Mining Information from US Census Bureau Data

Sebastian Harvey

Computer Science, Aston University, Birmingham, UK
harveys@aston.ac.uk

Wednesday, May 30, 2007

Abstract

This paper provides an insight into the development of classification models using CRISP-DM development cycle, particularly machine learning algorithms which can predict individual incomes using data provided by the US Census Bureau. The training data consisted of a result set of over 35,000 instances, with a small quantity (6%) of missing data. A prediction dataset with approximately 15,000 examples was supplied, which had the classifications (income) values missing. Models on the basis of both equal cost and cost matrices were developed and fine tuned by experimenting with their particular parameters, where the prediction data was then subject to the strongest of the models.

1.0 Introduction

The overall goal of this project is to develop classifier models to generalise whether a person (defined as an anonymous instance) has an annual income of less or equal to fifty thousand or greater than fifty thousand.

For this project, the Weka (Waikato Environment for Knowledge Analysis) data mining toolkit was used. This toolkit, written in Java at the University of Waikato, provides a considerable library of algorithms and models for classifying and generalising data.

Although not required in this project, Weka provides functionality for solving problems developed using Java and has superior compatibility of Java developed data mining applications.

To ensure accuracy, all development and testing of models will follow the CRISP_DM process.

- Exploration of the problem
- Exploration of the data and its information (meta)
- Data preparation
- Model development
- Evaluating outcomes

2.0 Data Exploration

For this problem, the training dataset which was to be processed was stored within an ARFF file, which was compatible with the Weka data mining toolkit.

The ARFF file could be viewed using a normal text-editor to view the information and its meta data contained within it. Its meta structure is displayed below.

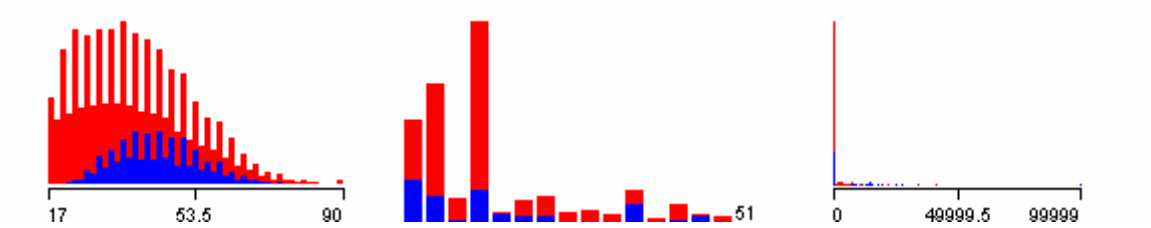
```
@relation cwork07train

@attribute age numeric
@attribute workclass {'Private' ... 'Never-worked'}
@attribute fnlwgt numeric
@attribute education {'Bachelors' ... 'Preschool'}
@attribute education-num numeric
@attribute   marital-status   {'Married-civ-spouse'   ...   'Married-AF-
spouse'}
@attribute occupation {'Tech-support' ... 'Armed-Forces'}
@attribute relationship {'Wife' ... 'Unmarried'}
@attribute race {'White', 'Asian-Pac-Islander' ... 'Black'}
@attribute sex {'Female', 'Male'}
@attribute capital-gain numeric
@attribute capital-loss numeric
@attribute hours-per-week numeric
@attribute native-country {'United-States'... 'Holand-Netherlands'}
@attribute class {'>50K', '<=50K'}
```

The meta data provided much insight into what the data was, how to use it, and the relationship between entries. This was exploited during the data exploration phase to gain a better understanding of the data. Initial exploration of the dataset is listed below:

- There were a total of 32,561 continuous and discrete instances.
- There are 15 attribute.
    - It appears that two attribute listed are mirrors of one another. *Education* and *Education-Num*, where *Education-Num* is a numeric representation of the other.
    - There is an odd looking *fnlwgt* attribute, which contains numeric values. Following research on this attribute [Chris Shoemaker, www.cs.wpi.edu/~cs4341/C00/Projects/fnlwgt/ accessed 10 March 2007], it appears to have no relation to the income that each instance has. Therefore has no predictive power and can be ignored.
- There is a mix of numeric and nominal attribute.
- Some of the data contained missing values, delimited by '?'.
    - Missing values only appeared to be in numeric variables.
- Using Weka visualisation tools – particularly scatter plots – the analysed data did not show strong class separation.
- There did not appear to be any spelling mistakes, which would cause an instance to be incorrectly classified.

- Some attribute appear to have an imbalanced distribution of values.
    - *Age*, *education*, *capital-gain* and *capital-loss* is very skewed towards the lower values
        - *Capital-gain* and *capital-loss* had very strong 0 values, and little occurrence of other values. (Data pre-processing may wish to discretize these values into two bins (‘0’ and ‘1’ or more)
- The large dataset maybe computationally expensive on algorithms such as *k*-nearest Neighbour and Logistic Expression.



The images show the non-Gaussian appearances of the *age*, *education* and *capital-gain* tables.

3.0 Data Pre-processing

The data pre-processing phase of the experiments is aimed at tidying up the data to help the algorithms [which are to be applied] run smoother, returning more accurate results.

Having investigated the dataset, it is apparent that there are approximately 6% of instances which have one or more missing values. These missing values have to be addressed to help the algorithms generalise more accurately.

In addition, there are a number of outliers in the data [rouge results that are offset somewhat from the majority of its class]. These outliers will also affect generalisation, and thus should be appropriately treated.

As noted earlier, the dataset may have to be sampled to allow for computationally expensive algorithms to perform on it.

These issues were addressed by:

1. Two experiments were setup, using the J4.8 algorithm. Each running on the full dataset. The first of the two experiments classified a full dataset, where missing values were replaced (using Weka’s unsupervised filters). By default, the values are replaced with the mode value in each attribute. The second test was applied to the full dataset, where missing values were removed.

   The results were:

Dataset	Accuracy	Leaves on tree	Size of tree
Replaced	**86.27**	**728**	**901**
Removed	85.72	572	736

Table 1, comparing data pre-processing

The results show that by replacing missing values, the model is able to generalise better to the data. It also enabled a larger tress to be built. Larger trees allow for more values to be fitted, thus reducing the chance of *over fitting*. Because of the increase in both the tree and its generalisation percentage, future experiments will be applied to the dataset, replacing missing values when appropriate.

2. Outliers will be treated by applying discretization, letting the filter chose the appropriate number of bins. This is later tested.

3. A sample dataset will be created, only using 10% of the total instances. This equates approximately 3256 instances to work with. Weka's *resample* filter will be used, which allows a sample to be chosen at random. We chose the random seed 4. We also set the *biasToUniformClass* parameter to be set to 1.0. This ensured that the data used accurately modelled the original dataset. IE: Maintaining a realistic distribution of the final results.

The sample dataset was also chosen for all experiments to allow for fair testing of each algorithm. It can be assumed, that while using a smaller sample, results accuracy will decrease. However, this will be proportionate across all models.

Other modifications could be made, IE: improvements in hardware or altering heap sizes to allow for computationally expensive models to be executed on a full dataset. However, this is outside the scope of this project and re-sampling will be used to address this problem.

4.0 Classification Models

In order to find a classifier algorithm(s) to best generalise the data, this section concentrates on identifying various classifiers, identifying which work better than others and choosing the most efficient algorithms and further refining their parameters further increase their generalisation accuracy.

This is broken down into five stages:
1. Benchmark models
2. Attribute selection
3. Model development
4. Combining models
5. Cost based modelling

4.1 Benchmark Models

Several models were chosen and applied to the sample dataset. These models included Naive Bayes, k-nearest neighbour, Logistic regression, J4.8, OneR, KStar, JRip, ZeroR. Each algorithm was applied using its default parameters. *K*-nearest neighbour's *k* value was chosen automatically by cross-validation, the value chosen is displayed on each table where appropriate.

Model	Accuracy
Naive Bayes	75.06
k-nearest neighbour (*k*=5)	77.33
Logistic regression	81.08
J4.8	81.20
OneR	75.61
KStar	69.99
JRip	80.98
ZeroR	51.53

Table 2, initial benchmark models

After applying each model to the sample dataset, we experimented at removing outliers to help classification. We repeated all experiments on the modified data and the following results were returned.

Model	Accuracy
Naive Bayes	**80.19**
k-nearest neighbour (*k*=9)	**79.36**
Logistic regression	**81.78**
J4.8	79.66
OneR	75.61
KStar	**80.49**
JRip	79.48
ZeroR	51.53

Table 3, initial benchmark models after applying pre-processing

The emboldened entries on the table highlight which values improved after applying discretization. It was noted that the ZeroR and OneR models did not either improve or worsen. We applied further tests on ZeroR and OneR on a complete dataset and noted very little change.

So far, this paper has concentrated on a sample subset which had the *biasToUniformClass* parameter to be set to 1.0. We decided to create a new subset (random seed 4), but setting the *biasToUniformClass* parameter to 0. We generated the following results:

Model	Accuracy
Naive Bayes	83.16

k-nearest neighbour (*k*=7)	82.00
Logistic regression	85.56
J4.8	86.33
OneR	80.15
KStar	77.88
JRip	83.41
ZeroR	75.65

Table 4, changing the sampling bias

There was clear indication that the *biasToUniformClass* parameter on the sampling was affecting the result accuracy. It was decided to create a new subset where this parameter was set to 0 for the remainder of the experiments. Missing values would be treated by replacing them with the mode values.

At this stage, we chose the following models to further develop:

1. Naive Bayes
2. k-nearest Neighbour
3. Logistic Regression
4. J4.8
5. JRip

The other models were dropped as they showed little sign of development in the early tests.

4.2 Attribute Selection

Having examined the data earlier in this paper, it was noted that attributes (such as *fnlwgt*) had no – or very little – predictive power. These attributes can be removed. This will not only speed up efficiency of the models, but will help them generalise better.

Firstly, we looked at several methods of identifying the best attributes to use and methods of ranking all attributes, to identify which are the best. (This selection was made from the 10% sample subset).

- *Cfs* picked seven attributes: 1, 5, 6, 8, 11, 12, 13
- *ChiSquared* ranked the attributes (in order of importance): 8, 6, 4, 7, 5, 11, 1, 13, 10, 12, 2, 14, 9, 3
- The *Information Gain* measure ranked the attributes (in order of importance): 8, 6, 7, 1, 4, 5, 11, 13, 10, 12, 2, 14, 9 ,3
- The *Symmetric Uncertainty* measure ranked the attributes (in order of importance): 11, 6, 8, 1, 5, 13, 12, 4, 7, 10, 2, 14, 9, 3

Upon study of these attributes, it appeared that the most important attributes were*: age(a1), education(a2), marital-status(a3), relationship(a4), capital-gain(a5), capital-loss(a6)* and *hours-per-week(a7).*

Only using the attributes identified, we classified our five main models on the data sample and recorded their progress.

Model	Accuracy
Naive Bayes	79.45
_k_-nearest neighbour (_k_=10)	**83.20**
Logistic regression	85.07
J4.8	84.58
JRip	**83.72**

Table 5, applying default parameters

It was noted that by removing all attributes, other than those selected – the speed in which the models executed was much improved. This was largely notable during the classification of *k*-nearest. However, most of the models were less accurate. Those that did show improvement, such as JRip, only reported a mere 0.31% increase.

As an additional measure, we ran the five models again on the sample subset, this time, only removing the *fnlwgt* and *education-num* attributes. We recorded the progress.

Model	Accuracy
Naïve Bayes	82.03
_k_-nearest neighbour (_k_=7)	**83.04**
Logistic regression	**85.74**
J4.8	84.45
JRip	**84.24**

Table 6, removing unnecessary attributes

Three out of the five models showed slight improvement when removing the two attributes. The other two models, Naïve Bayes and JRip did not chart in improvements, but their final accuracy variance was only slightly negative.

Both attribute selection experiments were compared against the sample dataset (where *biasToUniformClass* parameter was set to 0). Models that did improved are emboldened.

It was decided that for future model development, the *fnlwgt* and *education-num* attributes would be removed.

4.3 Model Development

Having experimented with the models this far, it was noted that models such as J4.8, will

classify the complete dataset. It was originally decided – for fairness – that each model would use an identical subset sample. However, it was in the nature of the project to find the best models that would work on the data.

J4.8 on a 10% sample classified at 81.20%. However, earlier experiments (on handling missing values) classified the full dataset considerably higher, with the highest classification value being 86.27%. At this stage it was decided that computationally inexpensive models would generalise on the complete dataset (with pre-processing for outliers and missing values), while others would generalise the sample subset.

Where a complete set has been used, this is clearly identified during each model development.

4.3.1 Naïve Bayes

Naïve Bayes' strongest accuracy (83.16%) was on the sample subset with no pre-processing. The model was further tested on a full dataset (with *fnlwgt* and *education-num* attributes removed and missing values replaced). This yielded an accuracy of **82.31%**.

This value was not as high as it was with a sample subset, however, a larger dataset would help limit over fitting. It was decided that the experiments would continue with the larger dataset.

One issue that affected Naïve Bayes was that numeric values. Some were not Gaussian in appearance, such as age, which shows a positive skew. For this phase, we applied discretization on the model. Attributes with any different values (*age*) were discretized into 10 bins, while those with few values (*capital-loss* & *capital*-gain) were discretized into two. The accuracy fell to **81.09%**.

Navive Bayes' *useSupervisedDiscretization* parameter was then set to true, this yielded **83.11%.** We then experimented by setting the *useKernelEstimator* parameter to true. This then yielded a record value accuracy of **84.91%**.

4.3.2 *k*-nearest Neighbour

So far, this models strongest performance (83.20%) was on the sample subset of 10% with only 7 selected attributes and a *k* of 10.

We further investigated this by applying the model again on the same sample subset and altering the models weightings. This allows you to adapt the influence of the neighbours according to their distance. The results are recorded as follows:

Weighting	Accuracy

Standard	83.20
1/distance	83.10
1-distance	**83.93**

Table 7, experimenting with parameters

1-distance demonstrated a record accuracy for this model at **83.93%**.

4.3.3 Logistic

Logistics was tested on the sample subset (with *fnlwgt* and *education-num* attributes removed and missing values replaced), as this yielded the models highest accuracy. We repeated the experiments, this time altering the ridge parameter. The results were noted:

Ridge Parameter	Accuracy
1 x 10-8	85.74
1 x 10-4	85.74
1	**85.81**
10	85.59
20	85.47
30	85.19
50	84.95
100	84.95

Table 8, experimenting with parameters

Logistics with a ridge parameter of 1 yielded the highest accuracy yet for this model at **85.81%**

4.3.3 Decision Trees – J4.8

So far, the models strongest performance (84.58%) was with a limited sample (with only 7 attributes). For developing this particular model further, J4.8 was tested on the full dataset (with *fnlwgt* and *education-num* attributes removed and missing values replaced).

Complexity Control	Parameter Value	Accuracy
Post-pruning	0.35	85.90
Post-pruning	0.30	85.88
Post-pruning	**0.25**	**86.02**
Post-pruning	**0.20**	**86.02**
Post-pruning	0.15	85.98
Post-pruning	0.10	85.78
Post-pruning	0.05	85.72
Reduced error pruning	TRUE	85.38

Minimum number of objects	10	85.85
Minimum number of objects	20	85.77
Minimum number of objects	30	85.75

Table 9, experimenting with parameters

The results show that post-pruning yields he best results, with a very modest 0.20 setting. This is the highest value this model has generalised too, at **86.02%.**

4.3.4 JRip

JRips highest accuracy value so far, was (84.24%) on a sample subset. For development, we continued developing on this sample subset (with *fnlwgt* and *education-num* attributes removed and missing values replaced), changing the number of folds that were being made during each classification process.

Folds	Accuracy
1	75.67
2	84.36
3	84.24
5	84.33
6	**84.67**
7	84.52
10	83.93
12	84.09

Table 10, experimenting with parameters

Setting JRip's fold parameter at 6 yielded the most accurate result for this model so far at **84.67%.**

4.4 Combining Models

Having identified eight models initially, we have chosen the five most accurate and further developed them to improve their accuracy. We now combined them into a committee, with voting used to make a unified decision.

The models chosen and their parameters were:

1. Naïve Bayes (*useKernelEstimator* set to true)
2. *k*-nearest Neighbour (*k*=10, weighting of 1-distance)
3. Logistics (*ridge* parameter set to 1)
4. J4.8 (post pruning set at 0.20)
5. JRip (*folds* set to 6)

As some models were developed on the full dataset, and others on a sample subset

(sometimes with attributes removed), the voting committee would be tested on the 10% sample subset with only 7 attributes [that were selected during the attribute selection phase].

The overall unified performance (with 10-fold cross validation) was **85.59%.** This figure was not the strongest value found during development. J4.8 (post pruning set at 0.20) was the strongest performance, almost 0.5% more accurate.

The reasons for this are explored later in this paper. Although it should be noted that 85.59% is a strong value – even compared against J4.8 – and has shown considerable development since initial benchmark tests, where the lowest value was 51.53% (ZeroR).

4.5 Cost-Based Modelling

For cost modelling, it was deemed that, *the cost of misclassifying a high income individual is 10 times that of misclassifying a low income individual*.

The cost matrix file used in this development is shown below:

```
%Rows  Columns
2      2
%Matrix elements
0.0    10.0
1.0    0.0
```

In the following subsections, we ran the models again with the parameters set at their strongest using the *CostSensitiveClassifier* meta-model.

4.5.1 Naïve Bayes

Was ran on a full dataset (*useKernelEstimator* set to true). The value returned from the confusion matrix represents a strong costing.

7476	365
8868	15852

Which corresponds to a cost of: 365 * 10.0 + 8868 * 1.0 = **12,518**

4.5.2 Logistic

Was ran on a sample subset of 10% (ridge parameter set to 1). The value returned was almost 90% lower than Naïve Bayes

741	51
799	1665

Which corresponds to a cost of: 51 * 10.0 + 799 * 1.0 = **1,309**

4.5.3 *k*-nearest Neighbour

Was ran on a sample subset of 10% (*k*=10, weighting of 1-distance). The results were similar to that of Logistic.

751	41
1064	1400

Which corresponds to a cost of: 41 * 10.0 + 1064 * 1.0 = **1,474**

4.5.4 Decision Trees – J4.8

Was ran on a full dataset (post pruning set at 0.20). This yielded the weakest value of all cost models and was inconsistent to the results of equal-cost modelling.

7371	470
8652	16968

Which corresponds to a cost of: 470 * 10.0 + 8652 * 1.0 = **13,352**

4.5.5 JRip

Was ran on a sample subset of 10% (*folds* set to 6). There were no notable increases in cost from Logistic.

735	57
981	1483

Which corresponds to a cost of: 57 * 10.0 + 981 * 1.0 = **1,551**

4.5.6 Model Selection

Logistics has proved itself to be the best performing model for the unequal cost. Each model was tested using the same parameters that were used in the equal cost development.

While this has several advantages, such as accuracy, further modifications could still be made and tested on the models to help increase their cost output.

While a voting committee of unequal cost models would be insightful, Logistics has performed consistently well.

5.0 Evaluation & Conclusion

The combined efforts of the voting committee returned 85.59% accuracy. While this value is very strong, especially with such as a large dataset, it was not the strongest result returned.

J4.8 was almost 0.5% more accurate than the voting committee (even though the J4.8 model, with the same parameters, was a member of the committee).

There maybe several reasons for this.

Some models were developed using a full dataset, where their processing costs were inexpensive, while others – notably *k*-nearest Neighbour – where developed using only a 10% sample subset. The problems for working with such a small subset are obvious.

Other reasons include the number of attributes used for various models. Some models performed better than others with less attributes. While other models preferred to run using a complete set of attributes.

Incidentally, Logistics had the best unequal cost result at **1,309**.

One disadvantage to this research is that different models were tested on different sized datasets. This was because of processing cost. Ideally, all models would have been developed on an identical dataset. Because of this, the reliability of the results cannot be assured.

*Post project note: Since completion of this paper, the combined voting committee was again tested on the sample dataset. After removing fewer attributes, the model classified better than initial development. This score was higher than all member models, at **87.03%**, proving that much work can still be made to the algorithms to further improve their accuracy.*

Two output files have been included with this paper:

1. Predictions using the voting committee model
2. Predictions using Logistic unequal costing model