**1. Data pre-processing (including cleansing, data splitting, identifying predictors) – 15%**

I don't mind how/where you process the data set - i.e. inside your program or externally (e.g. in Excel), cover how you explored the data, cleansed the data, how you split the data into data sets, standardised the data, identified suitable predictors, etc. When you have cleansed your data, you should still have well over 90% left to work with.

**2. Implementation of the MLP algorithm (including modifications / improvements) – 35%**

**The language used (and why you chose it); What libraries you have used.**

I used java, as it allowed me to use OOP to break the program into functional sections, which made debugging it much easier.

Libraries used:

FileNotFoundException - Signals that an attempt to open the file denoted by a specified pathname has failed, which allows me to see where I have made a mistake, instead of the code simply crashing.

Random – generates random values, allowing me to start from many points in the weight space to find the global minima of the error function.

Scanner –

File - allows me to read from the excel file

ArrayList – allows me to make a new

List – data structure …..

**How you implemented it – e.g. OO approach with an MLP class and what methods it has, how the data are stored/structured, etc.**

**The MLP algorithm – what additions did you make – e.g. momentum, annealing, bold driver. Did you try different transfer functions? Alternative training algorithms – e.g. conjugate gradients?**

**Are there limits on your code (e.g. have things been hard-coded or can it create any MLP with any number of inputs, hidden layers, outputs, etc).**

Can have any amount of hidden layer nodes, but only 1 hidden layer and 1 output

**3. Training and network selection – 20%**

**4. Evaluation of final model (including comparisons between different modifications to the algorithm) – 20%**

GRAPHS AND TABLES

**5. Comparison with another data driven model – 10%;**

**Program listing (i.e. the code you have written)**

Avoid hard-coding things. In other words, don’t write the program for the given data set. It should be easily modifiable for other data sets, different numbers of inputs, different numbers of hidden nodes, etc.

I need to be able to read (with my eyes!) your code. So please submit it as a listing embedded in your report (e.g. as an appendix), or a separate text file or separate pdf (you can submit a zip file with all these in if you wish), so I can actually look at it.

I WILL NOT be running the code - so do not submit it as a file created from your IDE (as I may not have the same IDE you have so might not be able to open it). I simply want to see your program listing with the backpropagation algorithm highlighted (along with any improvements) - to see that it has been implemented correctly. The way I will see that your program works – is seeing all the ANNs you produce and evaluated.

This means I don't care about things like the user interface, how your program stores results, etc. You are using your program to evaluate how good an MLP is at solving problems - so the focus is the MLPs you produce. I don't need a user guide submitting or a discussion of HCI etc.

Make sure you highlight (e.g. with lots of comments) the actual backpropagation algorithm (and other enhancements) in your code so I can find it easily when marking. I need to see the algorithm(s) in your code to mark it (them).

I will also be looking at the code to see that is well structured, well commented, and decent variable names have been used.