CS451 Introduction to Artificial Intelligence

Homework 3

Mert Erkol S017789

Introduction

In this homework I have implemented 3 different machine learning algorithms to learn from handwritten digits and try to guess the given handwritten digit. Machine Learning(ML) has been a hot topic since the 1980s and it uses several techniques to understand the data and try to learn a path. There are 4 types of ML algorithms: Regression, Classification , Clustering and Reinforcement Learning. In this work I have used classification algorithms to classify the handwritten digit. Regression or RL is not a quite good fit for this work because we are mapping the inputs to a discrete set of numbers. The outcome variable is not only discrete it is also nominal and categorical that is why classification techniques perform much better than regression. In the Fig-1 you can see the example dataset. In the dataset there are multiple writing types of the same digit. This adds accuracy to our data to classify a handwritten digit from another person.

Fig-1: Examples in the dataset

Implementation and Methodology

I have implemented 3 different classification algorithms: KNearestNeighbor, Decision Tree and Logistic Regression. After the implementation I tuned each algorithm and tried to find the best parameter for each classification. I also predict both test and train data and draw a confusion matrix to see the scores of each digit. I have used Python programming language version 3.9 and some libraries to make implementation easier like: NumPy, Pandas, Matplotlib, Seaborn and our ML library scikit-learn and there is a helper Python file called DataLoader to get our data ‘mnist_784’ with random_state 123 and 0.2 test size all of this results got from this variables. After the data gathering I have used 2 different scaling methods: Standard and Min-Max. I have tested both on the KNN algorithm and it seems the Min-Max algorithm performs better on the accuracy score because min-max scaling performs better if the variables are bounded like pixels in this work (0-255). You can see every algorithm's best performance variables classification report and confusion matrix in the results section. For KNN I have tuned the n_neighbors variable for both train and test set and it seems 3 neighbors is performing the best from numbers 1 to 5.For predicting the training set KNN overfits if the K number is small. For the Decision Tree model I tuned depth and the leaves values from 1 to 10 and from 2^0 to 2^5 in order. It seems the depth value of 9 and min leave for 4 performs best accuracy score according to GridSearch. GridSearch is a specific tool for tuning the ML algorithms. At last I tuned Logistic Regression for its C value from an array of C values [0.001, 0.01, 0.1, 1] for test set value C of 0.1 and for the train set the value of 1 performs the best according to the classification reports but Logistic Regression also overfits when predicting the train set overfitting increases with the C value.

Results

Classification report for number of k (test_data): 3

	precision	recall	f1-score	support
0	0.97	0.99	0.98	1381
1	0.97	0.99	0.98	1575
2	0.98	0.97	0.98	1398
3	0.97	0.97	0.97	1428
4	0.98	0.96	0.97	1365
5	0.97	0.96	0.97	1263
6	0.98	0.99	0.99	1375
7	0.97	0.97	0.97	1459
8	0.99	0.95	0.97	1365
9	0.95	0.96	0.95	1391
accuracy			0.97	14000
macro avg	0.97	0.97	0.97	14000
weighted avg	0.97	0.97	0.97	14000

Fig-2: Report of KNN with test data

Classification report for number of k (train_data): 3

	precision	recall	f1-score	support
0	0.99	1.00	0.99	5522
1	0.98	1.00	0.99	6302
2	0.99	0.98	0.99	5592
3	0.98	0.99	0.98	5713
4	0.99	0.98	0.99	5459
5	0.99	0.98	0.98	5050
6	0.99	0.99	0.99	5501
7	0.98	0.99	0.99	5834
8	1.00	0.96	0.98	5460
9	0.98	0.98	0.98	5567
accuracy			0.99	56000
macro avg	0.99	0.99	0.99	56000
weighted avg	0.99	0.99	0.99	56000

Fig-3: Report of KNN with train data

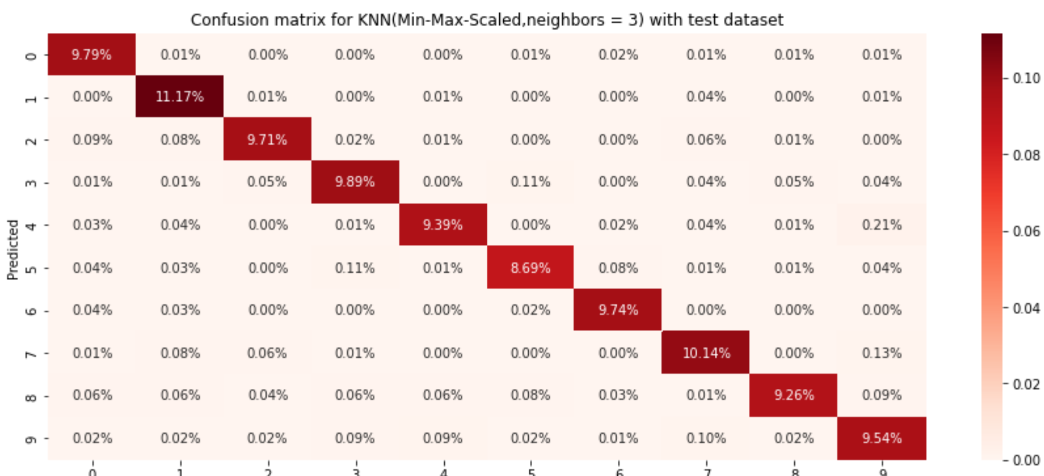

Fig-4: Confusion matrix of KNN with test data set

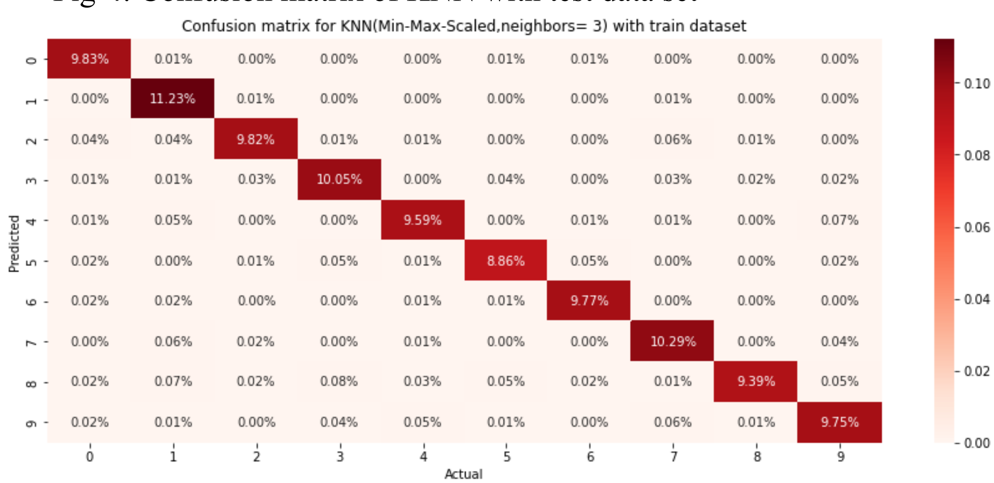

Fig-4: Confusion matrix of KNN with train data set

Decision tree classification report for test set with best params

	precision	recall	f1-score	support
0	0.90	0.91	0.91	1381
1	0.88	0.94	0.91	1575
2	0.80	0.83	0.82	1398
3	0.83	0.77	0.80	1428
4	0.84	0.83	0.83	1365
5	0.79	0.77	0.78	1263
6	0.89	0.86	0.88	1375
7	0.89	0.86	0.87	1459
8	0.76	0.77	0.76	1365
9	0.80	0.83	0.81	1391
accuracy			0.84	14000
macro avg	0.84	0.84	0.84	14000
weighted avg	0.84	0.84	0.84	14000

Fig-5: Report of DTree with test data

Decision tree classification report for train set with best params

	precision	recall	f1-score	support
0	0.93	0.95	0.94	5522
1	0.89	0.94	0.92	6302
2	0.84	0.87	0.85	5592
3	0.87	0.81	0.84	5713
4	0.86	0.84	0.85	5459
5	0.83	0.82	0.83	5050
6	0.91	0.89	0.90	5501
7	0.90	0.89	0.89	5834
8	0.80	0.80	0.80	5460
9	0.82	0.84	0.83	5567
accuracy			0.87	56000
macro avg	0.86	0.86	0.86	56000
weighted avg	0.87	0.87	0.86	56000

Fig-6: Report of DTree with train data

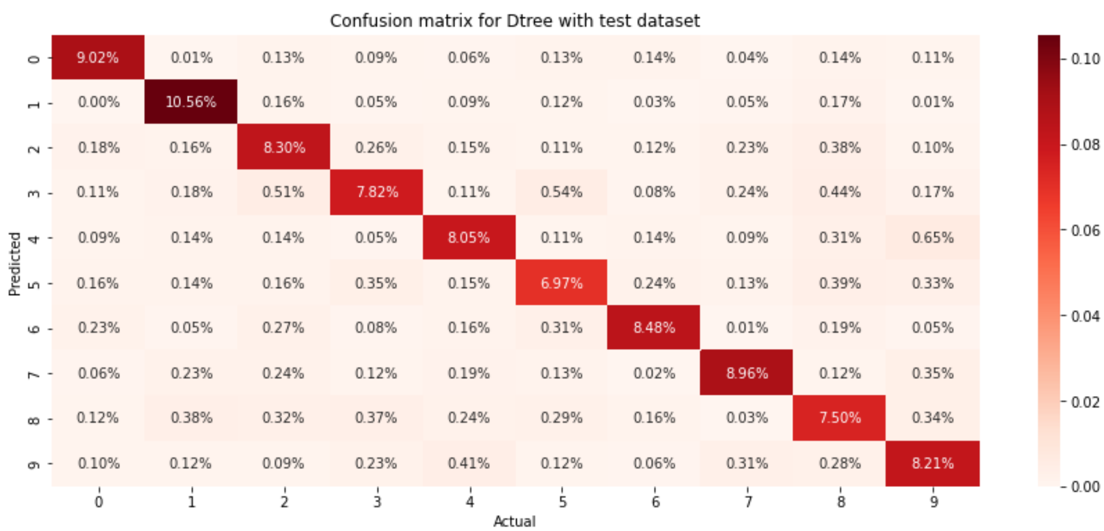

Fig-7: Confusion Matrix of DTree with test data

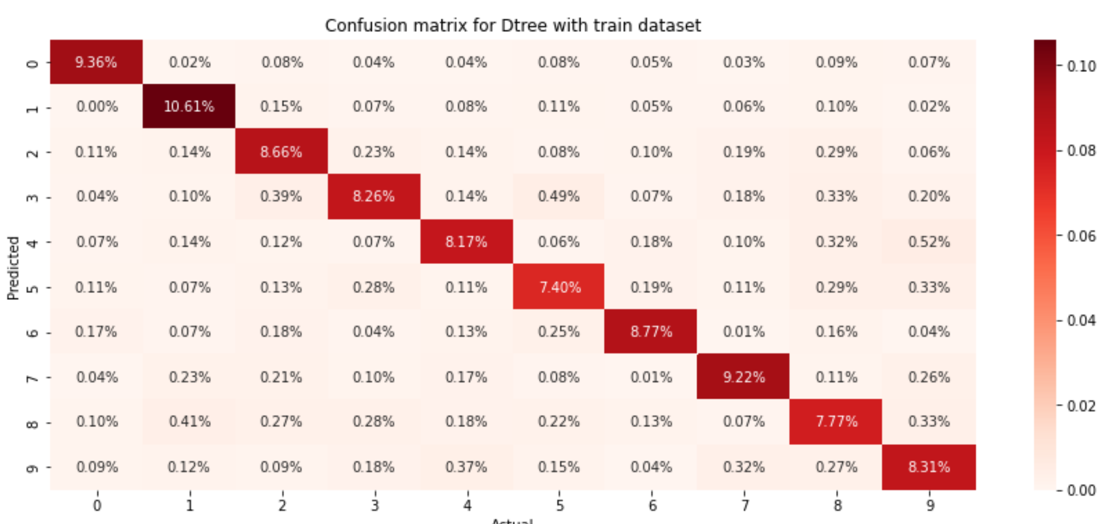

Fig-8: Confusion Matrix of DTree with train data

Classification report for value of C (test_data): 0.1

	precision	recall	f1-score	support
0	0.96	0.97	0.96	1381
1	0.96	0.98	0.97	1575
2	0.93	0.90	0.91	1398
3	0.89	0.90	0.90	1428
4	0.92	0.93	0.92	1365
5	0.88	0.87	0.88	1263
6	0.95	0.96	0.95	1375
7	0.93	0.93	0.93	1459
8	0.90	0.87	0.88	1365
9	0.90	0.91	0.90	1391
accuracy			0.92	14000
macro avg	0.92	0.92	0.92	14000
weighted avg	0.92	0.92	0.92	14000

Fig-9: Report of LogReg with test data

Classification report for value of C (train_data): 1

	precision	recall	f1-score	support
0	0.97	0.98	0.98	5522
1	0.96	0.98	0.97	6302
2	0.93	0.92	0.93	5592
3	0.93	0.92	0.92	5713
4	0.95	0.95	0.95	5459
5	0.92	0.91	0.91	5050
6	0.96	0.97	0.96	5501
7	0.95	0.95	0.95	5834
8	0.91	0.91	0.91	5460
9	0.92	0.92	0.92	5567
accuracy			0.94	56000
macro avg	0.94	0.94	0.94	56000
weighted avg	0.94	0.94	0.94	56000

Fig-10: Report of LogReg with trainset

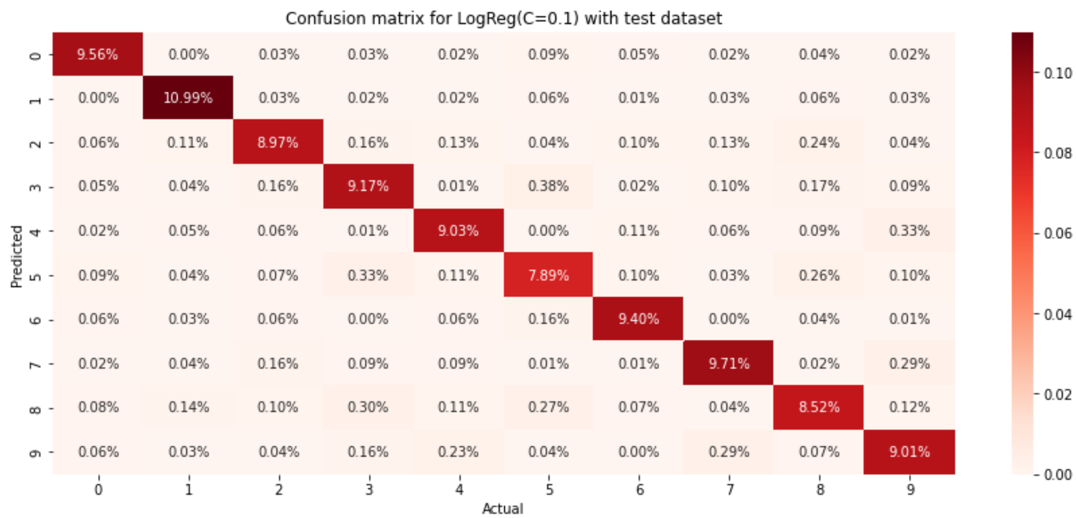

Fig-11: Confusion Matrix of LogReg with test dataset

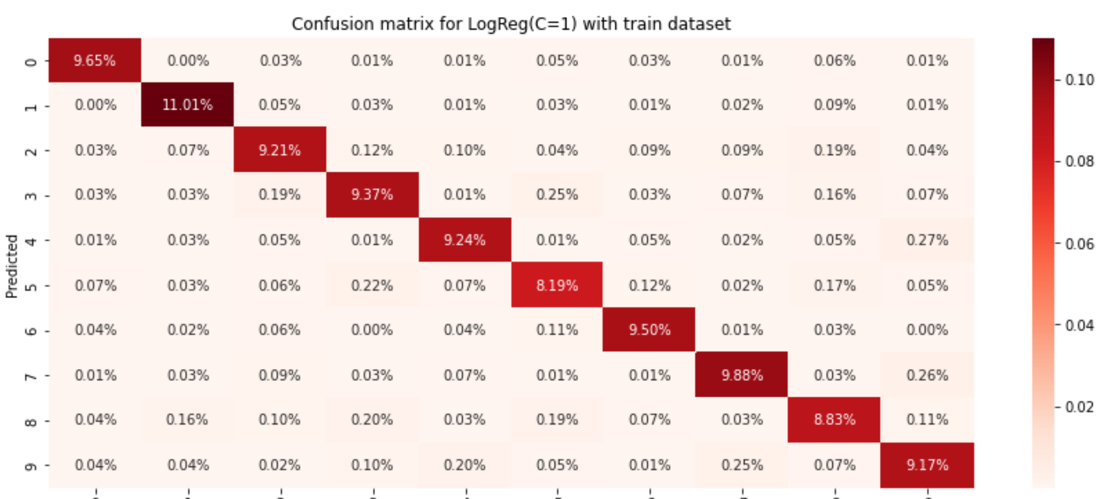

Fig-12: Confusion Matrix of LogReg with train dataset