Report

Name: Anowarul Kabir
Date: 24th October, 2019

Data distribution

Number of attributes: 16
Train set: 7494
Test set: 3498
Missing rows/columns: No
Data cleaning: No

System architecture

Operating system: Ubuntu 18.04
Language and library used: Python 3.6, scikit-learn v0.21.3 DecisionTreeClassifier

Github link

The code can be found in the following github link: Link

Scores

The *Table-1* shows accuracy, precision, recall based on several hyper-parameters such as criterion (gini, entropy), splitter (best, random), data-set (train, test), max-depth (none, 10). Best test accuracy is found on entropy, random, none (max-depth) combination. (Note that, max-leaf-nodes hyper-parameter makes the scores worse, that is why it is not mentioned). The green colored row shows the combination of the best accuracy score for test set.

Model	Splitter	Data Set	Max-depth	Accuracy	Precision	Recall
Gini	Best	Train	None	1.0	1.0	1.0
	Best	Test	None	0.917	0.921	0.917
	Random	Train	None	1.0	1.0	1.0
	Random	Test	None	0.913	0.913	0.912
	Best	Train	10	0.991	0.991	0.991
	Best	Test	10	0.909	0.914	0.909
	Random	Train	10	0.971	0.972	0.971
	Random	Test	10	0.900	0.901	0.903
Entropy	Best	Train	None	1.0	1.0	1.0
	Best	Test	None	0.913	0.915	0.913
	Random	Train	None	1.0	1.0	1.0
	Random	Test	None	0.936	0.938	0.937
	Best	Train	10	0.996	0.996	0.996
	Best	Test	10	0.922	0.923	0.922
	Random	Train	10	0.982	0.983	0.982

	Random	Test	10	0.913	0.915	0.914

Table 1: Accuracy, Precision, Recall for different Decision Tree Model with hyper-parameters. Best accuracy found colored in green.

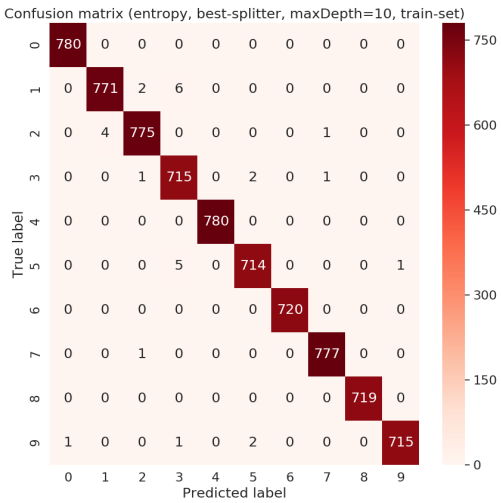
Table-2 shows 5-fold cross validation score on decision tree model. As the accuracy are very close, any model can be chosen as best model.

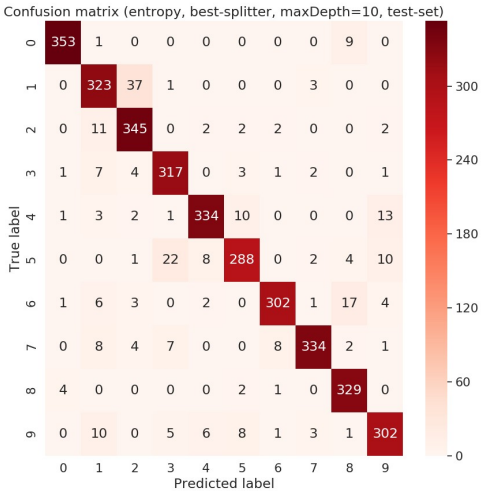
Decision Tree(with below mentioned model)	1st fold	2nd fold	3rd fold	4th fold	5th fold
Gini	0.956	0.964	0.958	0.951	0.965
Entropy	0.959	0.962	0.966	0.965	0.963

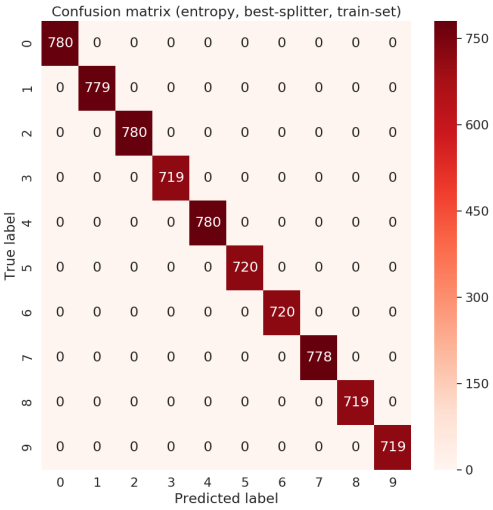
Table 2: 5-fold Cross Validation Score

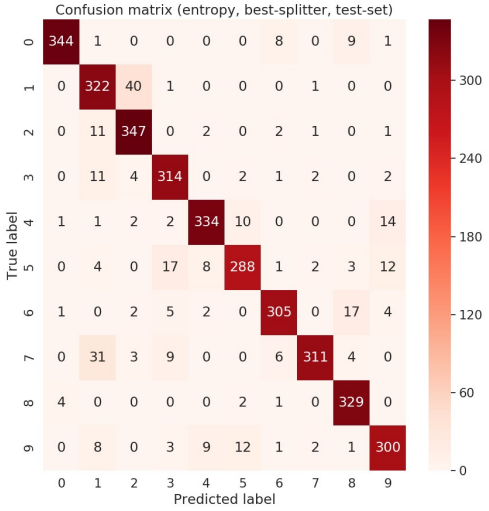
Confusion matrix:

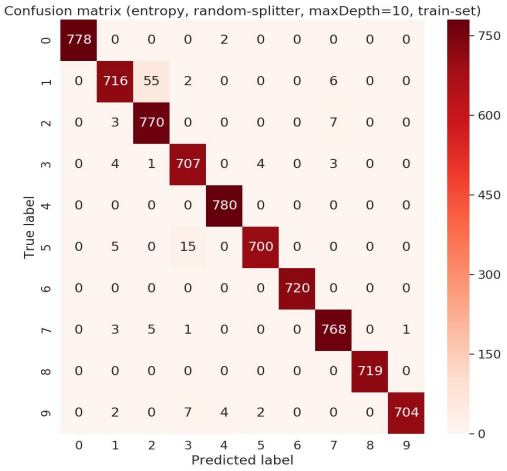
This section shows confusion matrix of corresponding Decision tree classifier with different hyper-parameters. The confusion matrix describes for which class the classifier is predicting wrong class label in a high degree. From the matrices, we can say the classifier is predicting wrong in case 1 vs. 7, 3 vs 5, 5 vs 9 and 4 vs. 9.

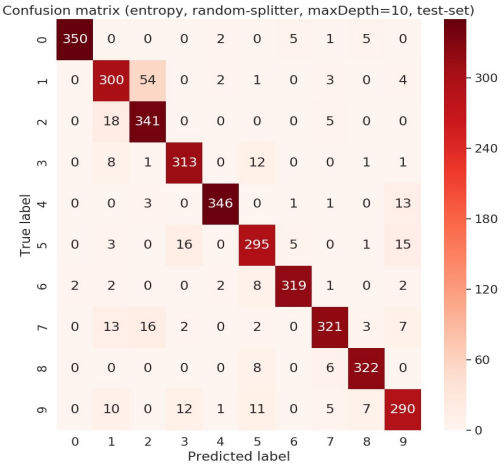


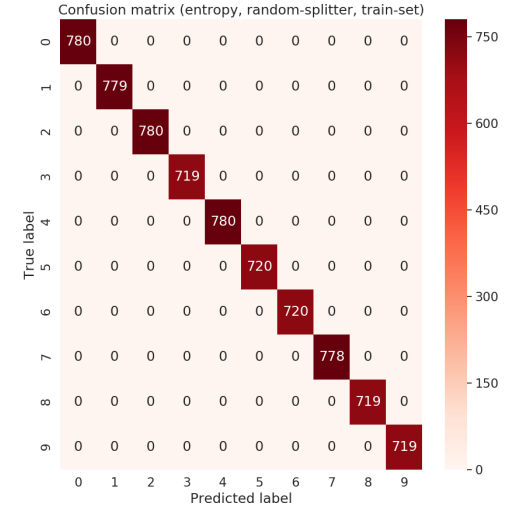


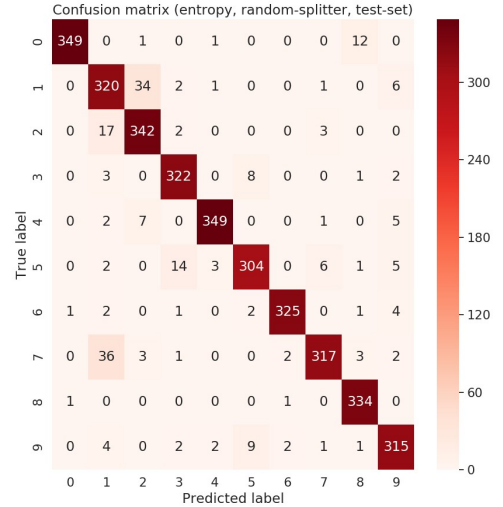

Confusion matrix (entropy, best-splitter, train-set)
True label
Predicted label
0 780 0 0 0 0 0 0 0 0 0
1 0 779 0 0 0 0 0 0 0 0
2 0 0 780 0 0 0 0 0 0 0
3 0 0 0 719 0 0 0 0 0 0
4 0 0 0 0 780 0 0 0 0 0
5 0 0 0 0 0 720 0 0 0 0
6 0 0 0 0 0 0 720 0 0 0
7 0 0 0 0 0 0 0 778 0 0
8 0 0 0 0 0 0 0 0 719 0
9 0 0 0 0 0 0 0 0 0 719
0 1 2 3 4 5 6 7 8 9
750 600 450 300 150 0

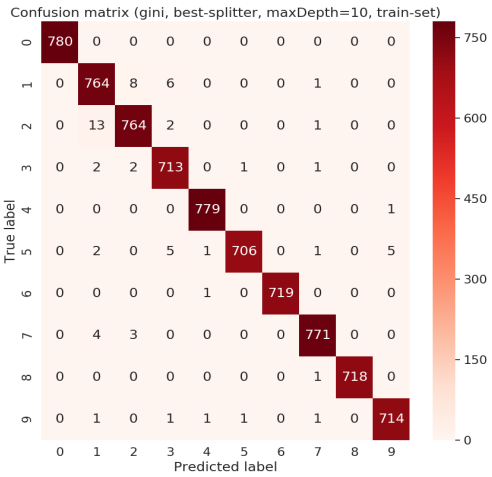

Confusion matrix (entropy, best-splitter, test-set)
True label
Predicted label
0 344 1 0 0 0 0 8 0 9 1
1 0 322 40 1 0 0 0 1 0 0
2 0 11 347 0 2 0 2 1 0 1
3 0 11 4 314 0 2 1 2 0 2
4 1 1 2 2 334 10 0 0 0 14
5 0 4 0 17 8 288 1 2 3 12
6 1 0 2 5 2 0 305 0 17 4
7 0 31 3 9 0 0 6 311 4 0
8 4 0 0 0 0 2 1 0 329 0
9 0 8 0 3 9 12 1 2 1 300
0 1 2 3 4 5 6 7 8 9
300 240 180 120 60 0

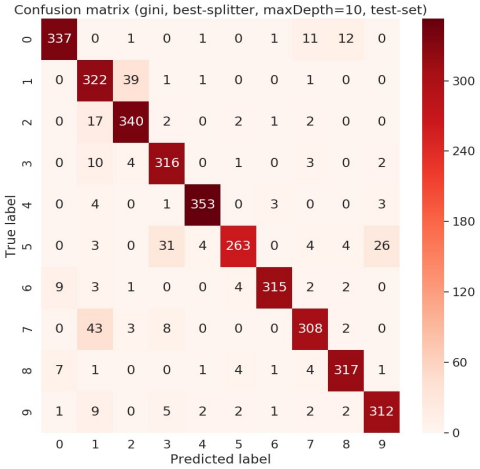

Confusion matrix (entropy, random-splitter, maxDepth=10, train-set)
True label
Predicted label
0 778 0 0 0 2 0 0 0 0 0
1 0 716 55 2 0 0 0 6 0 0
2 0 3 770 0 0 0 0 7 0 0
3 0 4 1 707 0 4 0 3 0 0
4 0 0 0 0 780 0 0 0 0 0
5 0 5 0 15 0 700 0 0 0 0
6 0 0 0 0 0 0 720 0 0 0
7 0 3 5 1 0 0 0 768 0 1
8 0 0 0 0 0 0 0 0 719 0
9 0 2 0 7 4 2 0 0 0 704
0 1 2 3 4 5 6 7 8 9
750 600 450 300 150 0

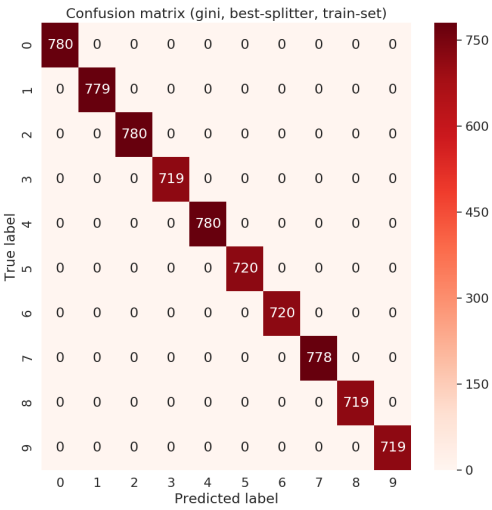

Confusion matrix (entropy, random-splitter, maxDepth=10, test-set)
True label
Predicted label
0 350 0 0 0 2 0 5 1 5 0
1 0 300 54 0 2 1 0 3 0 4
2 0 18 341 0 0 0 0 5 0 0
3 0 8 1 313 0 12 0 0 1 1
4 0 0 3 0 346 0 1 1 0 13
5 0 3 0 16 0 295 5 0 1 15
6 2 2 0 0 2 8 319 1 0 2
7 0 13 16 2 0 2 0 321 3 7
8 0 0 0 0 0 8 0 6 322 0
9 0 10 0 12 1 11 0 5 7 290
0 1 2 3 4 5 6 7 8 9
300 240 180 120 60 0

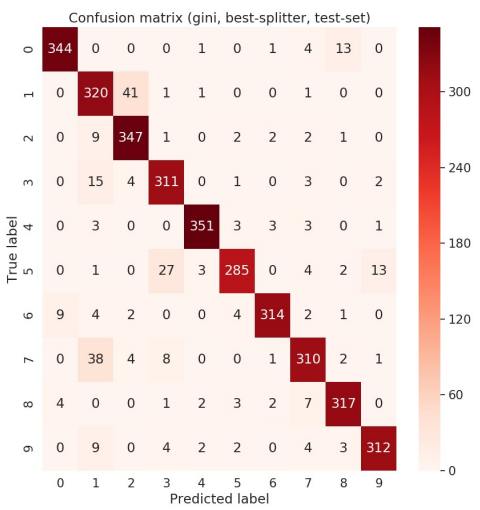

Confusion matrix (entropy, random-splitter, train-set)
True label
Predicted label
0 780 0 0 0 0 0 0 0 0 0
1 0 779 0 0 0 0 0 0 0 0
2 0 0 780 0 0 0 0 0 0 0
3 0 0 0 719 0 0 0 0 0 0
4 0 0 0 0 780 0 0 0 0 0
5 0 0 0 0 0 720 0 0 0 0
6 0 0 0 0 0 0 720 0 0 0
7 0 0 0 0 0 0 0 778 0 0
8 0 0 0 0 0 0 0 0 719 0
9 0 0 0 0 0 0 0 0 0 719
0 1 2 3 4 5 6 7 8 9
750 600 450 300 150 0

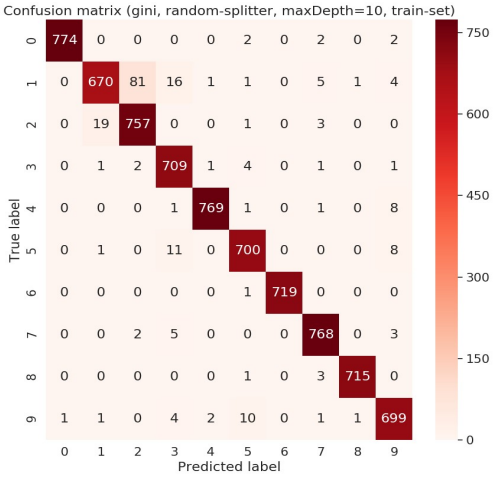

Confusion matrix (entropy, random-splitter, test-set)
True label
Predicted label
0 349 0 1 0 1 0 0 0 12 0
1 0 320 34 2 1 0 0 1 0 6
2 0 17 342 2 0 0 0 3 0 0
3 0 3 0 322 0 8 0 0 1 2
4 0 2 7 0 349 0 0 1 0 5
5 0 2 0 14 3 304 0 6 1 5
6 1 2 0 1 0 2 325 0 1 4
7 0 36 3 1 0 0 2 317 3 2
8 1 0 0 0 0 0 1 0 334 0
9 0 4 0 2 2 9 2 1 1 315
0 1 2 3 4 5 6 7 8 9
300 240 180 120 60 0

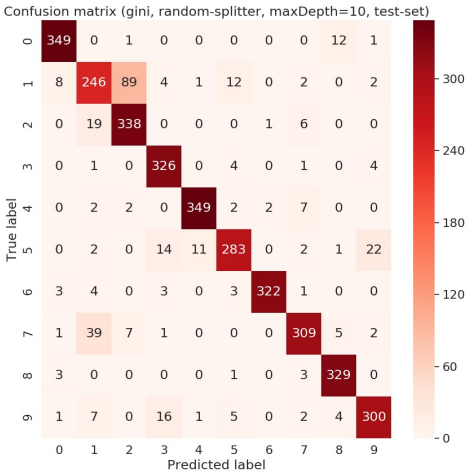

Confusion matrix (gini, best-splitter, maxDepth=10, train-set)
True label
Predicted label
0 780 0 0 0 0 0 0 0 0 0
1 0 764 8 6 0 0 0 1 0 0
2 0 13 764 2 0 0 0 1 0 0
3 0 2 2 713 0 1 0 1 0 0
4 0 0 0 0 779 0 0 0 0 1
5 0 2 0 5 1 706 0 1 0 5
6 0 0 0 0 1 0 719 0 0 0
7 0 4 3 0 0 0 0 771 0 0
8 0 0 0 0 0 0 0 1 718 0
9 0 1 0 1 1 1 0 1 0 714
0 1 2 3 4 5 6 7 8 9
750 600 450 300 150 0

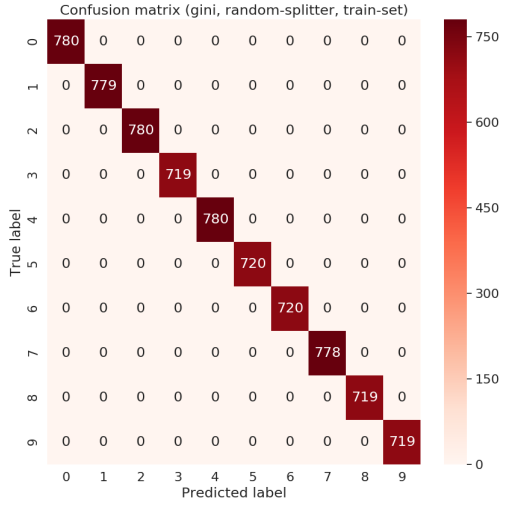

Confusion matrix (gini, best-splitter, maxDepth=10, test-set)
True label
Predicted label
0 337 0 1 0 1 0 1 11 12 0
1 0 322 39 1 1 0 0 1 0 0
2 0 17 340 2 0 2 1 2 0 0
3 0 10 4 316 0 1 0 3 0 2
4 0 4 0 1 353 0 3 0 0 3
5 0 3 0 31 4 263 0 4 4 26
6 9 3 1 0 0 4 315 2 2 0
7 0 43 3 8 0 0 0 308 2 0
8 7 1 0 0 1 4 1 4 317 1
9 1 9 0 5 2 2 1 2 2 312
0 1 2 3 4 5 6 7 8 9
300 240 180 120 60 0

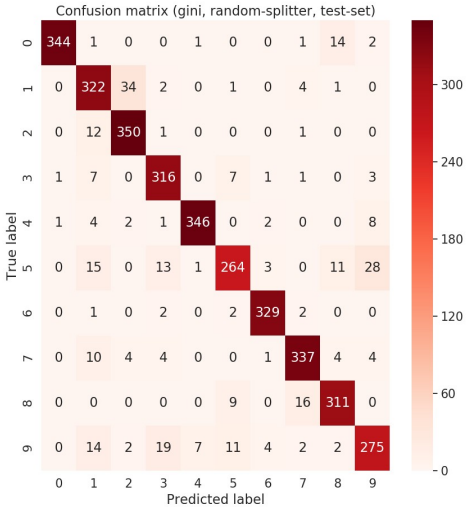

Confusion matrix (gini, best-splitter, train-set)
True label
Predicted label
0 780 0 0 0 0 0 0 0 0 0
1 0 779 0 0 0 0 0 0 0 0
2 0 0 780 0 0 0 0 0 0 0
3 0 0 0 719 0 0 0 0 0 0
4 0 0 0 0 780 0 0 0 0 0
5 0 0 0 0 0 720 0 0 0 0
6 0 0 0 0 0 0 720 0 0 0
7 0 0 0 0 0 0 0 778 0 0
8 0 0 0 0 0 0 0 0 719 0
9 0 0 0 0 0 0 0 0 0 719
0 1 2 3 4 5 6 7 8 9
750 600 450 300 150 0


Confusion matrix (gini, best-splitter, test-set)
True label
Predicted label
0 344 0 0 0 1 0 1 4 13 0
1 0 320 41 1 1 0 0 1 0 0
2 0 9 347 1 0 2 2 2 1 0
3 0 15 4 311 0 1 0 3 0 2
4 0 3 0 0 351 3 3 3 0 1
5 0 1 0 27 3 285 0 4 2 13
6 9 4 2 0 0 4 314 2 1 0
7 0 38 4 8 0 0 1 310 2 1
8 4 0 0 1 2 3 2 7 317 0
9 0 9 0 4 2 2 0 4 3 312
0 1 2 3 4 5 6 7 8 9
300 240 180 120 60 0


Confusion matrix (gini, random-splitter, maxDepth=10, train-set)
True label
Predicted label
0 1 2 3 4 5 6 7 8 9
774 0 0 0 0 2 0 2 0 2
0 670 81 16 1 1 0 5 1 4
0 19 757 0 0 1 0 3 0 0
0 1 2 709 1 4 0 1 0 1
0 0 0 1 769 1 0 1 0 8
0 1 0 11 0 700 0 0 0 8
0 0 0 0 0 1 719 0 0 0
0 0 2 5 0 0 0 768 0 3
0 0 0 0 0 1 0 3 715 0
1 1 0 4 2 10 0 1 1 699
750 600 450 300 150 0


Confusion matrix (gini, random-splitter, maxDepth=10, test-set)
True label
Predicted label
0 1 2 3 4 5 6 7 8 9
349 0 1 0 0 0 0 0 12 1
8 246 89 4 1 12 0 2 0 2
0 19 338 0 0 0 1 6 0 0
0 1 0 326 0 4 0 1 0 4
0 2 2 0 349 2 2 7 0 0
0 2 0 14 11 283 0 2 1 22
3 4 0 3 0 3 322 1 0 0
1 39 7 1 0 0 0 309 5 2
3 0 0 0 0 1 0 3 329 0
1 7 0 16 1 5 0 2 4 300
300 240 180 120 60 0


Confusion matrix (gini, random-splitter, train-set)
True label
Predicted label
0 1 2 3 4 5 6 7 8 9
780 0 0 0 0 0 0 0 0 0
0 779 0 0 0 0 0 0 0 0
0 0 780 0 0 0 0 0 0 0
0 0 0 719 0 0 0 0 0 0
0 0 0 0 780 0 0 0 0 0
0 0 0 0 0 720 0 0 0 0
0 0 0 0 0 0 720 0 0 0
0 0 0 0 0 0 0 778 0 0
0 0 0 0 0 0 0 0 719 0
0 0 0 0 0 0 0 0 0 719
750 600 450 300 150 0


Confusion matrix (gini, random-splitter, test-set)
True label
Predicted label
0 1 2 3 4 5 6 7 8 9
344 1 0 0 1 0 0 1 14 2
0 322 34 2 0 1 0 4 1 0
0 12 350 1 0 0 0 1 0 0
1 7 0 316 0 7 1 1 0 3
1 4 2 1 346 0 2 0 0 8
0 15 0 13 1 264 3 0 11 28
0 1 0 2 0 2 329 2 0 0
0 10 4 4 0 0 1 337 4 4
0 0 0 0 0 9 0 16 311 0
0 14 2 19 7 11 4 2 2 275
300 240 180 120 60 0

DM1.2

Data normalization

In Pen digit dataset [1], each image is 100x100 (gray scale), however in our MNIST each image 28x28 (gray scale) where as 20x20 is the actual container that contains the full image. For this reason we need to normalize the dimension of the image from 20x20 to 100x100. To do that, I leverage bilinear interpolation which is natural choice in compare to others (two other interpolation choices are cubic and nearest). Some examples from initial (28x28) and normalized image (100x100) of MNIST are as follows:

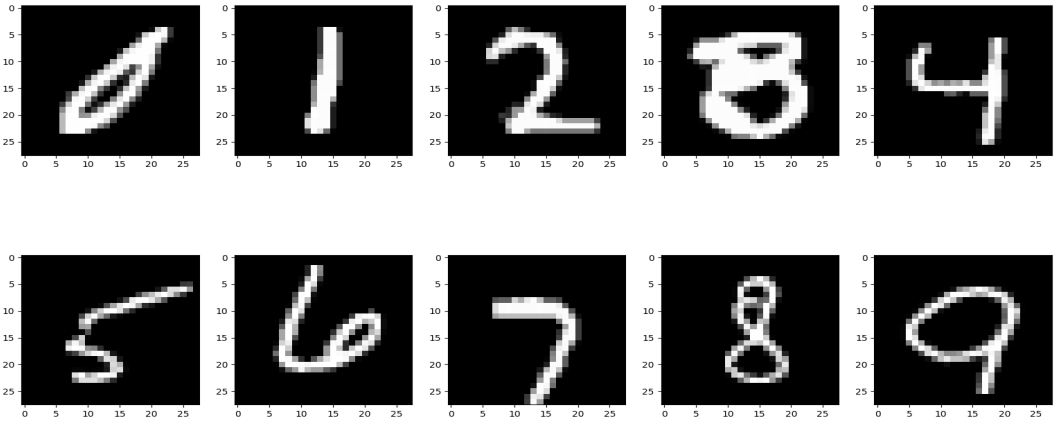


IMAGE 1: MNIST (28X28)

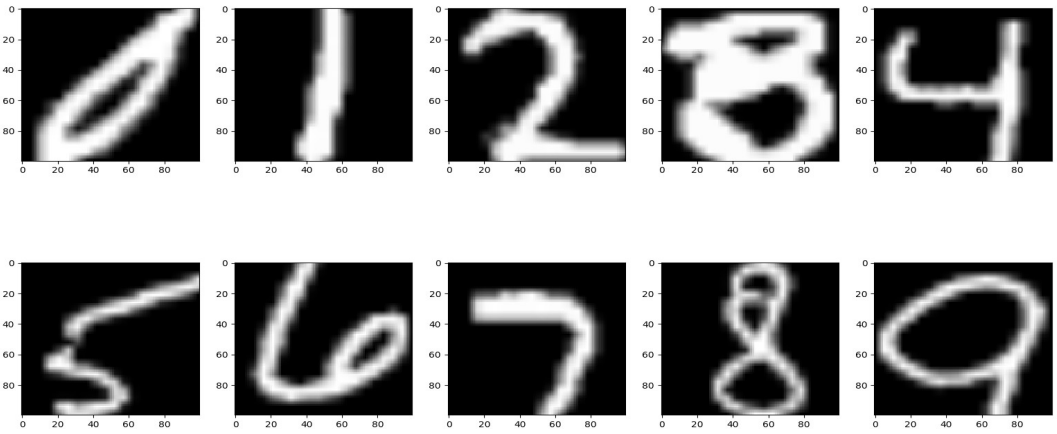


IMAGE 2: Normalized (100x100)

Data Sampling
We need to sample 8 regularly spaced (x,y) coordinates along the trajectory of a digit. To do that, I follows several heuristic methods. The are as follows:

Sampling Method 1 (SM1):
1. Choose a uniformly random point (x, y) a
2. If the intensity value of a is 0 reject it
3. else keep a as a point
4. Start from step-1 until we get 8 such points
5. Start from step-1 until we sample 50 feature vectors for each class

This random 8 points (x, y) are randomly uniformly captured from 100x100 image grid. Since every point chosen from the grid (either accepted or rejected) is uniformly distributed, the accepted 8 points all together follow uniformly random distribution. There are no guarantee that these points are evenly spaced. Even the base Pen Digit [1] article does not describe how they resample such evenly spaced 8 points.

Sampling Method 2 (SM2):
1. Divide 100x100 image into 4 grids as like *Drawing 1*
2. Select uniformly random two points (x1, y1, x2, y2) in a grid such that the intensity value is not 0
3. Do step-2 for 4 grids such that we can sample 8 points which are evenly spaces into 4 grids
4. Start from step-1 until we sample 50 feature vectors for each class

In this method we can say, four pairs of points are evenly spaced and uniformly randomly distributed.

Drawing 1: 4 Grid division

Sampling Method 3 (SM3):
1. Divide 100x100 image into 4 grids as like *Drawing 1*
2. Select uniformly random two points (x1, y1, x2, y2) in a grid such that the intensity value is not 0
3. Maximize distance between (x1, y1) and (x2, y2) for some β times. Throughout this work I have used β=10
4. Do step-2 and step-3 for 4 grids such that we can sample 8 points
5. Start from step-1 until we sample 50 feature vectors for each class

Sampling Method 4 (SM4):
1. Divide 100x100 image into 4 grids as like *Drawing 1*
2. Find two points (x1, y1) and (x2, y2) for which intensity value is not 0 and the maximum distance is guaranteed. I call this as *Sequential Search for two points* that are located in the maximum distance corner of the grid
3. Do step-2 for 4 grids such that we can sample 8 points
4. Start from step-1 until we sample 50 feature vectors for each class

This is the slowest sampling method.

Table 3 shows accuracy, precision and recall score on sampled data which are sampled using SM1, SM2, SM3 and SM4.

Selected Model	Sampling Method	Accuracy	Precision	Recall
Gini (Best)	SM1	0.094	0.089	0.094
	SM2	0.078	0.090	0.078
	SM3	0.112	0.122	0.111
	SM4	0.144	0.142	0.144
Gini (Random)	SM1	0.088	0.095	0.088
	SM2	0.07	0.041	0.07
	SM3	0.09	0.046	0.09
	SM4	0.076	0.037	0.076
Entropy (Best)	SM1	0.094	0.068	0.094
	SM2	0.1	0.076	0.1
	SM3	0.1	0.183	0.1
	SM4	0.128	0.111	0.128
Entropy (Random)	SM1	0.102	0.076	0.102
	SM2	0.08	0.048	0.08
	SM3	0.08	0.03	0.08
	SM4	0.106	0.06	0.106

Table 3: Accuray, Precision & Recall on sampled data

Analysis:
In a obvious manner, the first question comes into mind why such low scores over the sampled data. To answer this question, I read through the Preprocessing chapter of the Pen Digit [1] paper which consists of dynamic representation and static representation. In our work, we leverage dynamic representation which consists of normalization and resampling. Following normalization, I normalized 28x28 MNIST image to 100x100 image using bilinear interpolation which is natural interpolation method compared to nearest and cubic interpolation. So the issue resides in resampling methods. In the article [1] they have used spatial resampling method to collect evenly spaced 8 points through the arc of the digit. But they did not describe whether this resampling technique is manual (hand-made features) or automatic. They did not describe any algorithm on spatial resampling technique, instead they refer to other papers which I did not go through because of the scope of the assignment. Instead I followed some heuristic techniques as described above.

To push a little bit further, I resampled from the resampled data based on best recall scores. I took a set of data of a class for which SM methods generate best recall score. Which resampled data is taken on which SM method and what are the best recall score are shown in Table-4.

Class	Data size	Sampling Method (SM)	Best recall score on corresponding class only
0	50	SM2	0.75
1	50	SM4	0.36
2	50	SM3	0.25
3	50	SM2	0.12
4	50	SM2	0.06
5	50	SM4	0.18
6	50	SM4	0.40
7	50	SM4	0.44
8	50	SM1	0.65
9	50	SM8	0.39

Table 4: Best recall score using SMx

One important analysis is that if we take a look at the recall score of the class 2, 3, 4, 5; the corresponding recall score is very low in compared to others. From here we can conclude that these four classes are basically responsible for such low accuracy score.

Table-5 shows the final accuracy, precision and recall where I used the new resampled data and increased the accuracy more that by a factor of 2. The best accuracy is colored in green.

Model Selection	Best data (based on recall)	Accuracy	Precision	Recall
Gini (Best)	Best combination	0.258	0.222	0.258
Gini (Random)	Best combination	0.12	0.072	0.12
Entropy (Best)	Best combination	0.244	0.210	0.244
Entropy (Random)	Best combination	0.114	0.054	0.114

References:

1. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.25.6299&rep=rep1&type=pdf