MTL782: Data Mining
Assignment 2

Archisman Biswas	Atul	Harshit Goyal
2021MT10254	2021MT10623	2021MT10143

April 20, 2024

Contents

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

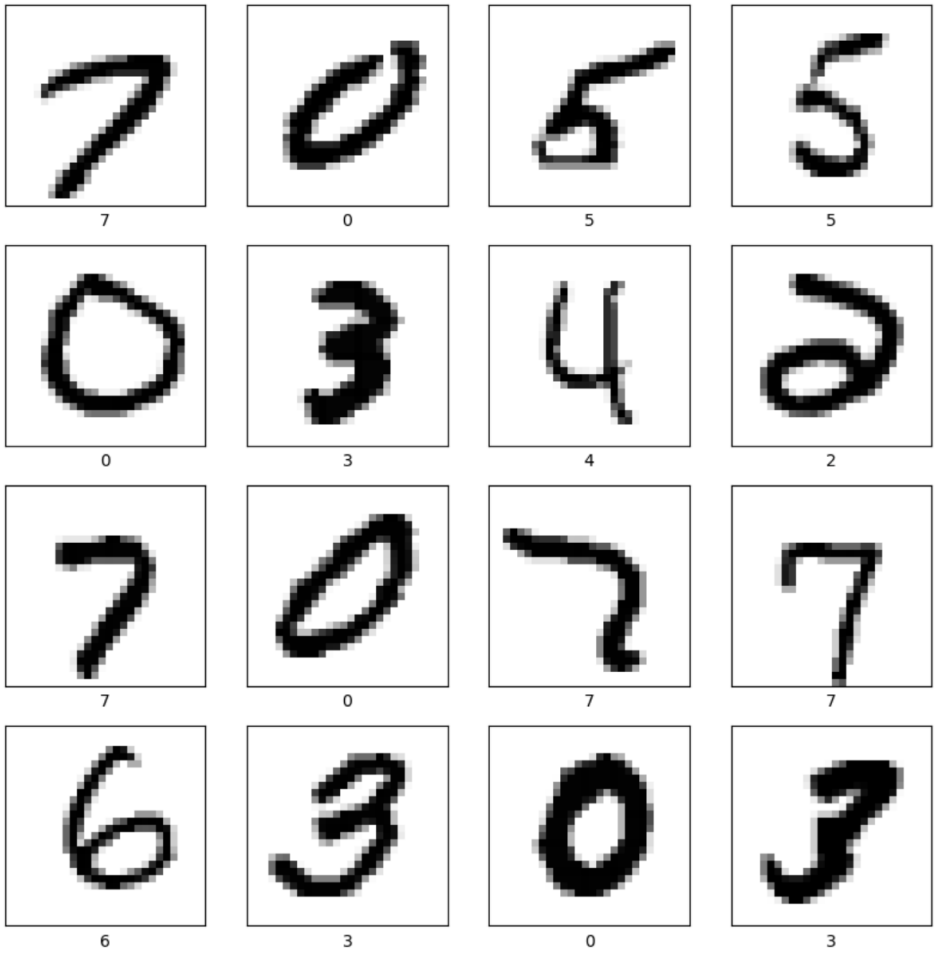

Figure 1: 16 images from the training data

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
`max_depth`	None
`min_samples_leaf`	1
`min_samples_split`	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
`max_depth`	None
`min_samples_leaf`	1
`min_samples_split`	2
`n_estimators`	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 **naive** 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.

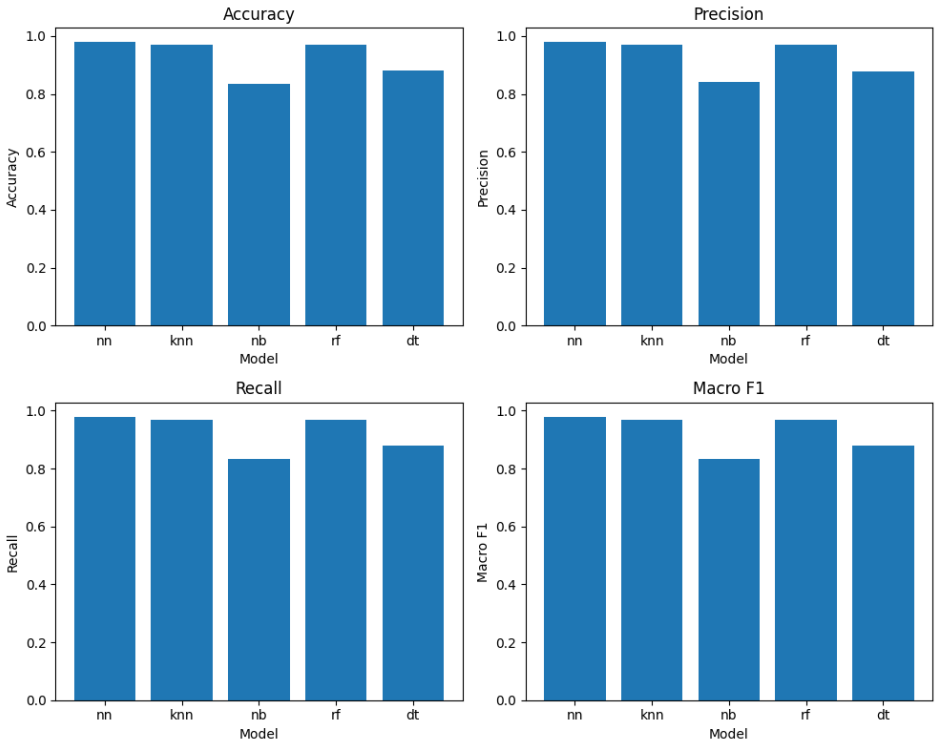

Figure 2: Metrics on test set for various models

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

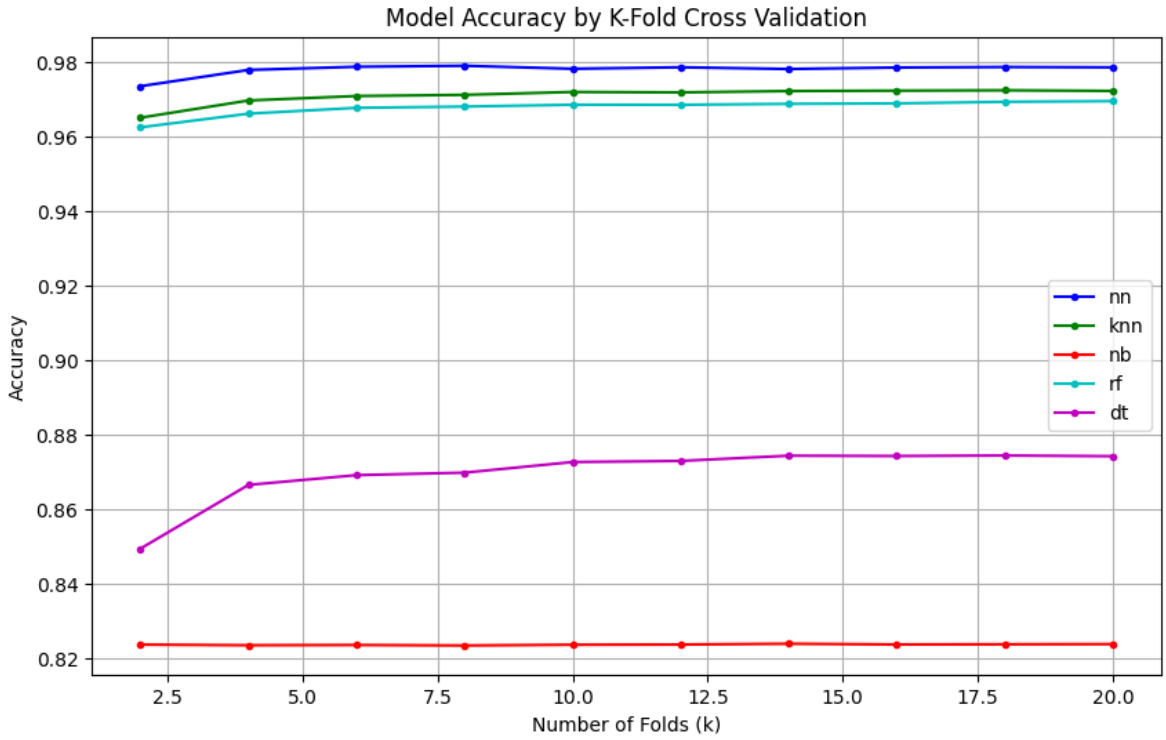

Figure 3: k-Fold Cross Validation on all models

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.

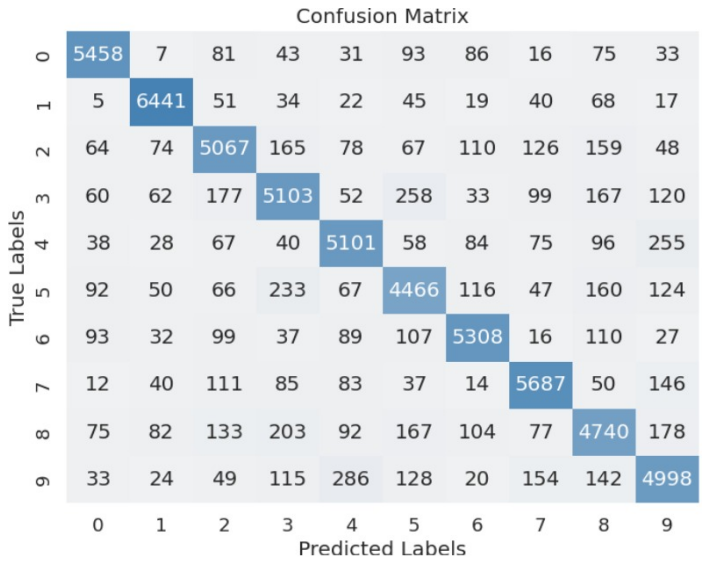

Figure 4: Confusion Matrix for Decision Tree

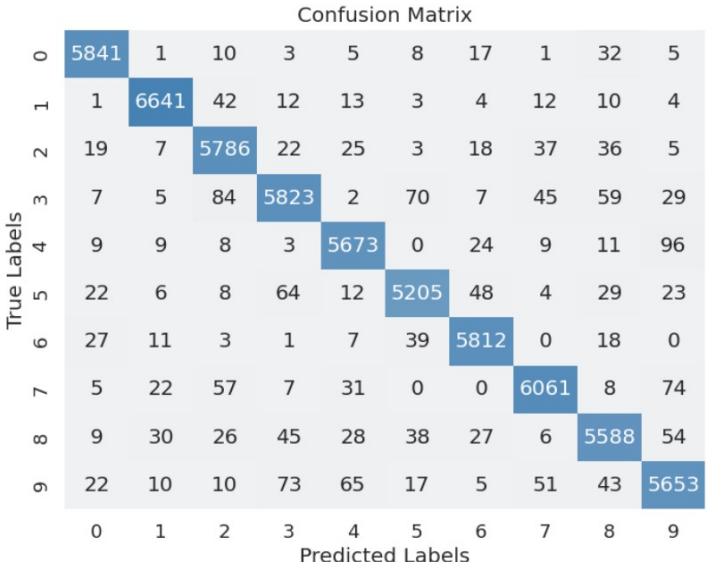

Figure 5: Confusion Matrix for Random Forest

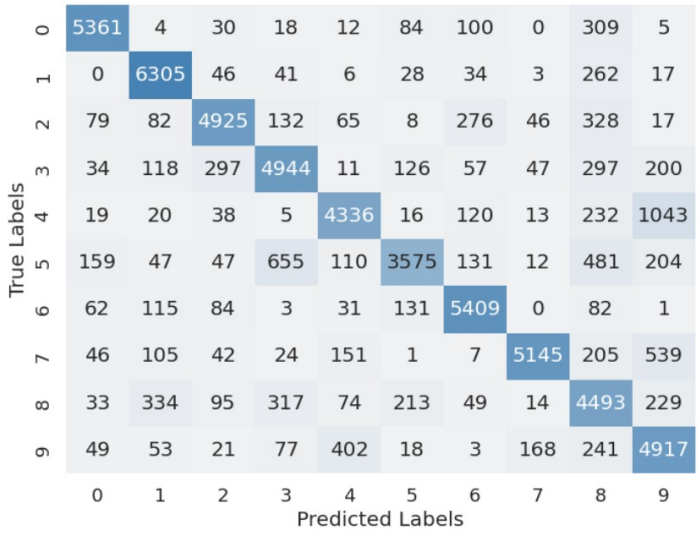

Figure 6: Confusion Matrix for Naive Bayes

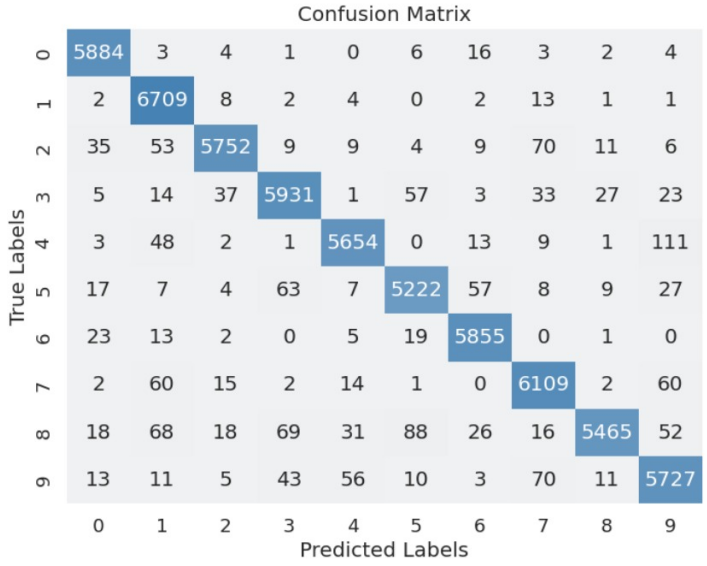

Figure 7: Confusion Matrix for KNN Classifier

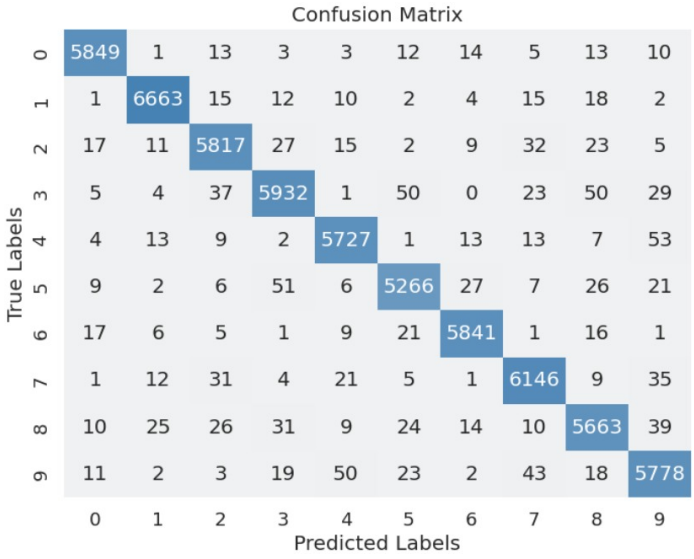

Figure 8: Confusion Matrix for Neural Network Classifier

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.