Introduction to Machine Learning

Assignment 4

I. Notebook for Question 1 – Data Preprocessing task
Filename - Omkar_Pandit_Lab_Assignment4_Q1_Part1

II. Notebook for Question 1 – Classification task
Filename - Omkar_Pandit_Lab_Assignment4_Q1_Part2

III. Notebook for Question 2 – MNIST dataset
Filename - Omkar_Pandit_Lab_Assignment4_Q2

Q1. Azure note book link for Question1 Part 1- https://omkarpandit-oap338.notebooks.azure.com/j/notebooks/Omkar_Pandit_Lab_Assignment4_Q1_Part1.ipynb

Azure note book link for Question1 Part 2- https://omkarpandit-oap338.notebooks.azure.com/j/notebooks/Omkar_Pandit_Lab_Assignment4_Q1_Part2.ipynb

a. Code part
b. Code part
c. Code part
d. Code part

e. Implementation method –

First, extracted images from three parts train, validation, test set folders. Using a computer vision API, I have extracted the low level and high-level image features for all the images present in given three sets.
After that, I have extracted all the associated questions and answers from given JSON files for training and validation dataset. For, test set, as I don't have the answers to the given questions, so, just extracted the questions and map it with the respective images' features.
In the end, I have created three different files for my next stage analysis. In first file I have combined all the input features and questions from all three sets i.e. train, validation, test together in single file. We have combined all together to convert textual data into numeric format using TF-IDF data modelling algorithm.
In second and third file, I have stored output values i.e. value that defines whether question is answerable or not? For training and validation dataset respectively.

I have converted all the textual data present in terms of image features and questions into quantitative terms using TF-IDF. After this conversion, I have split the dataframe into three different dataframes train, validation, test.
After data preprocessing, I have implemented multiple classifiers to achieve best classification results. For this, I have built the classifier model by fitting the train and train output data. Predicted output values for validation set data and calculated accuracy score by comparing the predicted values with actual validation set output values.
So, after comparing accuracy scores of all the classifiers, I have implemented the classifier which gave best accuracy score to predict output values for test dataset.

f. For the analysis, I have implemented multiple classifiers like KNN, SVM, Decision tree, Random Forest, Logistic regression, Voting classifier. I have also implemented 'OneVsRestClassifier' which I learned during Statistical learning subject. A
So, after comparing accuracy scores of all classifiers, I have got best scores for 'SVM' which is '0.74'.
SVM works best because of its highly flexible nature. In SVM, I have tried different combinations of hyperparameters like CurvC values, Gamma values, degree and choice of a kernel. I tried different combinations for hyperparameters and came up with best values for all the hyperparameters. As data is not linearly separable, I have used polynomial kernel to deal with non-linear data. SVM is best because it is more robust than other classifiers.

Best hyperparameter values for SVM –

CurvC – 0.001
Gamma – 0.001
Degree – 2
Kernel - poly

In my case Naïve Bayes performed worst and gave '0.46' accuracy score on validation set. I think reason behind this is, Naïve Bayes works on the assumption that variables are independent/ linearly separable which is not the case with our dataset.

Q2.

a. Azure note book link for Question 2 - https://omkarpandit-oap338.notebooks.azure.com/j/notebooks/Omkar_Pandit_Lab_Assignment4_Q2.ipynb

b. Code part

c. Report the hyperparameters.

Layer description	Train Accuracy	Test Accuracy
Layer 1 – Neurons - 10	0.96	0.91
Layer 2 – Neurons – 10	0.96	0.91
Layer 3 – Neurons – 10	0.96	0.91
Layer 4 – Neurons – 10	0.96	0.91
Layer 5 – Neurons – 9	0.96	0.91
Layer 6 – Neurons - 10	0.96	0.92
Layer 7 – Neurons – 9	0.96	0.91
Layer 8 – Neurons – 9	0.95	0.91
Layer 9 – Neurons - 10	0.96	0.91
Layer 10 – Neurons - 10	0.96	0.91

Batch size: auto
The solver for optimization / Gradient descent approach: adam
Activation Function: tanh
Learning Rate: 0.001
Max number of iterations: 200

So here, if you observe, I have obtained high train and test accuracy of 0.96 and 0.92 respectively with 6 hidden layers with 10 neurons in it.
As weights are depend on number of inputs and number of layers.

Given –
Number of inputs – 784
Number of outputs – 1

$$\begin{aligned} \text{Weight} &= (784 * 10) + (10 * 10) + (10 * 10) + (10 * 10) + \\ &\quad (10 * 10) + (10 * 10) + (10 * 1) \\ &= 8350 \end{aligned}$$

Therefore, I have obtained most optimal hyperparameters with 6 hidden layers, 10 neurons per hidden layer with a test accuracy of 0.92.

d. In my case most optimal parameters were found by varying number of hidden layers and number of neurons per hidden layer. So, in general, whenever I increased the number of layers and neuron number per layers, then I have achieved better accuracy. So, in my case I have achieved highest test accuracy of 0.92 with a model having 6 hidden layers and 10 neurons in each of these 6 layers.

Hidden layers are used to change the dimensions and to increase the prediction power of input features to best fit the training model. But after certain point, increase in hidden layers and neuron count might result in overfitting.

So, least number of nodes and least number of hidden layers not able to fit properly on training data and it results in less training and test accuracy. So, find the best combination of number of hidden layers and number of neurons is important to achieve highly accurate outputs from built model.