**Project Report for Data Mining (Homework #1)**

**HW 1.1: Classification of Digits using Pen digits Dataset**

**Name: Chaithanya Pramodh Kasula (G01197109)**

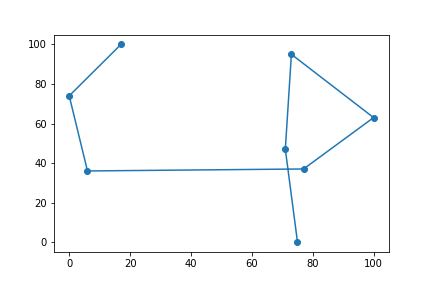
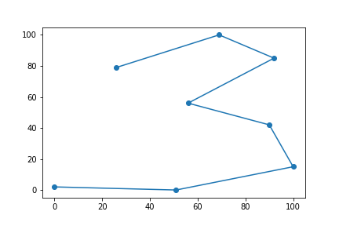
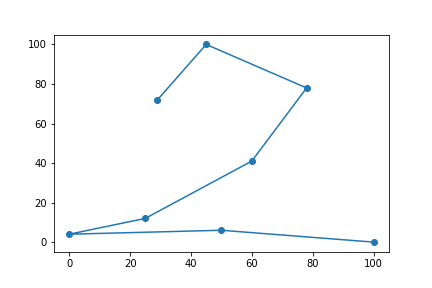
**Objective:**

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:**

**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 dataset [1] is a collection of handwritten pen digits which 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 can be visualized as follows.



**Fig. 1**

**Architecture:**

Model Training & Cross Validation

Train split

Train Set

Performance Evaluation

K-Fold split

CV split

Retraining the best performing model

Dataset

Performance Metrics

Prediction

Trained  
Model

Test Set

**Fig. 2**

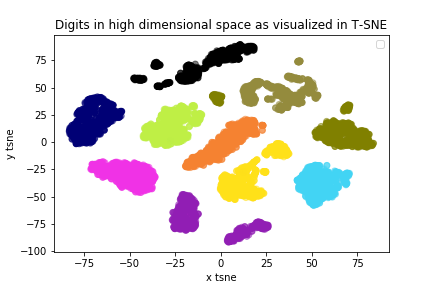
The architecture of the pendgit classification is shown in Fig. 2. The dataset is split into train and test sets. The train set is further divided into k-folds. Model is trained and cross validation is performed for all the folds. Average classification metrics are obtained, and the best performing model is selected through model selection techniques. The hyperparameters are determined and the model is retrained over the complete dataset. 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 nor any unwanted special characters. **Data transformation** has been performed during which the values of the dataset 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 necessary 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 zero. 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.



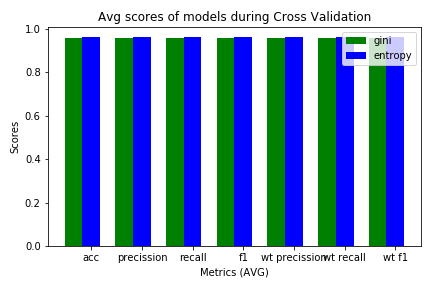
**Fig. 3**

**Model Training:**

**K-fold Cross Validation:** The training dataset 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 **‘information gain’ (entropy)**. Hence two models were built using the mentioned split criterion.

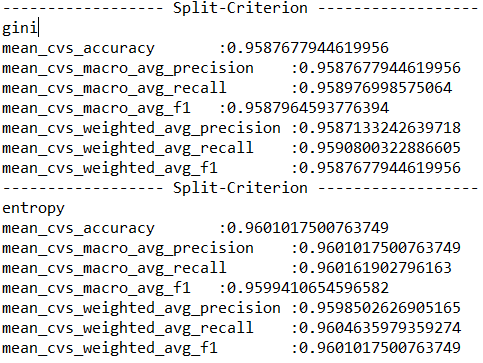
**Classification Metrics:**

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. 4.

****

**Fig. 4**

As depicted in Fig. 4, the decision tree classifier with ‘entropy’ as split criterion achieved the maximum in all the metrics. A snapshot of achieved metrics is represented in Fig. 5.



**Fig. 5**

**Hyper-parameter tuning:**

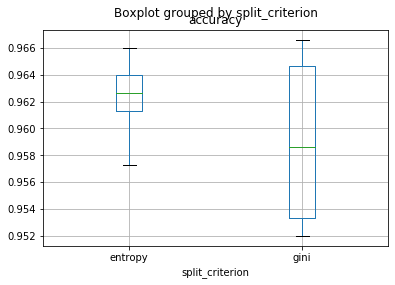
**Various hyperparameters** such as ‘max\_depth’, ‘min\_samples\_split’, ‘max\_features’ were assigned relevant values. The classification metrics were examined to determine the classifier that gives the best performance in terms of Accuracy, Precision, Recall and F1 score. The below hyperparameters were finally chosen as they were achieving higher values for the defined metrics.

**Model 1:** DecisionTreeClassifier (criterion= ‘gini’, splitter="best", random\_state=19)

**Model 2:** DecisionTreeClassifier (criterion= ‘entropy’, splitter="best", random\_state=19)

**Model Selection:**

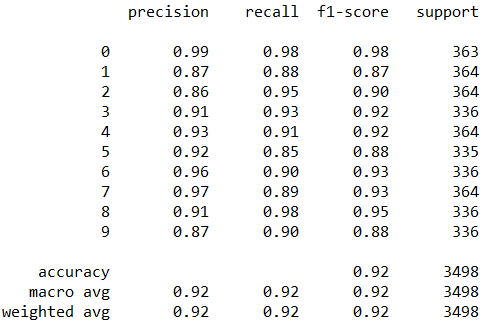
Although Model 2 with ‘entropy’ as split criterion was performing better than Model 1, box plots were plotted to look at the variation of error across all cross validations. Hence, with accuracy as the measure, box plots were plotted to confirm the selection of Model 1. This is shown in Fig. 6.

**  
Fig. 6**

It can be noted from Fig. 6 that the variation in the error measure is less for Model 2 (‘entropy’) than for Model 1 (‘gini’). Hence Model 2 was selected as the **best classifier**.

**Final Training and Testing over pendigits 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. 7.

**  
Fig. 7**

**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 dataset obtained from MNIST in the next module.

**HW 1.2: MNIST digit classification with Trained Pendigits 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 dataset [5].

**Architecture:**

Evaluation Metrics

Plotting Grey Scale Images

Feature Extraction

Test Set Preparation

Test with Pen digits Pre-trained Model

Dataset

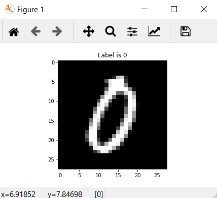
**Fig. 8**

The MNIST dataset 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 dataset. The trained pendigits 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. 9.

  
**Fig. 9**

**Dataset preparation and Feature Extraction:**

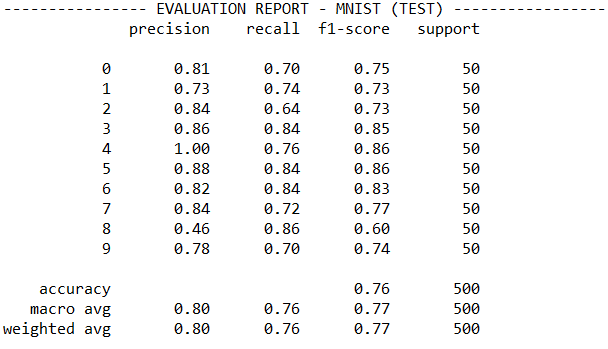
Raw data in the MNIST dataset represents a grey scale image. A data set is to be prepared by regularly selecting 8 co-coordinates that represent the grey scale image in the MNIST dataset. Images are visualized by simply plotting the grey scale images using the ‘imshow’ function of matplotlib. ‘mpl\_connect’ was used to connect the figure such that when the pixels were clicked on the displayed image, they were recognized and 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 that the first column represents the label/digit and the rest 16 columns represent the 16 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 to transform the range of the data set from 1-28 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 pendigits dataset 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. 10.

  
**Fig. 10**

**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.