ROB313 Assignment 1

Q1. Objective is to create the knn regression algorithm, knn_reg, in code. Also had to create an algorithm that can perform 5 fold cross-validation on a training dataset to estimate good K values and choose a good distance metric. This was done in one function, knn_5fold, and another function, knn_err, was created to run knn over a testing dataset and compare it to the true values to produce the RMSE of knn with its given k and distance metric.

knn_reg(xtest,xtrain,ytrain,k,l) - requires you to input k value and L (1 or 2) to specify distance metric (1=Euclidean, 2=Minkowski with p=3) as well as xtest, xtrain, and ytrain points. Will output array of y predictions for xtest.

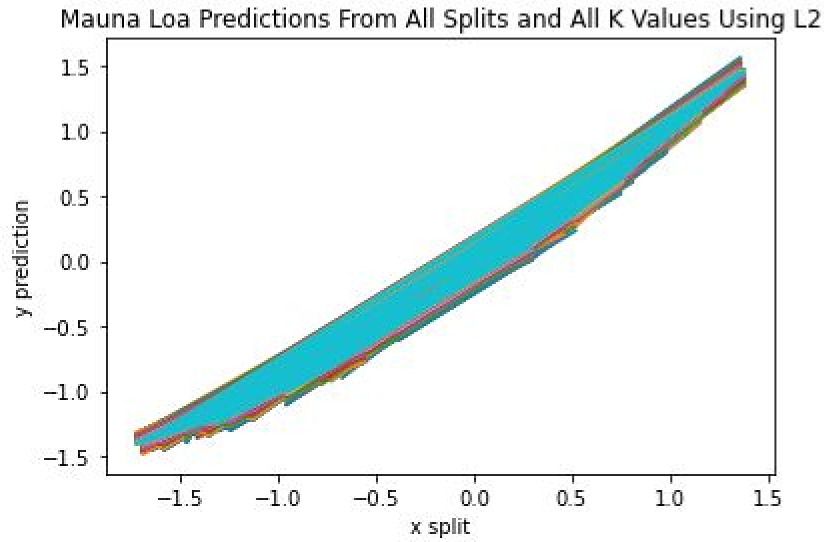
knn_5fold(xtrain,ytrain) - takes in xtrain and ytrain data sets. Performs 5 fold cross-validation on training sets and runs KNN over multiple k values and both distance metrics. Then calculates RMSE of output. Will print estimated k, L,and RMSE and return 0 on completion.

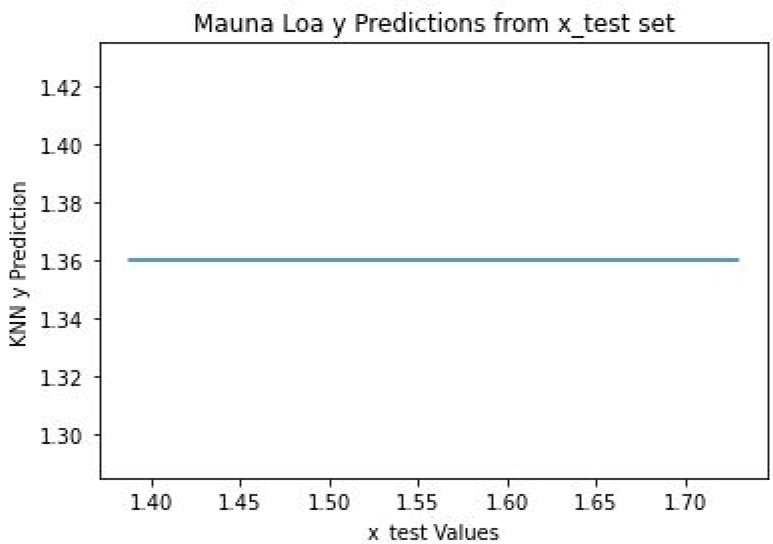
knn_err(xtest,ytest,xtrain,ytrain,k,l) - Will perform KNN, with given k and L value, on an input xtest set using input xtrain and ytrain sets then compare its predictions to the input known ytest values and then return the RMSE of the KNN with its given k and L settings.

My search procedure to estimate k and the distance metric was done by running my KNN 5 fold validation over every reasonable k ($k<\sqrt{N}$) and distance metric (L1 or L2) and calculating the RMSE of each outcome then compiling the RMSE, L, and k values into their own arrays (each had matching indices - for example the RMSE of fold 1 when k=1 and l=1 would have the same index throughout all arrays). I then used np.argmin to find the index of the lowest RMSE and thus estimate the preferred k and L combination.

	5 Fold RMSE	K	L	Test RMSE
Mauna_loa	0.00641377	10	2	0.06099168
Rosenbrock	0.629309131, 0.88420056	19	1	0.10574214

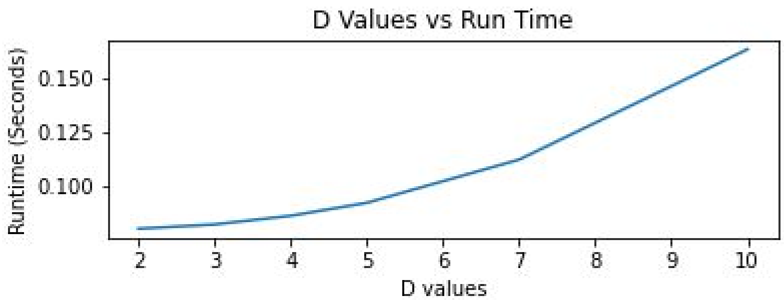
Pumadyn32nm	0.57328906, 0.51143701, 0.54528533, 0.51883375, 0.53818119, 0.56548371, 0.55752719, 0.54649141, 0.53099599, 0.54577932, 0.57964096, 0.52914075, 0.53305128, 0.59102843, 0.5387209, 0.80365903, 0.56508039, 0.55157636, 0.52837679, 0.5568946, 0.54855266, 0.53576814, 0.56411829, 0.54627673, 0.52828565, 0.53808016, 0.53376694, 0.5487159, 0.53633099, 0.55259372, 0.52805982, 0.53550876	37	1	0.36359724





The predictions from the 5 fold validation appear to follow a similar pattern to that of the training data with an overall positive slope. The x_test values appear to give a constant prediction for the y values. This is due to the fact that every x_test value is larger than any x_train value causing every x_test value to have the same K neighbours but with different distance values to them, giving the same prediction for every point.

Q2. Create knn_reg like in Q1 but using KDTrees to compute nearest neighbours.



The performance of KNN using KDTrees is much better compared to using brute-force to find nearest neighbours. When running my brute force code with rosenbrock with d=2 and n_train=5000 it took 0.67862 seconds compared to KDTrees only taking 0.08008 seconds for rosenbrock with n_train=5000 and d=2. The KDTrees algorithm has a faster runtime and better performance compared to the brute-force approach.

Q3.