Exploring the Effects of Noise on the Training Process of Naive Bayes Classifiers

Braeden Hunt BRAEDEN.HUNT@STUDENT.MONTANA.EDU
Department of Computer Science
Montana State University
Bozeman, MT 59715, USA

Tyler Koon TYLER.KOON@STUDENT.MONTANA.EDU
Department of Computer Science
Montana State University
Bozeman, MT 59715, USA

Editor: Braeden Hunt and Tyler Koon

Abstract

In this paper, we explore the effects of introducing noise into the training process of a Naive Bayes classifier. We consider how the introduction of noise impacts the relative performance of the model when compared with models whose training data remained unaltered. For this study, we quantify performance with two evaluation metrics: 0/1 Loss and an F_1 Score. This difference in relative performance is measured experimentally using 10-fold cross validation and a parametric t-test, across five independent and varied data sets. These five experiments produced results which defied our initial expectations of a difference in performance with statistical significance, suggesting that the Naive Bayesian learning algorithm may be more robust to noise than originally assumed.

Keywords: Bayesian Classifiers, Cross Validation, Noisy Data

1. Introduction

The task of classifying an observation into one or more categories has long been the subject of extensive research in the machine learning and data science communities—and for good reason. The capabilities of modern classification systems expose complex relationships hidden in large data sets. The extrapolation of these relationships can profoundly impact the research and development processes in nearly every field of study. These advantageous offerings subject classification systems to comprehensive study which attempts to develop a more complete understanding of the underlying learning algorithm's behavior. Our paper contributes to this goal by considering a common issue that accompanies the application of these classifiers: noisy data. In particular, our research explores how the introduction of noise into the training data set affects the resulting performance of a Naive Bayes classifier.

This question is explored in the context of five independent data sets, each of which were used to train two models: one with training data in which the values for 10% of the features were randomly shuffled (to induce noise), and a second model trained on unaltered data (control). The performance of these models was defined by the results of 10-fold cross validation with two evaluation metrics: 0/1 Loss and an F_1 Score. Given that that the the

Naive Bayes algorithm assumes each parameter is conditionally independent with respect to classification, we predicted that the two performance metrics would report statistically significant lower values for those models trained on noisy data. Our experiments provided interesting results with regards to the magnitude of this expected difference in performance, suggesting that the Naive Bayes learning algorithm may be more robust to noise than we initially expected.

The following sections provide a holistic overview of these results and their implications, starting with an in-depth review of our experimental design in Section 2. This discussion is complimented by a presentation of our results in Section 3, the implications of which are further discussed in Section 4. Lastly, we conclude with a summary of the experiment and consideration of future research in Section 5.

2. Experimental Design

In this section, we review the details of our experimental process. This includes a description of our chosen data sets (Section 2.1), an overview of our code-base used to perform the experiments (Section 2.2), a discussion of our process for quantifying the performance of a model (Section 2.3), a breakdown of our implementation of the Naive Bayes algorithm (Section 2.4), and finally an overview of our experiment's procedures (Section 2.5).

2.1 Data Sets

We used five different data sets from the UCI Machine Learning repository to test our hypothesis on. These data sets were: Wisconsin Breast Cancer Database (January 8, 1991), 1984 United States Congressional Voting Records Database, Glass Identification Database, Iris Plants Database, and Small Soybean Database. All of these sets took the form of comma separated values (CSV), offering a standard interface for working with the data in our program. These sets varied greatly in the types of data, the total number of records, and the number of classes that the data was grouped in. Accompanying this variation were features of the data sets that needed to be corrected within our preprocessing stage, such as dealing with missing values or discretizing continuous-valued parameters. Our response to these problems is documented in Section 2.2 and Section 2.5.

2.2 Program Design

To implement the experiments outlined in Section 2.5, we developed a set of tools using Python. These tools consisted of five classes, each of which correlated to a step in our experimental process. The first of these classes is responsible for loading and processing the raw data for a given data set. This *Preprocessor* class implements the Pandas data manipulation library to parse data from CSV files and perform basic data mutation such as binning and value conversion. This processed data is then stored in a Pandas DataFrame, which can be saved/loaded to/from persistent storage.cat.montana.edu/d2l/home

The *Algorithm* class is responsible for implementing our learning algorithm. It expects a Pandas DataFrame object containing preprocessed data, which it then uses to compute the probability distributions defined by our custom implementation of the Naive Bayes algorithm (see Section 2.4 for details). These learned probability distributions are then

exposed through a public *predict* method which returns the classification label with the largest probability for a novel sample.

Tools for evaluating model performance are split between two classes: *CrossValidation* and *EvaluationMeasures*. The first class is responsible for implementing the cross-validation logic used to compute the mean performance of a model. This includes folding of the data into k stratified folds, performing alterations to the training data (i.e. the shuffling process described in Section 2.5), and of course computing the four performance metrics described in Section 2.3 for each fold. This evaluation process is implemented generically, allowing for any valid learning algorithm and data set to be used. Additionally, the algorithms used to compute each performance metric are defined in the *EvaluationMeasures* class, with each algorithm having its own public class method. This separation of classes and generic implementation of the cross-validation method offered a more scalable and robust approach to model evaluation, which made "on-the-fly" changes more convenient during our experiments.

The last class in our program offers a set of utilities that facilitate our individual experiments. This *ExperimentHelper* class includes methods for bridging the *CrossValidation* and *EvaluationMeasures* classes, producing well-formatted data visualizations, and performing a parametric t-test and Kolmogorov-Smirnov test for normality.

2.3 Evaluation Methodology

For this study, we defined the performance of a model to be characterized by two metrics: 0/1 Loss and F_1 Score. In our evaluation process, the precision and recall metrics for each class were encoded by an F_β score with a beta value of 1 (F_1 Score). The reasoning behind our decision to use a β value of 1 for the F_β metric is quite simple: it equally weighs precision and recall. We believed this option to be most appropriate with regards to our goal of comparing the *general* performance of models. That said, considering other β values could be of interest in future work that targets more specific contexts.

These two performance metrics were chosen for their robustness and computational simplicity, a feature highly sought after given the time-constrained nature of this project. Perhaps more importantly however, each metric is based on a standardized scale which simplified the comparison process when considering the results for multiple models: that is, models with a larger reported value for a given metric were considered more performant than similar models with reportedly lower scores for the respective metric. Additionally, considering multiple performance metrics provides a more holistic overview of the model's performance (0/1 Loss) and a finer level of detail in regards to how the model performed on a class-to-class basis (F_1).

2.4 Algorithm

The underlying classifier used in this study was the Naive Bayes learning algorithm (Figure 1a). Besides being a requirement of the project, using a naive implementation of the Bayesian algorithm allowed for the assumption of conditionally independent parameters. This assumption helped address problems of underfitting, which was a concern given the relatively small data sets used in our experiments. Additionally, our implementation of the Naive Bayes classifier leveraged *Laplace Smoothing* to handle missing representation for

$$c(o) = \max_{c_j \epsilon C}\left(\frac{\text{count}(x \epsilon c_j)}{N} \Pi_{a=1}^{A} \frac{\text{count}(x_a = o_a \wedge x \epsilon c_j)}{N}\right)$$

(a) Naive Bayes

$$c(o) = \max_{c_j \epsilon C}\left(\frac{\text{count}(x \epsilon c_j)}{N} \Pi_{a=1}^{A} \frac{\text{count}(x_a = o_a \wedge x \epsilon c_j)+1}{N+A}\right)$$

(b) Naive Bayes w/ Laplace Smoothing

A = The set of all parameters being learned	C = The set of all classification levels
o = A novel observation containing o_a for all $a \epsilon A$	N = The number of all samples
j = A specific classification level $j \epsilon C$	a = A specific parameter $a \epsilon A$

Figure 1: Naive Bayes Implementations

a given parameter–class condition (Figure 1b), once again addressing the shortcomings of training on small data sets. Another important detail of our algorithm implementation is how we handle missing values for a given parameter. Instead of simply treating a missing value as being non-existent, which would compound the underfitting problem, we decided to consider the conditional probability for the given parameter-class combination to be 1. The reason for this assumption is that every combination of class with this missing parameter will have the same conditional probability (1), essentially removing the effects of the missing value on the resulting score for each class *relative* to all other classification scores.

2.5 Experiments

Our study consisted of five independent experiments corresponding to each of the five provided data sets. All of these experiments followed a similar design built around hypothesis testing. The structure for each experiment is broken down into three stages, each of which are described in the following paragraphs: Preprocessing, Training and Evaluation, and Hypothesis Testing.

The preprocessing stage involved loading and wrangling data. Though the extent of this stage was dependent on the data set being processed, the general steps included parsing the data from a CSV file, defining and normalizing the column names, removing unrelated columns (such as those used for relative IDs), and discretizing any continuous-valued columns. The discretization method we decided to implement was equal-width binning. Our motivations behind using this method stem from its simple yet effective nature. More advanced discretization methods were considered, such as equal-frequency binning, however their impact on the study's results were deemed negligible as we were only focusing on *relative* performance differences between the models trained on unaltered data and those trained on altered data. For this same reason, we decided to manually tune this hyperparameter. We concluded that implementing a hyperparameter search method such as random search or grid search would have little effect on the difference in relative performance between two models, as long as the number of bins for each parameter remained constant for both training sessions.

After preprocessing a given data set, we began two different 10-fold cross validations. In one of these validations, we shuffled 10% of the features within their respective columns in the training data for each iteration. The other validation did not experience any shuffling. This shuffling of data provided a controlled means of introducing random noise into a model, the effects of which were the subject of the experiment. These two validations were then compared with the two evaluation metrics described in Section 2.3 (0/1 Loss and F_1). The results of these metrics were stored for each fold of the cross validation, and evaluated in the third and final stage of our experiments.

The final stage of our experiments compares the performance of the model trained on unaltered data with the model trained on altered data (induced noise). In particular, a parametric t-test is performed for each of the four evaluation metrics described in Section 2.3, under the null hypothesis that the means of the performance metric is identical for the two models. We used the resulting t-statistic and p-value to quantitatively compare the difference in performance between the two models, considering a significance level of 0.05 or less ($\alpha = 0.5$) to indicate strong evidence that the two models performed differently. Before interpreting these results however, we performed Kolmogorov-Smirnov tests of normality to verify the normality conditions of the underlying test data. For situations in which this normality condition was not satisfied, we replaced the p-value with the label *Normality Not Met (N. N. M.)*. The results of these tests can be found in the respective tables from Section 3.

3. Results

Here we report our results for each of the data sets.

3.1 Breast Cancer Identification Results

The Wisconsin Breast Cancer Database (January 8, 1991) has instances of breast tumors. Each instance has nine categorical features, each having categories 1-10. These nine features were used to train the models. It has a tenth feature that is used as the class, whether or not the tumor is malignant or benign. This is a Boolean value, meaning there are only two classes. For this data set, we considered malignant to be "positive" and benign as "negative." There is a total of 699 instances in the data set, 458 benign, and 241 malignant. There are 16 instances missing a single attribute. There are no instances missing two or more attributes.

The results from the experiment:

Training Data	Measure	Mean	Min	Max	Std.
Unaltered	0/1 Loss	0.971	0.928	1.000	0.020
	F_1 Score	0.959	0.898	1.000	0.029
Altered	0/1 Loss	0.973	0.944	0.986	0.016
	F_1 Score	0.962	0.926	0.980	0.021

Measure	p-value
0/1 Loss	0.849
F_1 Score	0.823

3.2 Congressional Voting Identification Results

The 1984 United States Congressional Voting Records Database records instances of United States House of Representatives Congressmen's voting records. There are 16 ternary categorical features, representing "yea," "nay," and "present" votes on specific pieces of legislation. The 17^{th} feature is Boolean valued, representing whether the congressman is a Democrat or Republican. For this data set, we considered democrats to be a "positive" value and republicans to be a "negative" value based on a coin flip. The data set has 435 instances, 267 are democrats, and 168 are republicans. There are no missing attributes in this data set.

The results from the experiment:

Training Data	Measure	Mean	Min	Max	Std.
Unaltered	0/1 Loss	0.899	0.841	0.955	0.029
	F_1 Score	0.915	0.863	0.963	0.026
Altered	0/1 Loss	0.901	0.841	0.955	0.036
	F_1 Score	0.917	0.868	0.962	0.030

Measure	p-value
0/1 Loss	0.884
F_1 Score	0.890

3.3 Iris Plant Identification Results

The Iris Plants Database has 150 instances of Iris plants. It has four continuous features for different measurements of the plants, each of which were divided into 5 equal-width bins using our discretization process described in Section 2.5. It is separated into three classes: Setosa, Versicolour, and Virginica, with each class having 50 instances in the data set. There are no missing attributes in the data.

The results from the experiment:

Training Data	Measure	Positive Class	Mean	Min.	Max	Std.
Unaltered	0/1 Loss	—	0.920	0.733	1.000	0.088
	F_1	Setosa	1.000	1.000	1.000	0.000
		Versicolour	0.873	0.500	1.000	0.155
		Virginica	0.881	0.667	1.000	0.122
Altered	0/1 Loss	—	0.927	0.800	1.000	0.066
	F_1	Setosa	1.000	1.000	1.000	0.000
		Versicolour	0.889	0.727	1.000	0.101
		Virginica	0.888	0.667	1.000	0.104

Measure	Positive Class	p-value
0/1 Loss	—	0.850
F_1	Setosa	~ 1.000
	Versicolour	0.785
	Virginica	0.905

3.4 Soybean Identification Results

The Small Soybean Database has 47 instances of soybeans with 35 categorical features describing each one. The soybeans are classified into four groups: D1 (10 instances), D2 (10 instances), D3 (10 instances), and D4 (17 instances). There are no missing attributes in the data.

The results from the experiment:git push –set-upstream origin Implement-KS-Test

Training Data	Measure	Positive Class	Mean	Min.	Max	Std.
Unaltered	0/1 Loss	—	0.845	0.750	1.000	0.109
	F_1	D1	1.000	1.000	1.000	0.000
		D2	1.000	1.000	1.000	0.000
		D3	0.300	0.000	1.000	0.483
		D4	0.820	0.667	1.000	0.137
Altered	0/1 Loss	—	0.785	0.750	0.800	0.024
	F_1	D1	1.000	1.000	1.000	0.000
		D2	1.000	1.000	1.000	0.000
		D3	0.000	0.000	0.000	0.000
		D4	0.760	0.667	0.800	0.064

Measure	Positive Class	p-value
0/1 Loss	—	N. N. M.
F_1	D1	~ 1.000
	D2	~ 1.000
	D3	N. N. M.
	D4	N. N. M.

3.5 Glass Identification Results

The Glass Identification Database has 214 instances of glass. Each instance has nine continuous features, each of which were divided into 5 equal-width bins using our discretization process described in Section 2.5. The glass has seven classes labeled 1-7. The number and description for each class can be found in the table below.

Class ID	Description	Count
1	Float processed building window	70
2	Float processed vehicle window	17
3	Non-float processed building window	76
4	Non-float processed vehicle window	0
5	Container	13
6	Tableware	9
7	Headlamp	29

The results from the experiment:

Training Data	Measure	Positive Class	Mean	Min.	Max	Std.
Unaltered	0/1 Loss	—	0.528	0.391	0.778	0.112
	F_1	1	0.589	0.286	0.857	0.147
		2	0.392	0.000	0.750	0.227
		3	0.000	0.000	0.000	0.000
		4	—	—	—	—
		5	0.200	0.000	1.000	0.350
		6	0.100	0.000	1.000	0.316
		7	0.886	0.750	1.000	0.102
Altered	0/1 Loss	—	0.494	0.273	0.650	0.110
	F_1	1	0.539	0.143	0.737	0.179
		2	0.356	0.000	0.600	0.186
		3	0.000	0.000	0.000	0.000
		4	—	—	—	—
		5	0.200	0.000	0.667	0.322
		6	0.100	0.000	1.000	0.316
		7	0.887	0.667	1.000	0.128

Measure	Positive Class	p-value
0/1 Loss	—	0.500
F_1	1	0.506
	2	0.707
	3	~ 1.000
	4	—
	5	N. N. M.
	6	N. N. M.
	7	0.986

4. Exploration of Results

We now consider the results of these experiments and any conclusions we can draw relative to our Naive Bayes learning algorithm. Additionally, we consider limitations of the study and propose future research that could be done in this area of study.

4.1 Discussion

Our initial hypothesis was that across all of the data sets, the models that were trained on the noisy data would perform statistically worse than the models trained on data without introduced noise. After performing t-tests on the fold measures for each of the experiments, and using $\alpha \leq 0.05$, we were not able to reject the null hypothesis for any experiment. This means that there was not a single instance of the models trained on noisy data performing statistically worse than the models trained on unaltered data. This runs counter to what we expected and speaks to the robustness of this algorithm to noise. We suspect that this may be due to the algorithm placing very strong emphasis on matching a few features to many examples in the training data rather than matching many features to few examples in the training data. Altering only 10% of the examples still allows the algorithm to match

90% of the original training examples, which likely leads to its robustness. Future research should be conducted to determine the effects of increasing the amount of noise introduced into the data.

4.2 Limitations

One of the major limitations to these results is the data set sizes. A couple of our data sets were quite small, had classes with very few examples, or both. When running 10-fold cross validation, having less than 20 examples of a class means that each fold will have at most two examples of that class. This also meant getting measures from each fold that met normality was very difficult and improbable. This limited the t-tests that we were able to run. However, based on our other results, we do not believe that had these limitations been removed, our results would have changed drastically.

Another limitation to our algorithm's ability to classify data is our use of discretizing data. Our algorithm only works with categorical data, so we had to bin our continuous data. We used bins of equal-width, which can lead to a loss of resolution in the data, loosing any fine details within each bin and cross bin borders. Our algorithm also only uses nominal categories, whereas binned continuous data should be treated as ordinal. This also reduces the performance of the algorithm. However, since we controlled for this by using the same binning strategies for both the models trained on unaltered and altered data, this should not affect our results. We were not trying to determine the overall performance of the algorithm, but the differential performance between the two differently trained models.

5. Summary

The ability to accurately classify new data based on previous observations is highly desired. These classification systems can profoundly impact the research and development processes in nearly every field. Our research attempts to contribute to the understanding of these classification systems by exploring the effects of noise on the training processes for supervised learning models. In particular, we considered how the average performance of a Naive Bayes classifier differs when trained on altered (noisy) data and unaltered data. To assess this problem, we considered the performance of a model to be defined by the average 0/1 Loss and F_1 Score produced from performing 10-fold cross validation with stratification. Using this problem statement and deliverable, we developed a robust set of tools using Python and Pandas to perform our experiment on five independent and varied data sets. Our experimental design implemented stratified 10-fold cross validation on each of the trained models, returning data for each of the previously mentioned performance metrics. These data were then used to perform a parametric t-test (after verifying the normality condition), the results of which indicated that there was weak evidence against the null hypothesis of noisy training data resulting in a less performant model. These results contradicted our original hypothesis in which we expected the performance difference between models trained with altered (noisy) data and those trained on unaltered data to be statistically significant. As such, we concluded that the Naive Bayesian classifier is much more robust than initially expected. That being said, there are a number of limitations with our experimental design which warrant further research of the subject, such as exploring the effects of noise when training with much larger data sets.

References

We used the course materials provided to us by Dr. Sheppard.