

Name: Denagama Hemachandra

Student Reference Number: 10750003

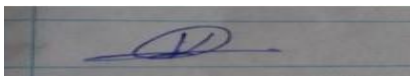
Module Code: PUSL3223	Module Name: AI and Machine Learning
Coursework Title: Assessment 1	
Deadline Date: 05/12/2022	Member of staff responsible for coursework: Dr Neamah Al-Naffakh
Programme: Bsc(Hons) Software Engineering	
Please note that University Academic Regulations are available under Rules and Regulations on the University website www.plymouth.ac.uk/studenthandbook .	
Group work: please list all names of all participants formally associated with this work and state whether the work was undertaken alone or as part of a team. Please note you may be required to identify individual responsibility for component parts.	
<p><i>We confirm that we have read and understood the Plymouth University regulations relating to Assessment Offences and that we are aware of the possible penalties for any breach of these regulations. We confirm that this is the independent work of the group.</i></p> <p>Signed on behalf of the group:</p>	
<p>Individual assignment: <i>I confirm that I have read and understood the Plymouth University regulations relating to Assessment Offences and that I am aware of the possible penalties for any breach of these regulations. I confirm that this is my own independent work.</i></p> <p>Signed : </p>	
Use of translation software: failure to declare that translation software or a similar writing aid has been used will be treated as an assessment offence.	
I *have used /not used translation software.	
If used, please state name of software.....	
<p>Overall mark _____ % Assessors Initials _____ Date _____</p>	

Table of Contents

○ Introduction.....	3
○ Literature review	4
○ Implementation	6
○ Discussion and conclusions	9
○ References	11
○ Appendix	12

Introduction

Artificial Intelligence is a most trending technology in the computing field. It is a computing technology which is working data and their attributes. Machine learning is a technology which is act like a human brain developed using computers. Machine learning is a branch of artificial intelligence in this context (AI). The main aim of machine learning is to comprehend the structure of data and fit it into models that people can comprehend and use. A subfield of computer science, machine learning is distinct from other conventional computer techniques. Ordinary computer programs carry out the predefined algorithms by instructing them and then modifying them to suit their preferences. However, machine learning (ML) is quite distinct from those computer programs since it involves training a data model utilizing historical data and producing an output that people can comprehend. Today almost all the fields are benefited from the uses of the Machin learning. Facial recognitions is common fact which is a result of Machin learning. Some social media platforms use this facial recognition facility for different purposes. This feature is also using for smart phone manufacturing industries as security feature. OCR is a commonly use ML part in many fields. OCR means Optical Character Recognition, it converts texts on the images into the editable computerized format. Recommendation engines, self-driving cars are also the developments of the Machin learning.(Raschka et al., 2020)

Literature review

In classical Machine learning there are three basic approaches to learn and improve the accuracy of a training model. Supervised learning, Unsupervised learning and Reinforcement learning are the basic approaches of traditional machine learning. These learning techniques are basically different from each other. This literature review is based on the common learning techniques called Supervised and Unsupervised learning and their real world implementations.

Supervised Learning - Algorithms for supervised machine learning are created to learn from examples. In supervised learning, the dataset is labeled and the data consists of input and output variables. Data that is labeled indicates that the dataset includes both the input data and the corresponding output. The algorithm looks for patterns in the input data throughout the training phase and correlates those patterns with the intended output data. After training, a supervised learning algorithm uses the unseen data as inputs and uses the training data to decide which label to assign the new inputs. A supervised learning model's goal is to forecast the accurate label for the recently supplied data.

Regression - There are two main problems solving approaches in supervised learning called regression and classification. If an input variable and an output variable have a connection, regression methods are used. It is utilized in situations when the output variable's value is continuous or actual, for as when predicting stock prices, weather, or home prices.

Classification - The process of classifying involves dividing the output into several groups based on one or more input factors. When an output variable's value is discrete or categorical, such as when an email is "spam" or "not spam," "illness" and "no disease," "rain" and "not rain," "Yes" or "No," and 0 or 1, classification is utilized.

As the example for supervised learning approaches can be defined as follows,

- Text categorization.

With use of digital documents text categorization and text identification facilities are become more important. Almost all the fields are now working with the very large amount of text documents. So text categorization or classification mainly divided into two parts called topic based classification and genre based classification. To classify text using supervised learning initial text are needed to train the model.(Ikonomakis et al., n.d.)

- Block-chain and stock prediction using past data.

Stock market is a most trending business field in the world. All the stock prices are changing rapidly due to several reasons. So with the use of past stock price changing patterns some machine learning model are trained to give the stock price prediction. To predict the price so many input attributes are need to be entered. After collection data about price changing rate, time and other reasons those ML models can have to predict new stock prices. This is also a supervised learning example because initial data is needed.(Nabipour et al., 2020)

Unsupervised learning - Since unsupervised learning works with unlabeled data, there are no equivalent output variables in this situation. In contrast to supervised machine learning, this. Users are not required to train or oversee the model in unsupervised learning. There is no supervisor to instruct, and there is no right output. The algorithm itself picks up knowledge from the input data and finds patterns and information to learn and organize the data into groups based on similarities. As a typical illustration of unsupervised learning, consider a dataset of images featuring various breeds of cats and dogs. The dataset is provided to the unsupervised learning algorithm. The algorithm has never been exposed to or trained on the provided dataset,

therefore it is completely unaware of its characteristics. The unsupervised learning algorithm's job is to let the picture characteristics speak for themselves. This work will be carried out by unsupervised learning algorithms that categorize the picture dataset into groups based on how similar the photos are to one another. Clustering and Association issues are additional categories under which the unsupervised learning algorithm may be divided.(Âraud & Ârot, n.d.)

Clustering -Unsupervised learning relies on the idea of clustering. It is employed to uncover any hidden patterns or structures in unclassified data. The uncategorized data are processed by the clustering algorithm, which separates them into several clusters (groups) such that things with a lot of similarities stay in the same group and have little to no similarities with objects in other groups. It creates the two groupings of cats and dogs.

Association - A technique for unsupervised learning is an association rule. It is used to uncover the connections among the data in a big dataset. It establishes the group of items that co-occur in the collection. Marketing strategy is more successful because to the association rule.

Unsupervised learning real world examples

- Malware detection systems

Malwares are a major threat to IT industries in these days. Those malwares are executing in the computer systems without asking any permission from the user. Those are also a computer viruses developed hacker or some unauthorized parties to breaking down the large scale of computer networks or hacking the highly secured systems. Using the unsupervised learning technology there are Machine learning models and systems to detect those malwares using past data.

- Automated email activity management systems

Email is a commonly use professional communication method. Every people have to a email because those email addresses are connected with every other accounts. Nowadays lots of emails are coming for different purposes. Those emails are saved in the inbox and overloaded and overwritten because email are not deleted automatically. Using unsupervised learning those emails are partitioned with the activities which they are associated. After identifying the activities and delete similar emails automatically.(Kushmerick & Lau, 2005)

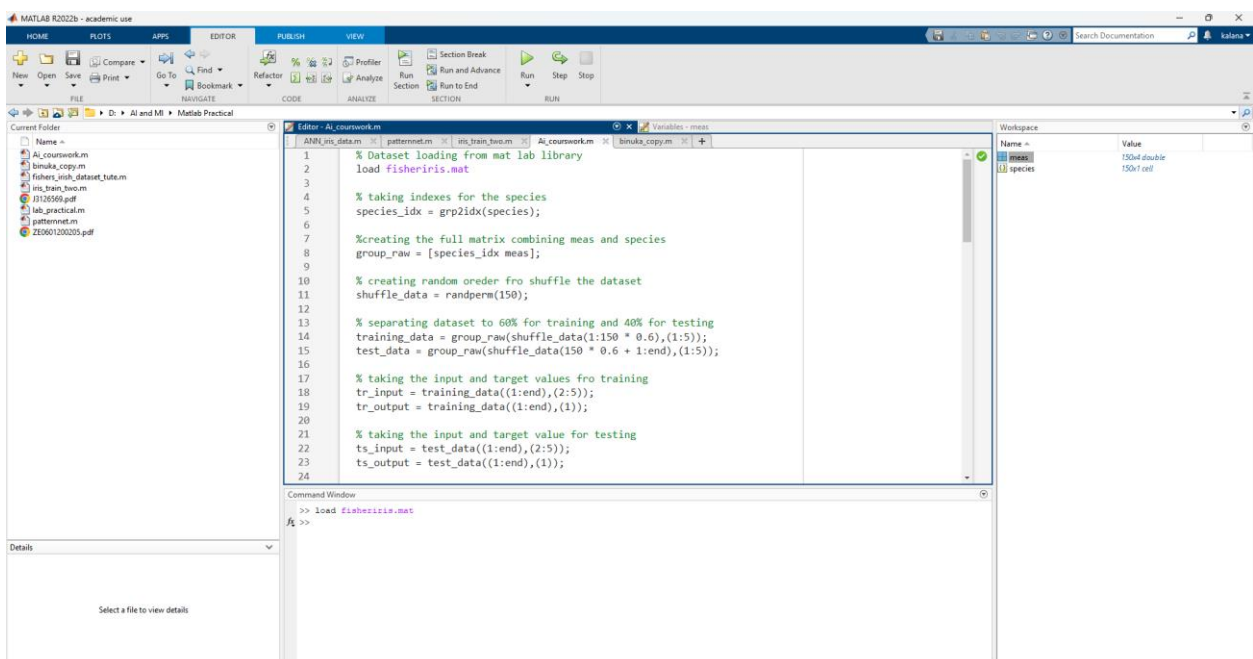
Implementation of Supervised Learning

Neural network is a graphical example of the human nervous system. In a neural network there are elements called neurons. Those neurons are separated into different layers called input layer, output layer and the hidden layers. Those layers are connected to the another layer with the connections. Those connection have value called weight and that values are changed according to the values of the neurons.

For implement the neural network some inputs values are need to give as the inputs. With the data which are input layer neurons are activated. Those activated neurons relate to the hidden layer neurons. There is function called activation function and this function decide which neurons to be activated or not. When activating a function bias is shift the value of the activation function.(Ball & Tissot, n.d.)

For Implementing the neural network for calcification here fisheriris data set had been used. Iris data set contains the attributes of the iris flower like sepal length, sepal width, petal length, petal width. 150 records categorized in to three spices of flowers. This dataset is the commonly used dataset for the beginners.

math lab contains this fisheriris dataset in the libraries. So we don't need to download the dataset and import it to the MATLAB. From the load function



load **fisheriris.mat**

Initially dataset need to prepare for train the neural network model. For data preparation iris dataset need to be loaded. So Loading the dataset from the MATLAB library is done in 2dn code line. Load is a built in function that is used to call datasets from the MATLAB libraries. After loading the dataset all the data can be view from the workspace through the matrix. In the meas matrix all the iris flower attributes are stored raw by raw. And the species matrix is the target of this training model.

```
species_to_numbers = grp2idx(species);
```

After loading the data those data need to be divided into two parts taking 60% for training and rest 40% training. But here's a problem, words or letters are difficult use for MATLAB calculations. To overcome this

problem species with words need to convert into numbers. In the species matrix there are three categories called setosa, virginica, versin. For calculation those categories need to converted into numbers like 1,2 and 3 using the following syntax.

```
rdm_idx_for_shuffle = randperm(150);
```

When dividing the data for training and testing data need to be shuffled because 60% is taking from 150 rows. $150 * 0.6 = 90$ so this 90 rows are takes form 1 to 90. If the data is not shuffled only Setosa and versicolor flower,s attributes separated as the input for training. To shuffle the dataset randomly here randperm function is used. This functions gives random indexes for shuffling. Accordingly to this random indexes meas and species matrix are shuffled. After shuffling the meas and rdm_idx_for_shuffle matrixes first 60% taking as the training data and rest 40% will be taken as the testing data. To filter this data below syntax is used.

```
training_input_60 = meas(rdm_idx_for_shuffle(1:150 * 0.6),(1:4));
```

```
testing_input_40 = meas(rdm_idx_for_shuffle(150 * 0.6 + 1:end),(1:4));
```

60 from meas matrix is taken as the training input and 60% from species_num matrix taken as the target data for the training inputs.

```
training_target_60 = species_to_numbers(rdm_idx_for_shuffle(1:150*0.6),(1:1));
```

```
testing_target_40 = species_to_numbers(rdm_idx_for_shuffle(1:150*0.6),(1:1));
```

Tr_target_data matrix contain with the 1,2 and 3 numbers for categorize. But to train the network traing inputa data columns and training target columns need to equal. To do that dummyvar function is called here. From that function 1 is converted as 0 0 1 and 2 is converted as 0 1 0 and 3 is converted as 1 0 0.

```
training_traget_dummy = dummyvar(training_target_60);
```

```
testing_traget_dummy = dummyvar(testing_target_40);
```

After converting the values input and target must have same amount of columns. But the generated matrixes are not have the same column. to train the network there should be the same samples of data as the input and target. So the dimensions of the matrixes are changed using the follow function.

```
final_training_input_data = training_input_60';
```

```
final_testing_input_data = testing_input_40';
```

after preparing the data to train the network for loop is used to train the network with different number of hidden layers. In the first loop 5,10,15 and 20 is automatically increased starting from 5.

```
for hidd_layer_count = 5:5:20
```

to run the each network 10 times another nested for loop have been used. This loop is starting from 1 and run 10 times and train the network 10 times repeatedly.

```
for experiment_count=1:1:10
```

inside the second loop all the network creation and training will be done. Here feedforwardnet neural network is used. 'trainlm' is used as the activation functions and 'mse' is used as the performance.

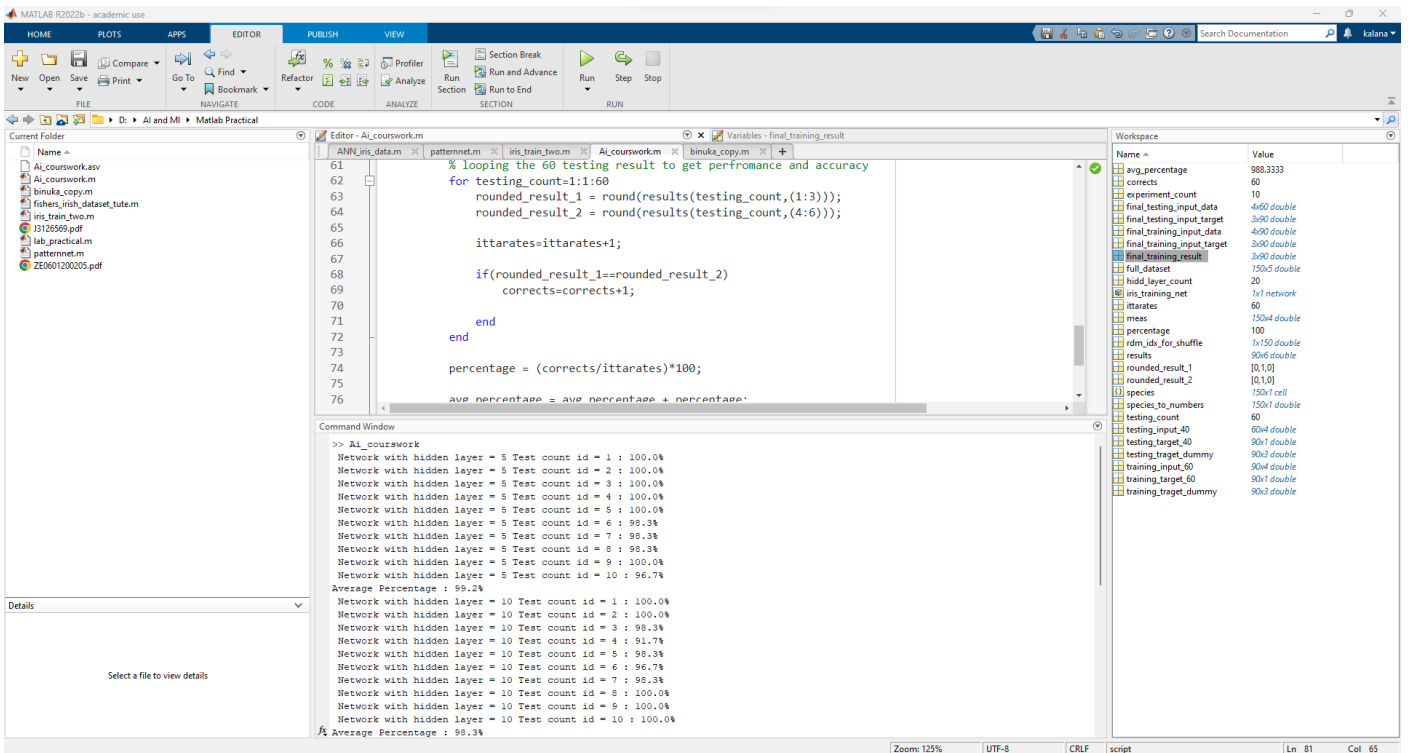
```
iris_training_net = feedforwardnet(hidd_layer_count);  
  
iris_training_net.trainFcn = 'trainlm';  
  
iris_training_net.performFcn = 'mse';  
  
iris_training_net = train(iris_training_net, final_training_input_data,  
    final_training_input_target);
```

after training the network to check if the trained network gives the correct prediction it should be checked inputting some data. For that testing data will input the network and take the performance and accuracy for each run using the following process.

```
percentage = (corrects/ittarates)*100;  
  
avg_percentage = avg_percentage + percentage;  
  
fprintf(' Network with hidden layer = %d Test count id = %d :  
%4.1f%%\n',hidd_layer_count,experiment_count,percentage);  
  
fprintf('Average Percentage : %4.1f%%\n',avg_percentage/10);
```


Discussion and conclusion

After training the network all the testing steps are printed with the performance and accuracy to check the difference of the performance according to the different hidden layers. Following screenshots gives the corresponding average performance and accuracy for each neural network training. Every training step have the different values of performance and accuracy.



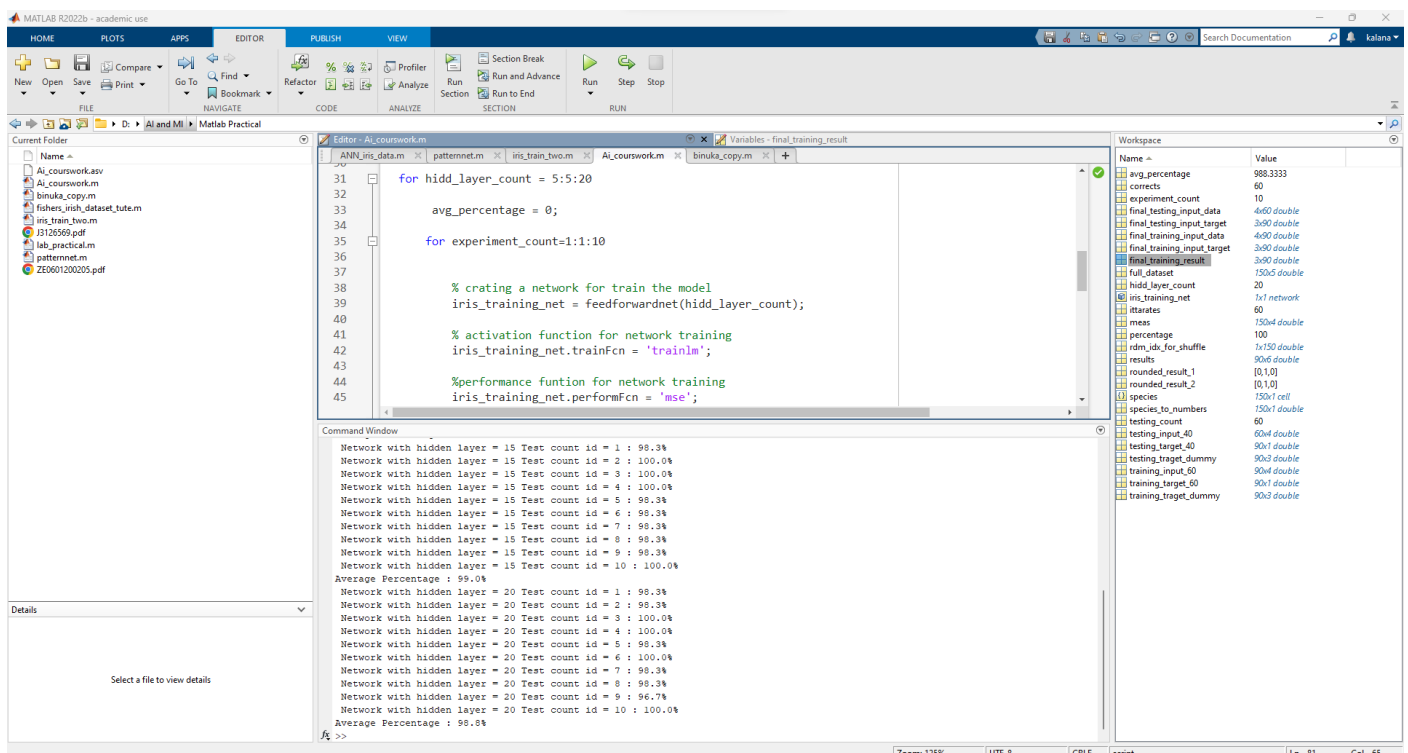
```
61 % looping the 60 testing result to get performance and accuracy
62 for testing_count=1:60
63     rounded_result_1 = round(results(testing_count,(1:3)));
64     rounded_result_2 = round(results(testing_count,(4:6)));
65
66     ittarates=ittarates+1;
67
68     if(rounded_result_1==rounded_result_2)
69         corrects=corrects+1;
70
71     end
72
73 end
74
75 percentage = (corrects/ittarates)*100;
76
77 avg_percentage = avg_percentage + percentage;
```

Command Window

```
>> AI_courswork
Network with hidden layer = 5 Test count id = 1 : 100.0%
Network with hidden layer = 5 Test count id = 2 : 100.0%
Network with hidden layer = 5 Test count id = 3 : 100.0%
Network with hidden layer = 5 Test count id = 4 : 100.0%
Network with hidden layer = 5 Test count id = 5 : 100.0%
Network with hidden layer = 5 Test count id = 6 : 98.3%
Network with hidden layer = 5 Test count id = 7 : 98.3%
Network with hidden layer = 5 Test count id = 8 : 98.3%
Network with hidden layer = 5 Test count id = 9 : 100.0%
Network with hidden layer = 5 Test count id = 10 : 96.7%
Average Percentage : 99.2%
Network with hidden layer = 10 Test count id = 1 : 100.0%
Network with hidden layer = 10 Test count id = 2 : 100.0%
Network with hidden layer = 10 Test count id = 3 : 98.3%
Network with hidden layer = 10 Test count id = 4 : 91.7%
Network with hidden layer = 10 Test count id = 5 : 98.3%
Network with hidden layer = 10 Test count id = 6 : 96.7%
Network with hidden layer = 10 Test count id = 7 : 98.3%
Network with hidden layer = 10 Test count id = 8 : 100.0%
Network with hidden layer = 10 Test count id = 9 : 100.0%
Network with hidden layer = 10 Test count id = 10 : 100.0%
Average Percentage : 98.3%
```

Workspace

Name	Value
avg_percentage	988.3333
corrects	60
experiment_count	10
final_testing_input_data	4x60 double
final_testing_input_target	3x60 double
final_training_input_data	4x60 double
final_training_input_target	3x60 double
full_dataset	150x5 double
hidd_layer_count	20
iris_training_net	1x1 network
ittarates	60
meas	150x4 double
percentage	100
rdm_idx_for_shuffle	1x150 double
results	90x6 double
rounded_result_1	[0,1,0]
rounded_result_2	[0,1,0]
species	150x1 cell
species_to_numbers	150x1 double
testing_count	60
testing_input_40	60x4 double
testing_target_40	90x1 double
testing_traget_dummy	90x3 double
training_input_60	90x4 double
training_target_60	90x1 double
training_traget_dummy	90x3 double



```
31 for hidd_layer_count = 5:5:20
32
33     avg_percentage = 0;
34
35     for experiment_count=1:1:10
36
37
38         % crating a network for train the model
39         iris_training_net = feedforwardnet(hidd_layer_count);
40
41         % activation function for network training
42         iris_training_net.trainFcn = 'trainlm';
43
44         %performance funtion for network training
45         iris_training_net.performFcn = 'mse';
```

Command Window

```
Network with hidden layer = 15 Test count id = 1 : 98.3%
Network with hidden layer = 15 Test count id = 2 : 100.0%
Network with hidden layer = 15 Test count id = 3 : 100.0%
Network with hidden layer = 15 Test count id = 4 : 100.0%
Network with hidden layer = 15 Test count id = 5 : 98.3%
Network with hidden layer = 15 Test count id = 6 : 98.3%
Network with hidden layer = 15 Test count id = 7 : 98.3%
Network with hidden layer = 15 Test count id = 8 : 98.3%
Network with hidden layer = 15 Test count id = 9 : 98.3%
Network with hidden layer = 15 Test count id = 10 : 100.0%
Average Percentage : 99.0%
Network with hidden layer = 20 Test count id = 1 : 98.3%
Network with hidden layer = 20 Test count id = 2 : 98.3%
Network with hidden layer = 20 Test count id = 3 : 100.0%
Network with hidden layer = 20 Test count id = 4 : 100.0%
Network with hidden layer = 20 Test count id = 5 : 98.3%
Network with hidden layer = 20 Test count id = 6 : 100.0%
Network with hidden layer = 20 Test count id = 7 : 98.3%
Network with hidden layer = 20 Test count id = 8 : 98.3%
Network with hidden layer = 20 Test count id = 9 : 96.7%
Network with hidden layer = 20 Test count id = 10 : 100.0%
Average Percentage : 98.0%
```

Workspace

Name	Value
avg_percentage	988.3333
corrects	60
experiment_count	10
final_testing_input_data	4x60 double
final_testing_input_target	3x60 double
final_training_input_data	4x60 double
final_training_input_target	3x60 double
full_dataset	150x5 double
hidd_layer_count	20
iris_training_net	1x1 network
ittarates	60
meas	150x4 double
percentage	100
rdm_idx_for_shuffle	1x150 double
results	90x6 double
rounded_result_1	[0,1,0]
rounded_result_2	[0,1,0]
species	150x1 cell
species_to_numbers	150x1 double
testing_count	60
testing_input_40	60x4 double
testing_target_40	90x1 double
testing_traget_dummy	90x3 double
training_input_60	90x4 double
training_target_60	90x1 double
training_traget_dummy	90x3 double

According to the above screenshots network with the 5 hidden layers have the best average performance value with 99.2%. network with the 10 layers have the 98.3% of average percentage. In here we can see the percentage value is decreased when number of hidden layers are increased. Network with the 15 hidden layers have the 99.0% percentage. Again the network with the 20 hidden layers have the 98.8% percentage. So this percentage in not parallelly increase with the number of hidden layers. It depends on another circumstances.

References

- Âraud, R. F., & Ârot, F. C. (n.d.). *A methodology to explain neural network classi@cation*.
www.elsevier.com/locate/neunet
- Ball, R., & Tissot, P. (n.d.). *Demonstration of Artificial Neural Network in Matlab*.
- Ikonomakis, M., Kotsiantis, S., & Tampakas, V. (n.d.). *Text Classification Using Machine Learning Techniques*.
- Kushmerick, N., & Lau, T. (2005). *Automated Email Activity Management: An Unsupervised Learning Approach*. <http://www.freshdirect.com/account.jsp>
- Nabipour, M., Nayyeri, P., Jabani, H., Mosavi, A., Salwana, E., & Shahab, S. (2020). Deep learning for stock market prediction. *Entropy*, 22(8). <https://doi.org/10.3390/E22080840>
- Raschka, S., Patterson, J., & Nolet, C. (2020). Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. In *Information (Switzerland)* (Vol. 11, Issue 4). MDPI AG. <https://doi.org/10.3390/info11040193>

Appendix

```
% Dataset loading from mat lab library
load fisheriris.mat

% taking indexes for the species
species_to_numbers = grp2idx(species);

% creating random order fro shuffle the dataset
rdm_idx_for_shuffle = randperm(150);

% separating dataset to 60% for training inputs and 40% for testing inputs
training_input_60 = meas(rdm_idx_for_shuffle(1:150 * 0.6),(1:4));
testing_input_40 = meas(rdm_idx_for_shuffle(150 * 0.6 + 1:end),(1:4));

% separating datas to 60% for training target and 40% for testing target
training_target_60 = species_to_numbers(rdm_idx_for_shuffle(1:150*0.6),(1:1));
testing_target_40 = species_to_numbers(rdm_idx_for_shuffle(1:150*0.6),(1:1));

% creating dummy variables for 1,2 and 3 numbers for catogorize
training_traget_dummy = dummyvar(training_target_60);
testing_traget_dummy = dummyvar(testing_target_40);

%changing the dimensions of input matrix
final_training_input_data = training_input_60';
final_testing_input_data = testing_input_40';

% changing the demensions if target matrix
final_training_input_target = training_traget_dummy';
final_testing_input_target = testing_traget_dummy';

for hidd_layer_count = 5:5:20

    avg_percentage = 0;

    for experiment_count=1:1:10

        % crating a network for train the model
        iris_training_net = feedforwardnet(hidd_layer_count);

        % activation function for network training
        iris_training_net.trainFcn = 'trainlm';

        %performance funtion for network training
        iris_training_net.performFcn = 'mse';

        % stariting the training of the network
        iris_training_net = train(iris_training_net, final_training_input_data,
final_training_input_target);

        %Testing the trained network
        final_training_result = iris_training_net(final_training_input_data);

        %Viewing the testing results
        results = [final_training_result' final_training_input_target'];

        ittarates = 0;
        corrects = 0;
```

```

% looping the 60 testing result to get performance and accuracy
for testing_count=1:1:60
    rounded_result_1 = round(results(testing_count,(1:3)));
    rounded_result_2 = round(results(testing_count,(4:6)));

    ittarates=ittarates+1;

    if(rounded_result_1==rounded_result_2)
        corrects=corrects+1;

    end
end

percentage = (corrects/ittarates)*100;

avg_percentage = avg_percentage + percentage;

fprintf(' Network with hidden layer = %d Test count id = %d :
%.1f%%\n',hidd_layer_count,experiment_count,percentage);

end
fprintf('Average Percentage : %.1f%%\n',avg_percentage/10);
end

```

Screenshots

The screenshot displays the MATLAB R2022b environment. The main editor window shows a script for training a neural network. A 'Neural Network Training' window is open, showing the training progress and results. The workspace window on the right lists various variables and their values.

Neural Network Training Window:

- Training Results:** Training finished. Met validation criterion.
- Training Progress:**

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	13	1000
Elapsed Time	-	00:00:00	-
Performance	0.798	0.00417	0
Gradient	1.33	0.0503	1e-07
Mu	0.001	1e-05	1e+10
Validation Checks	0	6	6
- Training Algorithms:**
 - Data Division: Random
 - Training: Levenberg-Marquardt
 - Performance: Mean Squared Error
 - Calculations: MEX
- Training Plots:** Performance, Training State, Error Histogram, Regression.

Workspace Window:

Name	Value
avg_percentage	970
corrects	57
experiment_count	10
final_testing_input_data	4x60 double
final_testing_input_target	3x60 double
final_training_input_data	4x90 double
final_training_input_target	3x90 double
final_training_result	3x60 double
full_dataset	150x5 double
hidd_layer_count	20
iris_training_net	1x1 network
iterates	60
meas	150x4 double
percentage	95
rdm_idx_for_shuffle	1x150 double
results	90x6 double
rounded_result_1	[0,1,0]
rounded_result_2	[0,1,0]
species	150x1 cell
species_to_numbers	150x1 double
testing_count	60
testing_input_40	60x4 double
testing_target_40	90x1 double
testing_target_dummy	90x3 double
training_input_60	90x4 double
training_target_60	90x1 double
training_target_dummy	90x3 double

The screenshot displays the MATLAB R2022b environment. The main editor window shows a script for training a neural network. A 'Feed-Forward Neural Network (view)' window is open, showing the network architecture. The workspace window on the right lists various variables and their values.

Feed-Forward Neural Network (view) Window:

- Input:** 4 nodes.
- Hidden:** 20 nodes.
- Output:** 3 nodes.
- Weights:** W and b.
- Activation Functions:** Sigmoid and ReLU.

Workspace Window:

Name	Value
avg_percentage	970
corrects	57
experiment_count	10
final_testing_input_data	4x60 double
final_testing_input_target	3x60 double
final_training_input_data	4x90 double
final_training_input_target	3x90 double
final_training_result	3x60 double
full_dataset	150x5 double
hidd_layer_count	20
iris_training_net	1x1 network
iterates	60
meas	150x4 double
percentage	95
rdm_idx_for_shuffle	1x150 double
results	90x6 double
rounded_result_1	[0,1,0]
rounded_result_2	[0,1,0]
species	150x1 cell
species_to_numbers	150x1 double
testing_count	60
testing_input_40	60x4 double
testing_target_40	90x1 double
testing_target_dummy	90x3 double
training_input_60	90x4 double
training_target_60	90x1 double
training_target_dummy	90x3 double

