Assumptions:

* The tables passed into the stepN functions are cleaned data; NA values have been purged or filled.
* The tables passed into the stepN functions are not normalized.
* The tables passed into the stepN functions have strings as their values.
* step2 bases the normalizing factors purely on the training set. If the instance is outside the min and max, it could skew the results.

To run the program, use python2.7, and make sure the auto-data.txt file is in the working directory.

Step 1: Create a classifier that predicts mpg values using least squares linear regression based on vehicle weight.

Results:

===========================================

STEP 1: Linear Regression MPG Classifier

===========================================

instance: [16.76445652669117, '8', '360.0', '175.0', '3821', '11.0', '73', '1', 'amc ambassador brougham', '4477', 3]

class: 3, actual: 1

instance: [24.95533387353168, '4', '121.0', '76.00', '2511', '18.0', '72', '2', 'volkswagen 411 (sw)', '3275', 6]

class: 6, actual: 5

instance: [18.615219652389484, '6', '250.0', '98.00', '3525', '19.0', '77', '1', 'ford granada', '4209', 4]

class: 4, actual: 4

instance: [30.345056219315282, '4', '76.00', '52.00', '1649', '16.5', '74', '3', 'toyota corona', '3344', 7]

class: 7, actual: 7

instance: [19.17169910572445, '8', '318.0', '150.0', '3436', '11.0', '70', '1', 'plymouth satellite', '2831', 4]

class: 4, actual: 4

Step 2: Create a nearest neighbor classifier for mpg that uses the number of cylinders, weight, and acceleration attributes to predict mpg for k=5.

Results:

===========================================

STEP 2: k=5 Nearest Neighbor MPG Classifier

===========================================

instance: ['12.0', '8', '429.0', '198.0', '4952', '11.5', '73', '1', 'mercury marquis brougham', '5151', 1]

class: 1 actual: 1

instance: ['17.0', '8', '302.0', '140.0', '3449', '10.5', '70', '1', 'ford torino', '2778', 3]

class: 2 actual: 3

instance: ['18.0', '8', '318.0', '150.0', '3436', '11.0', '70', '1', 'plymouth satellite', '2831', 4]

class: 2 actual: 4

instance: ['31.9', '4', '89.00', '71.00', '1925', '14.0', '79', '2', 'vw rabbit custom', '4799', 8]

class: 7 actual: 8

instance: ['20.0', '6', '232.0', '100.0', '2914', '16.0', '75', '1', 'amc gremlin', '2798', 4]

class: 4 actual: 4

Step 3: Computer the multi-class predictive accuracy and error rate of the two classifiers using separate training and test sets. Use a random subsampling approach and a stratified k-fold cross validation approach, both with k=10.

Results:

===========================================

STEP 3: Predictive Accuracy

===========================================

Random Subsample (k=10, 2:1 Train/Test)

Linear Regression: accuracy = 0.4 , error rate = 0.6

k Nearest Neighbors: accuracy = 0.4 error rate = 0.6

Stratified 10-Fold Cross Validation

Linear Regression: accuracy = 0.411067193676 , error rate = 0.588932806324

k Nearest Neighbors: accuracy = 0.695652173913 error rate = 0.304347826087

Step 4: Create confusion matrices for each classifier based on the stratified 10-fold cross validation results.

Results:

===========================================

STEP 4: Confusion Matrices

===========================================

Linear Regression (Stratified 10-Fold Cross Validation Results):

MPG 1 2 3 4 5 6 7 8 9 10 Total Recognition(%)

----- --- --- --- --- --- --- --- --- --- ---- ------- ----------------

1 13 3 6 2 1 0 0 0 0 0 25 52

2 6 9 6 9 2 0 0 0 0 0 32 28.125

3 2 10 7 5 2 0 0 0 0 0 26 26.9231

4 0 1 5 19 22 6 1 0 0 0 54 35.1852

5 0 0 1 7 21 16 1 0 0 0 46 45.6522

6 0 0 0 1 5 14 5 0 0 0 25 56

7 0 0 0 0 3 6 19 0 0 0 28 67.8571

8 0 0 0 0 0 2 13 0 0 0 15 0

9 0 0 0 0 0 0 2 0 0 0 2 0

10 0 0 0 0 0 0 0 0 0 0 1 0

k = 5 Nearest Neighbor (Stratified 10-Fold Cross Validation Results):

MPG 1 2 3 4 5 6 7 8 9 10 Total Recognition(%)

----- --- --- --- --- --- --- --- --- --- ---- ------- ----------------

1 20 5 0 0 0 0 0 0 0 0 25 80

2 6 22 3 1 0 0 0 0 0 0 32 68.75

3 1 6 11 8 0 0 0 0 0 0 26 42.3077

4 0 2 3 40 9 0 0 0 0 0 54 74.0741

5 0 0 0 11 33 2 0 0 0 0 46 71.7391

6 0 0 0 0 7 14 4 0 0 0 25 56

7 0 0 0 0 1 3 21 3 0 0 28 75

8 0 0 0 0 0 0 5 10 0 0 15 66.6667

9 0 0 0 0 0 0 0 2 0 0 2 0

10 0 0 0 0 0 0 0 0 0 0 1 0