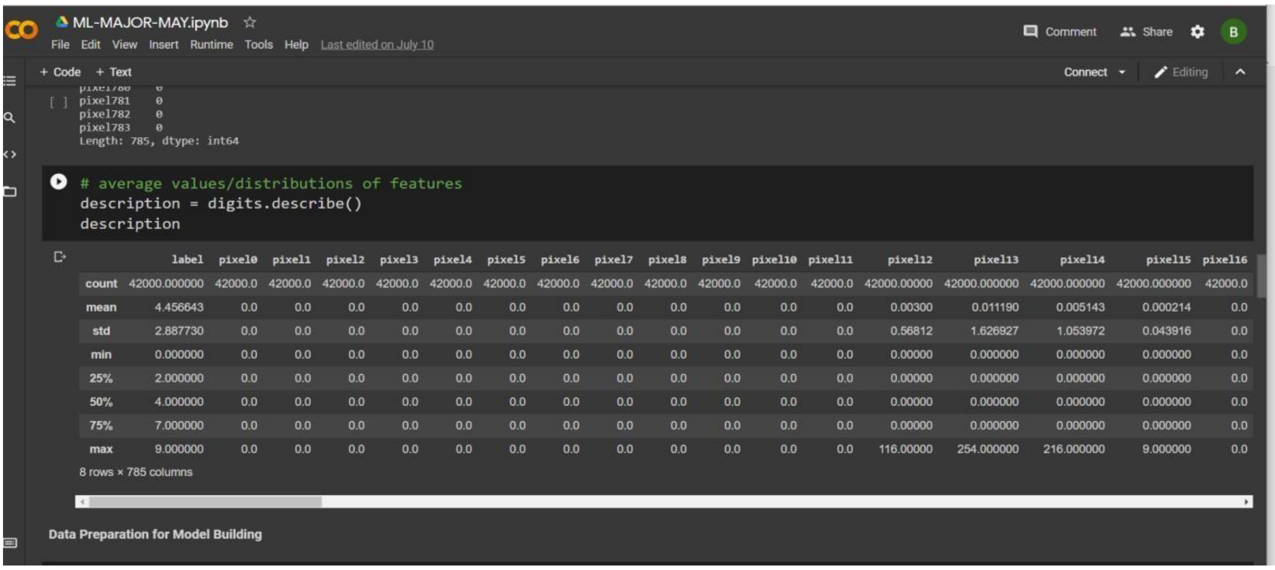

CO
ML-MAJOR-MAY.ipynb ☆
File Edit View Insert Runtime Tools Help Last edited on July 10
+ Code + Text
0
5
10
15
20
25

```
plt.imshow(four, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f55f417fb50>
```

```
# visualise the array
print(four[5:-5, 5:-5])
```

```
[[  0 220 179   6   0   0   0   0   0   0   0   0   9  77   0   0   0   0]
 [  0  28 247  17   0   0   0   0   0   0   0   0  27 202   0   0   0   0]
 [  0   0 242 155   0   0   0   0   0   0   0   0  27 254  63   0   0   0]
 [  0   0 160 207   6   0   0   0   0   0   0   0  27 254  65   0   0   0]
 [  0   0 127 254  21   0   0   0   0   0   0   0  20 239  65   0   0   0]
 [  0   0  77 254  21   0   0   0   0   0   0   0   0 195  65   0   0   0]
 [  0   0  70 254  21   0   0   0   0   0   0   0   0 195 142   0   0   0]
 [  0   0  56 251  21   0   0   0   0   0   0   0   0 195 227   0   0   0]
 [  0   0   0 222 153   5   0   0   0   0   0   0   0 120 240  13   0   0]
 [  0   0   0  67 251  40   0   0   0   0   0   0   0  94 255  69   0   0]
 [  0   0   0   0 234 184   0   0   0   0   0   0   0  19 245  69   0   0]
 [  0   0   0   0 234 169   0   0   0   0   0   0   0   3 199 182  10   0]
 [  0   0   0   0 154 205   4   0   0  26  72 128 203 208 254 254 131   0]
```


CO
ML-MAJOR-MAY.ipynb ☆
File Edit View Insert Runtime Tools Help Last edited on July 10
Comment
Share
B
+ Code + Text
Connect
Editing

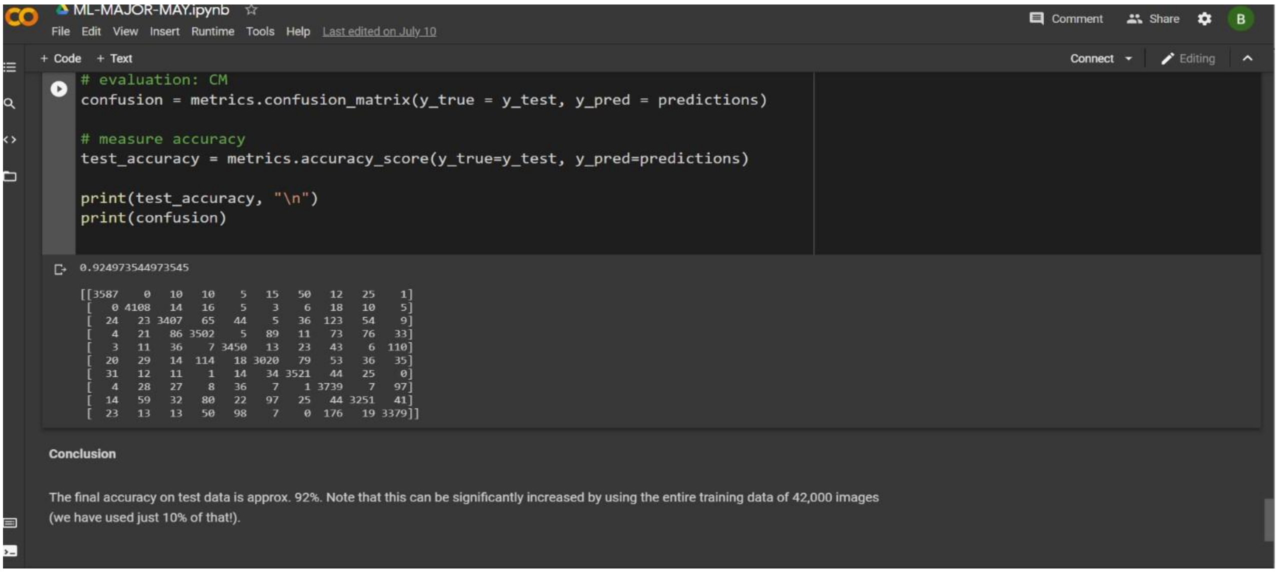
```
pixel780    0
pixel781    0
pixel782    0
pixel783    0
Length: 785, dtype: int64
```

```
# 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	42000.00000	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	0.00300	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	0.00000	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	0.00000	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	0.00000	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	0.00000	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	116.00000	254.000000	216.000000	9.000000	0.0

8 rows × 785 columns

Data Preparation for Model Building


CO
ML-MAJOR-MAY.ipynb ☆
File Edit View Insert Runtime Tools Help Last edited on July 10
Comment
Share
B
+ Code + Text
Connect
Editing

```
# 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 4108   14   16    5    3    6   18   10    5]
 [  24   23 3407   65   44    5   36  123   54    9]
 [   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   28   27    8   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!).