CS6735 Programming Assignment 2

Justin Kamerman 3335272

November 11, 2010

1 Assignment

1. Implement Naïve Bayes using Java. Evaluate your implementation on the datasets in data.zip using 10 times 5-fold cross-validation, and report the average accuracy and standard deviation. All datasets are for UCI machine learning repository. You can check the detailed descriptions from the following link:

   http://www.ics.uci.edu/ mlearn/MLRepository.html

   For breast cancer data see: http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+

   For car data see: http://archive.ics.uci.edu/ml/datasets/Car+Evaluation

   For ecoli data see:http://archive.ics.uci.edu/ml/datasets/Ecoli

   For letter recognition data see: http://archive.ics.uci.edu/ml/datasets/Letter+Recognition

   For mushroom data see: http://archive.ics.uci.edu/ml/datasets/Mushroom

2. Bonus question: For a given data set, can we detect whether the conditional independent assumption is true or not? How? Can we observe that Naïve Bayes works well only on the data sets on which the conditional independent assumption is true or roughly true? You may need to check some reference papers.

2 Learning Algorithm

A Java program was written implementing the Naïve Bayes Classifier as described in [1]. During the learning phase of the algorithm, the program calculates a marginal probability vector, with each position in the vector corresponding to a possible target value. A three-dimensional conditional probability matrix is created, one dimension corresponding to target value, another dimension corresponding to attribute class, and a third dimension corresponding to attribute value domain. In estimating conditional probabilities, the **m-estimate** technique described in [1] is used:

$$P(A_j = a_k | C = c_i) = \frac{n_c + mp}{n + m}$$

where, for a particular training set, n is the cardinality of the subset for which classification c_i holds and n_c, the cardinality of the subset of instances for which classification c_i holds and attribute class A_j has value a_k. p is a prior estimate of the probability we wish to calculate. In this case, we assume uniform distribution i.e. if the attribute in question has k possible values, we assign $p = \frac{1}{k}$. m is a constant value which we call the *equivalent sample size* and determines how heavily to weight p relative to the observed data.

Using *m-estimate* to estimate conditional probabilities addresses the problem of a sparse probability matrix. Without *m-estimate*, conditional probability matrix nodes which are not represented in the training data are set to zero, resulting in a biased underestimation of probabilities calculated using these terms. Using *m-estimate*, matrix nodes not represented in the training data are set to $\frac{mp}{n+m}$ instead.

In order to determine suitable m values for each of the data sets, various tests were performed to see how the selection of m effected mean accuracy of the Naïve Bayes Classifier during K-fold cross-validation. Figure 1 shows how in each case the mean accuracy was a maximum when m was small and decreases as m increases before stabilizing. It is the student's opinion that as m is increased, the probability estimates tends towards a uniform distribution, the *equivalent sample* parameters overwhelming $\frac{n_c}{n}$. From these results, a value of $m = 10$ was selected for all remaining tests.

Missing attribute values are handled by preprocessing data sets, replacing missing attribute value with the most probable value for the attribute in the data set.

3 Data Sets

All data set used in this assignment are taken from [2]. The program expects data in CSV format with attributes occurring first and classification at the end of each line. All data files have been preprocessed to fit this format.

car.data

- Number of Instances: 1728
- Number of Attributes: 6
- Attribute Values:

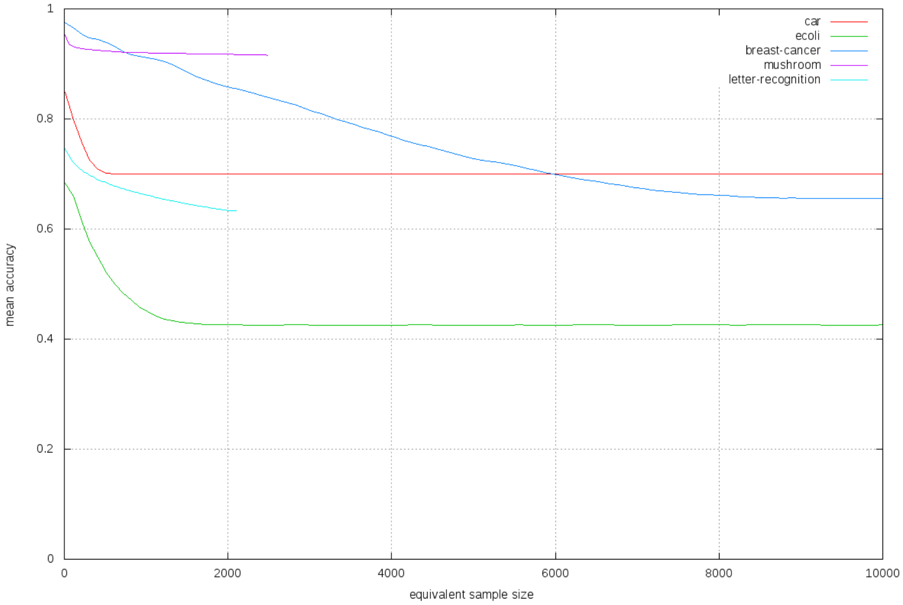

Figure 1: m-estimate tuning

buying	v-high, high, med, low
maint	v-high, high, med, low
doors	2, 3, 4, 5-more
persons	2, 4, more
lug_boot	small, med, big
safety	low, med, high

- Missing Attribute Values: none
- Class Distribution (number of instances per class)

unacc	1210
acc	384
good	69
v-good	65

ecoli.data

- Number of Instances: 336

- Number of Attributes: 7
- Attribute Values:

  mcg McGeoch's method for signal sequence recognition.

  gvh von Heijne's method for signal sequence recognition.

  lip von Heijne's Signal Peptidase II consensus sequence score. Binary attribute.

  chg Presence of charge on N-terminus of predicted lipoproteins. Binary attribute.

  aac score of discriminant analysis of the amino acid content of outer membrane and periplasmic proteins.

  alm1 score of the ALOM membrane spanning region prediction program.

  alm2 score of ALOM program after excluding putative cleavable signal regions from the sequence.

- Missing Attribute Values: none.
- Class Distribution (number of instances per class)

cp	143
im	77
pp	52
imU	35
om	20
omL	5
imL	2
imS	2

mushroom.data

- Number of Instances: 8124
- Number of Attributes: 22
- Attribute Information:

cap-shape	bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s
cap-surface	fibrous=f, grooves=g, scaly=y, smooth=s
cap-color	brown=n, buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y
bruises?	bruises=t, no=f
odor	almond=a, anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s
gill-attachment	attached=a, descending=d, free=f, notched=n
gill-spacing	close=c, crowded=w, distant=d
gill-size	broad=b, narrow=n
gill-color	black=k, brown=n, buff=b, chocolate=h, gray=g, green=r, orange=o, pink=p, purple=u, red=e, white=w, yellow=y
stalk-shape	enlarging=e, tapering=t
stalk-root	bulbous=b, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r, missing=?
stalk-surface-above-ring	fibrous=f, scaly=y, silky=k, smooth=s
stalk-surface-below-ring	fibrous=f, scaly=y, silky=k, smooth=s
stalk-color-above-ring	brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
stalk-color-below-ring	brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
veil-type	partial=p, universal=u
veil-color	brown=n, orange=o, white=w, yellow=y
ring-number	none=n, one=o, two=t
ring-type	cobwebby=c, evanescent=e, flaring=f, large=l, none=n, pendant=p, sheathing=s, zone=z
spore-print-color	black=k, brown=n, buff=b, chocolate=h, green=r, orange=o, purple=u, white=w, yellow=y
population	abundant=a, clustered=c, numerous=n, scattered=s, several=v, solitary=y
habitat	grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d

- Missing Attribute Values: 2480, all for attribute 11.

- Class Distribution:

edible:	4208 (51.8%)
poisonous:	3916 (48.2%)
total:	8124 instances

letter-recognition.data

- Number of Instances: 20000
- Number of Attributes: 17 (Letter category and 16 numeric features)
- Attribute Information:

lettr	capital letter (26 values from A to Z)
x-box	horizontal position of box (integer)
y-box	vertical position of box (integer)
width	width of box (integer)
high	height of box (integer)
onpix	total on pixels (integer)
x-bar	mean x of on pixels in box (integer)
y-bar	mean y of on pixels in box (integer)
x2bar	mean x variance (integer)
y2bar	mean y variance (integer)
xybar	mean x y correlation (integer)
x2ybr	mean of x * x * y (integer)
xy2br	mean of x * y * y (integer)
x-ege	mean edge count left to right (integer)
xegvy	correlation of x-ege with y (integer)
y-ege	mean edge count bottom to top (integer)
yegvx	correlation of y-ege with x (integer)

- Missing Attribute Values: None
- Class Distribution:

789 A	766 B	736 C	805 D	768 E	775 F	773 G
734 H	755 I	747 J	739 K	761 L	792 M	783 N
753 O	803 P	783 Q	758 R	748 S	796 T	813 U
764 V	752 W	787 X	786 Y	734 Z		

breast-cancer.data

- Number of Instances: 699
- Number of Attributes: 10
- Attribute Information: (class attribute has been moved to last column)

Clump Thickness	1 - 10
Uniformity of Cell Size	1 - 10
Uniformity of Cell Shape	1 - 10
Marginal Adhesion	1 - 10
Single Epithelial Cell Size	1 - 10
Bare Nuclei	1 - 10
Bland Chromatin	1 - 10
Normal Nucleoli	1 - 10
Mitoses	1 - 10

- Missing attribute values: 16
- Class distribution: (2 for benign, 4 for malignant)

Benign	458 (65.5%)
Malignant	241 (34.5%)

4 Program Design

The Naïve Bayes Classifier is implemented in a Java program. The only external dependency of the program is the Apache commons-cli library for parsing commend line options. To that end, the program is operated from the command line, taking options listed in table 1. Wrapper scripts, *run.sh* and *run.bat* are provided as a convenience for executing the program on UNIX and Windows platforms respectively. The header comments in each script contain the command lines used for executing the program with the various data files provided.

The program parses a data file, assumed to contain a set of training instances on each line. Each training instance line is a comma-separated list of attribute values terminated by a classification or target value. Of the training data supplied for the assignment, some had to be preprocessed to fit the expected format.

Option	Description
-f arg	Path of data file
-i arg	Number of iterations to perform. Default is 10
-n arg	A comma-separated list of attribute names, matching the order in which they appear in the data file
-o arg	Number of folds to create in the training data during. Default is 5.
-m arg	m-estimate equivalent sample size. Default is 10.
-h	Print help message

Table 1: Command line options

The program then executes a number of iterations (default is 10) over the entire data set. During each iteration, the data set is folded (default is 5 times) and the training set used to create a Naïve Bayes Classifier. The instances in the test set are then evaluated by the classifier and the accuracy of the classifications recorded for each test set evaluated. After the final iteration is complete, the mean accuracy and standard deviation are calculated and output by the program.

The program implements various mechanism to facilitate debugging. Throughout the code, log statements have been added using the Java logging framework. The logging output is controlled for individual classes via the *logging.properties* file which the program read on start-up.

The implementation classes and their relationships are represented in a UML class diagram in Figure 2. Following is a brief description of each class:

- **Attribute** represents an attribute's name and value.
- **AttributeSet** is a collection of attributes. It has convenience functions to determine attribute value range and probabilities of discrete values.
- **Classification** represents an instance's classification or the value of the target function.
- **ClassificationSet** is a collection of classifications. It has convenience functions to determine classification value range and probabilities of each classification. These functions are used extensively during the entropy calculation.
- **Classifier** is an interface implemented by any class which can take an instance and generate a classification for that instance. The interface is

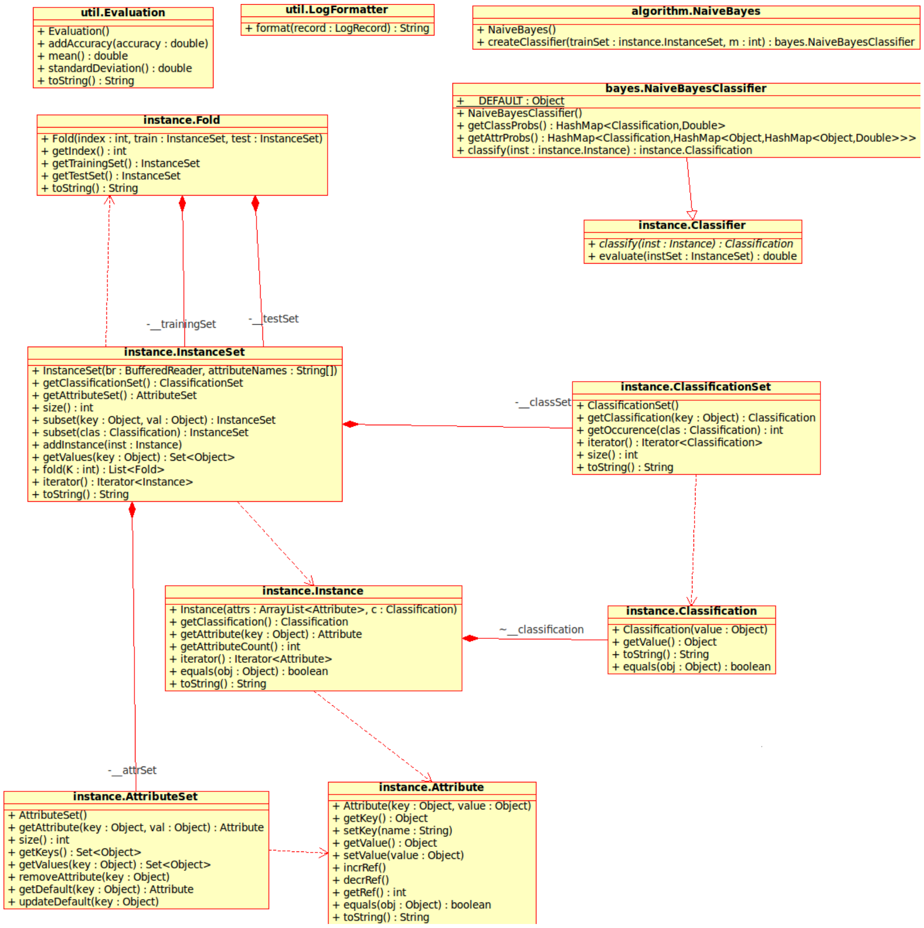
util.Evaluation
+ Evaluation()
+ addAccuracy(accuracy : double)
+ mean() : double
+ standardDeviation() : double
+ toString() : String
util.LogFormatter
+ format(record : LogRecord) : String
algorithm.NaiveBayes
+ NaiveBayes()
+ createClassifier(trainSet : instance.InstanceSet, m : int) : bayes.NaiveBayesClassifier
bayes.NaiveBayesClassifier
+ DEFAULT : Object
+ NaiveBayesClassifier()
+ getClassProbs() : HashMap<Classification,Double>
+ getAttrProbs() : HashMap<Classification,HashMap<Object,HashMap<Object,Double>>>
+ classify(inst : instance.Instance) : instance.Classification
instance.Fold
+ Fold(index : int, train : InstanceSet, test : InstanceSet)
+ getIndex() : int
+ getTrainingSet() : InstanceSet
+ getTestSet() : InstanceSet
+ toString() : String
instance.Classifier
+ classify(inst : Instance) : Classification
+ evaluate(instSet : InstanceSet) : double
-__trainingSet
-__testSet
instance.InstanceSet
+ InstanceSet(br : BufferedReader, attributeNames : String[])
+ getClassificationSet() : ClassificationSet
+ getAttributeSet() : AttributeSet
+ size() : int
+ subset(key : Object, val : Object) : InstanceSet
+ subset(clas : Classification) : InstanceSet
+ addInstance(inst : Instance)
+ getValues(key : Object) : Set<Object>
+ fold(K : int) : List<Fold>
+ iterator() : Iterator<Instance>
+ toString() : String
-__classSet
instance.ClassificationSet
+ ClassificationSet()
+ getClassification(key : Object) : Classification
+ getOccurence(clas : Classification) : int
+ iterator() : Iterator<Classification>
+ size() : int
+ toString() : String
instance.Instance
+ Instance(attrs : ArrayList<Attribute>, c : Classification)
+ getClassification() : Classification
+ getAttribute(key : Object) : Attribute
+ getAttributeCount() : int
+ iterator() : Iterator<Attribute>
+ equals(obj : Object) : boolean
+ toString() : String
~__classification
instance.Classification
+ Classification(value : Object)
+ getValue() : Object
+ toString() : String
+ equals(obj : Object) : boolean
-__attrSet
instance.AttributeSet
+ AttributeSet()
+ getAttribute(key : Object, val : Object) : Attribute
+ size() : int
+ getKeys() : Set<Object>
+ getValues(key : Object) : Set<Object>
+ removeAttribute(key : Object)
+ getDefault(key : Object) : Attribute
+ updateDefault(key : Object)
instance.Attribute
+ Attribute(key : Object, value : Object)
+ getKey() : Object
+ setKey(name : String)
+ getValue() : Object
+ setValue(value : Object)
+ incrRef()
+ decrRef()
+ getRef() : int
+ equals(obj : Object) : boolean
+ toString() : String

Figure 2: UML class diagram

implemented by the *DecisionTree*, *Rule*, and *RuleSet* classes.

- **Evaluation** is a helper class for capturing test run accuracies and calculating final mean and standard deviation values.
- **Fold** is an encapsulation of a training and a test set.
- **Instance** represents a collection of attributes and their classification. In the case of training data, we use the classification to select the target hypothesis. In the case of test data, the classification is used to validate output of the target hypothesis.
- **InstanceSet** is a collection of *Instances*. This is the class that parses the data file and generates individual Instances. Functions exist to calculate entropy and information gain, create a subset of instance based on a particular attribute's value and fold the set of instances to create test and training sets of instances.
- **LogFormatter** is a helper class to format log messages.
- **Main** is the class which is started from the command line. It parses command line options and starts the test iterations.
- **NaiveBayes** creates a Naïve Bayes Classifier given a training data *InstanceSet*. It calculates marginal and conditional probabilities and populates a probability matrix within the *NaiveBayesClassifier*. It uses the m-estimation technique to estimate conditional probabilities.
- **NaiveBayesClassifier** classifies an *Instance* using the Naïve Bayes Classifier, based on a matrix of marginal and conditional probabilities which are calculated from a previously seen training set.

5 Results

The results of processing each data file through ten iterations of 5-fold cross-validation are presented below. Table 2 tabulates accuracy scores for the Naïve Bayes classifier and ID3 Decision Tree for comparison. In three of the five data set, ID3 performed marginally better than Naïve Bayes and in one case, Naïve Bayes better than ID3. In the case of the ecoli data set, Naïve Bayes performed significantly better than ID3.

Data Set	Naïve Bayes Accuracy	ID3 Accuracy
car	0.841449	0.938377
ecoli	0.684776	0.384776
mushroom	0.952007	1.000000
letter-recognition	0.737670	0.760870
breast-cancer-wisconsin	0.972662	0.945899

Table 2: Comparison of Naïve Bayes and ID3 Decision Tree performance

car.data

```
[8:49:05][INFO][Main]: Loaded data from file data/car.data:
[size=1728][entropy=1.205741][classificationSet=[[size=4][classifications=
[[[good][occurence=69]][[unacc][occurence=1210]][[acc][occurence=384]]
[[vgood][occurence=65]]]]]][attributes=[[doors][maint][safety][buying][lug_boot][persons]]]
[8:49:05][INFO][instance.InstanceSet]: C = 0.000000; Cmax = 8.754888; C/Cmax = 0.000000
[8:49:05][INFO][Main]: total correlation = 0.000000
[8:49:05][INFO][Main]: Starting iteration 1 of 10
[8:49:36][INFO][Main]: Starting iteration 2 of 10
[8:50:09][INFO][Main]: Starting iteration 3 of 10
[8:50:47][INFO][Main]: Starting iteration 4 of 10
[8:51:20][INFO][Main]: Starting iteration 5 of 10
[8:51:54][INFO][Main]: Starting iteration 6 of 10
[8:52:26][INFO][Main]: Starting iteration 7 of 10
[8:52:58][INFO][Main]: Starting iteration 8 of 10
[8:53:30][INFO][Main]: Starting iteration 9 of 10
[8:54:01][INFO][Main]: Starting iteration 10 of 10
[8:54:32][INFO][Main]:

Test Results

Naive Bayes: [tests=50][mean accuracy=0.841449][standard deviation=0.023485]
```

ecoli.data

```
[8:55:16][INFO][Main]: Loaded data from file data/ecoli.data:
[size=336][entropy=2.188746][classificationSet=
[[size=8][classifications=[[[imS][occurence=2]][[cp][occurence=143]][[imL][occurence=2]]
[[om][occurence=20]][[omL][occurence=5]][[pp][occurence=52]][[im][occurence=77]][[imU][occurence=35]]]
[attributes=[[chg][aac][lip][gvh][alm1][mcg][alm2]]]
[8:55:16][INFO][instance.InstanceSet]: C = 20.792162; Cmax = 23.112473; C/Cmax = 0.899608
```

```
[8:55:16][INFO][Main]: total correlation = 20.792162
[8:55:16][INFO][Main]: Starting iteration 1 of 10
[8:55:39][INFO][Main]: Starting iteration 2 of 10
[8:56:00][INFO][Main]: Starting iteration 3 of 10
[8:56:22][INFO][Main]: Starting iteration 4 of 10
[8:56:43][INFO][Main]: Starting iteration 5 of 10
[8:57:04][INFO][Main]: Starting iteration 6 of 10
[8:57:25][INFO][Main]: Starting iteration 7 of 10
[8:57:46][INFO][Main]: Starting iteration 8 of 10
[8:58:06][INFO][Main]: Starting iteration 9 of 10
[8:58:27][INFO][Main]: Starting iteration 10 of 10
[8:58:48][INFO][Main]:

Test Results

Naive Bayes: [tests=50][mean accuracy=0.684776][standard deviation=0.053285]
```

mushroom.data

```
[9:00:27][INFO][Main]: Loaded data from file data/mushroom.data:
[size=8124][entropy=0.999068][classificationSet=
[[size=2][classifications=[[[e][occurence=4208]]
[[p][occurence=3916]]]]]][attributes=
[[cap-surface][veil-color][odor][ring-number]
[gill-size][stalk-surface-below-ring][stalk-root]
[cap-shape][gill-color]
[stalk-surface-above-ring][gill-spacing][habitat]
[veil-type][cap-color] [bruises?][stalk-color-above-ring][stalk-shape]
[stalk-color-below-ring][spore-print-color][gill-attachment][ring-type][population]]]
[9:00:28][INFO][instance.InstanceSet]: C = 18.239124; Cmax = 28.196665; C/Cmax = 0.646854
[9:00:28][INFO][Main]: total correlation = 18.239124
[9:00:28][INFO][Main]: Starting iteration 1 of 10
[9:43:37][INFO][Main]: Starting iteration 2 of 10
[10:14:24][INFO][Main]: Starting iteration 3 of 10
[10:44:10][INFO][Main]: Starting iteration 4 of 10
[11:16:12][INFO][Main]: Starting iteration 5 of 10
[11:57:28][INFO][Main]: Starting iteration 6 of 10
[12:48:49][INFO][Main]: Starting iteration 7 of 10
[1:40:13][INFO][Main]: Starting iteration 8 of 10
[2:31:27][INFO][Main]: Starting iteration 9 of 10
```

```
[3:22:06][INFO][Main]: Starting iteration 10 of 10
[4:12:59][INFO][Main]:

Test Results

Naive Bayes: [tests=50][mean accuracy=0.952007][standard deviation=0.006205]
```

letter-recognition.data

```
[11:40:12][INFO][Main]: Loaded data from file
data/letter-recognition.data: [size=20000]
[entropy=4.699811][classificationSet= [[size=26][classifications=[[[D]
[occurence=805]][[E][occurence=768]][[F][occurence=775]][[G][occurence=773]]
[[A][occurence=789]][[B][occurence=766]][[C][occurence=736]][[L][occurence=761]]
[[M][occurence=792]][[N][occurence=783]][[O][occurence=753]][[H][occurence=734]]
[[I][occurence=755]][[J][occurence=747]][[K][occurence=739]][[U][occurence=813]]
[[T][occurence=796]][[W][occurence=752]][[V][occurence=764]][[Q][occurence=783]]
[[P][occurence=803]][[S][occurence=748]][[R][occurence=758]][[Y][occurence=786]]
[[X][occurence=787]][[Z][occurence=734]]]]]][attributes=[[y-ege][xy2br][x-bar]
[x-ege][x2bar][x2ybr][width][xegvy][onpix][x-box][y2bar][y-bar][y-box][high][yegvx][xybar]]]
[11:40:17][INFO][instance.InstanceSet]: C = 35.346037; Cmax = 45.803145; C/Cmax = 0.771695
[11:40:17][INFO][Main]: total correlation = 35.346037
[11:40:17][INFO][Main]: Starting iteration 1 of 10
[1:41:01][INFO][Main]: Starting iteration 2 of 10
[3:39:41][INFO][Main]: Starting iteration 3 of 10
[4:55:15][INFO][Main]: Starting iteration 4 of 10
[5:54:03][INFO][Main]: Starting iteration 5 of 10
[6:52:51][INFO][Main]: Starting iteration 6 of 10
[7:52:01][INFO][Main]: Starting iteration 7 of 10
[8:51:05][INFO][Main]: Starting iteration 8 of 10
[9:49:54][INFO][Main]: Starting iteration 9 of 10
[10:48:47][INFO][Main]: Starting iteration 10 of 10
[11:47:39][INFO][Main]:

Test Results

Naive Bayes: [tests=50][mean accuracy=0.737670][standard deviation=0.007147]
```

breast-cancer-wisconsin.data

```
[9:21:52][INFO][Main]: Loaded data from file data/breast-cancer-wisconsin.data: [size=699][entropy=0.9
[9:21:52][INFO][instance.InstanceSet]: C = 12.120860; Cmax = 17.214665; C/Cmax = 0.704101
[9:21:52][INFO][Main]: total correlation = 12.120860
[9:21:52][INFO][Main]: Starting iteration 1 of 10
[9:23:42][INFO][Main]: Starting iteration 2 of 10
[9:25:24][INFO][Main]: Starting iteration 3 of 10
[9:27:02][INFO][Main]: Starting iteration 4 of 10
[9:28:52][INFO][Main]: Starting iteration 5 of 10
[9:30:18][INFO][Main]: Starting iteration 6 of 10
[9:31:46][INFO][Main]: Starting iteration 7 of 10
[9:33:12][INFO][Main]: Starting iteration 8 of 10
[9:34:44][INFO][Main]: Starting iteration 9 of 10
[9:36:09][INFO][Main]: Starting iteration 10 of 10
[9:37:33][INFO][Main]:

Test Results

Naive Bayes: [tests=50][mean accuracy=0.972662][standard deviation=0.012294]
```

6 Conditional Independence

The Naïve Bayes Classifier is based on the simplifying assumption that attribute values are independent of one another, given the target value. An attempt was made to quantify the conditional independence of each of our data sets to see if there was any correlation between conditional independence and the accuracy of our Naïve Bayes Classifiers. A property called *total correlation* [3] was selected as our measure of conditional independence between the attributes in each data set. *Total Correlation* expresses the amount of redundancy or dependency existing between a set of variables and is given by:

$$C(X_1, X_2, ..., X_n) = \sum_{i=0}^{n} H(X_i) - H(X_1, X_2, ..., X_n)$$

where $H(X_i)$ is the information entropy of variable X_i and $H(X_1, X_2, ..., X_n)$ is the joint entropy of the variable set $\{X_1, X_2, ..., X_n\}$. In terms of the discrete probability distributions on variables $\{X_1, X_2, ..., X_n\}$, the *total correlation* is given by

$$C(X_1, X_2, ..., X_n) = \sum_{x_1 \in X_1} \sum_{x_2 \in X_2} ... \sum_{x_n \in X_n} p(x_i, x_2, ..., x_n) log \frac{p(x_i, x_2, ..., x_n)}{p(x_1)p(x_2)...p(x_n)}$$

In our case, $\{X_1, X_2, ..., X_n\}$ was taken to be the set of attributes for each data set.

The sum $\sum_{i=0}^{n} H(X_i)$ represents the amount of information shared among the variables in the set in terms of the average code length to transmit the values of all the variables if each variable was (optimally) coded independently. The term $H(X_1, X_2, ..., X_n)$ quantifies the actual amount of information that the variable set contains in terms of the the average code length to transmit the values of all the variables if the set of variables was optimally coded together. The difference in these two terms represents the absolute redundancy (in bits) present in a given set of variables, and thus provides a general quantitative measure of the structure or organization embodied in the set [4]. The results of our *total correlation* calculations for each data set shown in Table 3. These results are plotted against mean accuracy of the associated Naïve Bayes Classifier in Figure 3. The car data set is the only set to exhibit zero *total correlation* but does not represent the best mean accuracy during evaluation. Looking at Figure 3, there does not seem to be any obvious relationship between accuracy and *total correlation*. This phenomenon could be accounted for by one or more of the following factors:

- While conditional independence is an assumption upon which the Naïve Bayes Classifier is based, the classifier still performs well in certain domains where conditional independence does not hold. In practise, the Naïve Bayes Classifier performs remarkably well in many text classification problems despite the incorrectness of the conditional independence assumption [1].

- The data sets provided are samples from a probability distribution covering all possible attribute combinations. Depending on how representative the sample is in each case, classifiers trained on the data will perform with varying degrees of success against unseen instances.

- In calculating joint entropy, $H(X_1, X_2, ..., X_n)$, for *total correlation*, it is difficult to estimate probabilities from the data set due to the large population size. Better estimates could be obtained by correlating attribute

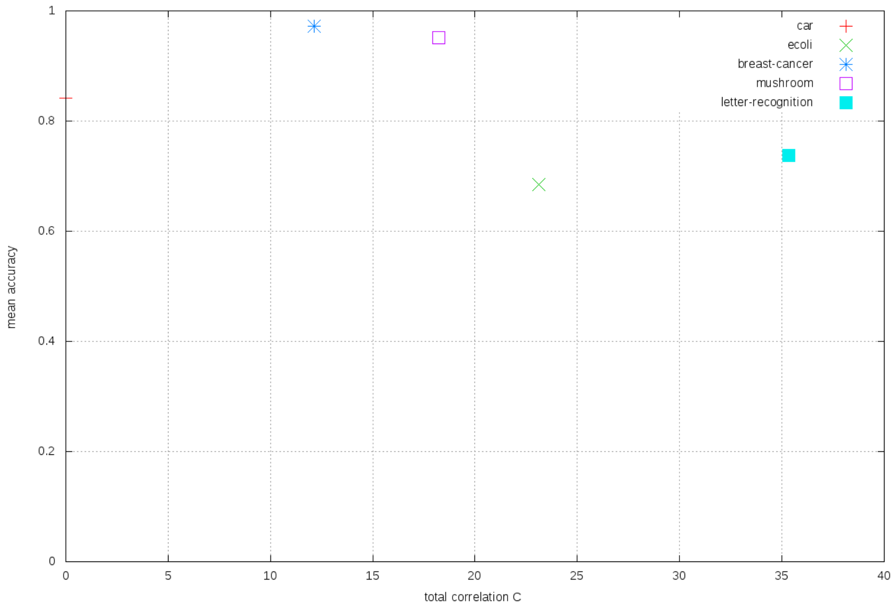

Figure 3: Total Correlation vs Accuracy

pairs rather than trying to calculate a single metric representing the entire data set. This modification was not implemented by the student due to time constraints.

Data Set	**Mean Accuracy**	**Total Correlation**
car	0.841449	0.0
ecoli	0.684776	23.112473
mushroom	0.952007	18.239124
letter-recognition	0.737670	35.346037
breast-cancer-wisconsin	0.972662	12.120860

Table 3: Total Correlation vs Accuracy

References

[1] T. M. Mitchell, *Machine Learning.* New York: McGraw-Hill, 1997.

[2] A. Frank and A. Asuncion, “UCI machine learning repository,” 2010. [Online]. Available: http://archive.ics.uci.edu/ml

[3] S. Watanabe, “Information Theoretical Analysis of Multivariate Correlation,” *IBM Journal of Research and Development*, vol. 4, no. 1, pp. 66–82, 1960.

[4] Wikipedia, “Wikipedia, the free encyclopedia,” 2010, [Online; accessed 10-Nov-2010]. [Online]. Available: http://en.wikipedia.org/wiki/Total_correlation