Q1)

a) I have written the predict function in the file named **Regression.py** which had regression class already made. I have tried to make the prediction using the standard linear regression formula. I have implemented fit using the sklearn library functions.

b)

	Fold	Train MSE	Validation MSE
0	5	5.117855	4.125120
1	1	3.838241	10.088946
2	2	5.406304	3.152181
3	3	4.711082	5.962976
4	4	5.191065	3.843438

The above table has been created using the regression class that was prepared in previous step.

- Firstly I have imported the dataset into a dataframe.
- Then separated target and features labels into different variables .
- Since there was only space delimiter I had to separate using r"\s+"
- I also implemented the encoding as data in first column was not numerical and was mostly of 3 types.
- Fold split was implemented using a different functions where I divided the dataframe into 5 parts.
- For each fold separately I have divided the fold and taken into consideration.
- MSE functions I have implemented numpy library subtracting both y and ypred and squaring them.
- I have taken mean of all fold MSE to calculate mean MSE for training and validation.
- Output from both mse and mse from sklearn are almost the same.

```
Training MSE Mean : 4.852910
Validation MSE Mean:5.434532
```

c) With the normal equation I have used the following formula.

$$\theta = \left(X^T X\right)^{-1} . \left(X^T y\right)$$

- Firstly I have imported the dataset into a dataframe.
- Then separated target and features labels into different variables .

- Since there was only space delimiter I had to separate using r"\s+"
- I also implemented the encoding as data in first column was not numerical and was mostly of 3 types.
- Fold split was implemented using a different functions where I divided the dataframe into 5 parts.
- For each fold separately I have divided the fold and taken into consideration.
- Then I have implemented the normal function for each fold separately.
- Calculated their predicted training and test sets.
- For each fold separately I have divided the fold and taken into consideration.
- MSE functions I have implemented numpy library subtracting both y and ypred and squaring them.

	Fold	Train MSE	Validation MSE
0	5	5.259382	4.218434
1	1	3.993254	10.185761
2	2	5.526878	3.342580
3	3	4.801772	6.325747
4	4	5.314625	4.009159

Number obtained here after the calculation here are similar and very close to the number that have appeared 1b.

d)

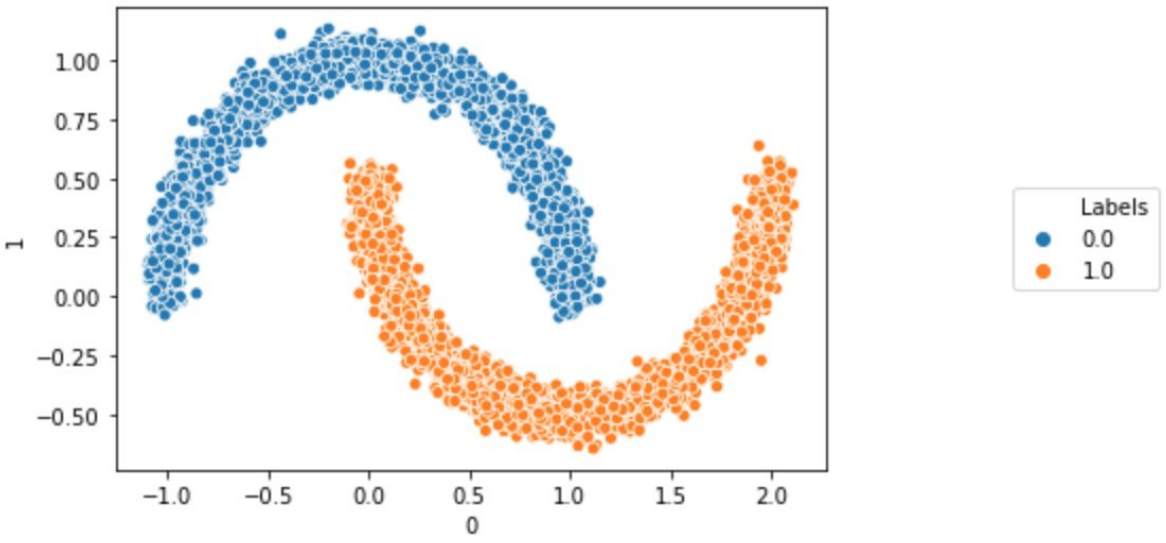
- Firstly I have imported the dataset into a dataframe.
- Then separated target and features labels into different variables .
- Since there was only space delimiter I had to separate using r"\s+"
- I also implemented the encoding as data in first column was not numerical and was mostly of 3 types.
- Fold split was implemented using a different functions where I divided the dataframe into 5 parts.
- For each fold separately I have divided the fold and taken into consideration.
- MSE functions I have implemented numpy library subtracting both y and ypred and squaring them.
- I have taken mean of all fold MSE to calculate mean MSE for training and validation.
- Output from both mse and mse from sklearn are almost the same.

	Fold	Train MSE	Validation MSE
0	5	5.117855	4.125120
1	1	3.838241	10.088946
2	2	5.406304	3.152181
3	3	4.711082	5.962976
4	4	5.191065	3.843438

The performance is similar to that of above calculations. So there is very minor difference in points and so performance is similar as given above.

Q2)

a)



- The data is in mat format so I have first imported loadmat with scipy.io
- Then used the loadmat to upload the data into a variable.
- Then I have samples and the labels into different variables.
- Then I have combined them together using numpy column_stack
- Further I have converted it into the dataframe.
- I have used then scatterplot from seaborn library to draw the above scatterplot.

b)

- I have implemented the logistic regression from the scratch as per given in the file.

- I have used the sigmoid functions and various other methods as per the logistic regression.

$$h_\theta(\mathbf{x}) = g(\theta^\top \mathbf{x})$$

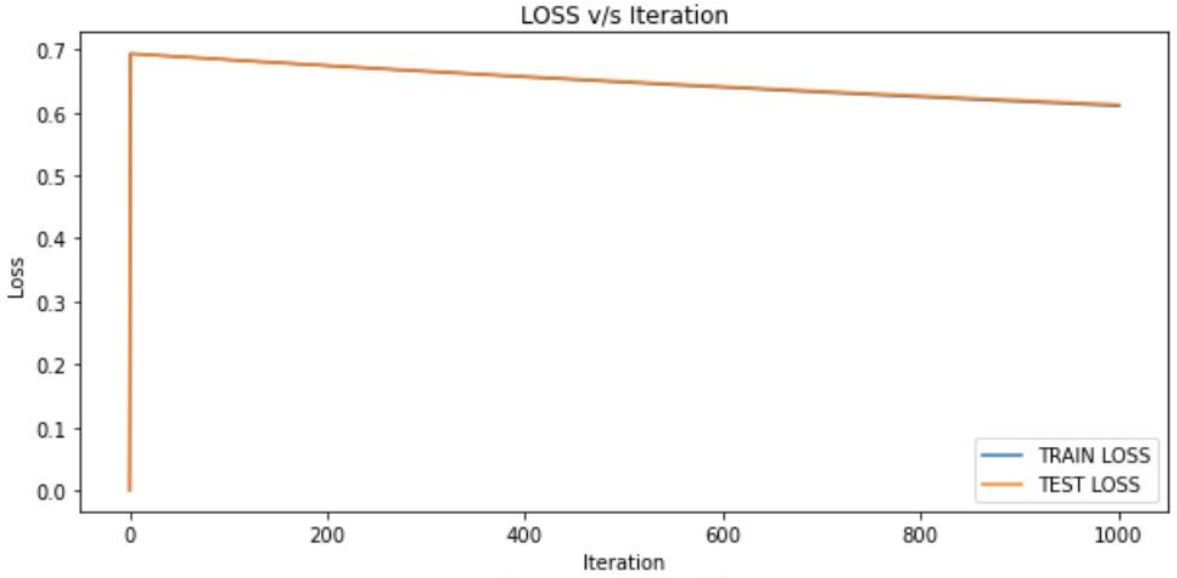
$$g(z) = \frac{1}{1 + e^{-z}}$$

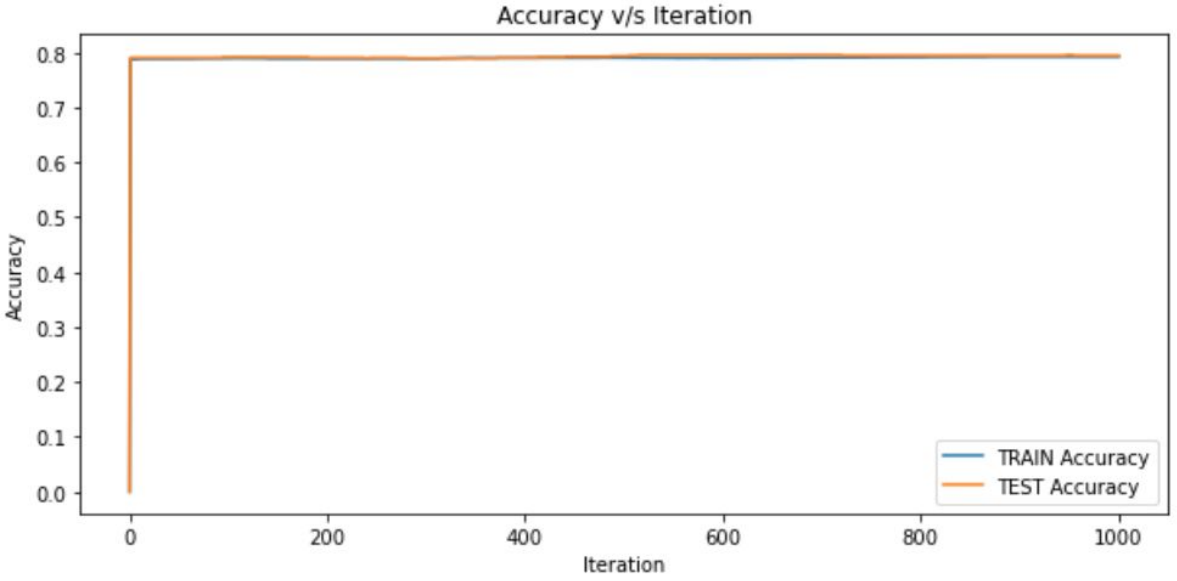
- Sigmoid function gives value between 0 and 1
- Linear models are further passed to this sigmoid formula to perform the task.

c)

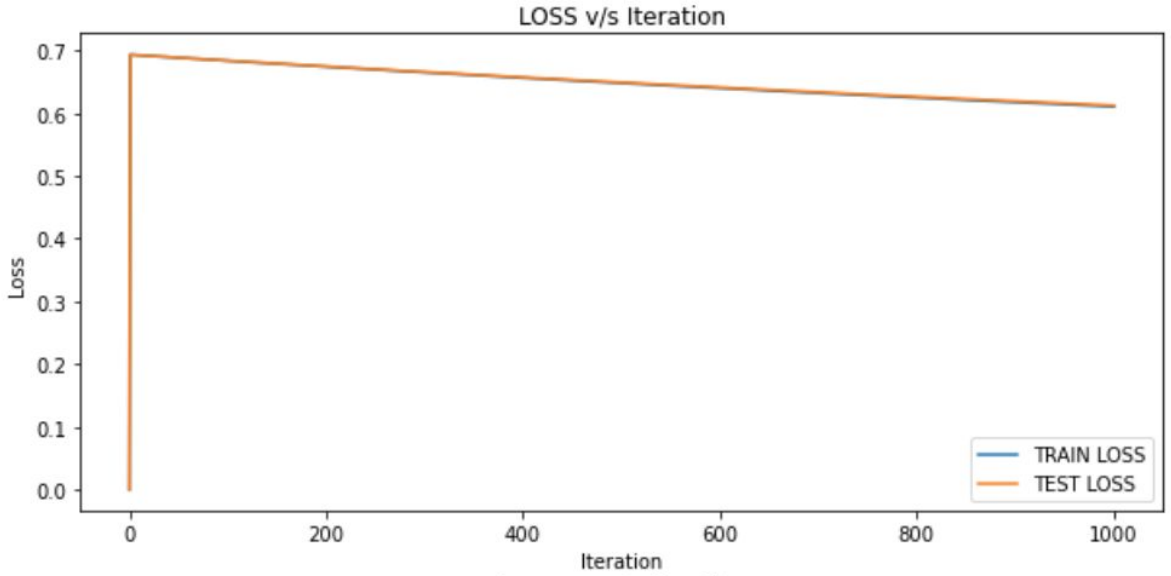
- The data is in mat format so I have first imported loadmat with scipy.io
- Then used the loadmat to upload the data into a variable.
- Then I have samples and the labels into different variables.
- Then I have combined them together using numpy column_stack
- Further I have converted it into the dataframe.
- Further I have divided the data into features and target variables as per the given scenario
- Fold split was implemented using a different functions where I divided the dataframe into 5 parts both for training and testing data.
- Further I have used the model that I have implemented in previous part.
  Then added further components into it.
- Divide into folds. Created different model for each of the fold and further calculated the train loss and test loss for each of the case
- The also calculated Accuracy loss and Accuracy Test for each of the case.
- Further for each fold its calculated so its gives 10 different graphs and a table for accuracy.

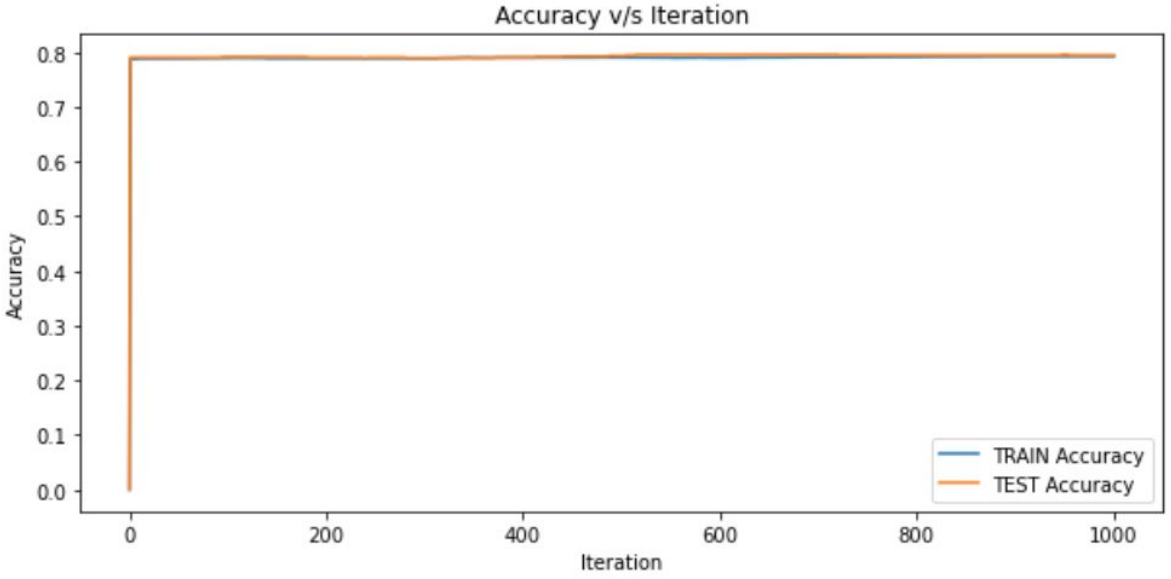
For Fold 5


LOSS v/s Iteration
Loss
Iteration
TRAIN LOSS
TEST LOSS
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
200
400
600
800
1000

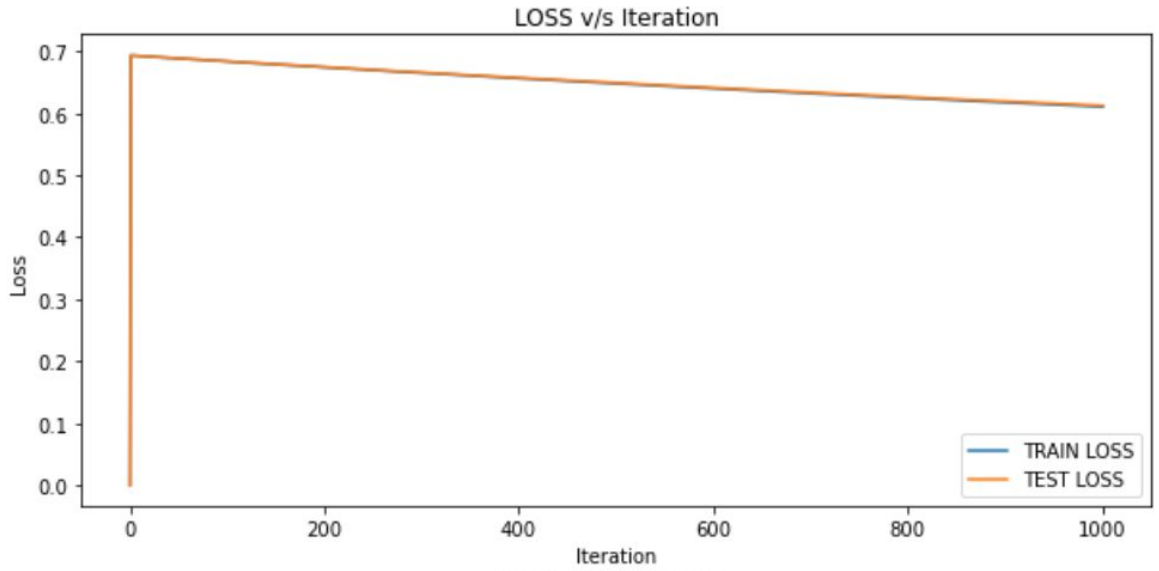

Accuracy v/s Iteration
Accuracy
Iteration
TRAIN Accuracy
TEST Accuracy
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
200
400
600
800
1000

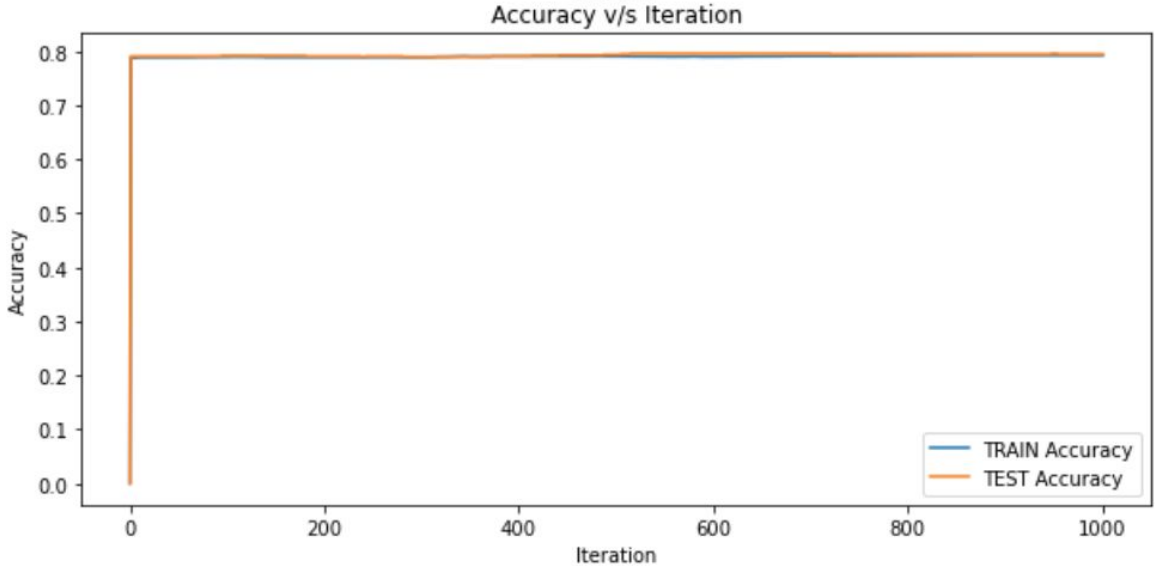
For Fold 1


LOSS v/s Iteration
Loss
Iteration
TRAIN LOSS
TEST LOSS
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
200
400
600
800
1000

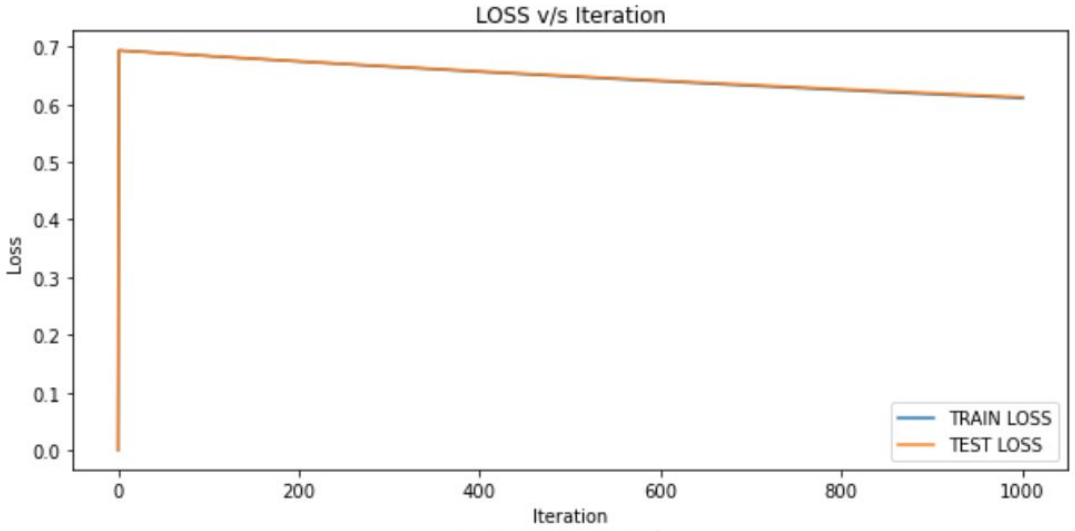

Accuracy v/s Iteration
Accuracy
Iteration
TRAIN Accuracy
TEST Accuracy
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
200
400
600
800
1000

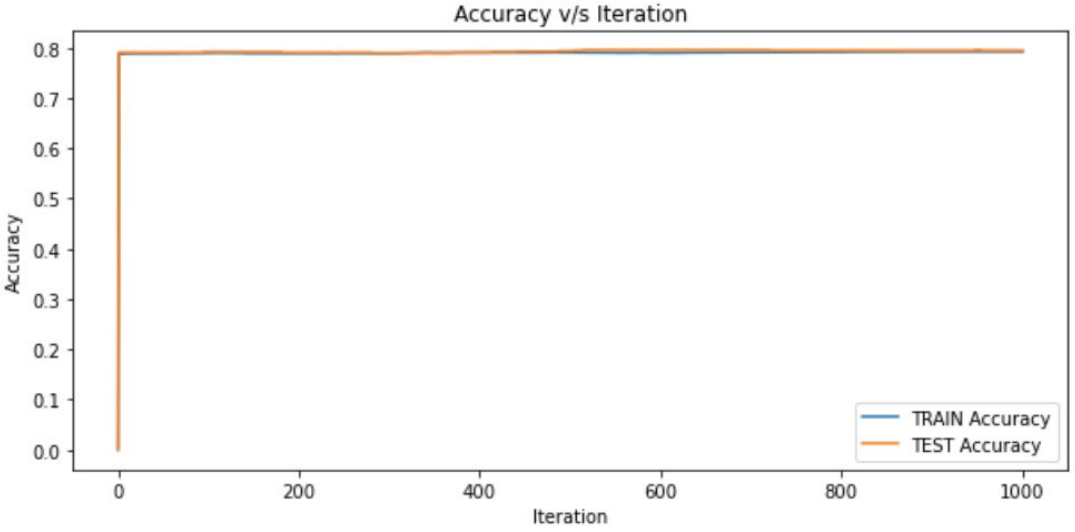
For Fold 2


LOSS v/s Iteration
Loss
Iteration
TRAIN LOSS
TEST LOSS
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
200
400
600
800
1000

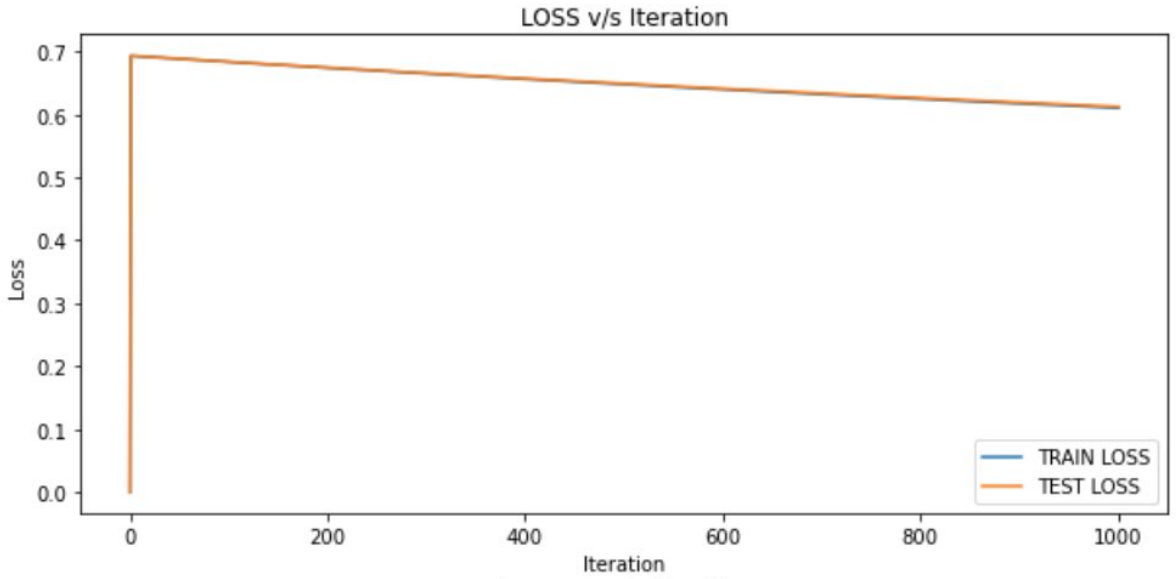

Accuracy v/s Iteration
Accuracy
Iteration
TRAIN Accuracy
TEST Accuracy
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
200
400
600
800
1000

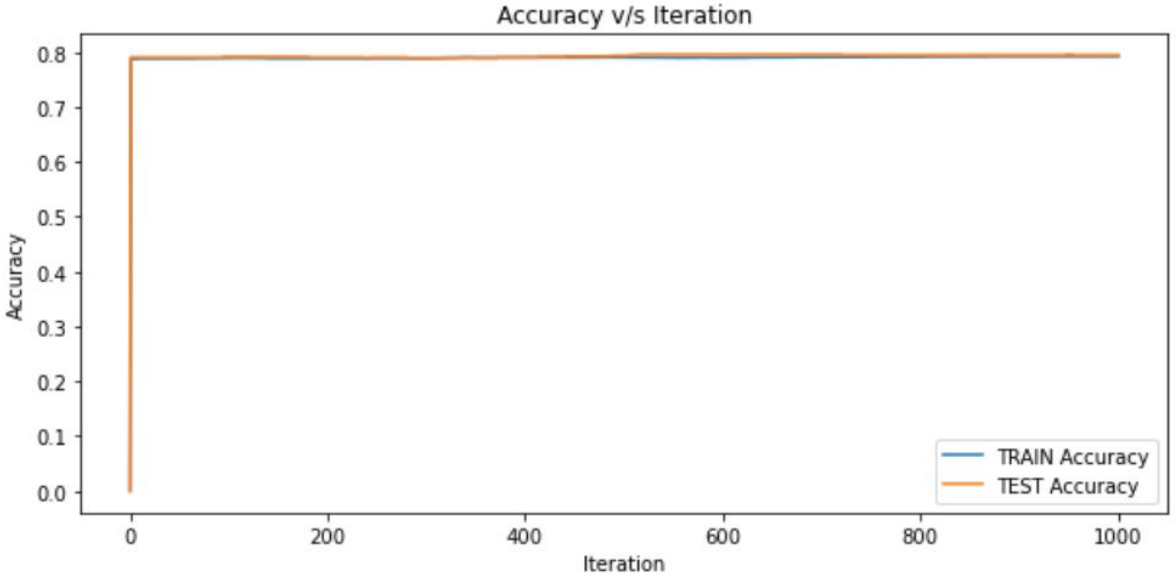
For Fold 3


LOSS v/s Iteration
Loss
Iteration
TRAIN LOSS
TEST LOSS
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
200
400
600
800
1000


Accuracy v/s Iteration
Accuracy
Iteration
TRAIN Accuracy
TEST Accuracy
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
200
400
600
800
1000

For Fold 4



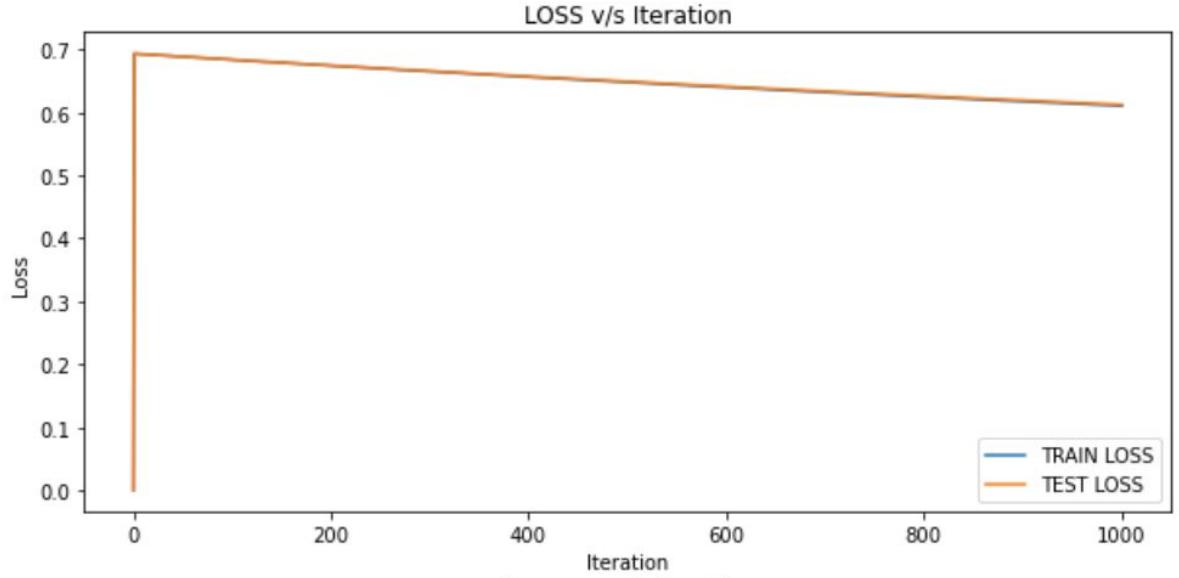


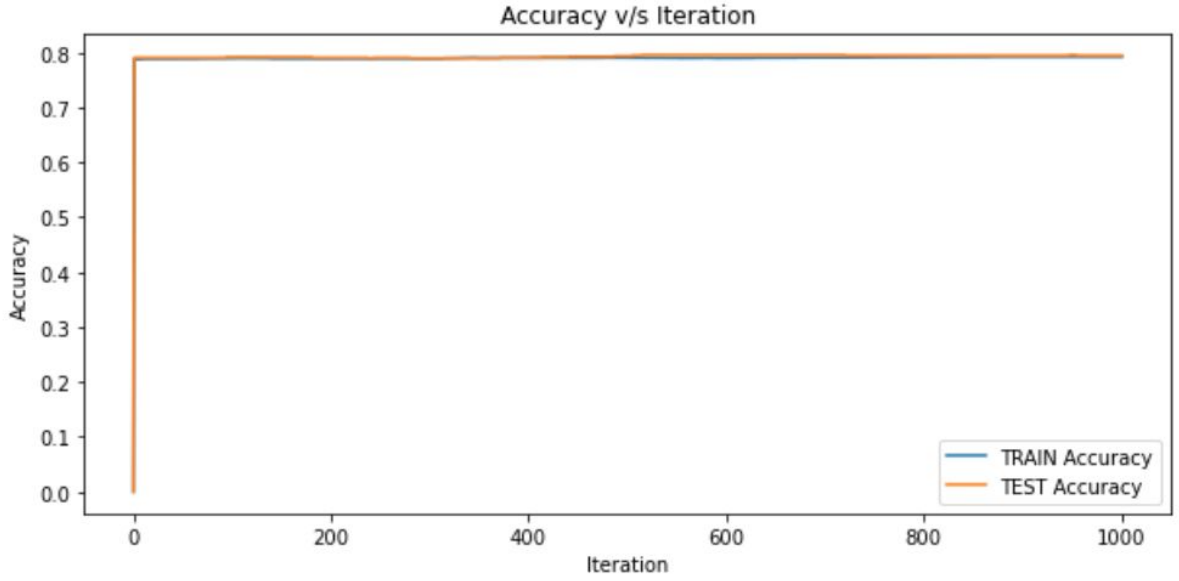
	Fold	Train Accuracy	Test Accuracy
0	5	0.79275	0.793
1	1	0.79325	0.805
2	2	0.79700	0.770
3	3	0.79225	0.795
4	4	0.79075	0.804

d)

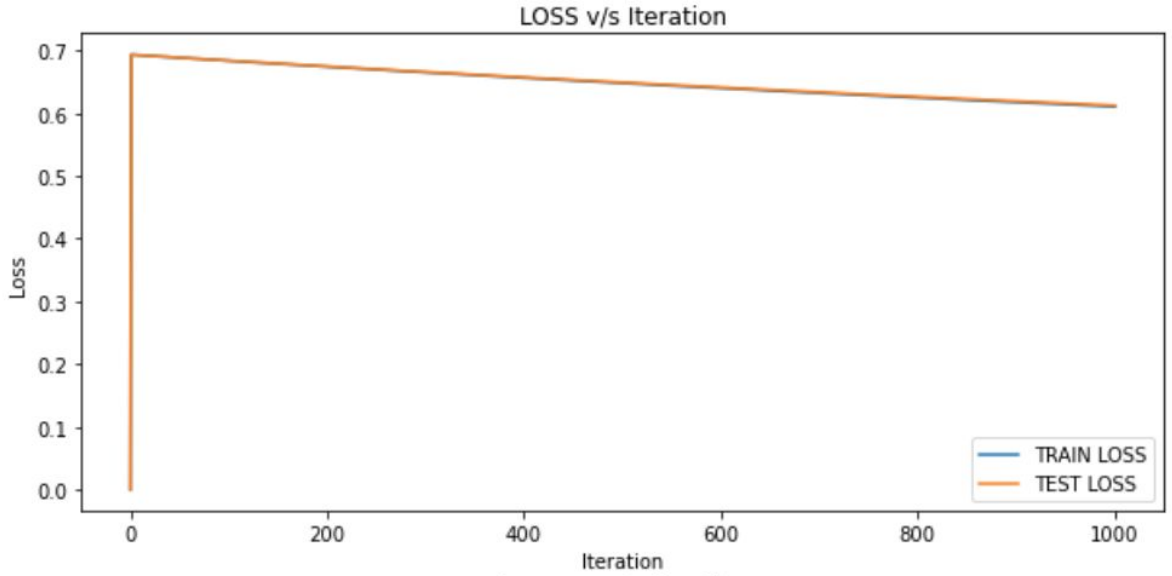
- The regularisation part improves the accuracy and further gives a better model based on better weights for the prediction.
- The data is in mat format so I have first imported loadmat with scipy.io
- Then used the loadmat to upload the data into a variable.
- Then I have samples and the labels into different variables.
- Then I have combined them together using numpy column_stack
- Further I have converted it into the dataframe.
- Further I have divided the data into features and target variables as per the given scenario
- Fold split was implemented using a different functions where I divided the dataframe into 5 parts both for training and testing data.
- Further I have used the model that I have implemented in previous part. Then added further components into it.
    - Divide into folds. Created different model for each of the fold and further calculated the train loss and test loss for each of the case
    - The also calculated Accuracy loss and Accuracy Test for each of the case.
    - Further for each fold its calculated so its gives 10 different graphs and a table for accuracy.

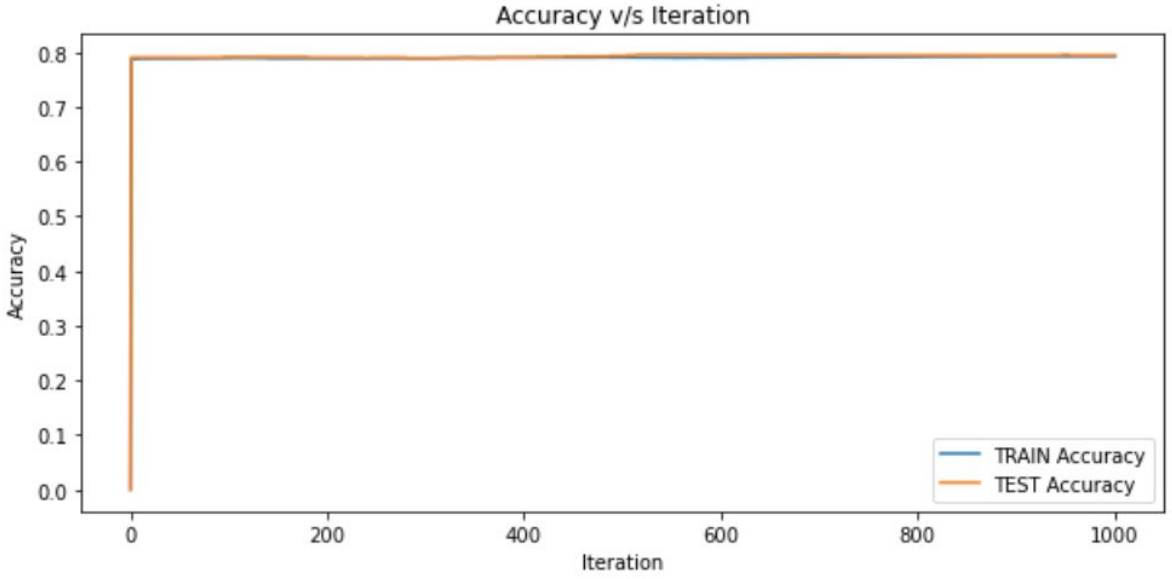
For Fold 5


LOSS v/s Iteration
Loss
Iteration
TRAIN LOSS
TEST LOSS
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
200
400
600
800
1000

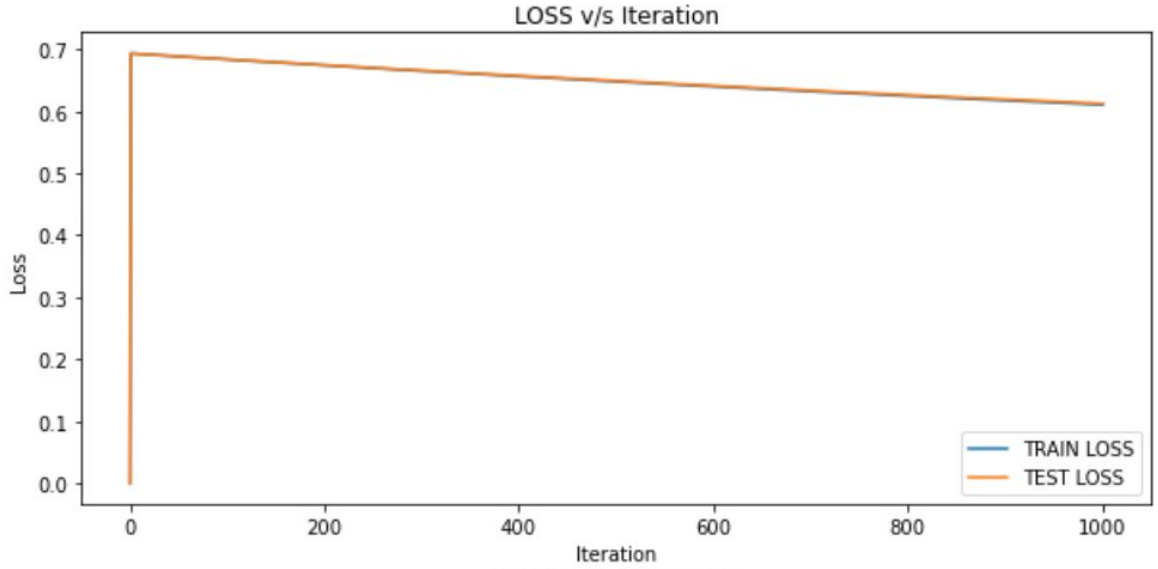

Accuracy v/s Iteration
Accuracy
Iteration
TRAIN Accuracy
TEST Accuracy
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
200
400
600
800
1000

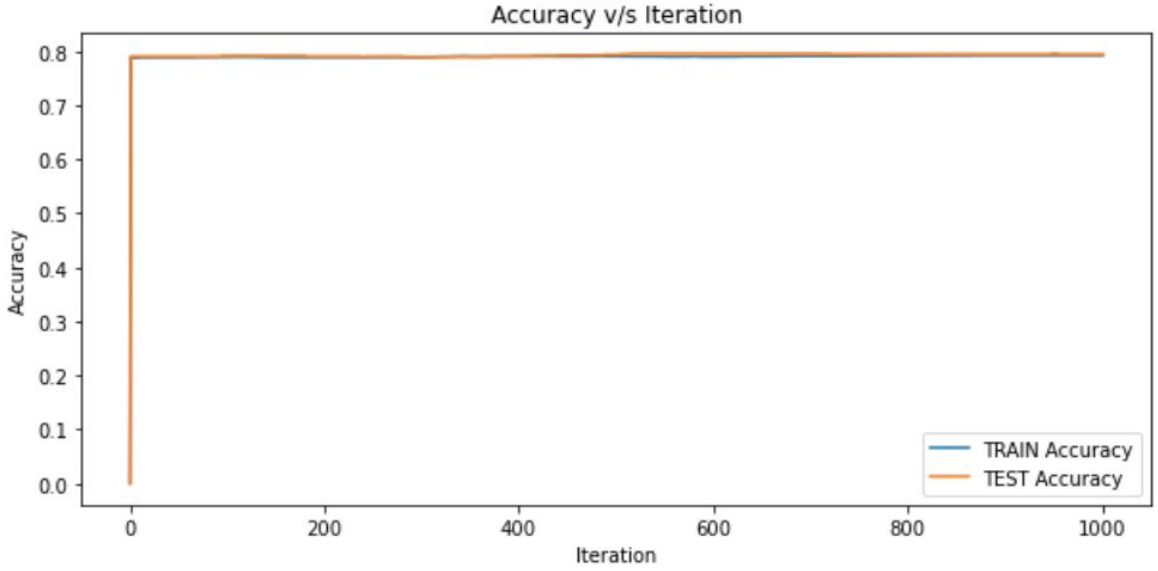
For Fold 1


LOSS v/s Iteration
Loss
Iteration
TRAIN LOSS
TEST LOSS
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
200
400
600
800
1000

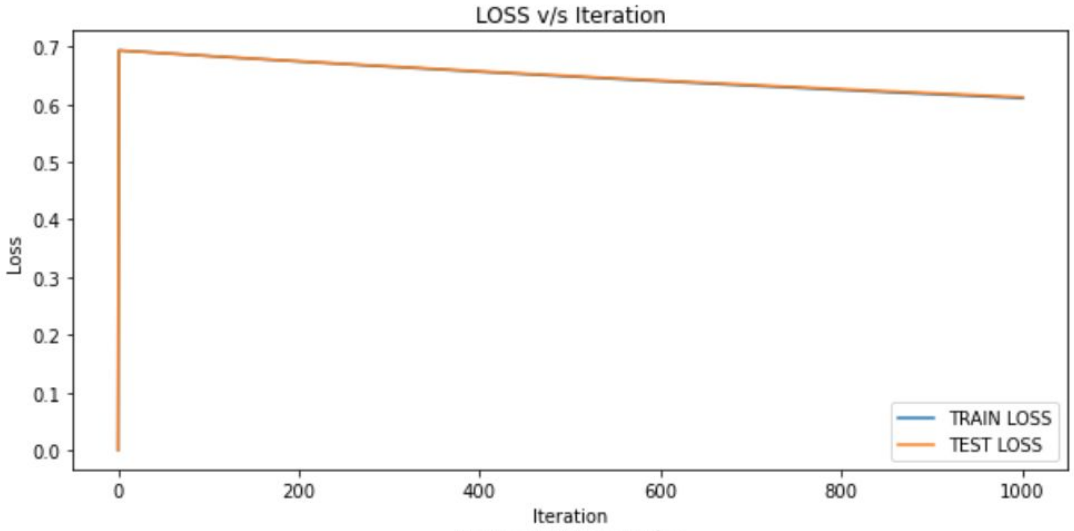

Accuracy v/s Iteration
Accuracy
Iteration
TRAIN Accuracy
TEST Accuracy
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
200
400
600
800
1000

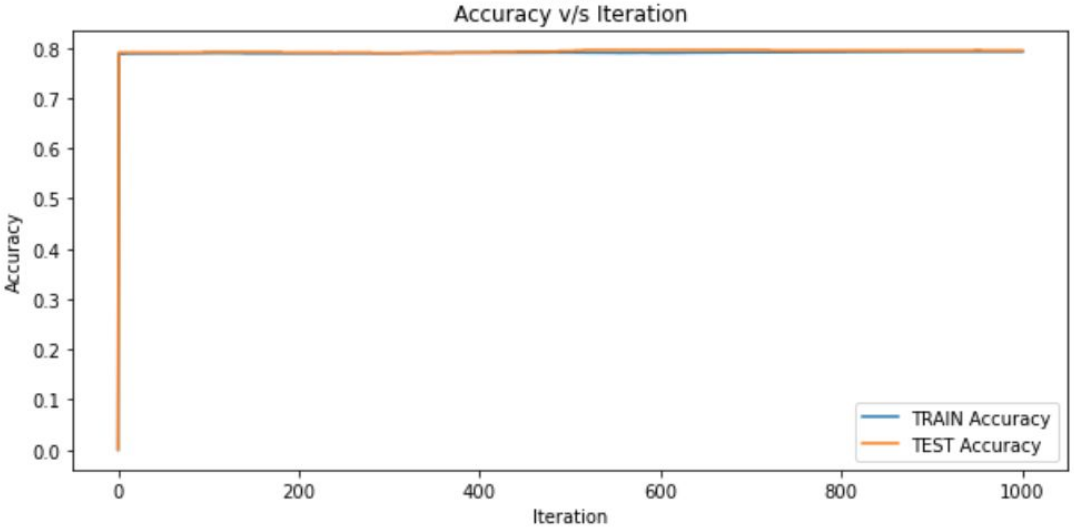
For Fold 2


LOSS v/s Iteration
Loss
Iteration
TRAIN LOSS
TEST LOSS
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
200
400
600
800
1000

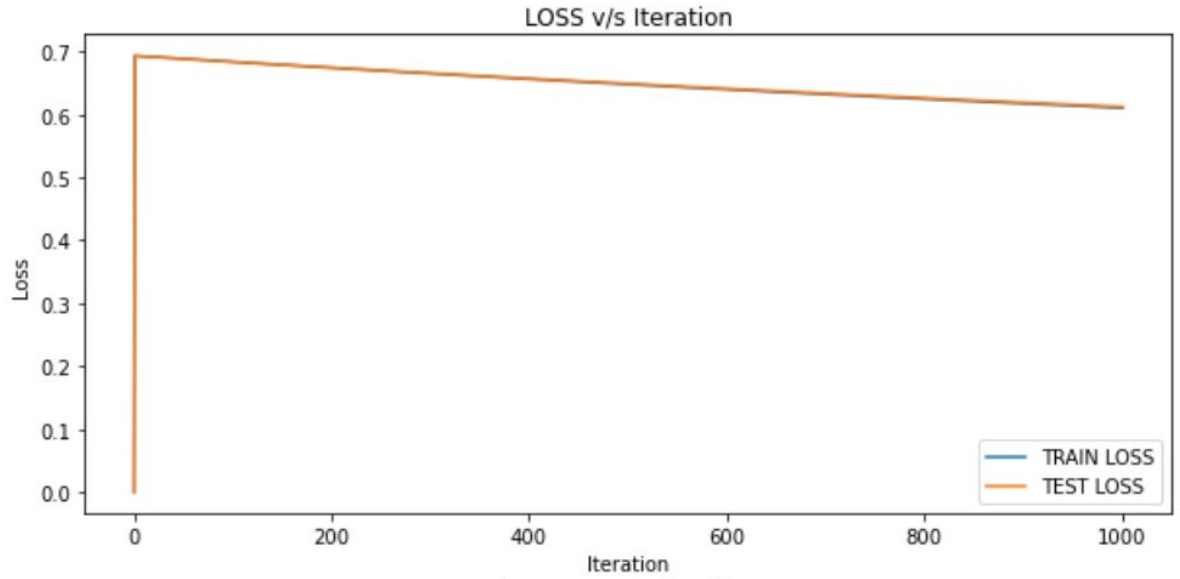

Accuracy v/s Iteration
Accuracy
Iteration
TRAIN Accuracy
TEST Accuracy
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
200
400
600
800
1000

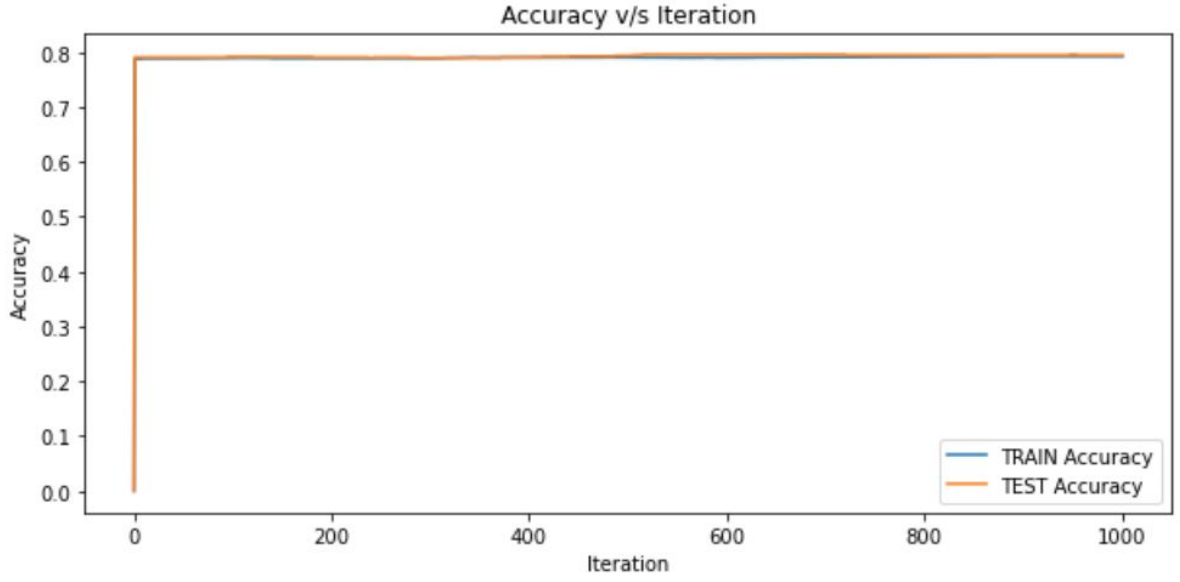
For Fold 3


LOSS v/s Iteration
Loss
Iteration
TRAIN LOSS
TEST LOSS
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
200
400
600
800
1000


Accuracy v/s Iteration
Accuracy
Iteration
TRAIN Accuracy
TEST Accuracy
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
200
400
600
800
1000

For Fold 4

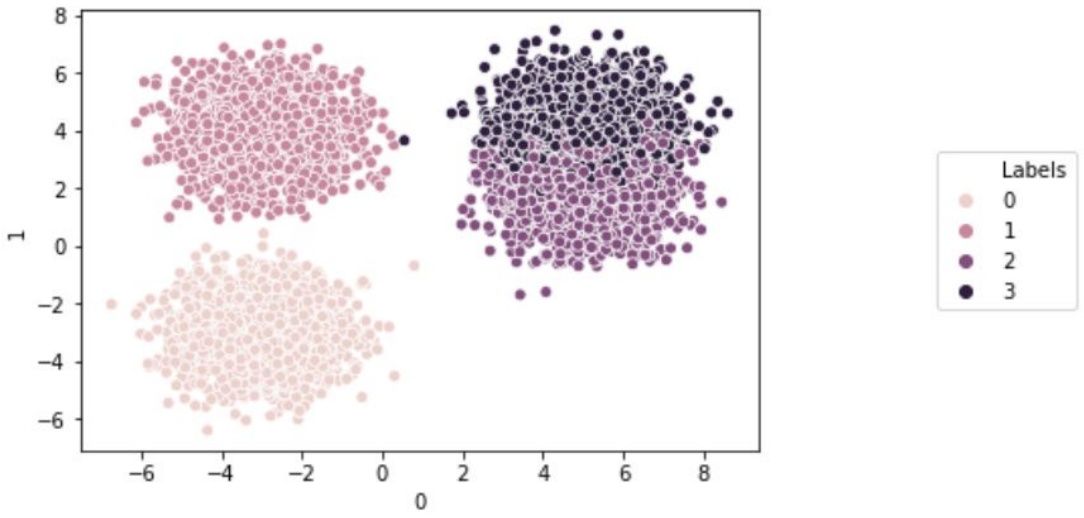




3)

a)

- The data is in mat format so I have first imported loadmat with scipy.io
- Then used the loadmat to upload the data into a variable.
- Then I have samples and the labels into different variables.
- Then I have combined them together using numpy column_stack
- Further I have converted it into the dataframe.
- I have used then scatterplot from seaborn library to draw the above scatterplot.


8
6
4
2
0
−2
−4
−6
1
−6
−4
−2
0
2
4
6
8
0
Labels
0
1
2
3