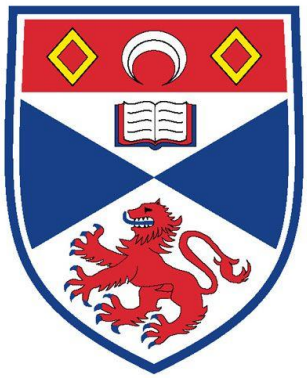


University
of
St Andrews

Fourth Practical-CS5011

Artificial Neural Networks - A Ticketing Routing Agent

Submitted by:

180030908

Submission Date: December 20, 2019

1. Introduction:

In this project, I have developed a ticket routing agent using technique of neural network. Some data set was given in tickets.csv file. The main task was to map the request of the user with a specific response team.

In this report, first I will introduce the basic terminologies in neural networks. I will discuss then the design and implementation details of the project that will include the design decisions I took and the steps I covered for implementation of different parts. I will use screenshots to demonstrate that as well. Lastly, I will provide an evaluation of my algorithm and techniques along with different test cases to support the validity of my work.

In this Assignment, I have implemented the complete sections of Basic and Intermediate agent. At the end of the report, I will also few limitations of any part of this implementation as well.

Neural Networks:

Artificial Neural Network were introduced in late 90's. They were designed for classification, clustering, pattern recognition and future predictions. ANNs can generalize data after learning from given input and their relationship, hence predicting unseen data and situations like human brain. [1]

ANNs has three interconnected layers. On the first layer input data is given through neurons which sends it to second hidden layer. From there, output result is passed to the third/output layer. Figure 1 shows architectural structure of the ANN.

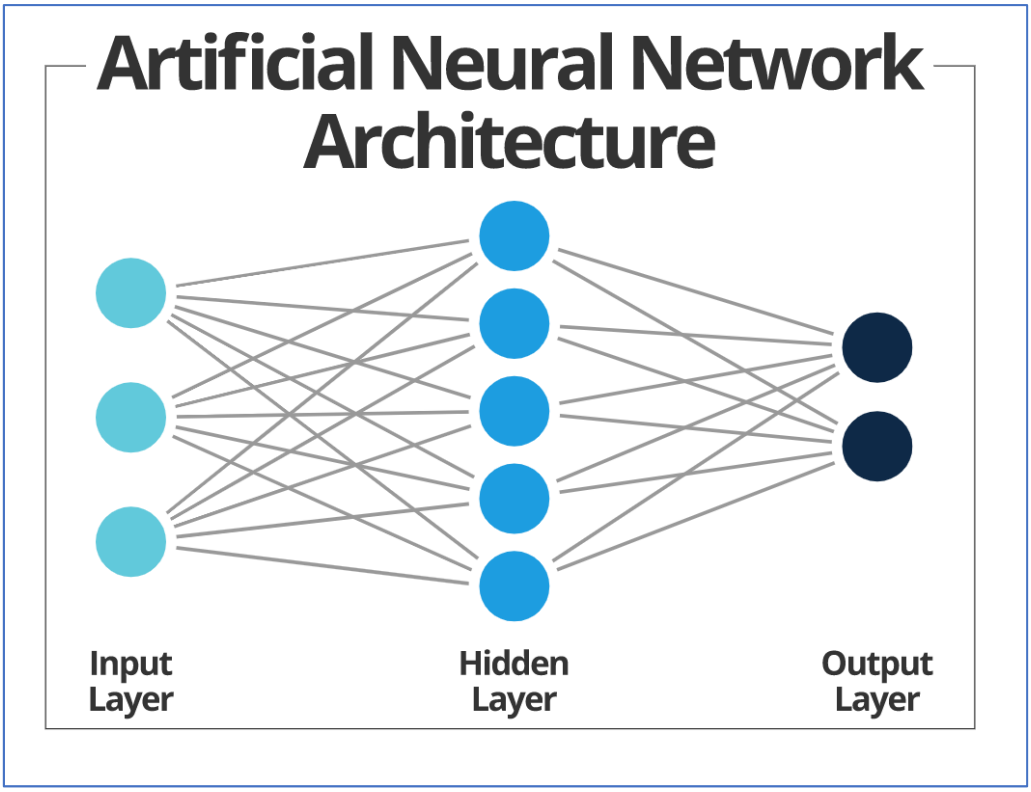
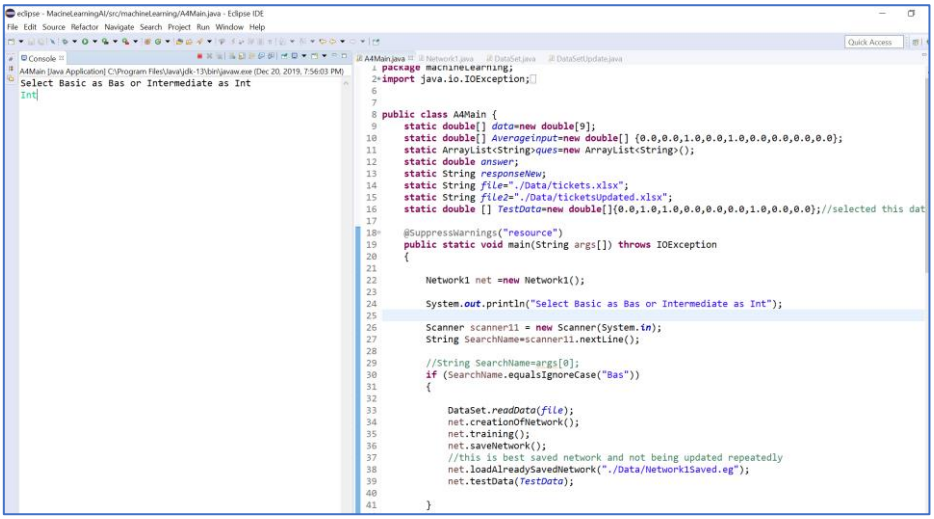


Figure 1: ANN Structure [2]

Instructions to Run the code:

I am unable to run the code from command line as it is giving me class path missing error. Hence this program needs to run through an IDE (intelliJ or Eclipse) from its console. After running the program from console, 'Bas' or 'Int' option should be given to the program to start the execution.



In intermediate part, user will be giving answers to the agent in 0 (for No) and 1 (for Yes) format to different questions that agent will ask. User can choose and write the name of the response team in the last question if he/she is not satisfied with the assigned response team.

2. Design and Implementation

Before starting development, I designed my solution with PEAS model.

PEAS Model:

Problem: Predict Response team for the user on unseen data after taking input data from the user based on the training that has been given to the agent.

Agent: Software Program

Performance: This determines the level of correctness of the prediction done by the agent. That is agent is determining correctly that

Environment: In-house Help Desk System for IT Services

Actuators: These are the actions performed by the agent so here this include giving predictions to users, asking questions from the users, getting trained on network

Sensors: This is something perceived by the agent and on this basis, different actions are performed. Here it is receiving input from user.

Implementation Strategy:

The first step in the implementation process was the reading of the data from the csv file. For this purpose, I used APACHI POI Library. I converted the given file into .xlsx file for better use within the library.

1. APACHE POI

It is an open source API that make use of Java program for modifying, reading, creating and displaying MS Office files [4] in a memory optimized way.

For this I imported this library within my code first and then using its built-in commands I read all the data from the file and stored it into an Array List.

2. One-Hot Encoding Technique

The second step was the encoding of the data in Array List into 0 and 1 format using one-hot encoding technique so that it can be fed into the neural network.

One hot encoding is a process by which categorical variables are converted into 0,1 form that could be provided to ML algorithms to do a better job in prediction. [3] With this a binary column for each category is created and a sparse matrix or dense array is returned.

I selected one-hot encoding technique for my implementation because it was easier for me to convert the input and output values into binary for this smaller data set. If the data set would have been a bigger one, I might have used some other encoding technique like one-to-n or binary encoding.

Here for first 9 columns of the whole data set that represents different properties of the request is encoded into 0 and 1 for input array. ‘Yes’ in the data is being encoded to 1 and ‘No’ is being encoded to 0. The last column of the data represents the name of

different response teams. The value of this column is encoded into 0 and 1 and then stored in the output array. Here I made use of hash map with hard coded values of five different types of response teams. This is mapped with the output data array in a way that a value is 1 for only the particular key to which it represents, and for all other values, the encoding is done to 0 at that particular moment. Following images will give a better understanding of the complete encoding procedure.

	A	B	C	D	E	F	G	H	I	J	K
1	No	No	No	Yes	Yes	Yes	No	Yes	No	Creden	s
2	No	No	Yes	Yes	Yes	No	No	No	No	Credentials	
3	Yes	Yes	No	Yes	No	No	No	No	Yes	Credentials	
4	No	No	No	Yes	Yes	No	No	No	Yes	Credentials	
5	Yes	Yes	No	Yes	Yes	No	No	No	Yes	Credentials	
6	Yes	Yes	No	Yes	Yes	Yes	No	No	Yes	Credentials	
7	Yes	Yes	No	Yes	Yes	No	No	No	Yes	Credentials	
8	No	No	No	Yes	No	No	No	No	Yes	Credentials	

Figure 2: Data in Excel File

```
No-No-No-Yes-Yes-Yes-No-Yes-No-Credentials-
No-No-Yes-Yes-Yes-No-No-No-No-Credentials-
Yes-Yes-No-Yes-No-No-No-No-Yes-Credentials-
No-No-No-Yes-Yes-No-No-No-Yes-Credentials-
Yes-Yes-No-Yes-Yes-No-No-No-Yes-Credentials-
Yes-Yes-No-Yes-Yes-Yes-No-No-Yes-Credentials-
Yes-Yes-No-Yes-Yes-No-No-No-Yes-Credentials-
```

Figure 3: Data Read by the program through APACHE POI

```
0.0 0.0 0.0 1.0 1.0 1.0 0.0 1.0 0.0
0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0
1.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0
0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 1.0
1.0 1.0 0.0 1.0 1.0 0.0 0.0 0.0 1.0
1.0 1.0 0.0 1.0 1.0 1.0 0.0 0.0 1.0
```

Figure 4: Encoding of first 9 rows to Input Data Array

```
Current key: Emergencies
1.00.00.00.00.0
Current key: Credentials
0.01.00.00.00.0
Current key: Datawarehouse
0.00.01.00.00.0
Current key: Networking
0.00.00.01.00.0
Current key: Equipment
0.00.00.00.01.0
```

Figure 5: Encoding for Output Data Array and mapping within HashMap

3. Creation of Network with Encog Library

Since this project is implemented in Java, I used Encog library for network creation, training, saving, loading and lastly testing of the network.

The first step I did here was the creation of the network. For that I give input and output units to the network.

After that I added input, output and hidden layers in the network with Sigmoid function as the activation function. The reason for using sigmoid function here instead of step function is that it provides better continuity in the results as the step function is non-differentiable at x = 0 which means there won't be progress in gradient descent during updating the weights and backpropagation will fail [5].

The number of input layer's units in a network is mostly the number of input features of the dataset which is 9 in this case and number of output layer's units is the number of output features which is 5 in this case.

The number of hidden units ideally within input and output units, so I set number of hidden units here to 9. I decided for this number after testing number on different values and then finalized this when I got the most accurate result (More detailed discussion is shown in evaluation section).

Here I also set a value for momentum and learning rate to 0.3 and 0.1 respectively. Momentum and Learning rate are hyperparameters that are used to make the training within a network more efficient by reducing error between actual and targeted output. Momentum further helps the network come out of local minima [6]. Figure 6 shows how momentum helps in speeding up convergence to the minimum by damping oscillations.

Choosing the learning rate and momentum was challenging as if the value is very small, it makes training process very slow and a large value may result in an unstable training process, missing important data points [1].

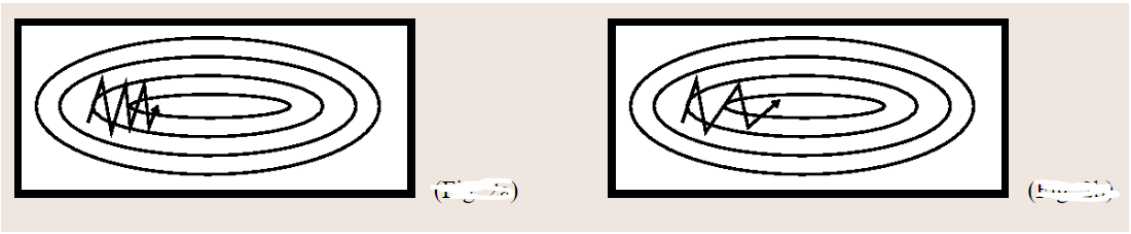


Figure 6: Without Momentum(left) and With Momentum (right) [6]

4. Training of Network with Backpropagation

Backward propagation of errors commonly known as Backpropagation. It is a way to effectively train a neural network through chain rule. It repeatedly adjusts the weights of the connections in the network in order to minimize a measure of the difference between the actual and targeted output. It aims to minimize the cost function by adjusting network's weights. These weights are by default set by the Encog library. The training is done on data until the error is reduced to less than 0.01.

5. Loading, Saving and Testing of Network

For the basic part of the assignment, I trained the system multiple times and the I saved the network with least error and least training time. The saved network file's name is 'Network1Saved.eg'. I then used this saved network for training and predicting in intermediate part of the assignment. However, in basic part, every time a new network is created and saved in Network1.eg' file.

After saving the network, I loaded the saved network for testing the network. For testing, I passed different unseen input data to the network and then obtained the name of the response team in the network and then checked it with actual output to ensure that network is predicting correctly.

6. User Input and Early Predictions

For the intermediate part, I am giving the option to the user to login a request. For that agent is asking different questions from the user and then providing the user with a response team. Here Instead of asking 9 different questions in one go, agent is asking three question in start and then agent provides user with the name of the response team. If user said that it is not correct, agent ask one more question from user and this process goes on till agent asks all questions about all 9 tags. Even after the last question, if user is not satisfied with the assigned response team, then, agent asks user to provide the name himself from the list of different response teams. Agent then add all the responses in the excel sheet and then retrain the network on this new data.

For this part, I am using 'ticketsUpdated.xlsx' file. Again, I am reading and adding rows in the file here through APACHE POI library.

For early predictions, for unseen responses of the users, I took average of yes and no for every tag and the tag with highest yes is set to 1 and set to 0 if the tag has more No values.

	Request	Incident	Web Service	Login	Wireless	Printing	Id Cards	Staff	Student
Yes	79	110	141	94	160	91	107	76	101
No	171	140	109	156	90	159	143	174	149

Request	Incident	Web Service	Login	Wireless	Printing	Id Cards	Staff	Student
0	0	1	0	1	0	0	0	0

Figure 7: Reasoning for early predictions

Code Structure:

In the code, I have four classes. One is main class for dealing with user inputs, one is the basic dataset class for reading data from excel sheets and encoding it into input and output array, third class is Network class where network is created and trained and last one is the class for adding new rows in the excel sheet.

3. Evaluation

To evaluate the system, I have used the error value and training time as bench mark. I have used different values of momentum, learning rate and hidden unit value to find the best value from each where network is providing maximum efficiency. Following table will show network performance on different values. The figures below show the results of the table as well.

Learning Rate	Momentum	Hidden Layer Units	Epochs	Error Value	Figure
0.1	0.3	9	104	0.0099	Figure 8
0.2	0.3	8	176523	0.083200	Figure 9
0.1	0.3	7	84561	0.008299	Figure 10
0.2	0.1	9	5099	0.009595	Figure 11
0.5	0.3	6	120931….	0.19999	Figure 12

The above table clearly shows that only the highlighted row was the most efficient one in terms of training time. In third we see less error but the training time was significantly high than the highlight row.

```
Epoch #77 Error:0.012915657451143515
Epoch #78 Error:0.012780207939114835
Epoch #79 Error:0.012647905074152378
Epoch #80 Error:0.01251855943773086
Epoch #81 Error:0.012391991885926122
Epoch #82 Error:0.012268032915787427
Epoch #83 Error:0.01214652217918906
Epoch #84 Error:0.012027308122687274
Epoch #85 Error:0.01191024773735638
Epoch #86 Error:0.011795206407179768
Epoch #87 Error:0.011682057848259986
Epoch #88 Error:0.011570684133722288
Epoch #89 Error:0.011460975800506494
Epoch #90 Error:0.011352832034008081
Epoch #91 Error:0.011246160924494838
Epoch #92 Error:0.011140879785211819
Epoch #93 Error:0.011036915516080317
Epoch #94 Error:0.01093420498912606
Epoch #95 Error:0.010832695422810977
Epoch #96 Error:0.010732344703249297
Epoch #97 Error:0.010633121602227937
Epoch #98 Error:0.010535005836698627
Epoch #99 Error:0.010437987913770602
Epoch #100 Error:0.010342068710829995
Epoch #101 Error:0.010247258753284989
Epoch #102 Error:0.010153577172615537
Epoch #103 Error:0.010061050353613755
Epoch #104 Error:0.009969710309219545
Trained succesfully.
Saving network
Loading network
Network1 Input = 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.0,
Actual Output = 0.046824321780077435, 0.9225106384474675, 0.0011113187808479527, 1.5404709246314975E-4, 0.08420998463581467,
The highest value in the array along with index is : 0.9225106384474675,1
Actual Output after 0 and 1 encoding = 0.0, 1.0, 0.0, 0.0, 0.0,
Your Response will be in Team: Credentials

Expected Output = 0.0, 1.0, 0.0, 0.0, 0.0,
```

Figure 8

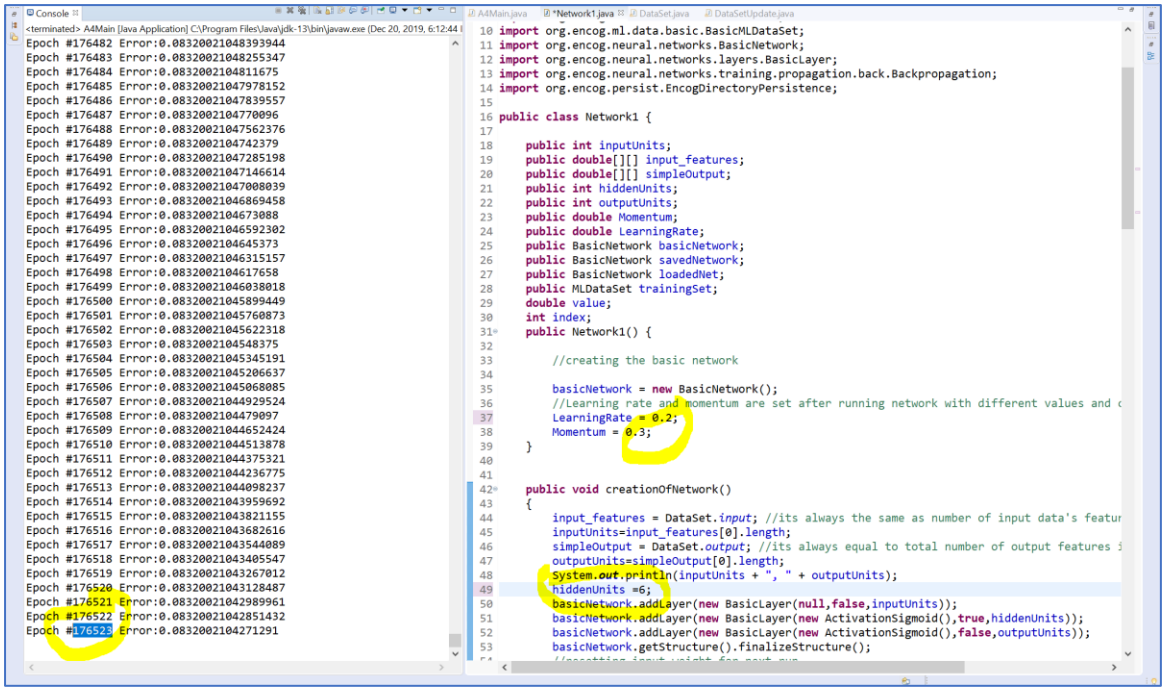


Figure 9

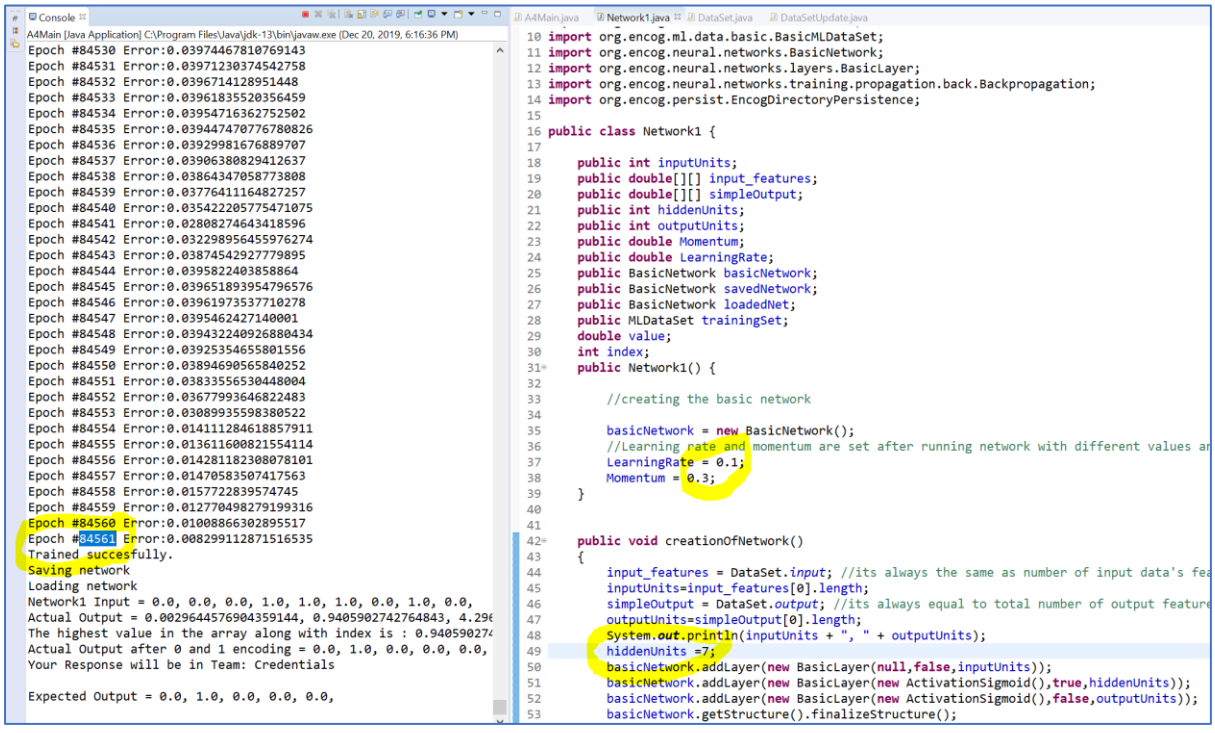


Figure 10

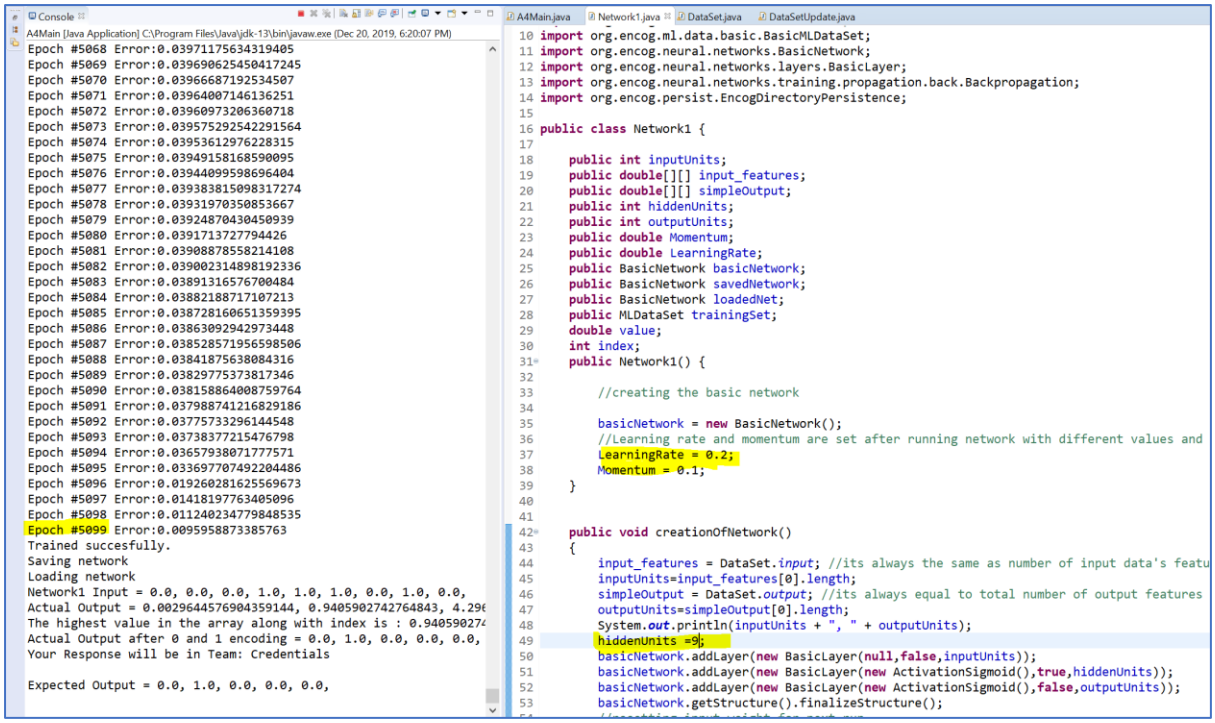


Figure 11

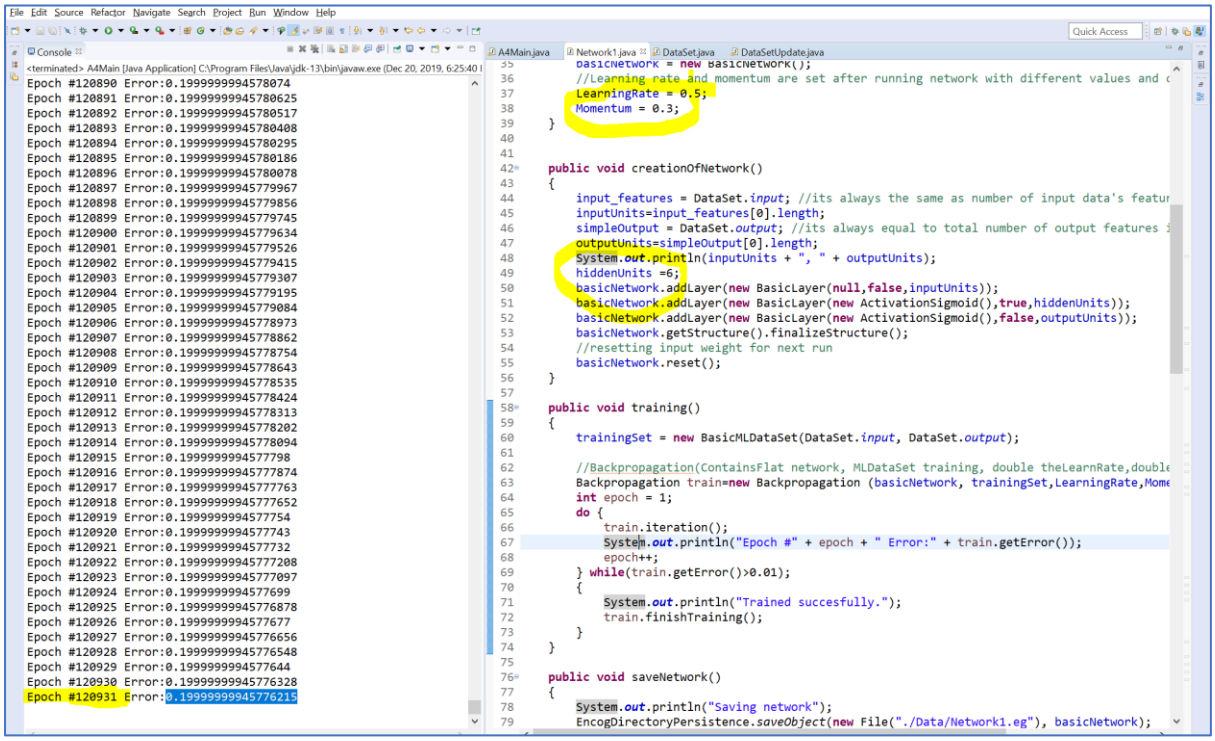
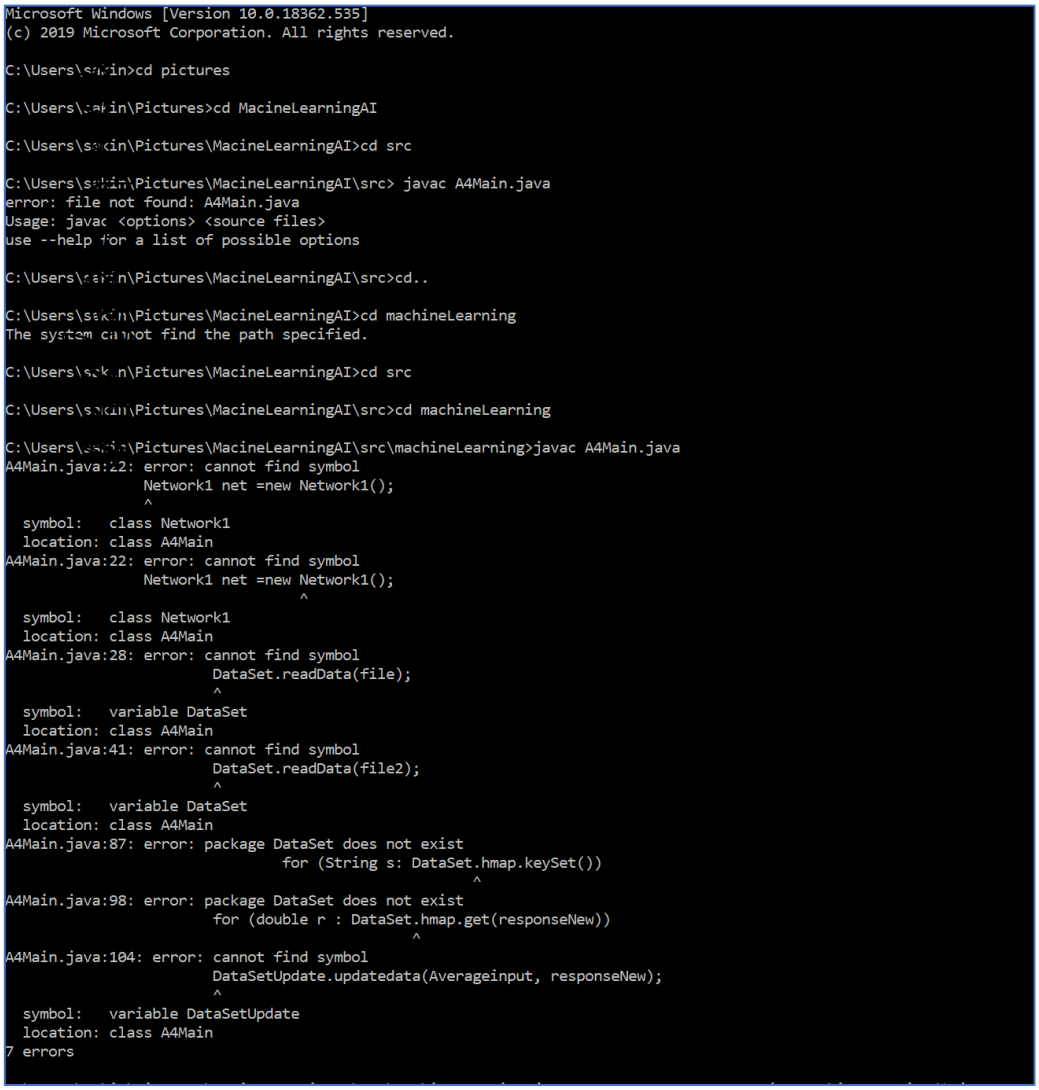


Figure 12

Program Limitations:

1. I did not use any validation data as given data set was very small but if I had a big data set, it would be an efficient approach to use the validation set for training.

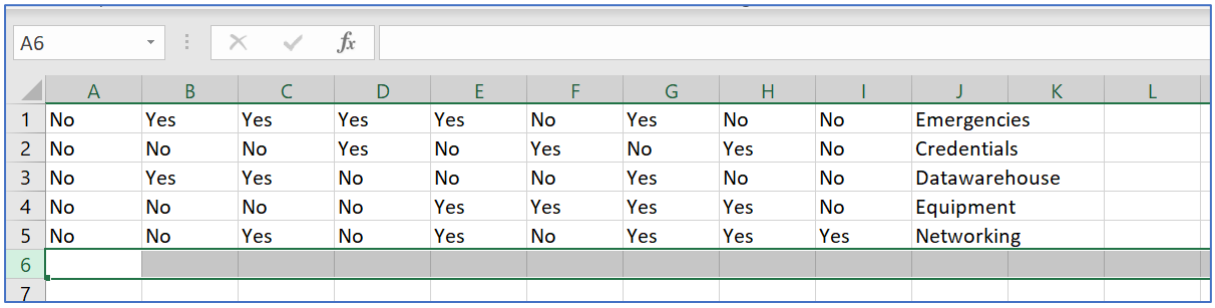
2. Currently at early guessing I am using the average of greater number of yes and no for each tag to decide the random input data that whether it should be 0 or 1, but after retraining with new data, I am not updating this decision. This could have been improved If I had more time and If I had not done this manually as shown in Figure 7.
3. User can only select from the list of present response teams and is not able to suggest new response teams.
4. The program is unable to run with command line instructions as shown below.



4. Testing

For testing purposes, I am taking an unseen data set of 5 rows in an excel sheet (I have added it in the data folder). From there I am taking one row and putting it manually into my program after encoding yes to 1 and 0 to no to evaluate the correctness of network.

Testing of Basic Part:



	A	B	C	D	E	F	G	H	I	J	K	L
1	No	Yes	Yes	Yes	Yes	No	Yes	No	No	Emergencies		
2	No	No	No	Yes	No	Yes	No	Yes	No	Credentials		
3	No	Yes	Yes	No	No	No	Yes	No	No	Datawarehouse		
4	No	No	No	No	Yes	Yes	Yes	Yes	No	Equipment		
5	No	No	Yes	No	Yes	No	Yes	Yes	Yes	Networking		
6												
7												

Figure 13: Unseen Data Set for Testing

First Test input: 0.0, 1.0,1.0,1.0,1.0,0.0,1.0,0.0,1.0

Expected Output: Emergencies

Actual output: Emergencies

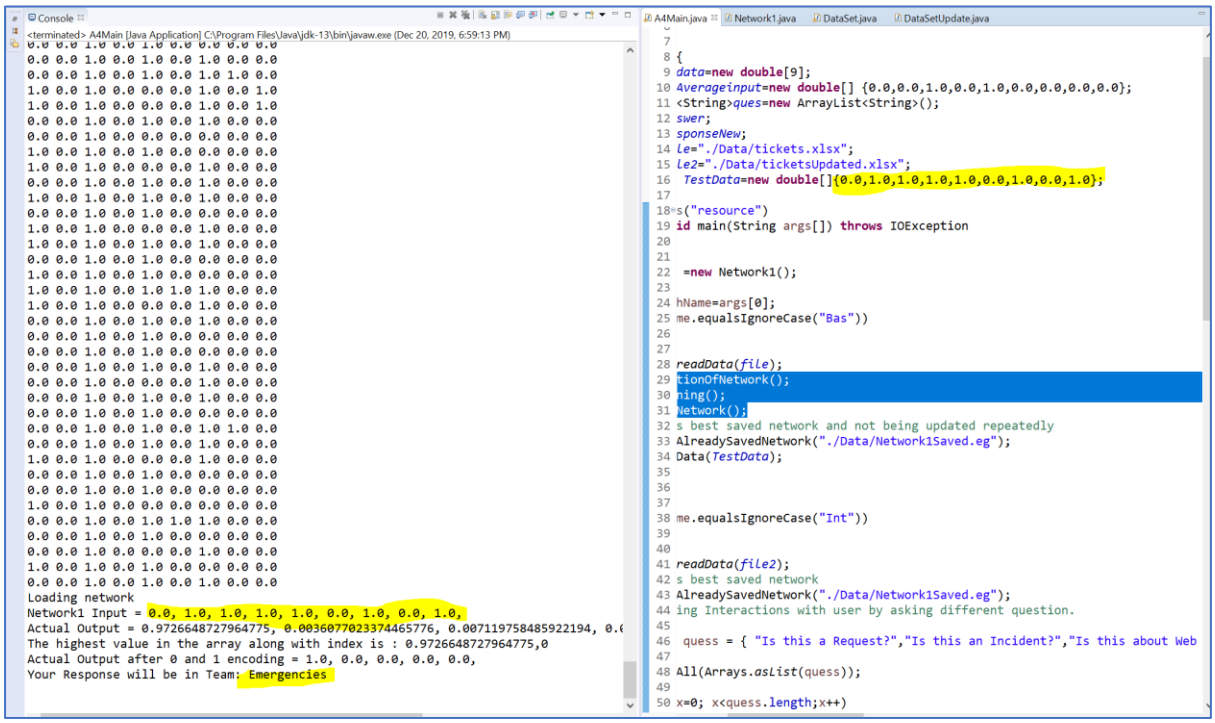


Figure 14

2nd Test input: 0.0, 0.0,0.0,1.0,0.0,1.0,0.0,1.0,0.0

Expected Output: Credentials

Actual output: Credentials

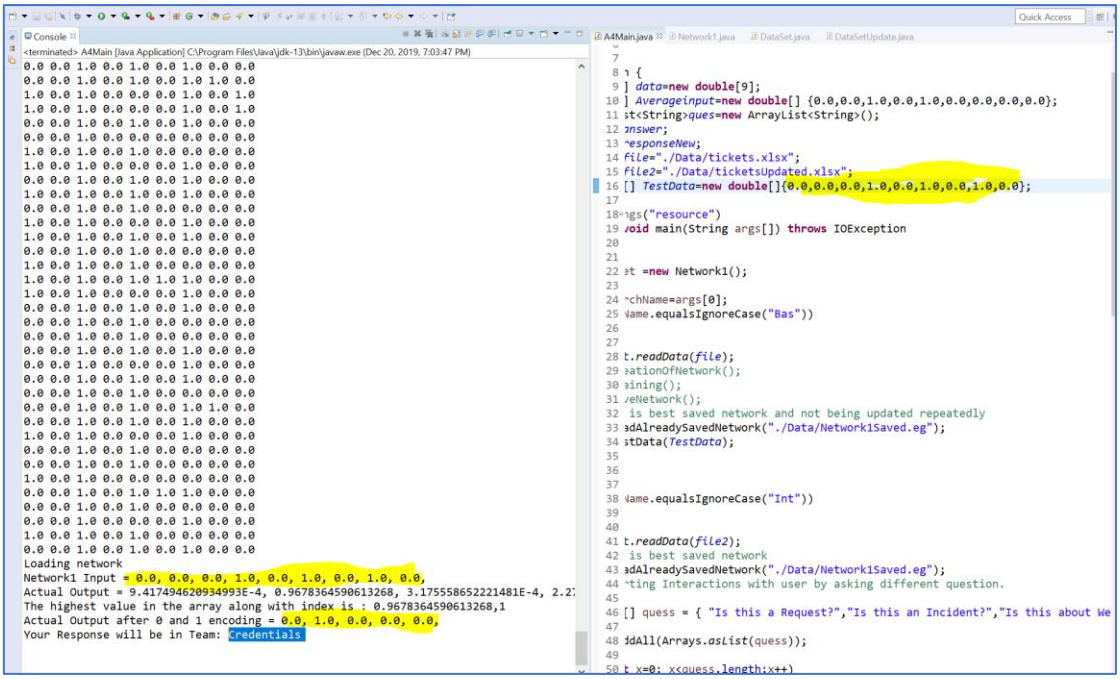


Figure 15

3rd Test input: 0.0, 1.0,1.0,0.0,0.0,0.0,1.0,0.0,0.0

Expected Output: Datawarehouse
Actual output: Datawarehouse

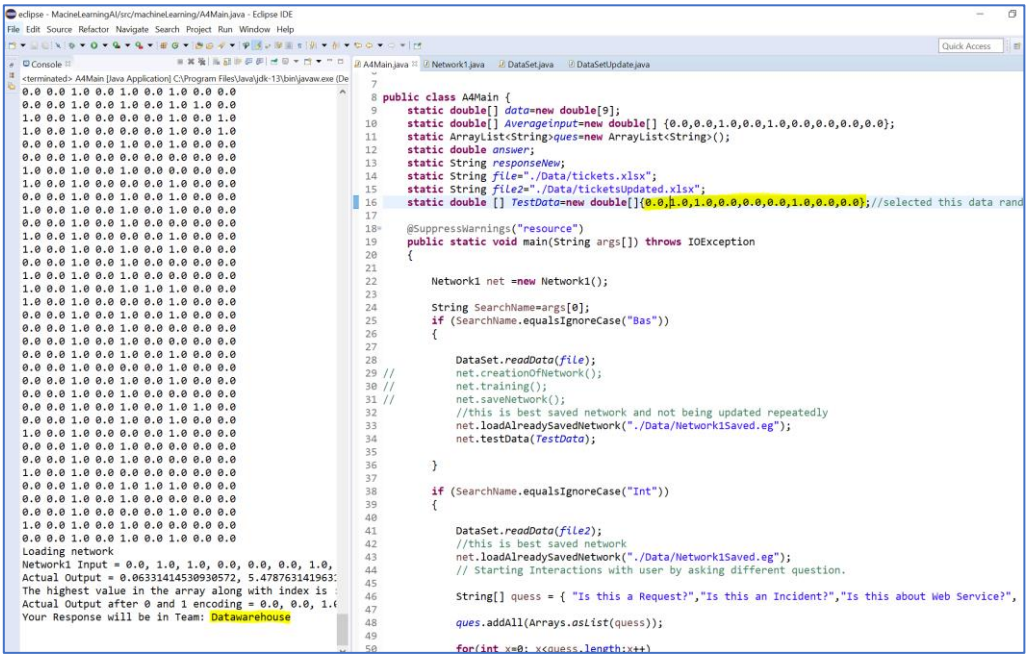
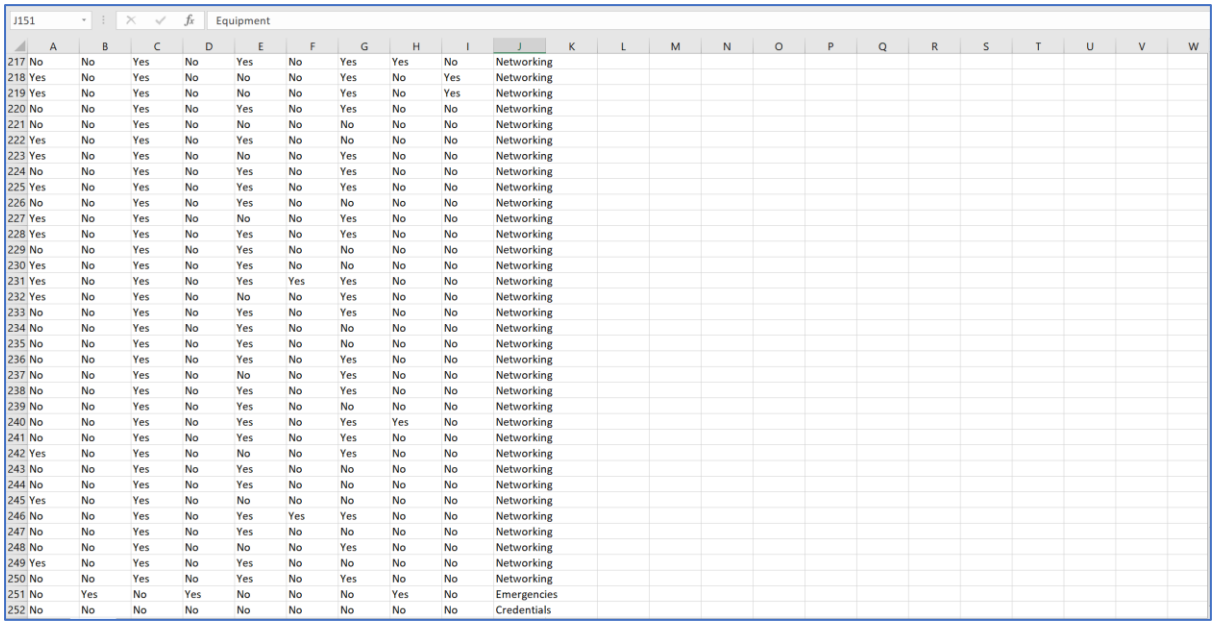


Figure 16

Testing of Intermediate Part:

Below is the training data set before agents take any output from the user.



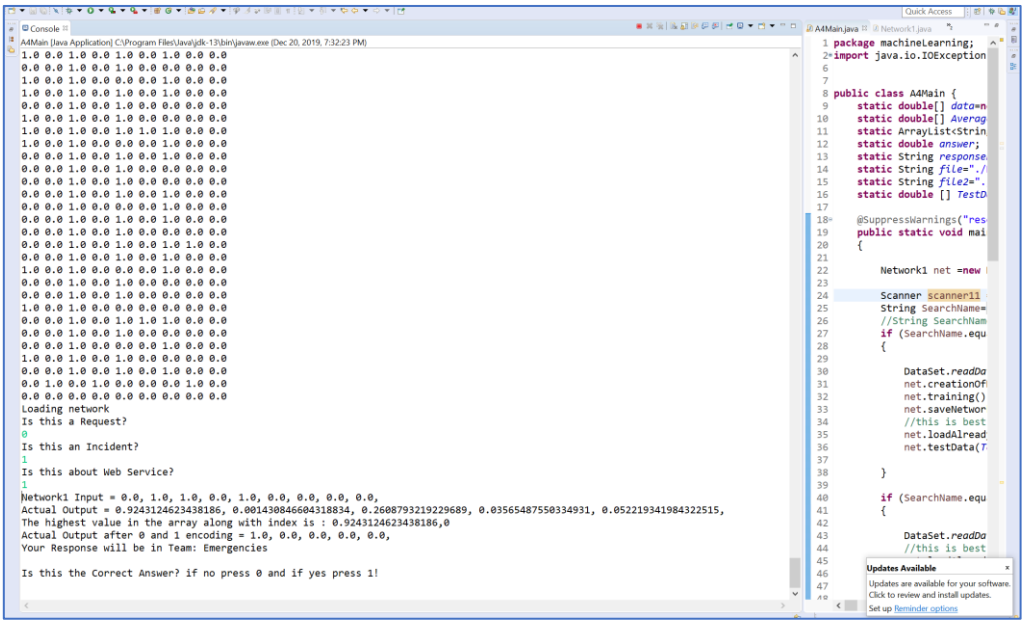
	A	B	C	D	E	F	G	H	I	J
217	No	No	Yes	No	Yes	No	Yes	Yes	No	Networking
218	Yes	No	Yes	No	No	No	Yes	No	Yes	Networking
219	Yes	No	Yes	No	No	No	Yes	No	Yes	Networking
220	No	No	Yes	No	Yes	No	Yes	No	No	Networking
221	No	No	Yes	No	No	No	No	No	No	Networking
222	Yes	No	Yes	No	Yes	No	No	No	No	Networking
223	Yes	No	Yes	No	No	No	Yes	No	No	Networking
224	No	No	Yes	No	Yes	No	Yes	No	No	Networking
225	Yes	No	Yes	No	Yes	No	Yes	No	No	Networking
226	No	No	Yes	No	Yes	No	No	No	No	Networking
227	Yes	No	Yes	No	No	No	Yes	No	No	Networking
228	Yes	No	Yes	No	Yes	No	Yes	No	No	Networking
229	No	No	Yes	No	Yes	No	No	No	No	Networking
230	Yes	No	Yes	No	Yes	No	No	No	No	Networking
231	Yes	No	Yes	No	Yes	Yes	Yes	No	No	Networking
232	Yes	No	Yes	No	No	No	Yes	No	No	Networking
233	No	No	Yes	No	Yes	No	Yes	No	No	Networking
234	No	No	Yes	No	Yes	No	No	No	No	Networking
235	No	No	Yes	No	Yes	No	No	No	No	Networking
236	No	No	Yes	No	Yes	No	Yes	No	No	Networking
237	No	No	Yes	No	No	No	Yes	No	No	Networking
238	No	No	Yes	No	Yes	No	Yes	No	No	Networking
239	No	No	Yes	No	Yes	No	No	No	No	Networking
240	No	No	Yes	No	Yes	No	Yes	Yes	No	Networking
241	No	No	Yes	No	Yes	No	Yes	No	No	Networking
242	Yes	No	Yes	No	No	No	Yes	No	No	Networking
243	No	No	Yes	No	Yes	No	No	No	No	Networking
244	No	No	Yes	No	Yes	No	No	No	No	Networking
245	Yes	No	Yes	No	No	No	No	No	No	Networking
246	No	No	Yes	No	Yes	Yes	Yes	No	No	Networking
247	No	No	Yes	No	Yes	No	No	No	No	Networking
248	No	No	Yes	No	No	No	Yes	No	No	Networking
249	Yes	No	Yes	No	Yes	No	No	No	No	Networking
250	No	No	Yes	No	Yes	No	Yes	No	No	Networking
251	No	Yes	No	Yes	No	No	No	Yes	No	Emergencies
252	No	No	No	No	No	No	No	No	No	Credentials

Here process starts with early predictions, here user wants to ask following inputs from the agent:

Let’s see below, how agent predicts it with asking 3 questions. Sometimes agent predicts it earlier and sometimes it has to ask several questions to predict the answers.

1) 0.0,1.0,1.0,1.0,1.0,1.0.0.0,0.0,1.1

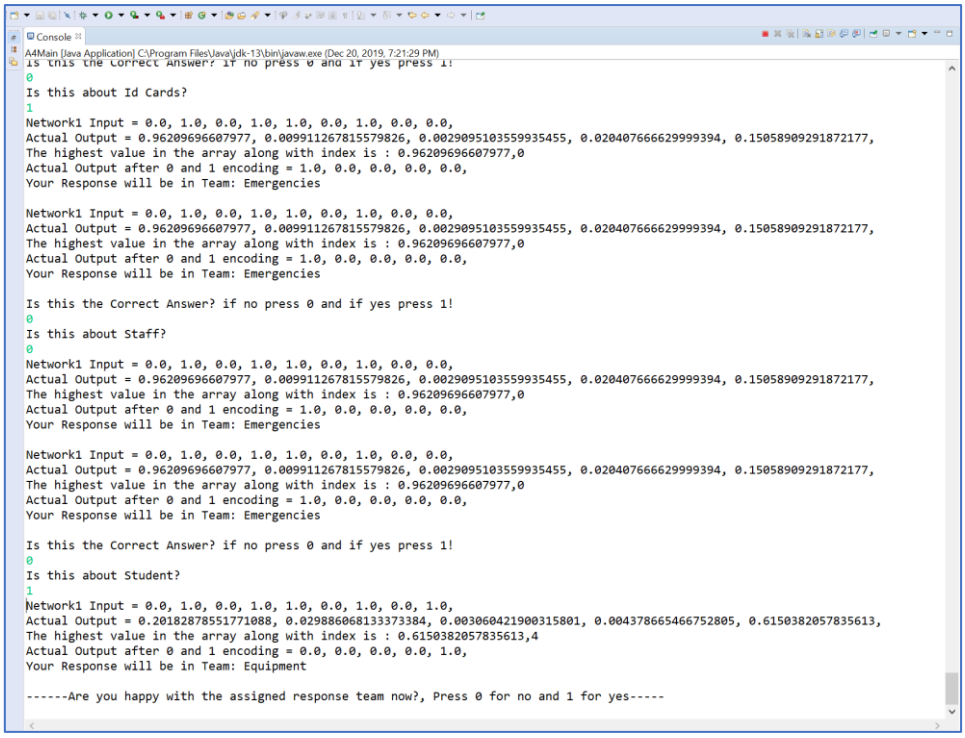
   Actual Response Team: **Emergencies**



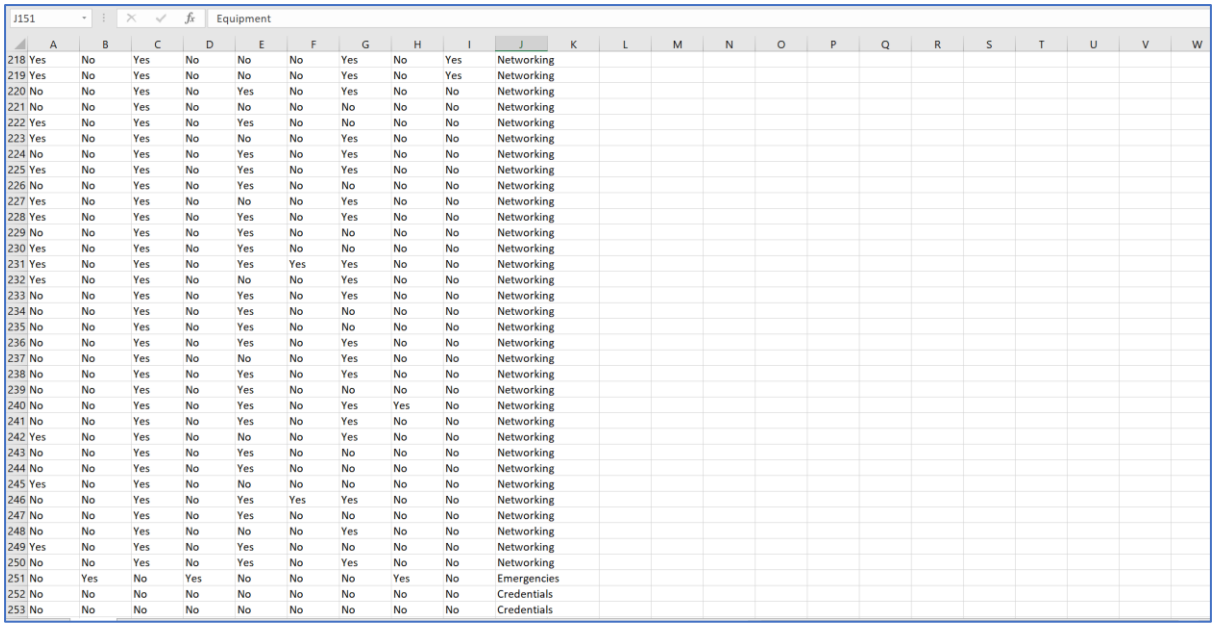
```
1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 0.0
0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0
1.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0
1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 0.0
0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0
1.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0
1.0 0.0 1.0 0.0 1.0 1.0 1.0 0.0 0.0
1.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0
0.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 0.0
0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0
0.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 0.0
0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 1.0 1.0 1.0 0.0 0.0
0.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 0.0
1.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0
0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0
1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 1.0 1.0 1.0 0.0 0.0
0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0
1.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 0.0
0.0 1.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
Loading network
Is this a Request?
0
Is this an Incident?
1
Is this about Web Service?
1
Network1 Input = 0.0, 1.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0,
Actual Output = 0.9243124623438186, 0.0014308466043188834, 0.2608793219229689, 0.03565487550334931, 0.052219341984322515,
The highest value in the array along with index is : 0.9243124623438186,0
Actual Output after 0 and 1 encoding = 1.0, 0.0, 0.0, 0.0, 0.0,
Your Response will be in Team: Emergencies

Is this the Correct Answer? if no press 0 and if yes press 1!
```

2) 0.0,1.0,0.0,1.0,1.0,0.0,1.0,0.0,1.1

   Actual Response Team: **Equipment**



Below I am testing the part of the intermediate where if user is not satisfied by the result, that user input is added into the network and its retrained again on that input. Below image show how the data set is updated with new record.



J151 | Equipment

	A	B	C	D	E	F	G	H	I	J
218	Yes	No	Yes	No	No	No	Yes	No	Yes	Networking
219	Yes	No	Yes	No	No	No	Yes	No	Yes	Networking
220	No	No	Yes	No	Yes	No	Yes	No	No	Networking
221	No	No	Yes	No	No	No	No	No	No	Networking
222	Yes	No	Yes	No	Yes	No	No	No	No	Networking
223	Yes	No	Yes	No	No	No	Yes	No	No	Networking
224	No	No	Yes	No	Yes	No	Yes	No	No	Networking
225	Yes	No	Yes	No	Yes	No	Yes	No	No	Networking
226	No	No	Yes	No	Yes	No	No	No	No	Networking
227	Yes	No	Yes	No	No	No	Yes	No	No	Networking
228	Yes	No	Yes	No	Yes	No	Yes	No	No	Networking
229	No	No	Yes	No	Yes	No	No	No	No	Networking
230	Yes	No	Yes	No	Yes	No	No	No	No	Networking
231	Yes	No	Yes	No	Yes	Yes	Yes	No	No	Networking
232	Yes	No	Yes	No	No	No	Yes	No	No	Networking
233	No	No	Yes	No	Yes	No	Yes	No	No	Networking
234	No	No	Yes	No	Yes	No	No	No	No	Networking
235	No	No	Yes	No	Yes	No	No	No	No	Networking
236	No	No	Yes	No	Yes	No	Yes	No	No	Networking
237	No	No	Yes	No	No	No	Yes	No	No	Networking
238	No	No	Yes	No	Yes	No	Yes	No	No	Networking
239	No	No	Yes	No	Yes	No	No	No	No	Networking
240	No	No	Yes	No	Yes	No	Yes	Yes	No	Networking
241	No	No	Yes	No	Yes	No	Yes	No	No	Networking
242	Yes	No	Yes	No	No	No	Yes	No	No	Networking
243	No	No	Yes	No	Yes	No	No	No	No	Networking
244	No	No	Yes	No	Yes	No	No	No	No	Networking
245	Yes	No	Yes	No	No	No	No	No	No	Networking
246	No	No	Yes	No	Yes	Yes	Yes	No	No	Networking
247	No	No	Yes	No	Yes	No	No	No	No	Networking
248	No	No	Yes	No	No	No	Yes	No	No	Networking
249	Yes	No	Yes	No	Yes	No	No	No	No	Networking
250	No	No	Yes	No	Yes	No	Yes	No	No	Networking
251	No	Yes	No	Yes	No	No	No	Yes	No	Emergencies
252	No	No	No	No	No	No	No	No	No	Credentials
253	No	No	No	No	No	No	No	No	No	Credentials

4. Bibliography

[1] Stuart Russell and Peter Norvig. (1995) Artificial Intelligence, A Modern Approach, Prentice Hall.

[2] John A. Flores. (2011) Focus on Artificial Neural Networks

[3] “What is One Hot Encoding? Why And When do you have to use it?", Hackernoon.com, 2019. [Online]. Available: https://hackernoon.com/what-is-one-hot-encoding-why-and-when-do-you-have-to-use-it-e3c6186d008f. [Accessed: 15- Dec- 2019]

[4] "Apache POI - the Java API for Microsoft Documents". [Online]. Available: https://poi.apache.org/. [Accessed: 16- March- 2019].

[5] S. function and J. Schauer, "Step function versus Sigmoid function", Stack Overflow, 2015. [Online]. Available: https://stackoverflow.com/questions/34469595/step-function-versus-sigmoid-function. [Accessed: 17- Dec- 2019]

[6] "Momentum and Learning Rate Adaptation", Cnl.salk.edu. [Online]. Available: https://cnl.salk.edu/~schraudo/teach/NNcourse/momrate.html. [Accessed: 12- Dec- 2019]