Improving Classification Accuracy for Adult Dataset*

Vinit Muchhala, Shanmukha Bharat Vakalapudi
Department Of Computer Science,
George Mason University,
Fairfax, VA 22030.
{**vmuchhal,svakalap**}@gmu.edu

Abstract: Adult Dataset is one of the most classic data mining dataset, it has been around for almost 2 decades now. This report discusses the classification tasks carried out on the Adult Dataset. This task is to predict whether, given the data about a particular adult, makes more than $50k per year or less. Our task is to gather the information about the previous results and study the classifiers used in previous experiments and improve on those results.
Keywords: Adult Dataset, Classification, Data mining

---

* *This paper is written as part of the term project for our Academic course, Theory and Applications of Data Mining(CS 659) at Volgenau School of Engineering, George Mason University*

1. Introduction

This report discusses the performance and accuracy of various classifiers, used previously. How these classifiers fared in the task of classifying the Adult Dataset, how we tried to improve the performance of these classifiers. In the process, we also preprocessed the data in several ways, to give us a more technical insight into the data at hand, understand the relation between attributes, how strongly they are related.

Naive Bayes, one of the most popular classifier, was used to classify the data, Naive Bayes works by calculating the posterior probability of each attribute with respect to class and assigns the new instance to the class with the most posterior probabilty. J48, a decision tree algorithm was used. NBTree algorithm was also used. NBTree is a hybrid of J48 and Naive Bayes classifiers, in that, it builds a decision tree like J48 but uses Naive Bayes at its nodes to assign the instance to a particular class. k-Nearest Neighbor(kNN from now on) Search algorithms were also used, 2 versions of it, kNN with 1 neighbor and kNN with 3 neighbors.

Several Bootstrap methods and Bootstrap aggregators were used, which were not used previously on the Adult Daset. AdaBoost was used. Random Forests tree classifier was used and Bagging classifier was also used.

2. Previous Usage

This dataset was used previously for classification purposes by Ron Kohavi in the paper "*Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid*" in the book "*Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*".

The data was used in the past to predict whether the level of income is greater than or less than a 50K mark by studying the demographics such as age, sex, etc. The previous implementation and classification processes were carried out using MLC++. The Background Error Rate in Table 1 is the error rate obtained by previous implementation by using MLC++, that is the error rate we will try to improve on. The Classification tasks in MLC++ were carried out using the default parameter.

Classifier/Error-rates	Background Error rate
J48	**15.54%**
KNN(1)	**21.42%**
KNN(3)	**20.35%**
NBtree	**14.10%**
Naïve Bayes	**16.12%**

Table 1 : Previous Error rats

3. Software Used

Weka : **http://www.cs.waikato.ac.nz/ml/weka/**
R : **http://www.r-project.org/**
Microsoft Excel : **http://ww.microsoft.com/**

4. Tasks

- Data source
- Data selection
- Data pre-processing
  - Data Cleaning
  - Feature selection and extraction; Dimensionality reduction
- Data mining
- Performance assessment
- Observed results
- Data Visualizations

4.1. Data Source

Original owners of database : US Census Bureau.
Donor of database : Ronny Kohavi and Barry Becker, Data Mining and Visualization. Silicon Graphics. e-mail: ronnyk@sgi.com.
We collected the dataset from UCI Machine Learning repository website.

4.2. Data Selection

Dataset 'Adult' is extracted from a larger census dataset. This dataset has 48842 instances and 14 attributes as part of training data. Out of 14 attributes 8 are nominal attributes and 6 are continuous attributes. We are using, as the previous users, 1/3rd of the instances for testing and rest for training.

Entire data is taken from a single source. The class labels and data was in different files, so integration had to be done. Moreover the missing values were not properly represented and were not detected by Weka as missing values, hence the dataset had to be standardized.

4.3. Data Pre-processing:

Data preprocessing includes processes that help us look at data and understand the distribution, get a feel for the data.

4.3.1. Data cleaning:

'Adult' has missing values for 3 attributes. They are

a. 2799 (6%) for 'workclass' attribute.
b. 2809 (6%) for 'occupation' attribute.
c. 857 (2%) for 'native-country' attribute.

We removed the instances with missing values from the dataset manually. We used 'MakeDensityBasedClusterer' on the dataset and visualized the result. There were no outliers in the dataset. There were no redundant instances in the dataset. Hence there were no inconsistent values in the dataset.

4.3.2. Dimensionality Reduction:

'Adult' has an attribute 'education-num' which represents the same data as another attribute 'education' but in numerical form. Hence, this is considered as a redundant attribute and it is removed from the dataset. Result is the dataset with 13 attributes.

We used the attribute evaluator on the result of the modified data set and calculated the information gain for each attribute. The result is as follows:

0.16537 7 relationship
0.15653 5 marital-status
0.11452 10 capital-gain
0.09754 1 age
0.09359 4 education
0.08412 6 occupation
0.05814 12 hours-per-week
0.05072 11 capital-loss
0.03717 9 sex
0.01572 2 workclass
0.00846 13 native-country
0.00838 8 race
0 3 fnlwgt

Attribute 'fnlwgt' has information gain of 0. Hence, we have removed this insignificant attribute from the dataset. This results in 12 attributes.Attribute 'workclass' has 9 labels out of which 'without-pay' and 'never-worked' represent the same meaning that says "they don't have any income". They are also of insignificant number when compared to the remaining labels. Hence, we generalized these 2 labels into a single label 'without-pay-or-never-worked'. The graph denoting their insignificance is shown below.

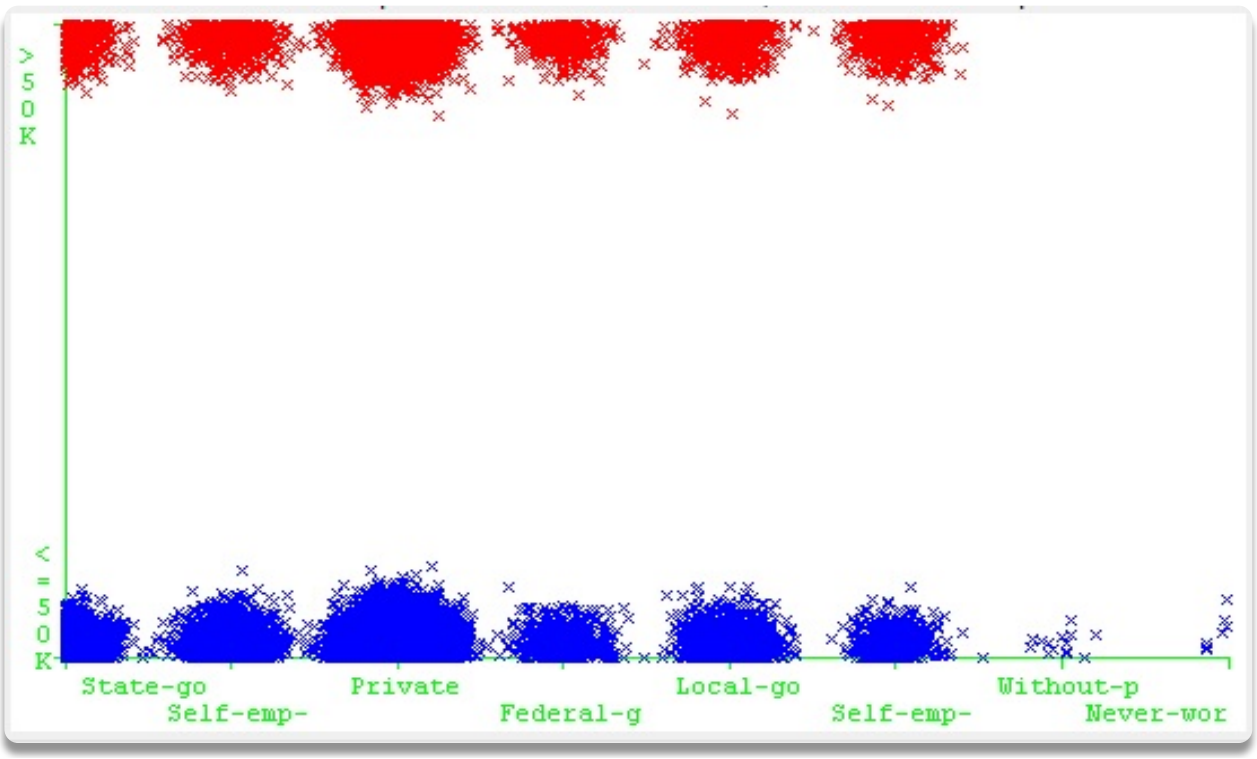

Figure 1 : Visualization of 'Workclass' vs 'Class'

Finally, the output of the data pre-processing step is 30162 instances and 13 attributes including the class.

4.4. Data Mining:

We used 8 classifiers for data mining. They are:

1. K-nearest neighbors (1)
   K-nearest neighbors (3)
2. J48
3. NB Tree
4. Naïve Bayes
5. Bagging
6. AdaBoost
7. Random Forests

4.5. Performance Assessment

In this section we will discuss each classifier, the preliminary results of the classifiers and how we achieved a increase in the accuracy of the classifier.

The preliminary results are results that are run in Weka with default settings, the final results are obtained in R, Weka by varying the parameters to give better results.

4.5.1. k-Nearest Neighbor Search Algorithms

k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The *k*-NN algorithm is among the simplest of all machine learning algorithms.
Both for classification and regression, it can be useful to weight the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones.

Classifier	Error rate – Preliminary	Error rate – Final
kNN (1)	20.59%	16.62%
kNN(3)	18.43%	15.59%

Table 2 : Comparison of *k*NN Algorithms

The above table shows the final error rate compared to the preliminary error rate. There were improvements for both KNN algorithms.
For preliminary results, Weka default parameter of normalized attributes were used. On using the un-normalized attributes, this drastic improvement was achieved.
Linear Search Algorithm was used. It is a brute force search algorithm for nearest neighbor search.
The simplest solution to the NNS problem is to compute the distance from the query point to every other point in the database, keeping track of the "best so far". This algorithm, sometimes referred to as the naive approach, has a running time of $O(Nd)$ where N is the cardinality of S and d is the dimensionality of M. There are no search data structures to maintain, so linear search has no space complexity beyond the storage of the database. Naive search can, on average, outperform space partitioning approaches on higher dimensional spaces.

4.5.2. J48

J48 is an open source Java implementation of C4.5 algorithm. C4.5 is an algorithm used to generate a decision tree developed by Ross Quinlan. C4.5 is an extension of Quinlan's earlier ID3 algorithm. The decision trees generated by C4.5 can be used for classification, and for this reason, C4.5 is often referred to as a statistical classifier.

Classifier	Error rate – Preliminary	Error rate – Final
J48	**14.62%**	**13.59%**

Table 3 : Comparison of J48

From the above table we can see that there is just a slight improvement for J48 algorithm, by sing binary splits for nominal values at nodes and reduced Confidence factor to 0.05 from Weka default 0.25.
Confidence Factor is the pruning factor in Weka, a higher Confidence factor leads to less pruning while a lower confidence factor leads to increased pruning.

4.5.3 Naive Bayes

A naive Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions. A more descriptive term for the underlying probability model would be "independent feature model".

Classifier	Error rate – Preliminary	Error rate – Final
Naïve Bayes	**18.21%**	**15.35**

Table 4 : Comparison of Naive Bayes

From the Error rate was decreased, more than just nominally, by using Kernel Estimator for estimating continuous attributes, rather than Gaussian distribution. kernel density estimation (KDE) is a non-parametric way to estimate the probability density function of a random variable. Kernel density estimation is a fundamental data smoothing problem where inferences about the population are made, based on a finite data sample.

Kernel Estimator vs Normal Distribution : Using a Kernel Density Estimation Function instead of a Normal Distribution in the Naive Bayes is called a Flexible Naive Bayes.

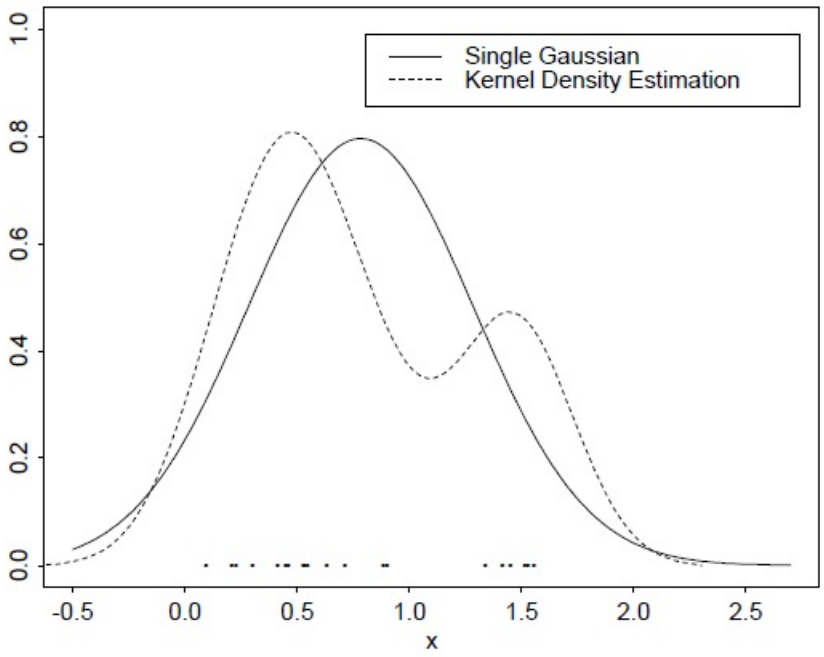

Figure 2: The effect of using a single Gaussian versus a kernel method to estimate the density of a continuous variable.

Figure 2 : Kernel Density

From the above graph it can be seen that, Kernel Density function fits more with more precision than Gaussian distribution for continuous attributes.
To further the understanding how a kernel density estimator works, we will differentiate the mathematical working of kernel estimator with a Normal distribution.
A Normal distribution looks like this:

$p(X = x \mid C = c) = g(x,\mu_i,\sigma_c)$
where $g(x,\mu_i,\sigma_c) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

Whereas the Kernel density estimator looks like:

$p(X = x \mid C = c) = \frac{1}{n}\sum g(x,\mu_i,\sigma_c)$

Thus we can see that although the Kernel density function fits the data more tightly, it requires more space, to store all the estimations of the previous instances.

4.5.4. NBTree
NBTree is a hybrid of J48 and Naive Bayes classifiers, in that, it builds a decision tree like J48 but uses Naive Bayes at its leaf nodes instead of assigning an instance to a particular class.

4.5.5 Bagging
Bagging, aka bootstrap aggregation, is a relatively simple way to increase the power of a predictive statistical model by taking multiple random samples (with replacement) from your training data set, and using each of these samples to construct a separate model and separate predictions for your test set. These predictions are then averaged to create a, hopefully more accurate, final prediction value.
We Used J48 as the Base Learner, with confidence factor set to 0.05 and using binary splits for nominal attributes with a Bag Size of 47% of total number of instances

4.5.6 AdaBoost
The basic idea of boosting is to associate a weight with each observation in the dataset. A series of models are built and the weights are increased (boosted) if a model incorrectly classifies the observation.
We used “Rattle” a GUI package in R for implementing AdaBoost. We mixed the training and the test data sets into a single .csv file and took a percentage split of 70% to 30% for training and testing. To use the dataset in “Rattle” the maximum number of labels allowed for an attribute are 32. But, our dataset contains an attribute ‘Country’ which has more than 32 labels. So, we generalized the labels by combining the insignificant labels and reduced the total number of labels to 9.
We used a discrete boosting algorithm which is the default parameter in rattle. Number of trees were 500 and all the remaining parameters were default values.

Classifier	Error rate – Final
AdaBoost	14.25%

Table 5 : Classifier Accuracy for AdaBoost

4.5.7 Random Forests

A random forest is an ensemble (i.e., a collection) of un-pruned decision trees. Random forests are often used when we have large training datasets and particularly a very large number of input variables (hundreds or even thousands of input variables). The algorithm is efficient with respect to a large number of variables since it repeatedly subsets the variables available.
We tried random forests on our dataset with the default values for all the parameters. Results were as follows:

Classifier	Error rate – Final
Random forests	13.97%

Table 6 : Classifier Accurac for Random Forest

Even if the dataset contains less number of input variables results are quite good relative to other algorithms.

4.6. Data Visualization:

We tried to find patterns in data by just visualizing them. The following are the visually discovered patterns in the Adult dataset which are statistically significant and interesting. This analysis was completed solely using the visualization and not by applying a clustering algorithm.

Age vs Relation:

Graphing participants' age versus their spouse type shows that young people tend to have never been married. This make common sense as the majority of people younger than 18 have not been married. Another interesting pattern is that the maximum age group of divorced couples follows the maximum age group of married couples. Interpreting this data, one can see that on average most couples who get married and then divorce do so typically after 5 years of marriage.

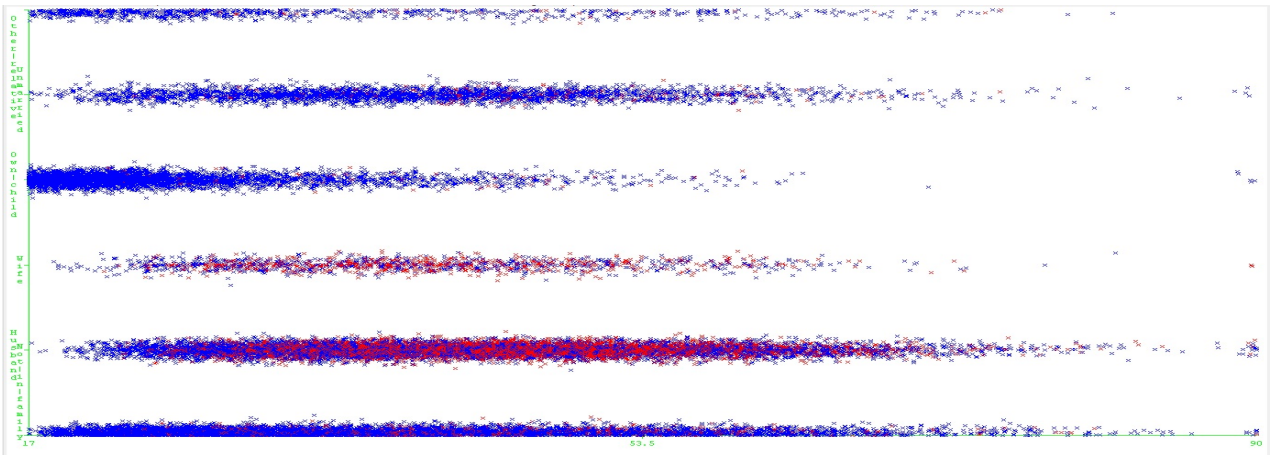

Figure 3: Age vs Relation

Salary vs Education:
By observing the class variable, whether people make under or over 50K, we can see that people who finish college are significantly more likely to earn over 50K.

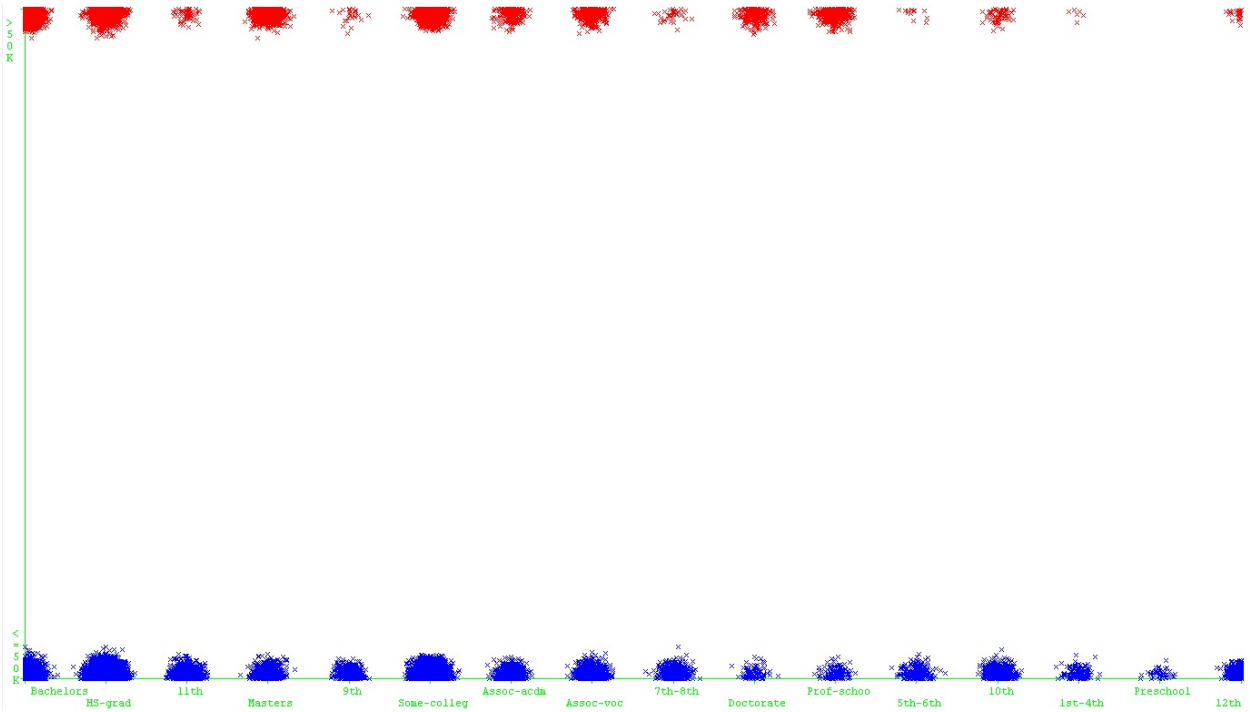

Figure 4 : Class vs Education

Relation vs Marital Status:
This is the plot of top 2 attributes, evaluated from the InfoGainAtrributeEval evaluator.

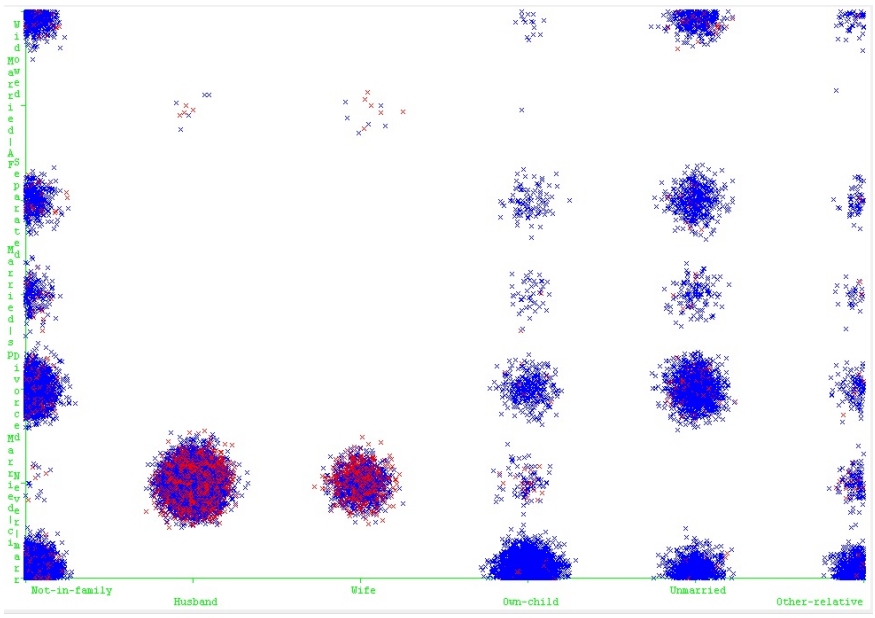

Figure 5: Relation vs Marital Stats

Age vs Hours per Week:

The plot shows the obvious relation. The decrease in number of hours worked per week decreases as the age of a person increases.

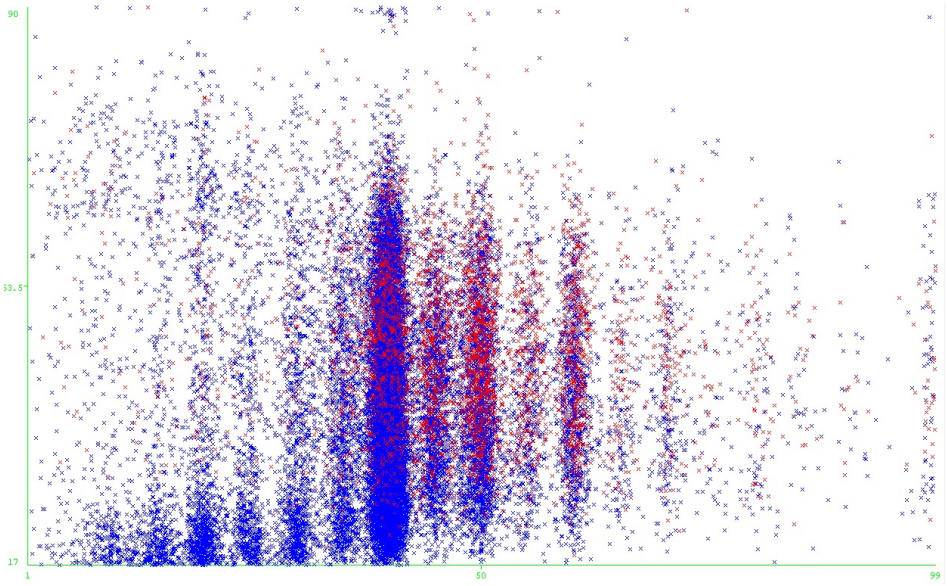

Figure 6 : Age vs Hours

4.7. Observed results:

Results of the classifier's on the testing dataset are compared to the results from previous usage and our preliminary results.

Classifier/Error-rates	Existing Error rate	Preliminary Error rate	Final error rate
J48	**15.54%**	**14.62%**	**13.59%**
KNN(1)	**21.42%**	**20.59%**	**16.62%**
KNN(3)	**20.35%**	**18.43%**	**15.591%**
NBtree	**14.10%**	**14.11%**	**13.94%**
Naïve Bayes	**16.12%**	**18.21%**	**15.35%**
Bagging	**-**	**-**	**13.34%**
Random forest	**-**	**-**	**13.97%**
Ada-Boost	**-**	**-**	**14.31%**

Table 7 : Summarized table of results of all classifiers

5. Discussion and Conclusion:

Already given that Naive Bayes assumes that there is independency within the attributes and the fact being evident that, the current dataset as a bit of dependency between the attributes, it can be a major reason for Naive Bayes' eccentric behavior. We were able to reduce the error rate in the Adult data set, marginally for some cases and by more for some others. The main highlights remain using Bagging to do classification using J48 decision tree, which gave us our best results. We believe that NBTree, J48 and Random forests also gave good results. For future work, we would want to train the NBTree algorithm to use Kernel estimator for the Naïve Bayes at the nodes of the tree it generates.

6. References

Bache, K. & Lichman, M. (2013). *UCI Machine Learning Repository* [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

George H. John & Pat Langley (1995). "*Estimating Continuous Distributions in Bayesian Classifiers*" In proceedings of the 11th conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, San Mateo, 1995.

Jason Anderson, Arlo White, Brett Bojduj & Michael Human(2009). In "*UCI Adult Dataset Analysis An Analysis Project.*"

Naive Bayes classifier. (2014, April 24). In Wikipedia, The Free Encyclopedia. from "*http://en.wikipedia.org/w/index.php?title=Naive_Bayes_classifier&oldid=605648856*".

Ron Kohavi(1996). "*Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid*" In KDD-96.

Bootstrap aggregating. (2014, April 28). In Wikipedia, The Free Encyclopedia. from *"http://en.wikipedia.org/w/index.php?title=Bootstrap_aggregating&oldid=606136088"*

AdaBoost. (2014, April 16). In Wikipedia, The Free Encyclopedia. from *"http://en.wikipedia.org/w/index.php?title=AdaBoost&oldid=604513894"*

Random forest. (2014, April 15). In Wikipedia, The Free Encyclopedia. from *"http://en.wikipedia.org/w/index.php?title=Random_forest&oldid=604320928"*

C4.5 algorithm. (2014, February 6). In Wikipedia, The Free Encyclopedia. from *"http://en.wikipedia.org/w/index.php?title=C4.5_algorithm&oldid=594269597"*

K-nearest neighbors algorithm. (2014, April 16). In Wikipedia, The Free Encyclopedia. from *"http://en.wikipedia.org/w/index.php?title=K-nearest_neighbors_algorithm&oldid=604493028"*