Machine Learning CSCI447 Project 1: Naive Bayes

Nicholas Stone nicholas.stone2@student.montana.edu

Matteo Björnsson matteo.bjornsson@student.montana.edu

Abstract

In this paper we present a small experiment analyzing the performance of the discrete Naive Bayes classifier learned and tested on five data sets. In addition to these five data sets, we fabricate five "noisy" versions of these data sets to test and compare. We use 10 fold cross validation to evaluate the performance of the Naive Bayes classifier using 0/1 Loss and F1 score. We hypothesized that data sets with large numbers of attributes would perform well, and data sets with a skewed distribution of class occurrences or few attributes would perform poorly. We found that our results did not fully match our expectations and that noise did not dramatically affect performance. More testing would need to be done to further evaluate the performance of Naive Bayes overall.

Keywords: Naive Bayes

1. Introduction

Naive Bayes is a classic supervised Machine Learning algorithm that is often used as a baseline and works surprisingly well given its simplicity (Hand and Yu, 2001). The algorithm is based on the premise of Bayesian Learning where the probability of a classification, given a new set of evidence, is predicted based on the probability distribution of past evidence and classification. Naive Bayes is a useful algorithm because, for certain problems, its performance is surprisingly comparable to more advanced algorithms (Friedman et al., 1997).

For this project, we implement, train, and test a discrete value Naive Bayes classifier using five different data sets. We then introduce noise into ten percent of the data set features and repeat the process. In section 2, we provide some background information on the learning algorithm and classifier. In section 3, we describe our hypotheses driving the experimental design. Section 4 details the experimental design, providing our results and discussion in section 5.

2. Background Information

Naive Bayes is derived from Bayes' Rule[1] by assuming that all observed features are conditionally independent. This simplification reduces the probability complexity from exponen-

1. Bayes' rule was introduced by Reverend Thomas Bayes in the 18th century (Bayes, 1763).

Table 1: Data set summary

Data set	Character	Type	#Sample	#Attr.	#Class	Notes
Soybean	Multivar	Categ.	47	35	4	One class has twice the occurence.
Iris	Multivar	Real	150	4	3	Even split between classes.
Cancer	Multivar	Categ.	699	9	2	Class split 65/35. Some missing attr.
Vote	Multivar.	Categ.	435	16	2	Class split 45/55. Some missing attr.
Glass	Multivar	Real	214	9	6	Uneven distribution of classes.

tial to linear, providing a simple and efficient algorithm for classification. The algorithm requires a set of correctly classified evidence to train.

A Naive Bayes classifier chooses the best class by essentially comparing new evidence against the frequency of that evidence having occurred in the past, given each possible class and the probability of that class occurring at all. A multinomial Naive Bayes classifier can be generated using the following equations. Equation 2 describes the attribute value frequencies per given class value. This is the function that is used to train the model. The $+1$ and d terms represent smoothing so that attributes not seen in the training set will not collapse the classification calculation. Equation 1 describes the frequency of each class found in the training set.

The Naive Bayes Classifier is composed of Equations 3 and 4, which pick the maximum a posteriori hypothesis. Simply put, the classifier picks the most likely hypothesis based on past evidence. When given a new feature vector, the effective probability of each possible class assignment is calculated, and the class with the largest associated value is chosen (Sheppard, 2020).

$$Q(C = c_i) = \frac{\#\{\mathbf{x} \in c_i\}}{N} \tag{1}$$

$$F(A_j = a_k, C = c_i) = \frac{\#\{\mathbf{x} \in a_k) \wedge (\mathbf{x} \in c_i)\} + 1}{N_{c_i} + d} \tag{2}$$

$$C(\mathbf{x}) = Q(C = c_i) \times \prod_{j=1}^{d} F(A_j = a_k, C = c_i) \tag{3}$$

$$class(\mathbf{x}) = \underset{c_i \in C}{\operatorname{argmax}}\, C(\mathbf{x}) \tag{4}$$

Finally, let us briefly describe the data sets used for this project, summarized in Table 1. All of the data sets are multivariate. Some of them are real valued and need to be binned into categorical values. The size of the data sets varies from 47 to 699, the number of attributes for each data set varies largely as well, and most data sets have 2 or 3 classes.

3. Hypotheses

In this section, we will discuss how we expect the Naive Bayes classifier to perform on each data set. Additionally, we will review how we expect the introduction of noise in the data sets will change this performance. A data set we expect to perform poorly is the Iris data

set. This is due to small number of attributes that describe the classifications. We expect that the more attributes available to the algorithm, the better it will perform. Here there are nearly as many class values as attributes.

When inspecting the Glass data set we discovered that some of the attribute values had a high precision but very little change between examples. This drew concerns that the binning of the continuous valued attributes would remove these differences. In our experimental design we bin real values to equal width bins. For this reason, we expect the glass data set may perform poorly. Additionally, this data set has what may be an unfavorable ratio of number of attributes to number of class values as well as a distinctly unbalanced distribution of class occurrences in the data set.

Due to the uneven distribution of Republican versus Democrat classes in the Vote data set, we wonder if the algorithm will perform poorly here. Additionally, the nature of this data set domain–politics–challenges the core algorithm assumption that all attributes are conditionally independent. Here we suspect that the attributes are dependent on each other in complex ways that may hurt the algorithm performance. However, there is a large number of attributes compared to the possible classifications, which leads us to suspect this data set may perform well.

The Cancer data set has a more significant split in the distribution. Since there is a larger number of one class over the other, it is possible the algorithm will always be more likely to classify a record as the more numerous class on new test data. However, the fact that there are only two possible classifications with a decent number of attributes leads us to believe this data set may perform adequately.

We expect Soybean to perform well because there are a very large number of attributes for each data point. Though the small sample size may lead to a high variability in the classifier accuracy, we think that over the course of ten experiments the data set will do well.

Finally, we speculate on the consequence of adding noise to ten percent of the attributes of each data set. We predict that overall this introduction of noise will not impact the accuracy of the classifier very much. This is due to the fact that for the most part, if one attribute is shuffled, nine still remain. We are concerned that ten percent effectively becomes 25% in the case of the Iris data set due to the fact that we need to shuffle at minimum one column. Here we may see an impact of the noise. If a much larger fraction of the features were shuffled, we would expect to see a significant difference in classifier accuracy.

4. Experimental Design

The experiment is divided into several distinct sections: Preprocessing, Learning and Classification, and Evaluation. Figure 1 illustrates the overall workflow for the experiment. Data sets are preprocessed and piped into the training algorithm and classifier. The results of the classification test are evaluated and saved to file.

4.1 Preprocessing

All of the data sets are preprocessed before they are introduced to the learning and classification algorithms. The algorithms implemented in this experiment require discrete data

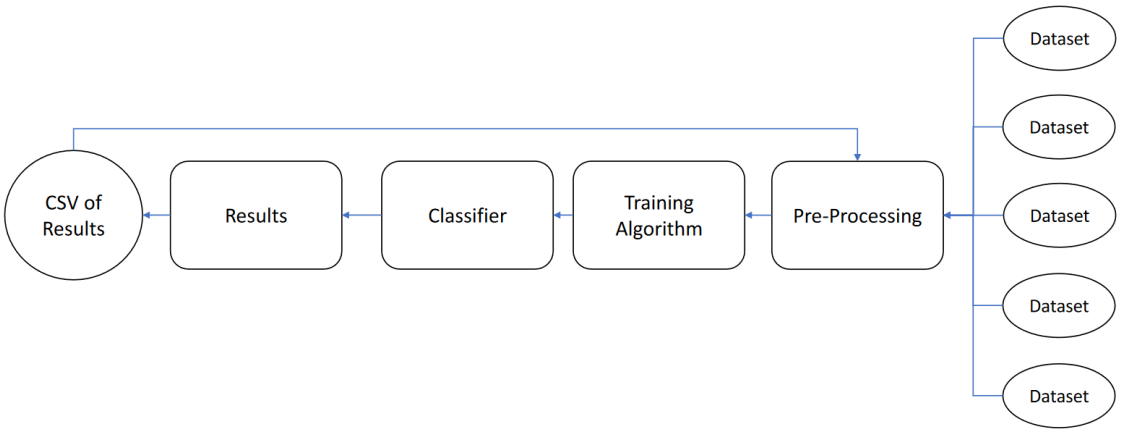

Figure 1: *Flowchart of the experimental design.*

as well as complete data. To transform the real valued data into discrete valued data we divide the range of values present in a feature and then assign those values to equal width bins. The number of bins that the feature values are divided into is a tunable parameter. For the two real valued data sets we trained our model on a random 90% of the data set and tested on the remaining 10%, iterating over an exponentially increasing number of bins. We evaluated the performance here using F1 Score and 0/1 loss, as discussed below. What we found is the performance of the classifier on the Iris data set was not obviously affected by the number of bins chosen, whereas the Glass data set showed a clear improvement around 10 - 50 bins (Figure 2). With this information we decided to bin real values into 2^5 distinct bins.

The second step of preprocessing is dealing with missing attribute values. We approached this problem by first checking to see if the missing attributes were confined to only as small number of examples, or if the missing values were confined to only a small number of attributes. In the case that the missing attributes can be deleted without significantly affecting the data set, the samples or attributes containing missing attributes are removed. We assigned this threshold to be 10% of the samples or attributes. We felt that deleting a small fraction of the dataset would cause a noticeable change. If for some reason the missing attributes were dependent on some other class or attribute, this could skew the accuracy of the classifier, but here we assumed the distribution of missing data was random. If missing attributes could not be removed, they were repopulated randomly to a value $\min_{A_j} \leq x \leq \max_{A_j}$, where A_j is the attribute.

The last step of preprocessing is to create a noisy version of each data set. To do this we select ten percent of the attributes (selecting the next smallest whole number greater than 0), and for each attribute, we randomly shuffle all of the values within that attribute.

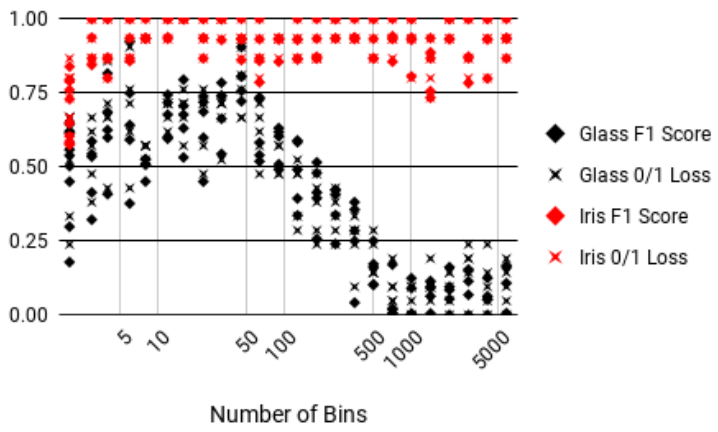

Figure 2: *Bin tuning for the two real-valued data sets. The horizontal axis is displayed on a log scale. The Glass data set was clearly affected by the change in bin size, whereas Iris did not observe the same trend.*

4.2 Learning and Classification

Once the data is preprocessed, the experiment is ready to begin. Here we use k-fold cross validation to evaluate our classifier, i.e., the first step is to divide the given data set into 10 subsets that are randomly populated. Across the the ten trials, each of the subsets is used to test the model, whereas the union of the remaining 9 sets are used to train the model each trial. This method preserves the independence between the test sets, though the training sets still share a large fraction of data points.

Once the class frequency and attribute value frequencies are counted, equations 1 and 2, the classifier, equation 4, classifies each sample in the test set. After the test data is classified, it is evaluated using two different loss functions, 0/1 Loss and F1 score.

4.3 Evaluation

The classified test set consists of examples that now have a ground truth class value and a classified class value chosen by the classifier. The performance of the algorithm on the particular test set is evaluated using 0/1 Loss and F1 score. 0/1 Loss is a measure of classification accuracy and only measures the percent of examples classified correctly. 0/1 score does not weight the consequence of misclassification, but such a measure is not required for this experiment. Furthermore, for each class the number of True Positive, True Negative, False Positive, and False Negative assignments are counted. From these counts the precision and recall are calculated on a per-class basis. The F1 score of each class is the harmonic mean of these two values. The total F1 score for the model is a weighted average of all of the per-class F1 scores.

$$\text{precision} = \frac{TP}{TP + FP}$$

$$\text{recall} = \frac{TP}{TP + FN}$$

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

The F1 and 0/1 Loss scores were averaged across the 10 fold results to produce a single F1 and 0/1 measurement for each data set.

At the end of each trial the average scores of each of the loss functions are generated and stored into a standalone csv file. The goal of this is to allow a programmer to view the overall performance of the algorithm and then tune parameters where necessary and then conduct more trials to see if the tuning resulted in a increase of decrease in overall accuracy.

5. Results and Discussion

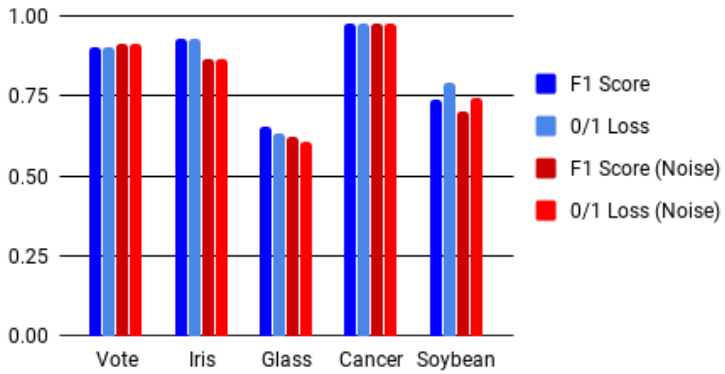

Figure 3: *F1 score and 0/1 loss results for all five data sets. The left two bars in each group represent the original data sets' F1 score and 0/1 loss, and the right two result bars reflect the results for data sets with added noise.*

The experimental results are summarized in Figure 3. The Cancer data set was the only data set that performed with an average accuracy above 95%. Otherwise we found that Vote and Iris both performed well, whereas Glass and Soybean under-performed. Overall, the impact of noise introduced into the data sets was not dramatic and resulted in similar outcomes as the original data sets. The only data set with a clear indication that noise may have affected the classification accuracy was Iris.

In forming our hypotheses we made some intuitive assumptions that the algorithm would yield the best results on the data sets that were already discrete, those that had a large number of attributes per possible class, and those with an even distribution of class occurrences.

The results for soybean did not match our hypothesis. Compared to the other data sets, the Naive Bayes classifier did not perform well. This could be a result of attributes that are not strongly correlated to a specific classification outcome, or it could be that Naive Bayes actually does not necessarily perform better with more attributes.

The Cancer data set had better results than we expected. This might be because, despite the skewed distribution of class values in the data set, all of the test sets also represented this same skew and a classification based on this proportion would in theory be accurate. We believe that this might not be the case if this classifier were tested on new data.

The Glass data resulted in the worst performance of Naive Bayes across all the data sets, as we predicted. It is unclear if it performed poorly for the reasons we stated, but we suspect the ratio of attribute values to possible class values may have played an important role.

While we predicted that the Vote data set may perform well, the results feel inconclusive. While the data set had around a 90% accuracy, we are unsure if this could be evaluated as having performed "well." The same holds true for the Iris data set. We predicted the Iris data set would perform poorly because it had few attributes, but it is unclear if a 92% accuracy can be described as poor performance or not. With respect to all other data sets, Naive Bayes performed relatively well on the Vote data set.

We may not have enough evidence to really conclude anything about the impact of noise. For each data set the F1 and 0/1 scores for the noisy vs non-noisy data sets are similar. In one case (Vote), the noisy data set performed better. The only data set of interest here is Iris, where it seemed that the noisy data set had a clear effect on the results. We suspect this is because though we only selected one attribute in the data set to shuffle, for the Iris data set this was effectively 25% of the attributes, a large fraction. We suspect that it is possible shuffled attributes may not have been strongly correlated to the classification outcome of each example. It's possible some or many of the attributes are not required for accurate classification. We would need to run this experiment many times selecting different attribute columns to shuffle and different percentages of shuffled columns to draw any meaningful conclusions.

Another point of interest is how much the number of bins used to discretize values affected classifier accuracy. In the case of the Glass data set, we found that the number of bins had quite a dramatic effect on the accuracy of the Glass data set. Despite this, tuning the bin count was not sufficient for achieving high accuracy overall. We suspected that a large bin count would provide a resolution granular enough to distinguish minor differences in attribute values. However, it is not clear if this was the reason the bin count greatly affected accuracy.

Both of these discussions lead to the overall question of when Naive Bayes should be used. While we know from the No Free Lunch theorem that no one algorithm will perform best, we have a few takeaways from this experiment that could bear further inspection. Overall we suspect that multivariate data sets with very few possible class values perform best with Naive Bayes. A more extensive analysis would be required to support this idea.

6. Conclusion

In this project, we applied Naive Bayes to five different data sets, as well as a manipulated version of each of these data sets that included noise. As expected from the No Free Lunch theorem, Naive Bayes is not guaranteed to work well on all data sets. Therefore, it would be important to perform an empirical study as well as a critical analysis to further understand the appropriate data sets for which to use Naive Bayes. When comparing the

proposed hypotheses to the experimental results, Naive Bayes performed well on data sets that were not expected to have good results, namely both the voting and the cancer data sets performed exceptionally well. The inverse of the previous statement is true as well, namely that some data sets did not perform as well as expected. This can be seen in the soybean data set results; we expected that the large number of attributes associated with the data set would allow the algorithm to run with great accuracy, however, the algorithm had one of the lowest accuracy rates for this data set. Furthermore, Naive Bayes does not perform with high accuracy across all data sets. This reinforces the idea that there may be a different algorithm to use on the glass data set that would have better performance. Overall the adding of noise to the data sets did no greatly impact the results (except possibly in the iris data set).

References

Rev. T. Bayes. An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions (1683-1775)*, 53:370–418, 1763. URL `http://www.jstor.org/stable/105741`.

Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, Nov 1997.

David Hand and Keming Yu. Idiot's bayes: Not so stupid after all? *International Statistical Review*, 69:385 – 398, 05 2001.

John Sheppard. Csci 447 machine learning: Project 1. Assignment handout, August 2020. Montana State University. Bozeman, MT. PDF.