1. When we state our first function (KNeighborsRegressor), we created a function generating function that takes a number of neighbors as an argument. When we use the “fit” function, we are giving the KNeigborsRegressor function the data set that it will run it’s k-neighbors regression on. That way, when we run the predict function, it has the data set to run the k-neighbors regression on, given the number of neighbors to predict on.
2. If the k-NN regression model uses k=n, then every predicted point will be an average of all points in the data set, so the output will be a line at the mean of the dataset.
3. If the value of k is too small (see the n=2 line of our graph), then the predicted regression line will have too much noise, however if k is too large (see the n=200 line of our graph), then the predicted regression line will not fit the data well enough and will hug the mean. So, as you can see in the graph, a moderate value for n gives the best results (see the n=50 and n=100 lines of our graph).
4. Some of the calculated R^2 values are indeed negative. This indicates that the prediction for the y-values is farther off from correct than the average of the y-values is. R^2 is not in fact a squared value because it is 1 – (predictor/mean), so it doesn’t break any rules of math that it is negative. It just means that the predictors were worse than the mean for every value of x.
5. An R^2 value of 0 means that the prediction performed exactly as well as the exact mean of the data set.
6. As k gets larger, the training and test set do exhibit similar trends; both the training and test decrease their R^2 value as k increased.
7. As the value of k increases, the value of (y – y^) approaches the value of (y-mean), implying that the prediction of y just approach the mean of y as k increases. If the prediction approaches the mean, then the value of R^2 will approach zero.