

# MTL782: Data Mining

## Assignment 2

Archisman Biswas  
2021MT10254

Atul  
2021MT10623

Harshit Goyal  
2021MT10143

April 20, 2024

### Contents

<b>1</b>	<b>Data</b>	<b>2</b>
<b>2</b>	<b>Evaluation Metrics</b>	<b>2</b>
<b>3</b>	<b>Models with default settings</b>	<b>4</b>
3.1	Decision Tree . . . . .	4
3.2	Random Forest . . . . .	4
3.3	Naïve Bayes Classifier . . . . .	4
3.4	KNN Classifier . . . . .	4
3.5	Neural Network Classifier . . . . .	5
<b>4</b>	<b>k-Fold Cross Validation</b>	<b>6</b>
<b>5</b>	<b>Confusion Matrices</b>	<b>7</b>
<b>6</b>	<b>Hyperparameter Tuning</b>	<b>10</b>
6.1	Grid Search . . . . .	10
6.1.1	Decision Tree Classifier . . . . .	10
6.1.2	Random Forest Classifier . . . . .	10
6.1.3	Naïve Bayes Classifier . . . . .	10
6.1.4	KNN Classifier . . . . .	11
6.1.5	Neural Network Classifier . . . . .	11
6.2	Conclusion . . . . .	11

## 1 Data

The [MNIST](#) dataset is loaded using [TensorFlow's Keras](#). `tf.keras.datasets.mnist` gives the split into `train` and `test` already.

The pixel values of the images are **normalized** by dividing by 255. This scales the pixel values to a range between 0 and 1, where 0 represents black, 1 represents white, and values in between represent various shades of gray. Since the image is black and white, it has only 1 channel.

Training size	60000
Testing size	10000
Shape of each example (original)	$28 \times 28 \times 1$
Shape of each example (flattened)	784
Number of classes	10

Table 1: Dataset Details

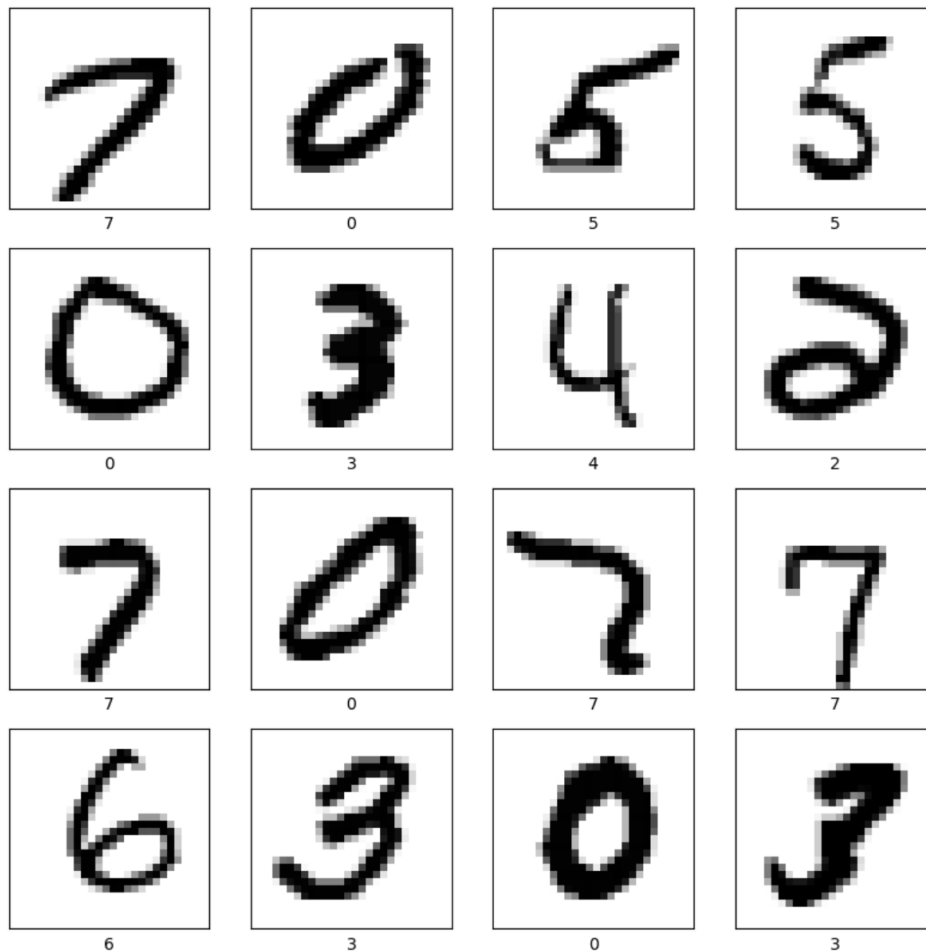


Figure 1: 16 images from the training data

## 2 Evaluation Metrics

The accuracy different models used to fit the above data is evaluated using different metrics which are stated as follows:

- **Accuracy:** This is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations

$$\text{Accuracy} = \frac{\text{Total Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

- **Precision (Macro Average):** Precision is the ratio of correctly predicted positive observations to the total predicted positive observations

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

- **Recall (Macro Average):** Recall (also known as sensitivity) is the ratio of correctly predicted positive observations to all observations in actual class

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

- **F1 Score (Macro Average):** The F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

### 3 Models with default settings

#### 3.1 Decision Tree

Library function used: `DecisionTreeClassifier` from `sklearn.tree`

Parameter	Default Value
<code>max_depth</code>	None
<code>min_samples_leaf</code>	1
<code>min_samples_split</code>	2

Table 2: Default parameters

Metric	Train	Test
Accuracy	100.0%	87.63%
Precision (macro)	100.0%	87.5%
Recall (macro)	100.0%	87.47%
F1-Score (macro)	100.0%	87.48%

Table 3: Performance Metrics

It is clear that Decision Trees cause overfitting, here especially because we didn't limit the depth of the tree.

#### 3.2 Random Forest

Library function used: `RandomForestClassifier` from `sklearn.ensemble`

Parameter	Default Value
<code>max_depth</code>	None
<code>min_samples_leaf</code>	1
<code>min_samples_split</code>	2
<code>n_estimators</code>	100

Table 4: Default parameters

Metric	Train	Test
Accuracy	100.0%	96.9%
Precision (macro)	100.0%	96.88%
Recall (macro)	100.0%	96.87%
F1-Score (macro)	100.0%	96.87%

Table 5: Performance Metrics

Random Forests give better test metrics than Decision Trees. Its important to note that they take longer to train.

#### 3.3 Naïve Bayes Classifier

Library function used: `MultinomialNB` from `sklearn.naive_bayes`

Parameter	Default Value
No parameters to tune	N/A

Table 6: Default parameters

Metric	Train	Test
Accuracy	82.46%	83.57%
Precision (macro)	83.15%	84.29%
Recall (macro)	82.16%	83.26%
F1-Score (macro)	82.32%	83.42%

Table 7: Performance Metrics

Naïve Bayes performs poorest. This is likely under-fitting. For image data, the **naïve** assumption fails miserably as pixel values are highly correlated when near.

#### 3.4 KNN Classifier

Library function used: `KNeighborsClassifier` from `sklearn.neighbors`

Parameter	Default Value
n_neighbors	5
weights	uniform

Table 8: Default parameters

Metric	Train	Test
Accuracy	98.19%	96.88%
Precision (macro)	98.22%	96.93%
Recall (macro)	98.16%	96.85%
F1-Score (macro)	98.19%	96.87%

Table 9: Performance Metrics

### 3.5 Neural Network Classifier

Library function used: [Layer](#), [Model](#) from `tensorflow.keras`

Parameter	Default Value
activation	relu
alpha	0.0001
hidden_layer_sizes	100
epochs	5

Table 10: Default parameters

Metric	Train	Test
Accuracy	99.12%	97.62%
Precision (macro)	99.13%	97.63%
Recall (macro)	99.12%	97.60%
F1-Score (macro)	99.12%	97.60%

Table 11: Performance Metrics

One **epoch** is one single run over the entire training data. Neural Network, with these settings give the best **test** accuracy in models shown above.

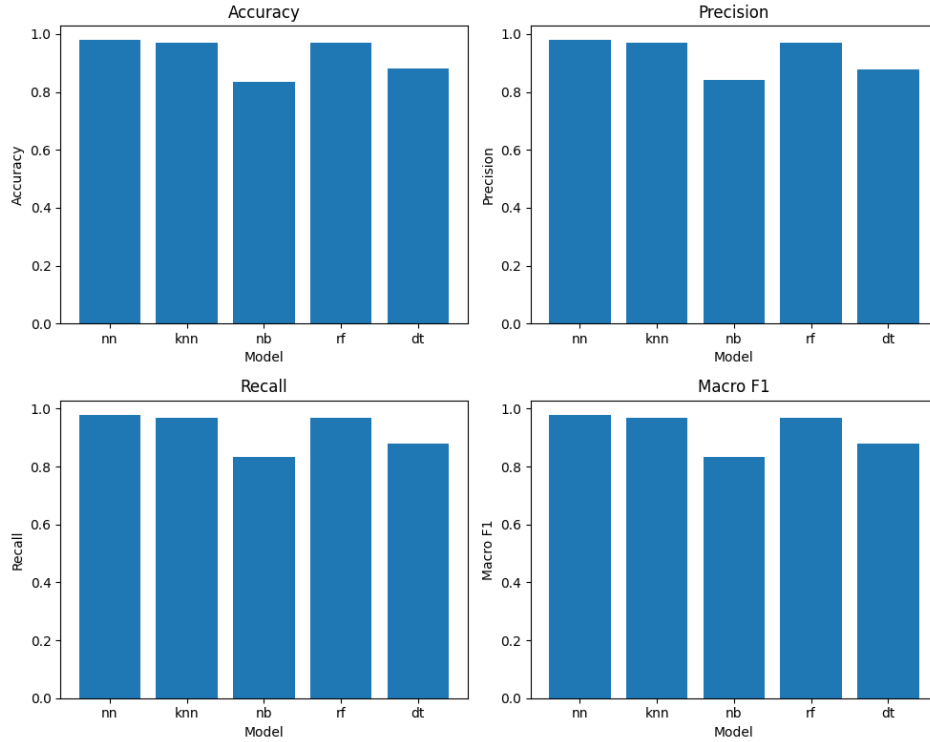


Figure 2: Metrics on test set for various models

## 4 k-Fold Cross Validation

This approach involves randomly dividing the set of observations into  $k$  groups, or folds, of approximately equal size. The first fold is treated as a validation set, and the method is fit on the remaining  $k - 1$  folds.

Library function used: `cross_val_score` from `sklearn.model_selection`

Note that the model hyperparameters used here are in their default settings

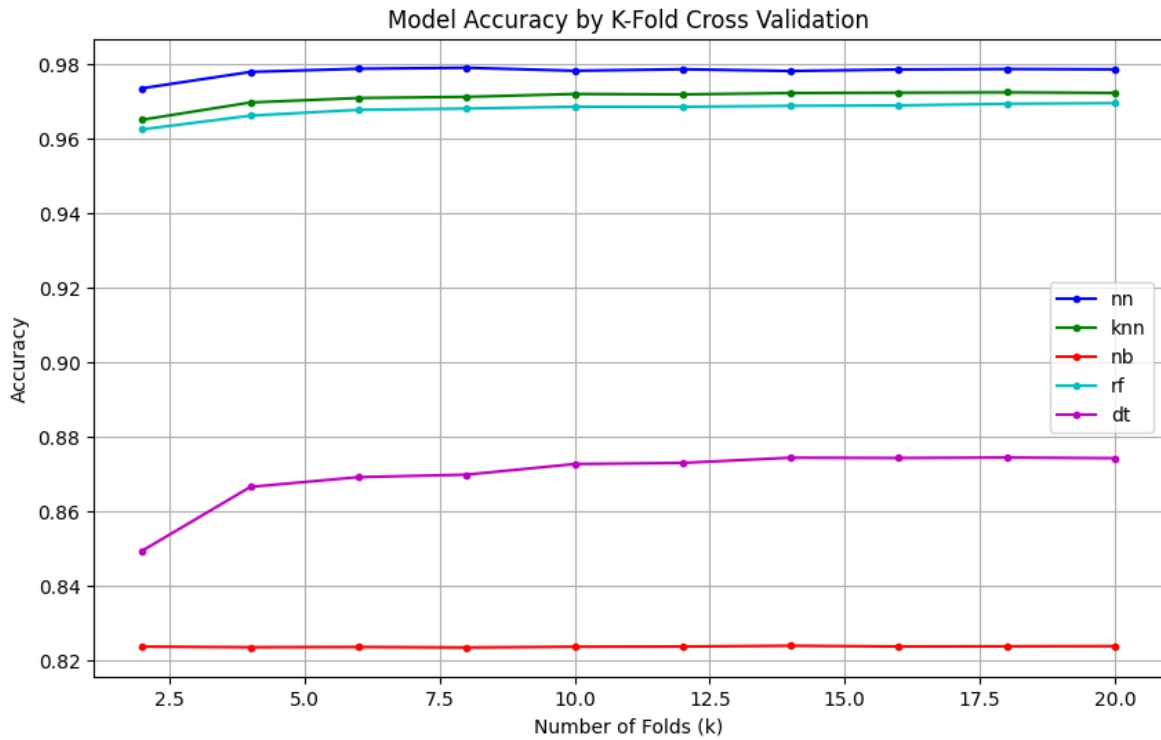


Figure 3: k-Fold Cross Validation on all models

## 5 Confusion Matrices

Figures [Figure 4](#), [Figure 5](#), [Figure 6](#), [Figure 7](#), [Figure 8](#) show the confusion matrices, for various classifiers. The diagonal entries, show the number correctly classified examples.

Confusion Matrix

0	5458	7	81	43	31	93	86	16	75	33
1	5	6441	51	34	22	45	19	40	68	17
2	64	74	5067	165	78	67	110	126	159	48
3	60	62	177	5103	52	258	33	99	167	120
4	38	28	67	40	5101	58	84	75	96	255
5	92	50	66	233	67	4466	116	47	160	124
6	93	32	99	37	89	107	5308	16	110	27
7	12	40	111	85	83	37	14	5687	50	146
8	75	82	133	203	92	167	104	77	4740	178
9	33	24	49	115	286	128	20	154	142	4998
	0	1	2	3	4	5	6	7	8	9

Predicted Labels

Figure 4: Confusion Matrix for Decision Tree

Confusion Matrix

0	5841	1	10	3	5	8	17	1	32	5
1	1	6641	42	12	13	3	4	12	10	4
2	19	7	5786	22	25	3	18	37	36	5
3	7	5	84	5823	2	70	7	45	59	29
4	9	9	8	3	5673	0	24	9	11	96
5	22	6	8	64	12	5205	48	4	29	23
6	27	11	3	1	7	39	5812	0	18	0
7	5	22	57	7	31	0	0	6061	8	74
8	9	30	26	45	28	38	27	6	5588	54
9	22	10	10	73	65	17	5	51	43	5653
	0	1	2	3	4	5	6	7	8	9

Predicted Labels

Figure 5: Confusion Matrix for Random Forest

True Labels	0	5361	4	30	18	12	84	100	0	309	5
	1	0	6305	46	41	6	28	34	3	262	17
	2	79	82	4925	132	65	8	276	46	328	17
	3	34	118	297	4944	11	126	57	47	297	200
	4	19	20	38	5	4336	16	120	13	232	1043
	5	159	47	47	655	110	3575	131	12	481	204
	6	62	115	84	3	31	131	5409	0	82	1
	7	46	105	42	24	151	1	7	5145	205	539
	8	33	334	95	317	74	213	49	14	4493	229
	9	49	53	21	77	402	18	3	168	241	4917
		Predicted Labels									
		0	1	2	3	4	5	6	7	8	9

Figure 6: Confusion Matrix for Naive Bayes

Confusion Matrix											
True Labels	0	5884	3	4	1	0	6	16	3	2	4
	1	2	6709	8	2	4	0	2	13	1	1
	2	35	53	5752	9	9	4	9	70	11	6
	3	5	14	37	5931	1	57	3	33	27	23
	4	3	48	2	1	5654	0	13	9	1	111
	5	17	7	4	63	7	5222	57	8	9	27
	6	23	13	2	0	5	19	5855	0	1	0
	7	2	60	15	2	14	1	0	6109	2	60
	8	18	68	18	69	31	88	26	16	5465	52
	9	13	11	5	43	56	10	3	70	11	5727
		Predicted Labels									
		0	1	2	3	4	5	6	7	8	9

Figure 7: Confusion Matrix for KNN Classifier



		Confusion Matrix									
True Labels	0	5849	1	13	3	3	12	14	5	13	10
	1	1	6663	15	12	10	2	4	15	18	2
	2	17	11	5817	27	15	2	9	32	23	5
	3	5	4	37	5932	1	50	0	23	50	29
	4	4	13	9	2	5727	1	13	13	7	53
	5	9	2	6	51	6	5266	27	7	26	21
	6	17	6	5	1	9	21	5841	1	16	1
	7	1	12	31	4	21	5	1	6146	9	35
	8	10	25	26	31	9	24	14	10	5663	39
	9	11	2	3	19	50	23	2	43	18	5778
		0	1	2	3	4	5	6	7	8	9
		Predicted Labels									

Figure 8: Confusion Matrix for Neural Network Classifier

## 6 Hyperparameter Tuning

### 6.1 Grid Search

In Grid Search, the domain of the hyperparameters is divided into a discrete grid. Then for every combination of values of this grid, [accuracy](#) is computed using k-fold cross-validation. The point of the grid that maximizes the average value in cross-validation, is the optimal combination of values for the hyperparameters.

Library function used: [GridSearchCV](#) from `sklearn.model_selection`

#### 6.1.1 Decision Tree Classifier

Parameter	Value
max_depth	15
min_samples_leaf	1
min_samples_split	2

Table 12: Best parameters

Metric	Default	Grid Search
Accuracy	87.63%	88.08%
Precision (macro)	87.5%	87.9261%
Recall (macro)	87.47%	87.9202%
F1-Score (macro)	87.48%	87.9156%

Table 13: Performance Metrics on test set

Time taken for grid search: 605.20 seconds.

#### 6.1.2 Random Forest Classifier

Parameter	Value
max_depth	None
min_samples_leaf	1
min_samples_split	2
n_estimators	150

Table 14: Best parameters

Metric	Default	Grid Search
Accuracy	96.9%	97.11%
Precision (macro)	96.88%	97.10%
Recall (macro)	96.87%	96.89%
F1-Score (macro)	96.87%	96.89%

Table 15: Performance Metrics on test set

Time taken for grid search: 4151.61 seconds.

#### 6.1.3 Naïve Bayes Classifier

Parameter	Value
No parameters to tune	

Table 16: Best Parameters

Metric	Default	Grid Search
Accuracy	83.57%	83.57%
Precision (macro)	84.29%	84.29%
Recall (macro)	83.26%	83.26%
F1-Score (macro)	83.42%	83.42%

Table 17: Performance Metrics on test set

Time taken for grid search: 1.25 seconds.

#### 6.1.4 KNN Classifier

Parameter	Value
n_neighbors	3
weights	distance

Table 18: Best parameters

Metric	Default	Grid Search
Accuracy	96.88%	97.17%
Precision (macro)	96.93%	97.19%
Recall (macro)	96.85%	97.13%
F1-Score (macro)	96.87%	97.15%

Table 19: Performance Metrics on test set

Time taken for grid search: 284.33 seconds.

#### 6.1.5 Neural Network Classifier

Parameter	Value
activation	relu
alpha	0.001
hidden_layer_sizes	300
epochs	5

Table 20: Best parameters

Metric	Default	Grid Search
Accuracy	97.62%	98.17%
Precision (macro)	97.63%	98.17%
Recall (macro)	97.60%	98.16%
F1-Score (macro)	97.60%	98.16%

Table 21: Performance Metrics on test set

Time taken for grid search: 3731.32 seconds.

We did not tune the epoch parameter for the Neural Network.

## 6.2 Conclusion

Model	Default	Grid Search
Decision Tree	87.63%	88.08%
Random Forest	96.9%	97.11%
Naïve Bayes	83.57%	83.57%
KNN Classifier	96.88%	97.17%
Neural Network	97.62%	98.17%

Table 22: Accuracy on test set tuned using default settings and Grid Searched hyperparameters

All models except Naïve Bayes show an improvement when parameters obtained from Grid Search are used. There is no improvement on Naïve Bayes as it does not take any hyper-parameters.