**ENGR421 – HW4 report**

First, I read the dataset, and turned it into a numpy array, aşso stored the number of all datapoints in a variable. Then I split the training adnd test datasets as said by the first 150 datapoints being training and the rest 122 as test set, by a simple array operation.

Then to implement the regressogram I set the bin width, origin and the largest datapoint of the x axis as the last point for the binning. After this I generated the values for the left and right borders of the bins using the *np.arrange* function. I then wrote a function to implement the regressogram which returns the predicted values. For that I initiated *count, sum\_ys* and *p\_hats* variables. Then using nested for loops with I checked all the datapoints for their x axis values if they fit into that particular bin by checking the left and right borders of that bin. If so, I incremented the count of the bin, which indicates the number of datapoints in the bind and sum the y values at *sum\_ys* variable. After finishing each bin, I assigned the p\_hat value to be the average of y values in the bin by storing *sum\_ys / count* in the p\_hats array with the index indicating the bin number. All these calculations are simply representative of the rregressogram algorithm which is expressed mathematically as (1), in words it means the sum of the y values in the bin divided by the number of datapoints in the bin. Then I plotted the data and the respective regressogram. And pringred the RMSE value which is simply the difference between the actual y values and the predicted values squared and divided by the number of datapoints at the test set.



(1)

*b(x,xi) = 1 if x is in the same bin as xi*

Further, to implement the Running Mean Smoother algorithm, which defines a bin that has a middle point as the new datapoint. Since the bin is defined in this way, the datapoints that are further away from this new datapoint by more than the half of the bin width aren’t a member of this bin. Therefore, what I did was to construct a nested for loop checking each test x datapoints distance with the train set x values and if the distance is less than the half of the bin width, I stored the y values in a list which than I averaged and saved as the predictions for this new datapoint’s y value. This again is a representation of the mathematical expression (2), which again in words mean average all the y values by the number of datapoints in the bin, with the bin defined by the distance between the training set x value and test x value being smaller than half of the bin width (as the test x is the middle of the bin). Finally, I plotted the data points and the running mean smoother. However, my plotting result is slightly different from the pdf, yet the RMSE value is identical.



(2)



u = (x-xi)/h

W(u) is 1 is abs(u) <=1/2

Finally, to implement the Kernel Smoother algorithm, I again used nested for loops to calculate u for all test x values for all train x values and calculated the K function, which is equivalent of the Gaussian density function, but uses u and assumes 0 variance. By multiplying this with the train y values and summing these multiplications divided by the sum of all K values I generated the predictions for all test y points. This again is equivalent of the mathematical expression of the Kernel Smoother algorithm (3).



(3)



K(u) =1/2pi \* exp(-u^2 /2)