Engineering Project I

Data Analysis with Matlab

Ayşıl Simge Karacan

1804010004

BEYKOZ UNIVERSITY

Department of Computer Engineering

PROJECT REPORT

Submission Date: 29 May 2020

Table Of Contents

Data Analysis with Matlab 1
Introduction 4
Some Concepts 5
- Matlab 5
- Neural Networks 5
- Machine Learning 6
- Multi Layer Preception (MLP) 6

Training The Iris Data with NNTool 7
Training Wine Data With Using Classification Learner Toolbox 16
Training Seeds Data With LIBSVM 20
- SVM 20
- LIBSVM 20
- Training Data With LIBSVM 21

Training Seeds Data With KNN 25
- KNN 25
- Training Data With KNN 25
- Picking a value for 'K' 27

Comparing SVM And KNN Methods 27
- Results For KNN 29
- Results For SVM 30
- KNN and SVM 31

Future Works 32
Conclusion 32
RESOURCES 33

Introduction

Purpose Of The Project

The purpose of this project is data analyzing by using artificial neural networks to learn how neural networks works, understanding the concepts of underneath the neural network. In addition to that, getting familiar with the Matlab tools when it's needed and learn these tools and toolboxes of the selected tool properly. Also training data without using any toolboxes, learning preparing/training and testing data, and comparing with some comparison methods.

During our first prototype our aim was to understand what is neural network, what is the difference between supervised and unsupervised learning and classification or regression methods. Also we learned how we can train data with using Matlab Toolboxes.

After our first prototype we started to continue our projects individually. My goal was understanding the SVM method and training the data without using any toolbox with LIBSVM.

After my second prototype I started to train the data with KNN method, and compared the two results I found.

This report will show you what is my evoluations, struggles and process.

Some Concepts

Matlab

MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages.i

Neural Networks

Neural systems are a set of of algorithms, demonstrated loosely after the human brain cells, that are intended to recognize patterns. They decipher tangible information through a sort of machine discernment, marking or bunching crude info. The examples they perceive are numerical, contained in vectors, into which all genuine information, be it pictures, sound, content or time arrangement, must be interpreted.

Neural Network is a structure established in layers. The first layer is called as input and the last layer is output. The layers in the middle are called 'Hidden Layers'. Each layer contains a certain number of neuron. These neurons are connected to each other by 'Synaps'. Synapses contain a coefficient. These coefficients say how important the information in the neuron to which they are connected is.

The value of a neuron is found by multiplying the inputs to that neurons by multiplying them with coefficients. This final result is put into an activation function. According to the result of the function, it is decided whether or not that neuron will be fired.

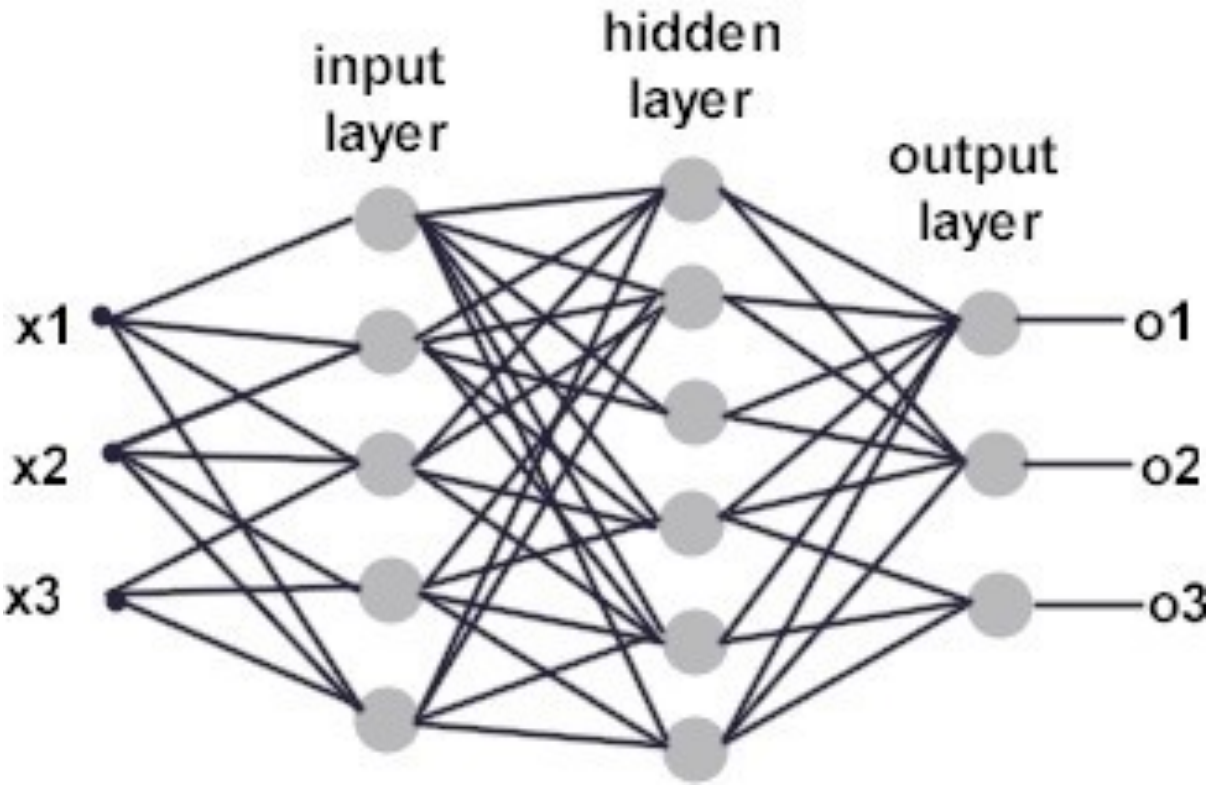
Machine Learning

Machine learning flips the script. We want the program itself to learn the rules that describe our data the best, by finding patterns in what we know and applying those patterns to what we don't know.

These algorithms are able to learn. Their performance gets better with each iteration, as it uncovers more hidden trends in the data.

Multi Layer Perception (MLP)

The perceptron is very useful for classifying data sets that are linearly separable. The MultiLayer Perceptron (MLPs) breaks this restriction and classifies datasets which are not linearly separable. They do this by using a more robust and complex architecture to learn regression and classification models for difficult datasets.[ii]



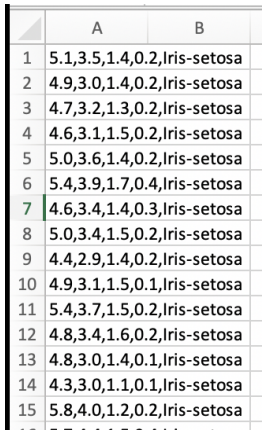
The network can be divided into three main layers.

1. Input Layers: This is the initial layer of the network which takes in an input which will be used to produce an output.
2. Hidden Layers: The network needs to have at least one hidden layer. The hidden layers perform computations and operations on the input data to produce something meaningful.
3. Output Layer: The neurons in this layer display a meaningful output.[iii]

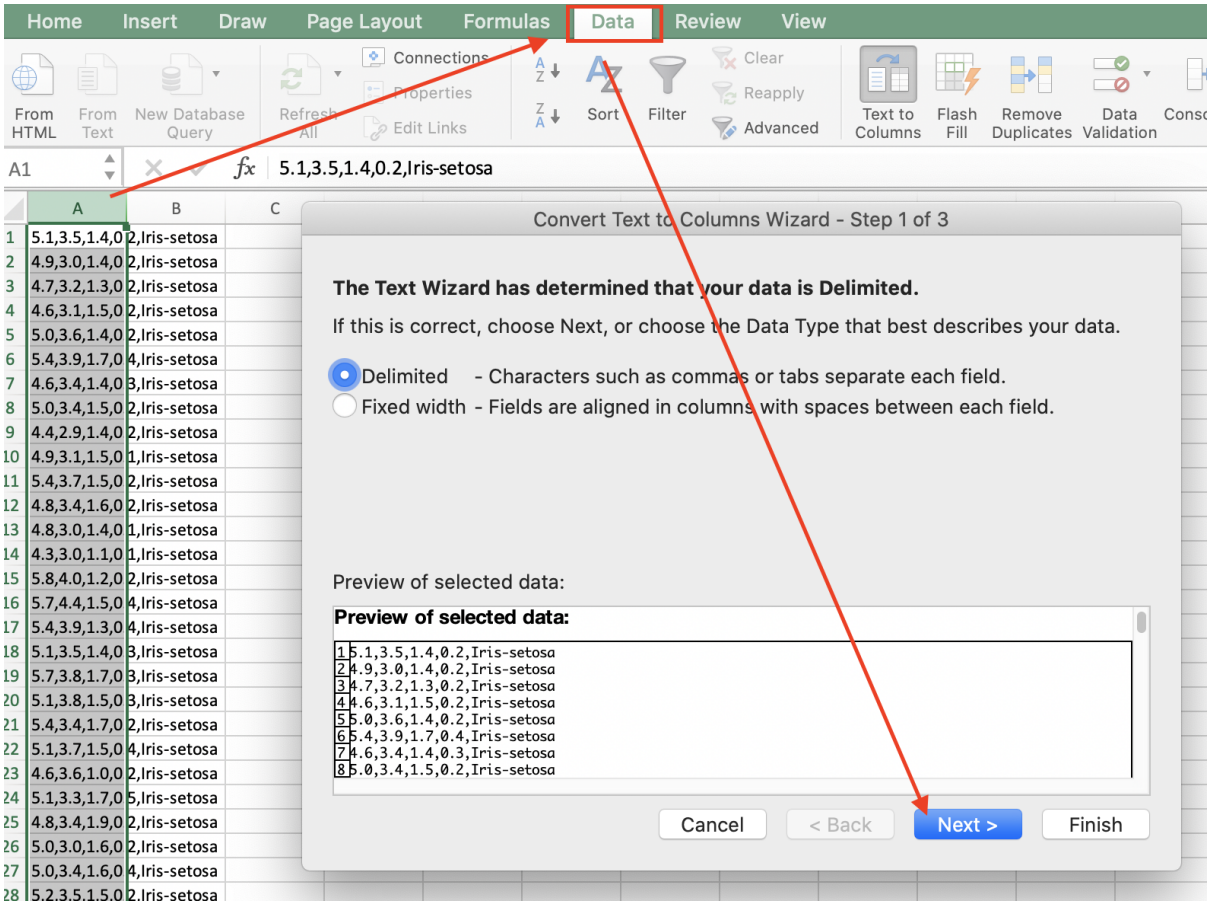
Training The Iris Data with NNTool

Data is sourced by UCI Repository. [iv]

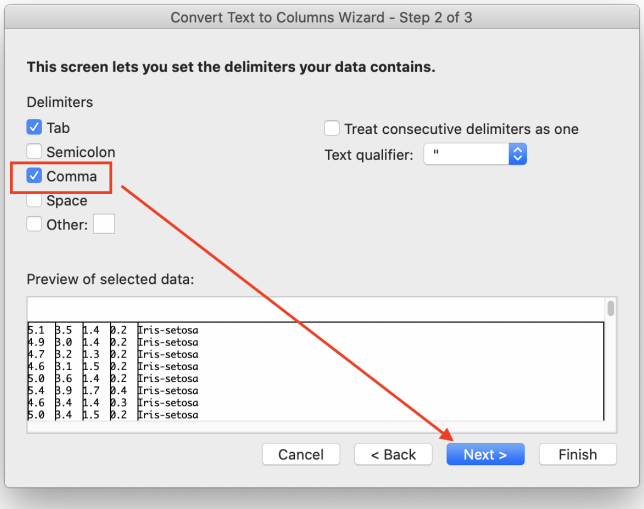
Our first struggle was when we open the data file with Excel, we found that the data is separated with comma as can be seen from the picture. In order to solve this problem we used this process:



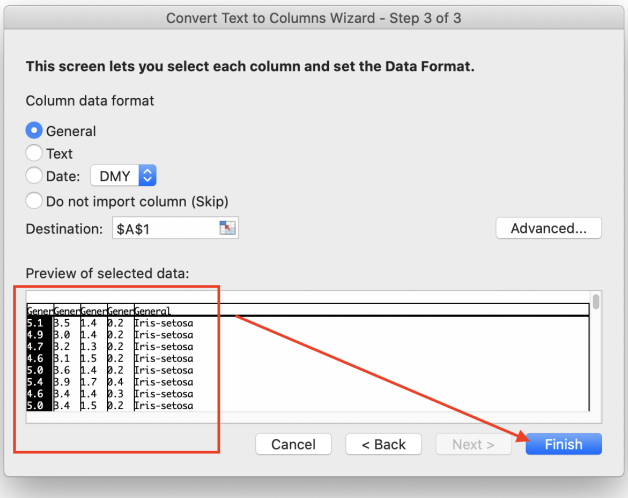
	A	B
1	5.1,3.5,1.4,0.2,Iris-setosa	
2	4.9,3.0,1.4,0.2,Iris-setosa	
3	4.7,3.2,1.3,0.2,Iris-setosa	
4	4.6,3.1,1.5,0.2,Iris-setosa	
5	5.0,3.6,1.4,0.2,Iris-setosa	
6	5.4,3.9,1.7,0.4,Iris-setosa	
7	4.6,3.4,1.4,0.3,Iris-setosa	
8	5.0,3.4,1.5,0.2,Iris-setosa	
9	4.4,2.9,1.4,0.2,Iris-setosa	
10	4.9,3.1,1.5,0.1,Iris-setosa	
11	5.4,3.7,1.5,0.2,Iris-setosa	
12	4.8,3.4,1.6,0.2,Iris-setosa	
13	4.8,3.0,1.4,0.1,Iris-setosa	
14	4.3,3.0,1.1,0.1,Iris-setosa	
15	5.8,4.0,1.2,0.2,Iris-setosa	



After selecting the entire column for the solution of the problem, follow the Excel Menu -> Data -> Next



Then we make sure Comma is Selected and click Next.

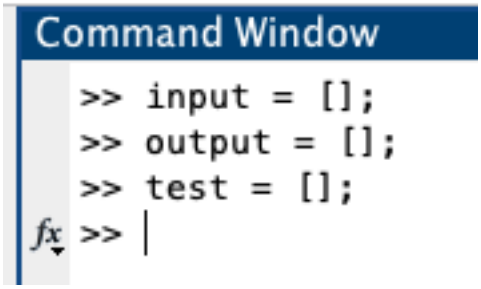


We can see the preview of selected data from here, if everything seemscorrect then we click Finish. Then the attributes are separeted and we can set our training and test datas.

	A	B	C	D	E
1	5.1	3.5	1.4	0.2	1
2	4.9	3.0	1.4	0.2	1
3	4.7	3.2	1.3	0.2	1
4	4.6	3.1	1.5	0.2	1
5	5.0	3.6	1.4	0.2	1
6	5.4	3.9	1.7	0.4	1
7	4.6	3.4	1.4	0.3	1
8	5.0	3.4	1.5	0.2	1
9	4.4	2.9	1.4	0.2	1
10	4.9	3.1	1.5	0.1	1
11	5.4	3.7	1.5	0.2	1
12	4.8	3.4	1.6	0.2	1
13	4.8	3.0	1.4	0.1	1
14	4.3	3.0	1.1	0.1	1
15	5.8	4.0	1.2	0.2	1
16	5.7	4.4	1.5	0.4	1
17	5.4	3.9	1.3	0.4	1
18	5.1	3.5	1.4	0.3	1
19	5.7	3.8	1.7	0.3	1
20	5.1	3.8	1.5	0.3	1

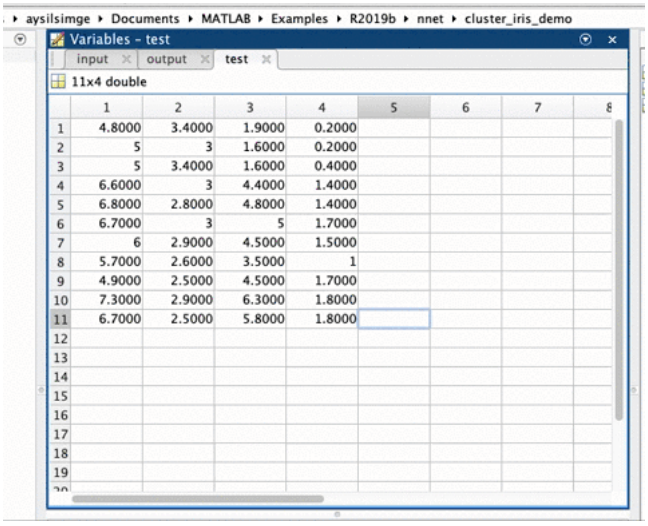
Attributes type in column A is Sepal Length for B is Sepal Witdth for C is Petal Width and for the last one (E) is the type of the flower (class).

We chose 11 random rows for testing and deleted them from the input data (Flower type will be our output and other data will be the input).



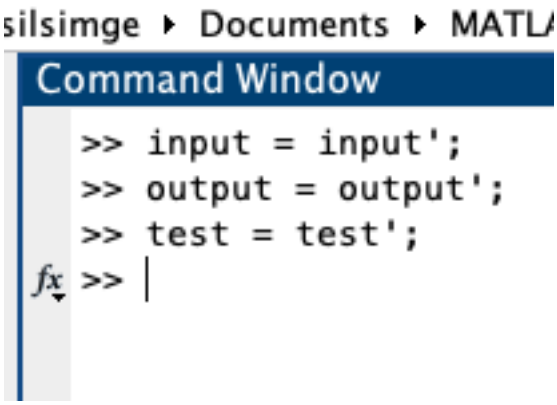
Now we imported our datas into the Matlab with creating three variables called output, input and test.

We opened the variables (arrays) we created and import the according data.

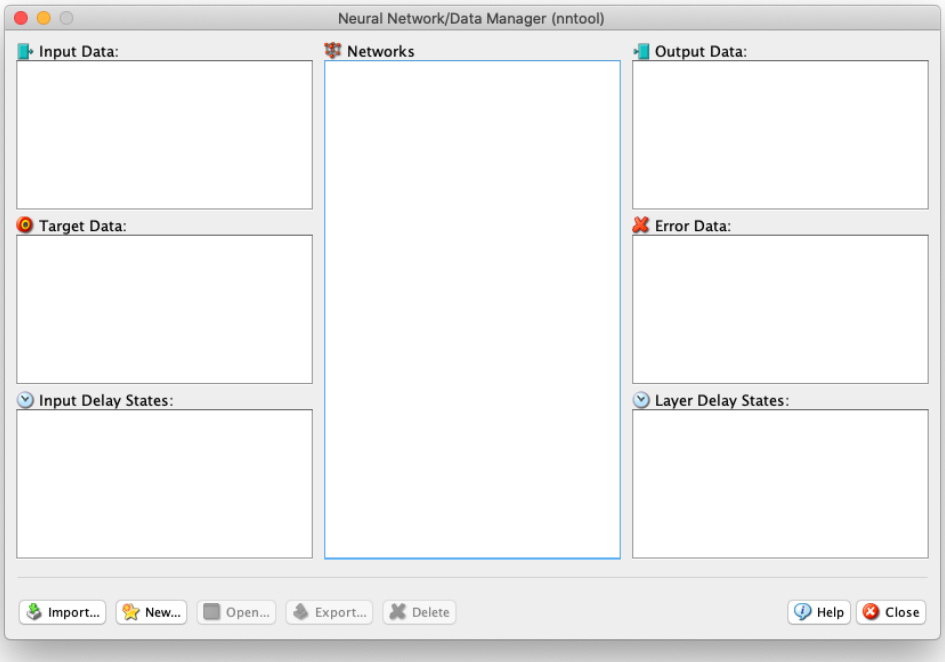


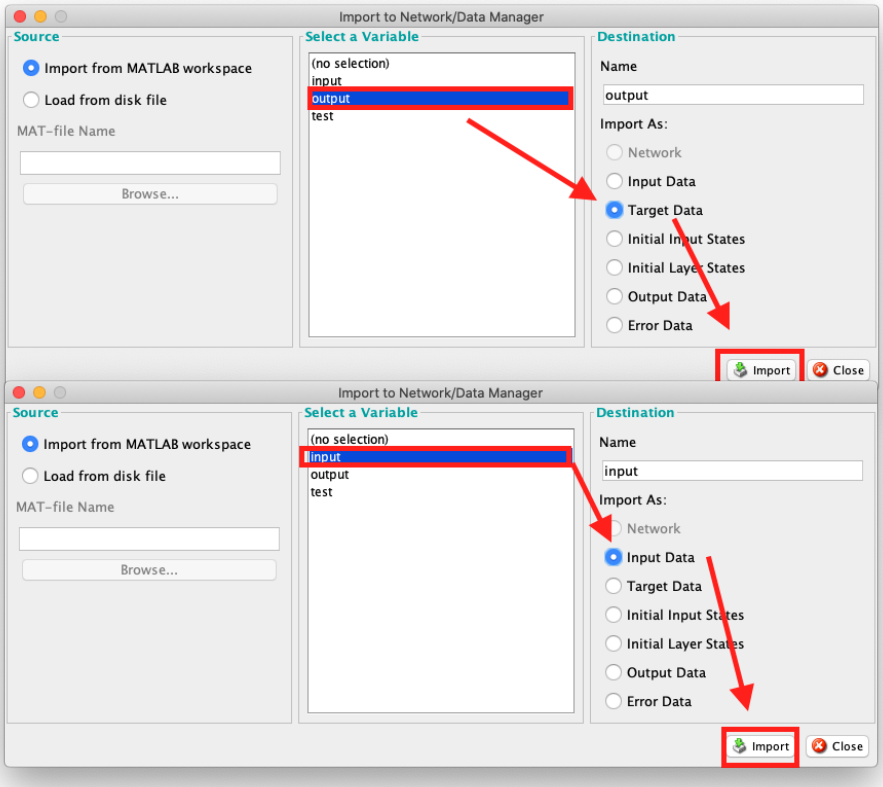
	1	2	3	4	5	6	7	8
1	4.8000	3.4000	1.9000	0.2000				
2	5	3	1.6000	0.2000				
3	5	3.4000	1.6000	0.4000				
4	6.6000	3	4.4000	1.4000				
5	6.8000	2.8000	4.8000	1.4000				
6	6.7000	3	5	1.7000				
7	6	2.9000	4.5000	1.5000				
8	5.7000	2.6000	3.5000	1				
9	4.9000	2.5000	4.5000	1.7000				
10	7.3000	2.9000	6.3000	1.8000				
11	6.7000	2.5000	5.8000	1.8000				
12								
13								
14								
15								
16								
17								
18								
19								

After that we converted rows in colums.



Then after preparing the data, we can open the Neural Network/Data Manager by typing nntool on the Command Window.

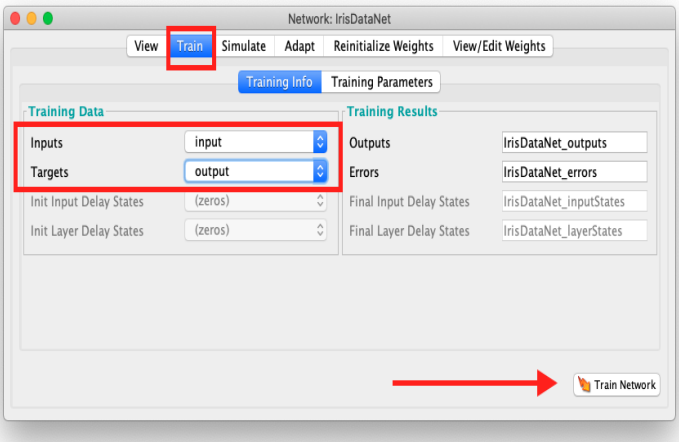


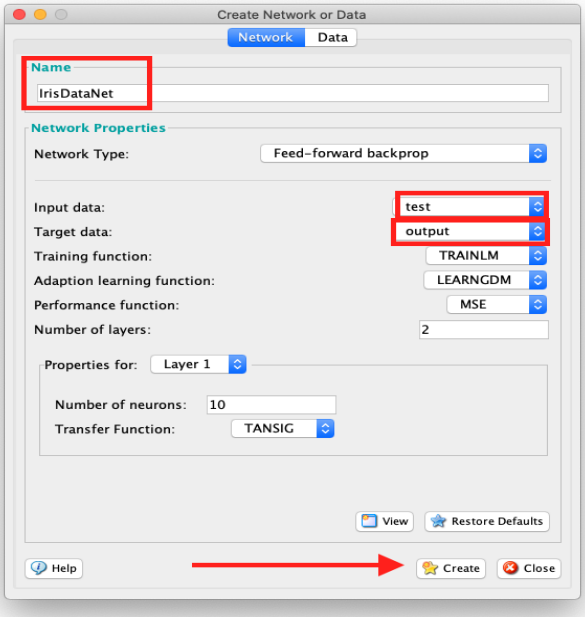


After that, We need to import our input to the input data and output to the target data.

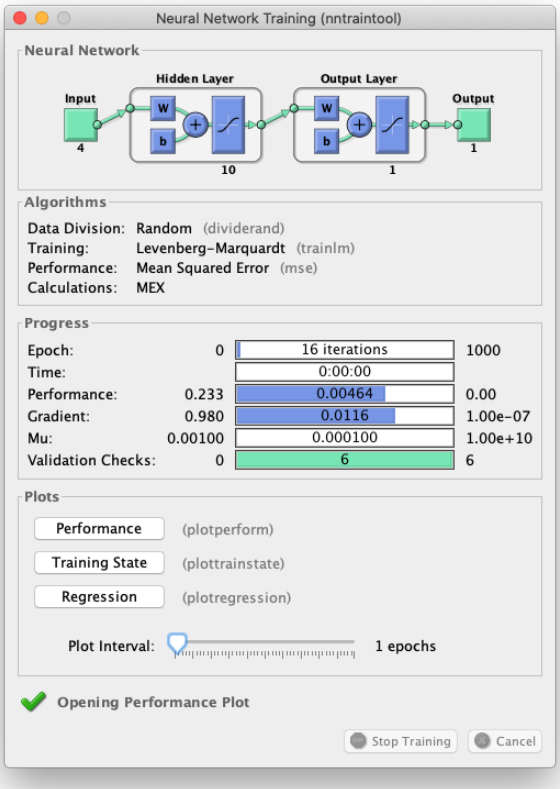
Now we are able to create a new Network. We choose our input and output data and network name. Then click Create.

Then we chose our Network and clicked Open. On the train section we chose our input and output variables and click Train Network.

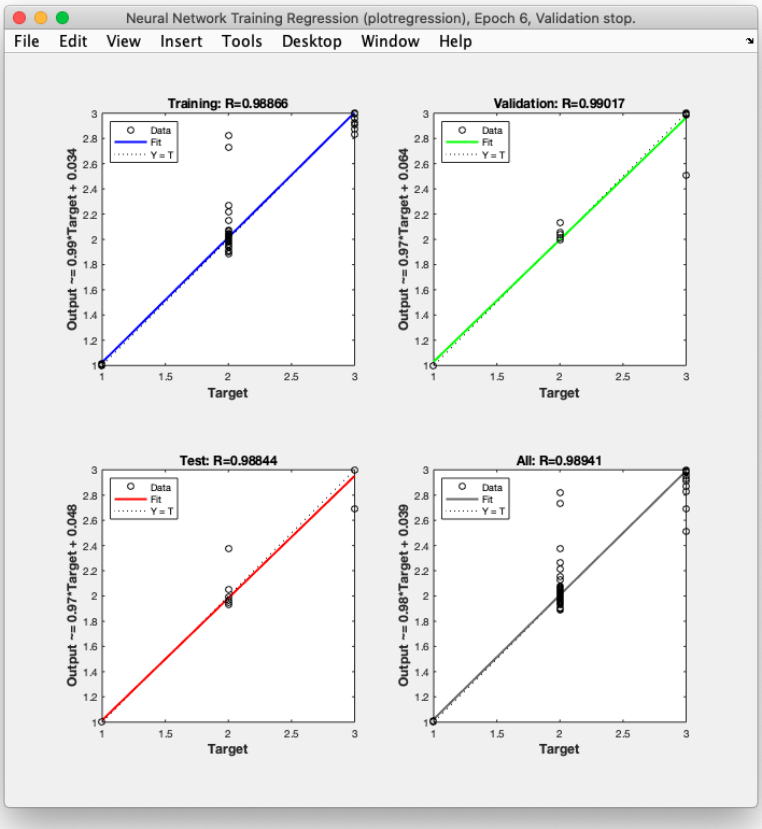




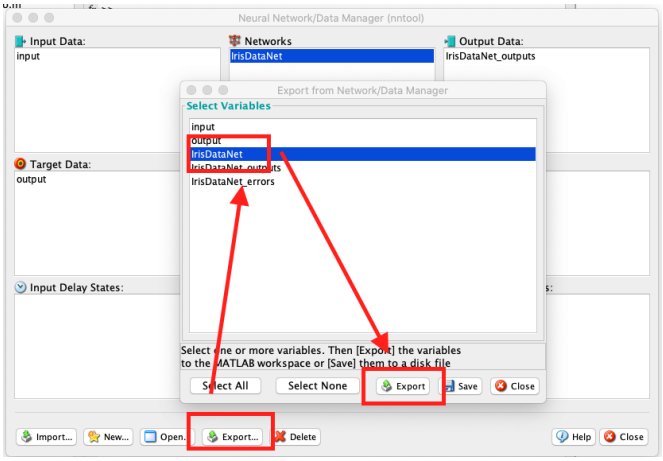
After clicking that option this window will show up.



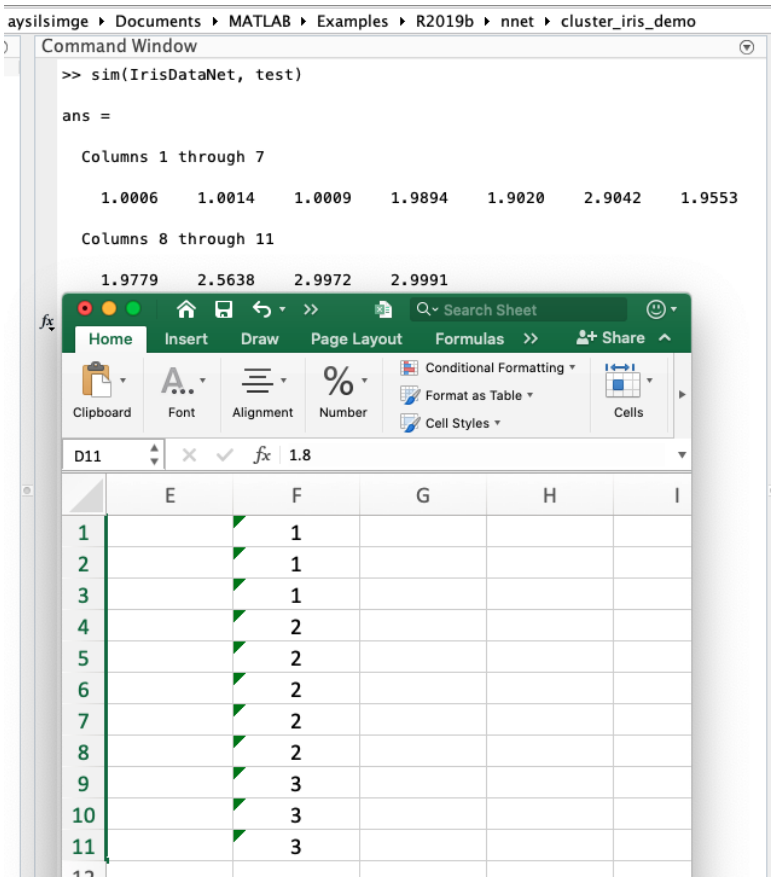
We keep training the data and checking the regression until the regression line is from the left corner to upper right corner.



Now we will export the IrisDataNet Network. We click the Export… button. Then we choose the network and click export. Then we can see our network in the Workspace.



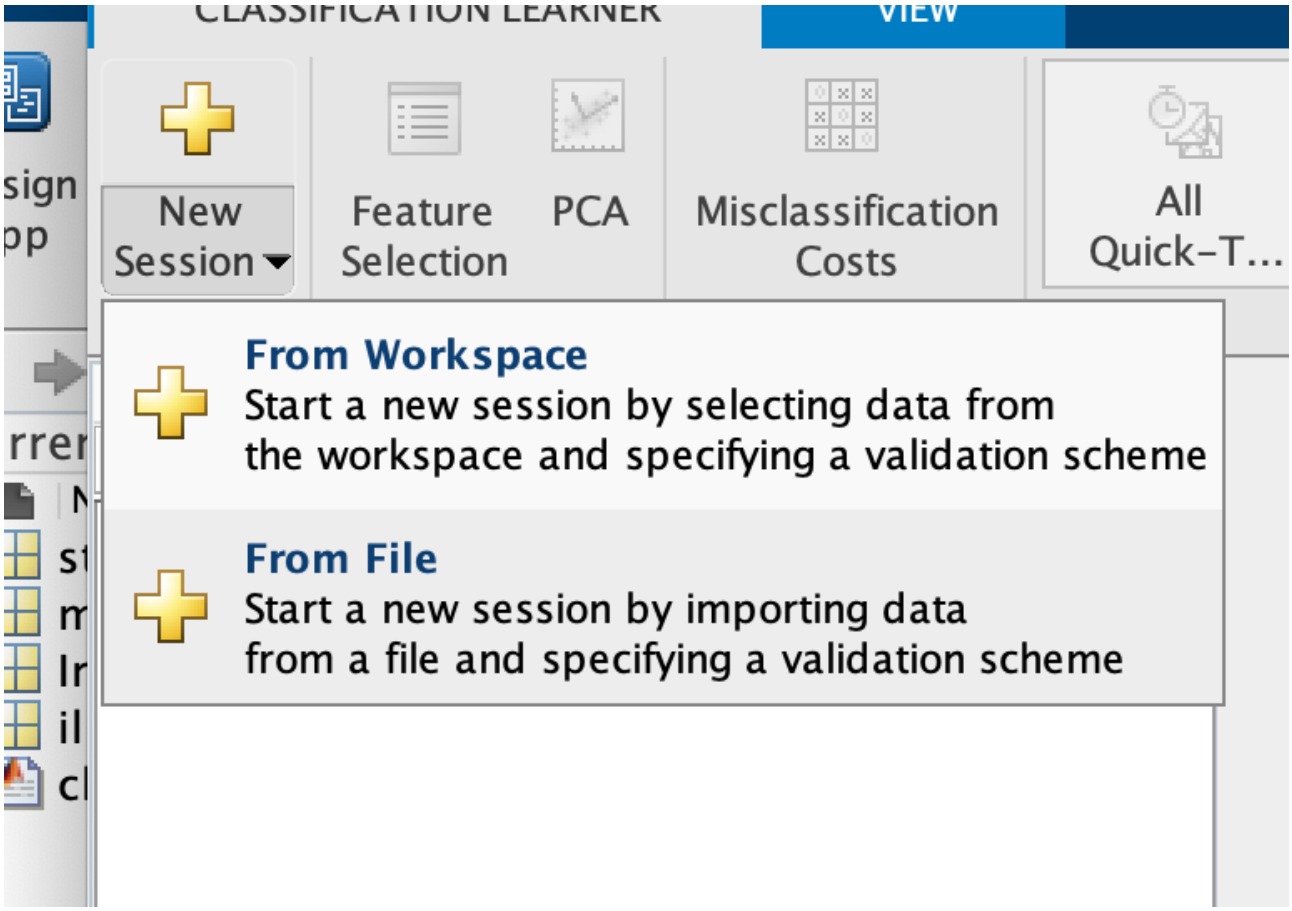
Now we are able to test the data. For testing the data we enter this command: sim(IrisDataNet, test) and we expect the results will be similar to the real values.



You can reach our matlab file from this source.[v]

Training Wine Data With Using Classification Learner Toolbox

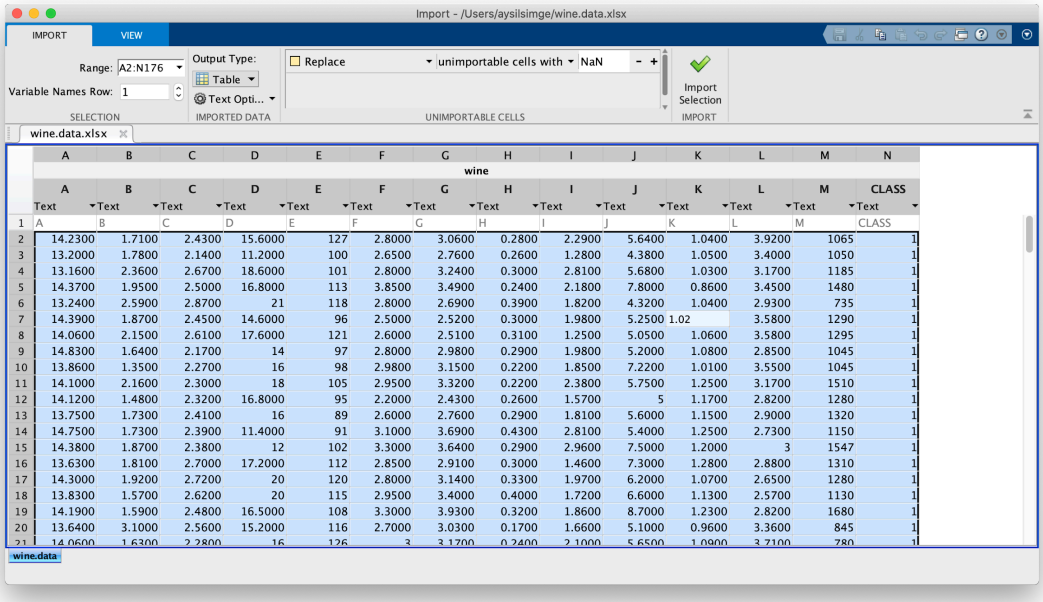
Firstly, we needed to prepare datasets (This is a very similar step with the last section). Then we openned Classification Learner Toolbox and created New Session in order to import our data.

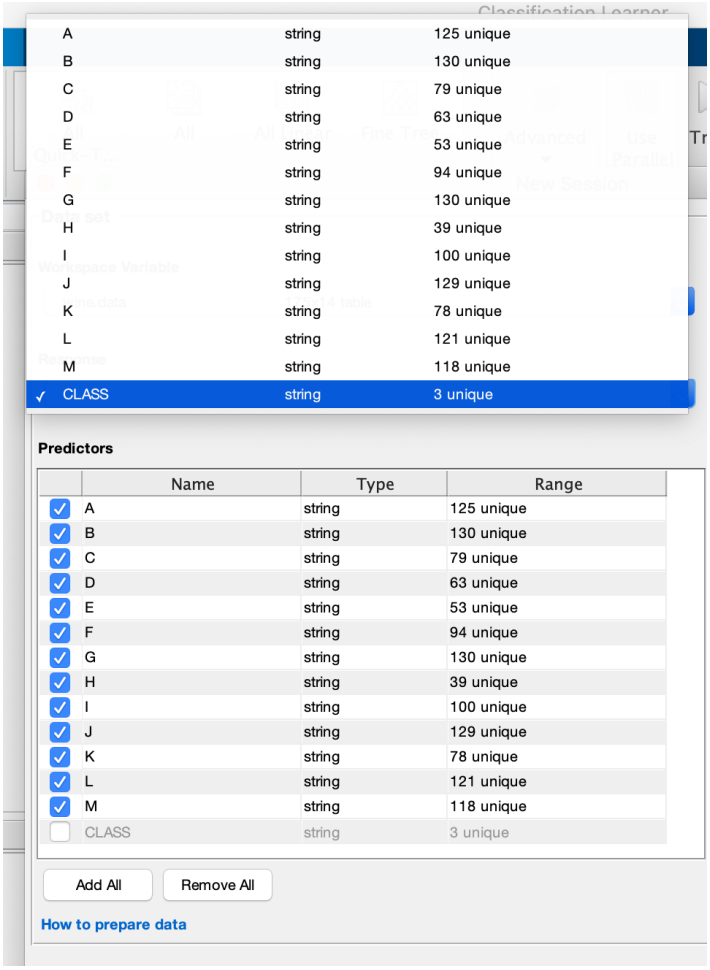


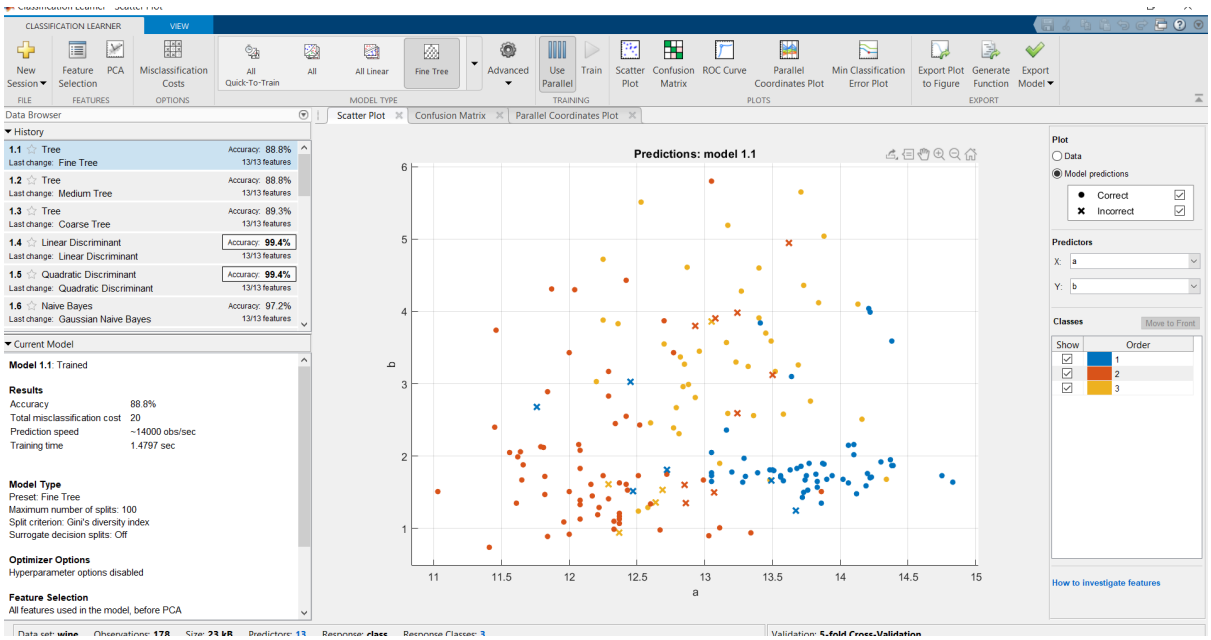
After choosing the file we selected all data and clicked Import Selection.

We made sure that our output (response) is chosen.

After that we chose our Model Type as All and after that we trained our data.





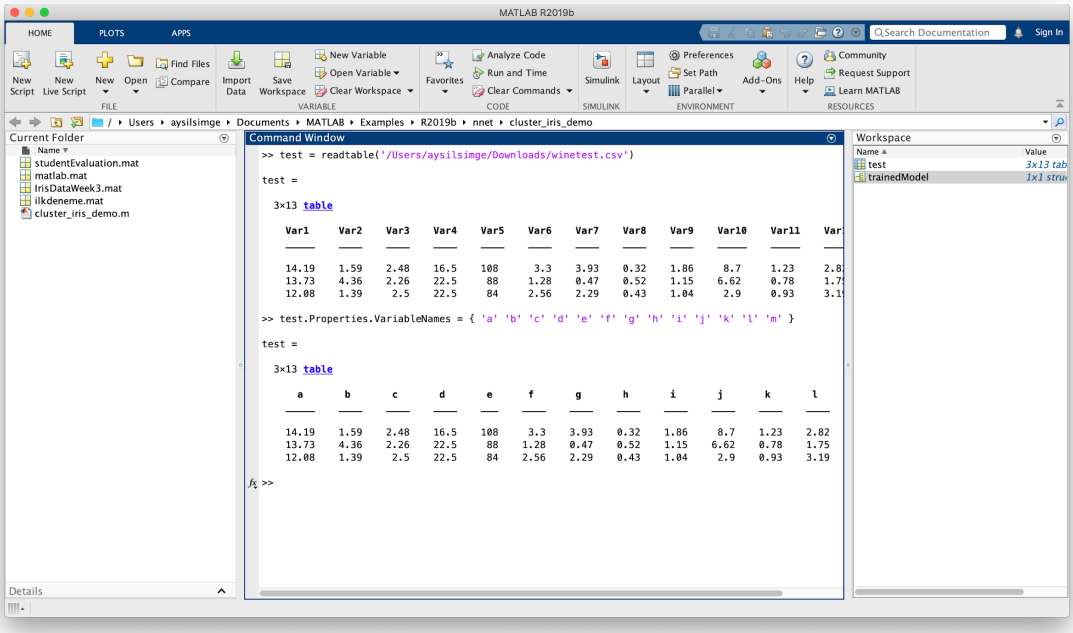


After training our data we can Export Model.

After exporting our model we will import our test data by using commands:

test = readtable('Directory of the document')

And we will change properties name same with the trainedModel's properties by using the command:



test.Properties.VariableNames = { 'a' 'b' 'c' 'd' 'e' 'f' 'g' 'h' 'i' 'j' 'k' 'l' 'm' }

After that we will test our model with the command:

yfit = trainedModel.predictFcn(test)

And the output is:

```
>> yfit = trainedModel.predictFcn(test)

yfit =

     2
     3
     1
```

Matlab file of this section is soruced with data files.[vi]

Training Seeds Data With LIBSVM

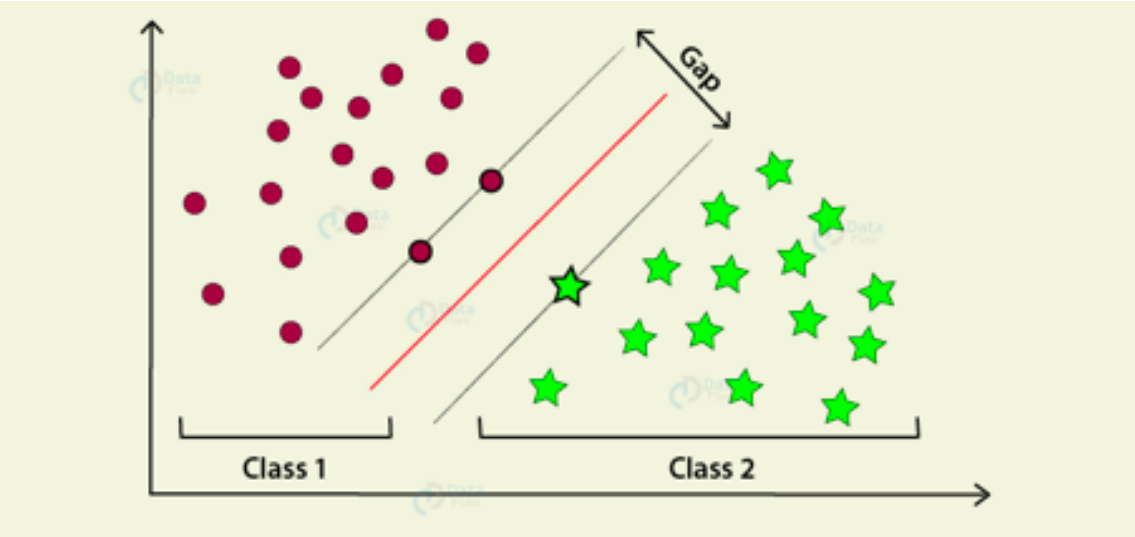
SEEDS DATA SET

Seeds Data Set has 210 instances and 7 attributes, and it is created for detecting three different varieties of wheat.

Seeds Data Set is sourced by UCI Repository[vii]

SVM

A Support Vector Machine (SVM) is a supervised machine learning algorithm that can be employed for both classification and regression purposes. SVMs are more commonly used in classification problems and as such, this is what we will focus on during the project.



In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the optimal hyper-plane that differentiate the two classes very well.

LIBSVM

SVM method generally used in datas which have 2 classes. Also, we can classify data which has more than two classes but we a different library called as LIBSVM which is a integrated software for support vector classification supports multiple classification. And I implement that library to my project. Then I trained seeds data with LIBSVM method.

I will train a data set to obtain a model using LIBSVM method and then I will predict information of testing data set with using the model.

Training Data With LIBSVM

I need to separate my data into two different files. In Features file will contain all 7 attributes and 3 labels will contain my classes. Then I need to import these files into MATLAB.

First we need to prepare values as libsvm method can understand. For this, we need to convert our features array to a different matrix type which is sparse and then we are using a method libsvmwrite to write our labels and features into a file and with the same way we will use a dirreferent method called libsvmread to read the file we have created.

I will separate the training and testing data as approximately %20 for testing and %80 for training.

Data Preperation

```
fs = sparse(features)

libsvmwrite('svm', labels, fs);

[y,x] = libsvmread('svm');

trainlabel = y(1:170,:);

traindata = x(1:170,:);

testdata = x(171:210,:);

testlabel = y(171:210,:);
```

Now we can train the data. Here we are creating our model with our trainlabel and traindata.

Data Training

```
model = svmtrain(trainlabel, traindata, '-s 0 -t 1 -d 2 -g 2.8 -b 1');
```

```
.................................................*.............*
optimization finished, #iter = 5676
nu = 0.000433
obj = -0.019703, rho = -11.179568
nSV = 7, nBSV = 0
Total nSV = 7
```

-s svm type: C-SVC

-t kernel type: Polynomial

-d degree

-g gamma

-b is for to make calculations for the accuracy rate

Then I will use svmpredict function with our model to test the data with using our test datas.

Testing the Data

```
[predicted_labeltest] = svmpredict(testlabel, testdata, model, '-q');
```

We can draw a confusion matrix and to find our accuracy rate we use a basic equation if we divide correct predicted data with total testing data and multiply with 100, we find the accuracy rate.

$$\text{Accuracy} = \frac{\text{Correctly Predicted Data}}{\text{Total Testing Data}} 100$$

```
rate = 95
```

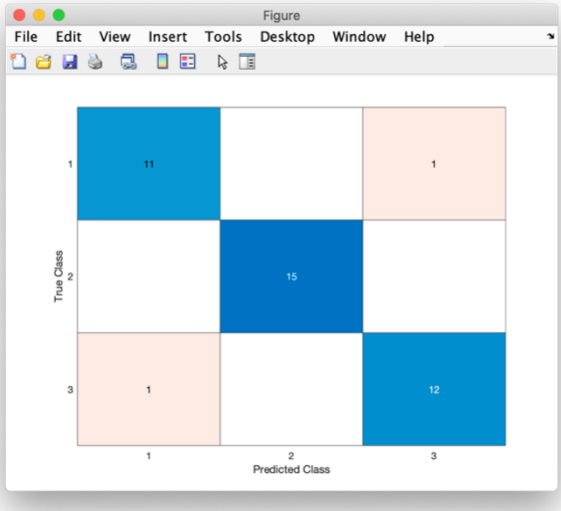
```
wrongPredictionsArr = testlabel == predicted_labeltest;
wrongPredictions = sum(wrongPredictionsArr(:) == 0);
noe = numel(testlabel);
rate = ((noe-wrongPredictions)/noe)*100
```

```
%confusion mat
ConfMat = confusionmat(predicted_testlabel, testlabel_array);
confusionchart(ConfMat);
```

Result for the testing data;

```
predicted_labeltest =

     2
     1
     2
     2
     2
     2
     3
     1
     1
     3
     3
     3
     2
     1
     1
     1
     2
     2
     1
     3
     2
     3
     2
     1
     2
```

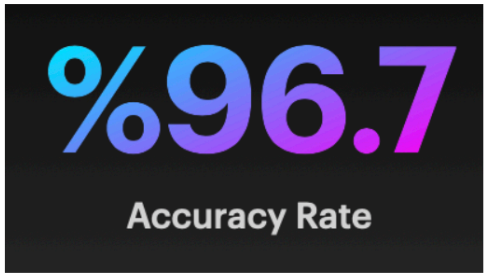



```
testlabel =

     2
     1     2
     2     1
     2     2
     2     3
     2     2
     3     3
     1     2
     1     1
     3     2
     3     3
     3     1
     2     3
     1     3
     1     3
     1     3
     2     3
     2     2
     1     1
     1
     2
     3
```

The accuracy rate is 95% for testing data set.

For all data the accuracy rate and confusion matrix,

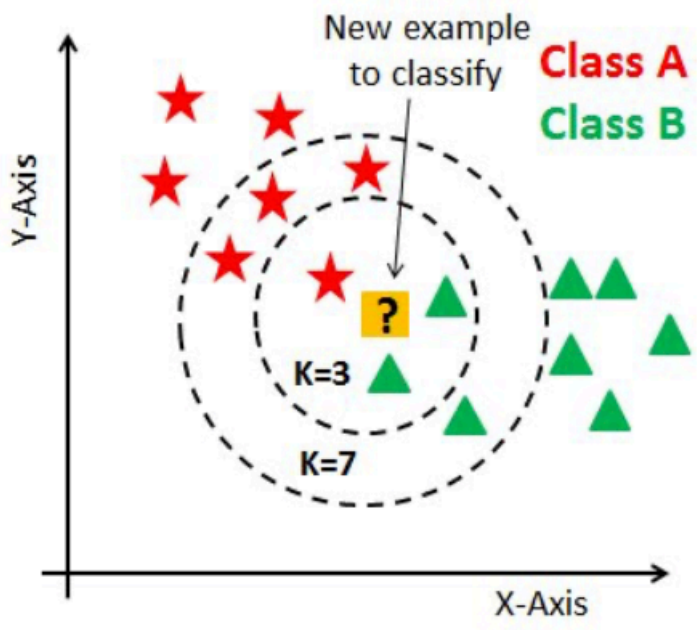


You can reach the matlab file of this section from sources.[viii]

Training Seeds Data With KNN

KNN

KNN (K Nearest Neighbor) is an algorithm that stores all the available conditions and classifies the new data based on a similarity measure. It is generally used to classify a data point dependent on how its neighbors are classified.



Training Data With KNN

Since we will be comparing SVM and KNN methods. Testing and training data sets same with the SVM method. We use approximately %20 of all data for testing and other %80 goes for training which is same with the SVM method.

Data Preparation

```
trainlabel = labels(1:170,:);

traindata = features(1:170,:);

testdata = features(171:210,:);

testlabel = labels(171:210,:);

% Converting the testlabel and labels to an array
% to compare with the predicted results

labels_array = table2array(labels);
testlabel_array = table2array(testlabel);
```

While training the data we will be using fitcknn() function and options given in the figure.

Training

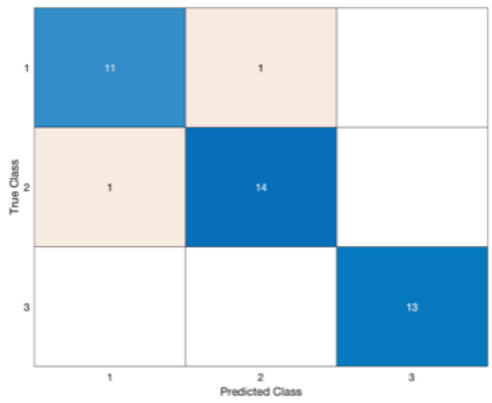
```
%model = fitcknn(traindata, trainlabel)

model = fitcknn(...
    traindata, ...
    trainlabel, ...
    'Distance', 'Euclidean', ...
    'Exponent', [], ...
    'NumNeighbors', 3, ...
    'DistanceWeight', 'Equal', ...
   'Standardize', true, ...
    'ClassNames', [1; 2; 3])
```

The distance equation I used is Euclidean and K is set to 3.

After training the data we will test the data check the accuracy rates and confusion matrix.

Results for the test data;

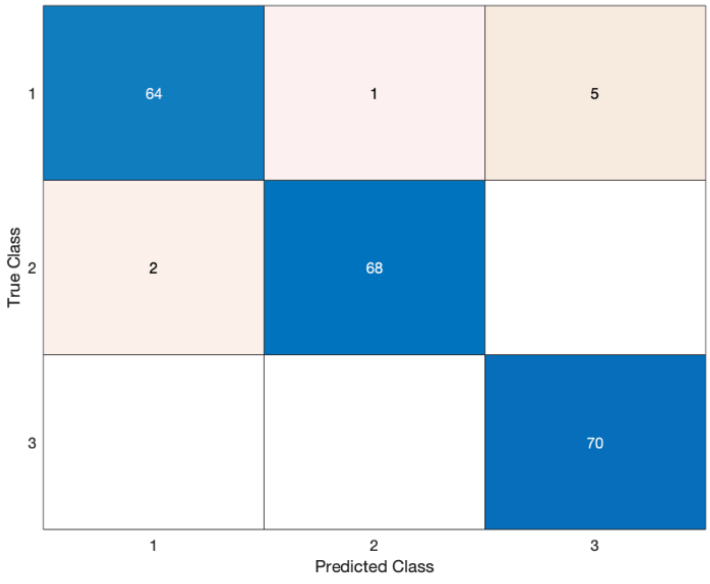



```
wrongPredictions = 2
```

Result for the all data;

```
wrongPredictions = 2
```

```
accuracyRate = 95
```



Picking a value for 'K'

A low value of k leads to a very sensitive model, while k is increasing classifier might be less sensitive and finding the k closest point might be computationally expensive. Also, larger values of k reduce the effect of noise on the classification, but make boundaries between classes less distinct. I chose K as 3 in the comparison with the SVM method.

Here you can see the effect of K on the accuracy rates.

K	Accuracy Rate
1	99.5
2	96.2
3	96.2
4	95.7
5	93.8
6	93.3
7	94.3
8	93.3
9	91.4
10	91.9
11	92.4
12	92.4
13	92.4
14	93.3

You can reach the matlab file of this section from sources.[ix]

Comparing SVM And KNN Methods

Methods are compared with some equations. Accuracy rate is already mentioned in the Training Seeds Data with LIBSVM section.

1. Precision: Precision is shows us how precise or accurate the model is, how many of them are actual positive out of the predicted positive values.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$= \frac{True\ Positive}{Total\ Predicted\ Positive}$$

2. Recall: Recall actually calculates how many of the Actual Positives our model capture through labeling it as Positive (True Positive). Applying the same understanding, we know that Recall shall be the model metric we use to select our best model when there is a high cost associated with False Negative.[x]

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$= \frac{True\ Positive}{Total\ Actual\ Positive}$$

3. F-1 Score: F-1 Score shows us the balance with Precision and Recall results. F1 Score might be a better measure to use if we need to seek a balance between Precision and Recall AND there is an uneven class distribution (large number of Actual Negatives). [xi]

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

4. Cohen's Kappa Score: Kappa score measures the degree of agreement between the true values and the predicted values which I used as the classifier's performance.[xii]

- 0.01 – 0.20 slight agreement
- 0.21 – 0.40 fair agreement
- 0.41 – 0.60 moderate agreement
- 0.61 – 0.80 substantial agreement
- 0.81 – 1.00 almost perfect or perfect agreement

RESULTS FOR KNN

I calculated Precision, Recall and F1 Score in MATLAB. For Kappa Score I used a calculator.[xiii]

	A	B	C	Total
A	64	1	5	**70**
B	2	68	0	**70**
C	0	0	70	**70**
Total	**66**	**69**	**75**	**210**

```
%Precision, Recall, F-1 Score

%Precision
for i =1:size(ConfMat,1)
    precision(i)=ConfMat(i,i)/sum(ConfMat(i,:));
end
precision
macro_precision = mean(precision)

%Recall

for i =1:size(ConfMat,1)
    recall(i)=ConfMat(i,i)/sum(ConfMat(:,i));
end
recall
macro_recall = mean(recall)

%F-1 Score

for i =1:3
    f1score(i) = 2 * precision(i) * recall(i) / (precision(i) + recall(i));
end
f1score
macro_f1score = mean(f1score)
```

```
precision = 1×3
    0.9143    0.9714    1.0000
macro_precision = 0.9619
```

```
recall = 1×3
    0.9697    0.9855    0.9333
macro_recall = 0.9628
```

```
f1score = 1×3
    0.9412    0.9784    0.9655
macro_f1score = 0.9617
```

Kappa= 0.943

RESULTS FOR SVM

	A	B	C	Total
A	64	0	6	**70**
B	0	70	0	**70**
C	1	0	69	**70**
Total	**65**	**70**	**75**	**210**

```
%Precision, Recall, F-1 Score

%Precision
for i =1:size(ConfMat,1)
    precision(i)=ConfMat(i,i)/sum(ConfMat(i,:));
end
precision
macro_precision = mean(precision)

%Recall

for i =1:size(ConfMat,1)
    recall(i)=ConfMat(i,i)/sum(ConfMat(:,i));
end
recall
macro_recall = mean(recall)

%F-1 Score

for i =1:3
    f1score(i) = 2 * precision(i) * recall(i) / (precision(i) + recall(i));
end
f1score
macro_f1score = mean(f1score)
```

```
precision = 1×3
    0.9143    1.0000    0.9857
macro_precision = 0.9667
```

```
recall = 1×3
    0.9846    1.0000    0.9200
macro_recall = 0.9682
```

```
f1score = 1×3
    0.9481    1.0000    0.9517
macro_f1score = 0.9666
```

Kappa= 0.950

KNN AND SVM

	KNN	***SVM***
Accuracy Rate	*96.19*	*96.67*
Macro-Precision	*0.9619*	*0.9667*
Macro-Recall	*0.9628*	*0.9682*
Macro-F1 Score	*0.9617*	*0.9666*
Cohen's Kappa Score	*0.943*	*0.950*

Future Works

In future works, In SVM I can try to change the Kernel Functions and other parameters, also In KNN I can change the calculation metric other than Euclidean. Also, data can be trained with other classification algorithms other than SVM and KNN, in addition to that, comparison can be done with other classifications. Finally, Visualization on classification algorithms and data training can be increased with using different variety of graphs and tables.

Conclusions

During this project we (as a team) tried to understand the MATLAB toolboxes, neural networks and classification algorithms. Then we tried to learn the toolboxes, so we trained the iris data with NNTool. Then, for our first prototype we chose wine data set and trained it with using Classification Learning Toolbox. For the second prototype my data set was seeds data set and I tried to train it with LIBSVM algorithm (not with a toolbox) and understand the SVM method. After the second prototype I tried to understand KNN algorithm and trained the seeds data set with KNN. Finally, I compared the SVM and KNN Algorithms which I trained.

RESOURCES

1. https://www.youtube.com/watch?v=Nu7pGuenbwo
2. https://www.mathworks.com/help/matlab/
3. https://www.csie.ntu.edu.tw/~cjlin/libsvm/
4. https://www.mathworks.com/help/stats/classificationknn.html
5. https://www.youtube.com/watch?v=-49ig_-ZnFc
6. https://medium.com/@sddkal/cpp-ile-libsvm-destek-vektör-makineleri-kütüphanesi-7919f6c57f44
7. https://towardsdatascience.com/kernel-function-6f1d2be6091
8. https://www.youtube.com/channel/UCFJPdVHPZOYhSyxmX_C_Pew
9. https://archive.ics.uci.edu/ml/index.php

---

[i] https://en.wikipedia.org/wiki/MATLAB

[ii] https://deepai.org/machine-learning-glossary-and-terms/multilayer-perceptron

[iii] https://www.educative.io/edpresso/what-is-a-multi-layered-perceptron

[iv] https://archive.ics.uci.edu/ml/datasets/Iris

[v] https://github.com/aysilsimgekaracan/Matlab_Project/blob/master/IrisData/IrisDataWeek3.mat

[vi] https://github.com/aysilsimgekaracan/Matlab_Project/tree/master/Wine%20Data%20Set

[vii] https://archive.ics.uci.edu/ml/datasets/seeds

[viii] https://github.com/aysilsimgekaracan/Matlab_Project/tree/master/LIBSVM

[ix] https://github.com/aysilsimgekaracan/Matlab_Project/tree/master/KNN

[x] https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

[xi] https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

[xii] https://towardsdatascience.com/multi-class-metrics-made-simple-the-kappa-score-aka-cohens-kappa-coefficient-bdea137af09c

[xiii] https://www.graphpad.com/quickcalcs/kappa1/