Data Mining Classification Models: an analysis

Daniel Hanspeter 6129, Daniel Graziotin 4801, Thomas Schievenin 5701

October / November 2010

Abstract

This document analyzes various techniques of analysis and data manipulation. In particular, it investigates about four different classification models, namely Decision tree, Naïve Bayesian Classifier, Logistic Regression and KNN classifier, that are summarized in the last page. The experiments are performed on two completely different datasets, blood-transfusion and census-income, specifically chosen in order to compare work done on small versus big datasets. Blood transfusion is relatively small and has a reduced amount of attributes, while census-income is bigger, having also a more relevant amount of attributes to investigate on. On the first dataset there is not a class so we introduced one, in order to perform our test and focus on classification and prediction. This was not needed for the second dataset, since it was already containing a class to be analyzed. Therefore we used this dataset mainly for benchmarking and performance analysis of the classification models by varying the data pre-processing techniques they have in common.

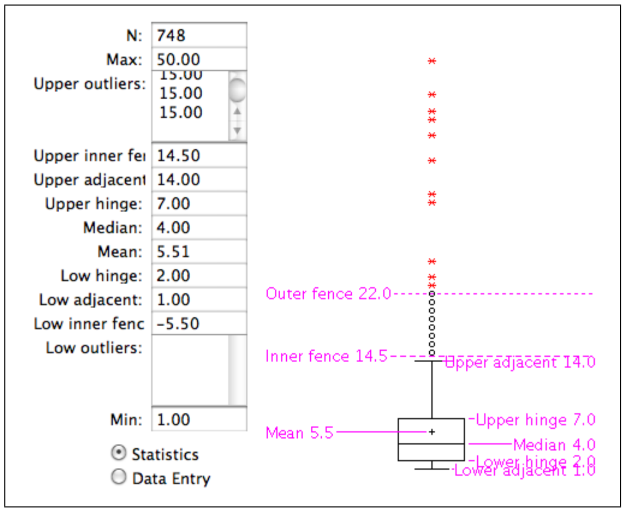
1 Data Examination

1.1 Blood Transfusion

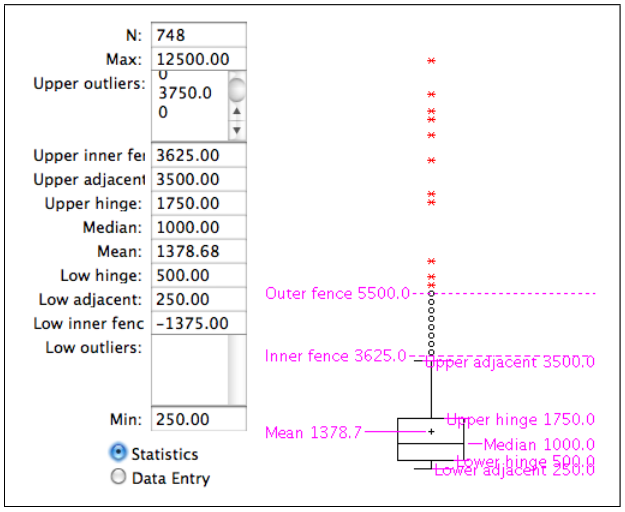
This dataset was collected at the Blood Transfusion Service Center in Hsin-Chu City in Taiwan. The data is collected randomly from donors and is used to achieve a Frequency-Metric-Model of donations focusing on the donor's situation, namely on the amount of done donations before, of the total amount of donations and so on, for a total of 748 instances and 5 attributes. The data collected is focused on March 2007, since one attribute collects exactly if one person donated in March 2007 or not.

```
Recency(months) NUMERIC – months since last donation
Frequency(times) NUMERIC – total number of donation
Monetary(c.c.blood) NUMERIC – total blood donated in c.c.
Time(months) NUMERIC – months since first donation
donatedmarch2007 NUMERIC – donated blood in March 2007 (1=donating blood; 0=not donating blood)
```

There are no missing values in the Dataset. As derivable from the Boxplots in the next picture, that contains the boxplots of attributes frequency and monetary distributions, both collected values there are Positive outliers. For the Monetary Boxplot (on the right side) we can say that the values are located in a very short range, by noticing that the whiskers are just below/above the box ranges. Outliers are present in both chosen attributes and they will be treated by applying discretization.



N: 748
Max: 50.00
Upper outliers: 15.00 15.00 15.00
Upper inner fe 14.50
Upper adjacent 14.00
Upper hinge: 7.00
Median: 4.00
Mean: 5.51
Low hinge: 2.00
Low adjacent: 1.00
Low inner fenc -5.50
Low outliers:
Min: 1.00
[✓ Statistics] Data Entry



N: 748
Max: 12500.00
Upper outliers: 0 3750.0 0
Upper inner fe 3625.00
Upper adjacent 3500.00
Upper hinge: 1750.00
Median: 1000.00
Mean: 1378.68
Low hinge: 500.00
Low adjacent: 250.00
Low inner fenc -1375.00
Low outliers:
Min: 250.00
[✓ Statistics] Data Entry

But how can we compute the similarity between the data objects according to the attribute types of your datasets? Since all values (except the binary value for doing transfusion in march 07) in this dataset are numeric ones, to compare on similarity we can calculate the distance using the Minkowski distance, by first cleaning up the data by bringing attributes to a unit-less form.

For this dataset we will introduce a class big and small donator, in order to know if someone is a influent donator or not, on the base of the time slot since the first donation and the frequency of donations. So, we can build a classifier applying the following rules:

```
If TIME(time since first donation)>49 & FREQ(freq. of donation)>24 THEN BIG
If TIME(time since first donation)<=49 & FREQ(freq. of donation)>11 THEN BIG
```

1.2 Census Income

The dataset contains 15 attributes and a total of 32561 instance. The following is a representation of the attributes and their types, extracted from the ARFF file we created.

```
age NUMERIC
workclass {Private, Self–emp–not–inc, Federal–gov, [..], Never–worked}
fnlwgt NUMERIC
education {Bachelors, Some–college, [..], Preschool}
education–num NUMERIC
marital–status {Married–civ–spouse, Divorced, [..], Married–AF–spouse}
occupation {Tech–support, Craft–repair, [..], Armed–Forces}
relationship {Wife, Own–child, Husband, [..], Unmarried}
race      {White, Asian–Pac–Islander, Amer–Indian–Eskimo, Other, Black}
```

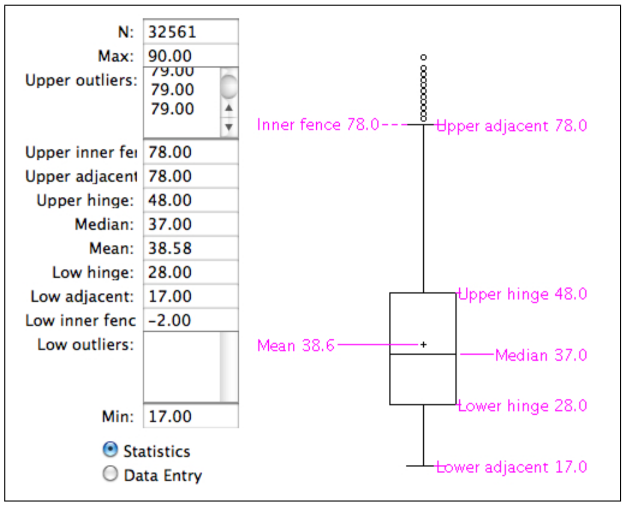
```
sex      {Female, Male}
capital-gain NUMERIC
capital-loss NUMERIC
hours-per-week NUMERIC
native-country {United-States, Cambodia, England, [..]}
class    { >50K, <=50K}
```

Education and **education-num** are referring to the same value, where education-num is a numerical representation of education.

Attribute **fnlwgt** does not appear to have any meaningful sense. Further researches confirmed our

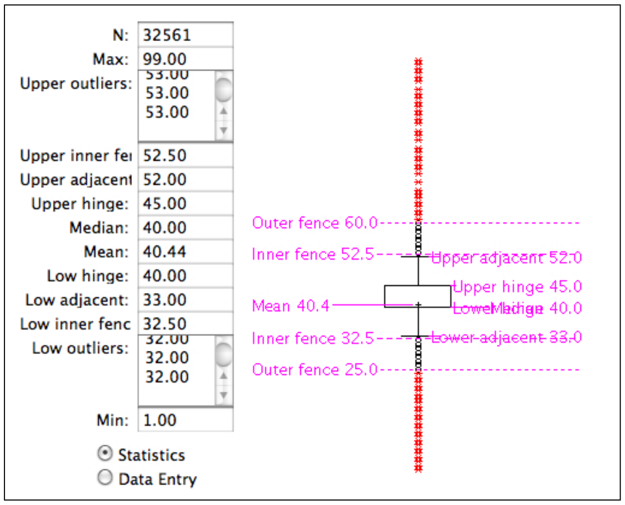
There are **missing values** for attributes workclass (6%), occupation (6%), native-country (2%). Two possible strategies to deal with the missing values are complete removal of the instances having missing values and substitution of the missing values with the mode value of the attribute.

The two attributes age and hours-per-week have been selected for further analysis. The following are their **boxplot** representation:



N:	32561
Max:	90.00
Upper outliers:	79.00 79.00 79.00
Upper inner fe	78.00
Upper adjacent	78.00
Upper hinge:	48.00
Median:	37.00
Mean:	38.58
Low hinge:	28.00
Low adjacent:	17.00
Low inner fenc	-2.00
Low outliers:	
Min:	17.00

[✓ Statistics] Data Entry



N:	32561
Max:	99.00
Upper outliers:	53.00 53.00 53.00
Upper inner fe	52.50
Upper adjacent	52.00
Upper hinge:	45.00
Median:	40.00
Mean:	40.44
Low hinge:	40.00
Low adjacent:	33.00
Low inner fenc	32.50
Low outliers:	32.00 32.00 32.00
Min:	1.00

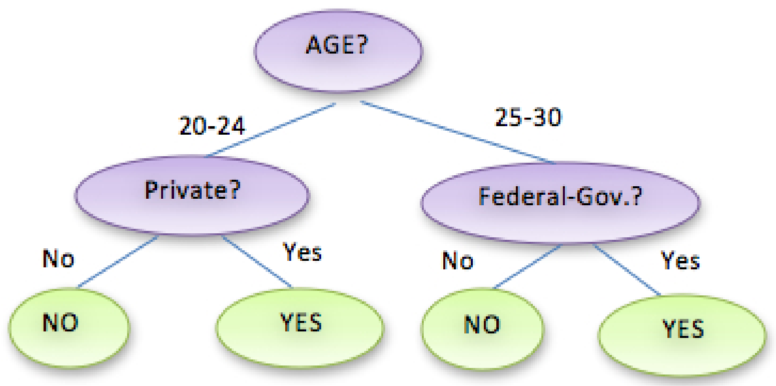
[✓ Statistics] Data Entry

There are **outliers** for both the selected attributes. They will be treated with discretization techniques.

Possible **data similarity** technique: here we have in addition to numeric values also nominal values. In this case we can do a simple matching using the formula d(I,j)= (p-m)/p where m is the number of matches and p is the number of variables or we could create a binary mapping for those values.

Possible data mining tasks: an interesting task could focus around the income class. For example we could construct a tree which tells us if a 20-24 or a 25-30 years old with different work class can be mapped in a >50K income class.

E.G Age + Work => Class income: >50K /<=50K



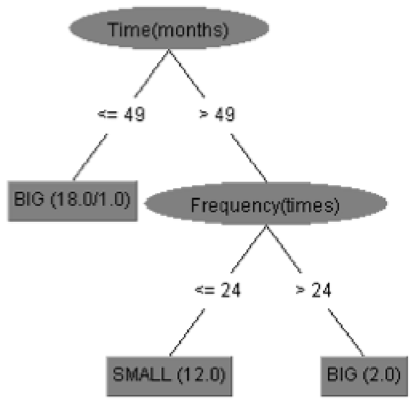
2 Analysis of Techniques

2.1 Blood Transfusion

Since this dataset is not so wide, we did not apply optimizations and we used the following techniques.

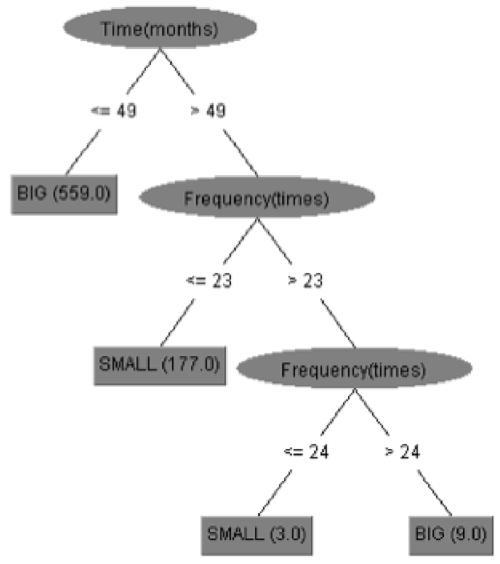
Decision Tree

On training set



```
Correctly Classified Instances      31    96.875 %
Incorrectly Classified Instances     1     3.125 %

=== Detailed Accuracy By Class ===

TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
          1       0.077      0.95     1         0.974     0.966   BIG
          0.923   0          1        0.923     0.96      0.966   SMALL
Weighted Avg.  0.969   0.046   0.97   0.969   0.969   0.966
```

On test set



```
Correctly Classified Instances      748      100    %
Incorrectly Classified Instances      0        0    %

=== Detailed Accuracy By Class ===

           TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
             1        0        1          1       1          1       BIG
             1        0        1          1       1          1       SMALL
Weighted Avg.  1      0        1          1       1          1
```

Naïve Bayesian Classifier

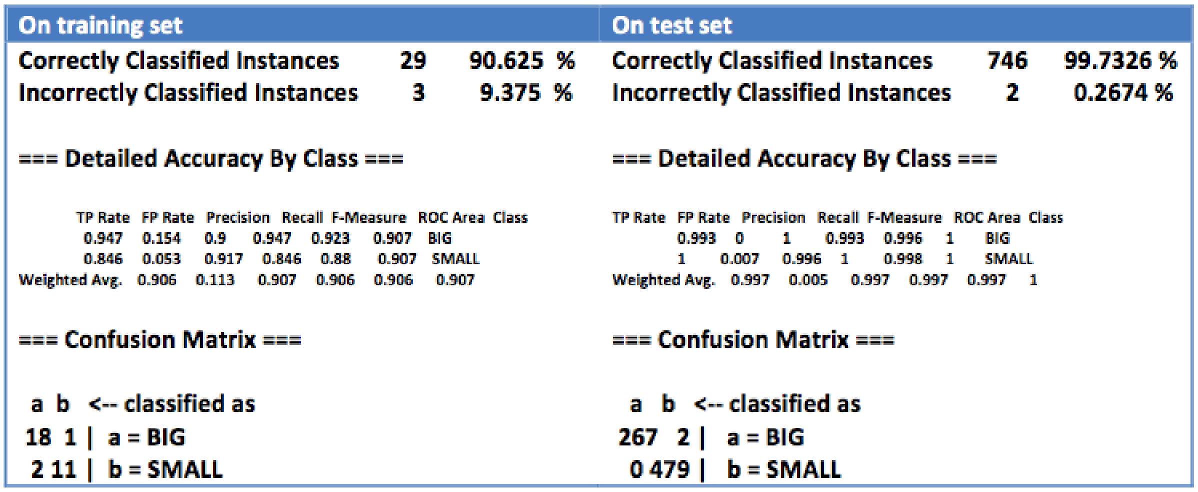
On training set

```
Correctly Classified Instances:    25     78.125 %
Incorrectly Classified Instances:   7     21.875 %

=== Detailed Accuracy By Class ===

          TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
           0.684    0.077    0.929     0.684    0.788      0.872    BIG
           0.923    0.316    0.667     0.923    0.774      0.872    SMALL
Weighted Avg.  0.781   0.174   0.822   0.781   0.782   0.872

=== Confusion Matrix ===

  a  b   <-- classified as
 13  6 |  a = BIG
  1 12 |  b = SMALL
```

On test set

```
Correctly Classified Instances:    719    96.123 %
Incorrectly Classified Instances:   29     3.877 %

=== Detailed Accuracy By Class ===

          TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
           0.969    0.118    0.988     0.969    0.978      0.981    BIG
           0.882    0.031    0.741     0.882    0.805      0.981    SMALL
Weighted Avg.  0.961   0.11    0.966   0.961   0.963   0.981

=== Confusion Matrix ===

  a   b   <-- classified as
659  21 |  a = BIG
  8  60  |  b = SMALL
```

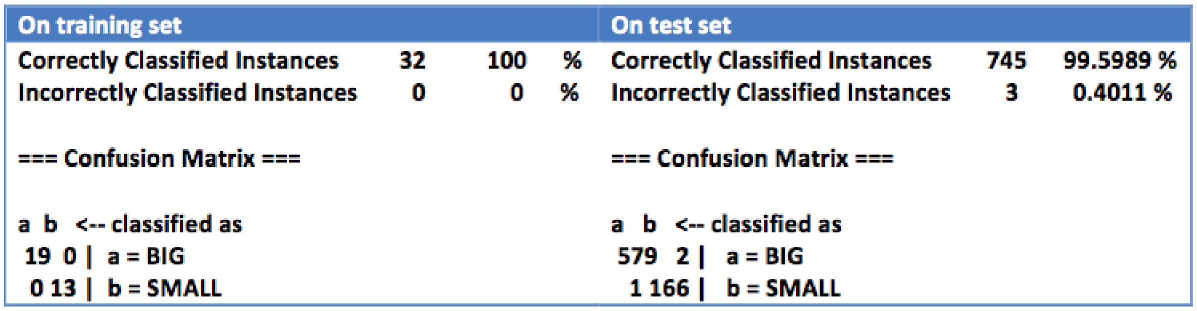
Logistic Regression



```
On training set
Correctly Classified Instances      29     90.625 %
Incorrectly Classified Instances     3      9.375 %

=== Detailed Accuracy By Class ===

         TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
          0.947    0.154    0.9        0.947   0.923      0.907     BIG
          0.846    0.053    0.917      0.846   0.88       0.907     SMALL
Weighted Avg.  0.906   0.113   0.907   0.906   0.906   0.907

=== Confusion Matrix ===

  a  b   <-- classified as
 18  1 |  a = BIG
  2 11 |  b = SMALL
```

```
On test set
Correctly Classified Instances     746    99.7326 %
Incorrectly Classified Instances     2     0.2674 %

=== Detailed Accuracy By Class ===

TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
          0.993    0        1        0.993   0.996      1        BIG
          1        0.007    0.996    1       0.998      1        SMALL
Weighted Avg.  0.997   0.005   0.997   0.997   0.997   1

=== Confusion Matrix ===

   a   b   <-- classified as
 267   2 |  a = BIG
   0 479 |  b = SMALL
```

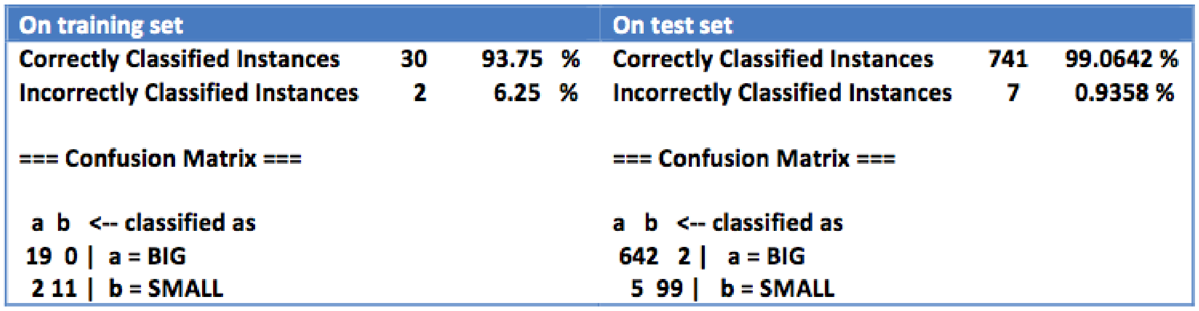
KNN Classifier (KNN = 1)



```
On training set
Correctly Classified Instances      32      100    %
Incorrectly Classified Instances     0        0    %

=== Confusion Matrix ===

 a  b   <-- classified as
 19  0 |  a = BIG
  0 13 |  b = SMALL
```

```
On test set
Correctly Classified Instances     745    99.5989 %
Incorrectly Classified Instances     3     0.4011 %

=== Confusion Matrix ===

 a   b   <-- classified as
 579   2 |  a = BIG
   1 166 |  b = SMALL
```

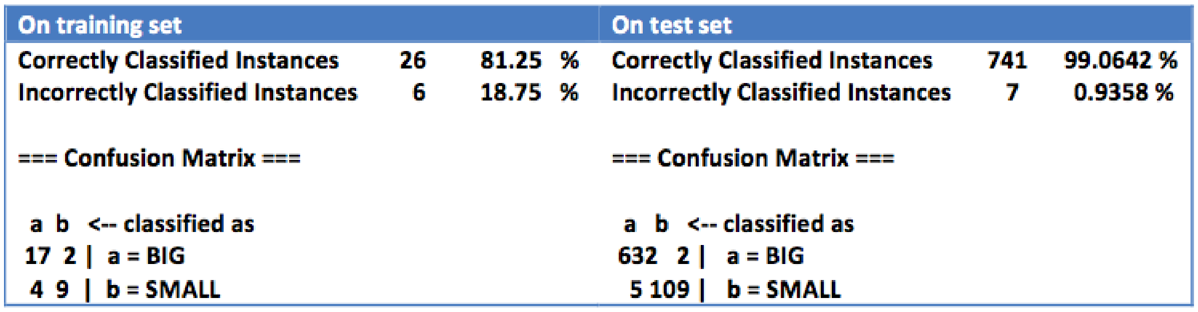
KNN Classifier (KNN = 3)



```
On training set
Correctly Classified Instances      30     93.75  %
Incorrectly Classified Instances     2      6.25  %

=== Confusion Matrix ===

  a  b   <-- classified as
 19  0 |  a = BIG
  2 11 |  b = SMALL
```

```
On test set
Correctly Classified Instances     741    99.0642 %
Incorrectly Classified Instances     7     0.9358 %

=== Confusion Matrix ===

 a   b   <-- classified as
 642   2 |  a = BIG
   5  99 |  b = SMALL
```

KNN Classifier (KNN = 5)



```
On training set
Correctly Classified Instances      26     81.25  %
Incorrectly Classified Instances     6     18.75  %

=== Confusion Matrix ===

  a  b   <-- classified as
 17  2 |  a = BIG
  4  9 |  b = SMALL
```

```
On test set
Correctly Classified Instances     741    99.0642 %
Incorrectly Classified Instances     7     0.9358 %

=== Confusion Matrix ===

   a   b   <-- classified as
 632   2 |  a = BIG
   5 109 |  b = SMALL
```

2.2 Census Income

Benchmark Framework

For analyzing the classification models by varying the data pre-processing techniques, we decided to prepare an evaluation framework (that is, a benchmark) including various possible combination of data pre-processing techniques. The following are the variables taken into account by preparing the framework.

Framework variables

Missing Values We decided to simply substitute the missing values of instances with the mode value of each attribute, as the method of removing instances is discouraged.

Outliers Outliers will be treated by applying discretization.

Attribute selection We will just remove the two attributes that we find of no particular interest. No other attribute selection techniques are taken into account, as we don't find comfortable with them.

Chosen Datasets We will use the four classification models on the full dataset.

Framework approach

The framework will be run as described in this paragraph. The four classification models will be run in this ways:

1. Without any preprocessing technique
2. By substituting missing values
3. By treating outliers
4. By treating outliers and substituting missing values
5. By removing fnlwgt and education-num
6. Removing fnlwgt, education-num, substituting missing values
7. Removing fnlwgt, education-num, treating outliers
8. Removing fnlwgt, education-num, substituting missing values, treating outliers

The best options for each classification algorithm will be chosen and used for further experiments.

Results

The following is the list of the results of the tests run using the configurations presented in the previous paragraph. We highlighted with different colors the casistics in which each algorithm performed better. We also enclosed in square brackets the best performing algorithm for each run. For steps 1, 2, 3 and 4 we also used different values for k parameter, as the results would not change otherwise, and keep the best found k for the next settings.

1. J48 87.8566%; Naïve Bayesian 83.4465%; Logistic Regression 84.896%; K-Nearest Neighbor (k=400): 85.3482%
2. J48 88.0102%; Naïve Bayesian 83.3144%; Logistic Regression 84.896%; K-Nearest Neighbor: (k=10): 85.9065%
3. J48 87.8566%; Naïve Bayesian 83.4557%; Logistic Regression 84.896%**;** K-Nearest Neighbor (k=7): 86.8063%
4. J48 88.0102%; Naïve Bayesian 83.2837%; Logistic Regression 84.896%; K-Nearest Neighbor (k=2): 89.9665%
5. J48 87.0858%; Naïve Bayesian 82.4729%; Logistic Regression 85.1663%; K-Nearest Neighbor (k=2): 89.8222%
6. J48 87.1380%; Naïve Bayesian 82.3193%; Logistic Regression 85.1663%; K-Nearest Neighbor (k=2): 89.856%
7. J48 85.1172%; Naïve Bayesian 81.1032%; Logistic Regression 85.326%; K-Nearest Neighbor (k=2): 87.1042%
8. J48 85.1172%; Naïve Bayesian 80.9588%; Logistic Regression 85.326%; K-Nearest Neighbor (k=2): 87.0643%

Delta accuracy results (best accuracy - worse accuracy):

- J48: 2,8930%
- Naïve Bayesian: 2,4969%
- Logistic Regression: 0,4300%
- K-NN (k=2): 2,9022%

Conclusions

The choice of two very different datasets allowed us to reason about different aspects of data mining. A small data size is easier and faster to be analyzed but less reliable for studies. A bigger one is slow to be analyzed (especially with K-NN model) but gives more reliable results.

Regarding the dataset blood transfusion, all models were based on different test sets. This allowed us to compare results between models that have been applied to the entire dataset and models applied to a training data subset. One advantage of the blood transfusion dataset is that it does not contain missing values. Therefore, experiments such as replacing or removing missing data were not taken into consideration on this one.

In addition, the blood transfusion dataset is composed also by a small amount of tuples, therefore the processing costs were dramatically cheap, even if techniques were tested on the entire dataset.

We also noticed that, while using a data training set, results accuracy decreases. We followed this kind of approach in order to demonstrate the previously shown comparisons, even if the dataset is small and the models applied to the entire dataset were inexpensive and reliable. All models were more accurate when tested on the entire dataset. The decision tree approach was the one that had the best performances.

The benchmark on data preprocessing techniques we run on census-income dataset showed that the classification models react very differently. The accuracy range can be improved by circa 3% by applying data preprocessing techniques. Only Logistic Regression does not show improvements when varying data preprocessing techniques.

	Decision tree	Naïve Bayesian Classifier	Logistic Regression	KNN classifier
Input Parameter	The attributes of a tuple	The posterior probabilities of Hypothesis H based on additional information	The measure of the total contribution of all independent variables used in the model.	A new unknown tuple for which a class has to be assigned.
Principle	The attributes of a tuple are tested against the decision tree and a path is traced from the root to a leaf node which holds the prediction for that tuple	Given a tuple X, the classifier will predict that X belongs to the class having the highest posterior probability conditioned on X.	Given x representing the exposure to some set of risk factors, LR predicts the probability of occurrence of an event by fitting data to a logistic curve, f(x), which represents the probability of a particular outcome	Nearest-neighbor classifiers compare a given test tuple with training tuples that are similar and described by n attributes and are stored in n-dimensional space ➔ Find the k-nearest tuples from the training set to the unknown tuple
ALG. FORMULA	At start, all the training tuples are at the root. Then, tuples are partitioned recursively based on selected attributes. The test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain). We stop when all samples for a given node belong to the same class or there are no remaining attributes for further partitioning ➔ majority voting.	At start, compute P(C) The prior probability ofC_i. Each class can be computed based on the training tuples. Then compute each independent probability for attribute xi in reference with Class C and multiply P $(X_i$|C). Example: The probability of class C to be yes is P(C) = 9/14. The amount of attribute Xi being test with respect to C is P$(x_i \| C_i) = \frac{xi=test}{Ci=yes} = \frac{2}{9}$ Finally, compute P(X|C_i)P(C_i) for each class ➔ The naïve Bayesian Classifier predicts C=yes for tuple X	Since x is defined as x =$\beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k$ the LR is $\frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_k X_k}}{1 + e^{\beta_0+\beta_1 X_1+\cdots+\beta_k X_k}}$ Estimate the parameters using the Maximum Likelihood Function and then by computing the partial derivatives of the log likelihood, equate each partial derivative to zero, and solve the resulting nonlinear equations	The ALG assigns for the unknown tuple the most common class among its k-nearest neighbor. When k=1 the unknown tuple is assigned the class of the training tuple that is closest to it. To measure the distance we can use the Euclidean distance $\sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2}$ **Choose K:** If k=1 the classification will be 1:1 sensitive to other data. If k=n we'll suffer high noise. Go through various K's and choose one giving lowest misclassification error!