CSCI 447: Native Bayes Analysis

Shane Costello SHANECOSTELLO@STUDENT.MONTANA.EDU
School of Computing
Montana State University
Bozeman, MT, USA

Hayden Perusich HAYDENPUERSICH@STUDENT.MONTANA.EDU
School of Computing
Montana State University
Bozeman, MT, USA

Abstract

Our paper investigates our simple Naive Bayes Classifier's ability to precisely and accurately classify data from 2 separate datasets. Our datasets are called Clean and Noisy. We got our clean dataset by making continuous values discrete and inputting absent values back into the dataset. The Noisy dataset is a mutated version of the clean dataset with 10% noise added to each feature. Each dataset had 5 subsets varying in number of rows, features, and classes. Our model first creates a probability table for each subset and uses those probability tables to classify. We used 10 fold validation to test each subsets for clean and noisy. This allowed us to get accuracy and F1 values for each and to produce a confusion matrix helping us analyze our results. We found that having a larger number of features and rows helped our model classify accurately more consistently. We think this is because by having more features and rows it gives more opportunity for each class to be strongly correlated with a feature. We also found that having more classes decreases our models ability to accurately more consistently. We believe this is because having more classes decreases the strength of that feature class correlation which would lead to decreased accuracy.

Keywords: Naive Bayes Classifier, noisy data, ten-fold cross validation

1 Introduction

This paper explores the functionality and effectiveness of the Naive Bayes Classifier by comparing its performance on two datasets. Dataset A (hereafter referred to as Clean Data) consists of five subsets sourced from the UCI Machine Learning repository. These subsets provide metrics for training and classification, with each containing a varying number of unique features and classes. The size (number of data points) of each subset is variable as well. Dataset B (hereafter referred to as Noisy Data) is derived from Clean Data, but with 10% added noise. The paper will compare the algorithm's ability to classify data points from Clean Data to its performance on Noisy Data. The goal of this experiment is to demonstrate how the Naive Bayes Classifier responds to the different subsets in our Clean Data, as well as its response to data with added noise. We hypothesize that subsets with a greater number of data points will be classified more accurately than those with fewer data points. We suspect this because a greater subset size will provide a larger training set, creating a more comprehensive understanding of the classes and unique features. Additionally, we hypothesize the algorithm will classify data points from Clean Data more accurately than from Noisy Data. We predict the added noise will loosen the correlation between unique feature values and their class, resulting in fewer accurate classifications.

2 Experimental Design

Our experiment employed the following procedure. We first process the data files that will act as our subsets. We then create Noisy Data by modifying the Clean Data set. Next, we set up our ten-fold cross-validation and perform iterations of training and testing. Once the iterations are complete, we are returned a confusion matrix for each subset in each of our sets.

2.1 Preprocess Data

Data retrieved from the UCI Machine Learning repository required us to take preprocessing measurements before it could be used for the experiment. We had to handle cases of missing data, as well as make continuous values discrete.

2.1.1 Data Imputation

Cases of absent data were denoted by question marks (?) in all datasets except house-votes-84.data, where question marks held real value. For the data imputation process, we calculated the mean value of the unique feature where there was missing data. We then input this value in place of any absent data within said feature. We chose to use the mean because it is well fit to the data and will not create an outlier in later calculations.

2.1.2 Data Discretization

For the Naive Bayes Classifier to work effectively, it requires discrete features. However, all but one dataset (house-votes-84.data) contain continuous features. The set of possible values for a continuous feature is all real numbers. This makes classification challenging, as we would need an exact match on a continuous value for proper classification. The process of turning continuous values into discrete values is called binning. We chose to bin our values based on the frequency distribution within each unique feature.

For each unique feature, we divided the data into four quartiles and binned the values accordingly. Figure 1 depicts the binning process for the Iris dataset. On the right side of each chart, we can see the continuous data values, separated by class. On the left-hand side, the box and whisker plot shows the data distribution, noting how it will be binned into quartiles. After binning, the set of possible values for our discrete feature is [Q1, Q2, Q3, Q4]. We repeated this process for each dataset with continuous features.

We chose to bin by quartile because it is a measure of frequency distribution within each unique feature. This allowed us to create bins with an even spread of data, creating more symmetric occurrences of possible values. Alternatively, we could have created bins of equal size (0 to 1, 1 to 2, etc.). This would likely result in a disproportionate occurrence of a particular bin, potentially skewing the model's training.

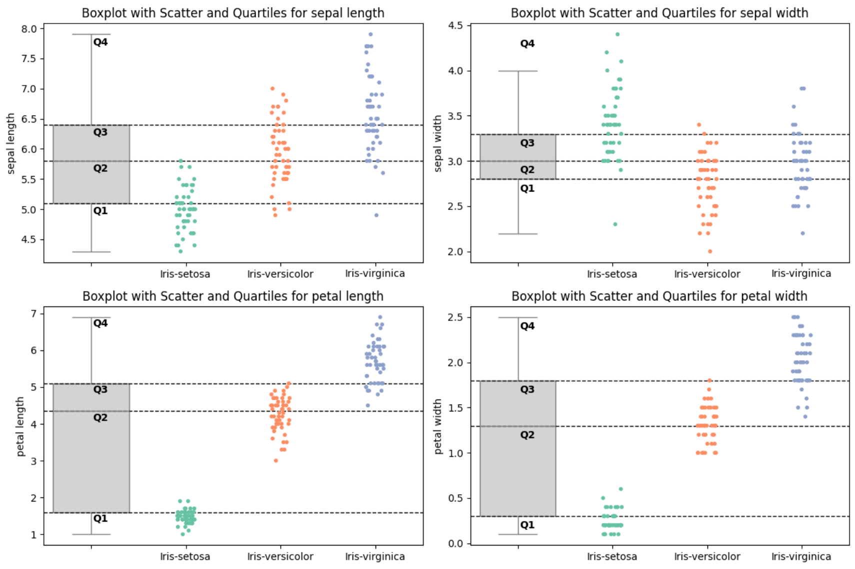


Figure 1: The binning process for each unique feature in the Iris dataset

2.2 Create Noisy Data

We took Clean Data and introduced a 10% noise to create Noisy Data. For each subset in Clean Data, we iterated over the set of unique features. For each feature, we randomly selected 10% of the values and shuffled them around, hence introducing noise. We added noise to the whole data set, which will become both training and testing sets. We chose to do this so that there is noise in the creation of the probability table and the classification process.

2.3 Ten-Fold Cross-Validation

For each of our datasets, Clean Data and Noisy Data, we performed ten-fold cross-validation. The first step in this process is to shuffle the data points inside each subset to remove any possible ordering. To create the folds, we divided the data points of each subset into ten even groupings (distributing any remainders through the folds). This allowed us, for each subset, to train our model on 90% of the data (9 folds) and test on the remaining 10% (1 fold). To perform the cross-validation, we rotated which folds belong to our training and test sets. We performed ten iterations so that each fold acted acts as the test set exactly once and is otherwise a member of the training set.

Ten-fold cross-validation allows us to remove any bias that might exist inside a particular division of our data. If we only ever trained and tested on the same folds, we would not know that a bias exists. The purpose of this examination is not to determine the presence of a bias but to eliminate it by shuffling our data in a structured manner.

2.4 Training

Given 90% of the data from a particular subset, we train our model by performing the following actions. We first determine the probability of each class by dividing the number of occurrences of a given class by the total number of data points. These values are stored in a dictionary where each key is a class name and the value is its probability. Then, for every class, we calculate the probability that each unique feature possesses each of the possible values (Q1, Q2, Q3, and Q4). This is done by counting the number of instances where a unique feature of a given class possesses one of the possible values and dividing it by the number of instances of that class. For each unique feature in each class, we iterate over the set of possible values to calculate these probabilities. These values are stored in a multi-dimensional dictionary called the probability table. The first dimension of keys represents the classes, the second dimension of keys represents the unique features, and the third dimension has key-value pairs of the possible values and their respective probability. We took an eager approach to creating our probability table in the training process because we predicted it would cause less error in the testing process.

There is the possibility that there are no cases where a particular unique feature possesses one of the possible values. This would result in a probability of zero and would create an issue in the classification process. To avoid this we employed the 'add one' method. In this method, we add a count of one to all numerators and four (the number of possible values) to all denominator. Since we do this to every probability in our probability table, it does not change the validity of the probabilities, it simply scales them by a given factor.

2.5 Testing

The testing process is also known as classification. For each data point in our test set, we calculate the probability of it belonging to each class. To calculate this, we first retrieve the probability of each class from our training data. Next, we iterate over the set of unique features in the current data point. For the value of each feature, we look up the probability that the given feature has that value in our probability table. We do this for each class. Finally, we multiply the class-dependent probabilities together and are returned a dictionary where the keys are the class names and the values are the probability that the data point is a member of that class. To classify, we select the class with the highest probability. This is called our predicted value.

For each test we perform, we store the predicted value in an array and the actual value in an array. These two arrays have a one-to-one mapping where the same indices represent the predicted value and actual value for the same data point. These two arrays exist for every subset in both data sets. These arrays are added to during each iteration of ten-fold cross-validation. When the iterations are complete, we use the predicted and actual value arrays to create a confusion matrix. This is an $N\ x\ N$ matrix where N is the number of classes in a subset.

2.6 Metrics of Comparison

When comparing the algorithm's performance on (1) the varying subsets and (2) regular versus noisy data we will be looking at their F1-score and accuracy rating. We will be analyzing these metrics on both a class-by-class basis (micro ratings) and by an entire subset (macro ratings). We chose these specific metrics because they provide a comprehensive rating of the algorithm's performance. The F1-score offers a balance between precision and recall scores. Accuracy, on the other hand, gives a straightforward measure of the algorithm's overall success in correctly classifying data points. By analyzing both micro and macro ratings, we can independently examine how one class may have affected the subset's overall score. These metrics are derived from the confusion matrix generated in our ten-fold cross-validation.

3 Results

We found that our Native Bayes model did well in general with both the clean and noisy subsets. Our accuracy was $0.799 \leq x \leq 1.00$ and our F1 score, which is a harmonic mean between precision and recall, was between $0.6497 \leq x \leq 1.00$. Adding noise to our data negatively affects our model's ability to accurately predict each class. This is shown with a decrease in both the accuracy and F1 scores across the datasets.

Adding more classes also negatively affected our model's performance when the dataset did not have enough rows and features to compensate. This is shown most evidently in the glass dataset where we have 7 classes but only 10 features and 214 rows.

Having a large number of features also has a significant impact on our results. Having more features increases our model's F1 and accuracy scores. The same can be said for the number of rows where when you increase the amount of data you feed the model it becomes more consistently accurate.

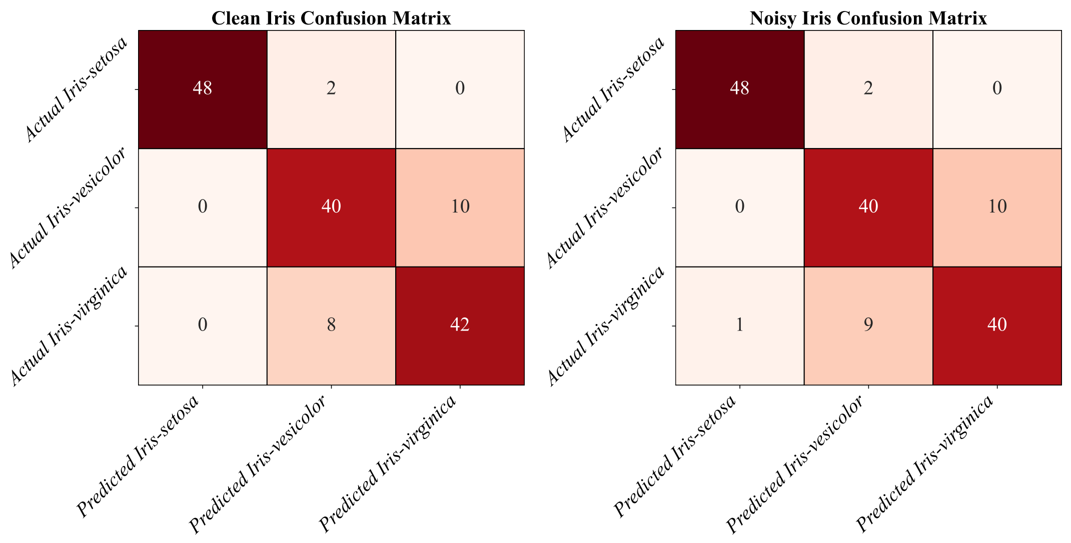
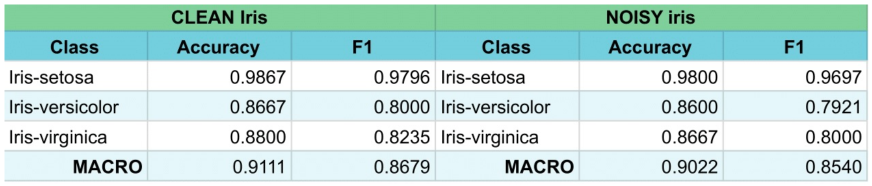


Figure 2: Iris confusion Matrix HeatMap



CLEAN Iris			NOISY iris		
Class	Accuracy	F1	Class	Accuracy	F1
Iris-setosa	0.9867	0.9796	Iris-setosa	0.9800	0.9697
Iris-versicolor	0.8667	0.8000	Iris-versicolor	0.8600	0.7921
Iris-virginica	0.8800	0.8235	Iris-virginica	0.8667	0.8000
MACRO	0.9111	0.8679	**MACRO**	0.9022	0.8540

Figure 3: Enter Caption

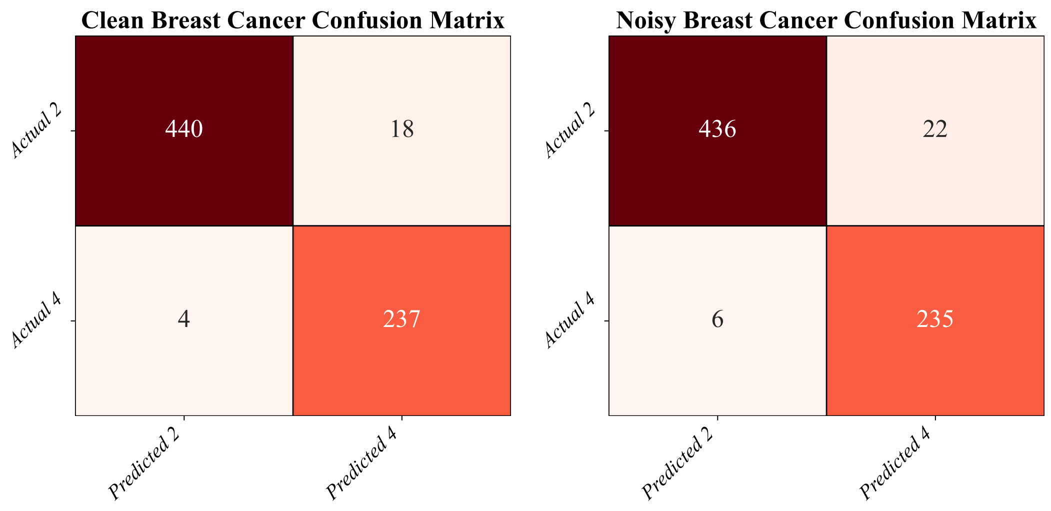
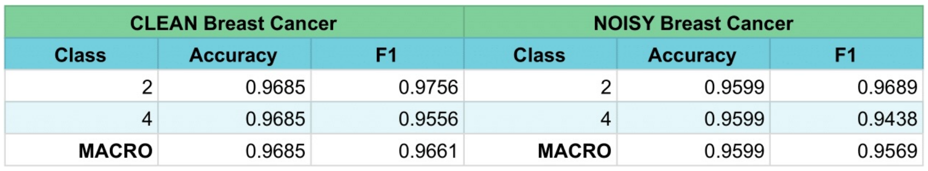


Figure 4: Breast Cancer Confusion Matrix HeatMap



CLEAN Breast Cancer			NOISY Breast Cancer		
Class	Accuracy	F1	Class	Accuracy	F1
2	0.9685	0.9756	2	0.9599	0.9689
4	0.9685	0.9556	4	0.9599	0.9438
MACRO	0.9685	0.9661	**MACRO**	0.9599	0.9569

Figure 5: Breast Cancer Accuracy / F1 Table

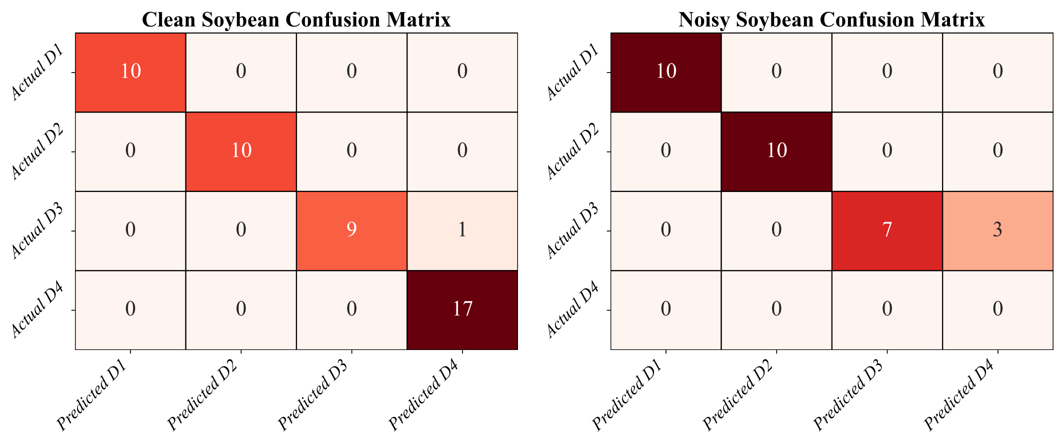
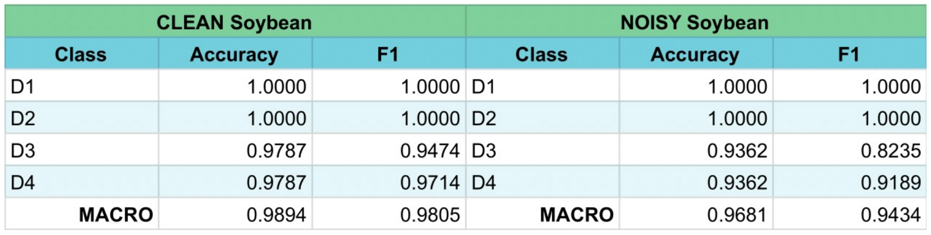


Figure 6: Soybean Confusion Matrix Heatmap



CLEAN Soybean			NOISY Soybean		
Class	**Accuracy**	**F1**	**Class**	**Accuracy**	**F1**
D1	1.0000	1.0000	D1	1.0000	1.0000
D2	1.0000	1.0000	D2	1.0000	1.0000
D3	0.9787	0.9474	D3	0.9362	0.8235
D4	0.9787	0.9714	D4	0.9362	0.9189
MACRO	0.9894	0.9805	**MACRO**	0.9681	0.9434

Figure 7: Soybean Accuracy / F1 Table

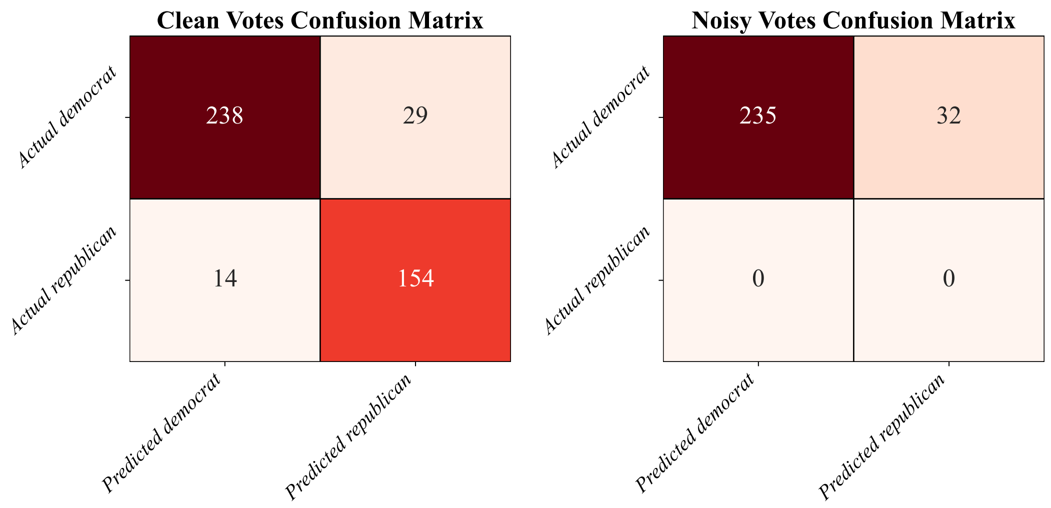


Figure 8: Votes Confusion Matrix HeatMap

CLEAN Vote			NOISY Vote		
Class	Accuracy	F1	Class	Accuracy	F1
democrat	0.9011	0.9171	democrat	0.8943	0.9109
republican	0.9011	0.8775	republican	0.8943	0.8701
MACRO	0.9011	0.8985	**MACRO**	0.8943	0.8921

Figure 9: Votes Accuracy / F1 Table

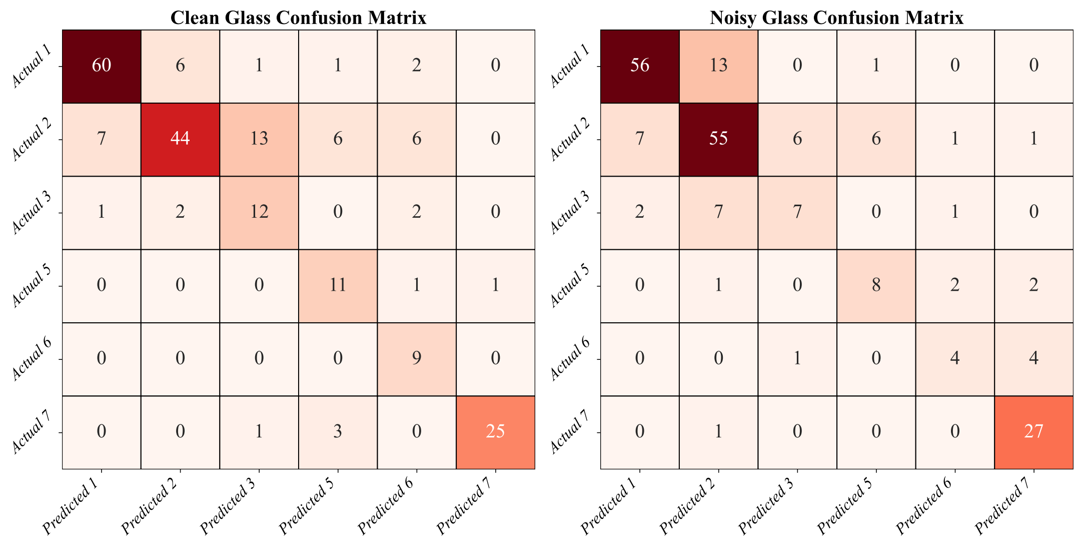


Figure 10: Glass Confusion Matrix HeatMap

CLEAN Glass			NOISY Glass		
Class	Accuracy	F1	Class	Accuracy	F1
1	0.9112	0.8613	1	0.8925	0.8296
2	0.8505	0.7895	2	0.7991	0.7190
3	0.9533	0.6875	3	0.9159	0.4375
5	0.9626	0.7333	5	0.9439	0.5714
6	0.9813	0.8000	6	0.9579	0.4706
7	0.9766	0.9123	7	0.9579	0.8571
MACRO	0.9393	0.8011	**MACRO**	0.9112	0.6497

Figure 11: Glass Accuracy / F1 Table

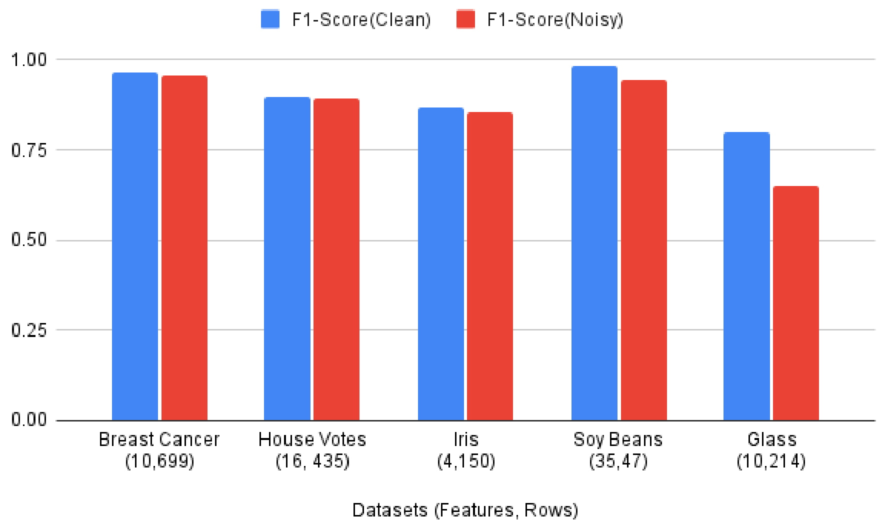


Figure 12: Bar Graph showing the F1 scores for each dataset and the features and rows of that dataset

4 Discussion

The number of features, rows, and classes all played a part in the performance of our dataset. On average the smaller number of classes paired with many features and rows allowed our model to perform the best. This can be seen with the breast cancer and soybean datasets which contained the most rows and features. They both had F1 scores greater than 0.94 and an accuracy greater than 0.96. Figure 12.

We believe a larger number of classes decreases performance because when our classifier has to accurately classify for an increased number of classes its ability to make predictions with high certainty accurately goes down just because there are more classes to choose from. This greater number of choices might decrease the strength of that feature class correlation which would lead to decreased accuracy. This is validated by the Glass dataset which did the worst by far. This dataset had 7 classes which is the largest number of classes we had to accurately classify. It only has 10 features and 214 rows to compensate. Fig 10.

As seen from Figure 2 - 11, not all classes were created equal when it came to classification. Some classes did well with a majority of true positives while others like class 6 for the glass dataset only got 9/20 correct. Figure 10. We believe this is because as shown in Figure 1 certain features strongly correlate with certain classes while other features don't. We believe that specific classes in our datasets don't contain fewer features with a stronger correlation than some of the other classes. Our decision matrices (Figure 3-7) are evidence for this as you can see certain classes among all datasets have significantly more false positives than others. This problem is amplified when noise is added and the classes with weakly correlated features become weaker.

Adding noise to our dataset decreased both our accuracy and F1 scores. This makes sense as adding 10% noise to each dataset would decrease our model's certainty when classifying as there would be fewer unique groups of features for each unique class. Despite this Naive Bayes seemed to perform well when noise was added to datasets with a smaller number of classes. This means that Naive Bayes can classify with enough certainty that adding 10% noise is not enough to significantly decrease its performance. Fig 2.

This is not the case on datasets with a larger number of classes. (Think 6+ classes). When noise is added it significantly decreases the model's ability to predict. We think this is because adding noise to these datasets, more effectively jumbles the data even more as there are more classes associated with each feature.

5 Summary

We can conclude that our original hypothesis *"that subsets with a greater number of data points will be classified more accurately than those with fewer data points and the algorithm will classify data points from Clean Data more accurately than from Noisy Data."* has evidence to support it. We found that as the number of features and rows increased so did the accuracy and F1 scores. We also found that although we did not mention it in our hypothesis the number of classes in each subset also had a large effect on the effectiveness of our model. Adding more classes greatly decreased our model's ability to correctly classify them. Especially when 10% noise was added.

Appendix A.

In this appendix we will explore the division of labor and contributions from each team member.

Paper

Abstract: Hayden Perusich
Section 1: Shane Costello
Section 2: Shane Costello
Section 3: Hayden Perusich
Section 4: Hayden Perusich
Section 5: Hayden Perusich

Video

Script: Shane Costello
Recording: Shane Costello
Editing: Hayden Perusich

Code

naive_bayes.py: Hayden Perusich
ten_fold.py: Hayden Peruscih
confusion_matrix.py: Shane Costello
preprocess.py: Shane Costello
preprocess_votes.py: Shane Costello
demo.py: Shane Costello