Problem 1

Both Adaline and Perceptron use only one neuron to learn patterns in the data while Perceptron adjusts the weights after mis-prediction while Adaline uses Gradient Descent approach to to move forward to global minimum in a linearly separable problem. Adaline converges faster than the Perceptron although requires more time to train and performs heavier computations. Both of the algorithms utilizes the use of learning rate as the hyper parameter.

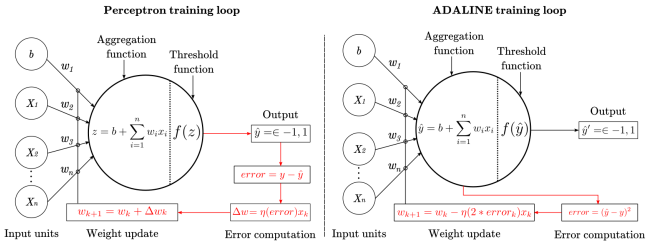

Figure 1: Adaline architecture vs Perceptron

Problem 2

Pearson correlation hitmaps for training data, validation data and testing data:
training data correlation hitmap
validation data correlation hitmap
testing data correlation hitmap
correlation histograms:
training data correlation histogram
validation data correlation histogram
testing data correlation histogram removing duplicate columns (features) or the ones which are the linear combination of the other columns results in a better accuracy of the model. also features which are highly correlated doesn't add extra information and insight to the model, so its preferred to delete one of the highly correlated features (between two features) and select only ones which provide a higher gain ratio. According to histograms and hitmaps, some features are too correlated to other ones (the red cells), so its better to delete those features.

Problem 3

(note: in all problems, the dataset is normalized with min-max scaler, shuffled, and the bias is added)
$f(x) = 1$ for $x > 0$ and -1 for $x < 0$ is used to train the Perceptron unit since its range is suited for binary classification while it's not differentiable and could not be used in Gradient Descent algorithm.
training and validation costs history is available in Perceptron cost history
according to the figure, the network is converged in epoch 405. the point is the **early stopping**

point. no change in the validation error is observed from there because of the accuracy of 100.00%, so this is a good point to stop the iteration.
results:
Perceptron accuracy on training data: 100.00%
Perceptron tn, fp, fn, tp on training data: 72, 0, 0, 72
Perceptron accuracy on testing data: 76.74%
Perceptron tn, fp, fn, tp on testing data: 21, 7, 3, 12

Problem 4

$f(x) = x$ is used to train the Adaline unit since it's continuous and differentiable and could be used in Gradient Descent algorithm.
training and validation costs history is available in Adaline cost history
according to the figure, the network is converged in epoch 240. the point is the **early stopping** point. in the further iterations, the validation error is being increased because of overfitting, so we should stop iteration at epoch 240.
results:
Adaline accuracy on training data: 88.89%
Adaline tn, fp, fn, tp on training data: 73, 8, 8, 55
Adaline accuracy on testing data: 81.40%
Adaline tn, fp, fn, tp on testing data: 14, 4, 4, 21 Adaline is converged faster than Perceptron and also is more accurate. it seems Perceptron model in this example is a very high variance model and is overfitted on the training data and doesn't predicts (inferences) well.

Problem 5

to fit non linear models, it's sufficient to add polynomial features to the training dataset.
training and validation costs history for second order Perceptron is available in second order perceptron cost history
by comparing this model with simple Perceptron model we can conclude that the network converges faster (here with almost 160 epochs) since it can fit non-linear shapes. (generally we could say the accuracy increases by making the model more sophisticated).
results
second order Perceptron accuracy on training data: 100.00%
second order Perceptron tn, fp, fn, tp on training data: 80, 0, 0, 64
second order Perceptron accuracy on testing data: 76.74%
second order Perceptron tn, fp, fn, tp on testing data: 16, 7, 3, 17

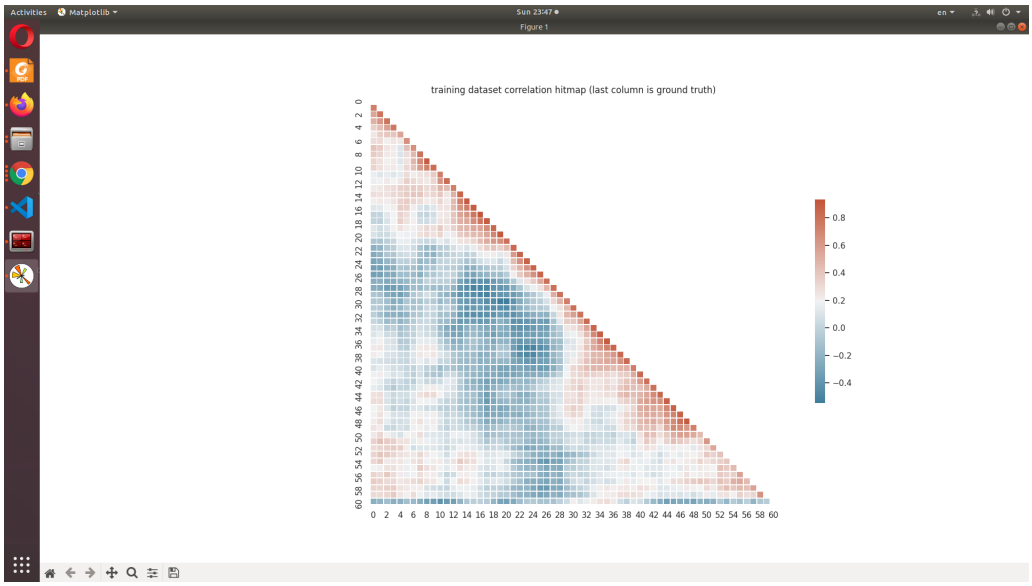

Figure 2: training data correlation hitmap

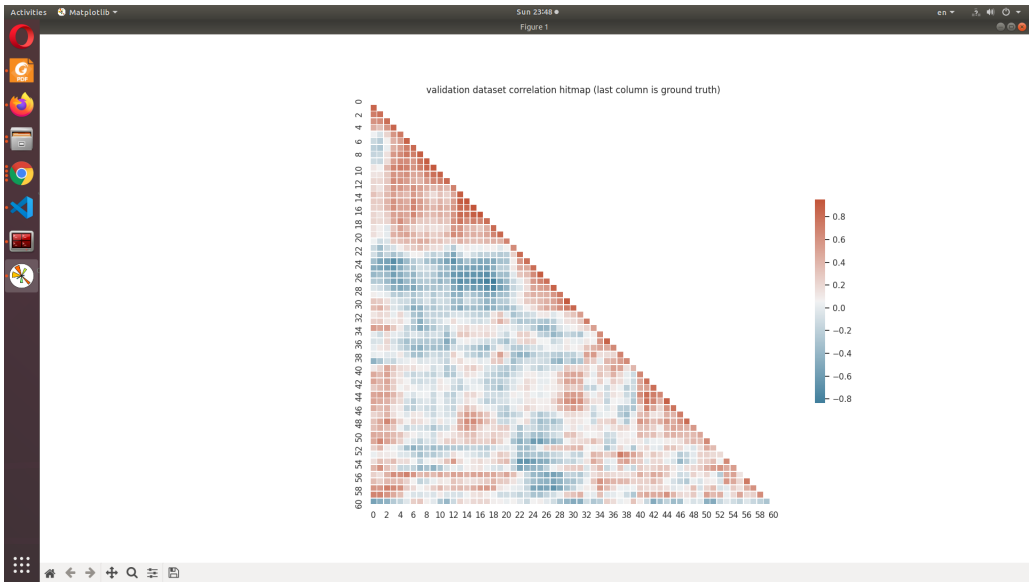

Figure 3: validation data correlation hitmap

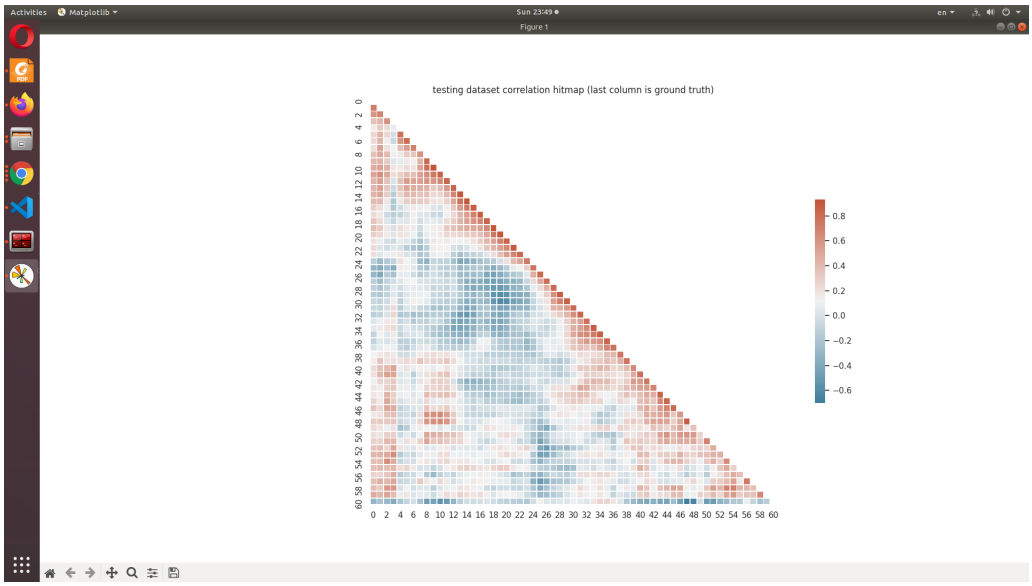

Figure 4: testing data correlation hitmap

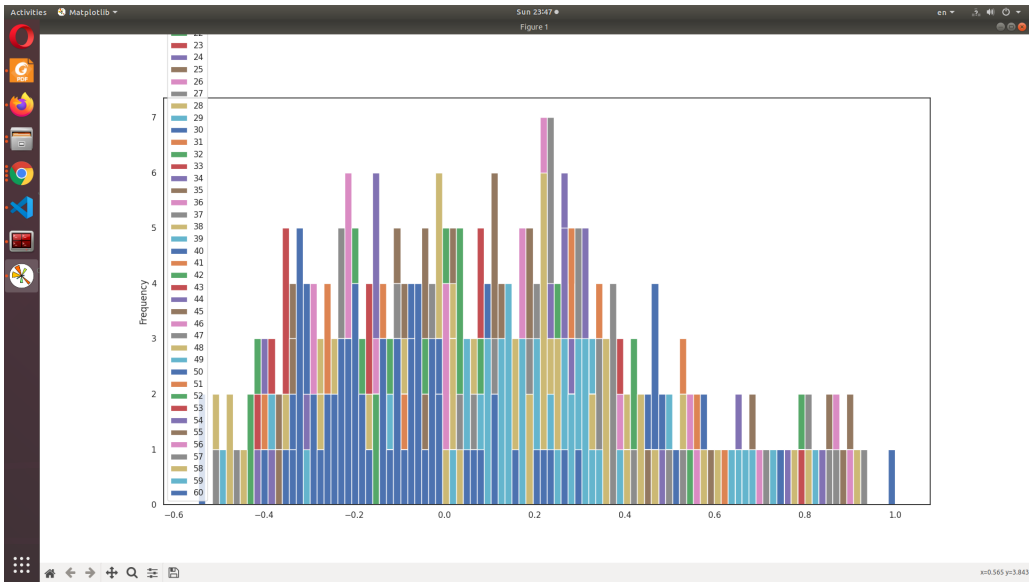

Figure 5: training data correlation histogram

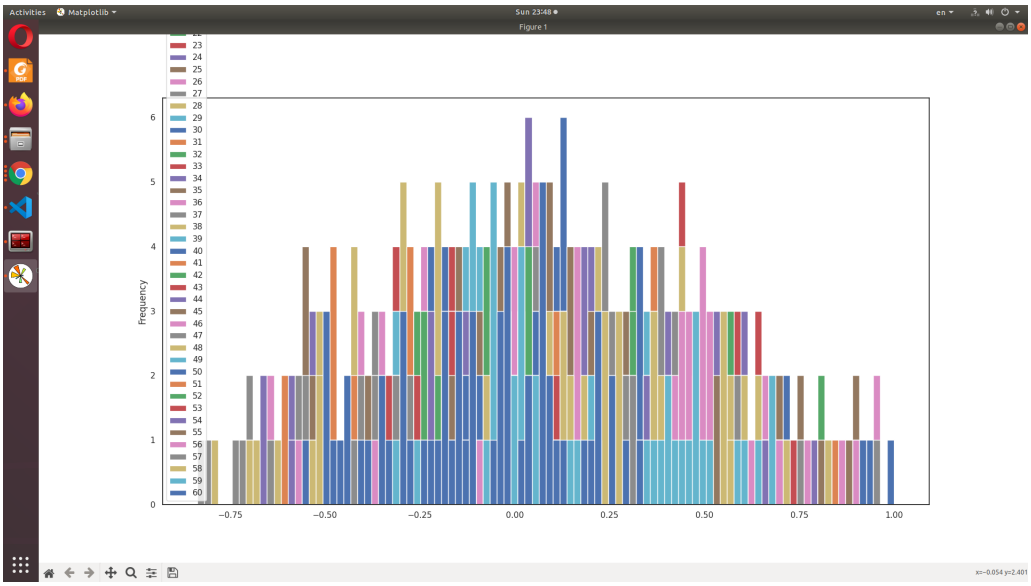

Figure 6: validation data correlation histogram

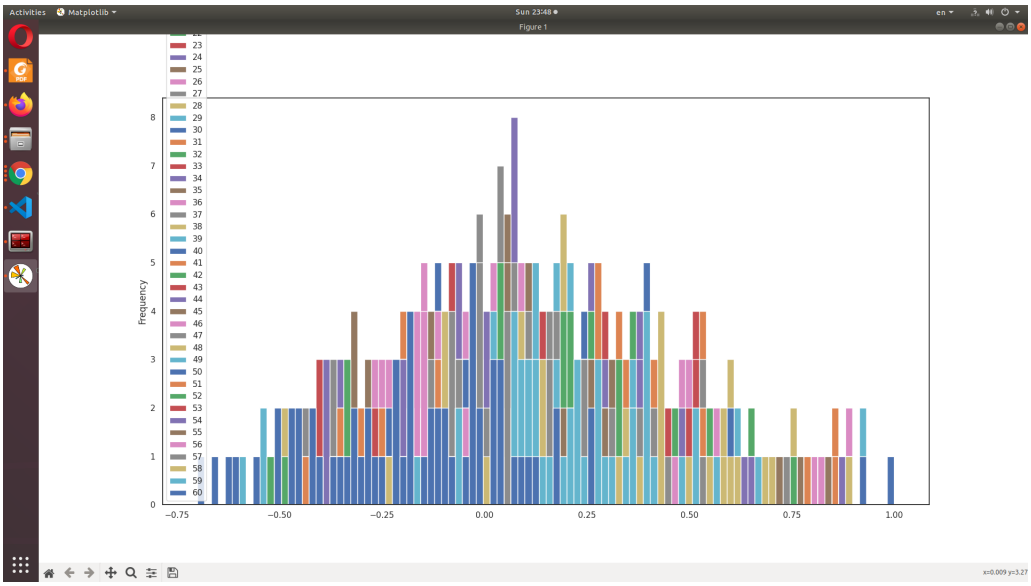

Figure 7: testing data correlation histogram

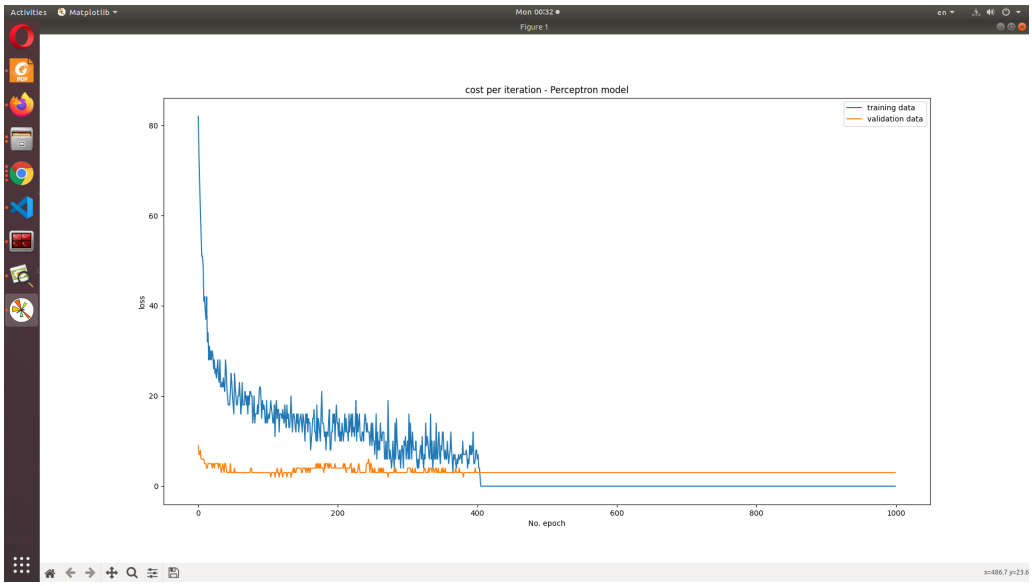

Figure 8: Perceptron cost history

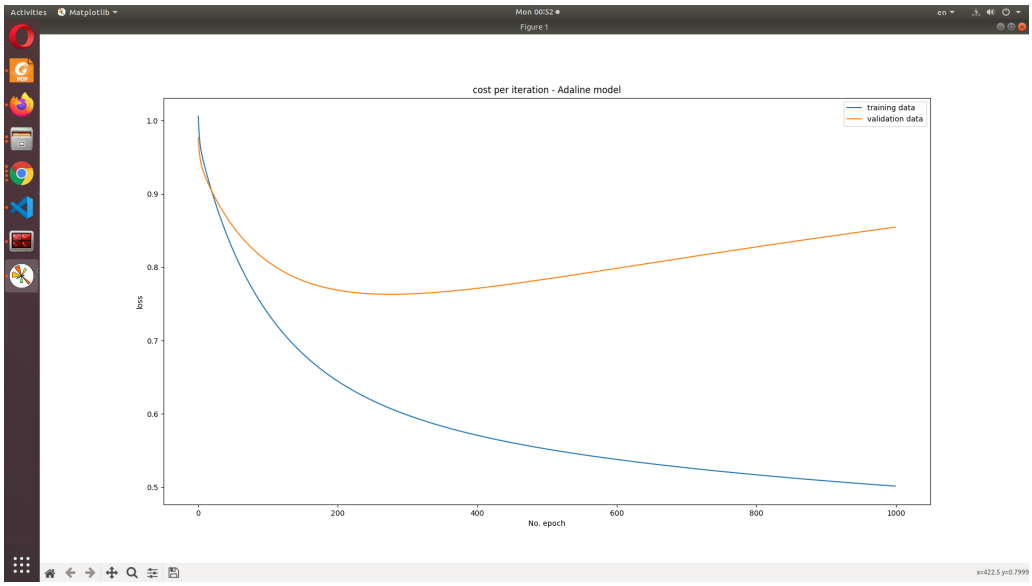

Figure 9: Adaline cost history

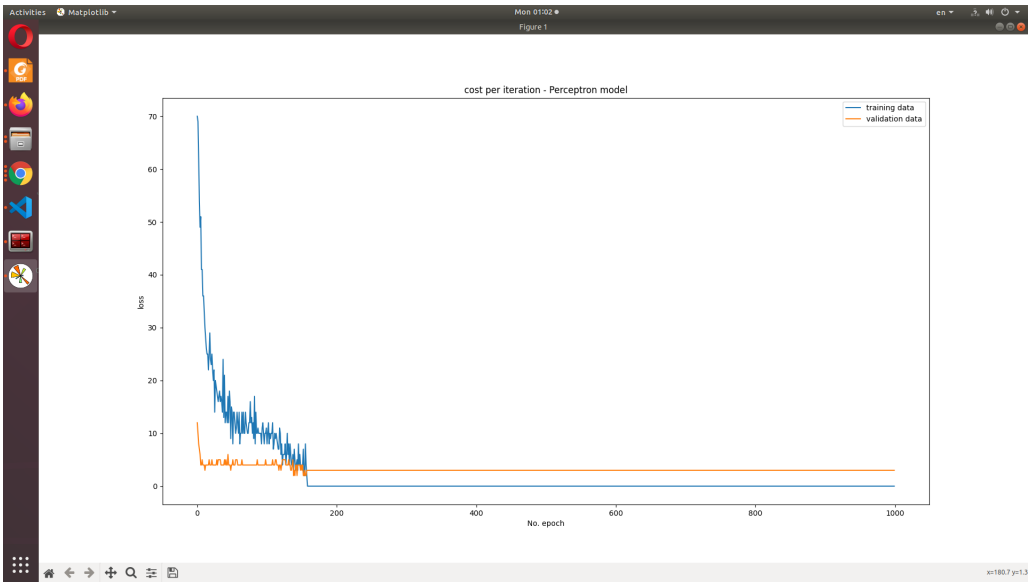

Figure 10: second order perceptron cost history