Compared Results of Nave Bayes Classifiers Across Five Data Sets

Kyle J Brekke brekke.kylej@gmail.com
Gianforte School of Computing
Montana State University
Bozeman, MT 59717-2220, USA

Ren H. Wall renwall@protonmail.ch
Gianforte School of Computing
Montana State University
Bozeman, MT 59717-2220, USA

Nicholas J. Rust nicholasrust@protonmail.com
Gianforte School of Computing
Montana State University
Bozeman, MT 59717-2220, USA

Alexander Mershon alexander.mershon@student.montana.edu
Gianforte School of Computing
Montana State University
Bozeman, MT 59717-2220, USA

Editor: Kyle J Brekke

Abstract

In this paper we tested the effect of introducing noise into five data sets used to evaluate the Nave Bayes algorithm. Our implementation of Naive Bayes utilizes a 10-fold cross validation, and is measured using a paired t test. While the introduction of noise to the *breast-cancer-wisconsin*, *house-votes-84*, *glass*, *iris*, and *soybeans* data sets certainly appeared to affect our implementation's accuracy on the data sets, we were unable to make many conclusions as a result of the small size of many of the data sets and low performance of the algorithm in our tests. As such we were unable to produce significant results for all but the *breast-cancer-wisconsin* data set, of which we concluded had support for the hypothesis that the addition of noise would significantly reduce the ability of Naive Bayes to classify items in the data set.

Keywords: Machine Learning, Statistics, Nave Bayes Classifiers

1. Introduction

The purpose of this project is to accurately analyze data sets in a manner which serves as a "gentle introduction" to machine learning. In this report, we will utilize Nave Bayes Classifiers to categorize data into different classes via the use of training and testing sets. All data sets used in this report are courtesy of the University of California, Irvine, School of Information and Computer Sciences' Machine Learning Repository (Dua and Graff, 2017).

Particular recognition is provided for the *breast-cancer-wisconsin* data set, produced by Dr. William H. Wolberg (Mangasarian and Wolberg, 1990; Wolberg, 1995).

The testing of the data will take place in the form of 10-fold cross-validation utilized on two instances of each data set. The first will contain the original, unaltered values of each data set (aside from those removed or discretized during preprocessing), whereas the second sets will have 10% of their respective features shuffled in order to introduce an additional level of noise to the sets. After experimentation has been conducted on the sets, we will compare the results, and determine whether a significant difference exists between the efficacy of Naive Bayes on the unaltered and altered sets. In general, we expect that the data which has been shuffled will have a significant decrease in efficacy on data sets with few categories, and similar accuracy on data sets with many categories.

1.1 Hypotheses

1.1.1 breast-cancer-wisconsin

With 699 entries and only two classes, the breast-cancer-wisconsin data set is larger than most, in conjunction with its 9 attribute categories we hypothesize that the extra noise will have less of an impact on this set than many others.

1.1.2 glass

At 214 entries, 6 classes, and 9 attributes the glass data set is a decent middle-ground. The larger number of classes already makes the data set more difficult to classify, and we hypothesize this will only be exacerbated by increased noise.

1.1.3 house-votes-84

At 435 entries, 2 classes, and 16 attributes the house votes dataset has one of the larger attribute sets seen in this data sample. We hypothesize that even though the data set has only two classes, the increased number of attributes and noise will lead to a marginal decrease in precision.

1.1.4 iris

With 150 entries, 3 classes, and 4 attributes the iris data set has a comparatively low number of classes and attributes, we hypothesize that since 1/4 of the data is going to be noise, this data set will be extremely affected by the increased noise.

1.1.5 soybean-small

Soybean small has 35 attributes, 47 entries, and 4 classes. With 4/35 of the data being noise, a good number of classes, and a minuscule data set, we hypothesize that soybean-small will see a marginal decrease in performance.

2. Implementation

2.1 Summary

We started by cleaning the data and splitting each of the five data sets by class into a respective *.csv* file for each class. These are taken in and turned into DataFrames, then split into ten parts. One of each of the ten parts from every DataFrame is then added to a testing set. These testing sets are merged into ten different training and testing set pairs, one testing set is returned as the testing set, and the other four are merged and returned as the training set. These sets are then used to train and test, that data is analyzed and run through the loss functions.

2.2 Set Creation and 10 Fold Cross Validation

Training and testing set creation begins by importing and cleaning the data from a *.csv* file, using the python package *pandas*. The retrieved data is then placed in a DataFrame - a *pandas* object which functions as a table. We then use these DataFrames to conduct our tests using 10-fold cross-validation, an application of the k-fold cross-validation method discussed in *Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms* (Dietterich, 1997). The result of our 10-fold cross-validation can be written as such:

$$k = \frac{1}{10}\sum_{i=1}^{10} c(L = x_j, T = x_i) \tag{1}$$

1. Where $c(L, T)$ is the result of our Naive Bayes implementation, L is the training set, and T is the set which the resulting classifier will be tested on.

2. Where $x_i \in \hat{x} = \{x_1, x_2, ..., x_{10}\}$, given $\hat{x}$ is the set of all entries in the current data set, and $x_1, x_2, ..., x_{10}$ are roughly-equal sized subsets of $\hat{x}$.

3. Where x_j is the union of all $x \in \hat{x}$, excluding x_i.

In order to implement 10-fold cross validation, the *.csv* files for each class are split into ten parts (by uniform random sampling) and merged with exactly one set from each *.csv* file, creating ten evenly distributed sets which have a proportional amount of each class relative to the original data set. A notable exception to this procedure though, is that the calculated subset scale is rounded down to accommodate datasets which do not have a total number of entries evenly divisible by ten. Any additional entries are added to the first class to ensure no values are dropped. All but one of the subsets are merged together, with each subset being used as the testing set once. This creates ten unique training/testing set pairs, which are then passed on for training and classifying.

2.3 Training and Classifying

For our classifier we went with a nested dictionary approach. We accomplished this by first inputting a pandas DataFrame into a method which counts the number of instances of each attribute response for each class and attribute. This cumulative count is put at the leaf of our nested dictionary. The dictionary had the following hierarchy: First key is class, second is either "Probability" or "Attribute," the actual attributes are the keys under "Attribute," this maps to the unique attribute responses, and finally, each unique attribute response maps to the number of times it is counted for that class. The "Probability" key maps to an integer of the elements that are of that specific class.

This is a template for the final nested dictionary we will use for our classification. Next we call our trainer method that goes through and uses the equation for the probability of each attribute response for the given class to give us a fraction we can use to calculate the probability of getting this response in this attribute within this class. Then we go and change the count of each class into the probability of that class, their count divided by the total number of elements. After it finalizes each probability, it puts the finished dictionary into a global dictionary that is used by our classify method.

Our classify method takes in a testing set, which is a pandas DataFrame, iterates through each row and, for each class we saw in our training set, we multiply each attribute response probability together then multiply the class probability and add the total to a tuple list. We then proceed to the next class. After each class is tested for a row, it finds the class with the largest probability and adds that class to the predictions list, which is returned after all the data in the training set has been run. If there was no case of an attribute response in the training set but there is one in the test set, then an amendment is made to the nested dictionary to create a probability with a count of 0 for that response.

3. Experimental Approach

3.1 Data Preprocessing

There were a number of steps taken for data preprocessing. First, manual modifications were made to a few of the data sets before any cleaning methods were applied. All of the original *.data* files were converted into *.csv* documents. All *.csv* files had a header row containing feature names added as well. Additionally, the ID attribute was removed from the glass and Wisconsin breast cancer data sets, due to its being unnecessary for our purposes.

Of the five total data sets at our disposal, one of them contained missing values in its entries. This required us to develop a method for handling them. We concluded that we would use a 5% threshold to determine whether entries containing missing values would be imputed, or stricken from the set outright. That is, if the total number of entries containing missing values in a data set was less than 5% of the total number of entries for that set, then those entries would be removed from it.

Within the single data set with missing values – the Wisconsin breast cancer set – roughly 2.69% of all entries contained missing values, which was under our selected threshold. As such, all entries with missing values were removed from the Wisconsin breast cancer set during cleaning.

After cleaning was completed, it was necessary for us to modify the *iris* and *glass* data sets to fit our model. Since the data in these sets are real values, they conflicted with our implementation of Naive Bayes as a result of the model being designed for binary and discrete/categorical feature spaces. Our solution to this problem was to categorize each set of attribute values in these data sets into roughly even-sized categories, such that the attributes would be categorized from least to greatest. With this approach, we chose to categorize the *iris* features into six groups, and the *glass* data set into eight groups. Our reasoning for these values was based on loose testing as to what number of categories would yield the greatest classification accuracy in the non-shuffled data sets.

3.2 Testing Procedures

Each data set will be tested in two contexts: 1) Using the base values as they are assigned in the processed data sets, and 2) using modified copies of the cleaned data sets which have had values randomly reassigned within 10% of their respective features.

Once these instances have been tested, we will compare their results using paired t tests to evaluate whether our implementation of Naive Bayes is more or less efficient when data is scrambled when compared to the base case. The comparisons we will be making in this report are the results of three loss functions: *error*, *precision*, and *recall.*

$$error = \frac{c_{01} + c_{10}}{c_{11} + c_{10} + c_{01} + c_{00}} \tag{2}$$

$$precision = \frac{c_{11}}{c_{11} + c_{10}} \tag{3}$$

$$recall = \frac{c_{11}}{c_{11} + c_{01}} \tag{4}$$

The values c_{11}, c_{10}, c_{00}, and c_{01} in the above equations are respectively the counts of all true positives, false positives, true negatives, and false negatives which resulted from the classifications conducted.

Error is representative of the total number of misclassified entries, relative to the total number of all classified entries. The actual value of *error* indicates the overall ratio of misclassifications in the dataset.

Precision indicates how many classifications into a given category where correct. The actual value of *precision* is the ratio of all true positive classifications, relative to all classifications asserted to be positive.

Recall represents the number of correct classifications of a given class out of all actual instances of that class in the dataset.

Once the loss functions have been calculated for all classes, they are averaged for their corresponding data sets. We then apply the paired t test illustrated in *Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms* (Dietterich, 1997):

$$t = \frac{\bar{p}\sqrt{10}}{\sqrt{\frac{\sum_{i=1}^{10}(p^{(i)} - \bar{p})^2}{9}}} \tag{5}$$

1. Where $\bar{p}$ is the overall average of one of our loss functions in a shuffled data set, subtracted from the average of the same loss function in its corresponding unshuffled data set.

2. Where $p^{(i)}$ is the difference of an average loss function for a data set i, and the corresponding value of its paired shuffled data set.

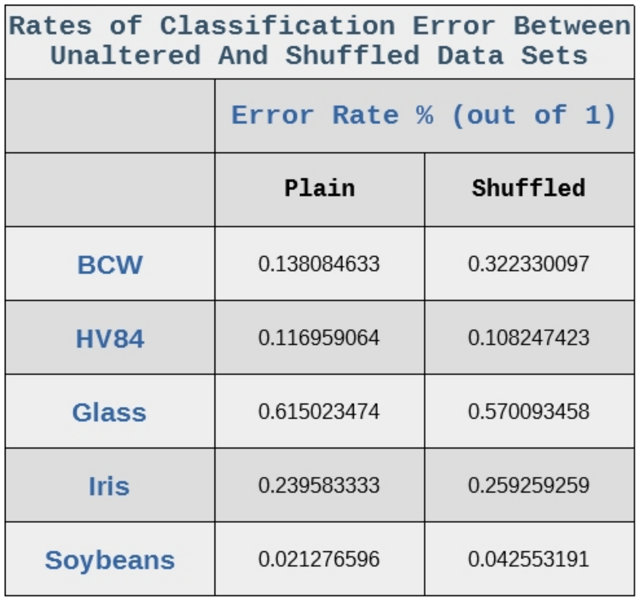
The resulting t values we acquire can then be used to determine the significance of any differences we observe between the unaltered and shuffled data sets.

4. Results

After running our tests, we acquire a text dump of data which allows us to gleam a wide array if information about our implementation and data sets.

Of note, we are able to observe that the lowest performing data set in both the plain and shuffled tests was by far the *Glass* data set with an error rate of around 61.502% in the plain data tests, and 57.009% in the shuffled tests. Meanwhile the highest performing data set was the *Soybean* set, with the lowest error rate of roughly 2.128% in the plain tests, and 4.256% in the shuffled tests (see *Figure 1*).

We can also observe that two of our five datasets actually *increased* in accuracy after the data was shuffled, though neither seemed to by a particularly major amount. Despite this though, we can also observe marginally stronger trend in the other two loss functions which align with our initial assumptions (see *figures 3* and *4*).



Rates of Classification Error Between Unaltered And Shuffled Data Sets

	Error Rate % (out of 1)	
	Plain	**Shuffled**
BCW	0.138084633	0.322330097
HV84	0.116959064	0.108247423
Glass	0.615023474	0.570093458
Iris	0.239583333	0.259259259
Soybeans	0.021276596	0.042553191

Figure 1: Table containing the error rates of our five data sets.

Based off both of the above figures, we have a mildly stronger case for our hypothesis. Despite what we may be able to derive from our findings, we are not able to determine the significance of our data until we evaluate the results of our paired t tests.

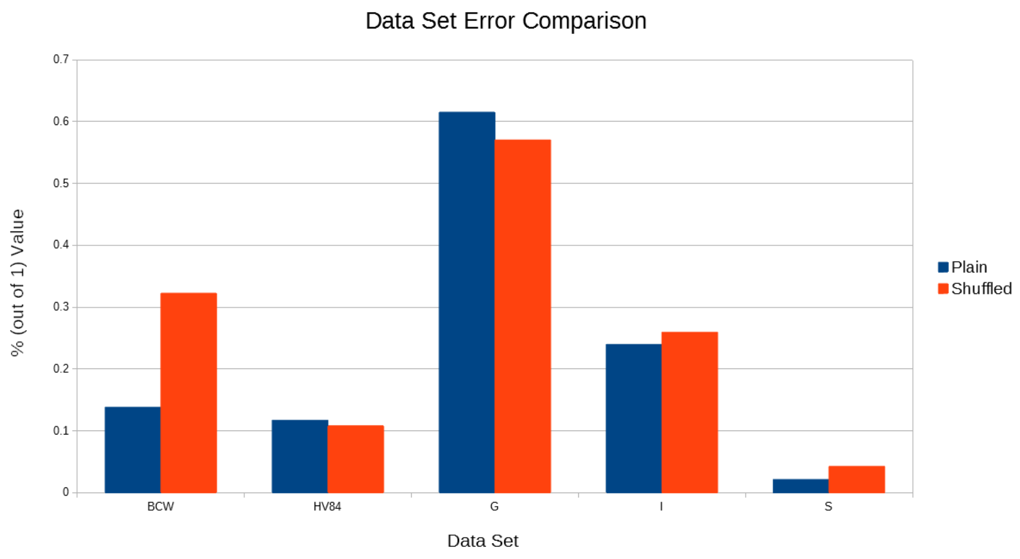


Figure 2: Graph comparing the error rates of our five data sets. From left-to-right: Breast Cancer in Wisconsin (BCW), House Votes in 1984 (HV84), Glass (G), Iris (I), and Soybeans (S)

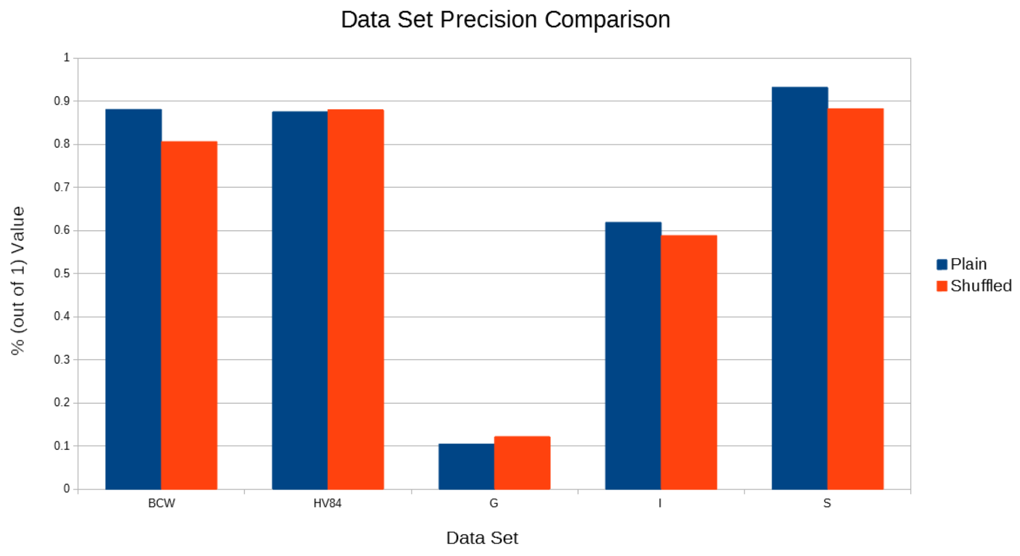


Figure 3: Graph comparing the precision values of our five data sets

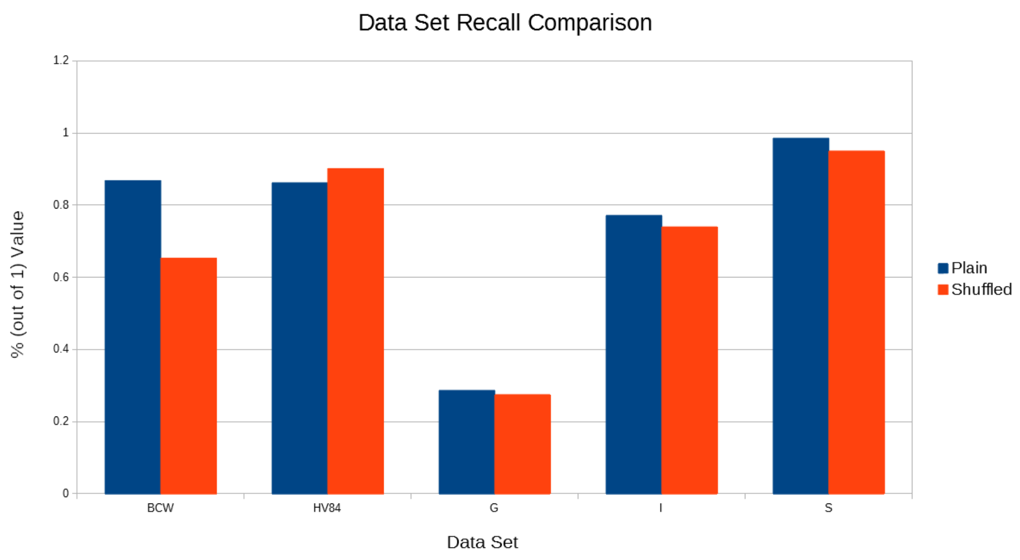


Figure 4: Graph comparing the recall of our five data sets.

5. Conclusions

Using the *error* values of our datasets does not allow us to make many statements regarding the affects of adding noise to the dataset. For *breast-cancer-wisconsin*, *house-votes-84*, *glass*, *iris*, and *soybeans*, the t values are:

$$t_{9,.975} = |-6.90699| = 6.90699$$
$$t_{9,.975} = 0.40645$$
$$t_{9,.975} = 0.99174$$
$$t_{9,.975} = |-0.29689| = 0.29689$$
$$t_{9,.975} = |-0.55708| = 0.55708$$

Given these values, we are only able to reject the null hypothesis for the *breast-cancer-wisconsin* data set, with 95% confidence. We are unable to reduce our confidence interval, due to the magnitude of such a reduction. To consider even the second highest t value significant would require us to lower our confidence to 60%, while all other data sets would still be non-significant. As such, we can not make conclusions about any data set aside from the *breast-cancer-wisconsin* set.

Because we are able to consider the results of our experimentation with *breast-cancer-wisconsin* significant, we support the experimental hypothesis that introducing noise to a data set has a measurable effect on the accuracy of Naive Bayes on the set.

6. Summary

In this report, we utilized Naive Bayes to classify data into different categories through training and testing sets. The sets which we tested on were the *breast-cancer-wisconsin* (Mangasarian and Wolberg, 1990), *house-votes-84*, *glass*, *iris*, and *soybeans* datasets from the UCI Machine Learning Repository (Dua and Graff, 2017).

The testing of the data took place in the form of 10-fold cross-validation run on two instances of each data set, an unaltered instance and a shuffled instance. After experimentation was complete, we compared the results to determine whether a discernible difference in accuracy arose between the plain and shuffled sets.

Our initial assumptions led us to believe that the introduction of noise would significantly reduce the efficacy of Naive Bayes on the data sets. However, the results of our paired t test did not yield sufficient data to form a conclusion on any set other than the *breast-cancer-wisconson* set. We suspect that this was a result of the relatively small size of the data sets, and the poor performance of the Naive Bayes model during our test.

From the conclusions we are able to develop, we can affirm that the data supports the experimental hypothesis for the *breast-cancer-wisconsin* set, that the addition of noise reduced the effectiveness of our model significantly.

References

Thomas G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. December 1997.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

O. L. Mangasarian and W. H. Wolberg. Cancer diagnosis via linear programming. *SIAM News, Volume 23, Number 5*, 1990.

Dr. WIlliam H. Wolberg. Breast cancer wisconsin (original) data set. https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Original), July 1995. Accessed on 3 Sept. 2019.