Project Report for Data Mining (Homework #1)

HW 1.1: Classification of Digits using Pen digits Data set

Name: Chaithanya Pramodh Kasula (G01197109)

Objective:

To build a Machine Learning model that can classify a hand-written digit which is represented in the form of a sequence of co-ordinates.

Data Capture and simple visualization:

Source: https://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits

The data was captured from UCI Machine Learning repository through the above link. The data set [1] is a collection of handwritten pen digits [0-9] whose co-ordinates are in the range of 0 to 100. Each row represents a collection of 8 co-ordinates of the digit it represents. The total number of columns are 17 where 1 to 16 columns represent the co-ordinate pairs and the 17th column denotes the label/digit. The total number of rows in the train and test sets are 7494 and 3498 respectively. Sample images when plotted, are **visualized** as follows.

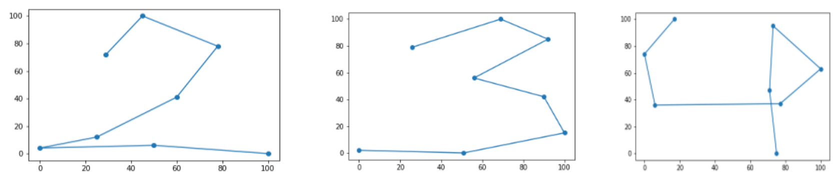


Fig. 1

Architecture:

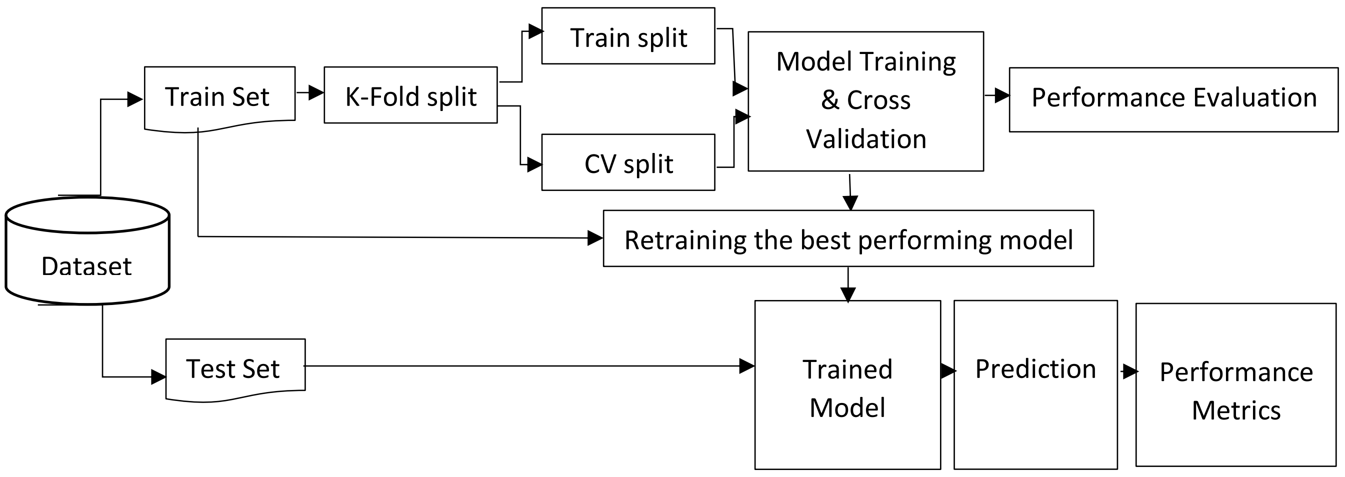


Fig. 2

The architecture of the pen digit classification is shown in Fig. 2. The data set is split into train and test sets. The train set is further divided into k-folds. Model is trained and cross validation is performed for all the folds. Average classification metrics are obtained, and the best performing model is selected through model selection techniques. The hyperparameters are determined and the model is retrained over the complete data set. The test set is fed into the best performing classifier and performance evaluation is carried out.

Data Preprocessing:

Data cleaning was not necessary as there were no missing values or any unwanted special characters. **Data transformation** has been performed during which the values of the data set have been transformed from the range of 0 – 100 to 0 – 1 (Normalization). The 'MinMaxScaler' [2] from sklearn was utilized to perform the operation. Normalization is necessary because the data in HW 1.2 (MNIST) will be used to extract data that would be in the range of 1 – 28 and will also be transformed to 0-1 scale. Hence it is important for the model to be trained on a common scale [0-1] so that it can be used on both the data.

Data Visualization – Dimensionality Reduction:

For the purpose of data visualization, high dimensional data (16 dimensions) was reduced to two dimensions by using a dimensionality reduction algorithm known as t-SNE (t-Distributed Stochastic Neighbor Embedding) [3]. The reduced dimensions when plotted are as shown in Fig 3. Each color represents a unique digit. Clear clusters can be noted from Fig. 3. For example, all the points in bright orange represent all the rows of digit/class 0. Vivid yellow points represent all the instances of the digit 1. Distinct boundaries can be drawn in between digits. In such cases, the classification algorithm is expected to perform better. It is to be noted that, the features are reduced only for visualization but not for training or testing.

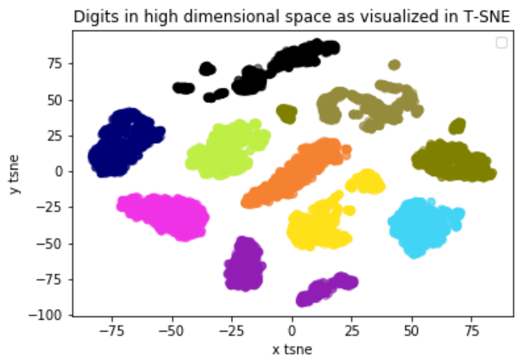
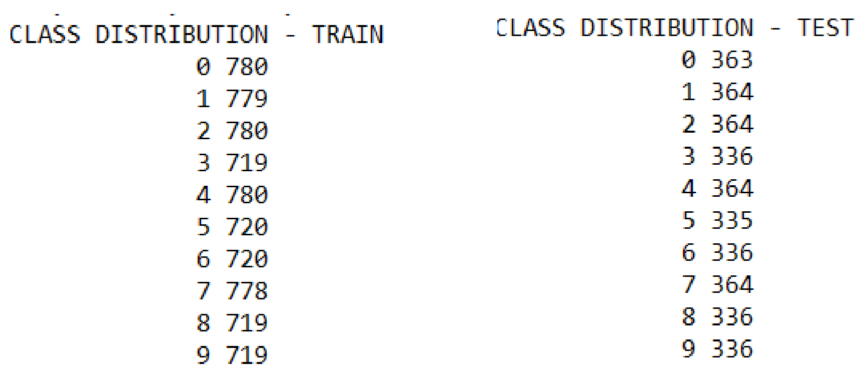


Fig. 3

Class distribution:

The class distribution for train and test sets is depicted in the Fig. 4. The number of samples for 0, 1, 2, 4, and 7 is close to 780 where as the number of samples for classes 3, 5, 6, 8 and 9 is nearer to 720. Since the values are very close to each other. Class imbalance problems are not expected to affect the performance majorly. The class distribution can be stated as nearly balanced.



```
CLASS DISTRIBUTION - TRAIN        CLASS DISTRIBUTION - TEST
                0 780                             0 363
                1 779                             1 364
                2 780                             2 364
                3 719                             3 336
                4 780                             4 364
                5 720                             5 335
                6 720                             6 336
                7 778                             7 364
                8 719                             8 336
                9 719                             9 336
```

Fig. 4

Model Training:

K-fold Cross Validation: The training data set was divided into k-folds (5-folds). In each fold, 4 parts of data was used for training and the remaining one part was used for testing. The 'DecisionTreeClassifier' [4] was used to build the classification tree. The available split criterion in the package are **'gini'** and **'entropy'**. Hence, two models were built using the mentioned split criterion.

Classification Metrics:

The classification metrics used in the current project are Accuracy, Precision, Recall and F1-Score. Further, for cross validations, average accuracy, macro average precision, macro average recall, macro average f1 score, weighted average precision, weighted average recall, weighted average f1 score were also computed. The results are as shown in Fig. 5.

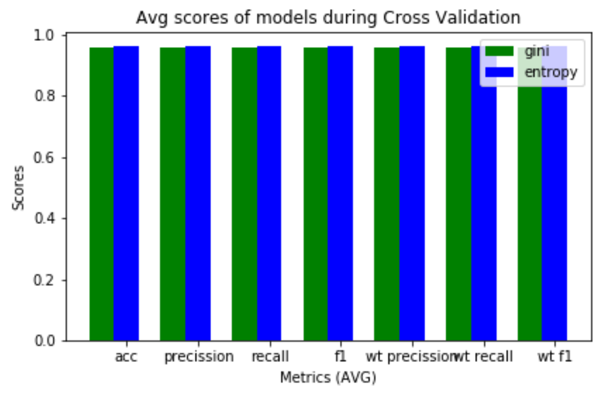
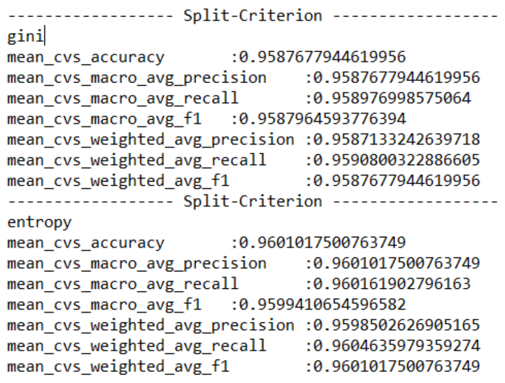


Fig. 5

As depicted in Fig. 5, the decision tree classifier with 'entropy' as split criterion achieved the maximum of all the metrics. A snapshot of achieved metrics is represented in Fig. 6.



```
----------------- Split-Criterion -----------------
gini
mean_cvs_accuracy       :0.9587677944619956
mean_cvs_macro_avg_precision    :0.9587677944619956
mean_cvs_macro_avg_recall       :0.958976998575064
mean_cvs_macro_avg_f1   :0.9587964593776394
mean_cvs_weighted_avg_precision :0.9587133242639718
mean_cvs_weighted_avg_recall    :0.9590800322886605
mean_cvs_weighted_avg_f1        :0.9587677944619956
----------------- Split-Criterion -----------------
entropy
mean_cvs_accuracy       :0.9601017500763749
mean_cvs_macro_avg_precision    :0.9601017500763749
mean_cvs_macro_avg_recall       :0.960161902796163
mean_cvs_macro_avg_f1   :0.9599410654596582
mean_cvs_weighted_avg_precision :0.9598502626905165
mean_cvs_weighted_avg_recall    :0.9604635979359274
mean_cvs_weighted_avg_f1        :0.9601017500763749
```

Fig. 6

Hyper-parameter tuning:

Various hyperparameters such as 'max_depth', 'min_samples_split', 'max_features' were assigned relevant values. The classification metrics were examined to determine the classifier that gives the best performance in terms of Accuracy, Precision, Recall and F1 score. The below hyperparameters were finally chosen as they were achieving higher values for the below defined metrics.

Model 1: DecisionTreeClassifier (criterion= 'gini', splitter="best", random_state=19)

Model 2: DecisionTreeClassifier (criterion= 'entropy', splitter="best", random_state=19)

Model Selection:

Although Model 2 with 'entropy' as split criterion was performing better than Model 1 ('gini'), box plots were plotted to look at the variation of error across all folds. Hence, with accuracy as the measure, box plots were plotted to confirm the selection of Model 2. This is shown in Fig. 7.

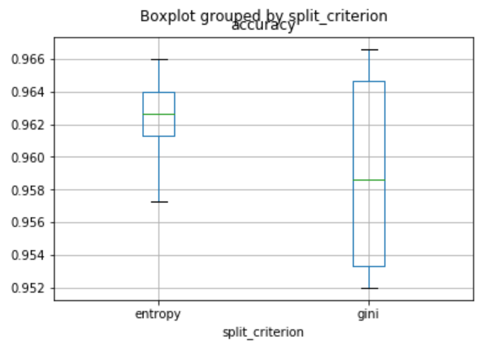
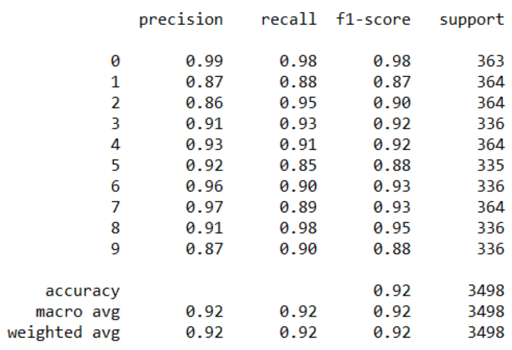


Fig. 7

It can be noted from Fig. 7 that the variation in the error measure is less for Model 2 ('entropy') than for Model 1 ('gini'). Hence Model 2 ('entropy') was selected as the **best classifier**.

Final Training and Testing over pen digits data:

The chosen classifier with the same set of hyperparameters is trained over the whole train set of 7494 records and tested over 3498 records respectively. The final evaluation metrics are shown in Fig. 8.



```
              precision    recall  f1-score   support

           0       0.99      0.98      0.98       363
           1       0.87      0.88      0.87       364
           2       0.86      0.95      0.90       364
           3       0.91      0.93      0.92       336
           4       0.93      0.91      0.92       364
           5       0.92      0.85      0.88       335
           6       0.96      0.90      0.93       336
           7       0.97      0.89      0.93       364
           8       0.91      0.98      0.95       336
           9       0.87      0.90      0.88       336

    accuracy                           0.92      3498
   macro avg       0.92      0.92      0.92      3498
weighted avg       0.92      0.92      0.92      3498
```

Fig. 8

Visualizing Decision Tree:

The trained classifier is visualized by using **'graphviz'** module. The function takes the classifier as input and prints a pdf of the decision tree. A sample part of the decision tree is shown in Fig. 9.

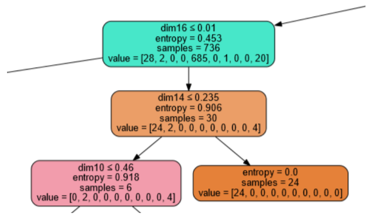
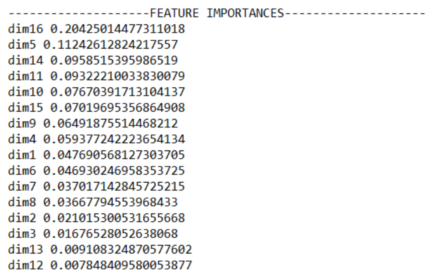

dim16 ≤ 0.01
entropy = 0.453
samples = 736
value = [28, 2, 0, 0, 685, 0, 1, 0, 0, 20]
dim14 ≤ 0.235
entropy = 0.906
samples = 30
value = [24, 2, 0, 0, 0, 0, 0, 0, 0, 4]
dim10 ≤ 0.46
entropy = 0.918
samples = 6
value = [0, 2, 0, 0, 0, 0, 0, 0, 0, 4]
entropy = 0.0
samples = 24
value = [24, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Fig. 9

Feature Importance:

The feature importance of all the 16 dimensions are retrieved from the classifier and pictured below. Feature selection during cross validation has been performed and tested with the classifier. However, it was not giving favorable results. The importance of features for the final model are pictured in Fig. 10.



```
--------------------FEATURE IMPORTANCES--------------------
dim16 0.20425014477311018
dim5 0.11242612824217557
dim14 0.0958515395986519
dim11 0.09322210033830079
dim10 0.07670391713104137
dim15 0.07019695356864908
dim9 0.06491875514468212
dim4 0.059377242223654134
dim1 0.047690568127303705
dim6 0.046930246958353725
dim7 0.037017142845725215
dim8 0.03667794553968433
dim2 0.021015300531655668
dim3 0.01676528052638068
dim13 0.009108324870577602
dim12 0.007848409580053877
```

Fig. 10

Learning curves for Model Performance:

The learning curves were plotted for the decision tree classifier to evaluate its performance. The red line in Fig. 11 shows the training error and the green line shows the validation error. It can be observed that the training and validation curves converge at similar values. Additionally, the gap between the lines is small. The smaller the gap, the better the model. Hence, the model fits trained data **perfectly** and **generalizes** very well for the validation data.

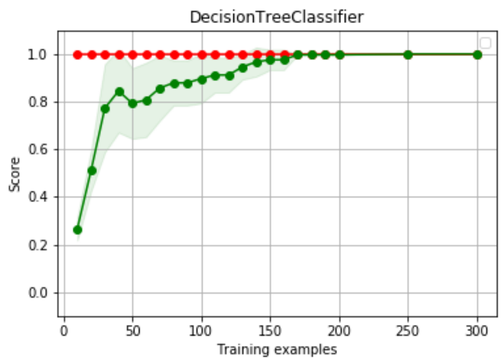


Fig. 11

Saving the model to a file:

The trained classifier is saved to a file (.sav format) by using 'pickle' package available in python. The file is then used to load the trained model to test the data set obtained from MNIST in the next module.

HW 1.2: MNIST digit classification with Trained Pen digits Model

Objective:

Transfer learning is technique where a model that is trained in a specific task is used for another related task. For example, a model that is trained to recognize cats can be used as a starting point for detecting dogs. The model trained on cats is expected to recognize features such as eyes, ears and nose etc. on dogs too. Similarly, the objective of this module is to know how a classifier trained on pen digits would perform over co-ordinates of digits extracted from MNIST data set [5].

Architecture:

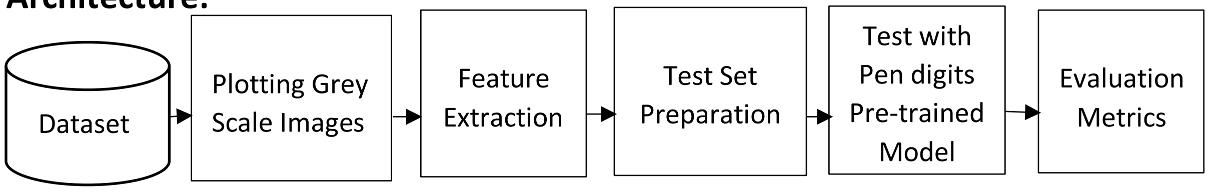

Dataset
Plotting Grey Scale Images
Feature Extraction
Test Set Preparation
Test with Pen digits Pre-trained Model
Evaluation Metrics

Fig. 12

The MNIST data set is used to plot digits. Feature extraction is performed by selecting 8 evenly spaced co-ordinates from the plotted grey scale image. Test set is prepared by collecting such co-ordinates from randomly sampled rows of the MNIST data set. The trained pen digits model is loaded from the saved file and the generated test set is passed through the pre-trained pen digits model for prediction. Evaluation metrics are reported.

Data Capture and Visualization:

Source: https://www.kaggle.com/oddrationale/mnist-in-csv

The data was captured from Kaggle. It consists of train and test sets with 60,000 and 10,000 examples respectively. Each image represents a vector of pixels (0-783) of a hand-written digit (28x28 box) in grey scale. The pixel values range from 0 – 255. The total number of columns are 785. The first column represents the digit/label. The rest represent the pixel values in the 28x28 matrix [1-783]. A sample plotted image is shown in Fig. 13.

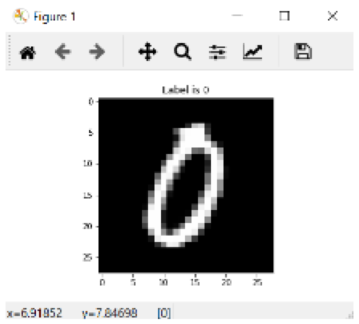


Fig. 13

Data set preparation and Feature Extraction:

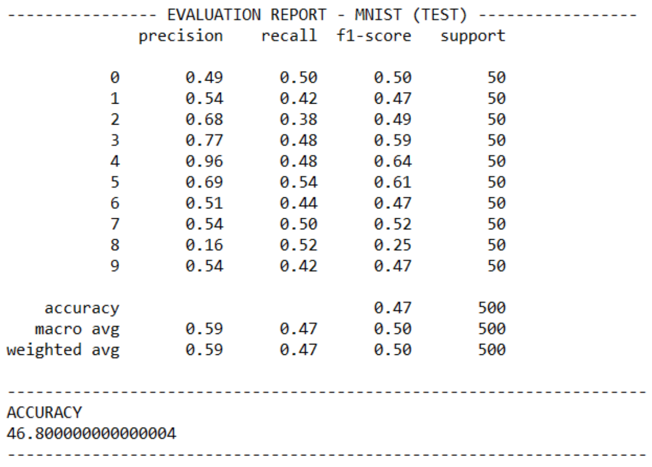
Raw data in the MNIST data set represents a grey scale image. A data set is to be prepared by regularly selecting 8 co-coordinates that represent the grey scale image in the MNIST data set. Images are visualized by simply plotting the grey scale images using the 'imshow' function of matplotlib. 'mpl_connect' was used to connect the figure such that when the pixels were clicked on the displayed image, their position in the image matrix (28x28) was returned as an output. 50 samples of each class representing 0-9 digits were randomly selected from the MNIST train data set and plotted by using imshow. The 8 most representative co-ordinates were collected for all the randomly sampled rows and written into a csv file. Thus, the sampled data set consists of 17 columns such that the first column represents the label/digit and the rest 16 columns represent the 16 values of the 8 extracted co-ordinates. The total number of rows are 500 which constitute samples representing 50 samples of each digit. The range of the values in the data set is 1 (min) – 28 (max).

Data Preprocessing:

Data cleaning was performed, and the missing values are replaced by the closest representative co-ordinate for that digit. Although, the occurrence of missing values is rare, it is still a possibility. **Data transformation** has been carried out to transform the range of the data set from minimum and the maximum numeric of the collected co-ordinates to [0-1]. The data set is also sorted according to the class labels, i.e., all the sampled instances of a digit occur together.

Model testing and Evaluation:

The saved model trained over pen digits data set is loaded from the pickle file. The collected data set of 8 co-ordinate pairs of 500 digits after normalization is named as test set. Predictions are made by the classifier for all the rows of the test set. The classification metrics per class/digit and the overall evaluation is pictured in Fig. 14.



```
---------------- EVALUATION REPORT - MNIST (TEST) -----------------
              precision    recall  f1-score   support

           0       0.49      0.50      0.50        50
           1       0.54      0.42      0.47        50
           2       0.68      0.38      0.49        50
           3       0.77      0.48      0.59        50
           4       0.96      0.48      0.64        50
           5       0.69      0.54      0.61        50
           6       0.51      0.44      0.47        50
           7       0.54      0.50      0.52        50
           8       0.16      0.52      0.25        50
           9       0.54      0.42      0.47        50

    accuracy                           0.47       500
   macro avg       0.59      0.47      0.50       500
weighted avg       0.59      0.47      0.50       500

--------------------------------------------------------------------
ACCURACY
46.800000000000004
--------------------------------------------------------------------
```

Fig. 14

Software and Hardware used:

Software:

Python, scikit-learn, Anaconda, graphviz and other essential packages.

Hardware:

Processor: Intel(R) Core(TM) i7-9750H CPU@2.60 GHz (12 CPUs), ~2.6GHz
Memory: 16384MB RAM
Operating System: Windows 10 Home 64-bit

Conclusion:

The application of **transfer learning** was finely demonstrated in the given task. The performance metrics of the trained model over the co-ordinates extracted from the MNIST digits data was low. An advanced algorithm which is better than a Decision Tree Classifier could have performed better. Also, the trained model can be fine-tuned/adjusted further to increase performance metrics.

References:

[1] Pen-Based Recognition of Handwritten Digits Data Set (n.d.). Retrieved from https://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits

[2] Pedregosa, F., Varoquaux, Ga"el, Gramfort, A., Michel, V., Thirion, B., Grisel, O., … others. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825–2830.

[3] Van der Maaten, L. & Hinton, G. (2008). Visualizing Data using t-SNE. Journal of Machine Learning Research, 9, 2579--2605.

[4] Sklearn DecisionTreeClassifier. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

[5] LeCun, Y. & Cortes, C. (2010). MNIST handwritten digit database.