CSCI 447: Machine Learning Project 1

Nitasha Fazal NITASHA.FAZAL@MONTANA.EDU
School of Computing
Montana State University
Bozeman, MT 59715, USA

1. Problem Statement

The main goal of Project 1 is to implement and test the Naïve Bayes algorithm for the classification of 5 datasets. Naïve Bayes is a supervised machine learning model that classifies objects based on probability. It is known for its efficiency and simplicity. The algorithm is called "naïve" because it assumes that the features used to predict the class are conditionally independent, given the class label.
Naïve Bayes is grounded in Bayes' theorem, which describes the probability of an event based on prior knowledge of conditions related to the event. Mathematically, Bayes' theorem is expressed as:

$$P(A|B) = \frac{P(B|A).P(A)}{P(B)} \tag{1}$$

where $P(A|B)$ is the posterior probability, that is the probability of A given B occurred.
$B(B|A)$ is the probability of B given A has occurred.
P(A) is the probability of event A and P(B) is the probability of event B
GeeksforGeeks (2024)
Given the feature data for each class, the model learns the conditional probabilities of the features for each class. It then uses these probabilities to predict the class of new, unseen data based on the attributes of the test data.
We are given 5 datasets in this project to train and test our naïve Bayes algorithm on, these are iris, cancer, voting, and soybean and glass. I aimed to implement a generic model that can be trained and tested on each dataset for classification.

1.1 Hypothesis

In this project, we have the same hypothesis for each of the five datasets.
Null Hypothesis: Noise in the dataset will have no effect on the model's classification prediction
Alternate Hypothesis: Noise in the dataset will affect the model's classification prediction.
We have used evaluation measures, explained in the next section, to accept or reject the null hypothesis.

2. Experimental Approach and Program Design

In this section, I will explain the algorithm steps, the formula used to train and test the Naive Bayes model, and the experimental approach used to evaluate the model.

2.1 Data Pre-Process

After loading the dataset into the system, our first step is to preprocess the data. It is crucial for any machine learning project, as the model's performance and sometimes even its functionality depend on it. For this step, I have used the following functions:

- Shuffling: This step was not strictly necessary, but it improved the model's performance. Since the datasets are organized by class, I shuffled the rows before processing to ensure that each fold in the cross-validation contains a mix of all classes.

- discretization: This function is used for two datasets, namely Iris and Glass, both of which contain continuous values. To convert these continuous values into discrete categories, I used the Python functions cut and qcut. The cut function divides the continuous values into fixed bins, and we can label these bins as ”small,” ”medium,” ”large,” etc. On the other hand, qcut divides the values based on quantiles.
  For the Glass dataset, qcut with 10 bins worked well, but for the Iris dataset, discretizing the values resulted in a drop in the algorithm's accuracy.

- Imputation: I automated the process of handling missing values by identifying any '?' or null values in the dataset. If a dataset has more than one missing value, they are replaced by the mode of the respective column for the 'Voting' dataset, as its values are binary categorical ('y' or 'n'). For all other datasets, missing values are replaced by the median. I opted for the median over the mean to avoid the influence of outliers on the imputed values.

- 10 Folds creation: This function divides the dataset into 10 equal parts (folds). In each iteration, 9 folds are used for training, and 1 fold is used for testing. This is repeated 10 times, so each fold serves as the test set once, while the remaining 9 are used for training.

2.2 Naive Bayes Implementation

As mentioned earlier, this project involves implementing the ”plus-one,” ”additive,” or ”Laplace” smoothing version of the Naive Bayes algorithm. The algorithm is then trained and tested on five datasets. Below are the details of my implementation.

- Train Function: In the train function, I implemented the two formulas given in the project description,i.e
  $$Q(C = c_i) = \frac{\#(x \in c_i)}{N} \tag{2}$$
  for this step, I calculated the prior probability for each class by dividing the number of examples in each class by the total number of data points in the dataset.

After that, I separated the data into their respective classes. Then, I calculated the following function for each attribute of each class.

$$F(A_j = a_k, C = c_i) = \frac{\#(x_{A_j} = a_k) \wedge (x \in c_i) + 1}{N_{c_i} + d} \tag{3}$$

That is, for each attribute value, divide the number of examples that match that attribute value (plus one) by the total number of examples in the class. Once we store these attribute values in a dictionary for each class, the training part is complete.

- Test Function: For testing the model, I implemented the formula given in the project description, with a slight modification. Below is the formula:

$$C(x) = Q(C = c_i) \times \prod_{j=1}^{d} F(A_j = a_k, C = c_i) \tag{4}$$

  That is, for each class, I calculated that class probability by adding the prior probability with the attribute count for that class with the attributes of the test data. If we do not have the same attribute in the training set, then either we make the dataset discrete, or I use the below function.

$$C(x) = log\frac{1}{k+1} \tag{5}$$

  Raschka (2014)
  Once we have the probabilities for each class, I select the class with the maximum probability. This gives us the prediction for the class of the test dataset.

- Evaluate Results: After obtaining the test results, I created the confusion matrix and calculated TP, TN, FP, and FN for each class and each fold. I then computed all 5 loss functions for each dataset, focusing on the relevant ones based on the nature of the data. For averaging the results of each fold, I used macro averaging for each dataset. This involves first calculating the metric for each class in each fold, then averaging these metrics across all classes for each fold. Finally, I averaged these results across all folds.
  All the processes mentioned above are repeated 10 times. I implemented 10-fold cross-validation, passing a different fold to the test function each time and the remaining 9 folds to the train function.

3. Results

Using the above experimental approach, the confusion matrix below represents the aggregated results for each dataset, obtained by combining the predictions from all the folds. This confusion matrix is not an average of the individual folds but is generated by aggregating the results after each fold.
In contrast, the accuracy, zero-one loss, and F1-score were first averaged in the folds using macro-averaging, calculated based on the following formula:

$$Acc = \frac{Acc_a + Acc_b + Acc_c}{3} \tag{6}$$

Exchange (2016)
These macro-averaged values are then averaged across all folds.

3.1 Iris DataSet

Confusion Matrix:

Actual ↓ Predicted →	Iris Setosa	Iris Versicolour	Iris Virginica
Iris Setosa	50	0	0
Iris Versicolour	0	46	4
Iris Virginica	0	4	46

Zero One Loss Function: 0.035, Accuracy: 0.96, Precision: 0.94, Recall: 0.95, F1-Score: 0.94
Results with Noise
Confusion Matrix

Actual ↓ Predicted →	Iris Setosa	Iris Versicolour	Iris Virginica
Iris Setosa	50	0	0
Iris Versicolour	0	46	4
Iris Virginica	0	4	46

Zero One Loss Function: 0.035, Accuracy: 0.96, Precision: 0.94, Recall: 0.95, F1-Score: 0.94

3.2 Glass Dataset

Confusion Matrix

Actual ↓ Predicted →	Glass Type 1	Glass Type 2	Glass Type 3	Glass Type 5	Glass Type 6	Glass Type 7
Glass Type 1	63	1	5	0	1	0
Glass Type 2	1	70	2	0	3	0
Glass Type 3	0	0	17	0	0	0
Glass Type 5	0	0	0	13	0	0
Glass Type 6	0	0	0	0	9	0
Glass Type 7	0	0	0	0	0	29

Zero One Loss Function: 0.03, Accuracy: 0.97, Precision: 0.85, Recall: 0.87, F1-Score: 0.85
Results with Noise
Confusion Matrix

Actual ↓ Predicted →	Glass Type 1	Glass Type 2	Glass Type 3	Glass Type 5	Glass Type 6	Glass Type 7
Glass Type 1	64	0	5	0	1	0
Glass Type 2	0	69	2	1	4	0
Glass Type 3	0	0	17	0	0	0
Glass Type 5	0	0	0	11	2	0
Glass Type 6	0	0	0	0	9	0
Glass Type 7	0	0	0	0	2	27

Zero One Loss Function: 0.05, Accuracy: 0.94, Precision: 0.78, Recall: 0.79, F1-Score: 0.77

3.3 House-Votes-84 Dataset

Confusion Matrix

Actual ↓ Predicted →	republican	democrat
republican	158	10
democrat	34	233

Zero One Loss Function: 0.10, Accuracy: 0.90, Precision: 0.90, Recall: 0.90, F1-Score: 0.89
Results with Noise
Confusion Matrix

Actual ↓ Predicted →	republican	democrat
republican	159	9
democrat	34	233

Zero One Loss Function: 0.09, Accuracy: 0.90, Precision: 0.89, Recall: 0.91, F1-Score: 0.89

3.4 Soybean-Small Dataset

Confusion Matrix

Actual ↓ Predicted →	D1	D2	D3	D4
D1	10	0	0	0
D2	0	10	0	0
D3	0	0	9	1
D4	0	0	0	17

Zero One Loss Function: 0.01, Accuracy: 0.98, Precision: 0.76, Recall: 0.77, F1-Score: 0.76
Results with Noise
Confusion Matrix

Actual ↓ Predicted →	D1	D2	D3	D4
D1	10	0	0	0
D2	0	10	0	0
D3	0	0	2	8
D4	0	0	0	17

Zero One Loss Function: 0.05, Accuracy: 0.94, Precision: 0.58, Recall: 0.6, F1-Score: 0.58

3.5 Breast-Cancer-Wisconsin DataSet

Confusion Matrix

Actual ↓ Predicted →	Benign (2)	Malignant (4)
Benign (2)	458	0
Malignant (4)	2	239

Zero One Loss Function: 0.00, Accuracy: 0.997, Precision: 0.997, Recall: 0.996, F1-Score: 0.997

Results with Noise

Confusion Matrix

Actual ↓ Predicted →	Benign (2)	Malignant (4)
Benign (2)	457	1
Malignant (4)	5	236

Zero One Loss Function: 0.008, Accuracy: 0.991, Precision: 0.992, Recall: 0.989, F1-Score: 0.990

3.6 Graphs

To clarify the results, I have included graphs to help visualize them. In the next section, I have also performed a statistical analysis to confirm the results, which are depicted here.

Accuracy with and without Noise across all Datasets

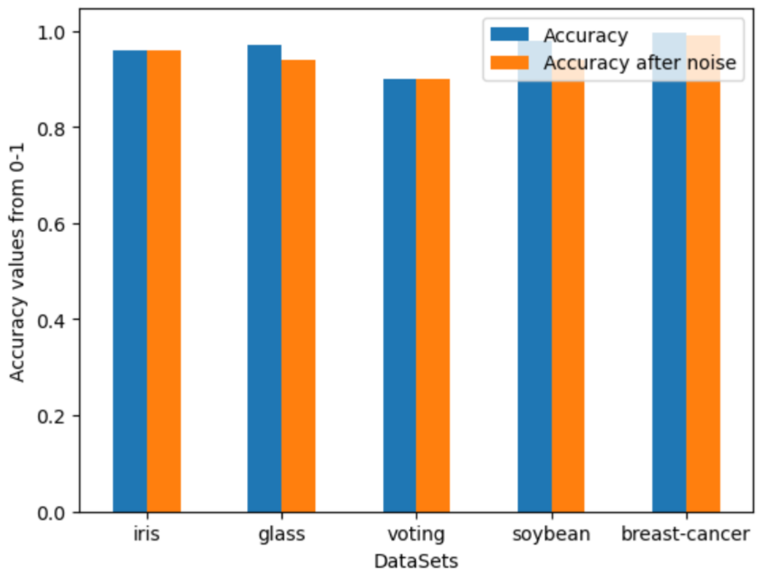

Figure 1: Caption

Zero One Scores with and without Noise across all Datasets

F1 Score with and without Noise across all Datasets

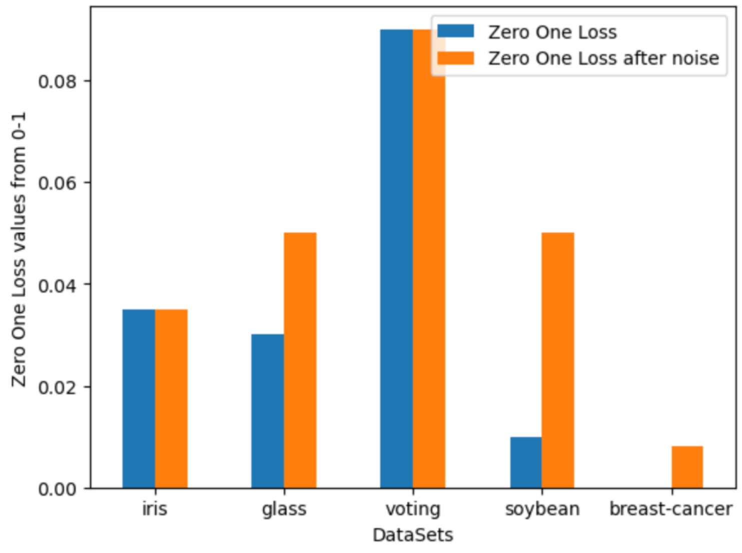

Figure 2: Caption

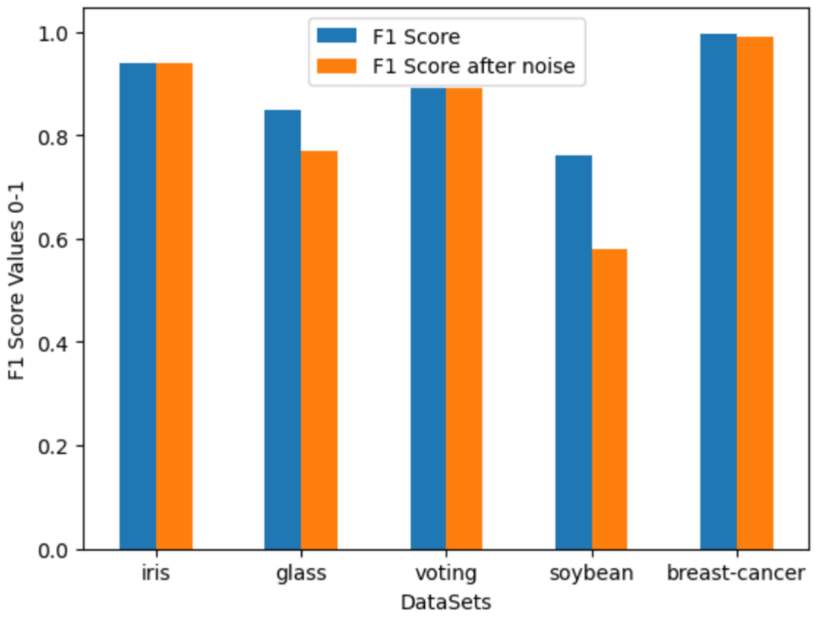

Figure 3: Caption

4. Conclusion and Discussion

In the previous section, I discussed the results for accuracy, zero-one loss, and the F1 score. Let's start with the accuracy, which tells us the proportion of correctly classified instances out of the total instances. From the results, we can observe that introducing 10% noise does not significantly affect the accuracy across all datasets. To further investigate this, I

conducted a paired t-test to assess whether the observed differences in accuracy are statistically significant. The p-value was compared to a threshold of 0.05.
The difference in accuracy before and after introducing noise was not statistically significant (t-value: 1.82, P-value: 0.14), indicating that the noise did not affect the model's performance. Additionally, with 95% confidence, the true mean difference in accuracy due to the noise lies between -0.082 and 0.11, further supporting the conclusion.
Zero One Los function is the fraction of incorrectly classified instances in a dataset, as it returns 1 returns a 1 for miss classification. .
The difference in zero-one loss function result before and after introducing noise was not statistically significant (t-value: -1.34, P-value: 0.25), Hence we didn't get evidence to reject the null hypothesis that the noise did not affect the model's performance. Additionally, with 95% confidence, the true mean difference in accuracy due to the noise lies between -0.035 and 0.01, further supporting the conclusion.
Just as accuracy tells us the proportion of correctly classified instances and zero-one loss indicates the misclassified instances, I also calculated precision, recall, and F1 scores. Recall is particularly important when the dataset is imbalanced or when the risk of missing true positives is high, as it measures how well the model identifies positive instances. However, it does not account for false positives. On the other hand, precision reflects the model's ability to correctly identify actual positive instances, minimizing false positives
The F1 Score is the harmonic mean of precision and recall, and the difference in F1 score result before and after introducing noise was also not statistically significant (t-value: 1.52 p-value 0.20), Hence we didn't get evidence to reject the null hypothesis that the noise did not affect the model's performance. Additionally, with 95% confidence, the true mean difference in accuracy due to the noise lies between -0.035 and 0.01, further supporting the conclusion.
From the above results, I did not find statistically significant evidence to reject the null hypothesis that noise has no effect on the performance of the Naive Bayes model. Therefore, I reject the alternative hypothesis.

Appendix A. Citations

- For Grammatical errors and sentence formations, I used OpenAI (2024) and Inc. (2024)

References

Data Science Stack Exchange. Micro-average vs. macro-average performance in a multiclass classification setting, 2016. URL `https://datascience.stackexchange.com/questions/15989/micro-average-vs-macro-average-performance-in-a-multiclass-classification-settin`. Accessed: 2024-09-12.

GeeksforGeeks. Naive bayes classifiers. `https://www.geeksforgeeks.org/naive-bayes-classifiers/`, 2024. Accessed: 2024-09-09.

Grammarly Inc. Grammarly, 2024. URL `https://www.grammarly.com`. Accessed: YYYY-MM-DD.

OpenAI. Chatgpt: Correcting english writing, 2024. URL `https://www.openai.com/chatgpt`. Accessed: YYYY-MM-DD.

Sebastian Raschka. Naive bayes classifier, 2014. URL `https://sebastianraschka.com/Articles/2014_naive_bayes_1.html`.