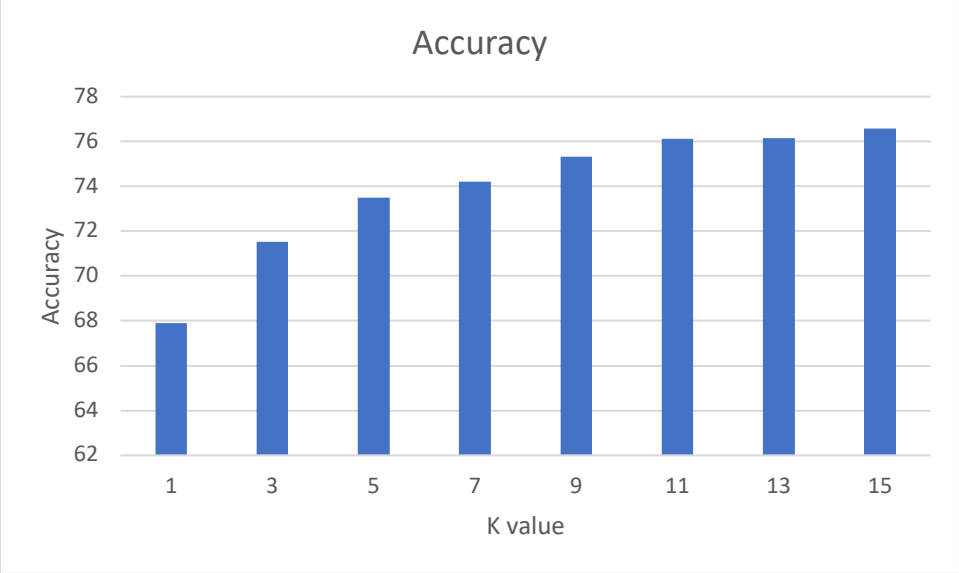
1. I implemented the K-nearest neighbor algorithm with the option for linear regression and weighted distance. The fit function saves the data point, with the option of normalizing the data. The predict function is an n^2 operation, for every row in the test data set, it will get the distance from it to every point in the saved train data set. The getDistance function calculate both the weighted distance and the Euclid distance from each point in test_data to each point in train_data. I used numpy's argpartition function to get the kth lowest value. I was able to achieve 93.57% accuracy on the seimic-bumps dataset.
2. Running the KNN algorithm without distance weighting on the magic telescope dataset, with the K value as 3; it seems like without normalization results in higher accuracy. Without normalization the accuracy is 80.82 percent, however with normalization the accuracy is 71.53 percent. With normalization, the best k value is 15. It seems like the more neighbors the algorithm checks, the higher the accuracy. It seems like this dataset is tightly compact with very few outliers

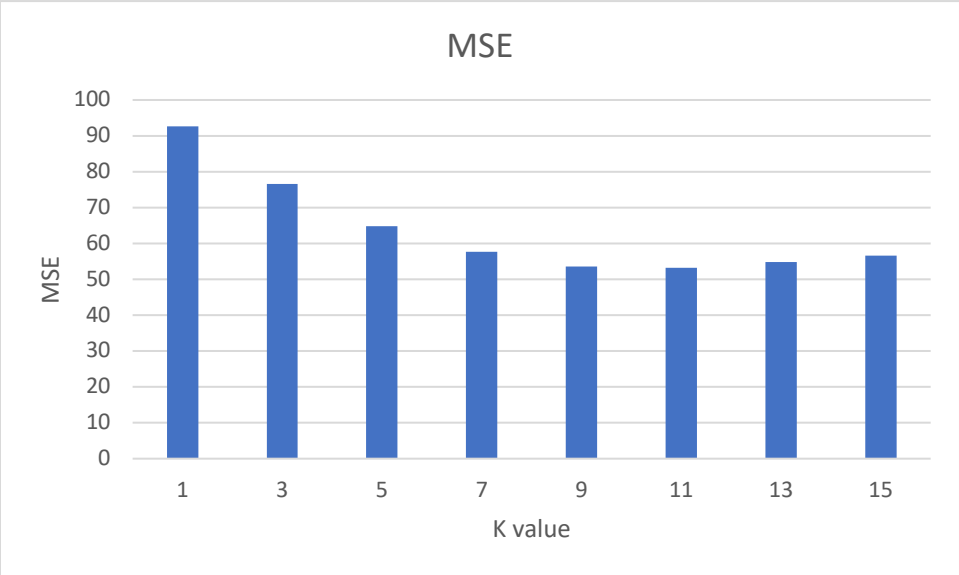
K	Accuracy
1	67.91
3	71.53
5	73.49
7	74.22
9	75.31
11	76.13
13	76.16
15	76.58



3. Running the linear regression variation of the KNN without distance weighting the house pricing prediction dataset produced more interesting results. It seems like the best K value was 11, it had the lowest Mean Squared Error. Increasing the K value would result in higher error as well as longer compute time. I did not normalize the regression output

because that would muddle the results. It would decrease the penalty for really bad predictions and would decrease the reward for really good predictions.

K	MSE
1	92.56
3	76.51
5	64.77
7	57.59
9	53.54
11	53.23
13	54.79
15	56.61



4. Repeating my experiment with distance weighting produced the exact same result. For the telescope dataset, it seems like the nearest neighbor is the same classification as the most common neighbor. For the house price prediction data set, it seems like the with linear regression version of the algorithm, it doesn't matter if inverse weight distance is active, it will give the exact same result.
5. Running the KNN algorithm on the credit approval data set required modification to my code, since the dataset contain continuous and nominal attributes. I inserted an if statement in my get_distance function, after the target row is subtracted from the train data row. For the column that have real values, if the difference is not zero, then the difference will be set to one. I implemented this distance metric because it was what the paper implemented as a distance metric for real values. I used k = 7 with a train test split of 70/30 and was able to achieve a 72% accuracy.
6. Running the SK learn version of the KNN algorithm on the magic telescope dataset produced the exact same accuracy as the not normalized data set run of my algorithm. However their version was extremely fast compared to mine, it was light years faster. For the housing price predictor, the SK learn version was way better than mine. I'm

unsure of how the SK learn version calculate score, but even if it was RMSE and not MSE; it would have still beaten mine handedly.

7. I used the magic04.csv dataset, which is a classification dataset that classify whether or not an image in the sky is or is not a blackhole. Without the PCA and the wrapper algorithm, I was able to achieve about 84% accuracy. With the PCA and the wrapper algorithm, I was able to achieve about 84% accuracy. I think that because the dataset has fewer attributes and all the attributes were important to the classification, that the PCA and the wrapper algorithm didn't remove any attributes; so the accuracy stayed the same.