COMP 309
Assignment one

Name: Siwen Feng
ID: 300363512
Dataset: Abalone

2.1: Perform classification using your chosen method on the dataset

Naive Bayes (Bayesian):

```
=== Summary ===

Correctly Classified Instances        1021               24.4434 %
Incorrectly Classified Instances      3156               75.5566 %
Kappa statistic                          0.1466
Mean absolute error                      0.0574
Root mean squared error                  0.1906
Relative absolute error                 89.743  %
Root relative squared error            106.5697 %
Total Number of Instances             4177

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     1
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     2
                 0.867    0.010    0.245      0.867    0.382      0.458    0.995     0.311     3
                 0.421    0.015    0.286      0.421    0.340      0.336    0.979     0.270     4
                 0.443    0.045    0.219      0.443    0.293      0.284    0.939     0.237     5
                 0.371    0.070    0.259      0.371    0.305      0.255    0.862     0.249     6
                 0.327    0.111    0.233      0.327    0.272      0.186    0.803     0.232     7
                 0.257    0.123    0.247      0.257    0.252      0.132    0.704     0.240     8
                 0.284    0.178    0.240      0.284    0.260      0.100    0.647     0.239     9
                 0.203    0.112    0.246      0.203    0.223      0.099    0.657     0.224     10
                 0.458    0.174    0.258      0.458    0.330      0.225    0.725     0.260     11
                 0.007    0.001    0.286      0.007    0.015      0.037    0.680     0.128     12
                 0.000    0.000    ?          0.000    ?          ?        0.642     0.070     13
                 0.000    0.000    ?          0.000    ?          ?        0.664     0.068     14
                 0.000    0.000    ?          0.000    ?          ?        0.687     0.075     15
                 0.000    0.000    0.000      0.000    0.000      -0.002   0.750     0.081     16
                 0.052    0.003    0.214      0.052    0.083      0.099    0.733     0.055     17
                 0.000    0.000    ?          0.000    ?          ?        0.754     0.060     18
                 0.000    0.000    ?          0.000    ?          ?        0.712     0.016     19
                 0.077    0.004    0.118      0.077    0.093      0.091    0.812     0.035     20
                 0.000    0.000    ?          0.000    ?          ?        0.834     0.089     21
                 0.000    0.000    ?          0.000    ?          ?        0.813     0.080     22
                 0.000    0.000    ?          0.000    ?          ?        0.746     0.117     23
                 1.000    0.001    0.286      1.000    0.444      0.534    1.000     1.000     24
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     25
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     26
                 0.500    0.009    0.027      0.500    0.051      0.115    0.927     0.252     27
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     29
Weighted Avg.    0.244    0.100    ?          0.244    ?          ?        0.718     0.206

=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   k   l   m   n   o   p   q   r   s   t   u   v   w   x   y   z  aa  ab   <-- classified
   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   a = 1
   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   b = 2
   0   0  13   2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   c = 3
   0   0  25  24   7   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   d = 4
   0   0  11  33  51  17   3   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   e = 5
   0   0   3  17  79  96  45  12   5   1   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   f = 6
   0   0   1   6  55 119 128  50  25   4   3   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   g = 7
   0   0   0   2  18  63 143 146 132  40  24   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   h = 8
   0   0   0   0  11  34  93 124 196 112 119   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   i = 9
   0   0   0   0   7  24  51  76 150 129 195   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2   0 |   j = 10
   0   0   0   0   4   8  32  54  92  65 223   3   0   0   0   0   0   0   0   0   0   0   2   0   0   4   0   0 |   k = 11
   0   0   0   0   0   6  18  42  50  41  99   2   0   0   0   0   0   0   0   3   0   0   0   0   0   6   0   0 |   l = 12
   0   0   0   0   1   2  15  26  54  31  68   0   0   0   0   0   2   0   0   2   0   0   0   1   0   0   1   0 |   m = 13
   0   0   0   0   0   1   6  21  31  26  31   1   0   0   0   0   3   0   0   1   0   0   0   1   0   0   4   0 |   n = 14
   0   0   0   0   0   0   5  19  21  30  24   0   0   0   0   0   1   0   0   1   0   0   0   1   0   0   1   0 |   o = 15
   0   0   0   0   0   0   6   4  17  10  21   0   0   0   0   0   2   0   0   3   0   0   0   0   0   0   4   0 |   p = 16
   0   0   0   0   0   0   0   5  19   9  15   0   0   0   0   0   3   0   0   1   0   0   0   0   0   0   6   0 |   q = 17
   0   0   0   0   0   0   3   4   5   9  15   0   0   0   0   0   1   0   0   2   0   0   0   0   0   0   3   0 |   r = 18
   0   0   0   0   0   0   1   2  10   7  12   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   s = 19
   0   0   0   0   0   0   0   2   3   9   7   0   0   0   0   0   0   0   0   2   0   0   0   0   0   0   3   0 |   t = 20
   0   0   0   0   0   0   0   2   2   1   5   0   0   0   0   0   1   0   0   2   0   0   0   0   0   0   1   0 |   u = 21
   0   0   0   0   0   0   0   1   2   0   1   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   1   0 |   v = 22
   0   0   0   0   0   0   0   1   3   1   3   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   w = 23
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2   0   0   0   0 |   x = 24
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0 |   y = 25
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0 |   z = 26
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   1   0 |  aa = 27
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1 |  ab = 29
```

Important aspects :

- *Correctly classified instances*
- *Incorrectly classified instances*

These two aspects indicated how well Naive Bayes classify the Abalone dataset. Based on the results, we can find that it leads to very low correctness and underfitting may occur.

Multilayer perceptron (Connectionist):

```
=== Summary ===

Correctly Classified Instances        1266               30.3088 %
Incorrectly Classified Instances      2911               69.6912 %
Kappa statistic                          0.2045
Mean absolute error                      0.0567
Root mean squared error                  0.1686
Relative absolute error                 88.5453 %
Root relative squared error             94.252  %
Total Number of Instances             4177
```

```
=== Detailed Accuracy By Class ===

                 TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
                 0.000     0.000     ?           0.000   ?          ?        0.175     0.000     1
                 0.000     0.000     ?           0.000   ?          ?        0.222     0.000     2
                 0.400     0.003     0.353       0.400   0.375      0.373    0.994     0.335     3
                 0.246     0.003     0.519       0.246   0.333      0.351    0.987     0.428     4
                 0.409     0.017     0.398       0.409   0.403      0.386    0.967     0.408     5
                 0.340     0.034     0.400       0.340   0.367      0.330    0.914     0.356     6
                 0.230     0.038     0.385       0.230   0.288      0.243    0.876     0.336     7
                 0.577     0.185     0.329       0.577   0.419      0.316    0.790     0.346     8
                 0.254     0.104     0.326       0.254   0.285      0.167    0.744     0.325     9
                 0.543     0.293     0.249       0.543   0.341      0.190    0.720     0.273     10
                 0.191     0.044     0.366       0.191   0.251      0.198    0.775     0.307     11
                 0.045     0.005     0.375       0.045   0.080      0.112    0.739     0.169     12
                 0.172     0.047     0.158       0.172   0.165      0.120    0.785     0.152     13
                 0.008     0.000     0.500       0.008   0.016      0.060    0.807     0.106     14
                 0.078     0.006     0.258       0.078   0.119      0.130    0.826     0.120     15
                 0.134     0.005     0.300       0.134   0.186      0.192    0.846     0.175     16
                 0.224     0.011     0.224       0.224   0.224      0.213    0.896     0.178     17
                 0.071     0.001     0.429       0.071   0.122      0.172    0.835     0.146     18
                 0.000     0.000     0.000       0.000   0.000      −0.002   0.871     0.079     19
                 0.000     0.002     0.000       0.000   0.000      −0.003   0.936     0.076     20
                 0.000     0.000     ?           0.000   ?          ?        0.202     0.002     21
                 0.000     0.000     ?           0.000   ?          ?        0.203     0.001     22
                 0.000     0.000     ?           0.000   ?          ?        0.226     0.001     23
                 0.000     0.000     ?           0.000   ?          ?        0.016     0.000     24
                 0.000     0.000     ?           0.000   ?          ?        0.078     0.000     25
                 0.000     0.000     ?           0.000   ?          ?        0.130     0.000     26
                 0.000     0.000     ?           0.000   ?          ?        0.053     0.000     27
                 0.000     0.000     ?           0.000   ?          ?        0.049     0.000     29
Weighted Avg.    0.303     0.101     ?           0.303   ?          ?        0.791     0.282
```

```
=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   k   l   m   n   o   p   q   r   s   t   u   v   w   x   y   z  aa  ab   <-- classified as
   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   a = 1
   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   b = 2
   0   0   6   5   4   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   c = 3
   0   0   5  14  31   3   4   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   d = 4
   0   0   3   6  47  40  16   2   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   e = 5
   0   0   0   2  25  88  69  59  11   3   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   f = 6
   0   0   1   0   8  61  90 188  23  19   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   g = 7
   0   0   0   0   2  18  34 328  90  91   4   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   h = 8
   0   0   0   0   1   7   8 218 175 246  17   1  14   0   1   0   1   0   0   0   0   0   0   0   0   0   0   0 |   i = 9
   0   0   0   0   0   1   7  96 110 344  47   3  21   1   1   1   1   1   0   0   0   0   0   0   0   0   0   0 |   j = 10
   0   0   0   0   0   1   3  43  58 259  93   3  21   0   3   0   2   1   0   0   0   0   0   0   0   0   0   0 |   k = 11
   0   0   0   0   0   0   2  24  32 136  34  12  20   0   0   2   3   1   0   1   0   0   0   0   0   0   0   0 |   l = 12
   0   0   0   0   0   1   1  15  14  95  25   2  35   0   2   3   5   0   1   4   0   0   0   0   0   0   0   0 |   m = 13
   0   0   0   0   0   0   0   8   6  66   7   1  23   1   2   3   8   0   1   0   0   0   0   0   0   0   0   0 |   n = 14
   0   0   0   0   0   0   0  10   8  45   5   0  21   0   8   0   6   0   0   0   0   0   0   0   0   0   0   0 |   o = 15
   0   0   0   0   0   0   0   2   3  26   4   3  11   0   3   9   5   1   0   0   0   0   0   0   0   0   0   0 |   p = 16
   0   0   0   0   0   0   0   1   1  21   4   2  12   0   3   1  13   0   0   0   0   0   0   0   0   0   0   0 |   q = 17
   0   0   0   0   0   0   0   2   2  13   5   1  10   0   1   1   4   3   0   0   0   0   0   0   0   0   0   0 |   r = 18
   0   0   0   0   0   0   0   0   2   9   4   0  13   0   2   0   2   0   0   0   0   0   0   0   0   0   0   0 |   s = 19
   0   0   0   0   0   0   0   0   1   4   1   3   7   0   3   4   3   0   0   0   0   0   0   0   0   0   0   0 |   t = 20
   0   0   0   0   0   0   0   0   0   4   1   1   4   0   0   2   2   0   0   0   0   0   0   0   0   0   0   0 |   u = 21
   0   0   0   0   0   0   0   0   0   0   0   0   3   0   2   0   1   0   0   0   0   0   0   0   0   0   0   0 |   v = 22
   0   0   0   0   0   0   0   0   0   2   1   0   4   0   0   1   1   0   0   0   0   0   0   0   0   0   0   0 |   w = 23
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0 |   x = 24
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0 |   y = 25
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0 |   z = 26
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   1   0   0   0   0   0   0   0 |  aa = 27
   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  ab = 29
```

Important aspects :

- *Correctly classified instances*
- *Incorrectly classified instances*

These two aspects indicated how well Multilayer perceptron classify the Abalone dataset. Based on the results we can find that it leads to very low correctness and underfitting may occur.

IBk-k-nearest neighbors(Analogizer):

```
=== Summary ===

Correctly Classified Instances        4177               100      %
Incorrectly Classified Instances         0                 0      %
Kappa statistic                          1
Mean absolute error                      0.0005
Root mean squared error                  0.0012
Relative absolute error                  0.7167 %
Root relative squared error              0.691  %
Total Number of Instances             4177
```

```
=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     1
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     2
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     3
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     4
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     5
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     6
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     7
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     8
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     9
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     10
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     11
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     12
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     13
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     14
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     15
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     16
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     17
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     18
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     19
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     20
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     21
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     22
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     23
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     24
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     25
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     26
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     27
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     29
Weighted Avg.    1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000
```

```
=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   k   l   m   n   o   p   q   r   s   t   u   v   w   x   y   z  aa  ab   <-- classified as
   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   a = 1
   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   b = 2
   0   0  15   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   c = 3
   0   0   0  57   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   d = 4
   0   0   0   0 115   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   e = 5
   0   0   0   0   0 259   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   f = 6
   0   0   0   0   0   0 391   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   g = 7
   0   0   0   0   0   0   0 568   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   h = 8
   0   0   0   0   0   0   0   0 689   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   i = 9
   0   0   0   0   0   0   0   0   0 634   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   j = 10
   0   0   0   0   0   0   0   0   0   0 487   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   k = 11
   0   0   0   0   0   0   0   0   0   0   0 267   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   l = 12
   0   0   0   0   0   0   0   0   0   0   0   0 203   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   m = 13
   0   0   0   0   0   0   0   0   0   0   0   0   0 126   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   n = 14
   0   0   0   0   0   0   0   0   0   0   0   0   0   0 103   0   0   0   0   0   0   0   0   0   0   0   0   0 |   o = 15
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  67   0   0   0   0   0   0   0   0   0   0   0   0 |   p = 16
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  58   0   0   0   0   0   0   0   0   0   0   0 |   q = 17
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  42   0   0   0   0   0   0   0   0   0   0 |   r = 18
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  32   0   0   0   0   0   0   0   0   0 |   s = 19
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  26   0   0   0   0   0   0   0   0 |   t = 20
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  14   0   0   0   0   0   0   0 |   u = 21
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   6   0   0   0   0   0   0 |   v = 22
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   9   0   0   0   0   0 |   w = 23
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2   0   0   0   0 |   x = 24
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0 |   y = 25
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0 |   z = 26
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2   0 |  aa = 27
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1 |  ab = 29
```

Important aspects :

- *Correctly classified instances*
- *Incorrectly classified instances*

KNN shows a strange result that correctness is 100% and incorrectness is 0%. However, this scenario is highly possible when overfitting occurs.

J48—decision tree (Symbolist):

```
=== Summary ===

Correctly Classified Instances        3163               75.7242 %
Incorrectly Classified Instances      1014               24.2758 %
Kappa statistic                          0.7271
Mean absolute error                      0.023
Root mean squared error                  0.1072
Relative absolute error                 35.9313 %
Root relative squared error             59.958  %
Total Number of Instances             4177

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
               1.000    0.000    0.500      1.000    0.667      0.707    1.000     0.500     1
               0.000    0.000    ?          0.000    ?          ?        1.000     0.500     2
               0.400    0.000    0.857      0.400    0.545      0.585    0.998     0.627     3
               0.789    0.006    0.662      0.789    0.720      0.719    0.996     0.738     4
               0.748    0.009    0.705      0.748    0.726      0.718    0.994     0.818     5
               0.768    0.012    0.809      0.768    0.788      0.775    0.991     0.880     6
               0.808    0.027    0.754      0.808    0.780      0.757    0.985     0.858     7
               0.849    0.041    0.764      0.849    0.804      0.773    0.982     0.884     8
               0.821    0.048    0.773      0.821    0.797      0.755    0.975     0.873     9
               0.806    0.055    0.725      0.806    0.763      0.720    0.971     0.835     10
               0.782    0.033    0.757      0.782    0.770      0.739    0.980     0.854     11
               0.678    0.014    0.764      0.678    0.718      0.702    0.986     0.822     12
               0.695    0.009    0.801      0.695    0.744      0.734    0.992     0.853     13
               0.532    0.006    0.736      0.532    0.618      0.616    0.990     0.742     14
               0.573    0.004    0.776      0.573    0.659      0.660    0.994     0.786     15
               0.716    0.004    0.750      0.716    0.733      0.729    0.997     0.830     16
               0.552    0.003    0.727      0.552    0.627      0.629    0.996     0.755     17
               0.238    0.001    0.667      0.238    0.351      0.395    0.994     0.544     18
               0.375    0.001    0.800      0.375    0.511      0.546    0.997     0.682     19
               0.308    0.001    0.727      0.308    0.432      0.471    0.997     0.599     20
               0.429    0.000    0.857      0.429    0.571      0.605    0.999     0.759     21
               0.333    0.000    1.000      0.333    0.500      0.577    1.000     0.722     22
               0.444    0.000    1.000      0.444    0.615      0.666    1.000     0.805     23
               0.000    0.000    ?          0.000    ?          ?        1.000     0.500     24
               0.000    0.000    ?          0.000    ?          ?        0.999     0.167     25
               0.000    0.000    ?          0.000    ?          ?        1.000     0.500     26
               0.000    0.000    ?          0.000    ?          ?        1.000     0.333     27
               0.000    0.000    ?          0.000    ?          ?        1.000     0.500     29
Weighted Avg.  0.757    0.031    ?          0.757    ?          ?        0.982     0.840

=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   k   l   m   n   o   p   q   r   s   t   u   v   w   x   y   z  aa  ab   <-- classified as
   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   a = 1
   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   b = 2
   0   0   6   8   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   c = 3
   0   0   0  45   9   2   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   d = 4
   0   0   0  10  86   7   7   4   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   e = 5
   0   0   1   2  15 199  28   6   2   4   1   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   f = 6
   0   0   0   1   1  18 316  20  17  11   4   1   1   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   g = 7
   0   0   0   2   3   8  20 482  24  18   5   4   0   0   1   0   1   0   0   0   0   0   0   0   0   0   0   0 |   h = 8
   0   0   0   0   4   5  20  30 566  32  17   8   2   1   2   1   1   0   0   0   0   0   0   0   0   0   0   0 |   i = 9
   0   0   0   0   0   2   8  29  46 511  25  10   0   1   0   0   2   0   0   0   0   0   0   0   0   0   0   0 |   j = 10
   0   0   0   0   2   2   7  19  24  39 381   6   2   2   1   0   1   0   1   0   0   0   0   0   0   0   0   0 |   k = 11
   0   0   0   0   0   1   4   7  17  29  20 181   4   3   0   0   0   0   1   0   0   0   0   0   0   0   0   0 |   l = 12
   0   0   0   0   1   1   3  11  13  14  14   3 141   0   0   1   0   1   0   0   0   0   0   0   0   0   0   0 |   m = 13
   0   0   0   0   0   1   5   4   4  12  11   5   9  67   4   1   2   0   0   1   0   0   0   0   0   0   0   0 |   n = 14
   0   0   0   0   0   0   1   5   7  12   7   5   3   3  59   0   0   0   0   1   0   0   0   0   0   0   0   0 |   o = 15
   0   0   0   0   0   0   0   2   3   3   2   1   5   1   1  48   0   1   0   0   0   0   0   0   0   0   0   0 |   p = 16
   0   0   0   0   0   0   0   3   0   6   4   4   2   2   3   2  32   0   0   0   0   0   0   0   0   0   0   0 |   q = 17
   0   0   0   0   0   0   0   4   5   3   6   2   2   2   3   4   0  10   0   1   0   0   0   0   0   0   0   0 |   r = 18
   0   0   0   0   0   0   0   2   2   5   2   2   3   1   0   2   1   0  12   0   0   0   0   0   0   0   0   0 |   s = 19
   0   0   0   0   0   0   0   2   1   3   2   2   2   1   0   2   0   2   0   8   1   0   0   0   0   0   0   0 |   t = 20
   0   0   0   0   0   0   0   0   0   0   2   0   0   1   2   1   2   0   0   0   6   0   0   0   0   0   0   0 |   u = 21
   0   0   0   0   0   0   0   0   0   1   0   1   0   1   0   0   1   0   0   0   0   2   0   0   0   0   0   0 |   v = 22
   0   0   0   0   0   0   0   0   0   0   0   1   0   1   0   1   1   1   0   0   0   0   4   0   0   0   0   0 |   w = 23
   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   1   0   0   0   0   0   0   0   0   0 |   x = 24
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0 |   y = 25
   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |   z = 26
   0   0   0   0   0   0   0   0   0   0   0   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  aa = 27
   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  ab = 29
```

Important aspects :

- *Correctly classified instances*
- *Incorrectly classified instances*

Based on the given result, the decision tree had the best performance when we just use a single training set. 75% correctness is not excellent but reasonable.

2.1.2 Why each technique was selected

In the following part, I will give a detailed discussion regarding the four techniques I talked above. None of the four techniques is the master algorithm, thus, they are pointing to a precisely different direction. I will give a detailed explanation of all four techniques.

Naive Bayes:

Description and representation:

Naïve Bayes is a probability-based algorithm and it is based on the Bayes's theorem. In the classification process, the algorithm will choose the best result which is with the highest probability. However, an important criterion of success is that all features should be independent with each other. Representation of Bayesian can be graphical models. The whole algorithm uses the Bayesian formula to calculate the probability. Therefore, it belongs to the **Bayesian** family.

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$

Bayesian formula

Evaluation and optimization:

The posterior probability is an effective way to evaluate Bayesian. The performance of the algorithm depends on the value of posterior probability. The performance and posterior probability have a positive correlation. The higher posterior probability, the better the performance we can get. Hence, we can use probabilistic inference to optimize the result. The probability will alter when there is more evidence.

Multilayer Perceptron:

Description and representation:

From the indication of the name, I found that this algorithm is specific and an upgraded version of the perceptron. Perceptron is a feedforward neural network algorithm. We can receive a group of the output values from loading a set of input values. Multilayer perceptron means that the model contains more than 1 hidden layer. Meanwhile, backpropagation is the technique to train the data. In the real world, multilayer perceptron is trying to simulate how the brain works. The characteristics of this technique imply that it belongs to the **connectionist.**

Evaluation and optimization:

There are two stopping criteria: 1. The epochs of MLP has reached the limited number. 2. The result which generated by MLP satisfied the threshold (the Squared error is smaller than a value which is chosen by you). To avoiding overfitting, validation control is commonly implemented.

J48 – Decision Tree:

Description and representation:

J48 is a practical application of the decision tree, the predictions can be obtained after we input related symbols at the beginning. We can get an outstanding performance when it comes to deal with categorical features. In terms of my dataset abalone, the leaf nodes of the tree are the attribute of abalone's physical measurement. The outcome of the algorithm is a logical inference, hence, it a **symbolist** algorithm.

Evaluation and optimization:

Gini impurity is a typical method to evaluate the performance of the decision tree. It could measure the number of times the data is wrongly labeled. Implementation of Gini impurity could also effectively balance the impurity of children nodes. Prune is an optimizing method for the decision tree. Pruning is cut off from the complex of the subtrees and replaced with a simple tree structure. In the real world, we are not only considering the accuracy of the DT, but we also taking into account the time cost. Pruning could help us find the balance and optimizing point of the algorithm.

$$G = \sum_{i=1}^{C} p(i) * (1 - p(i))$$

Gina impurity formula

IBk – k nearest neighbors:

Description and representation:

KNN is a typical and widely used supervised classification algorithm. KNN will classify an unseen instance based on the K nearest neighbors from the training set. We count the number of the K neighbors and label the instance into the class which is dominant (the largest population of neighbors are from this class). It is a member of **Analogizers**. It is instance-based learning. The algorithm will memorize each instance and classify and label unseen instances by comparing the distance from the nodes.

Evaluation and optimization:

Euclidean distance is a popular and common measuring method for KNN.
To optimize the performance of KNN, we should carefully decide the K value. It can vary from scenarios. We should always choose a suitable K value to maximize performance.
`

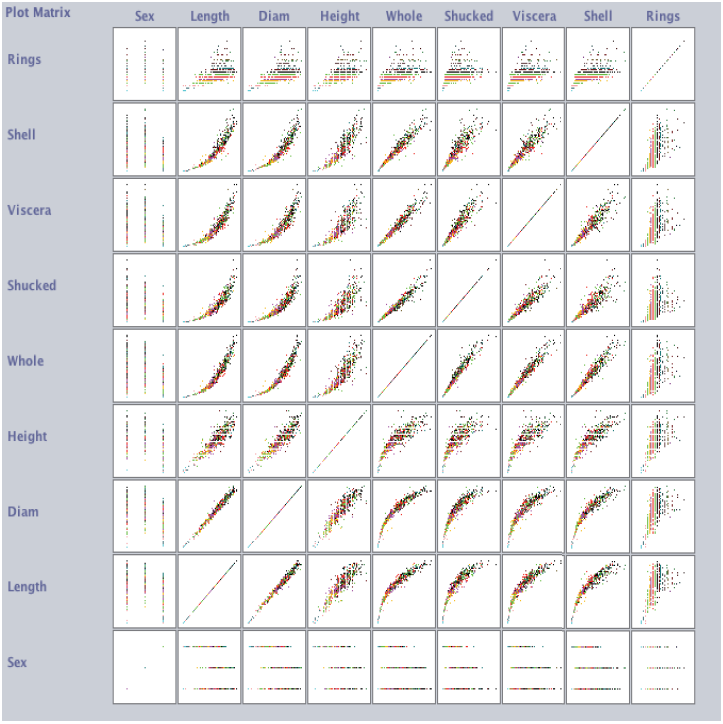
$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$
$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

Euclidean distance formula

2.1.3 techniques analysis based on dataset characteristics

Explaination for data visualisation:

The graph on the right side is the visualization of the Abalone data. As we can see, the graph shows an obvious and special pattern. All the attributes have a strong positive correlation which means that an increase in one attribute can lead to an increase on another attribute. Thus, the best algorithm should match with the data's unique feature.



Why not Naive bayes?

Naïve Bayes is a probability-based algorithm. However, my dataset is a strong positive correlation. In other words, all attributes are dependent with each other. Naïve Bayes needs attributes to be independent. Hence, it explains why Naïve Bayes has such poor performance on the abalone dataset.

Why not Multilayer Perceptron?

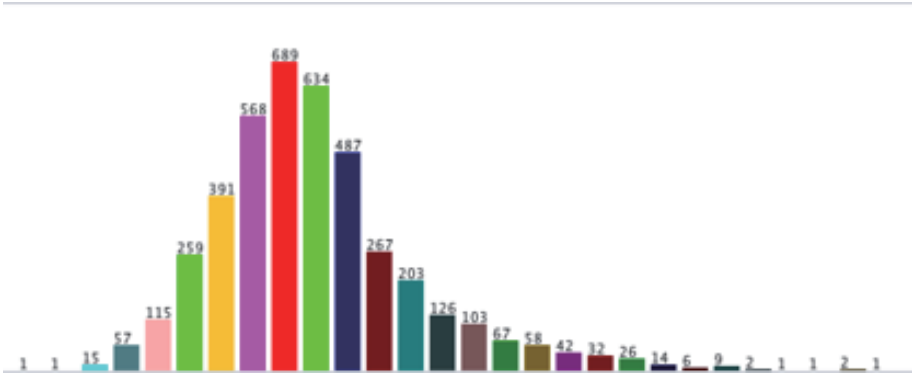
Multilayers perceptron is a member of the Neuron network. In the family of ANN, its characteristics indicate that MP is more suitable for dealing with non-linear correlation dataset. For example, recognition of handwriting, classification of images. Nevertheless, Abalone data is linear correlated dataset which best CNN can give a bad result. As we can see above, we only get roughly 30% correct classification. MP is not an appropriate algorithm for this group of data.

Why not KNN?

An interesting situation occurs when we use KNN to classify the instances. I found that correctness is 100%. The reason why it shows the perfect match because overfitting occurs. Abalone dataset has bad data with low complexity but it contains many noise features. For the initial dataset, the noise hasn't been cleared and we were classifying the data without splitting. KNN might capture the noise of abalone dataset which leads to overfitting. Thus, 100% correctness is inaccurate and unreliable. Overall, KNN is not a suitable algorithm when only one dataset available.

The last algorithm I tried is called J48 which is based on the decision tree algorithm. It has the most outstanding performance amongst 4 algorithms (75.72%). The initial data shows a very imbalanced data.

There are 29 classes, the classes 5-14 of rings contain 90% of instances. However, the decision tree can handle this situation better than the rest of the other three techniques. It is probably the reason the decision tree has the best result.



2.2 Consider a Pipeline for Dataset Processing

2.2.1 Business understanding:

Abalone dataset records all physical measurements of the abalone individuals. We can use ML techniques to help to classify the age of the abalone.

- The age of abalone can help sea farming industry to identify whether a abalone has reached a mature age to hunt. Therefore, the sustainability of sea farming could be achieved by stopping the hunt of immature abalones.
- Abalone's age distribution can also help ocean scientist understand the influence brought by climate changes. Thus, the government should alter environmental policy to adapt to the changes.

2.2.2 Data understanding:

Abalone is a common sea creature globally. Abalone dataset is a dataset which contains all measuring information of abalone individual. However, the rings of abalone are the most difficult part to measure. We have explored some existing techniques in part one. In this part, I will try to optimize the dataset to improve performance. Abalone dataset contains some drawbacks which become barriers for getting a better result. To achieve the goal, a few things have to be solved.

1. The dataset is of low complexity but high noise. Some irrelevant attributes should be eliminated.
2. The class of rings has 29 values, too many classes will make the classification process less accurate.
3. The distributions of the representations are imbalanced. Thus, it will also cause a decrease in the accuracy of classification.

2.2.3 Data Preparation:

2.2.3.1 Dealing with noises:

In this dataset, we are predicting the age the Abalone based on the ring numbers. However, 8 attributes may contain some useless data which we can delete, then we can receive concise and clear data.

- Sex is the non-relevant attribute regarding age.
- The whole weight is the only useful attribute for predicting age, other weights can be considered as noises.

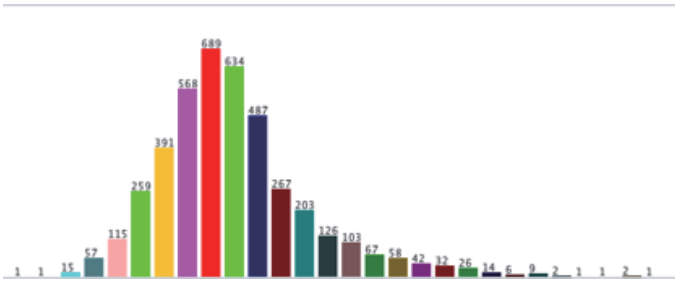
While we used the pipeline, there is a filter called attributes selection. This function can help us to clear out the noises.

2.2.3.2 Simplify Classes:

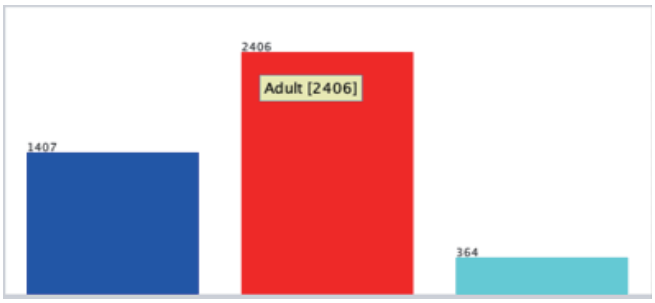
The number of rings varies from 1 to 29. When people try to use the data to estimate the age, class of ring will confuse the user and make classification process hard. For simplifying the class. I categorized 29 classes from rings into 3 distinct classes (Child, Adult, Old-age).

Rings	Age
1 to 7	Child
8 to 14	Adult
15 to 29	Old-age

After implementing changes, we can find a very straightforward difference for distributions of representatives.



Before



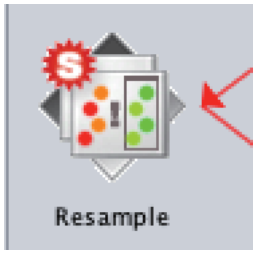
After

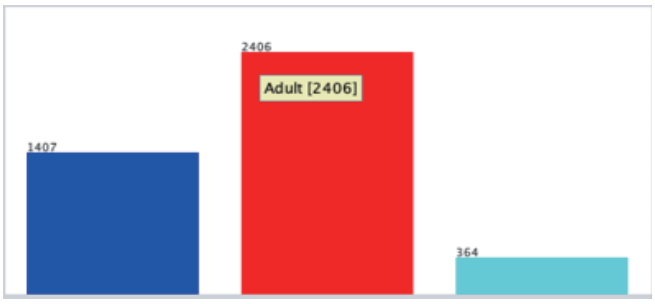
Thus, new class labels bring a clear and even distribution than the original dataset.

2.2.3.3 Dealing with imbalanced data:

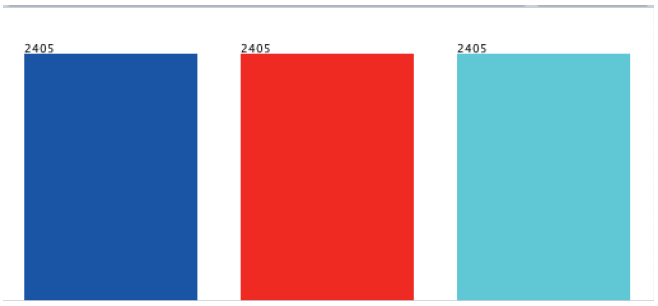
There are many methods that can deal with imbalanced data, In this assignment, I will use **resample** to deal with the dataset and make evenly distributed with a certain ratio. Resample can make sure all the class labels have the same amount of the instances. Thus, a more accurate and sophisticated model can be generated. Then, a better classification can be achieved.

I used a function called resample in the pipeline, I changed the parameter “biasToUniformClass” to 1.0 and “sampleSizePercent” to 172.8%.



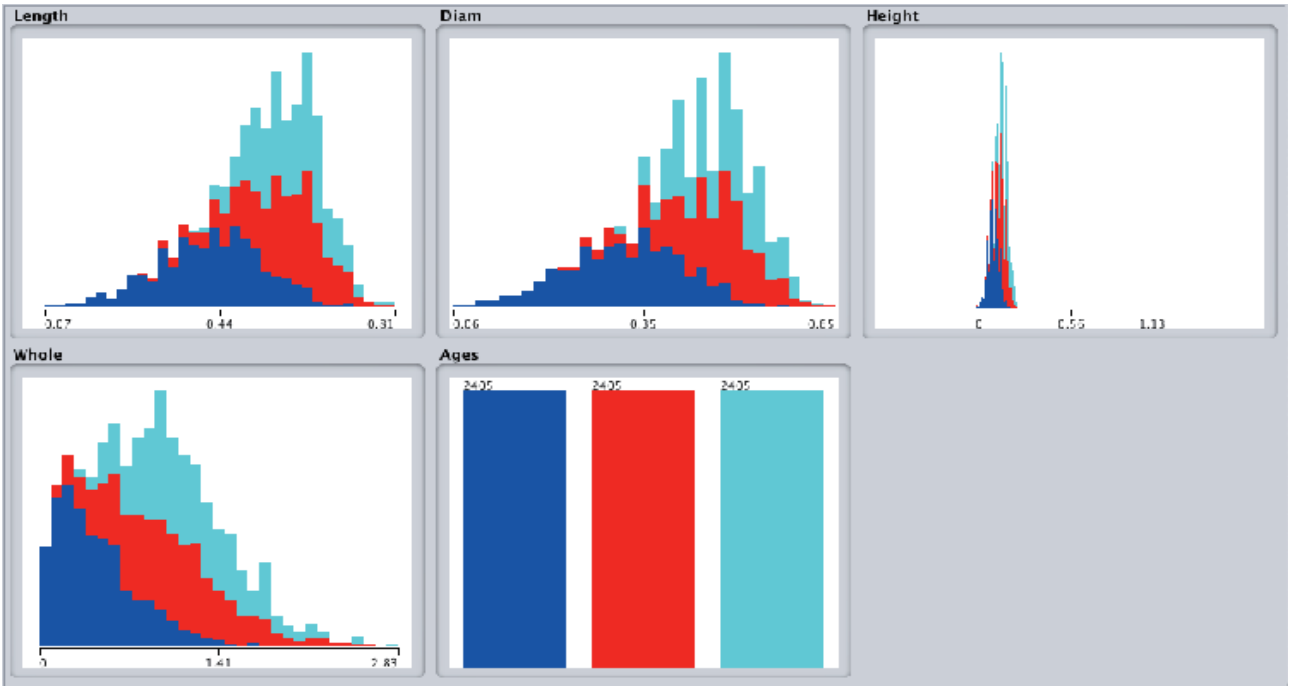


Before resample



After resample

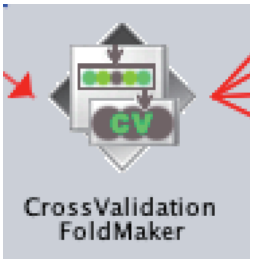
Three steps optimized the initial dataset and make it easily accomplish our goal, the different class label indicates more obvious characteristics which help us to investigate the pattern of the Abalone dataset. The following graph is optimized visualization:



2.2.4 Modelling:

2.2.4.1 K-fold cross validation:

Cross-validation technique has been introduced to solve the problems of limited sample size and overfitting. In the first part of the assignment, we found that KNN has 100% correctness. It is because we only used one training set without any validation. In the pipeline simulation, we used a 10 fold cross-validation to prevent overfitting.



2.2.4.2 Result of Modeling:

All four technologies have improved to varying degrees. And the data obtained is more readable and understandable. Pipeline model simulates the process of that optimized dataset goes through four algorithms while using the cross-validation. We use ClassAssigner to identify the Age as our Class and use resample to balance the dataset.

2.2.5 Evaluation:

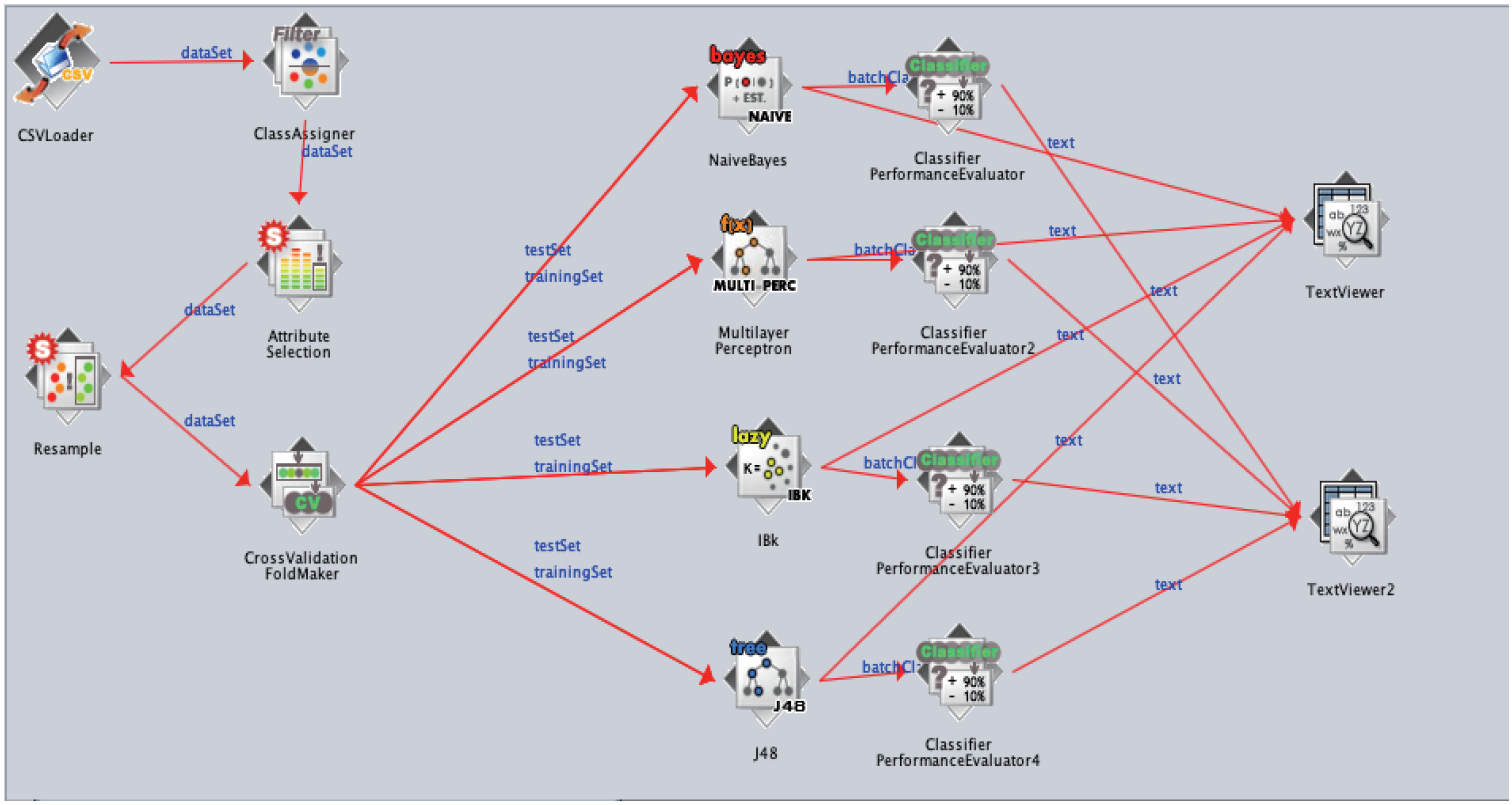
After implementing pipeline methodology and optimized dataset. I received an impressive result – an increase in performance for four techniques. One obvious change for KNN is that we can find the correctness is 92.6% and it is better than any other techniques. 10-folds cross-validation solved the problem of overfitting and brought us a good performance of KNN. Performance of Naive Bayes and Perceptron increased but they still not appropriate for this dataset. Meanwhile, decision tree has a good performance but the tree is way too complex to understand. Thus, the optimization method is needed for it.

2.2.5 Deployment:

The determination of the deployment is up to the fact that can the result generated by the modeling process contribute to achieving the real business goals. Based on the performance, we can find KNN deliver accurate predictions. Therefore, the prediction could offer fisher a chance to protect immature Abalone. In other words, deploying pipeline modeling can be beneficial for the achievement of the business objectives.

2.3 reevaluation for the selected techniques

2.3.1 Pipeline simulation:



2.3.2 Comparision:

Naive Bayes:

```
=== Evaluation result ===

Scheme: NaiveBayes
Relation: Abalone-newage-weka.filters.unsupervised.attribute.ClassAssigner-Clast-weka.filters.supervised.att


Correctly Classified Instances        4068               56.4765 %
Incorrectly Classified Instances      3135               43.5235 %
Kappa statistic                          0.3471
Mean absolute error                      0.3036
Root mean squared error                  0.4273
Relative absolute error                 68.3112 %
Root relative squared error             90.6446 %
Total Number of Instances             7203

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
               0.780     0.155     0.716       0.780   0.747      0.613    0.895     0.830     Child
               0.250     0.196     0.389       0.250   0.305      0.062    0.631     0.394     Adult
               0.664     0.302     0.524       0.664   0.586      0.346    0.770     0.578     Old Age
Weighted Avg.  0.565     0.218     0.543       0.565   0.546      0.340    0.765     0.601

=== Confusion Matrix ===

    a    b    c   <-- classified as
 1873  366  162 |    a = Child
  514  601 1286 |    b = Adult
  230  577 1594 |    c = Old Age
```

Multilayer Perceptron:

```
=== Evaluation result ===

Scheme: MultilayerPerceptron
Options: -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a
Relation: Abalone-newage-weka.filters.unsupervised.attribute.ClassAssigner-Clast-weka.filters.supervised.att


Correctly Classified Instances        4452               61.8076 %
Incorrectly Classified Instances      2751               38.1924 %
Kappa statistic                          0.4271
Mean absolute error                      0.3174
Root mean squared error                  0.4025
Relative absolute error                 71.4234 %
Root relative squared error             85.3829 %
Total Number of Instances             7203

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.740    0.117    0.759      0.740    0.750      0.627    0.893     0.812     Child
                 0.484    0.228    0.515      0.484    0.499      0.260    0.694     0.500     Adult
                 0.631    0.228    0.581      0.631    0.605      0.395    0.807     0.641     Old Age
Weighted Avg.    0.618    0.191    0.618      0.618    0.618      0.427    0.798     0.651

=== Confusion Matrix ===

    a    b    c   <-- classified as
 1777  355  269 |    a = Child
  415 1161  825 |    b = Adult
  148  739 1514 |    c = Old Age
```

J48 - Decision Tree:

```
=== Evaluation result ===

Scheme: J48
Options: -C 0.25 -M 2
Relation: Abalone-newage-weka.filters.unsupervised.attribute.ClassAssigner-Clast-weka.filters.supervised.att


Correctly Classified Instances        6198               86.0475 %
Incorrectly Classified Instances      1005               13.9525 %
Kappa statistic                          0.7907
Mean absolute error                      0.1191
Root mean squared error                  0.2799
Relative absolute error                 26.7951 %
Root relative squared error             59.3671 %
Total Number of Instances             7203

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.878    0.083    0.841      0.878    0.859      0.787    0.940     0.849     Child
                 0.713    0.053    0.871      0.713    0.784      0.699    0.876     0.837     Adult
                 0.990    0.074    0.871      0.990    0.927      0.891    0.973     0.911     Old Age
Weighted Avg.    0.860    0.070    0.861      0.860    0.857      0.792    0.930     0.866

=== Confusion Matrix ===

    a    b    c   <-- classified as
 2107  236   58 |    a = Child
  393 1713  295 |    b = Adult
    5   18 2378 |    c = Old Age
```

IBk - KNN:

```
=== Evaluation result ===

Scheme: IBk
Options: -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-la
Relation: Abalone-newage-weka.filters.unsupervised.attribute.ClassAssigner-Clast-weka.filters.supervised.att


Correctly Classified Instances        6675               92.6697 %
Incorrectly Classified Instances       528                7.3303 %
Kappa statistic                          0.89
Mean absolute error                      0.0493
Root mean squared error                  0.2205
Relative absolute error                 11.0923 %
Root relative squared error             46.7824 %
Total Number of Instances             7203

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.932    0.044    0.914      0.932    0.923      0.884    0.959     0.899     Child
                 0.850    0.031    0.932      0.850    0.889      0.839    0.917     0.875     Adult
                 0.998    0.035    0.934      0.998    0.965      0.948    0.980     0.925     Old Age
Weighted Avg.    0.927    0.037    0.927      0.927    0.926      0.890    0.952     0.900

=== Confusion Matrix ===

    a    b    c   <-- classified as
 2237  145   19 |    a = Child
  210 2042  149 |    b = Adult
    0    5 2396 |    c = Old Age
```

	Naïve Bayes	Multilayer Perceptron	J48-DT	IBk -KNN
Original data	24.40%	30.30%	75.72%	100%
Pipeline optimized	56.48%	61.80%	86.05%	92.67%

1. **Naive Bayes:** There is a 32% increase in performance, the reason why it will increase even using cross-validation is that the dataset is optimized by pipeline and 29 class labels are categorized into 3. Thus, the performance increased but it is still not an ideal algorithm for this dataset.

2. **MP:** 31% increase in performance, similarly, the growth is due to the changes in the dataset. 61.8% is still an average performance. It is not the right algorithm for the Abalone data.

3. **Decision Tree:** In the first part, we found that the decision tree has a good result. Better dataset brings to a finer performance. However, the problem for DT is not about accuracy but complexity. Although an outstanding performance is received, the tree is becoming incomprehensible. Thus, the next step for DT is using pruning and optimizing technique to reduce the complexity.

4. **KNN:** Thanks to exploiting cross-validation, the overfitting has been resolved. Thus, KNN became the best performer. Because KNN can easily build the model based on the numeric attributes. Sufficient instances make the linear correlation more obvious. Overall, KNN is the most matching algorithm after overcoming the overfitting.

Classifers for the techniques

```
Scheme:   MultilayerPerceptron
Relation: Abalone-newage-weka.filters.unsupervised.attribute.ClassAssigner-Clast-weka.filters.supervised.a

Sigmoid Node 0
    Inputs    Weights
    Threshold    -1.0260819790413862
    Node 3    3.567226186021965
    Node 4    -8.960053853520082
    Node 5    0.1964372069118022
Sigmoid Node 1
    Inputs    Weights
    Threshold    0.5965686533221877
    Node 3    -1.426104978523153
    Node 4    -1.5700218419940573
    Node 5    -1.5005113861701345
Sigmoid Node 2
    Inputs    Weights
    Threshold    -1.102815153560948
    Node 3    -7.6783093025797
    Node 4    2.4047571461092323
    Node 5    1.633585388774311
Sigmoid Node 3
    Inputs    Weights
    Threshold    -26.223948934694295
    Attrib Diam    -1.4731554315854942
    Attrib Height    -29.048294255120325
    Attrib Whole    -3.9440899658779647
Sigmoid Node 4
    Inputs    Weights
    Threshold    22.387736198697592
    Attrib Diam    6.975755830711806
    Attrib Height    38.82114416077602
    Attrib Whole    -4.917817273238708
Sigmoid Node 5
    Inputs    Weights
    Threshold    32.25633147519861
    Attrib Diam    -41.87106028504566
    Attrib Height    20.447907734688915
    Attrib Whole    17.10431124593586
Class Child
    Input
    Node 0
Class Adult
```

MP

```
Height <= 0.12
|   Whole <= 0.6435
|   |   Whole <= 0.497
|   |   |   Height <= 0.1: Child (1078.0/99.0)
|   |   |   Height > 0.1
|   |   |   |   Whole <= 0.293
|   |   |   |   |   Diam <= 0.32
|   |   |   |   |   |   Height <= 0.105
|   |   |   |   |   |   |   Whole <= 0.2685: Child (14.0/2.0)
|   |   |   |   |   |   |   Whole > 0.2685
|   |   |   |   |   |   |   |   Whole <= 0.2825: Adult (4.0)
|   |   |   |   |   |   |   |   Whole > 0.2825
|   |   |   |   |   |   |   |   |   Diam <= 0.26: Child (2.0)
|   |   |   |   |   |   |   |   |   Diam > 0.26
|   |   |   |   |   |   |   |   |   |   Whole <= 0.2845: Child (2.0)
|   |   |   |   |   |   |   |   |   |   Whole > 0.2845: Adult (2.0)
|   |   |   |   |   |   Height > 0.105
|   |   |   |   |   |   |   Diam <= 0.275
|   |   |   |   |   |   |   |   Whole <= 0.2445: Adult (5.0/1.0)
|   |   |   |   |   |   |   |   Whole > 0.2445: Child (6.0)
|   |   |   |   |   |   |   Diam > 0.275
|   |   |   |   |   |   |   |   Diam <= 0.3: Adult (12.0)
|   |   |   |   |   |   |   |   Diam > 0.3: Child (3.0/1.0)
|   |   |   |   |   |   Diam > 0.32: Old Age (8.0)
|   |   |   |   |   Whole > 0.293
|   |   |   |   |   |   Height <= 0.115
|   |   |   |   |   |   |   Height <= 0.105
|   |   |   |   |   |   |   |   Diam <= 0.365
|   |   |   |   |   |   |   |   |   Whole <= 0.336: Child (20.0)
|   |   |   |   |   |   |   |   |   Whole > 0.336
|   |   |   |   |   |   |   |   |   |   Diam <= 0.305: Adult (6.0)
|   |   |   |   |   |   |   |   |   |   Diam > 0.305
|   |   |   |   |   |   |   |   |   |   |   Diam <= 0.345: Child (47.0/7.0)
|   |   |   |   |   |   |   |   |   |   |   Diam > 0.345
|   |   |   |   |   |   |   |   |   |   |   |   Whole <= 0.4165: Adult (7.0)
|   |   |   |   |   |   |   |   |   |   |   |   Whole > 0.4165: Child (15.0/1.0)
|   |   |   |   |   |   |   |   |   Diam > 0.365: Adult (5.0)
|   |   |   |   |   |   |   Height > 0.105
|   |   |   |   |   |   |   |   Diam <= 0.3: Child (18.0)
|   |   |   |   |   |   |   |   Diam > 0.3
|   |   |   |   |   |   |   |   |   Whole <= 0.457
|   |   |   |   |   |   |   |   |   |   Diam <= 0.31: Adult (5.0)
|   |   |   |   |   |   |   |   |   |   Diam > 0.31
```

Decision tree

```
=== Classifier model ===

Scheme:   NaiveBayes
Relation: Abalone-newage-weka.filters.unsupervised.attri

Naive Bayes Classifier

                Class
Attribute       Child   Adult Old Age
                (0.33)  (0.33)  (0.33)
=======================================
Diam
  mean          0.3244  0.4494  0.4692
  std. dev.       0.09  0.0727   0.061
  weight sum      2161    2161    2161
  precision     0.0055  0.0055  0.0055

Height
  mean          0.1075  0.1554  0.1706
  std. dev.     0.0396  0.0307  0.0294
  weight sum      2161    2161    2161
  precision     0.0231  0.0231  0.0231

Whole
  mean          0.4464  1.0126  1.1603
  std. dev.     0.3112  0.4443  0.4214
  weight sum      2161    2161    2161
  precision     0.0015  0.0015  0.0015
```

Naive Bayes

```
=== Classifier model ===

Scheme:   IBk
Relation: Abalone-newage-weka.filters.unsupervised.attribute.Cl

IB1 instance-based classifier
using 1 nearest neighbour(s) for classification
```

KNN-IBk

2.4.1 Pipeline simulation:

2.4.2 EC classification result:

```
TEST RESULTS
============
Classifier= .a/abalone-optimised/abalone-optimised
Fold 0 : CORRECT=0.7320574162679425 N/C=0.0
Fold 1 : CORRECT=0.7248803827751196 N/C=0.0
Fold 2 : CORRECT=0.7129186602870814 N/C=0.0
Fold 3 : CORRECT=0.7751196172248804 N/C=0.0
Fold 4 : CORRECT=0.7392344497607656 N/C=0.0
Fold 5 : CORRECT=0.722488038277512 N/C=0.0
Fold 6 : CORRECT=0.715311004784689 N/C=0.0
Fold 7 : CORRECT=0.7458033573141487 N/C=0.0
Fold 8 : CORRECT=0.7721822541966427 N/C=0.0
Fold 9 : CORRECT=0.7266187050359711 N/C=0.0
Global Classification Error + N/C:
0.26333861140752474
stddev Global Classification Error + N/C:
0.02076595176296808
Correctly classified:
0.7366613885924753
Global N/C:
0.0

TRAIN RESULTS
============
Classifier= .a/abalone-optimised/abalone-optimised
Summary of data, Classifiers: .a/abalone-optimised/abalone-optimised
Fold 0 : CORRECT=0.7398244213886672 N/C=0.0
Fold 1 : CORRECT=0.7432827879755254 N/C=0.0
Fold 2 : CORRECT=0.7454110135674381 N/C=0.0
Fold 3 : CORRECT=0.7347698856078744 N/C=0.0
Fold 4 : CORRECT=0.7382282521947326 N/C=0.0
Fold 5 : CORRECT=0.7430167597765363 N/C=0.0
Fold 6 : CORRECT=0.7443469007714818 N/C=0.0
Fold 7 : CORRECT=0.7404255319148936 N/C=0.0
Fold 8 : CORRECT=0.7396276595744681 N/C=0.0
Fold 9 : CORRECT=0.7398936170212767 N/C=0.0
Global Classification Error + N/C:
0.25911731702071056
stddev Global Classification Error + N/C:
0.0030185411210853197
Correctly classified:
0.7408826829792894
Global N/C:
0.0
```

2.4.3 Evolutionary Computation vs four techniques:

Definition:

Evolutionary computation is belonging to the family of nature-inspired algorithms and the purpose of this algorithm is to achieve global optimization. It is also a population-based technique. It has two characteristic – metaheuristic and stochastic.

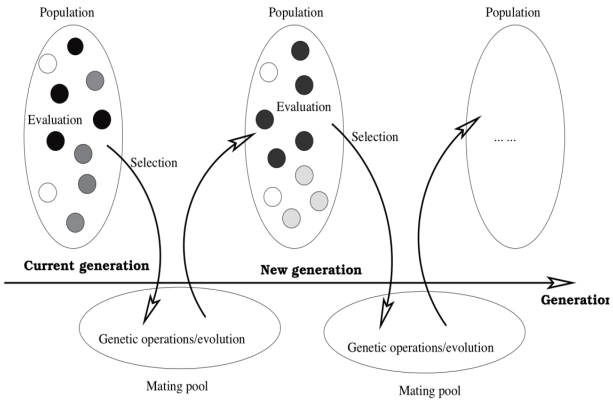
Advantages:

It can solve the problems that the rest of the fours techniques cannot.

1. It can overcome local optima and find global optima.
2. It gives reasonable assumptions.
3. No need dealing with a large number of parameters.
4. The flexible structure of the solution.

In terms of Abalone dataset:

As we can see, the average performance for classifying this dataset is around 73%. It is lower than KNN and DT. The reason may because EC is mainly used for optimization. However, we need to use this dataset to predict the age and the visualization shows a strong linear correlation. EC's optimizing method is not suited for this scenario. Therefore, it ineffectively performed for this dataset.



Process of Evolutionary Algorithm

Optimization and representation:

The representation in EC can be a tree structure or a binary string.

There are three ways to optimizing the result:

1. Reproduction: choosing good performing individual into new generation. It is also called Elitism.
2. Mutation: alter one specific part of the individual and put into a new generation.
3. Crossover: choosing two individuals and swap parts and create two new individuals, then put them into a new generation.

2.4.4 WEKA vs KEEL:

	Difficulty	Targeted user	User interface	Scope of the tool	Report
WEKA	Easy	Everyone	Friendly	4 tribes of AI exclude EC	Easy to access and read
KEEL	Medium	User with basic knowledge	Friendly most time	All 5 tribes of AI	Use terminal run the script first

The advantages of WEKA:

WEKA is supporting visualization of internal and external relation of the dataset. The report is easy to access because it is inside of the application. Unlike Keel, we should download a zip file to access. It also provides a user-friendly interface and this tool is more suitable for users who are new to machine learning.

The advantages of KEEL:

KEEL provides the Evolutionary computation solution which WEKA was not able to make. The Construction of the pipeline is easy to read. Moreover, KEEL provides a complete collection of algorithms. If the user takes the time to explore, then, they can get good user experience.