3)

For the question, I implemented K-Nearest Neighbors and Polynomial Regression using the SciKit Learn (SKLearn) library in python to try to minimize the noise in the two functions given in the question. First, I implemented functions that represented the functions provided in the problem for 'bumps' and for 'doppler'. Then, I followed the same equation given by the problem to generate 2048 linearly spaced points between 0 and 1 and executed the function over those values to generate the noise-free function plots. The root loss function was then defined by using the equation given in the problem statement as well. From the graphs provided by Donoho et al., the sigma value of the noise was provided to be 1, and the signal scaling value was estimated based on the graphs provided. The scaling factor for the bumps function was estimated to be 10, and the scaling factor for the doppler function was estimated to be 25. For each replication, Gaussian noise was generated by taking 2048 random values from the standard normal ($z \sim N(0,1)$). Polynomial regression and KNN regressions were fit per the documentation provided by the SKLearn library, and both the root-mean-squared error and the loss were captured for each replication performed. After all replications were completed, the mean of the 20 root loss values were taken for each regression type and each function, as well as the average RMSE as evidenced below. Additionally, the graph of the last run is provided below to also identify the fit.

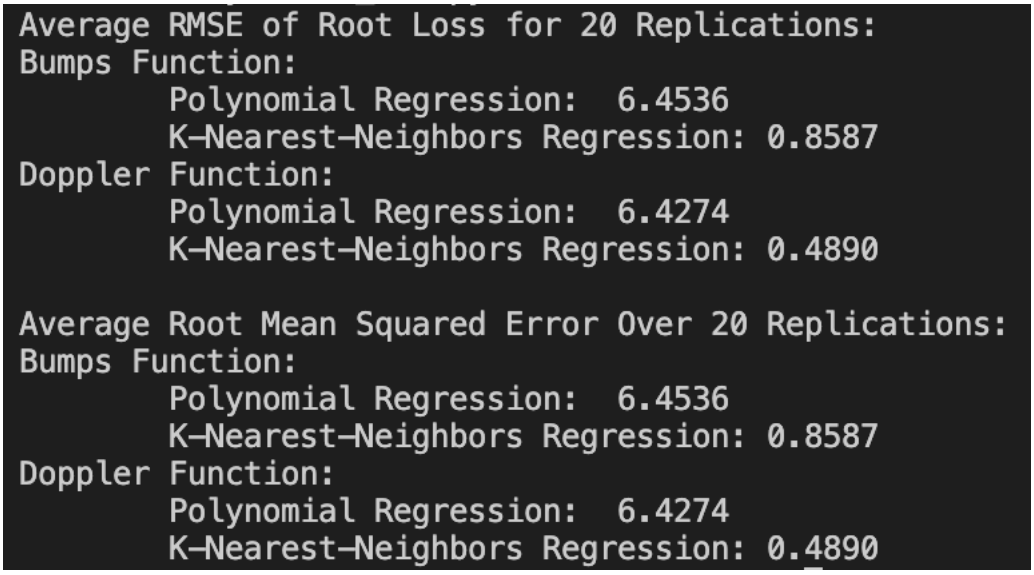

Figure 1: Output from the provided python function. The Average RMSE of the Root Loss and Average Root Mean Squared Error for 20 Replications for both types of regression for both functions provided in the homework question. .

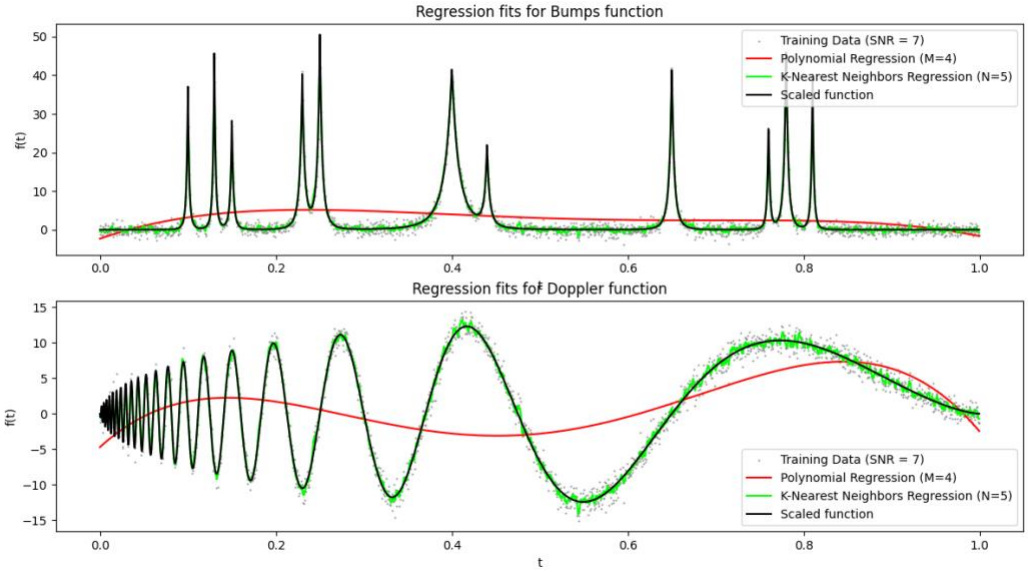

Figure 2: Graphs indicating the fit for both functions. The green line is the KNN regression, and the red line is polynomial regression. The gray dots are the noisy values that the two regression functions were trained and fit against.

To supplement, we see that the polynomial regression does an extremely poor job of estimating both functions, even without taking the noise into account. Neither function was able to be well defined using a polynomial of degree 4, which is why polynomial regression was beat by the wavelet methods in both cases by significant margins. To summarize, the polynomial regression was not the correct regression method to use in this scenario, and thus, the loss value and RMSE values were large.

Regarding the K-Nearest-Neighbors regression with k=5, we see that it does a much better job estimating the original function than the polynomial regression method used previously. We see that the loss/RMSE values for the KNN regression were 0.8587 and 0.4890 for the Bumps function and the Doppler function, respectively. For the Bumps function, we see that the RMSE is beaten by all the wavelet methods except for VisuShrink using S8. For the Doppler function, we see that the RMSE of the KNN regression is better than some and worse than some, roughly somewhere in the middle when it comes to efficiency. I believe that wavelet shrinkage was superior to KNN regression in these cases because of the logic behind the algorithms. KNN takes the nearest k neighbors to a point and analyzes the approximate value of what the point should 'look' like, which does not account for noise, meaning that noise would affect the regression function. This is in contrast to wavelet transforms, which splits the data into approximations (lowpass) and detail (highpass) portions, minimizes the highpass terms, and takes the inverse to identify the de-noised signal. In essence, the reason why wavelet shrinkage was better than KNN is because the wavelet shrinkage minimzed the effects of noise on the regression, while KNN regression took noise into the regression pattern, and did not account for it.