### Checklist

• 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).

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

### Checklist

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

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

3. Training and network selection – 20%.

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

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

### Using python

I decided to use python as the programming language of choice, this is because I have practice with using the language and it provides good libraries to work with. I will be using a few external libraries in my coursework. Firstly, I will be using **Pandas** for data handling and manipulation and **NumPy** for support for arrays and matrices and **Matplotlib** for plotting the data into graphs. After reading in the data with Pandas, the data will mainly be handled using NumPy arrays or python lists.

I aim to code the neural network using three separate classes: **Node**, **Layer** and **Network**. The goal of the node class will be to handle the main computation, including forward propagation, backwards propagation and weight updates. It will store the weights, last inputs and have its own delta value. The Layer class will act as a connection between the layers so that every previous layer is connected to the next. The network class will handle the main MLP training algorithm and adding layers so that the user can easily change the structure of the network.

### Data pre-processing

The first step I took was to clean the data using excel, removing any obvious errors. After my first review I found the following errors:

* There were a few values
* There were a few ‘a’ values
* There was a “#” value in row 784
* There were a few outlying rows (for example, there was a value of 5000 for Arkengart when the closest value to that was 225, and in Malham the highest was 9000 and the closest to that was) so I deleted these rows to prevent it from skewing the data too much.

I then and deleted each of these rows manually using Excel. I then standardized the data set using the formula:

Text

Description automatically generated

To get the data within a range of . I then checked the data for any correlations between features and I found that Crakehill, Westwick and Skip Bridge (gauging stations) all had a linear correlation with Skelton,

GRAPHS HERE

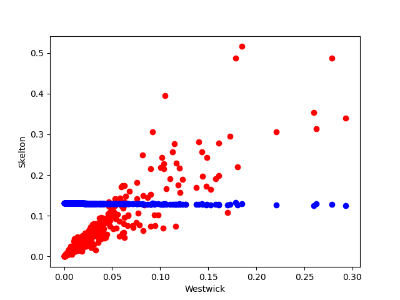
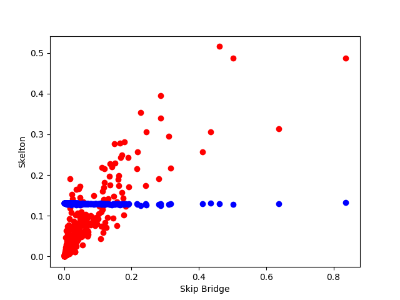
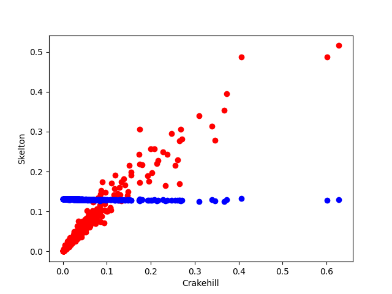
Whereas the rainfall stations had no such correlation, rather it was more scattered as can be seen below

GRAPHS HERE

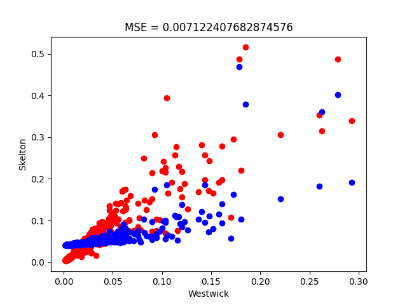
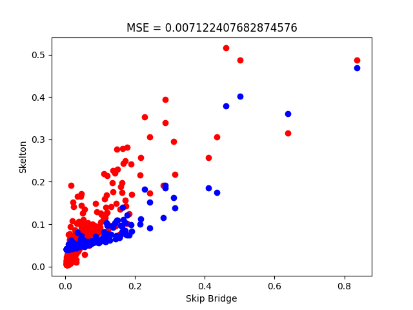
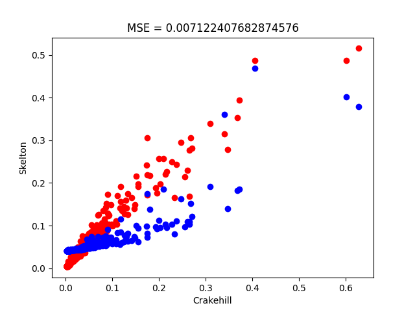
Therefore, I chose the gauging stations as my input features; the aim of my neural network prediction was to predict this linear trend for new data.

### Implementing the MLP Learning Algorithm

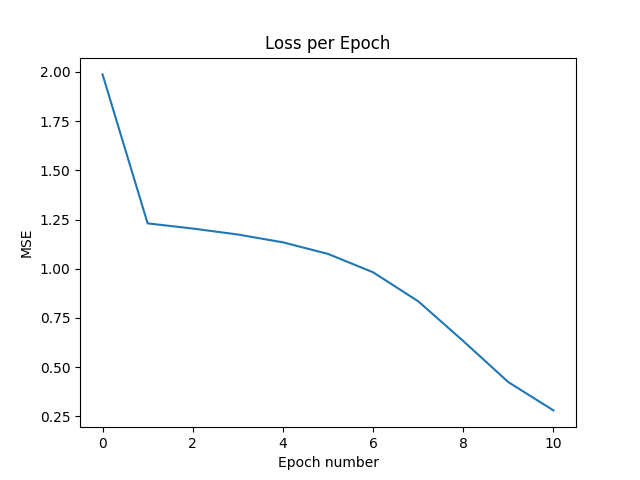
After first implementation all of the output values were converging to the same/similar value each time for different inputs. For example, after training the neural network with 6 hidden nodes and 1 output node, I got the following results (red is the expected linear trend we want, and blue is the trend we have predicted):



After playing around with the values, I eventually noticed that the algorithm was changing the weights very minimally each update, so I tried increasing the learning rate from 0.1 to 0.5 to increase the size of the change, which improved the model drastically:



From here I trialled different learning rates to find the best outcome:



Learning rate = 0.6

### Adding Momentum

### Bold driver

Unable to implement

### Annealing

Unable to implement

### Weight decay

Unable to implement