Assignment2i.py runs on test.csv and train.csv that was provided.

Assignment2ii.py runs on the Haberman dataset.(<https://archive.ics.uci.edu/ml/datasets/Haberman's+Survival>

The dataset is Large\_train.csv.

Information about the dataset is present in Haberman.txt

In Large\_train.csv

X1: Age of patient at time of operation (numerical)

X2: Patient's year of operation (year - 1900, numerical)

X3: Number of positive axillary nodes detected (numerical)

y: Survival status (class attribute)

1 = the patient survived 5 years or longer

2 = the patient died within 5 year

**# TODO: Remember to implement the preprocess method:**

We split the dataset into train and test portion using **model\_selection.train\_test\_split(df, test\_size=0.2,shuffle= True, stratify=df["y"]).** Thereis 73.5% of Class 1 and 26.5% of Class 2.We have stratified on the output data so that the train and test data also has 73.5% of class 1 and 26.5% of class 2.

In the preprocess method we have implemented Min Max scaling on the input data and changed the output data from 1,2 to 0,1 where 0 indicates the patient survived 5 years or longer and 1 indicates the patient died within 5 years.

Our data didn’t have missing values or categorical input. If there were missing values we could have replaced by mean or median or interpolate based on the data provided. For standardization we can deduct the mean from each data divide by standard deviation for that attribute. Since all the input attributes were of different type like age, year and number of positive axillary nodes detected we chose to do Min Max Scaling.

If there was numerical data present we could have performed one hot encoding.

**# TODO: Define the function for tanh, ReLu and their derivatives**

We have implemented the functions in the code.

**# TODO: Implement the predict function for applying the trained model on the test dataset.**

The predict function is also implemented in Assignment2ii.p

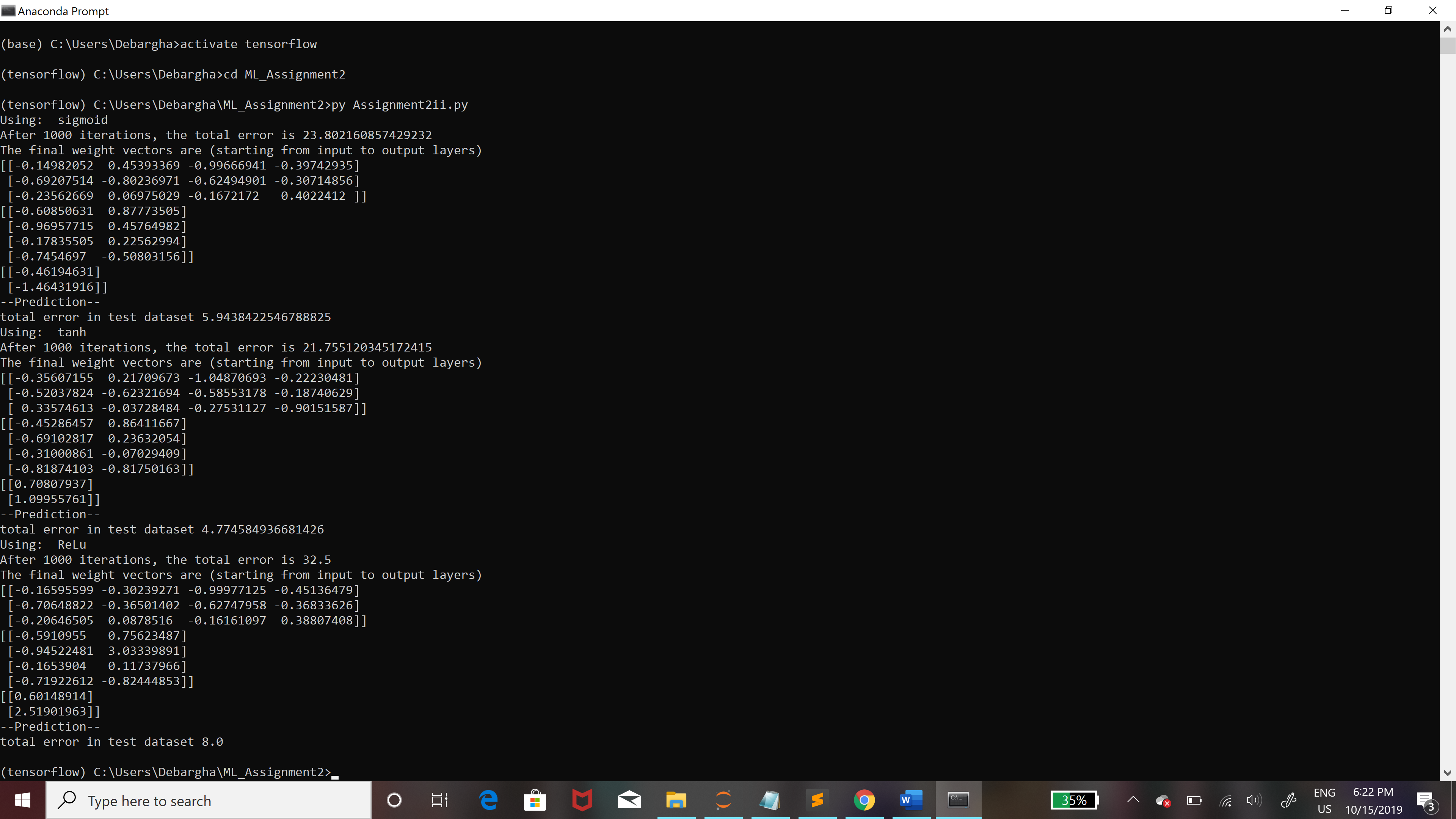
**• Output for your dataset summarized in a tabular format for different**

**combination of parameters**

Summary: We observed that Sigmoid function fared better than Tanh and ReLu in all the cases. The minimum train and test errors and their corresponding parameters have been highlighted. Sigmoid function works on binary classification. Since our output had binary classification it worked best. produced a probability output in the range of 0 to 1 that can easily and automatically be converted to crisp class values.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **LEARNING RATE** | **SIGMOID** | | **TANH** | | **RELU** | |
| Train Error | Test Error | Train Error | Test Error | Train Error | Test Error |
| **0.05** | 20.6165 | 5.3715 | 89.4996 | 22.9999 | 32.5 | 8.0 |
| **0.1** | 19.9634 | 5.3494 | 89.4999 | 22.9999 | 32.5 | 8.0 |
| **0.01** | 23.6214 | 5.8932 | 20.7404 | 5.4334 | 32.5 | 8.0 |
| **0.001** | 23.8056 | 5.9342 | 21.4057 | 5.0734 | 32.5 | 7.9557 |
| **0.0001** | 30.6916 | 7.7959 | 23.2404 | 5.8701 | 23.7284 | 6.9336 |
| **0.00001** | 41.2458 | 10.5574 | 25.6776 | 6.6297 | 29.9477 | 7.9529 |

Screenshot of Output with learning rate 0.001



I have made the following changes in the code:

1. We have added activation as a parameter in the following functions.

* Train, predict, forward\_pass, backward\_pass

1. In preprocess function we have added t\_o\_t which indicates if the data is test or train and df is the dataframe of the entire dataset.
2. We have added the corresponding code for tanh and relu functions.
3. The predict also takes the dataframe of the total dataset as parameter.