INTELLIGENT SYSTEMS

A coursework completed as part of the requirement for

SUBJECT NAME: **INTELLIGENT SYSTEMS**
SUBJECT CODE: **CC306**
LECTURER: **DR. KYAW KYAW HTIKE**

Entitled

ASSIGNMENT TITLE: **ASSIGNMENT 1**

Submitted on

DATE OF SUBMISSION: **28th MARCH 2018**

Produced by

Name	Student ID	Course
LOUIS JULIENDO	**1001540916**	**BSc (Hons) Computing**
JOHN LOH ERN-RONG	**1001439193**	**BSc (Hons) Computing**
REGINE LIM	**1001541495**	**BSc (Hons) Computing**
FAUSTIAN TRIFENISON GUNARTO	**1001645612**	**BSc (Hons) Computing**

1.0 INTRODUCTION

The term “Artificial Intelligence” is considered one of the most famous words that are brought up by the community today. It is also believed that the future of computing will revolve around this aspect for AI, Artificial Intelligence for short. This concept was born for more automation that allow machines to carry out tasks automatically humans. Under the branches of AI include machine learning, which is a current application of AI. Machine Learning refers to the idea that machines are given access to certain data and let the machine use the data and learn for themselves.

On the other hand, looking deeper into Machine Learning, there are many machine learning algorithms for computers to perform on. One of the most commonly applied machine algorithms is the Artificial Neural Networks (ANN).

The function of an artificial neural network is that it can classify data just by recognizing its pattern. For example, an application of this algorithm is to expect when the upcoming event such as a storm will occur. This algorithm cannot be programmed or configured in an intended way. The name means as it says, the neural network acts like the typical human brain’s neural network even though it is artificial. It needs to learn (or trained) on how to accomplish the task.

Dating back to 1957, Frank Rosenblatt, invented an algorithm for artificial neural network that is called Perceptron. Perceptron is an algorithm for supervised learning of binary classifier. It is a type of linear classifier that makes its predictions based on a linear predictor function combining a set of weights with the feature vector.

The component of perceptron consists of input, weight, weight summation, activation functions, and output. Input is the all the perceptron features. Weight are the values that are computed over the time of training model. Weighted Summation is the sum of value that is calculated after the multiplication of each weight associated each feature value. Activation function is for making neural network non-linear. And finally, the weighted summation is passed to the step/activation function and whatever value calculated after the computation is the predicted output.

2.0 Background

The system that we have developed is a Perceptron Machine Learning algorithm system. The aim of this system is to calculate and find out the best and final accuracy of a particular data set. The system that will be implemented applies the perceptron machine learning algorithm to the given data sets from LIBSVM. The overall workaround of the system works by splitting the datasets into 3 different sets, that are training, validation, and test sets. This is to ensure the validity of the final accuracy acquired in the end. Java is used as the main and only programming language in the development of this system.

Some of the other functions of this system include:

i) To train, validate and train the given data sets. A total of 11 data sets are given and assigned to our team from LIBSVM. Each data set has to be divided into 3 major sections, which are the training set, validation set and the testing set.

ii) To randomize the weights for the training, validating and testing of data. This system will randomize the weights based on the number of data available in each data set.

iii) To evaluate and find out the best final accuracy of the data set. After the training, validating and the testing of the data set is done, the system will be able to evaluate and output the best accuracy based on the best weights at a finished epoch.

iv) To allow the end user to choose their desired data set. Users can choose the data set that they want for the evaluation of data set by keying in in the console using the Scanner input function in Java.

.

3.0 METHODOLOGY

3.1 Methods

The following are the methods we have constructed in our source code which will be explained one by one:

Our system contains only one class named "Perceptron". This Perceptron class contains 12 methods that are **train()**, **validate()**, **test()**, **parseFile()**, **randomizeWeights()**, **updateWeights()**, **signum()**, **storeWeights()**, **storeAccuracy()**, **findIndexofLargestNum()**, **writeToFile()** and **openFile()**.

The 1st method **train()** is used to train the training dataset which has 60% of total dataset. It will train the dataset in a number of epoch specified by the user. The main goal of this method is to get the prediction by applying sign function to the dot product. After prediction has been computed, it will check whether the prediction is equals to the actual output or not. If it is equals, then increment **counter** variable by 1 in order to count the correct number of steps in an epoch. But if the prediction is not equals to the actual output, then it will update the weights. At the end, it will compute the training accuracy and show it to the output. The weights and training accuracy for every epoch will be stored in an array structure which will be used later to provide the training accuracy which will correspond to the highest validation accuracy. Finally, it will call the **validate()** method after the 1st epoch of training has been completed. The training will lasts until the maximum epoch has been reached.

The 2nd method **validate()** is used to validate the validation dataset which has 30% of total dataset. The main goal of this method is to get the prediction by applying sign function to the dot product. After prediction has been computed, it will check whether the prediction is equals to actual output or not. If it is equals, then the **correct** variable will be incremented by 1 to keep track of the number of correct steps. At the end, it will compute the validation accuracy and show it to the output. The validation accuracy will be stored in a 2-D array which will be used later to find the largest validation accuracy.

The 3rd method **test()** is used to test the test dataset which has 10% of total dataset. The main goal of this method is to get the prediction by applying sign function to the dot product. After prediction has been computed, it will check whether the prediction is equals to actual output or not. If it is equals, then the **correct** variable will be incremented by 1 to keep track of the number of correct steps. At the end, it will compute the test accuracy and show it to the output. At the end of the testing process, all the accuracy for training, validating, and testing will be shown in the output.

The 4th method **parseFile()** is used to parse file that contains the datasets. It used a **BufferedReader** to read the characters line by line. As long as the line still has data to be read, it will continue to split the space and store the splitted characters into an array called **inputPairs**. The inputPairs[0] is always the output for each data. The inputPairs[1] until inputPairs[inputPairs.length] are the inputs. It splits the ':' from the inputPairs and take the first index which is 0 to get the components. Next, it splits the ':' from the inputPairs and take the second index which is 1 to get the data/value of the input vectors. We will also specify the first input which is input[0] as 0 since it is a pre-determined value. Finally, we add the output and input accordingly. Next, we will read line by line and repeat the same process.

The 5th method **randomizeWeights()** is used to randomize the weights which range is set previously by the user. In this case, we set the minimum weights to -10 and maximum weights to 10. The method use a built-in class called **Random** to generate a random number from -10 to 10 and assign the number to the weights.

The 6th method **updateWeights()** is used to update the weights whenever the output predicted is not the same with the calculated output. In this case, the weight is updated by adding the initial with the multiplied predicted output by the input data.

The 7th method **sign()** is for calculating the sign value at the end of every step. In this context, if the answer is a positive value or more than zero, the value returned will be 1. Otherwise, it will return -1.

The 8th method **storeWeights()** is used to store the weights for every particular epoch. The allWeights is a two-dimensional array that takes in the number of epoch as its row and

the index as its column. The purpose of this array is to get the accuracy at the end of every epoch while storing every weight.

The 9th method **storeAccuracy()** is used to store the accuracy of the training and validating data. The purpose is to store the training accuracy and validation accuracy to a 2-D array called **train_and_eval_accuracy**.

The 10th method **findIndexofLargestNum()** is used to take in a two-dimensional array as an argument to find the highest accuracy that corresponds to the index. This purpose of this method is so that we can find the largest accuracy.

The 11th method **writeToFile()** is used to write to a particular file. In this case, BufferedWriter is used to write the StringBuilder object into a file (text file). This is so that the steps of all the computation will be printed properly in a text file for better viewing.

The last method **openFile()** is used at the end of the program where it prompts the user whether if the user wants the program to open output file that was created earlier with writeToFile() method. If the user presses “y”, then the output file will be automatically be opened right before the program ends.

3.2 Pseudocode

For Training

```
BEGIN
FOR each epoch
        SET correct TO 0
        FOR each training set
            INITIALIZE dot product
            FOR each weights
                ADD AND ASSIGN dot product TO weights MULTIPLY
BY  each inputs
                SET AND APPLY sign function on dot product AS prediction
                IF actual output is 0 THEN
                        SET actual output TO -1
                END IF
                IF prediction is not equals to actual output THEN
                        UPDATE weights
                ELSE
                        INCREMENT correct BY 1
                    END IF
            INITIALIZE total steps TO number of training steps
            CALCULATE training accuracy AS correct steps DIVIDE BY total
steps
            STORE weights of each epoch
            STORE training accuracy of each epoch
            CALL validate
END
```

For validating

BEGIN

FOR each epoch

SET correct **TO** 0

FOR each validation set

INITIALIZE dot product

FOR each weights

ADD AND ASSIGN dot product **TO** weights **MULTIPLY BY** each inputs

SET AND APPLY sign function on dot product AS prediction

IF prediction is equals to actual output **THEN**

INCREMENT correct **BY** 1

END IF

INITIALIZE total steps **TO** number of validation steps

CALCULATE validation accuracy **AS** correct steps **DIVIDE BY** total steps

STORE validation accuracy of each epoch

END

For testing

BEGIN

FOR each epoch

SET correct **TO** 0

FOR each testing set

INITIALIZE dot product

FOR each weights

ADD AND ASSIGN dot product **TO** weights **MULTIPLY BY** each inputs

SET AND APPLY sign function on dot product **AS** prediction

IF prediction is equals to actual output **THEN**

INCREMENT correct **BY** 1

END IF

INITIALIZE total steps **TO** number of validation steps

CALCULATE testing accuracy **AS** correct steps **DIVIDE BY** total steps

STORE testing accuracy of each epoch

END

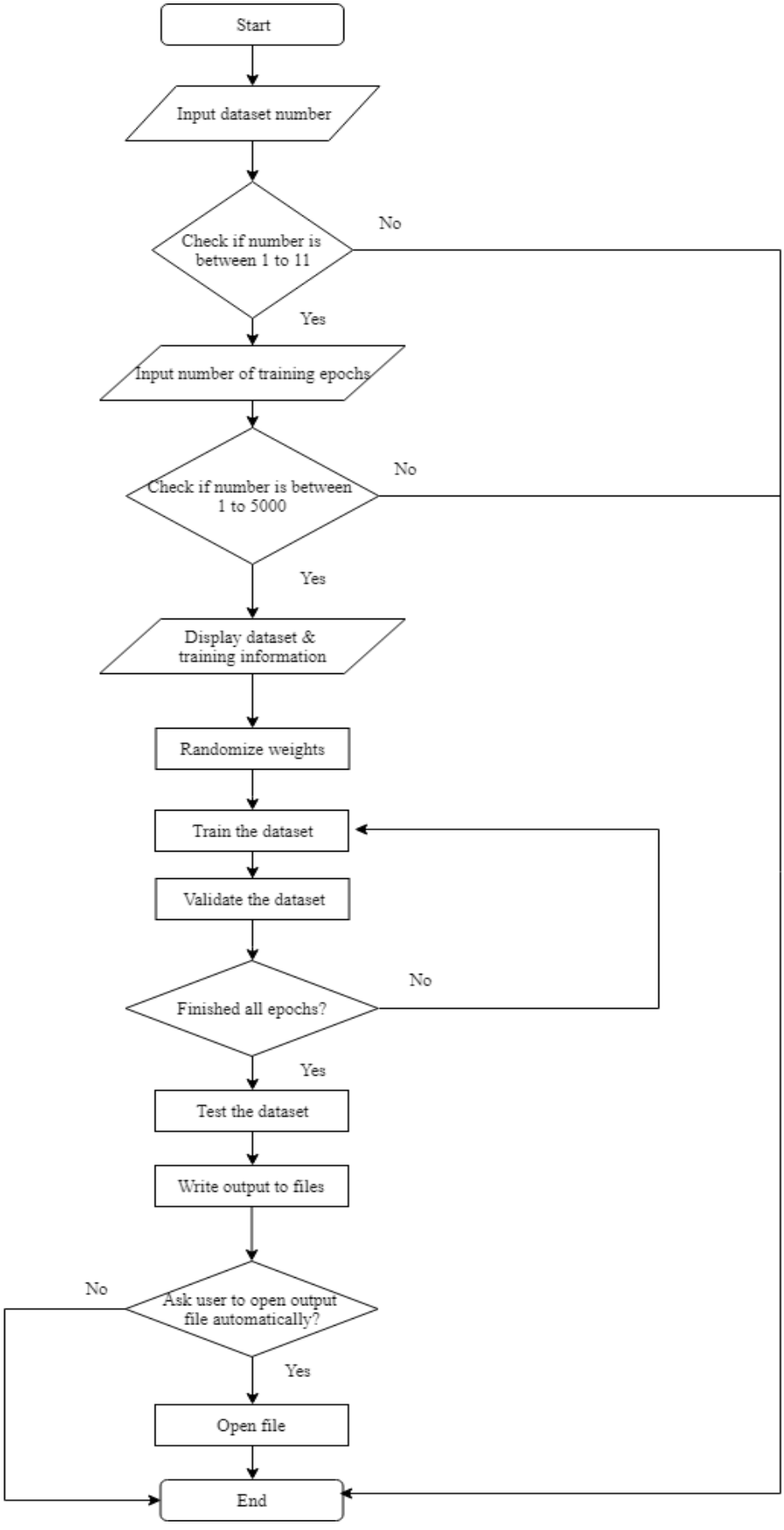
3.3 Error validation

This system has two error validation (exception handling) to help catch errors done by the user when using the system. As this system does not have a GUI, user inputs are only done at the console of an IDE, which in our case, NetBeans IDE.

i) Validate if users enter a number outside the range of 1 to 11. At the very start of the system, users are ONLY allowed to enter numbers from 1 to 11 as they are only 11 data sets for them to choose from. If users enter numbers outside of this range, the system will exit. This validation is done by the if-else block.

ii) Validate if users enter an epoch outside the range of 1 to 5000. After determining the dataset, the system is programmed to only accept a maximum of 5000 epochs. If the user enters more than 5000 epochs or less than 1 epoch, the system will exit. This validation is also done by the if-else block.

3.4 Flowchart



4.0 RESULTS

4.1 Data Set Analysis

The following shows the best accuracy that we have found through the training, validating and testing of data for each and every data set.

i) Duke Breast Cancer

Training accuracy: 88.46%

Validation accuracy: 100.00%

Testing accuracy: 100.00%

It was found that the validation accuracy reached 100.00% after training and validating at **Epoch 4/1000**.

ii) Heart

Training accuracy: 77.16%

Validation accuracy: 85.19%

Testing accuracy: 77.78%

It was found that the test accuracy reached 77.78% after training and validating at **Epoch 1932/2000.**

iii) Ionosphere

Training accuracy: 88.10%

Validation accuracy: 95.24%

Testing accuracy: 94.29%

It was found that the test accuracy reached 94.29% after training and validating at **Epoch 1754/2000.**

iv) German Nummer

Training accuracy: 70.83%

Validation accuracy: 79.33%

Testing accuracy: 84.00%

It was found that the test accuracy reached 84.00% after training and validating at **Epoch 419/1000.**

v) A1A

Training accuracy: 78.82%

Validation accuracy: 85.03%

Testing accuracy: 83.12%

It was found that the test accuracy reached 83.12% after training and validating at **Epoch 406/1000.**

vi) SVMGuide1

Training accuracy: 99.95%

Validation accuracy: 100.00%

Testing accuracy: 100.00%

It was found that the test accuracy reached 100.00% after training and validating at **Epoch 1/1000.**

vii) W3A

Training accuracy: 99.59%

Validation accuracy: 95.79%

Testing accuracy: 87.37%

It was found that the test accuracy reached 86.35% at **Epoch 1000/1000**

viii) W4A

Training accuracy: 99.37%

Validation accuracy: 95.02%

Testing accuracy: 85.05%

It was found that the test accuracy reached 85.05% at **Epoch 996/1000**

4.2 Screenshots of System

The following diagrams are screenshots of the system that is developed.

i) The first part of the program when first run

```
List of data sets:
1) Duke Breast Cancer
2) Heart
3) Ionosphere
4) German Nummer
5) A1A
6) SVMGuide1
7) W3A
8) W4A
9) RCV1 Train
10) Real Sim
11) Criteo

Enter the number of your desired dataset:
```

ii) Sample output in console after determining the dataset and number of epochs

```
Enter the number of your desired dataset: 1
Enter the number of epochs: 200
Dataset: 1) Duke Breast Cancer
Data: 44
Features: 7130
Training data: 26
Evaluation data: 13
Test data: 4
Output has been written to file successfully!
Do you want to open the output file? Press 'y' or 'n'
```

iii) Snippet of the sample output file created

```
---------- Training and Validating ---------- ↓
Epoch 200↓
Training accuracy: 100.00%↓
Validation accuracy: 100.00%↓
Weights: [-2.527196471260549, 3.5388633763458075, 9.804387380153392, -8.249935828
↓
(Training and Validating process have been completed!) ↓
↓
---------- Validation Results ---------- ↓
Best validation accuracy: 100.00%↓
Found in epoch: 1↓
Final weights: [-3.527196471260549, 3.7370703763458075, 10.256259380153391, -7.33
↓
---------- Testing ---------- ↓
Test accuracy: 100.00%↓
↓
---------- Summary ---------- ↓
Training accuracy: 65.38%↓
Validation accuracy: 100.00%↓
Testing accuracy: 100.00%↓
←
```

iv) Error validation if users enters less than 1 or more than 11 for the number of dataset

```
Enter the number of your desired dataset: 12
Please enter a value between 1 to 11!
```

v) Error validation if users enters less than 1 or more than 5000 for the number of epochs

```
Enter the number of your desired dataset: 1
Enter the number of epochs: 9000
Please enter epoch between 1 to 5000!
```

5.0 DISCUSSION

There were several challenges that all of us faced when working on the project from the very start to the very end. The very first challenge that we faced was understanding of requirements. We were not sure as to how we could get the best accuracy of the training set. Although we know we need to find the plateau of the accuracy and the number of epochs, we could not find out what is the best way to implement this. Because of this, brainstorming sessions were organised very regularly and we took quite some time understanding the requirements.

Another challenge was the parsing and reading of files which has caused a Java heap space error due to insufficient memory. Datasets that were used to be read to get the accuracy could not be read because of the large file size that goes up to 26GB per text file. Although the Java Virtual Machine (JVM) can be tweaked to increase the heap memory, the text file is still considered too large for reading (as we need to read them to split and perform the training, validating and testing of data).

Furthermore, we were not sure what was the right way of getting the correct accuracy of an epoch. After much discussion, we have decided to total up the correct number of steps in a single epoch and divide them over the total steps and multiply by 100% to get the percentage of accuracy.

5.1 Strengths of the system

i) Flexibility for users to select the dataset

The greatest strength for our system is the inclusion of Scanner function. In the system, users are able to enter the dataset that they wish to access. Users will be shown a list of dataset which is available with the corresponding numbers to the data. Then, they can enter the number for the dataset to train, validate and test for that particular dataset.

ii) Separation of logics into multiple methods

There are certain operations that are being broken down into several meaningful methods. Some of the methods include writeToFile, openFile, parseFile, train, validate and test. The codes are displayed in a neatly form, which makes it easy to read. By having these methods, confusions on reading and understanding the codes can be avoided, and thus makes the codes easy to understand.

iii) Automatically open files for output

Another strength that exists in our program is the function of opening the files for output automatically. After the users have entered all the required data, the system will start running the program. Then, the output will be written in a .txt file that stores in the user's computers, and it will be opened up automatically when the program finishes the run. The software for opening the .txt file depends on the default software being set in the user's computer.

5.2 Limitations and future improvements of the system

The system being developed has achieved its main goal as an overall. However, there are still some limitations that can be further improved in the future. This section will discuss about the limitations and future improvements on the system.

i) Unable to access to certain dataset

Among 11 datasets, there are 3 dataset that are inaccessible. The three datasets are rcv1 train, real-sim and criteo. The reason is due to the size of the dataset being too big. The details of the three dataset can be shown below:

a. rcv1 train

Number of data: 20,242

Size: 35 MB

b. real-sim

Number of data: 72,309

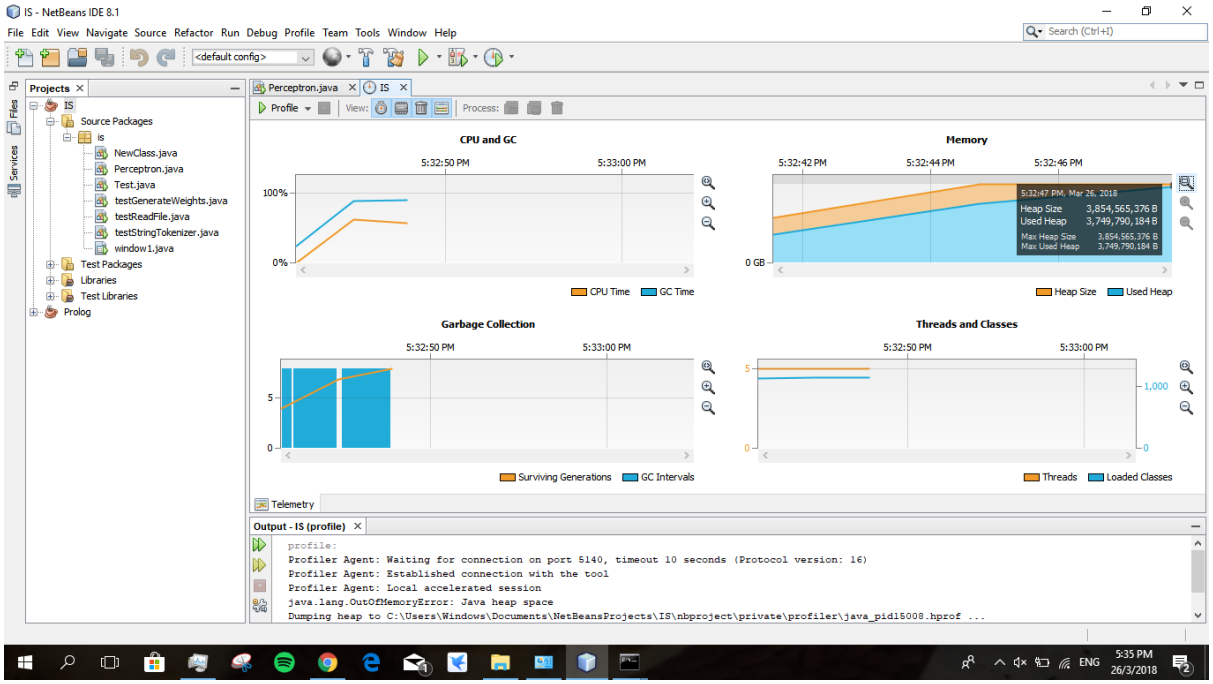
Size: 88 MB

c. criteo

Number of data: 45,840,617

Size: 26 GB

As we can see from the details, three of these datasets contain large number of data, and also having large sizes. Due to this reason, it consumes more memory compared to the other 8 datasets which have smaller sizes and number of data. The features that are being allocated in the array also restricted the execution of the dataset.



This figure shows the profiling processes where the program takes too much memory in order to run the program. 4GB of RAM has been allocated to the maximum java heap space memory so that it could allocate more memory for further processing but the program consumes up to 3.8 GB of RAM in a few seconds which indicate that the data and feature passed to the array are too big and therefore our program can't handle it.

ii) Manually enter for epoch

The next limitation is users need to manually enter the numbers of epoch to execute the dataset. By default, the number of epoch should be generated by the program. However, in this case, users are only able to execute the program after they have entered the number of epoch for the training of the dataset.

To further improve our current system, several recommendations are being discussed. We wish to include a GUI that can show the trend and the graph for validation process in the future. Having the stated features can help users to have a better vision on what is going on throughout the whole process. The other improvement that we wish to have is having a better presentation of results. In the current system, the presentation of the result is too plain, and the user might accidentally misreads or misses any important information. This improvement can help the users to have a clearer insight on the outputs of the program.

6.0 TEAMWORK

Our team consists of 4 people which are as below:

Group Leader:

Louis Juliendo

Group Members:

John Loh Ern-Rong, Regine Lim, Faustian Trifenison Gunarto

There are 2 types of meetings that were conducted which were face-to-face discussions and online discussions. The workload is divided equally through all of the team members to ensure that everyone contributes to the project. Whenever a task is complete, it will be checked by Louis to approve and ensure that there are no errors so that other tasks can be proceeded.

6.1 Planning phase

Before the start of the project, a face-to-face discussion is conducted to discuss the ideas and features that will be added and also to divide tasks for all team members. During this planning phase, we have made many plans for a face-to-face meeting schedules that are set on Sundays and Mondays, from 12PM until 6PM. Next, we planned on the division of tasks for team members and when the tasks can be completed. Certain platforms are used online during the development of this project such as:

i) Google Drive: to upload project codes and report and to update team members with the latest changes

ii) Facebook : to communicate with team members

iii) Video collaboration tool (appear.in): to explain our progress to the team

The project officially starts on the **5th of March 2018**.

6.2 Design phase

This phase refers to the final output which we wish to achieve to reach the objective of our program. John and Regine were tasked in this phase to check and discuss what kind of output would be best seen by the user after getting the best and final accuracy of the data set. Then, the idea is passed to Louis and he will then check if the idea follows the specified requirements.

6.3 Implementation phase

The source code is divided to all group members by the methods used in our program which are shown below:

Louis: **train()**, **storeWeights()** and **storeAccuracy()**
John: **validate()**, **randomWeights()** and **openFile()**
Regine: **test()**, **updateWeights()** and **parseFile()**
Faustian: **writeToFile()**, **sign()** and **findIndexofLargestNum()**

Since Louis is the group leader, all group members have to review their codes with him to check if the code works well or not. Once all codes are approved, all team members have worked together to integrate all the code with standardized variable names and compiled in one single Java class.

6.4 Documentation phase

In the documentation phase, the report is being written and produced by the end of the project. All team members were assigned parts to work on the documentation which later will be checked by John. Firstly, the introduction and background were written by Faustian and he was supervised by John. Next, Louis wrote the methodology which includes the pseudocodes, explanation of methods and the error validation. John, Louis and Regine were responsible for the designing of the flowchart and the results that contain a brief description and screenshot of our system features and error validation. Faustian and Regine were also assigned to write for the discussion while John and Faustian were tasked to conclude the documentation of this project.

Louis was also responsible to write for the teamwork as he was the leader of the team who knew the task division among all the team members. After the report has been done, John will proofread and check for any grammatical mistakes and finalize them. Finally, all team members will double-check the report and point out if there are any mistakes that should be changed in order to produce an excellent report.

7.0 CONCLUSION

In conclusion, while making this project, putting all the technical skills aside, we learned that time management is a critical aspect upon making this project. This is due to the different time schedules we all have and that each of us are given a certain period of time to finish our work. This has made us more aware on how we organize our time. Through this project, we have also learned how to compile a good report as well, to make everything not just good to read, but also understandable by the readers (as this assignment is focused in a lot of analysis of data sets).

We have learned that the human brain and computer machine learning work quite alike. While the brain has a neural network, computers have an artificial neural network. Even though that its artificial, but it works very similar to the real one. The artificial neural network has shown us its importance, which brought much significance to us when we were developing this project.

As this is a very challenging project for us. Since dealing with Machine Learning algorithms were a new thing to all of us, it was indeed quite difficult for us to determine certain values in our code. As a team, we have tried many ways to solve the problems one at a time and through sheer teamwork and effort, we managed to overcome most problems even though not all. While we might have benefited a lot from this project, we believe there are still a lot more room for improvement in our work. Also, through doing this project, we are now more confident in our coding skills. Therefore, this will make us feel more certain that our skills will be to lead to greater levels of our future careers.