Answer 1:
Here's a simple version of perceptron using Python and NumPy. It will take two inputs and learn to act like the logical **OR** function.
I mapped the possible input to the expected output. The first two entries of the NumPy array in each tuple are the two input values. The second element of the tuple is the expected result. And the third entry of the array is a "dummy" input (also called the bias) which is needed to move the threshold (also known as the decision boundary) up or down as needed by the step function. Its value is always 1, so that its influence on the result can be controlled by its weight. I choose three random numbers between 0 and 1 as the initial weights. The errors list is only used to store the error values so that they can be plotted later on. The eta variable controls the learning rate. And n specifies the number of learning iterations.
In order to find the ideal values for the weights w, I tried to reduce the error magnitude to zero. In this simple case n = 100 iterations.
First we get a random input set from the training data. Then we calculate the dot product of the input and weight vectors. This is our (scalar) result, which we can compare to the expected value. If the expected value is bigger, we need to increase the weights, if it's smaller, we need to decrease them. This correction factor is calculated in the last line, where the error is multiplied with the learning rate (eta) and the input vector (x). It is then added to the weights vector, in order to improve the results in the next iteration.
It's easy to see that the errors stabilize after few iteration. If you doubt that the errors are definitely eliminated, you can re-run the training with an iteration count of 500 or more and plot the errors.

Answer 2:
For Non Linear example eI tried to model XOR function, it's not possible to model an XOR function using a single perceptron, because the two classes (0 and 1) of an XOR function are not linearly separable. Thus you cannot see any convergence.

Answer 3:
After cleaning the Titanic data set I used Age, Sex , PClass and made two CSV for train & test data. I applied the Adaline batch and Adaline SGD both for the data and could get 58 % and 42% accuracy.

Answer 4:
According to my solutions and working PClass, Age & sex are the most predictive features of Titanic Model. I have assumed that Women & children and high class people have survived the most. I used these three features to perform Adaline and Adaline SGD. Also during the modelling I observed these features have the maximum weight to provide the prdiction and accuracy.