

Bayesian Learning Experiment

course project for CS573x(machine learning)
Spring 2002

Jie Bao
Dept, of Computer Science
Iowa State University
Ames, IA, 50010
baojie@iastate.edu, www.cs.iastate.edu/~baojie

2002-04-02 to 2002-04-24

Requirement:

Naive Bayes Classifier which assume that the attributes are independent given the class label. This is equivalent to building a classifier using first order statistics. Your algorithm should be able to handle missing attribute values.

A Bayesian Network for classification using statistics of order at most k. (where k is a user-supplied parameter and $k \ll n$ where n is the total number of attributes). Your algorithm should be able to handle missing attribute values.

An algorithm for estimating the parameters (conditional probability tables associated with each node) for a Bayesian network with a fixed topology. Your algorithm should be able to handle missing attribute values. You may make use of appropriate priors for estimating probabilities from small samples.

An algorithm for constructing a Bayesian network for classification by searching the space of candidate networks. Since exhaustive search is not feasible, you may use heuristic search (using an appropriately designed scoring function). You are free to devise any reasonable search strategy.

Implement:

1. Naive Bayes Classifier:

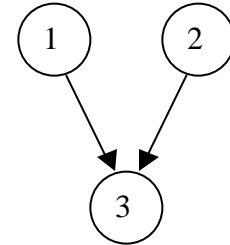
see class **BayesianNaive**

It's implemented by Weka, and I rewrite it in **BayesianNaive**, but change the superclass from **DistributionClassifier** to **Classifier**. Actually, there are only one function **classifyInstance()** in **DistributionClassifier**. I move it into **BayesianNaive** to see a more clear processing flow shabby

2. Network storage

see int [][] BayesianNetwork::m_Network;

We use adjacent table to store the bayesian network as a graph. It's a two-dimensional matrix. Every row and column in that matrix corresponding to a node, or an attribute in the instances. Each element 1 in the matrix represents an edge in the graph. For example, for following graph, the matrix will be:



	1	2	3
1	0	0	1
2	0	0	1
3	0	0	0

3. Loop detect

see: BayesianNetwork::LoopCheck() and DFS()

Before add any edge in the network, we make a loop detection to ensure there is no loop in the network. We use **Depth First Search** algorithm, start from the end node of the edge we want to add and find if we can meet this node again during search. If such node has been visited for more than once, more new edge will be added into the network.

4. Storage of probability table

see : class **Protable**

BayesianNetwork:: private Protable []m_Table;

The probability table stores the conditional probability values at certain node. For example, if node k stands for attribute X_k with parent nodes X_i, X_j, X_m , and the class attribute is C, then the table item means:

$$P(X_k | X_i, X_j, X_m, C)$$

the length of the table is

$$|X_k| |X_j| |X_m| |C|$$

|a| means the number of possible discrete values of a.

If X_k has no parent, the table stores:

$$P(X_k | C)$$

and the table length is

$$|X_k || C|$$

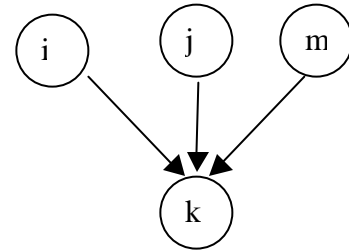
5. Table build

see BayesianNetwork:: **BuildTable()**

when the structure is fixed, we can use BayesianNetwork::**BuildTable()** to determine its CPT(conditional probability table)

for node like X_k with parent nodes X_i, X_j, X_m , to calculate $P(X_k | X_i, X_j, X_m, C)$, we do following things:

- a) Count all combination of X_i, X_j, X_m, X_k on all instances for every class value C_m
- b) Normalize the result by number of total instance that have class value C_m



The high dimensional table (eg. for above example, the table is 5-dimension) has been stored as 1-dimension array. The combination of attributes and the array index have been translated by function BayesianNetwork::**Combination2Index()** and BayesianNetwork::**Index2Combination()**. for example, the instance(suppose there are 4 attributes)

$$X_i=a, X_j=b, X_m=c, X_k=d, C=e$$

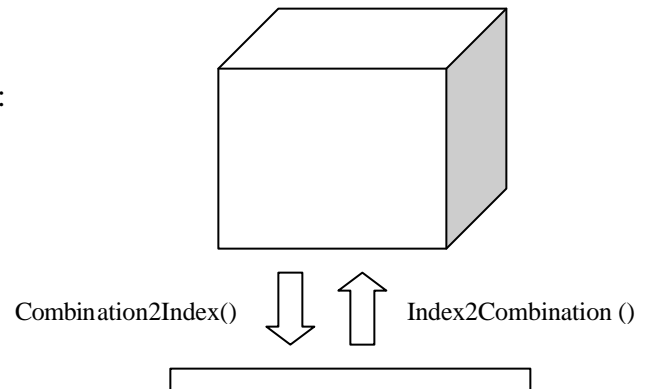
then , $index = e + |C|(a+|X_i|(b+|X_j|(c+|X_k|d)))$

In decoding, reverse translation is done as following:

```

index % |C| ≙ e
index = (index - e) / |C|
index % |Xi| ≙ a
.....
index = (index - c) / |Xk|
index ≙ d

```



6. The search of structure

see BayesianNetwork:: **SearchNetwork()**

We use algorithm proposed in [D. Matgaritis etc. NetCube: A Scalable Tool for Fast Data Mining and Compression] to search a structure for give dataset. It's a greedy hill-climbing algorithm. It may not give optimal structure

1. G = empty
 2. score = -inf
 3. do
 - a) maxscore = score
 - b) for each attribute pair (A,B) do
 - try
 - 1) Add edge<A,B>, build table T, newscore1= ScoreFromData(G,T)
 - 2) Add edge<A,B>, build table T, newscore2= ScoreFromData(G,T)
 - 3) delete <A,B> if there was one, build table T, newscore3= ScoreFromData(G,T)
 - c) score = max (newscore1, newscore2, newscore3)
 - d) maxscore = max (maxscore, score)
 - e) choose corresponding structure.
- while(score > maxscore)
4. return G

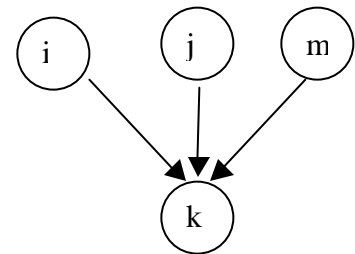
the computation of score can be found in that paper.

7. Classify instances

see BayesianNetwork:: **distributionForInstance()**

When we have finished building network and constructing CPT, we will try to classify the instances with it. For give instance $i = (x_1, x_2, \dots, x_n)$, where x_j is the value for attribute X_j , the posterior probability of i belonging to class c_t is.

$$p_t = \prod_{j=1}^n P(X_j = x_j | Pa_j, C = c_t)$$



for example, as in left network,

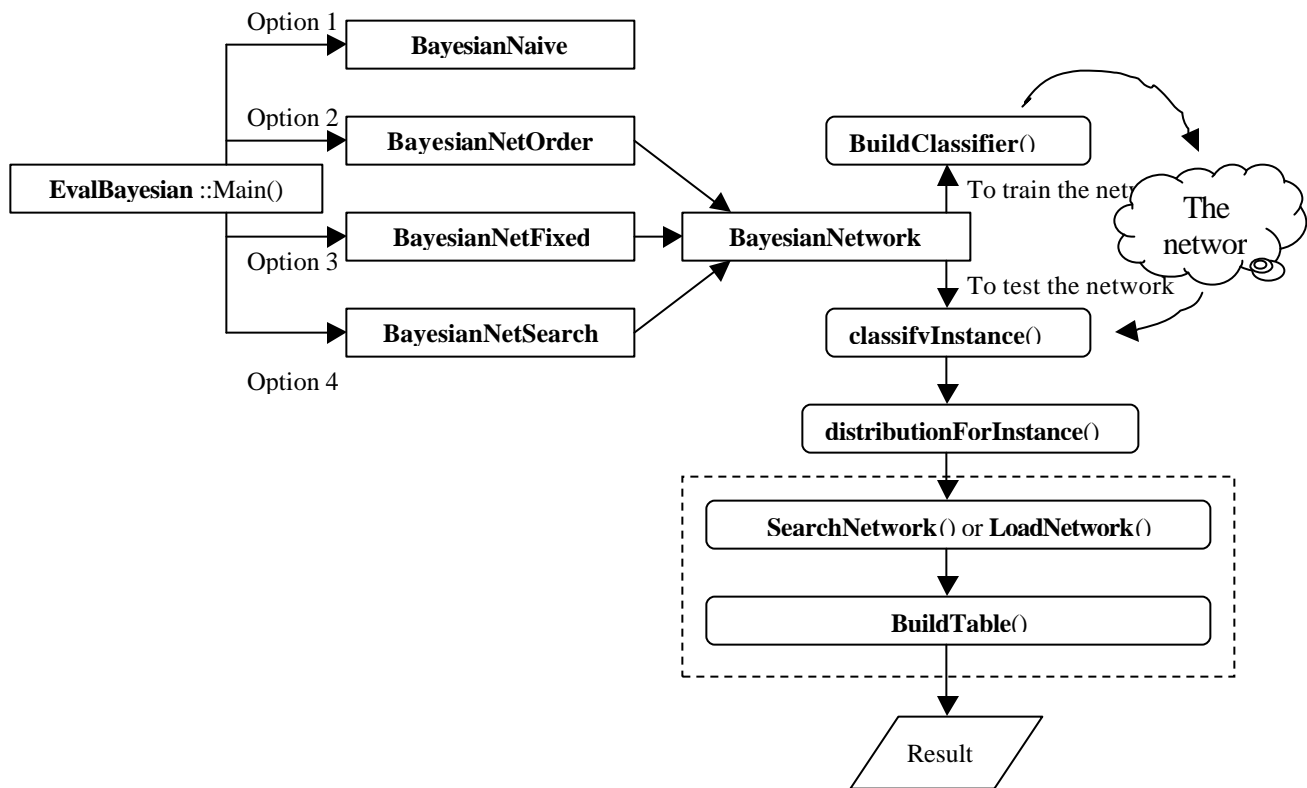
$$p = P(X_i = x_i | C = c_t)P(X_j = x_j | C = c_t)P(X_m = x_m | C = c_t) \\ P(X_k = x_k | X_i = x_i, X_j = x_j, X_m = x_m, C = c_t)$$

we will try all possible class value, and choose the one with maximal posterior possibility as classification result.

8. Weka

We implement the algorithm in Weka frame. for more information about weak, please refer to <http://www.cs.iastate.edu/~cs573x/weka.html> and <http://www.cs.waikato.ac.nz/ml/weka/> for detailed information.

Basic processing flow is as the following:



9. Missing value

see EvalBayesian::assignargPro()

we use weka.filters.ReplaceMissingValuesFilter to fill up missing values

10. Numeric value

see EvalBayesian::assignargPro()

we use weka.filters.DiscretizeFilter to fill up missing values

Result:

1. Naive Bayes Classifier

see class BayesianNaive

the result on weather.nominal.arff

My Naive Bayes (simple)

number of Class 2

number of Attribute: 5

class attribute's index: 4

number of instances: 0

Class yes: $P(C) = 0.625$

Attribute outlook

sunny	overcast	rainy
0.25	0.41666667	0.33333333

Attribute temperature

hot	mild	cool
0.25	0.41666667	0.33333333

Attribute humidity

high	normal
0.36363636	0.63636364

Attribute windy

TRUE	FALSE
0.36363636	0.63636364

Class no: $P(C) = 0.375$

Attribute outlook

sunny	overcast	rainy
0.5	0.125	0.375

Attribute temperature

hot	mild	cool
0.375	0.375	0.25

Attribute humidity

high	normal
0.71428571	0.28571429

```
Attribute windy
TRUE FALSE
0.57142857 0.42857143
```

```
Time taken to build model: 0.08 seconds
Time taken to test model on training data: 0.04 seconds
```

```
=== Error on training data ===
```

```
Correctly Classified Instances    13    92.8571 %
Incorrectly Classified Instances   1     7.1429 %
Kappa statistic                   0.8372
Mean absolute error               0.0714
Root mean squared error           0.2673
Relative absolute error           15.3846 %
Root relative squared error       55.7386 %
Total Number of Instances        14
```

```
=== Confusion Matrix ===
```

```
a b <-- classified as
9 0 | a = yes
1 4 | b = no
```

```
=== Stratified cross-validation ===
```

```
Correctly Classified Instances    8     57.1429 %
Incorrectly Classified Instances   6     42.8571 %
Kappa statistic                   -0.0244
Mean absolute error               0.4286
Root mean squared error           0.6547
Relative absolute error           90 %
Root relative squared error       132.6919 %
Total Number of Instances        14
```

```
=== Confusion Matrix ===
```

```
a b <-- classified as
7 2 | a = yes
4 1 | b = no
```

the result on 1.train.arff and 1.test.arff (votes cast)

My Naive Bayes (simple)

number of Class 2

number of Attribute: 17

class attribute's index: 16

number of instances: 0

Class democrat: $P(C) = 0.63701068$

Attribute handicapped-infants

y n

0.58333333 0.41666667

Attribute water-project-cost-sharing

y n

0.43888889 0.56111111

Attribute adoption-of-the-budget-resolution

y n

0.91666667 0.08333333

Attribute physician-fee-freeze

y n

0.04444444 0.95555556

Attribute el-salvador-aid

y n

0.21111111 0.78888889

Attribute religious-groups-in-schools

y n

0.48888889 0.51111111

Attribute anti-satellite-test-ban

y n

0.78888889 0.21111111

Attribute aid-to-nicaraguan-contras

y n

0.83888889 0.16111111

Attribute mx-missile

y n

0.77777778 0.22222222

Attribute immigration

y n

0.49444444 0.50555556

Attribute synfuels-corporation-cutback

y n

0.47777778 0.52222222

Attribute education-spending

y n

0.15 0.85

Attribute superfund-right-to-sue

y n

0.28888889 0.71111111

Attribute crime

y n

0.38333333 0.61666667

Attribute duty-free-exports

y n

0.58888889 0.41111111

Attribute export-administration-act-south-africa

y n

0.95555556 0.04444444

Class republican: $P(C) = 0.36298932$

Attribute handicapped-infants

y n

0.22330097 0.77669903

Attribute water-project-cost-sharing

y n

0.46601942 0.53398058

Attribute adoption-of-the-budget-resolution

y n

0.18446602 0.81553398

Attribute physician-fee-freeze

y n

0.97087379 0.02912621

Attribute el-salvador-aid

y n

0.9223301 0.0776699

Attribute religious-groups-in-schools

y n

0.88349515 0.11650485

Attribute anti-satellite-test-ban

y n

0.27184466 0.72815534

Attribute aid-to-nicaraguan-contras

y n

0.22330097 0.77669903

Attribute mx-missile

y n

0.13592233 0.86407767

Attribute immigration

y n

0.5631068 0.4368932

Attribute synfuels-corporation-cutback

y n

0.15533981 0.84466019

Attribute education-spending

y n

0.76699029 0.23300971

Attribute superfund-right-to-sue

y n

0.78640777 0.21359223

Attribute crime

y n

0.97087379 0.02912621

Attribute duty-free-exports

y n

0.11650485 0.88349515

Attribute export-administration-act-south-africa

y n

0.69902913 0.30097087

Time taken to build model: 0.09 seconds

Time taken to test model on training data: 0.08 seconds

=== Error on training data ===

Correctly Classified Instances	252	90.3226 %
Incorrectly Classified Instances	27	9.6774 %
Kappa statistic	0.7953	
Mean absolute error	0.0968	
Root mean squared error	0.3111	
Relative absolute error	20.9383 %	
Root relative squared error	64.731 %	
Total Number of Instances	279	

=== Confusion Matrix ===

```
a b <-- classified as
159 19 | a = democrat
 8 93 | b = republican
```

=== Error on test data ===

Correctly Classified Instances	141	89.2405 %
Incorrectly Classified Instances	17	10.7595 %
Kappa statistic	0.7788	
Mean absolute error	0.1076	
Root mean squared error	0.328	
Relative absolute error	22.8674 %	
Root relative squared error	67.0556 %	
Total Number of Instances	158	

=== Confusion Matrix ===

```
a b <-- classified as
84 12 | a = democrat
```

5 57 | b = republican

2. A Bayesian Network for classification using statistics of order at most k.

see class **BayesianNetOrder**. The found network will be stored as "newNet.net".

K=3 in following experiments

On weather data: (the result is shown in the figure)

Bayesian Network

number of Class 2

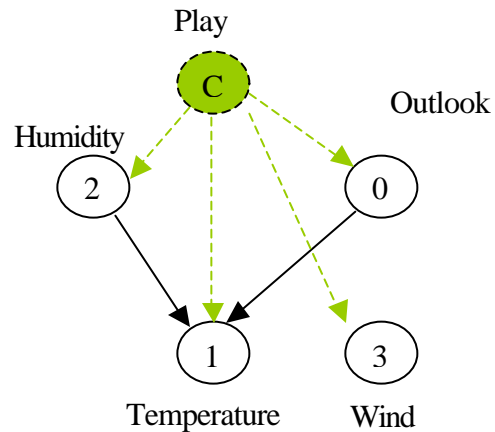
number of Attribute: 5

class attribute's index: 4

Structure of the Network

Edge: from 0 to 1

Edge: from 2 to 1



Time taken to build model: 0.13 seconds

Time taken to test model on training data: 0.03 seconds

=== Error on training data ===

Correctly Classified Instances	10	71.4286 %
Incorrectly Classified Instances	4	28.5714 %
Kappa statistic	0.4286	
Mean absolute error	0.24	
Root mean squared error	0.4591	
Relative absolute error	51.6904 %	
Root relative squared error	95.7554 %	
Total Number of Instances	14	

=== Confusion Matrix ===

a b <- classified as

6 3 | a = yes

1 4 | b = no

On veto data (start from an empty graph)

Bayesian Network

number of Class 2

number of Attribute: 17

class attribute's index: 16

Structure of the Network

Edge: from 0 to 1

Time taken to build model: 310.07 seconds

Time taken to test model on training data: 0.16 seconds

=== Error on training data ===

Correctly Classified Instances	252	90.3226 %
Incorrectly Classified Instances	27	9.6774 %
Kappa statistic	0.7962	
Mean absolute error	0.0954	
Root mean squared error	0.2948	
Relative absolute error	20.6491 %	
Root relative squared error	61.3524 %	
Total Number of Instances	279	

=== Confusion Matrix ===

```
a b <-- classified as
158 20 | a = democrat
 7 94 | b = republican
```

=== Stratified cross-validation ===

Correctly Classified Instances	252	90.3226 %
Incorrectly Classified Instances	27	9.6774 %
Kappa statistic	0.7962	

Mean absolute error	0.0979
Root mean squared error	0.2994
Relative absolute error	21.1782 %
Root relative squared error	62.2996 %
Total Number of Instances	279

=== Confusion Matrix ===

```

a  b  <-- classified as
158 20 | a = democrat
 7 94 | b = republican

```

start from a random DAG

```

Bayesian Network
number of Class 2
number of Attribute: 17
class attribute's index: 16

```

```

Structure of the Network
Edge: from 4 to 5
Edge: from 13 to 1
Edge: from 13 to 3
Edge: from 13 to 4
Edge: from 13 to 5
Edge: from 13 to 11
Edge: from 13 to 12

```

```

Time taken to build model: 0.07 seconds
Time taken to test model on training data: 0.17 seconds

```

=== Error on training data ===

Correctly Classified Instances	209	74.9104 %
Incorrectly Classified Instances	70	25.0896 %
Kappa statistic	0.5241	
Mean absolute error	0.2464	
Root mean squared error	0.4924	

```
Relative absolute error      53.3084 %
Root relative squared error  102.4496 %
Total Number of Instances    279
```

```
=== Confusion Matrix ===
```

```
  a  b  <-- classified as
110 68 | a = democrat
  2 99 | b = republican
```

```
=== Stratified cross-validation ===
```

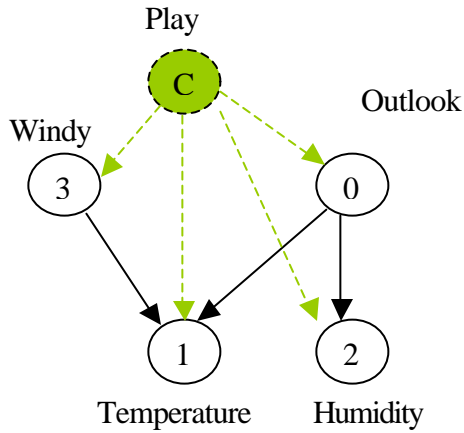
```
Correctly Classified Instances    209      74.9104 %
Incorrectly Classified Instances   70      25.0896 %
Kappa statistic                   0.5241
Mean absolute error               0.2484
Root mean squared error           0.4953
Relative absolute error           53.7272 %
Root relative squared error       103.0504 %
Total Number of Instances         279
```

```
=== Confusion Matrix ===
```

```
  a  b  <-- classified as
110 68 | a = democrat
  2 99 | b = republican
```

I have modified the original algorithm of NetCube in that I omit the deleting operation in search structure. That's because when Naïve model works well on some dataset, the heuristic search from an empty graph will most likely converge to the empty graph itself, which is actually a Naïve Bayesian Classifier. When using original NetCube on weather and veto dataset, we get two empty networks. It's easy to understand because naïve model has high accuracy on those two datasets.

3. An algorithm for estimating the parameters



see class **BayesianNetFixed**

it will read a network file names "test.net". please rename your network file to this name under same directory. Two networks are given: "weather.net" and "veto.net".

Test on weather data is based on the network in left. C stands for a virtual Class node because every node has conditional probability over C such as $P(X1|X2,C)$. The result list as below.

Detailed CDT can be printed out by `Protable::print()`

Bayesian Network

number of Class 2

number of Attribute: 5

class attribute's index: 4

Structure of the Network

Edge: from 0 to 1

Edge: from 0 to 2

Edge: from 3 to 1

Time taken to build model: 0.06 seconds

Time taken to test model on training data: 0.04 seconds

=== Error on training data ===

Correctly Classified Instances	11	78.5714 %
Incorrectly Classified Instances	3	21.4286 %
Kappa statistic	0.5882	
Mean absolute error	0.2572	
Root mean squared error	0.4612	
Relative absolute error	55.4016 %	
Root relative squared error	96.1893 %	

Total Number of Instances 14

=== Confusion Matrix ===

```
a b <-- classified as
6 3 | a = yes
0 5 | b = no
```

=== Stratified cross-validation ===

Correctly Classified Instances	10	71.4286 %
Incorrectly Classified Instances	4	28.5714 %
Kappa statistic	0.4717	
Mean absolute error	0.2819	
Root mean squared error	0.4911	
Relative absolute error	59.1911 %	
Root relative squared error	99.5363 %	
Total Number of Instances	14	

=== Confusion Matrix ===

```
a b <-- classified as
5 4 | a = yes
0 5 | b = no
```

Test on veto data with fixed network:

Bayesian Network

number of Class 2

number of Attribute: 17

class attribute's index: 16

Structure of the Network

Edge: from 0 to 1

Edge: from 0 to 2

Edge: from 2 to 3

Edge: from 2 to 4
 Edge: from 3 to 1
 Edge: from 3 to 11
 Edge: from 4 to 6
 Edge: from 5 to 11
 Edge: from 6 to 7
 Edge: from 8 to 6
 Edge: from 9 to 14
 Edge: from 10 to 14
 Edge: from 11 to 0
 Edge: from 11 to 13
 Edge: from 12 to 15
 Edge: from 15 to 4

(note that

0. handicapped-infants {y,n}
1. water-project-cost-sharing {y,n}
2. adoption-of-the-budget-resolution {y,n}
3. physician-fee-freeze {y,n}
4. el-salvador-aid {y,n}
5. religious-groups-in-schools {y,n}
6. anti-satellite-test-ban {y,n}
7. aid-to-nicaraguan-contras {y,n}
8. mx-missile {y,n}
9. immigration {y,n}
10. synfuels-corporation-cutback {y,n}
11. education-spending {y,n}
12. superfund-right-to-sue {y,n}
13. crime {y,n}
14. duty-free-exports {y,n}
15. export-administration-act-south-africa {y,n}

)

Time taken to build model: 0.07 seconds

Time taken to test model on training data: 0.19 seconds

=== Error on training data ===

Correctly Classified Instances	230	82.4373 %
Incorrectly Classified Instances	49	17.5627 %
Kappa statistic	0.6222	
Mean absolute error	0.1775	
Root mean squared error	0.418	
Relative absolute error	38.4106 %	
Root relative squared error	86.9713 %	

```

Total Number of Instances      279

=== Confusion Matrix ===

  a  b  <-- classified as
152 26 | a = democrat
 23 78 | b = republican

=== Stratified cross-validation ===

Correctly Classified Instances      225      80.6452 %
Incorrectly Classified Instances     54      19.3548 %
Kappa statistic                     0.5755
Mean absolute error                  0.2013
Root mean squared error              0.4395
Relative absolute error              43.5546 %
Root relative squared error          91.4356 %
Total Number of Instances           279

=== Confusion Matrix ===

  a  b  <-- classified as
154 24 | a = democrat
 30 71 | b = republican

```

Because the structure is not optimal, the performance is even worse than naïve Bayesian classifier.

4. **An algorithm for constructing a Bayesian network for classification by searching the space of candidate networks.**

see **BatesianNetSearch**

Actually, it's a more general case of problem 2 with a big k. the difference is only that we use maxDegree =10 instead of small k(2,3) in **BayesianNetOrder**. Same result is got on weather dataset. As to veto dataset, the running is too time consuming (in problem 2, it takes 2 hours for k = 3 on veto dataset) and the result is not included here.