**Linear regression:**

1a) Implemented the linear regression learner which takes in 1d data and gives out the predictions as per the equation.

1b) Plot of all the function depth with data is available in the **results** folder with name starting with 1\_(\_bit).

1c) The MSE for training data reduces with increase in depth but for test data it starts to increase. For depth 5 and 6 we get maximum MSE (0.023 and 0.024 respectively) with test data which suggests that overfitting has occurred. So, the best fit would lie somewhere around depth 1 and 2 although depth 0 has less test error. Given more test data points it would have been clear to recognize the best fit.

**Locally Weighted Linear Regression:**

2a) implemented the locally weighted linear regression learner which takes in 1d data and gives out the prediction as per the equation without any nonlinear parameters.

2b) Plot of all the function depth with data is available in the results folder with name starting with 2\_(\_bit).

2c) The performance of this implementation is better than normal linear regression as we get a lower test error(0.019) almost similar to the training error(0.026).This specifies that there is less overfitting compared to the implementation in 1b.

2d) While using only 20 elements we fit the data properly for training set but when we evaluate using test set the data is not fitting better than the earlier implementation. We get a test MSE as 0.24 and train MSE as 0.11 which clearly shows overfitting. This must be happening as there are less data points to train and the learner will be more biased only with few local training weights.

2e) **Yes,** I believe that the data set used was actually derived from a function that is consistent with the function format in question 1 as the error with 0 function depth which doesn’t have any sin or cos parameters was similar to the implementation in 2\_c and 2\_d but when the depth increased we saw training error reduced in question 1\_c which signifies fitting the given equation appropriately.

**SoftMax Regression:**

3a) Implemented the SoftMax regression learner to take any classification data and predict the desired class for each input.

3b) the performance obtained for k=5 and Manhattan distance in hw1 with all 4 features was **53.33%** whereas with SoftMax regression we obtained performance of **67.5%**. This shows that SoftMax is better at predicting the given data set. SoftMax is better here as our dataset is linearly separable for different classes given and the dataset is supervised with multiple features or dimensions to predict. KNN works well for non-parametric data.

3c) After removing the fourth attribute which is color, we obtained performance of **83.33%** for SoftMax regression and **80%** for KNN. This shows color was not a good attribute for prediction. Here again SoftMax regression performs better than KNN due to the similar reason mentioned in 3b. Again if we run our regression on more epochs or iteration we will have better results compared to any k value for KNN.