Computational Intelligence- CA

Hybrid Neural Networks

JAWAD [A0119964R]
GOKUL KRISHNAA [A0120066W]

Contents

Part A: Diabetes Prediction

Problem description

Given the Diabetes dataset our aim is to train a neural network to classify a patient as Diabetic/Non Diabetic based on the attributes presented. This would be a classification problem, where if the network predicts the patient to be Diabetic it assigns the output as one, else zero.

Data description

Following are the attributes presented.

1. Number of times pregnant
2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index (weight in kg/ (height in m) ^2)
7. Diabetes pedigree function
8. Age (years)
9. Class variable (0 or 1)

Total Instances	768
Class 0	500
Class 1	268

Although we can see the disparity in the numbers of Class '0' and '1' it is still an acceptable limit. Hence we will not be altering the proportion of data that will be fed to the Neural networks. Additionally the data quality in terms of outliers and skewness is within acceptable range. There are no missing values, the data is clean and hence no pre-processing will be required.

Neural network selection

Since it is a classification problem with supervised learning, the following neural networks are selected to train with the data.

1. Multi-layer feed forward- with Back propagation(MLFF-BP)
2. Probabilistic neural network

Analysis 1 (MLFF-BP)

Neuroph

MLFF -BP stands for Multi-layer feed forward network with error back propagation, minimum of one hidden layer is used in this type of network, since the Delta rule cannot be applied to correct the error , due to the presence of more than one hidden layer a back propagation algorithm is applied, this has been implemented in R and Neuroph.

Data Partitioning:

The Diabetes.csv file was sampled and portioned into 3 sets of (70/15/15) for training, cross-validation and testing using R. To test the effectiveness of normalization of data, a similar partition was constructed and the inputs were normalized.

Training algorithms were run with default Neuroph values for the following parameters,
Max error: 0.01
Learning rate: 0.2

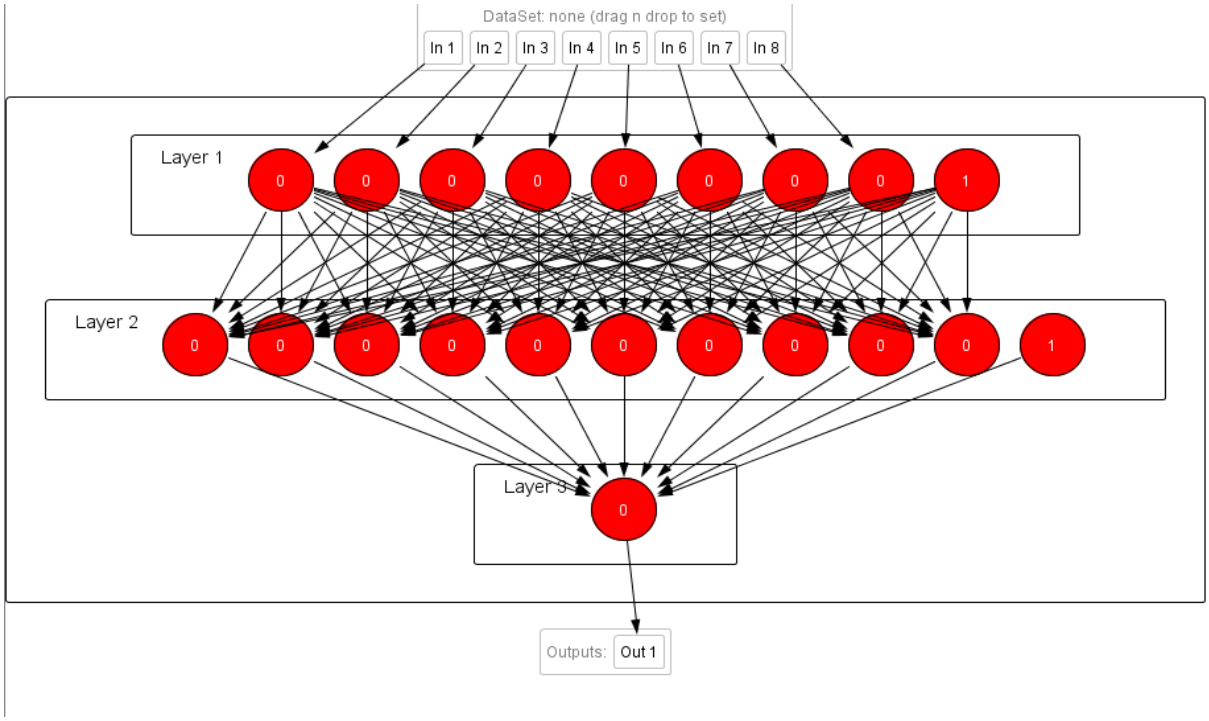


Table 1

S.no	Hidden nodes	Normalized	Algorithm	Error propagation	MSE train	MSE CV	NW error
1	3	No	MLFFBP	BP	0.22893836	0.214183685	0.12866784
2	10	No	MLFFBP	BP	0.228461034	0.214522045	0.12053591
3	10*2	No	MLFFBP	BP	0.228083375	0.215827744	0.123773882
4	3	Yes	MLFFBP	BP	0.134931431	0.185578994	0.060454375
5	10	Yes	MLFFBP	BP	0.197186149	0.179633302	0.105063219
6	10*2	Yes	MLFFBP	BP	0.074130459	0.240514266	0.03
7	10	Yes	MLFF	Resilient BP	0.087286506	0.24474098	0.043686085
8	7	Yes	MLFF	Resilient BP	0.113883183	0.230957742	0.055457546

Observation:

1. From Table 1, it can be seen that the MSE-CV (Mean square Error for Cross Validation) data set results in better prediction when the neural network is trained with normalized data.

2. The Error on the training data tends to decrease as we increase the number of hidden nodes in the network, but then the aspect of over fitting kicks in and the network does not generalize well with the cross validation data.

3. From the tested data, the neural network trained with 10 hidden nodes and normalized input data turns out the optimum prediction.

4. The Error propagation technique was altered from Back propagation to resilient back propagation and the neural network was trained (case 7). It can been seen that the network was trained with least error, and created the highest fit to the training data, but could not generalize well with the cross validation data.

5. Resilient back propagation is able to learn the data with minimal number of hidden nodes but when it comes to generalizing we have seen that back propagation works best.

As it can be seen that MSE is just the measure of goodness of fit for the neural network, but for a classification problem metrics such as sensitivity can only be captured through a confusion table, and Neuroph does not supply with such a provision.

MLFF-BP (R)

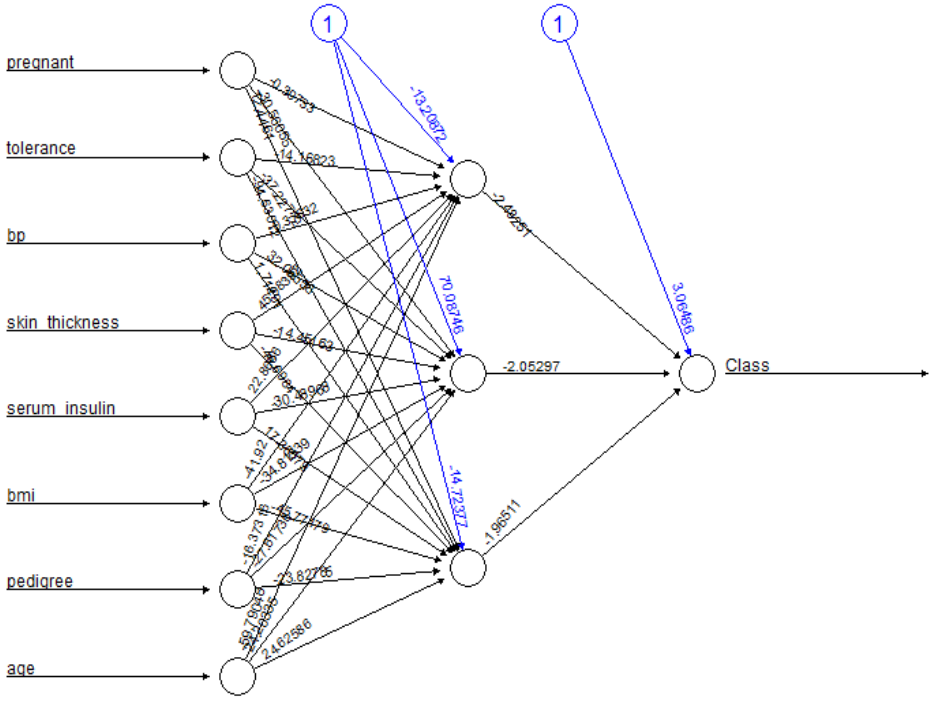
A similar analysis was conducted using "neuralnet" package in R. In this case the data was normalized and was split in the following ratio: training (60), cross validation (20), testing (20). The task in particular was to predict a value using the attribute list consisting of the normalized values of "pregnant+tolerance+bp+skin_thickness+serum_insulin+bmi+pedigree+age". The sigmoid activation function was used as activation function and cross entropy was used as the measure of error.
As stated before back propagation was used to alter the weights.

Learning stage:

1. As stated above the training data (60%) consists of 460 observations and this was feed to a neural network with a single hidden layer with 4 nodes.
2. Once we have this trained algorithm, we can evaluate the algorithms effectiveness by making predictions on the cross validation set.
3. But the result since is the range of 0-1, we need to decide an optimal threshold value.
4. There are various metrics that can be used to decide the best threshold score. Since the prediction is going to be diabetic diagnosis, it is best to have a very good recall rate. But we don't want to flag every patient as diabetic.
5. So the threshold value for the best f score was calculated.
6. All the steps from 1-5 were repeated for varying configurations of the neural net architectures. The variations were produced by altering the number of nodes/layer and then by changing the layers themselves.
7. The models and architecture that performed best on the cross validation set was finally selected and tested on the test data set.

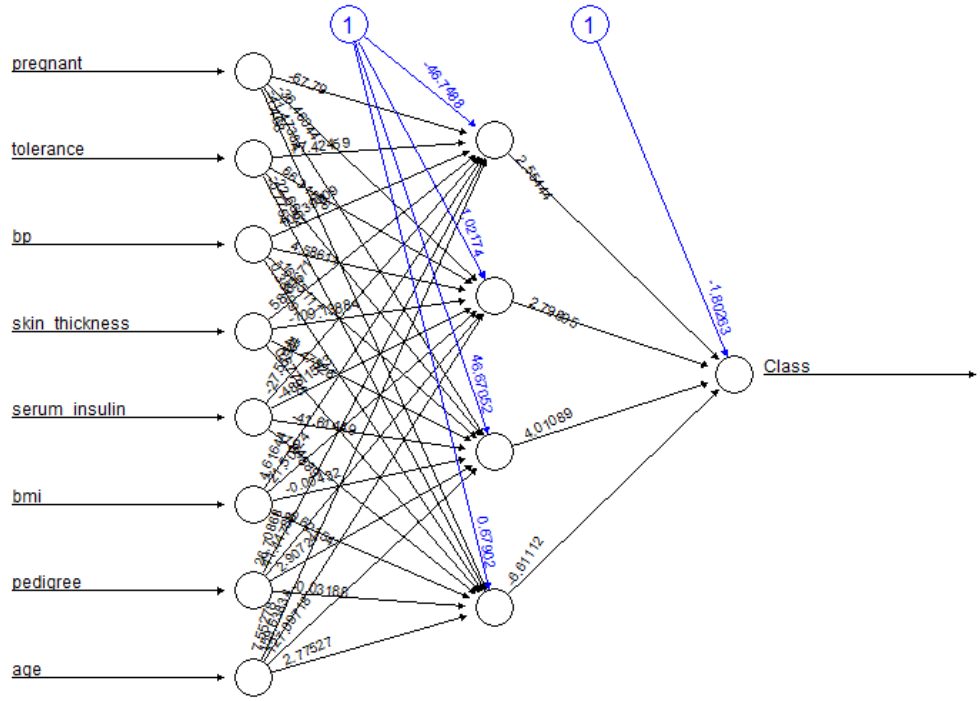
Single Layered Architectures & F Scores

1. One Layer Three Nodes – F Score 0.5170068027



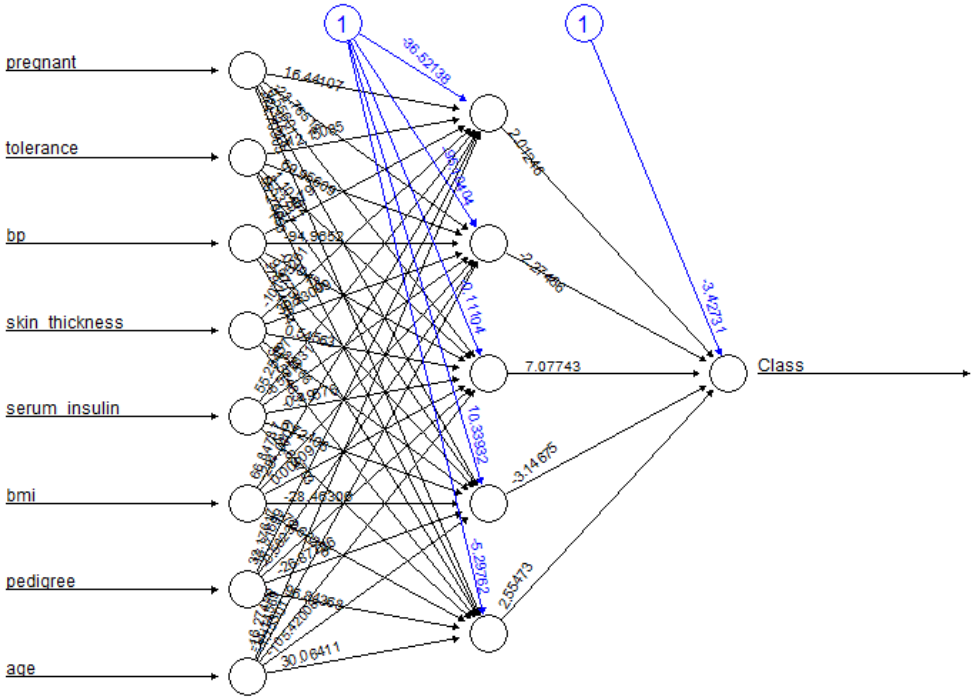
Error: 191.633412 Steps: 4370

2. One Layer Four Nodes – F Score 0. 4689655172



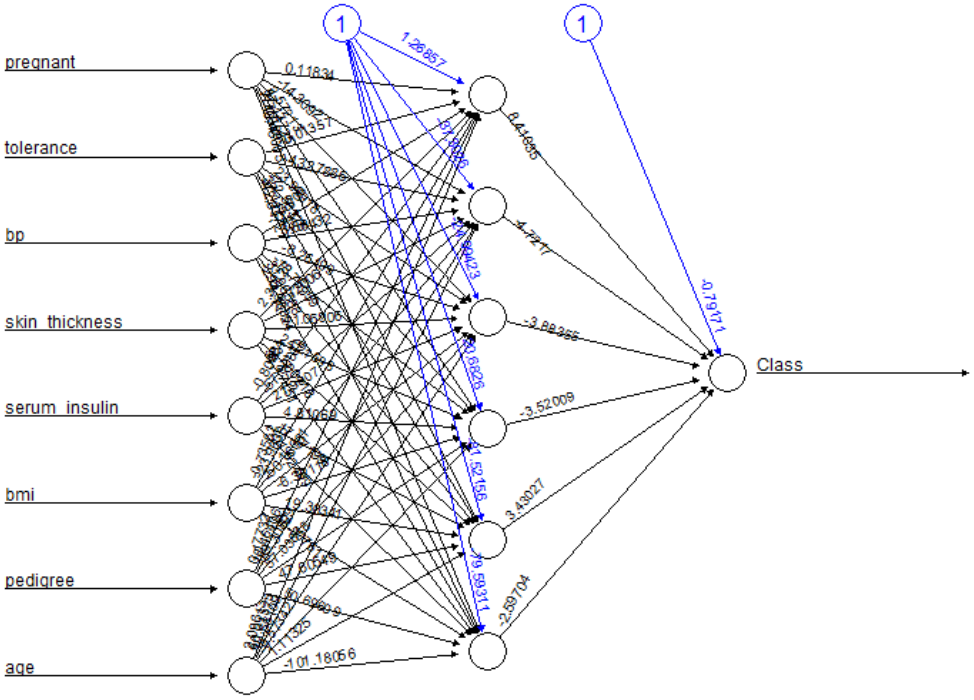
Error: 162.595673 Steps: 10076

3. One Layer Five Nodes – F Score 4931506849



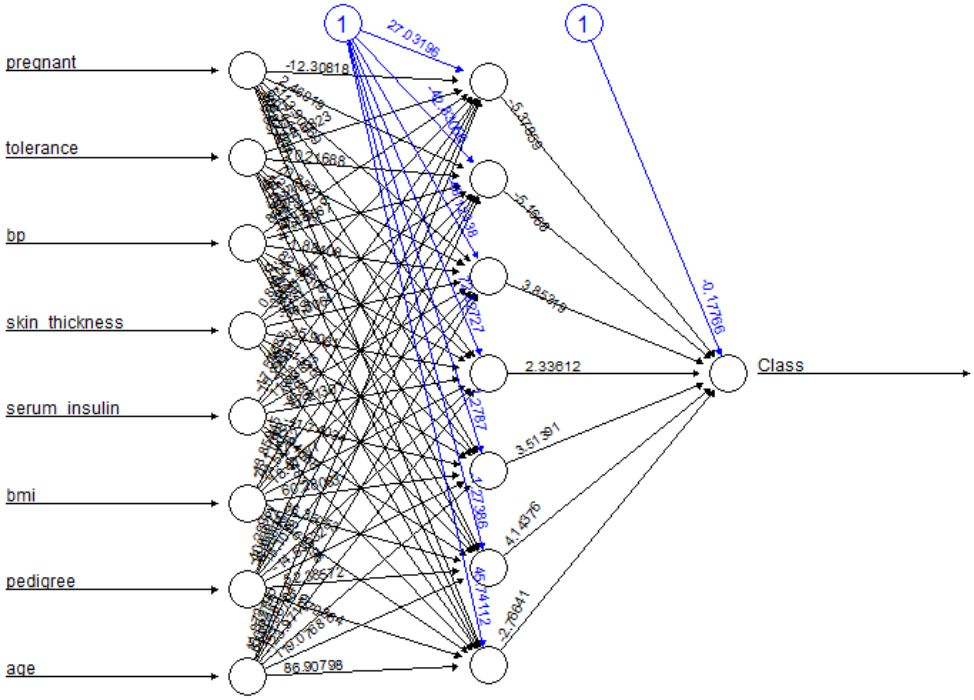
Error: 166.819933 Steps: 6858

4. One Layer Six Nodes – F Score 0. 5254237288

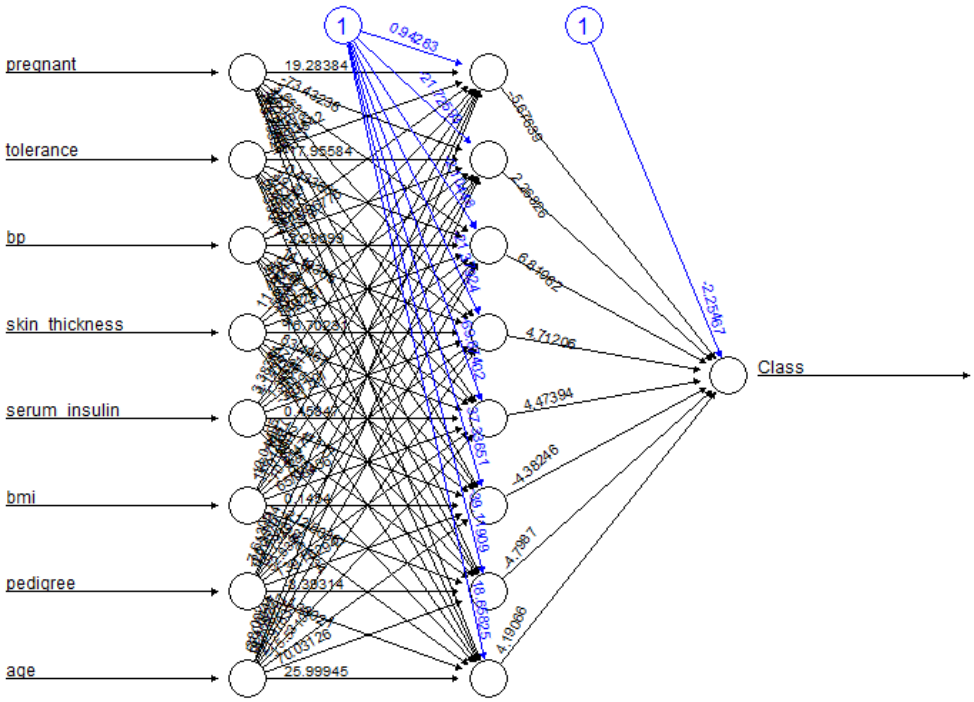


Error: 135.135947 Steps: 8683

5. One Layer Seven Nodes – F Score 0. 4531250000



6. One Layer Eight Nodes – F Score 0. 5565217391



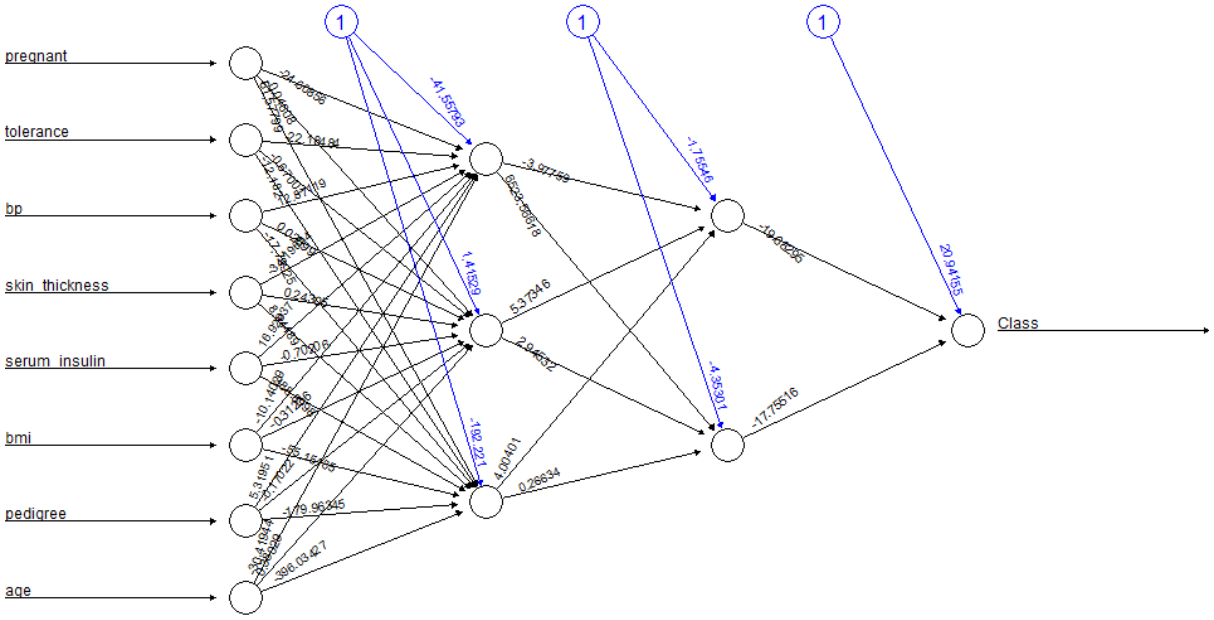
Evaluation

From the above models we can see that the best f score has been when we have 8 nodes in the hidden layer. Thus we can use that architecture on test data to see how well it generalizes. The result is tabulated as below

Confusion Matrix		Predicted	
		0	1
Actual	0	58	42
	1	14	42

Precision	0.5
Recall	0.75
f score	0.6

Variants were tried with two levels and the following results were obtained. Higher number of nodes and layers have problems converging.



Confusion Matrix		Predicted	
		0	1
Actual	0	34	66
	1	6	50

Precision	0.43103448
Recall	0.89285714
f score	0.58139535

Analysis 2 (PNN)

PNN- using R

Probabilistic Neural networks are also suitable for classification problem solving and they are quite effective in training the data in sparse data sets.

Data preparation:

Data has been sampled, normalized and split using R (training/validation/test – 75/15/15)

Script used to train the neural net:

```
pnn <- learn(data)
pnn<- smooth(pnn, sigma =8.3)
pnn$sigma
perf(pnn)
```

Observations:

1. Using PNN package neural network can be trained in R. The only parameter that can be altered using the package is the smoothing parameter, sigma.
2. If the sigma value is unknown, the package automatically searches for the appropriate value for sigma.
3. 'learn' function trains the neural network and 'smooth' function alters the decision boundary accordingly.

Table: 2

Sigma	Error rate
1	0.7318436
8.584193	0.7486034
15	0.7467412

4. As we can see from the above table, the MSE varies with sigma, but the variations itself is not significant.
5. The 'perf' function provided in the package claims to use a hold back approach to test the data trained by the neural network. But when perf function is used, PNN tries to predict the output of the entire training data. Hence an alternate approach of using the separate cross validation data is taken to test the prediction capability of the constructed neural network.
6. Below mentioned is the R code snippet for making the guess and plotting the confusion matrix

```
## Read sampled cross validation data file
CVdata <- read.csv('scCV.csv',header=T)
## Remove the target variable
WCCVdata <- CVdata[2:9]
## Calculate the predicted output
pred<-0
for (i in 1:116)
{
   pred[i] <-guess(pnn, as.matrix(WCCVdata[i,]))$category

}
print(pred)
#Plot the confusion matrix between the actual output value and the predicted value
table(CVdata[1:116,1], pred)
```

		Predicted		
		0	1	
Actual	0	73	7	0.9125
	1	18	18	
		0.8021978	0.72	0.784482759

Sensitivity
Specificity
Precision
Accuracy

7. The guess function provides the confusion matrix, using which we calculate the sensitivity of the model.

8. In a medical diagnosis sensitivity of the model should be the prime factor in deciding the usefulness.

Analysis 3 (RBF)

To try out how other techniques fare in performance, we have tested with RBF Neural network with normalized data and here is the analysis.

Nodes	Learning rate	Network error	Validation error
9	0.2	0.17287077	0.31118
16	0.2	0.169108454	0.30935
24	0.2	0.168155	0.31414
9	0.5	0.1729545	0.31117
16	0.5	0.169115	0.309217
24	0.5	0.16822	0.315276

As we can see the best MSE is received with 16 nodes and the accuracy is still far from what we receive from MLFF-BP.

Part B: Assessing Wine Quality

Data Description

The dataset contains set of features about white wines, and a quality value that indicates the quality of the wine, 0 for low and 10 for best.

Input variables (based on physicochemical tests):

1 - fixed acidity
2 - volatile acidity
3 - citric acid
4 - residual sugar
5 - chlorides
6 - free sulphur dioxide
7 - total sulphur dioxide
8 - density
9 - pH
10 - sulphates
11 – alcohol

Output variable (based on sensory data):

12 - quality (score between 0 and 10)

Regarding the quality of the data, it is said to have no missing values and there are certain outlier variables.

Neural network selection

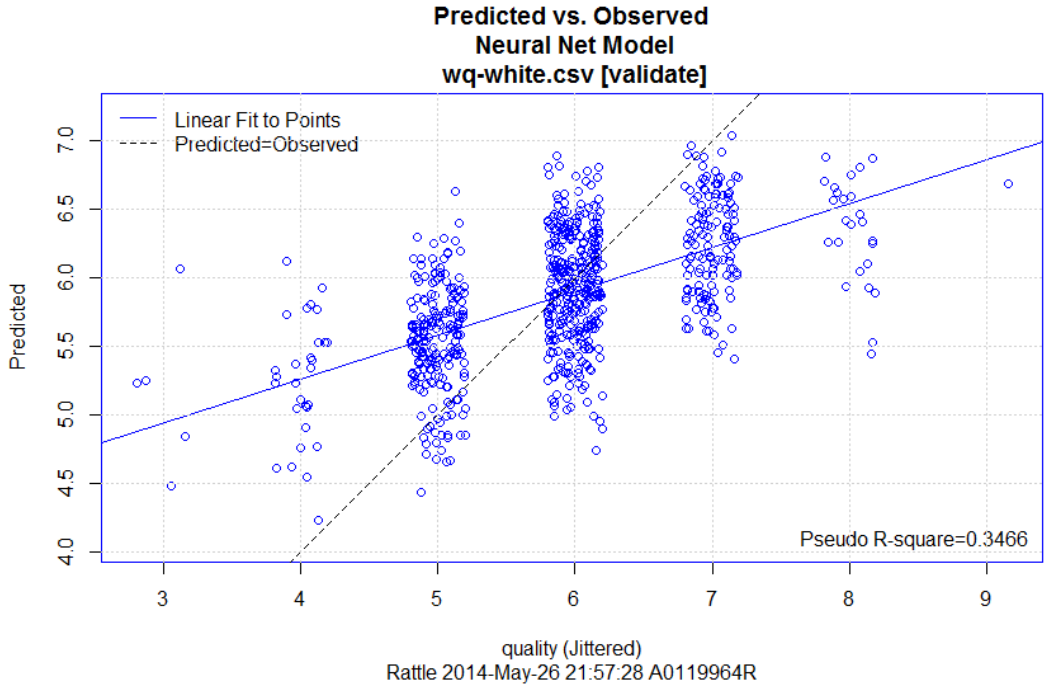
Since this is again a supervised learning, the following neural networks are selected to train with the data.

1. Multi-layer feed forward- with Back propagation(MLFF-BP)

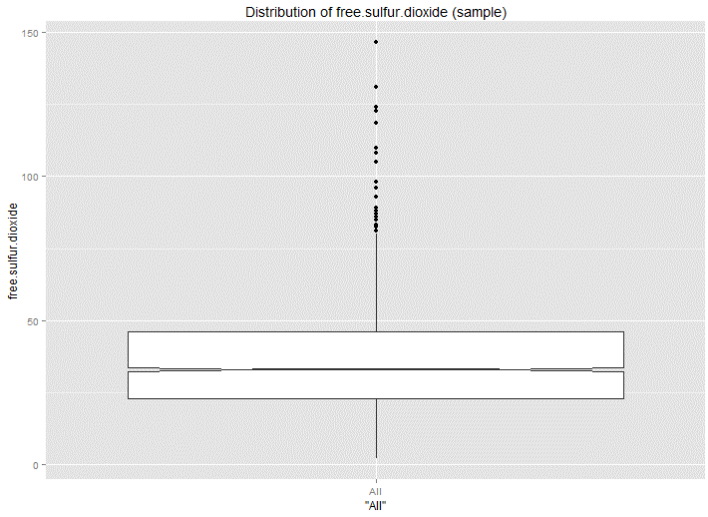
2. General Regression neural network

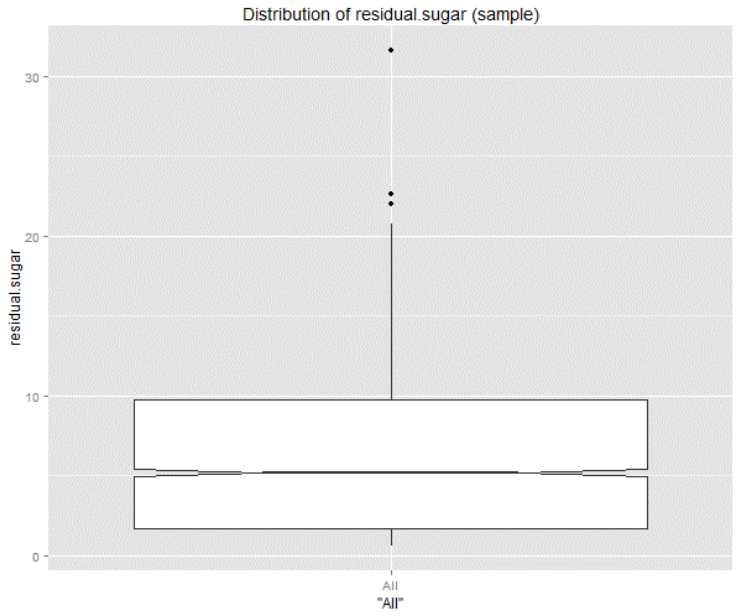
Analysis 1 (Rattle)

1. Data is prepared by splitting into training, cross-validation and testing (70/15/15).
2. Rattle GUI is implemented in R, using nnet package which is an implementation of the feed forward neural network with multinomial log linear models.
3. We commence the process of constructing neural network with a base line model to establish the performance, in this step all the data fed to neural net is the base data set (training). By doing so we train the neural network and test it on the validation data set to obtain the Pseudo-R square rate of 0.3466.



4. Since rattle internally normalizes the data, feeding normalized input to the neural net does not affect the performance of the network
5. We then check for the Outliers available in the data set and find the below mentioned variables containing outliers.

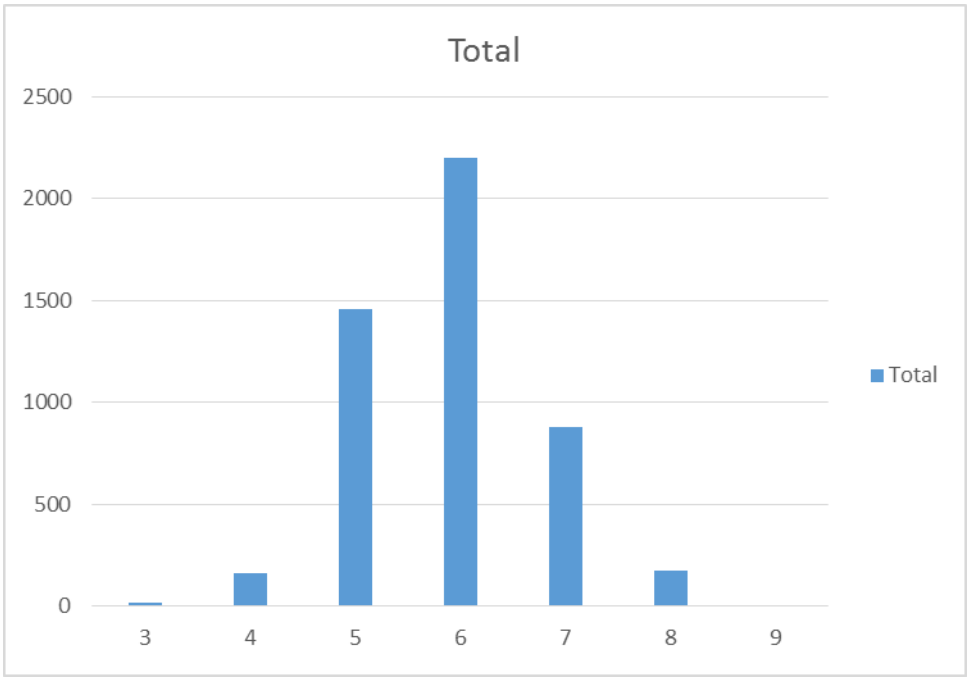




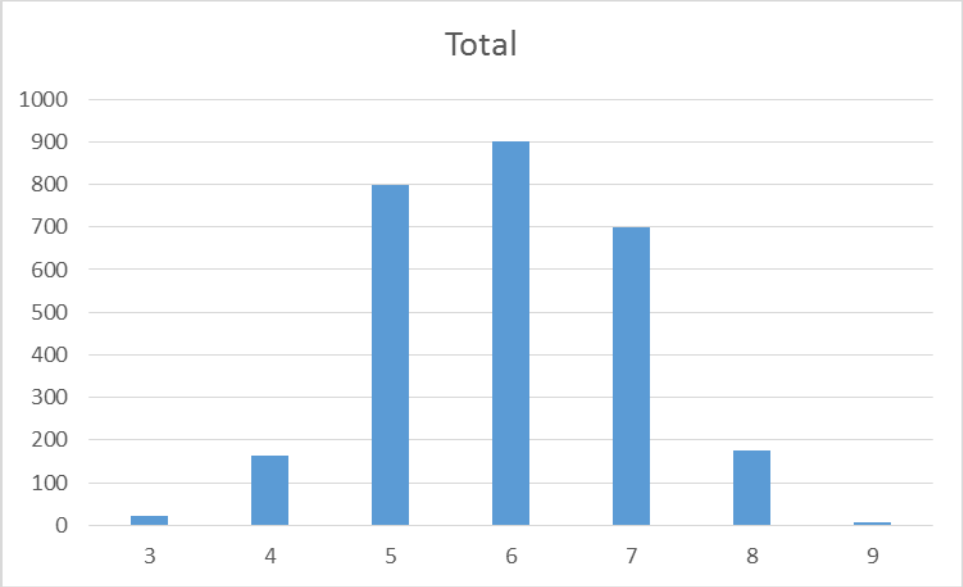
6. We see negative impact of removing the outliers from the data.

Issue in training:

7. On close analysis of the predicted output by the neural network we can see that the nn performs badly for extreme cases. This is maybe due to the fact that the given input data does not provide much information about the extreme cases and hence the network is trained for the values that dominate the training data. Hence to resolve this issue, an alternate data set was constructed with the following distribution.



Class	Frequency	Altered Frequency
3	21	21
4	164	164
5	1458	800
6	2199	900
7	881	700
8	176	176
9	6	6



Due to the lack of volume in extreme cases, we tend to reduce the number of observations of frequently occurring cases.

8. The altered dataset is fed into rattle and neural network is trained, and results obtained are provided below.

Complete Data	Hidden Layers	Pseudo-R square
	10	0.3637
	20	0.3153
	30	0.334
Conditioned Data	10	0.357
	20	0.3853
	30	0.3681

We can clearly see that the neural net is able to train better when the data is distributed well for all cases, but in the absence of sufficient data for the extreme cases we could only do as much equal distribution so as not to affect the performance of other classes.

We then derive the MSE of the trained neural network using the test data and find it to be the following.

MSE= 0.57

Analysis 2: GRNN

The General Regression Neural Network is a class of neural networks used to make predictions about real valued variables. In our case this would be the quality of the wine. The GRNN was implemented in R using a package called 'grnn'.

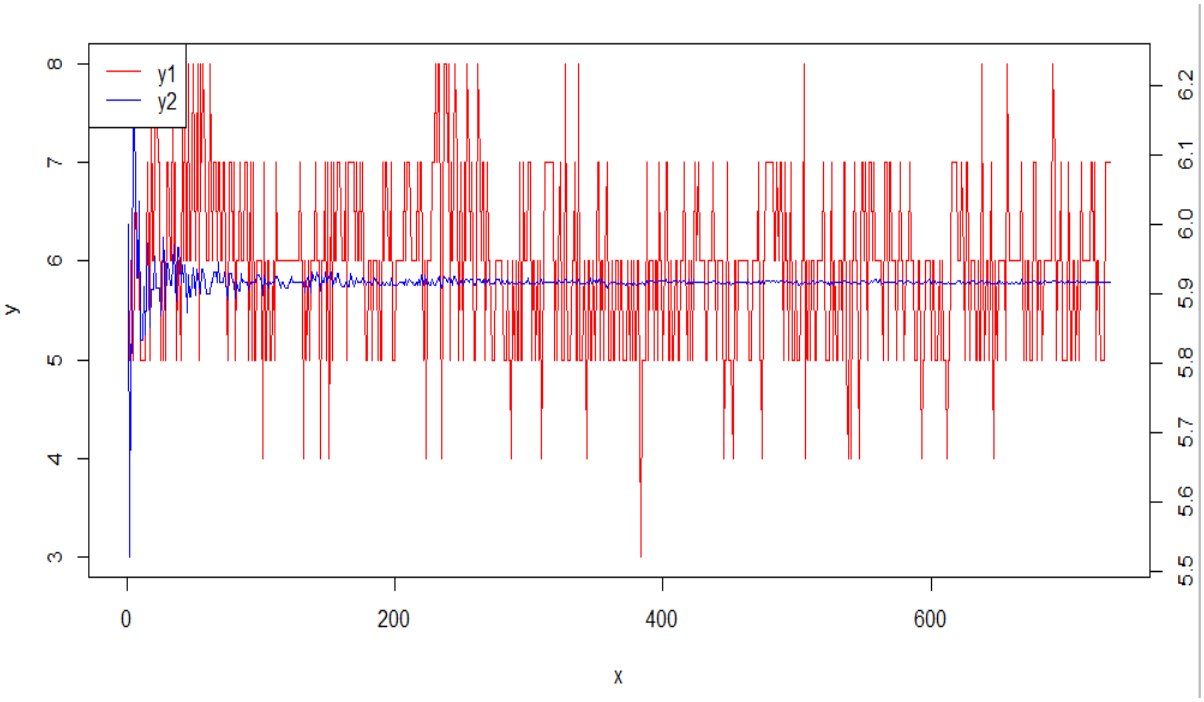
Setting Up Data:

All the input variables were scaled to have values in ranges that are comparable to the each other. Once this basic step is complete we move on to partition the data into training (60), validation (20) and testing (20).

Training the data using GRNN

The training data was feed to the GRNN. The result is a trained neural network. This trained network can be further smoothed with some smoothing parameter. The choice of the smoothing parameter greatly determines how well the network generalizes. We smoothed the network for various values of the smoothing parameter. We observed that though there were changes they were not that significant in that, the predicted result did not vary that greatly. However we used the best result, obtained when sigma was set to 0.1, and then tested on the target variable.

The following are the results:



Y1<- Actual values

Y2<- Predicted values.

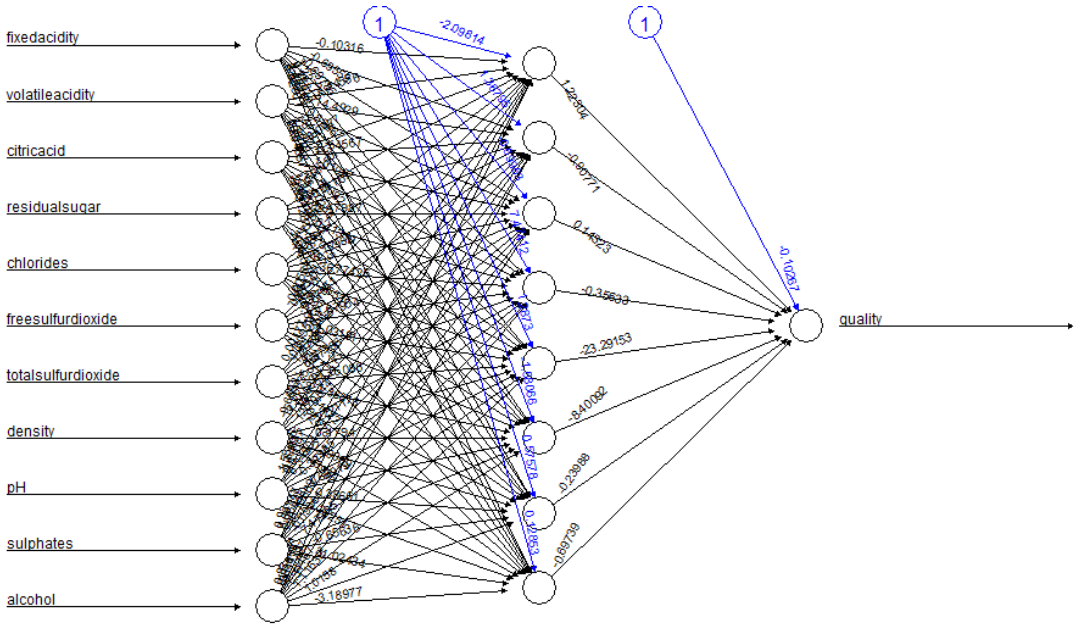
As we can see the grnn is not all that accurate. The ordinary mean square error for the grnn is about 0.4. The OMS should ideally be as close to 1 as possible.

Analysis 3: MLFF BP

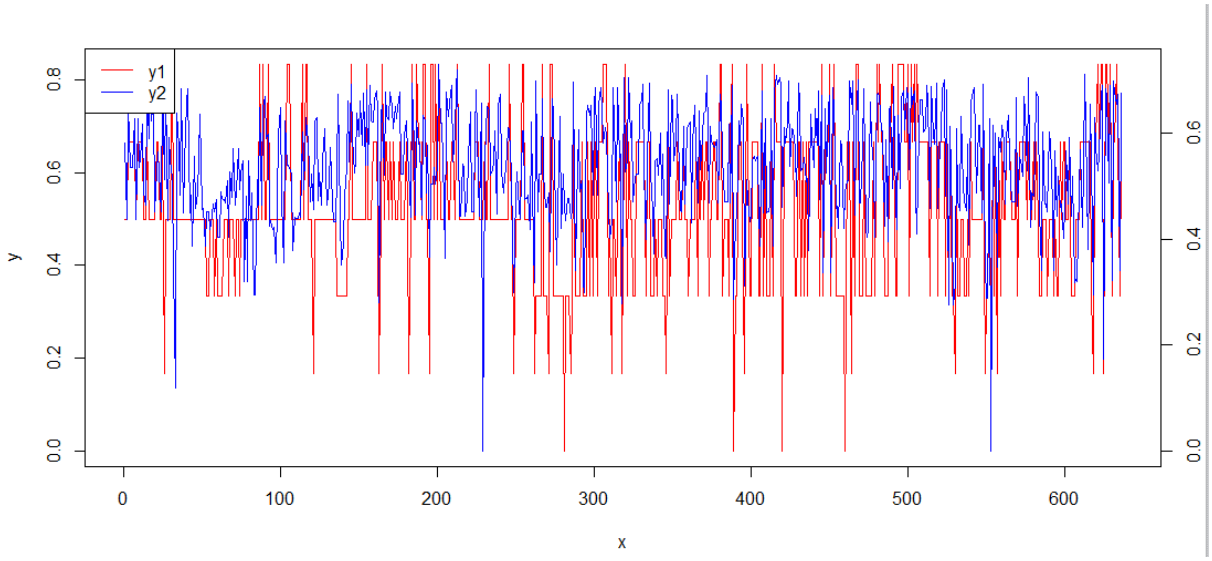
We carried the out modelling process using the 'neuralnet' package in R. Here again the goal was to predict the value of the quality of wine using the attribute set available to us. Data was split in the following ratio: training (70), cross validation (15), testing (15). The task in particular was to predict a value using the attribute list consisting of the normalized values of "quality~fixedacidity+volatileacidity+citricacid+residualsugar+chlorides+freesulfurdioxide+totalsulfurdioxide+density+pH+sulphates+alcohol". Here one important point to be remembered is output variable needs to be scaled to a range between 0 and 1. This will enable us to use the sigmoid activation function. Sum of square errors was used as a measure of the error. Back propagation was used to learn the weights.

As before the same learning process was applied. The network was trained using the training data and the different parameters were modified to evaluate at the cross validation stage. The network with the best settings was used with the test set.

Network Architecture



Predicted vs Actual Output



Y1<-Actual

Y2<-Predicted

MLFF-BP performed much better than GRNN in this case.

Analysis 4: SVM

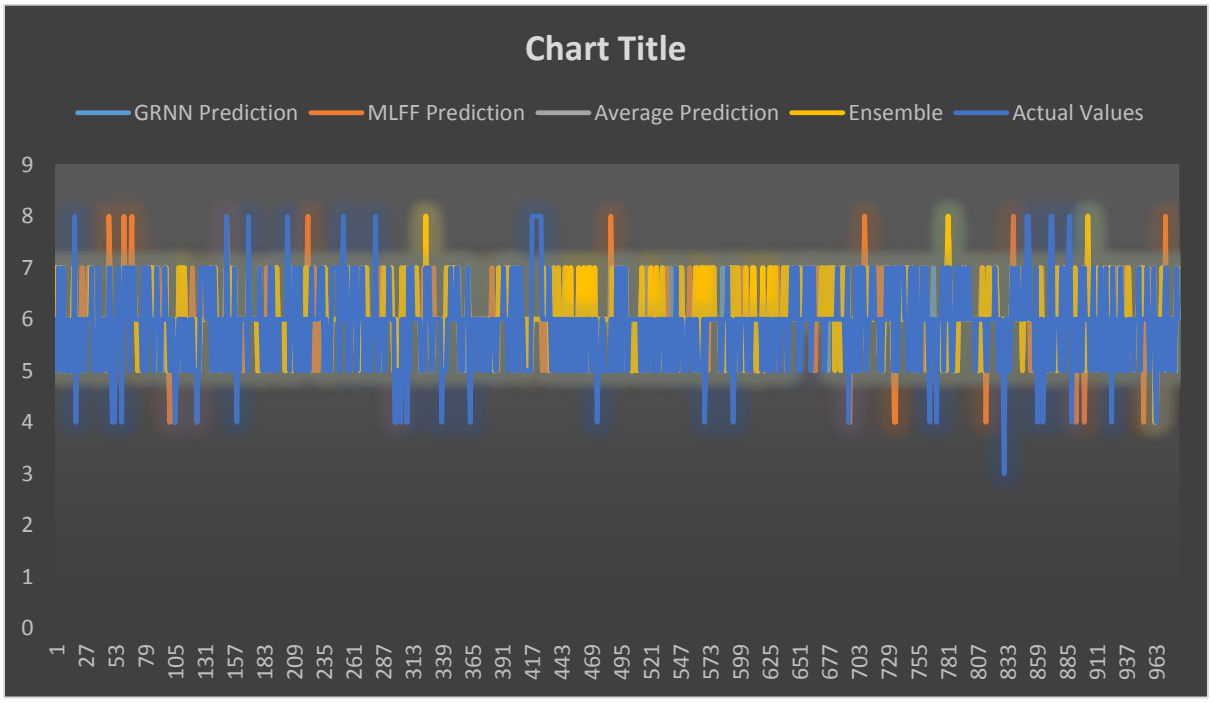
Support Vector Machines are a class of learning algorithms that can be used to carry out classification. Since there are 10 discrete values indicating the quality of the wines dataset, we can also treat this as a kind of classification problem. The normalized data set was provided to 'rattle' was modelled with SVM. Of the different kernel functions used by SVM, RBF gave the best results. The result is tabulated as below:

```
Error matrix for the SVM model on winequality-white.csv [test] (counts):

        Predicted
Actual    3    4    5    6    7    8    9
     3    0    0    1    2    0    0    0
     4    0    0   24    8    0    0    0
     5    0    0  128   83    1    0    0
     6    0    0   65  276   18    0    0
     7    0    0    5   77   30    0    0
     8    0    0    0   14    4    0    0

Error matrix for the SVM model on winequality-white.csv [test] (%):

        Predicted
Actual   3   4   5   6   7   8   9
     3   0   0   0   0   0   0   0
     4   0   0   3   1   0   0   0
     5   0   0  17  11   0   0   0
     6   0   0   9  38   2   0   0
     7   0   0   1  10   4   0   0
     8   0   0   0   2   1   0   0
```

Analysis 5: Ensemble

The final approach to predicting wine quality is by calculating an ensemble model. The Ensemble model is provided the output of the other neural network models namely GRNN and MLFF-BP. One approach would be to take the average of MLFF-BP & GRNN as the prediction. Thus we now have 3 sets of predicted values. Once we have all this in place we can use a voting approach to decide what the result should be. If more than 2 models predicted the same values, then we output the result as that value.

The following is a line graph illustrating how the values predicted by various models varies with the actual target variable.



Confusion Matrix for Ensemble 1

	ensemble					
actual	3	4	5	6	7	8
4	0	0	1	0	0	0
5	1	13	109	56	2	0
6	0	11	149	342	79	10
7	0	1	8	118	69	10
8	0	0	0	2	1	0

As the GRNN model did not perform well it will be prudent to try ensemble with other modelling approaches. We have thus tried to create an ensemble for SVM, NN and the average of the above two. The result thus obtained is tabulated below.

Confusion Matrix for Ensemble 2:

	ensemble			
actual	4	5	6	7
3	0	4	0	0
4	1	11	8	1
5	0	84	117	2
6	0	32	252	55
7	0	0	80	51
8	0	1	19	16

Further Suggestions:

There are certain issues in working with the wine data, we would like to highlight them so that better neural networks could be designed.

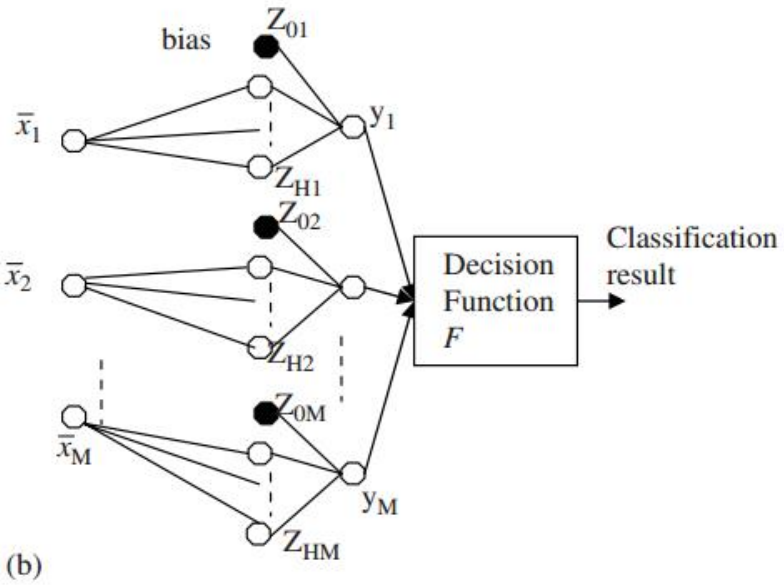
1. Wine dataset could be classified as a multi class classification problem, the data set does not contain enough information about the extreme cases and it is most populated at middle level, hence neural network is unable to read the characteristics of extreme cases.
2. Even though we try to minimize this phenomena by reducing the number of observations from the mid-level quality wines, there is still insufficient information about extreme quality wines, another approach could be to duplicate these observations in the data set and train the neural nets.
3. Neural network classification works best when there are few number of classes (2 classes as in best case, either 0 or 1), we can see that trained networks which have been tried earlier are able to provide a rough estimate eg, when the actual has to be, the predicted values are near 5, one possible approach to handle this problem well would be 'One Vs All' approach.

One VS All

In this approach a single classifier is trained per class to distinguish that class from all other classes. Prediction is then performed by predicting using each binary classifier and choosing the prediction with highest confidence score. However implementing this approach is time consuming.

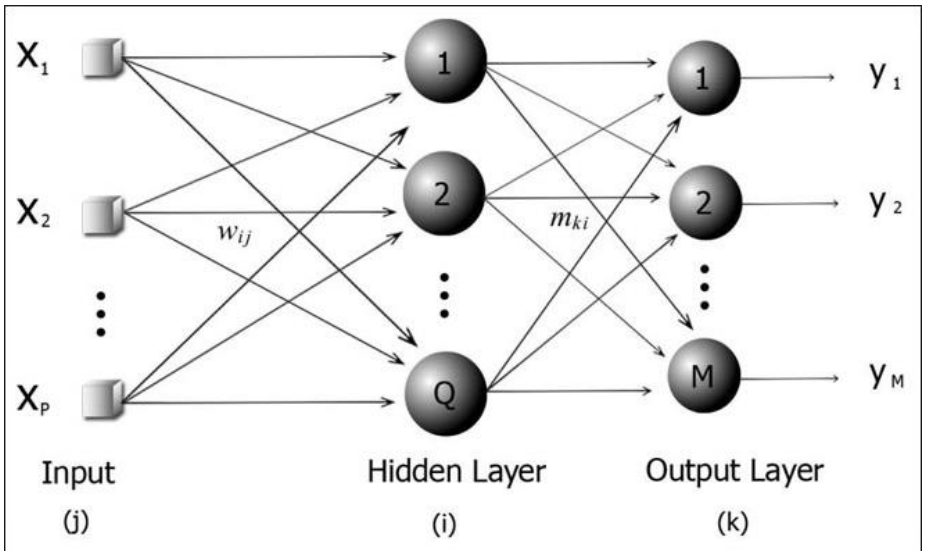
Steps:

1. For the same set of input variables we need to mask all other classes except the class we wish to train the classifier on.
2. We need to repeat step 1 for all the available 10 classes.
3. When we have all the necessary data sets, we can commence the training. Each classifier is trained with a separate data set. Thus we will have ten trained classifiers.
4. To make evaluation and prediction, we need to provide the same input to ten classifiers.
5. Of the all the ten values (using sigmoid activation, this is between 0 and 1 for output node) obtained we classify the input as belonging to the class with the highest output.



Where $x_1 x_2 \ldots x_m$ represent the different data sets used for each classes.

As an improvement, the same "One vs. All" classification can be performed by training a single neural net with ten output nodes and training each node for one class.



Where $y_1 y_2 \ldots y_m$ represent the predictions for various classes.