* 1. K-Nearest Neighbor classification:

1. Description of algorithm:

K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions).

A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function.

Since the three distance measures (Manhattan, Euclidean and Minkowski) are only valid for continuous variables, in the instance of categorical variables the Hamming distance must be used.

It also brings up the issue of standardization of the numerical variables between 0 and 1 when there is a mixture of numerical and categorical variables in the dataset.

1. Data Pre-processing:

Based on the above constraints, the car auction data has to be processed to satisfy the requirement. Firstly, we use Weka to smoothing the missing values by either assigning the average if it is a numeric attribute or assigning the mode of the value if it is a category value. Afterwards, those invalid or NULL value of the numeric data are filled with average of the current attribute to minimize the prediction error.

1. Classification Implementation:

The KNN algorithm is implemented in Python 2.7, with the external import of “numpy” library only.

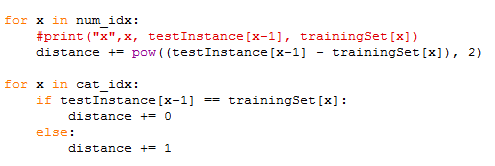
The whole algorithm is broken into four parts as below:

1. Handle Data: Open the dataset from CSV to store into test/train datasets (list)
2. Similarity: Calculate the distance between two data instances.
3. Neighbors: Locate k most similar data instances.
4. Response: Generate a response (Classify by vote from k neighbors) from a set of data instances.

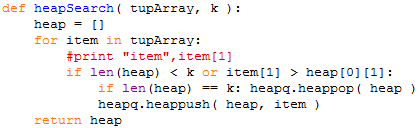
For the 1st part, the key thing is to differentiate the numerical data and categorical data (nominal data). For numerical data, it is converted to float; otherwise, they are stored as string. Also, since the distances between both numerical and nominal data have to be compatible, the normalization of the numeric data is utilized as below:



For the similarity part, I define 2 distance calculating functions, namely Euclidean Distance and Hamming Distance functions:



As for finding k nearest neighbors, a heapsort is used to find the k minimal distances with the dataset entry as a form of tuple and return a list of such tuples.



After that, the getResponse function will count the vote of classification labels from the k neighbors and out put the result.

4. Chosen of Parameters:

KNN algorithm is known as a non-parametric algorithm. The only parameter that need to be set is k. After comparing different results based on Weka, we choose to use k = 3.

5. Improvement Analysis:

The main problem of KNN is the efficiency with big size of training data and memory consuming. As the memory is not the focus for our improvement, we paid great attention on how to improve time complexity.

One thing we have improved is the sorting of distance. Since we only need to find the k nearest neighbor, we replace the original O(nlogn) sort with heap sort with running time O(nlogk). We also eliminate noise data by doing data preprocessing.

Based on research, we find that a K-D tree data structure can be utilized due to the curse of dimensionality. With that, the distance in 32 dimensions can be calculated faster by pruning the tree.

Another potential improvement that we yet have time to do is do PCA on the training data. It can on the one hand improve the predication accuracy and on the other head reduce time complexity.

Distribution of labor:

We have a team of 2 people, Guangyu Zhou an Qi Zhang. We worked together on discuss and analyze how to smooth the data and which algorithm to pick up.

After that, each of us implement an algorithm. Qi implemented Naïve Bayes and Guangyu implemented K-Nearest Neighbor, which includes data preprocessing, algorithm learning and coding, evaluation based on Kaggle and improvement.

The report is collaborated with both of us.