ML (PG) ASSIGNMENT - 2
REPORT

Student Name: Drishti De
Degree: M.Tech
Roll No.: MT20075

CONTENTS

1. Approach

The approach used in this assignment is to analyze the type of data that is required to be dealt with in each of the questions. Based on the type of data, data preprocessing is done and the data is stored in final data-frame format using Pandas. These data frames obtained are then analyzed based on the instructions given in the questions. The main tasks involved in this assignment involve but may not be limited to the following:
i. Data pre-processing
ii. Data analysis
iii. Model training
iv. Visualizing the data or the results obtained after model training and evaluation.

JobLib is used to save the models created and replicate the results from the saved models. The models are imported from classes created in the *.py code files and objects for the classes are created and saved using joblib for replicating the results later.

Here, in order to maintain the modularity of the python codes for this assignment, the different functions required for model training and testing purposes are created and called as required. For instance, the n-fold split function is made in such a way as to call them as and when required to create five different train and test folds for validation.

The different datasets are uploaded into the coding interface (google colab) using Google Drive mounted into the colab interface.

2. Data Preprocessing

2.1 Dataset for Question 1 :

The data that Question 1 (on linear regression) deals with mostly is the data stored in the data file called “Dataset.data”. This is a 2D data where both the samples and the labels are in 2D. The labels in Dataset.data consist of different number labels such as: 1, 2, 3, ... 29.
Here the data is loaded with the help of the read_csv functionality of pandas using delimiters as spaces, skiprows as 0th row, header = ‘None’ and skip_blank_lines = True. This data is then explored and the column with character data is factorized before model training and validation on the data.

2.2 Dataset for Question 2 :

The data that Question 2 deals with is stored in the MATLAB file called “dataset_1.mat”. It is a 2-Dimensional data and therefore need not be reshaped to convert them into a pandas data frame. The labels are in 2D dimension therefore it is reshaped before including in the dataframe for data analysis. After converting them into a data frame, the shape of the data frame is 5000 rows and 3 columns (including the labels column).
This data is then utilized for model training and testing using logistic regression created manually and stored in a separate .py file and the accuracy is compared with that of sklearn to understand if our own made estimators have the same characteristics as that of the sklearn model.

2.3 Dataset for Question 3 :

The data that Question 3 deals with is stored in the MATLAB file called “dataset_2.mat”. It is a 2-Dimensional data and therefore need not be reshaped to convert them into a pandas data frame. The labels are in 2D dimension therefore it is reshaped before including in the dataframe for data analysis. After converting them into a data frame, the shape of the data frame is 10000 rows and 3 columns (including the labels column).

3. Analysis and Assumptions

This section of the report mainly focuses on the analysis results and visualizations obtained from the data or after model training and evaluation as per different the instructions given in each question.

3.1 For results in Question 1 :

Part (a): Here, the regression class is made as per instruction and imported to the Q1 py file for model training and validation of the data provided.

Part (b): In this question, the regression class is made as per instruction in regression.py to include the calculation for MSE and imported to be used in this py file for question 1. The MLEs obtained for all the 5 folds of training and validation sets are shown in the table given below:

	Validation_MSE	Validation_MSE_sk	Training_MSE	Training_MSE_sk
0	4.795719	4.795719	4.842452	4.842452
1	4.634395	4.634395	4.882984	4.882984
2	4.857885	4.857885	4.826641	4.826641
3	6.158319	6.158319	4.638720	4.638720
4	4.566846	4.566846	4.899562	4.899562

Here validation_MSE and training_MSE belong to the MSE function made inside the regression class whereas the Validation_MSE_sk and Training_MSE_sk columns belong to the MSE metrics of scikit-learn library.

From the above table, we see that the MSE implementation in the regression class and the one in scikit-learn library gives the same result for the 5 folds of the dataset provided. The mean MSEs for the columns above are as given below:

```
Validation MSE:  5.002633047274259
Validation MSE (sklearn):  5.002633047274259
Training MSE:  4.8180718797488655
Training MSE (sklearn):  4.818071879748866
```

Part (c): Here, as per the question, linear regression was implemented with the help of normal equations using the value of theta and then using it to predict the values of the target variable. A similar table to that of part b was generated for the MSEs obtained from from the predicted values as shown below:

	Validation_MSE	Validation_MSE_sk	Training_MSE	Training_MSE_sk
0	4.795719	4.795719	4.842452	4.842452
1	4.634395	4.634395	4.882984	4.882984
2	4.857885	4.857885	4.826641	4.826641
3	6.158319	6.158319	4.638720	4.638720
4	4.566846	4.566846	4.899562	4.899562

We can see from the above table that the values of MSE are exactly as that of part b and therefore the implementation for linear regression matches for both part(b) and part(c) respectively.

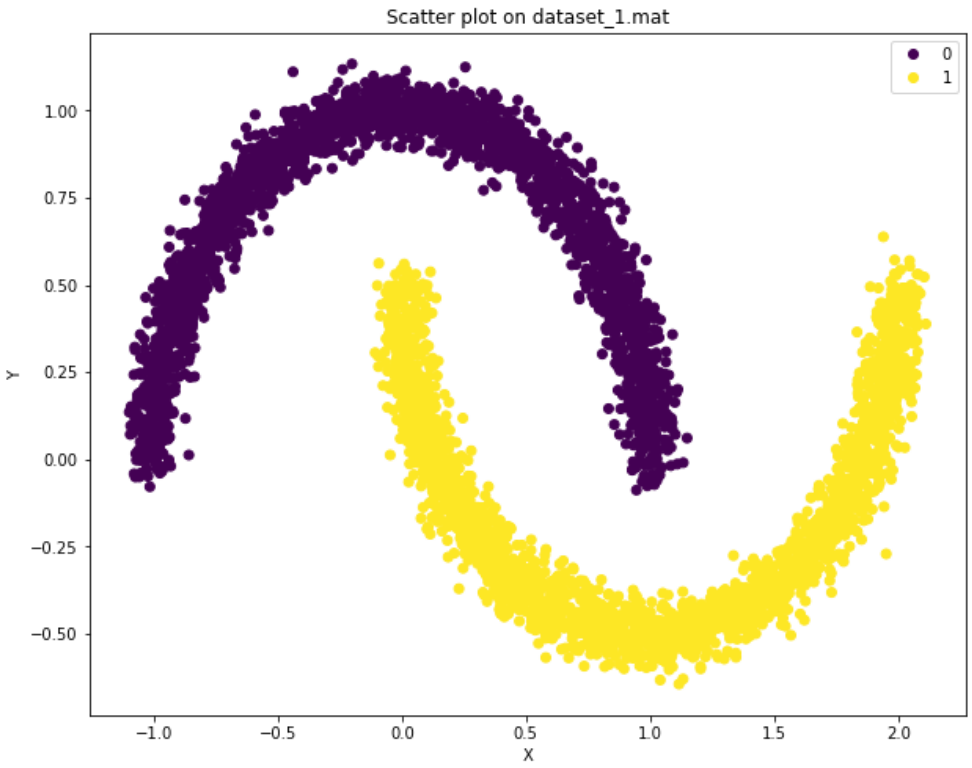
Part (d): In this part, the linear regression used is that of the scikit-learn library and then the MSEs of the predicted values are checked with the test values as similar to part(c) and (d) as shown in the following table:

	Validation_MSE	Validation_MSE_sk	Training_MSE	Training_MSE_sk
0	4.795719	4.795719	4.842452	4.842452
1	4.634395	4.634395	4.882984	4.882984
2	4.857885	4.857885	4.826641	4.826641
3	6.158319	6.158319	4.638720	4.638720
4	4.566846	4.566846	4.899562	4.899562

This table is similar to the values generated in part(b) and part(c) for the 5 folds of the test and training sets.

3.2 For results in Question 2 :

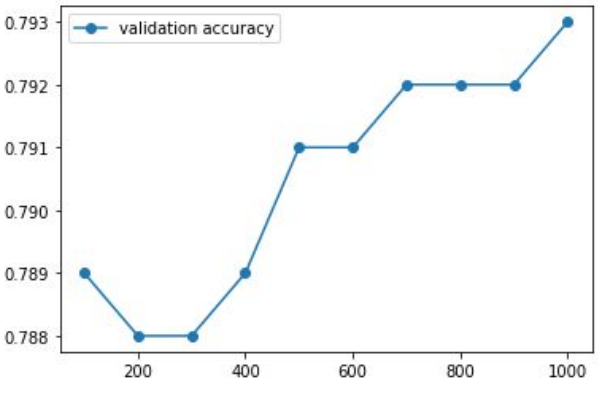
Part (a): The scatter plot for the given dataset (dataset_1.mat) is as shown below:

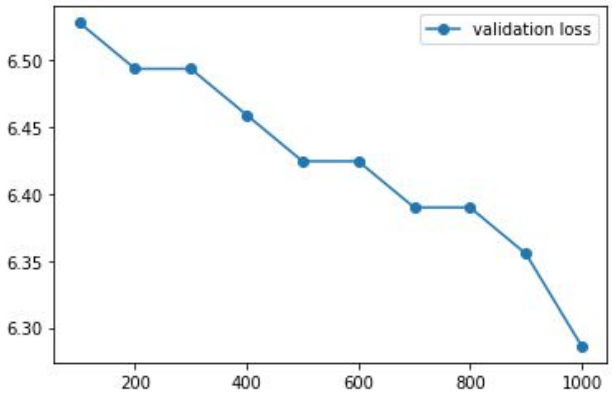


According to the plot legend, the purple scatter curve belongs to class label 0 and the yellow scatter curve belongs to class label 1.

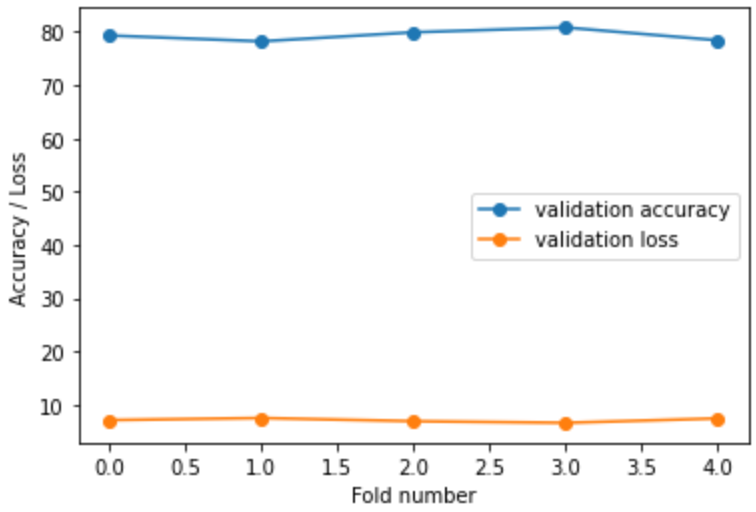
Part (b): As per the instruction given in the question, the logistic regression class is made without the usage of sklearn functionalities and imported into the Q2 py file.

Part (c): Here the logistic regression class model is implemented into the data for model training and validation in order to observe the training and the log loss curve for different number of iterations. These curves for a specific fold can be as shown below:

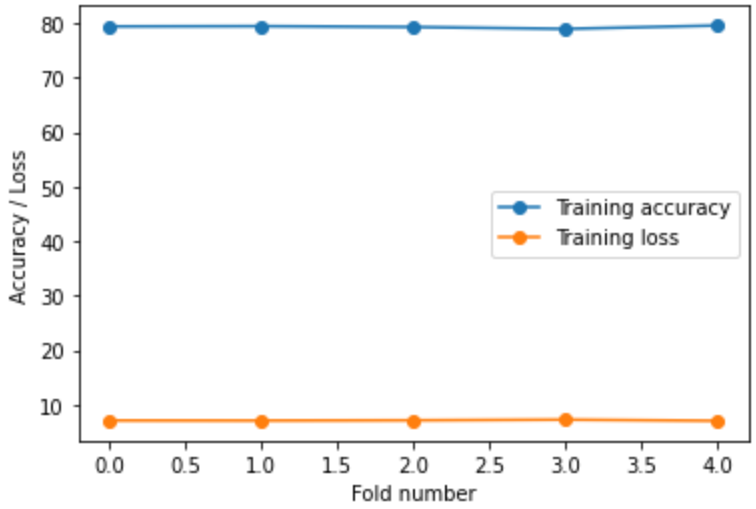




The validation accuracy and loss for all the 5 folds of validation are as shown in the plot below:



The training accuracy and loss for all the 5 folds of training data are shown below:



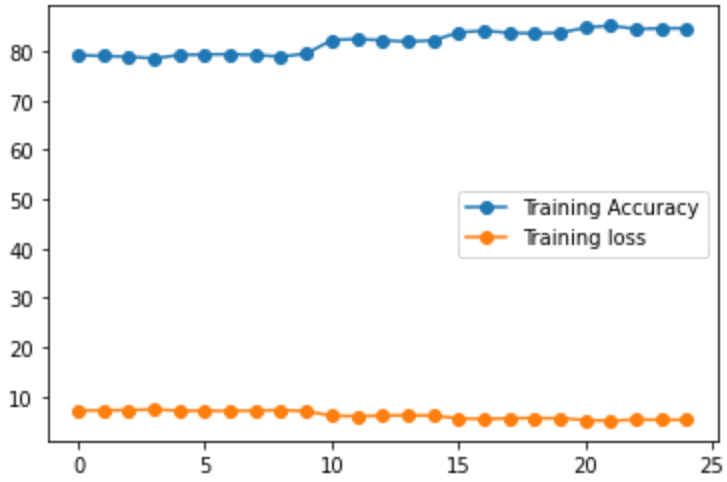
These plots show that for the 5 folds of validation and training data, the graph trend remains almost the same, which means that the accuracy and loss for the five folds are similar in nature.

Part (d): Now, in order to find the optimal lambda (regularization constant) in l2 regularization, we try to input the lambda value for all the 5 folds of the validation and training data. The values of lambda given are: 0.0001, 0.001, 0.01, 0.02, 0.03 for all the 5 folds. The generated table is as shown below:

	Lambda Value	Validation Accuracy	Validation loss	Training Accuracy	Training loss
0	0.0001	79.0	7.253255	79.200	7.184180
1	0.0001	78.2	7.529581	79.050	7.235990
2	0.0001	79.2	7.184174	78.900	7.287794
3	0.0001	80.5	6.735172	78.550	7.408688
4	0.0001	78.1	7.564113	79.250	7.166910
5	0.0010	79.3	7.149632	79.350	7.132365
6	0.0010	78.2	7.529576	79.400	7.115096
7	0.0010	79.9	6.942394	79.250	7.166901
8	0.0010	80.8	6.631545	78.900	7.287795
9	0.0010	78.4	7.460491	79.525	7.071920
10	0.0100	82.7	5.975281	82.275	6.122071
11	0.0100	80.5	6.735144	82.475	6.052994
12	0.0100	82.4	6.078896	82.225	6.139340
13	0.0100	83.3	5.768043	81.950	6.234325
14	0.0100	82.4	6.078899	82.200	6.147976
15	0.0200	84.3	5.422648	83.800	5.595344
16	0.0200	81.8	6.286129	84.175	5.465823
17	0.0200	84.1	5.491725	83.725	5.621248
18	0.0200	84.8	5.249952	83.625	5.655788
19	0.0200	84.0	5.526267	83.725	5.621248
20	0.0300	85.2	5.111794	84.775	5.258586
21	0.0300	82.7	5.975276	85.225	5.103160
22	0.0300	85.2	5.111792	84.500	5.353569
23	0.0300	85.6	4.973637	84.650	5.301761
24	0.0300	85.1	5.146336	84.675	5.293125

Therefore, according to the table shown above, the best value for lambda for all the 5 folds seems to be 0.03, which can be used further in logistic regression for better results in the data.

The plot for accuracy and loss for the above table is as shown below:



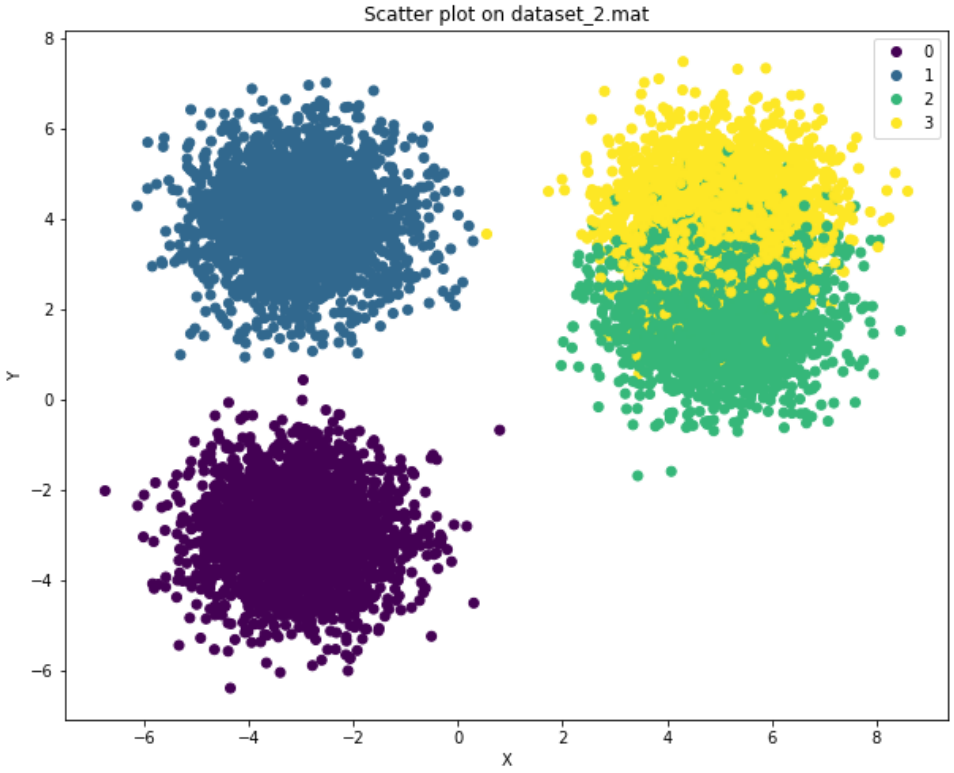
Part (e): Using scikit-learn library, we calculate the accuracy and loss for both the training and validation folds as shown below:

	Validation Accuracy (sklearn)	Validation Log-loss (sklearn)	Training Accuracy (sklearn)	Training Log-loss (sklearn)
0	88.5	3.972005	89.200	3.730231
1	88.7	3.902932	89.100	3.764770
2	89.6	3.592075	88.950	3.816578
3	89.5	3.626613	88.925	3.825215
4	89.2	3.730226	89.050	3.782041

As we see from the table above, the accuracies have enhanced for the lambda value of 0.03 using l2 regularization.

3.3 For results in Question 3 :

Part (a): The scatter plot for this question is as shown below:



The above plot shows 4 clusters for 4 class labels namely 0, 1, 2 and 3 with different colours representing each of the class labels.

Part (b): A different py file "LogRegression_OVO1" is made for incorporating One-vs-one classification in logistic regression class. Since in this type of classification, we look at each possible pairs of labels and assign one of them as 1 and the other as 0, therefore the new functionality in the logistic regression class returns the pairs for which we need to classify one label as 1 and the other as 0 and put them for model training and validation for all the different folds.

The table for validation and training accuracy of each fold and pairs of class labels are as shown below:

Validation Accuracy:
(Mean: 97.10114977214312)

	Sample No.	Class i	Class j	Accuracy
0	0	0	1	100.000000
1	0	0	2	100.000000
2	0	0	3	100.000000
3	0	1	2	100.000000
4	0	1	3	100.000000
5	0	2	3	81.835749
6	1	0	1	99.902153
7	1	0	2	100.000000
8	1	0	3	100.000000
9	1	1	2	100.000000
10	1	1	3	100.000000
11	1	2	3	83.742331
12	2	0	1	100.000000
13	2	0	2	100.000000
14	2	0	3	100.000000
15	2	1	2	100.000000
16	2	1	3	100.000000
17	2	2	3	83.042636
18	3	0	1	100.000000
19	3	0	2	100.000000
20	3	0	3	100.000000
21	3	1	2	100.000000
22	3	1	3	100.000000
23	3	2	3	82.801236
24	4	0	1	100.000000
25	4	0	2	99.902913
26	4	0	3	100.000000
27	4	1	2	100.000000
28	4	1	3	99.896907
29	4	2	3	81.910569

Training Accuracy:
(Mean: 97.11593662910475)

	Sample No.	Class i	Class j	Accuracy
0	0	0	1	99.975217
1	0	0	2	99.975050
2	0	0	3	100.000000
3	0	1	2	100.000000
4	0	1	3	99.974950
5	0	2	3	82.976040
6	1	0	1	100.000000
7	1	0	2	99.974893
8	1	0	3	100.000000
9	1	1	2	100.000000
10	1	1	3	99.975106
11	1	2	3	82.471407
12	2	0	1	99.975198
13	2	0	2	99.975143
14	2	0	3	100.000000
15	2	1	2	100.000000
16	2	1	3	99.974855
17	2	2	3	82.711694
18	3	0	1	99.974817
19	3	0	2	99.975100
20	3	0	3	100.000000
21	3	1	2	100.000000
22	3	1	3	99.974900
23	3	2	3	82.675602
24	4	0	1	99.974900
25	4	0	2	100.000000
26	4	0	3	100.000000
27	4	1	2	100.000000
28	4	1	3	100.000000
29	4	2	3	82.943227

Part (c): A different py file "LogRegression_OVR1" is made for incorporating One-vs-rest classification in logistic regression class. Since in this type of classification, we look at each label and assign them as 1 and the other labels as 0, therefore the new functionality in the logistic regression class takes one label at a time and assigns it as 1 and the rest labels as zero and returns the target variable class labels.

The table for validation and training accuracy of each fold and pairs of class labels are as shown below:

Validation Accuracy (mean: 94.0375)

	Sample No.	Class Number	Accuracy
0	0	0	99.90
1	0	1	99.35
2	0	2	86.95
3	0	3	88.50
4	1	0	100.00
5	1	1	99.35
6	1	2	88.55
7	1	3	89.75
8	2	0	99.95
9	2	1	99.50
10	2	2	87.40
11	2	3	88.80
12	3	0	100.00
13	3	1	99.50
14	3	2	87.45
15	3	3	90.55
16	4	0	99.95
17	4	1	98.95
18	4	2	87.60
19	4	3	88.75

Training Accuracy (mean: 94.0275)

	Sample No.	Class Number	Accuracy
0	0	0	99.9750
1	0	1	99.3625
2	0	2	87.7250
3	0	3	89.3125
4	1	0	99.9500
5	1	1	99.3250
6	1	2	87.3500
7	1	3	89.0875
8	2	0	99.9625
9	2	1	99.2875
10	2	2	87.5750
11	2	3	89.5375
12	3	0	99.9500
13	3	1	99.3250
14	3	2	87.5500
15	3	3	89.2250
16	4	0	99.9625
17	4	1	99.3875
18	4	2	87.5000
19	4	3	89.2000

Part (d): Now, implementing One-vs-one and One-vs-rest classifier in logistic regression using appropriate sklearn libraries, we get the following tables for validation and training accuracy as shown below:

1. One-vs-one classifier in Logistic regression

	Fold Number	SkLearn_OVO_Test_Accuracy
0	0	96.20
1	1	97.95
2	2	97.20
3	3	97.15
4	4	97.00

(mean: 97.1)

	Fold Number	SkLearn_OVO_Train_Accuracy
0	0	97.3109
1	1	96.8859
2	2	97.0984
3	3	97.0734
4	4	97.2109

(mean: 97.1159)

2. One-vs-rest classifier in Logistic Regression

	Fold Number	SkLearn_OVR_Test_Accuracy
0	0	93.3875
1	1	94.8875
2	2	93.9875
3	3	94.0875
4	4	93.8375

(mean: 94.0375)

	Fold Number	SkLearn_OVR_Train_Accuracy
0	0	94.2475
1	1	93.8475
2	2	93.9975
3	3	93.9475
4	4	94.0975

(mean: 94.0275)

which is similar to the ones found while implementing OVO and OVR using logistic regression class in .py file.