Danny North

3/16/16

CS 478

Dr. Martinez

Instance Based Learning Lab (KNN)

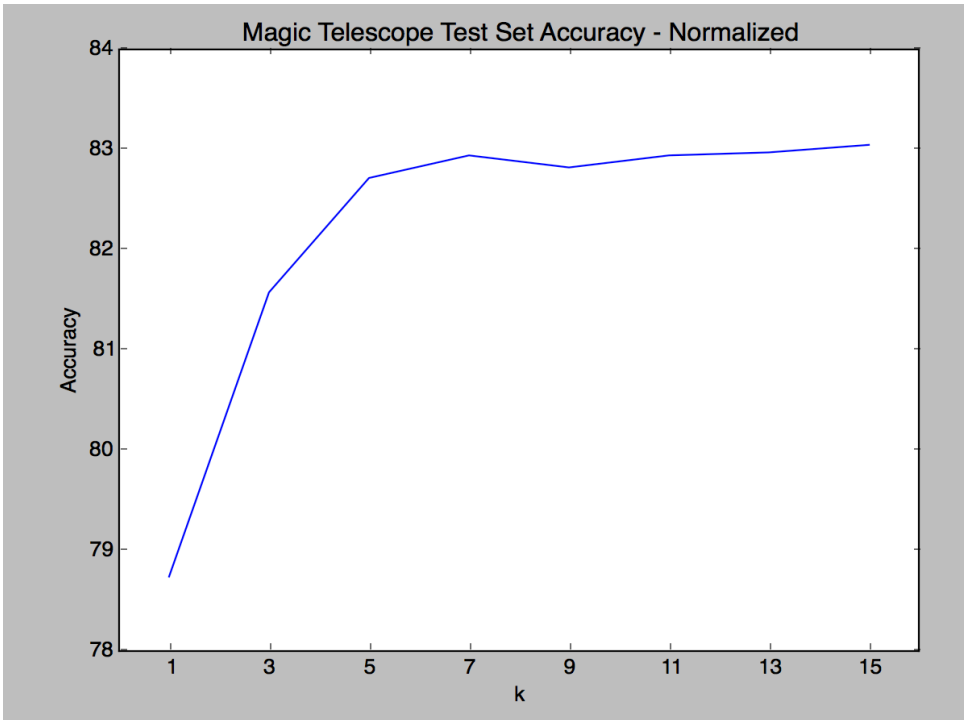
I. Magic Telescope Dataset

Below is the data for the knn algorithm with k=3 without distance weighting.

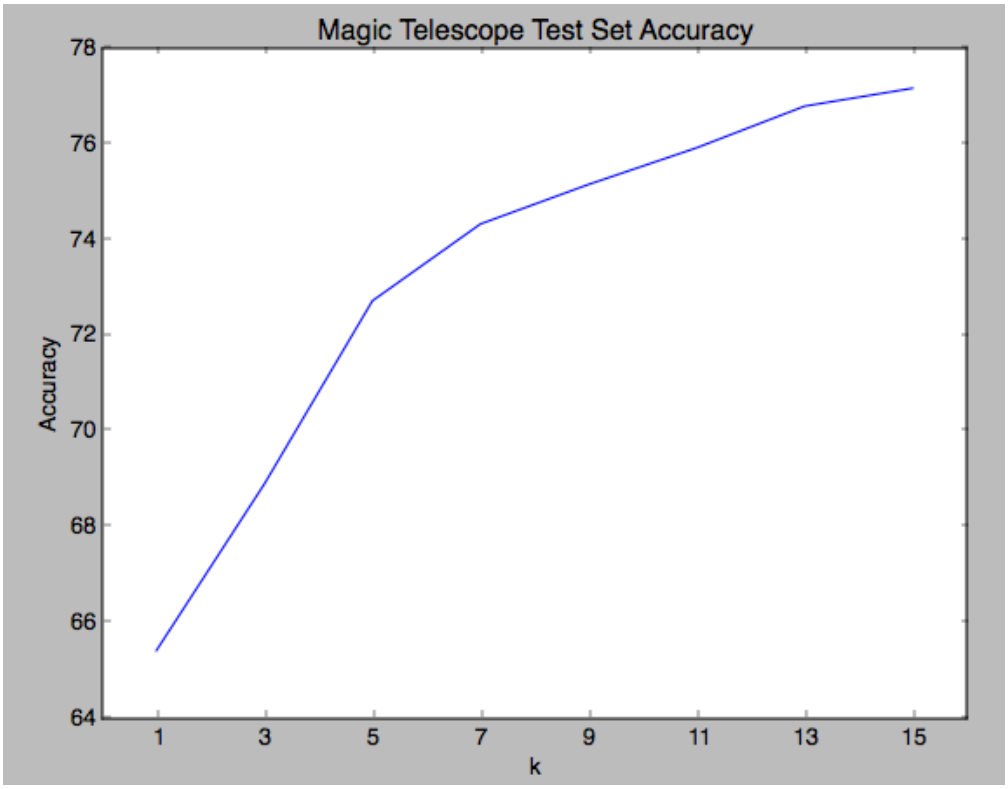
Normalized: `k: 3, accuracy: 81.578%`

Non-Normalized: `k: 3, accuracy: 80.828%`

As you can see, normalizing the dataset works with slightly greater accuracy at k=3. This is because some attributes can have more weight than necessary with non-normalized data. An attribute with a range of 100 – 1000 would completely wash out an attribute with a range of 0 – 0.1. However, a larger range does not necessarily make the first attribute more important. Which is why normalization can make a big difference with a dataset. Now I will test multiple values of k to see which is the most accurate.



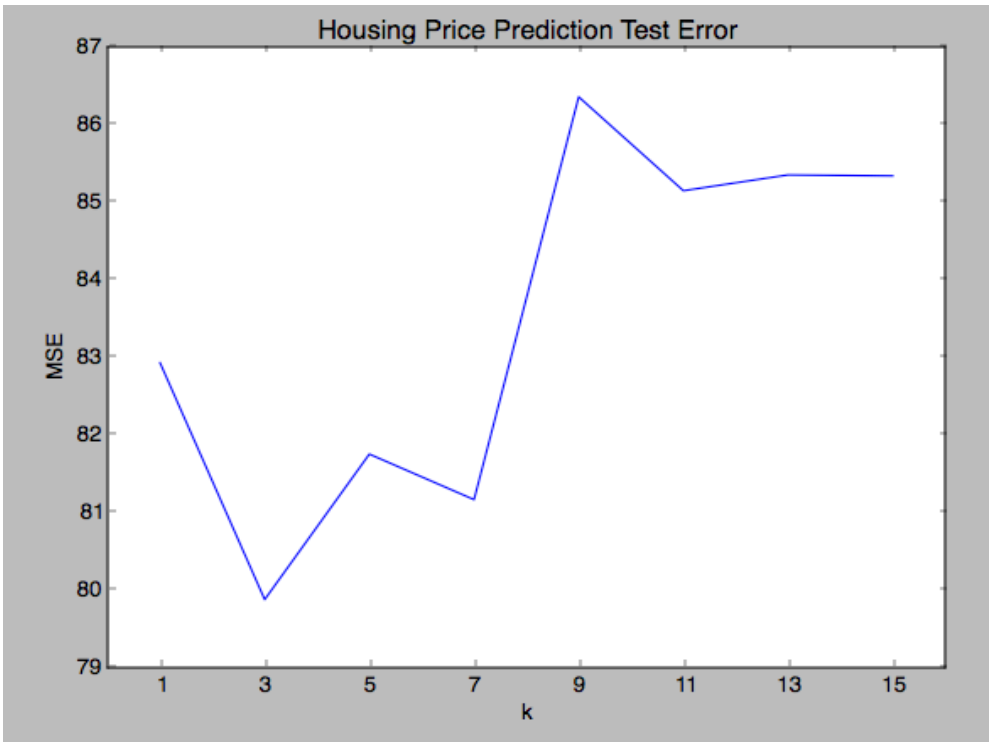
It seems as though 15 is the best for k-value for this dataset. However, we notice that we can have a k-value of 7 and not lose much accuracy. This might be a good choice if computation power is limited.



Attempting to do ***distance-weighted voting*** on this dataset didn't seem to provide as good results as choosing the most common did. Here is a graph with the different k-values and while the accuracy does improve at a similar rate to its non-weighted counterpart, its accuracy starts at a lower value.

II. **House Pricing Dataset**

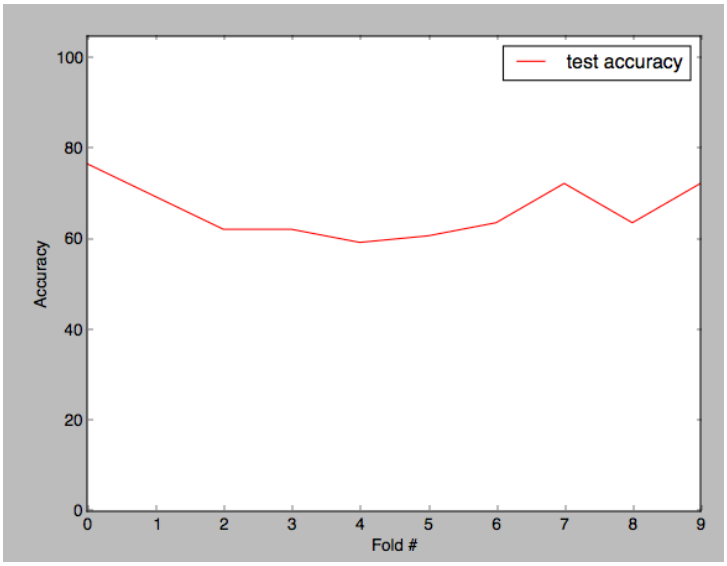
Testing the regression variant of the algorithm on the house pricing dataset gave the following Mean Square Error with normalized data results shown in the graph below:



The prediction gets stronger, or with less error, as the k-value grows. In this case, a k-value of 3 is the best. Here is a graph with the same data but now including ***distance-weighted voting***. The graph is very similar in shape, we just seem to get a sharper downward dip, or better accuracy, when the accuracy is already at its best.

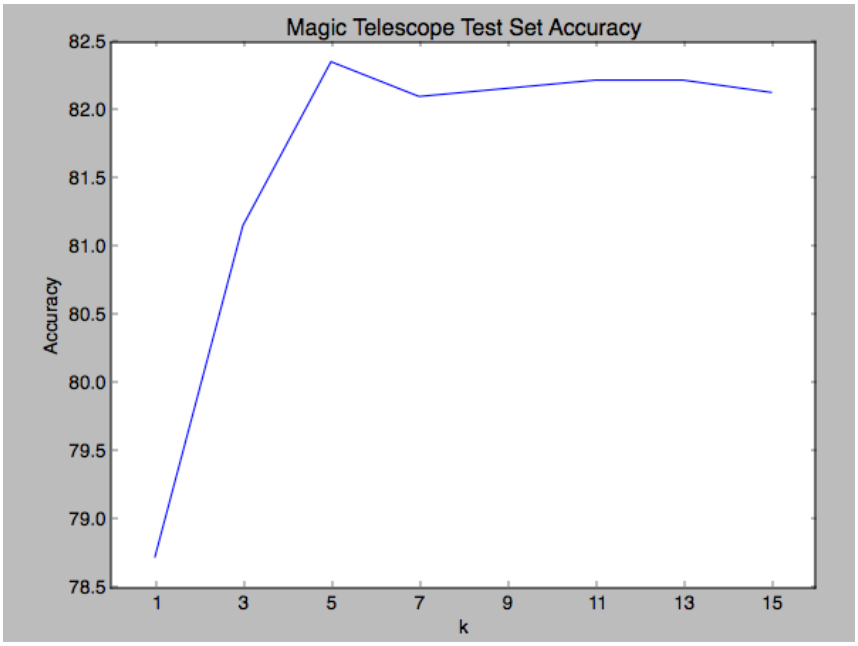
III. **Credit Approval Dataset**

I didn't do very much tweaking for the credit approval dataset other than to deal with unknown values and to set a k-value. I found that a k-value of 13 worked the best for this dataset (a k-value of 620 was actually pretty similar in accuracy so that's something interesting to note.) Using a 10-fold cross validation set, I was able to get these results with an overall accuracy of 68.26%. This was a little on the low side, and it might be because of the way I handled the unknown data. I decided to set the unknown data to 1. While I did experiment with other values such as 0, 1000, 60, etc. I realized that just putting a fixed value did not change the accuracy much. In the future I might try a function that more closely resembles a mode for unknown values. However, here are the results: (The graph is smaller in order to fit it on this page)



IV. Experiment

The experiment I chose to do for this project was to change the way the data was being normalized and see how it had an effect on the accuracy. So for this I disregarded the true normalization and instead only divided each item in the column vectors by its maximum without first subtracting by the minimum. This had a surprising effect because the accuracy did not change much, as can be seen by the graph below.



While I did also try a reduction algorithm, the results were not as impressive as I would have liked. The reduction algorithm I implemented was a decremental algorithm where I removed training points if all of its surrounding k-neighbors had the same target as its own target. I tried this with a few different k-values and it didn't seem to affect the outcome much. The results are shown below with different k-test values but a fixed k-training value of 7. The data points went from 12,354 to 11,746 and the accuracies were as follows: