Project 1

Joshua Aney joshua.aney@student.montana.edu
Owen Cool owen.cool@student.montana.edu
Nicolas Perl nicolas.perl@student.montana.edu

Abstract

Bayes Classifiers have had a large impact on the field of machine learning, but given their remarkable sample complexity, the Naive Bayes classifier emerged as a simpler and remarkably effective classification algorithm. Our experiment explored the Naive Bayes classifier's relative performance when tested on 5 different datasets before and after adding noise to the datasets by shuffling all values within 10% of the features, and its performance across the different datasets. Using 10-fold cross-validation, we found that the model often experienced a remarkably low reduction in precision, accuracy, and recall in response to the addition of noise and performed significantly better on some datasets than others. The results of the experiment reiterate established belief that the Naive Bayes Classifier is a very useful machine learning algorithm given its complexity, but that it can often experience major difficulty in specific circumstances.

1 Introduction

The Naive Bayesian classification algorithm exploits the Bayes rule for its function approximations, which states that in order to learn a function that maps features x to some targets t, training data is used to learn estimates of the likelihood P(x|t) and prior probability P(t). P(x) is the evidence term (scaling factor 0 to 1).

$$P(t|x) = \frac{P(x|t)P(t)}{P(x)}\,. \tag{1}$$

With these estimated probability distributions and Bayes rule, new samples can be classified by the most probable hypothesis given a set of training data.

$$t_{map} = arg\max_{t_j \epsilon T} P(x_1, ..., x_n|t_j)P(t_j))\,. \tag{2}$$

Naive Bayes assumes all attributes describing X are conditionally independent given Y, which solves the problem of otherwise requiring an unrealistic number of training examples.

$$P(x_1, ..., x_n|t_j) = \prod_i P(x_i, t_j)\,. \tag{3}$$

$$t_{map} = arg\max_{t_j \epsilon T} \prod_i P(x_i, t_j)\,. \tag{4}$$

Naive Bayes classification algorithm is very popular due to its simplicity and great performance. In this experiment, the linear Naive Bayes classification algorithm's performance is tested on 5 different (multi-class) classification problems, also deeply investigating cases in which noisy data is provided.

2 Problem Statement

Our experiment was conducted on 5 datasets from the UCI Machine Learning repository, namely the Wisconsin Breast Cancer Database, the Glass Identification Database, the 1984 United States Congressional Voting Records Database, the Iris Plants Database, and the Small Soybean Database. Our research focused on two questions: "How does the addition of noise to these datasets affect the Naive Bayes algorithm's efficacy?" and, "How does the Naive Bayes classifier perform across the five datasets?"

We **hypothesized** that the classifier would perform worse after the addition of noise for all datasets, as the value of the shuffled feature would no longer have any bearing on a given instance's class. We further **hypothesized** that the classifier's performance on the Iris Plants Database would be particularly susceptible to this effect, as it contained the fewest features to shuffle – only four – and therefore rather than exactly 10% of features being shuffled, 25% of features would be shuffled, making a larger share of the data "noise" than any other dataset.

With regard to the latter research question, we **hypothesized** that the classifier's performance would be best on the Breast Cancer Database and have the least variability between folds during cross validation, due to its relatively large number of instances (683 after removal of instances with missing values, compared to the second most, 435, from the Voting Records Database), and its few classes in which to classify instances. Similarly, we **hypothesized** that the Soybean Database would perform worse and experience greater variability in performance in different folds of cross validation due its small size (just 47 instances), and inability to compensate for "outlier instances" that are inconsistent with the rest of the instances appearing in the test set. We also **hypothesized** that the classifier would perform slightly worse on the Glass Database due to the high number of classes (6 present in the dataset, compared to the second most, 4, in the Soybean Database) to choose from, the limited instances of some of those classes (just 9 of the 214 instances were tableware glass), and its continuous data. Lastly, we **hypothesized** that the classifier trained on the Voting Records Database would perform relatively poorly because of its nature as political data and the high number of abstentions present in the data.

3 Experimental Approach and Program Design

We approached the problem stated in section one with a very sequential manner. Our first task was to create a hypothesis based on the data we were given and our prior knowledge on the task. After this, we spent time on creating a design document that would outline the steps that would be required to test our hypothesis. This document included a list of requirements, a basic structure of our desired code, a flow of how we were going to use that code, and a testing strategy. This would help discretize our goals and would allow us to have a strategy to attack the problem. This was then followed by understanding and

coding an implementation of the Naive Bayes algorithm that was provided in the project description.

This algorithm required the following steps:

1. For each class in the training set, calculate

$$Q(C = c_i) = \frac{\#\{\mathbf{x} \in c_i\}}{N}. \tag{5}$$

2. Separate the data into their respective classes.
3. For each attribute A_j in the class-specific training set, calculate

$$F(A_j = a_k, C = c_i) = \frac{\#\{(\mathbf{x}_{A_j} = a_k) \wedge (\mathbf{x} \in c_i)\} + 1}{N_{c_i} + d}. \tag{6}$$

4. After these are complete for the entire training set, to classify, do the following for each class. Calculate only for the attribute values a_k that exist in the example.

$$C(x) = Q(C = c_i) \times \prod_{j=1}^{d} F(A_j = a_k, C = c_i). \tag{7}$$

5. Then return

$$class(x) = \arg\max_{c_i \in \mathcal{C}} C(x). \tag{8}$$

Once we had coded the algorithm, we had to do some data preprocessing to be able to implement this on the five datasets from the UCI machine learning repository. We decided that we would throw out any rows that had missing continuous data points.

Once everything was up and running, we tested our hypothesis regarding adding noise to the data by performing 10-fold cross-validation. We used the following metrics to evaluate our algorithm.

1. Precision
2. Recall
3. Raw Accuracy

We then compared these metrics on the the data before and after the noise had been added.

For our program design we went for a relatively basic approach. We created a separate class for each of the five datasets so that we could complete the specific preprocessing needed for each set. We then had a class that represented the Naive Bayes model. This class contained all of the class probabilities as well as the specific attribute value probabilities. This class also contained functions that could be used to set all the probabilities and a function used to evaluate future data points. We also had a set of helper functions that would help us implement the 10-fold cross-validation and our performance metrics.

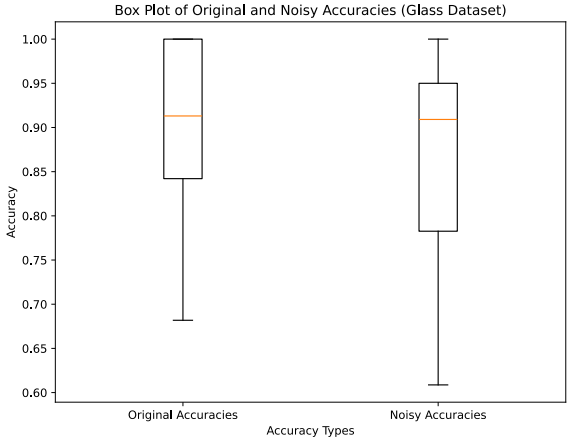
4 Experiment Results

Throughout the experiment, we received relatively consistent results regarding accuracy. Even when we added noise, the metrics that we used to evaluate our model did not indicate any significantly worse performance.

There were a few exceptions that we did have to address. Originally, the glass dataset did not have discrete values. We ended up having to bin all of the features to get precision, recall, and accuracy values comparable to the other datasets.

(a) Confusion matrix for raw glass data

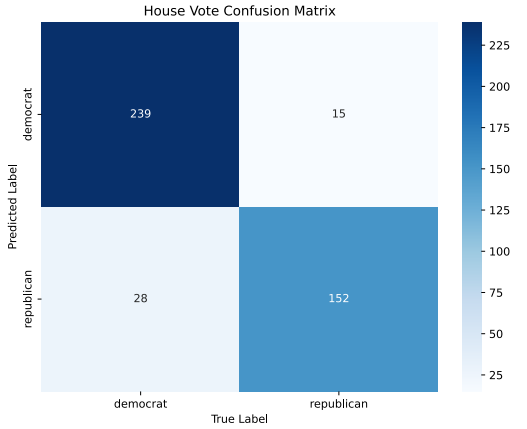
(b) Confusion matrix for noisy glass data



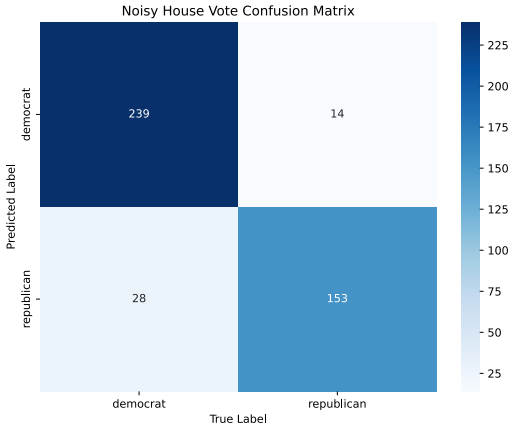
(c) Box and whisker plot for glass data

Figure 1: Results of the glass data classification

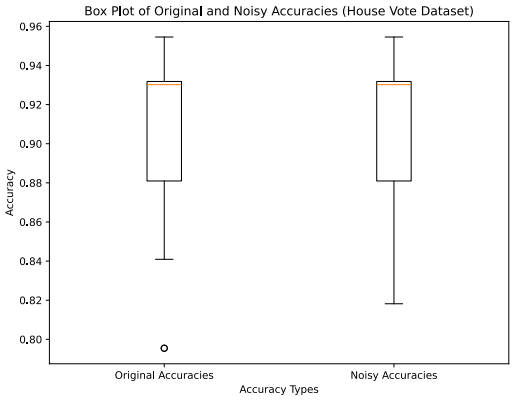
The House Vote data also performed as expected while using our Naive Bayes classifier. There is an outlier than can sometimes affect the recall, precision and accuracy values, but it is relatively insignificant.



(a) Confusion matrix for raw house vote data



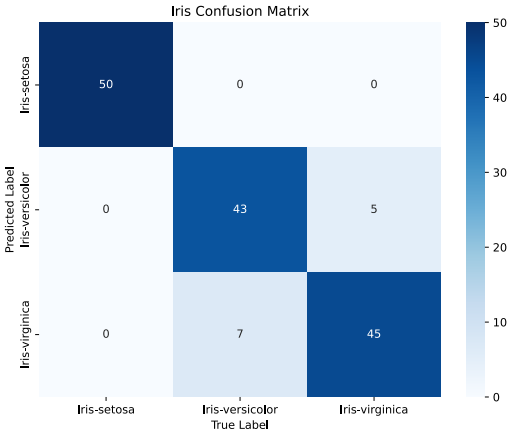
(b) Confusion matrix for noisy house vote data



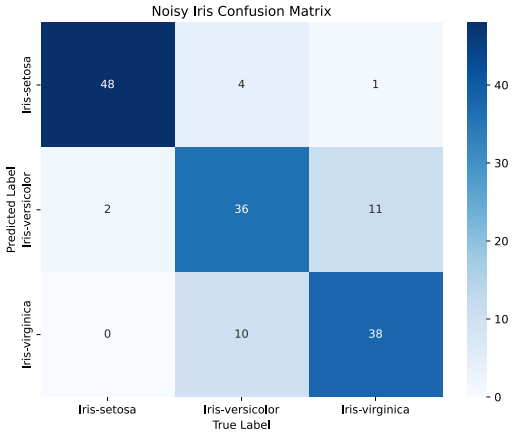
(c) Box and whisker plot for house vote data

Figure 2: Results of the house vote data classification

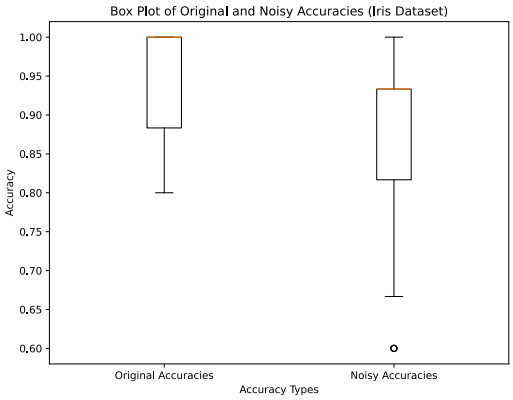
The Naive Bayes Classifier handled the Iris, Soy, and Breast Cancer datasets with little to no issue. There was no data preprocessing needed for the Iris and Soy datasets and only minor handling of missing data points needed in the Breast Cancer set. All of the metrics we used showed that the model was performing exceptionally well on the data it was given.



(a) Confusion matrix for raw iris data

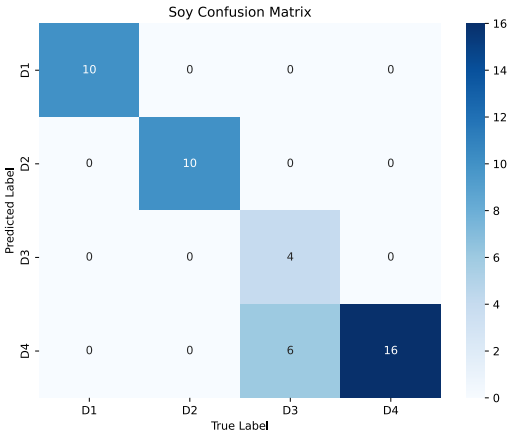


(b) Confusion matrix for noisy iris data

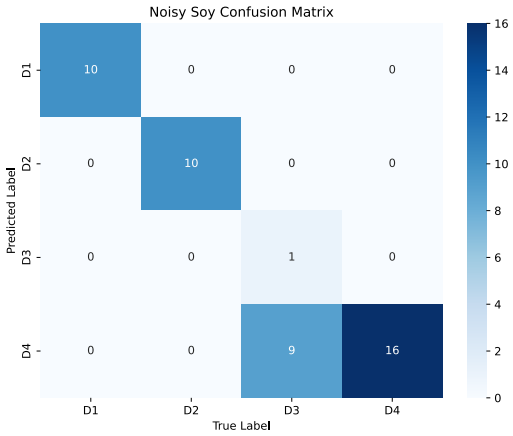


(c) Box and whisker plot for iris data

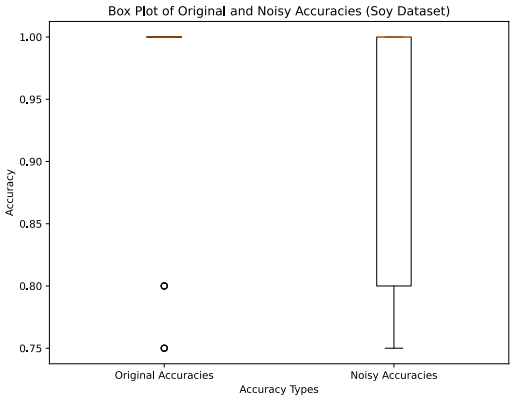
Figure 3: Results of the iris data classification



(a) Confusion matrix for raw soy data

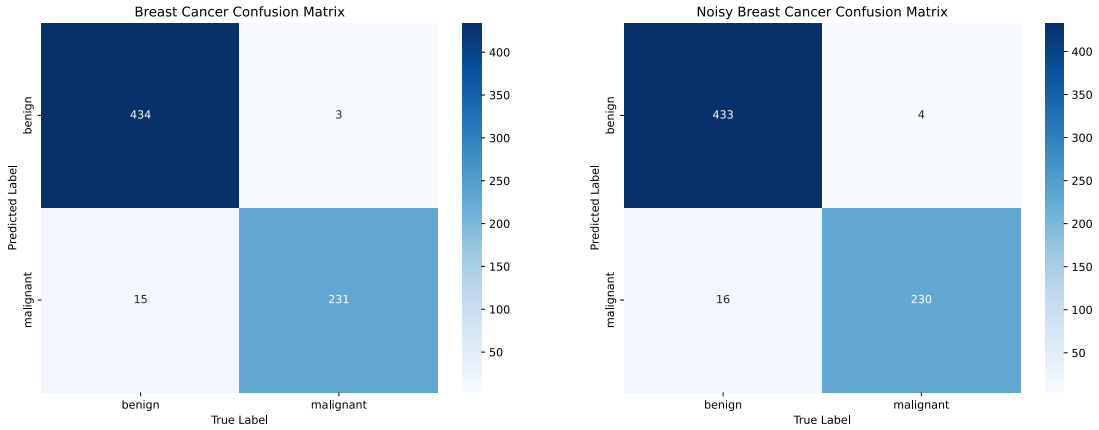


(b) Confusion matrix for noisy soy data

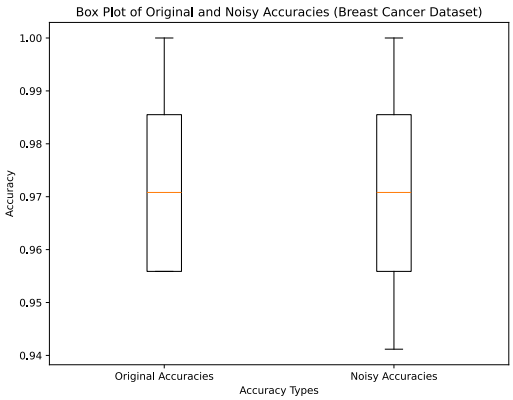


(c) Box and whisker plot for soy data

Figure 4: Results of the soy data classification



(a) Confusion matrix for raw breast cancer data (b) Confusion matrix for noisy breast cancer data



(c) Box and whisker plot for breast cancer data

Figure 5: Results of the breast cancer data classification

Once we had our classifier performing well on the datasets we were given we inserted artificial noise into the data by shuffling a feature across the dataset.

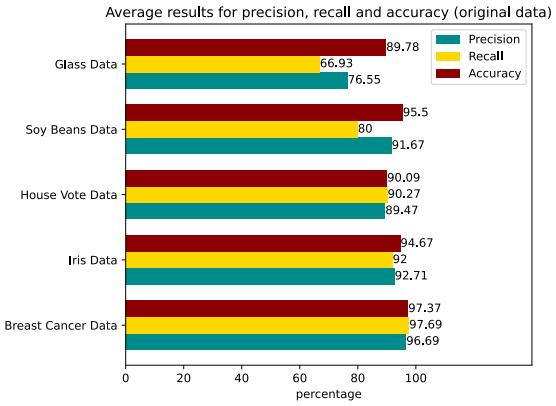


Figure 6: This is a bar graph to represent the different average values of the metrics collected in the experiment prior to the implementation of the artificial noise.

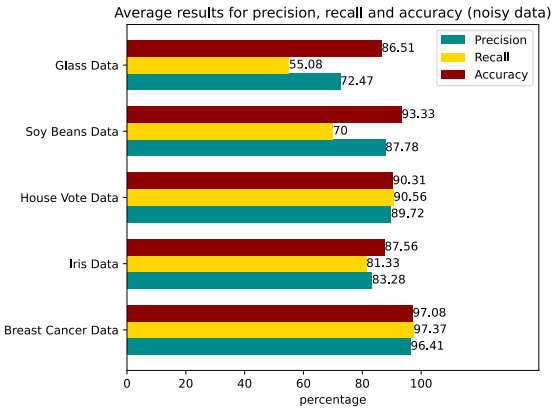


Figure 7: This is a bar graph to represent the different average values of the metrics collected in the experiment after the artificial noise was implemented.

5 Discussion

As shown in figures 1-5, our results did not demonstrate a significant difference in the classifier's performance following the addition of noise into the data. While figure 3 seems to demonstrate some loss in accuracy following the addition of noise for the Iris Database, the mean accuracy for this noisy database remains well within the margin of error for the accuracy of the original database. While these results do not definitively prove our hypothesis, they certainly do not disprove it. The trend – if present – seems to be that adding noise reduces the efficacy of the classifier, and theory supports this. Rather, it is likely that our Databases are small enough that we experience wide variability in efficacy on different folds of our cross-validations, thus clouding any effects that we may see the noise having on the model. Notably, though, the largest loss in average accuracy was in the Iris Database, as hypothesized. This may be a slight indication that making noise over a larger percentage of the database's features leads to a larger loss in efficacy.

Examining figures 6 and 7, we can see that the classifier for the Breast Cancer Database did indeed perform better than classifiers for other databases on all three performance metrics (precision, recall, and accuracy) and in both the noisy and the original data, consistent with our hypothesis. However, this difference is relatively insignificant, with its performance metrics only being a couple percentage points higher than that of the iris or soybean data. This is a likely indication that while more data does help, the Naive Bayes Classifier simply does not need a significant amount of data to effectively learn most of a classification problem. This tentatively reinforces its utility as an efficient algorithm for practical purposes.

Meanwhile, looking at the performance of the classifier on the Glass Database (again in figures 6 and 7), we can see that while the accuracy is not far off that of the other databases, the recall experiences a significant drop. As stated in our hypothesis, this is likely due to

the relatively large number of classes in that database. In fact, we may get some insight into how exactly this hinders performance by examining the wide disparity between the performance metrics for the Soybean Database. While the false negatives expected of a classifier with many classes are mediated in accuracy as it also factors in true negatives, true negatives do not similarly factor into recall, which may cause the discrepancy we see in our results. However, we did not see the expected drop in efficacy for the accuracy of the glass database due to its continuous data, indicating that binning was an effective strategy for this database. It remains to be determined if this strategy consistently produces such results.

Looking back at figure 4c, we can see that the classifier on the Soybean Database exhibited much more significant outliers in accuracy than the other databases. This is consistent with our hypothesis, and the relatively high, narrow distribution of accuracy on other folds compared to these outliers further reinforces this theory. Essentially, we can see that when a database has limited instances, test sets having just a few outliers or otherwise incorrect predictions can radically change our accuracy. However, this does not necessarily indicate the model's poor performance – only that the test sets were quite small.

Lastly, looking at the Voting Records Database, we did not see a significant decline in performance relative to the other databases as expected. This may be an indication of Naive Bayes classifiers' versatility, performing well on widely varied databases, though it is also likely due to a faulty hypothesis, as abstentions may be more useful as data than previously thought. It is difficult to determine the exact situation at hand without further research.

6 Conclusion

Our experiment was consistent with past research and theory, demonstrating Naive Bayes classifiers as a versatile and remarkably useful machine learning algorithm given their relative simplicity, but also showing their shortcomings in certain domains. However, the size of our experiment was quite limited, and more robust experiments with larger and more varied datasets on which to test would provide more insight into their workings. Further, our experiment was quite broad in scope, and while there was some indication of the number classes' impact on recall or of test set size's impact on result variability, experiments more tailored to these questions would need to be carried out before drawing definitive conclusions.

7 Appendix

Group member complications
On the day the Design Document was due, Nicolas' team member decided to drop out, which started a whole series of unexpected, awkward situations. On September 1st, a new group alignment with Ms.Fazal was announced, which was followed up by a first group meeting discussing the situation and future schedule. Later on the 3rd of September, another team member, Mr.Vernooy, joined the group. Towards the end of the week, after looking into Ms. Fazal's coding and first program extensions by Nicolas, several uncomfortable emails were exchanged between the other members. Mr.Vernooy decided to drop out, leaving the others again in an awkward situation. With the due date approaching, meetings with Dr.Sheppard were held and solutions were found. Ms.Fazal continued on her own, while Nicolas joined this group on the 9th of September, leaving only a few days for the final group submission.

While eager to contribute and participate in the group work, I ask for your indulgence as the time constraints did not allow for more group work participation from my part.

Group work distribution
Paper — We mainly worked together on the paper. We each started by writing specific sections, then we proofread each other's work. We also worked together to get all the figures in a format that we liked.

Code — The code was done mainly by Owen Cool and Josh Aney. It was a combination effort throughout. Owen Cool wrote the stratification methods and many of the dataset classes, and Josh Aney wrote the Naive Bayes model and the methods that came along with evaluating it. Nicolas Perl wrote the code for data and results visualization.

Design Document — Josh Aney and Owen Cool worked together to do the design document. It was a combined effort to write each section and proofread it.

Video Essay — It was a group effort to plan the video essay, and then Josh Aney recorded the video.