Naïve Bayes Project - CS-5333

Rylee Buchert

January 2022

Abstract

Naïve Bayes is a machine learning algorithm based on Bayes' rule in statistics that is used for classification problems. Although the Naïve Bayes models are built on simplistic probability calculations and assumptions, research has demonstrated that the algorithm performs well in classification tasks. In this project, two versions of Naïve Bayes classifiers were implemented: the multivariate Bernoulli and multinomial event models. These models were tested on multiple datasets to evaluate the strengths and weaknesses of both approaches. A text classification dataset containing user reviews and a hand-written digit recognition file were utilized to evaluate both models. The multinomial event model demonstrated significantly better performance than the Bernoulli model when tasked with digit recognition, while both models performed similarly in text sentiment classification tests.

1 Introduction

Naïve Bayes classifiers are a family of probabilistic machine learning algorithms that categorize a problem instance based on a given feature set. The algorithms in the Naïve Bayes family are built upon Bayes' theorem in statistics, used for calculating conditional probabilities based on prior event knowledge. Using this rule and the fundamental assumption of conditional feature independence, Naïve Bayes models have demonstrated high classification accuracy while generating predictions in a computationally efficient manner. The independence assumptions allow each attribute parameter to be estimated separately, resulting in a substantial reduction of calculation requirements. Despite the "naïve" assumption, which rarely is valid for a dataset, the model still performs very well in classification tasks.

This form of supervised learning has had many real-world applications, ranging from text classification to sentiment analysis and medical diagnoses. This project tasked us with implementing and testing multiple versions of the Naïve Bayes algorithm to compare the efficacy of the model. The algorithms were evaluated on two datasets in this project: a collection of user reviews for text sentiment classification and a group of hand-written numbers for digit recognition. Naïve Bayes models have been commonly used for similar datasets in many practical applications, highlighting the relevance of developing such learning methods.

Two forms of the Naïve Bayes algorithm were implemented in this project: the multivariate Bernoulli and multinomial event models. Both versions take a similar approach in estimating the probabilities of category labels, combining knowledge of class priors and the likelihood of each feature to generate predictions. The main difference between the models regards how they represent Bayes' formula. The multinomial model factors in repetitions of feature instances during calculations, while the multivariate Bernoulli model does not consider repetitions, focusing only on whether the features are present. The multinomial model demonstrated significantly better prediction accuracy for the digit recognition dataset, while both models performed similarly well when tasked with classifying text sentiment.

2 Data

2.1 User Review Text Classification

The first dataset was collected from the University of California-Irvine's Machine Learning Repository (Kotzias, Denil, De Freitas, & Smyth, 2015). The "Sentiment Labeled Sentences" dataset includes 3,000 user reviews collected from IMDB, Amazon, and Yelp with 'sentiment' labels for each instance. Reviews are labeled '1' if positive and '0' if negative. This resource provides an effective way to measure both models' aptitude in text classification, utilizing a realistic way in which Naïve Bayes is applied.

A few modifications to the data were required to correctly input the training and test sets into both event models. The original data included columns for both the review and associated label. Reviews were in sentence form, containing the exact text of the review posted on the respective websites. In order to get each review in a form the model can recognize, the text was modified into a list of words with punctuation and capitalization removed. An instance of the dataset which previously looked like:

Review	Label
"The mic is great."	1

Would be transformed into:

Review	Label
[the, mic, is, great]	1

This modified form allows for the simple creation of a vocabulary and word dictionary. The vocabulary is a set of all words included in user reviews and is used in probability calculations/creation of the word dictionary. The word dictionary includes a row for every review in the training set and a column for each word in the vocabulary. For the Bernoulli model, the word dictionary is one-hot encoded, taking a value of '1' for each word in the review instance. For the multinomial model, the dictionary includes the number of times each word appears. This data table simplifies the calculation of feature probabilities and class priors when fitting the models.

2.2 Hand-Written Digit Recognition

The second dataset utilized in this report comes from the Modified National Institute of Standards and Technology (MNIST) database, which contains an extensive collection of hand-written digits (Deng, 2012). Each digit image is 28 pixels in height and width, combining for 784 total pixels. For every digit instance, an integer value between 0 and 255 is listed for each column (Pixel0, Pixel1, ... , Pixel783). The integer corresponds to the "intensity" of the pixel, with higher numbers indicating a more significant intensity value. The images depicted in figure 1 are examples of hand-written digits included in the data.

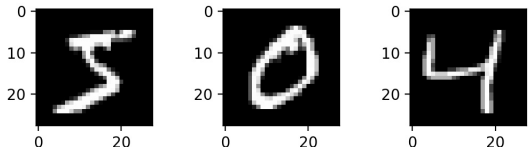

Figure 1: Examples of Digit Data

The MNIST data required less pre-processing than the review dataset since the pixels were already in 'dictionary' form. The only required change to the test set was to modify pixel intensity to a binary value of 'off' or 'on', rather than a continuous number. This allowed the event models to calculate feature probabilities based on whether each individual pixel was on, rather than formulating based on intensity. The final set contained 42,000 unique images used for training and testing purposes.

3 Models

3.1 Background

Machine learning algorithms can be broken down into two categories: discriminative and generative models. Both classifier models try to estimate $P(Y|X)$, the conditional probability

of event Y occurring given that event X is true. Discriminative classifiers attempt to find a function $f(x)$ which maximizes the conditional probability of $P(Y|X)$. Models such as logistic regression will find a decision boundary and classify new points based on where they fall relative to the line.

Generative learning algorithms, including Naïve Bayes, attempt to find $P(Y|X)$ using a different approach. These classifiers learn the characteristics of each class label and construct a model of the data's distribution. Using these models, generative classifiers will evaluate the likelihood that a new instance will fall into each class, choosing the one with the higher probability. Accordingly, Generative models can create new data instances, which discriminative models cannot do. Naïve Bayes algorithms are able to estimate $P(Y|X)$ this way using Bayes' rule:

$$P(Y|X) = \frac{P(X|Y) * P(Y)}{P(X)}$$

Where $P(X|Y)$ refers to the conditional probability of event X given Y is true, and $P(X)$ & $P(Y)$ are the probabilities of events X and Y occurring without any added conditions. This computation is performed for each class label, determining the probability that a new data instance fits into each potential category. This is formally noted as:

$$\underset{y}{\text{argmax}}\, P(Y|X) = \underset{y}{\text{argmax}}\, P(X|Y) * P(Y)$$

Because the denominator in Bayes' theorem, $P(X) = P(X|Y = 1) * P(Y = 1) + P(X|Y = 0) * P(Y = 0)$, is the same for all classes, it can be dropped from the maximum likelihood equation. Thus, to create a generative model for $P(Y|X)$, we only have to calculate $P(Y)$ and $P(X|Y)$. $P(Y)$ can be obtained by dividing the number of instances with each class label by the total number of instances. These are known as the class priors.

Estimating $P(X|Y)$ is a more difficult computation. Using the example of text classification, the feature vector X would be represented by a list of the length of the vocabulary. Each entry in the feature list X_i is set to 1 if the i-th word in the vocabulary is included in the review instance and set to 0 if not included. A feature vector for the review "the mic is great" would be represented:

Vocab:	a	aardvark	...	mib	mic	mica	...	the	...	zyzzyva
x:	[0	0	...	0	1	0	...	1	...	0]

With the feature vectors created for each review, the next step is to model $P(X|Y)$ for each word in the vocabulary. If the vocabulary contained 10,000 words with $x_i \in \{0, 1\}^{10000}$, $P(X|Y)$ would be estimated by:

$$P(X_1, ..., X_{10000}|Y) = P(X_1|Y)P(X_2|Y, X_1)P(X_3|Y, X_1, X_2)...P(X_{10000}|Y, X_1, X_2, ..., X_{9999})$$

This computation requires too many parameters to estimate, necessitating a change to the formulation. The assumption of conditional feature independence, known as the Naïve Bayes assumption, is adopted to simplify the calculation. When modeling $P(X|Y)$, the

assumption is made that all features X_i are conditionally independent given Y. Using the review classification example, this would mean that knowledge of a particular word being included in a review does not change the belief that any other word would also be included. With this assumption, $P(X|Y)$ can be modeled:

$$\begin{aligned} P(X_1, ..., X_{10000}|Y) &= P(X_1|Y)P(X_2|Y)P(X_3|Y)...P(X_{10000}|Y) \\ &= \prod_{i=1}^{n} P(X_i|Y) \end{aligned}$$

This can be computed very easily in comparison to the formulation above. With $P(X|Y)$ and $P(Y)$ known, all of the components for the Naïve Bayes classifier have been obtained.

3.2 Multivariate Bernoulli Event Model

The first version of the Naïve Bayes algorithm implemented in this project is the multivariate Bernoulli event model. This is the most basic version of the classifier used when the features are discretized. For a training set $\{(X^{(i)}, Y^{(i)}); i = 1, ..., m\}$, the joint likelihood is determined by:

$$\mathcal{L}(\phi_y, \phi_{j|y=0}, \phi_{j|y=1}) = \prod_{i=1}^{m} P(X^{(i)}|Y^{(i)})$$

Using maximum likelihood estimation, the parameters $\phi_y, \phi_{j|y=0}, \phi_{j|y=1}$ are expressed:

$$\phi_y = P(Y = 1) = \frac{\sum_{i=1}^{m} 1\{Y^{(i)} = 1\}}{m}$$

$$\phi_{j|y=0} = P(X_j = 1|Y = 0) = \frac{\sum_{i=1}^{m} 1\{X_j^{(i)} = 1 \wedge Y^{(i)} = 0\}}{\sum_{i=1}^{m} 1\{Y^{(i)} = 0\}}$$

$$\phi_{j|y=1} = P(X_j = 1|Y = 1) = \frac{\sum_{i=1}^{m} 1\{X_j^{(i)} = 1 \wedge Y^{(i)} = 1\}}{\sum_{i=1}^{m} 1\{Y^{(i)} = 1\}}$$

The intuition behind these parameter calculations is straight-forward. ϕ_y, the class prior, is the fraction of total instances belonging to class $Y = 1$. $\phi_{j|y=0}$ is calculated as the number of instances where class $Y = 0$ and feature $X_j = 1$ divided by the total number of instances where class $Y = 0$. The same is true for $\phi_{j|y=1}$ except for instances where $Y = 1$.

Once all parameters are known, $P(Y|X)$ can be calculated:

$$P(Y = 1|X) = \frac{(\prod_{i=1}^{n} P(X_i|Y = 1)) * P(Y = 1)}{(\prod_{i=1}^{n} P(X_i|Y = 1)) * P(Y = 1) + (\prod_{i=1}^{n} P(X_i|Y = 0)) * P(Y = 0)}$$

This is calculated for all class values, and the highest posterior probability is chosen. When implementing this model on a computer, the probabilities of individual feature values can become very small with a large vocabulary. To prevent underflow error, the equation is log-transformed. After dropping the denominator (as discussed earlier), the resultant equation is given by:

$$P(Y = 1|X) = log(P(Y = 1)) + \sum_{i=1}^{n} log(P(X_i|Y = 1))$$

3.3 Multinomial Event Model

The multinomial event model is a slight alteration of Naïve Bayes, which can account for repetitions in features when calculating posterior probabilities. Going back to the review classification example, the feature vector X_i is now represented as a list of the length of the i-th review instance. $X_i \in \{1, ..., |V|\}$, where $|V|$ is the length of the vocabulary. A review instance of length n is represented as the vector $(X_1, X_2, .., X_n)$. In this list, each X-value corresponds to the dictionary location of the specified word.

The multinomial model is still of the form $P(Y|X) = P(Y) * \prod_{i=1}^{n} P(X_i|Y)$, but takes on a different meaning than the Bernoulli event model. Instead of $X_i|Y$ being modeled as a Bernoulli distribution, it is now multinomial, allowing the model to factor in word repetitions during probability calculations. As noted above, the word dictionary is modified to count the number of times each word appears in a training example, rather than a binary value if the word is present.

With these adjustments made to the model, the parameters can be estimated. Similar to the Bernoulli formulation, for a training set $\{(X^{(i)}, Y^{(i)}); i = 1, ..., m\}$ where $X^{(i)} = (X_1^{(i)}, X_2^{(i)}, ..., X_{n_i}^{(i)})$ the likelihood can be expressed:

$$\mathcal{L}(\phi, \phi_{i|y=0}, \phi_{i|y=1}) = \prod_{i=1}^{m}(\prod_{j=1}^{n_i} P(X_j^{(i)}|Y; \phi_{i|Y=0}, \phi_{i|Y=1})) * P(Y^{(i)}; \phi_y)$$

Using maximum likelihood estimates for the parameters:

$$\phi_y = P(Y = 1) = \frac{\sum_{i=1}^{m} 1\{Y^{(i)} = 1\}}{m}$$

$$\phi_{k|y=0} = P(X_j = k|Y = 0) = \frac{\sum_{i=1}^{m}\sum_{i=1}^{n_i} 1\{X_j^{(i)} = k \wedge Y^{(i)} = 0\}}{\sum_{i=1}^{m} 1\{Y^{(i)} = 0\} * n_i}$$

$$\phi_{k|y=1} = P(X_j = k|Y = 1) = \frac{\sum_{i=1}^{m}\sum_{i=1}^{n_i} 1\{X_j^{(i)} = k \wedge Y^{(i)} = 1\}}{\sum_{i=1}^{m} 1\{Y^{(i)} = 1\} * n_i}$$

3.4 Laplace Smoothing

The final piece in putting together the Naïve Bayes algorithm is a slight change to the probability calculation known as Laplace smoothing. Suppose the model encounters a word that it had not seen before when determining probabilities for review classification. The posterior probability of this feature ($P(X_i|Y)$) is 0 since it is not located in the model's vocabulary. Because $\prod_{i=1}^{n}$ includes the 0 term multiplied into it, the model would estimate a 0% probability the word appears in either class. As a result, the model cannot estimate class probabilities when it has not seen a given feature before.

To eliminate this problem, Laplace smoothing implements a small change in the probability calculations. Instead of setting the posterior probability of an unknown word to 0, Laplace smoothing acts as if the model has seen every potential word at least once. The parameter maximum likelihood estimate is now updated to be:

$$\phi_{j|y=0} = \frac{\sum_{i=1}^{m} 1\{X_j^{(i)} = 1 \wedge Y^{(i)} = 0\} + 1}{\sum_{i=1}^{m} 1\{Y^{(i)} = 0\} + k}$$

$$\phi_{j|y=1} = \frac{\sum_{i=1}^{m} 1\{X_j^{(i)} = 1 \wedge Y^{(i)} = 1\} + 1}{\sum_{i=1}^{m} 1\{Y^{(i)} = 1\} + k}$$

Where k is the number of classes in the dataset. A similar change is applied to the multinomial model as well, with the slight variation that α is added to the numerator and $\alpha * |V|$ is added to the denominator. α is a small number, usually around 0.1-3.0. For this project, the optimal α was chosen using cross-validation. The updated parameter maximum likelihood estimate for multinomial models is given by:

$$\phi_{k|y=0} = P(X_j = k|Y = 0) = \frac{\sum_{i=1}^{m} \sum_{i=1}^{n_i} 1\{X_j^{(i)} = k \wedge Y^{(i)} = 0\} + \alpha}{\sum_{i=1}^{m} 1\{Y^{(i)} = 0\} * n_i + \alpha * |V|}$$

$$\phi_{k|y=1} = P(X_j = k|Y = 1) = \frac{\sum_{i=1}^{m} \sum_{i=1}^{n_i} 1\{X_j^{(i)} = k \wedge Y^{(i)} = 1\} + \alpha}{\sum_{i=1}^{m} 1\{Y^{(i)} = 1\} * n_i + \alpha * |V|}$$

4 Experimental Results

4.1 User Review Text Classification

To evaluate Naïve Bayes' effectiveness in classifying text, the review sentiment dataset was tested on both event models. After the required data was loaded into the IDE and cleaned, the dataset was split into four categories for evaluation which includes training and test sets for both label and feature columns. This split was accomplished using the sk-learn train-test-split functionality at a 20% test size. Both models iterated over the data one hundred times to determine the average model accuracy. For each iteration, a new split of the original data was created to obtain an accurate measurement of model accuracy.

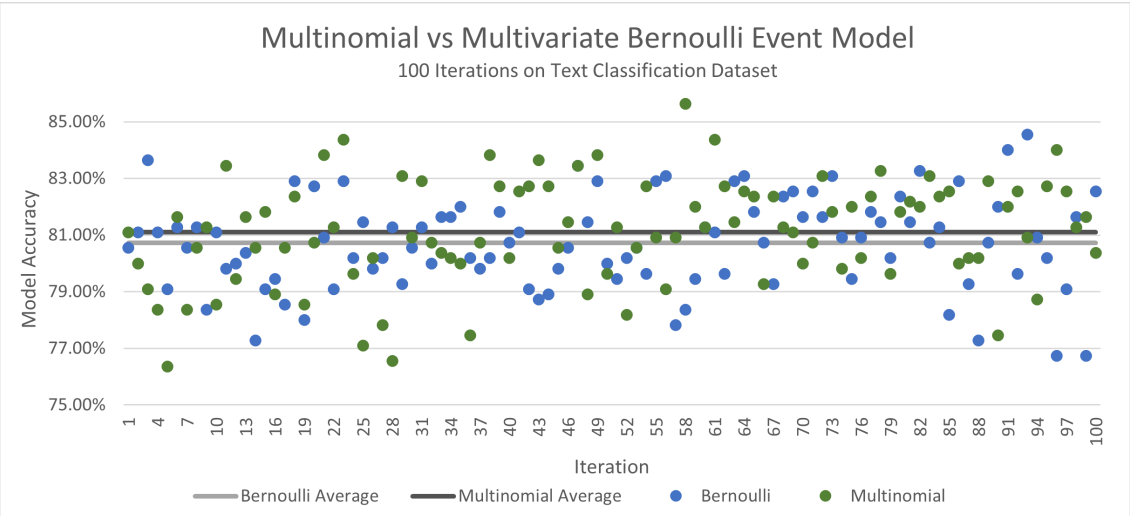

Figure 2: Multivariate Bernoulli vs Multinomial Text Classification Accuracy

Both models performed similarly when fit with the review classification data. The multivariate Bernoulli and multinomial event model results are depicted in figure 2. The graph depicts model accuracy for each iteration of the data and includes an average line for all instances. The multinomial event model classified review sentiment with an 81.11% accuracy while the Bernoulli model achieved an accuracy of 80.72%.

To further visualize the performance of the models, confusion matrices were created which assist in understanding the error rates for each label category. The confusion matrices shown in figure 3 and figure 4 show the count and rate for each classification combination. Both event models were most effective at classifying negative reviews in the sample, shown by 88% and 86% accuracy rates for the Bernoulli and multinomial models, respectively. When confronted with a positive review, the models' performance dropped, only categorizing 75% and 76% of the instances correctly. Moreover, both models exhibited a similar pattern of classifying positive-sentiment reviews as negative, with almost a fourth of all positive reviews categorized in this manner.

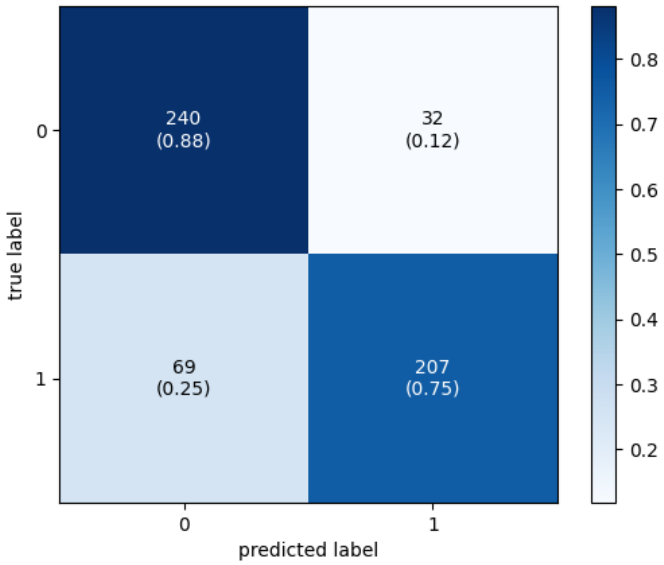

Figure 3: Multivariate Bernoulli Confusion Matrix

These results were not expected when beginning the project. The multinomial model is generally accepted to perform better in text classification tasks than the multivariate Bernoulli model. Although the multinomial event model performed slightly better in the experiment, the difference in model accuracy is marginal and can likely be explained by random noise in the data. One of the primary reasons behind these unexpected results is the limited sample size of the dataset. Less than 3,000 reviews were included in the data, making it more difficult to determine a difference between the event models in the results. The multinomial model also tends to perform better when the dataset contains text instances with many repeated words. In this dataset, the average review length is only 13 words, which does not allow for many potential repetitions that the multinomial model can utilize in probability calculations.

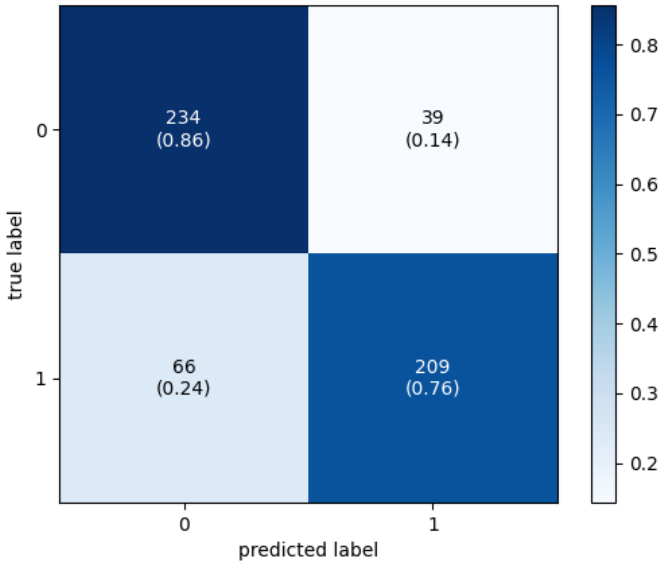

Figure 4: Multinomial Confusion Matrix

4.2 Hand-Written Digit Recognition

In contrast to the text classification results, the event models displayed starkly different performances for the digit recognition data. While the Bernoulli model was only able to label 69.16% of digits correctly (on average), the multinomial model averaged a score of 82.58%. Figure 5 highlights the difference in model performance, where there is over a 14% disparity in classification accuracy between the tests.

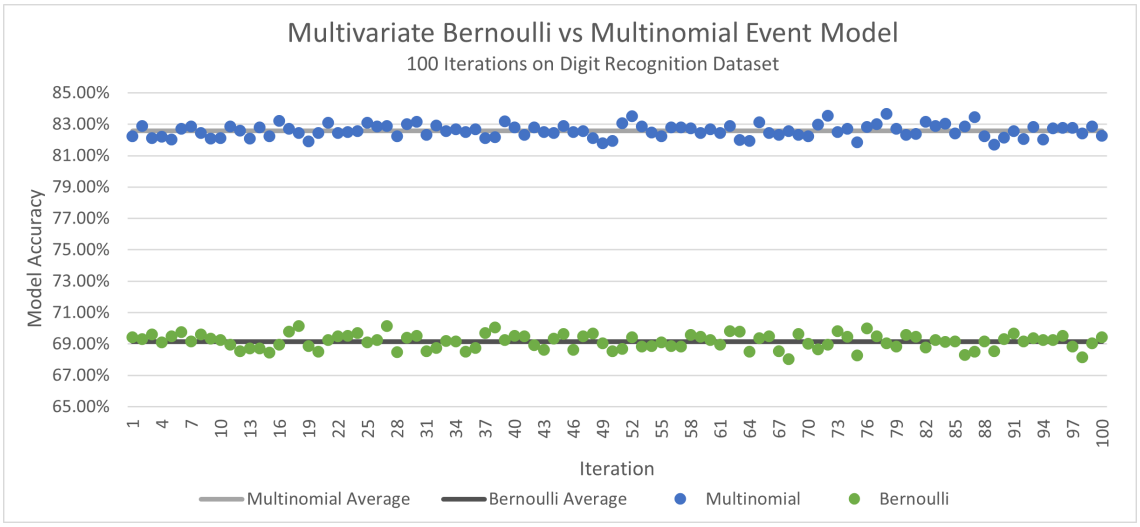

Figure 5: Multivariate Bernoulli vs Multinomial Digit Recognition Accuracy

The differing maximum likelihood estimation approaches are what produced the significant variation in event model accuracy. For the digit dataset, which consists of intensity

values for each pixel, the multinomial approach of probability calculation outperformed the Bernoulli method. While the Bernoulli model divides the number of pixel instances for each class by the number of class instances, the multinomial model divides by the number of pixel instances in each class. Although this distinction in probability calculations is small, the experimental results demonstrate that it has an important impact on model accuracy.

4.3 Laplace Smoothing

5-fold cross-validation was used to select the 'α' laplace smoothing parameter for both multinomial model tests. This hyperparameter selection method works by iterating over the training data k times (in this case, 5) to determine the best parameter value. k-fold cross-validation for a dataset of n instances would consist of k subsets, each of the size $\frac{n}{k}$. Each subset is used once as a test set, and the parameter which produces the highest model accuracy is selected as the α value. An array of α values: $[0.1, 0.3, 1.0, 2.0, 3.0]$ was used to select the optimal hyperparameter. For the text classification dataset, 1.0 was chosen as the α value and for digit recognition, 0.1 was selected.

5 Discussion

I found this project exceptionally helpful in deepening my understanding of how the Naïve Bayes algorithms function. Putting the mathematics behind the algorithms into my own words necessitated a deep understanding of how the models operate. Getting to step through and see how each of the calculations were made helped me develop a firmer grasp on the core concepts of both Naïve Bayes and probability theory. Furthermore, constructing the model from scratch helped familiarize myself with many data structures and programming algorithms that will be used in future projects and machine learning models.

When implementing the algorithms in Python, the most challenging aspect was creating models that were generalizable to multiple datasets where the number of classes differ. For the sentiment classification problem, there were only two labels that the reviews could fall under (0 or 1). The limited number of labels allowed for the simple implementation of a pandas dataframe for each class label with columns dedicated to the subsequently required calculations (priors, number of features, conditional probabilities, etc.) Using these data structures worked for the sentiment classification problem but did not extrapolate well to the MNIST classification dataset.

The digit recognition dataset contained ten potential class labels for each instance, one for each number. When applying the previously created models to this problem, the program ran into issues creating and choosing from which dataframe to pull the data. Instead of having a separate dataframe for each class label, I elected to store each class calculation in separate dictionaries, with the keys given by each label. This change in how information was stored vastly simplified the creation and search process, as it allowed the model to iterate over the class labels and read/write from the dictionaries accordingly.

Another aspect of this project I found very helpful was building the validation techniques used to measure model accuracy. When using sk-learn or other online machine learning

repositories, it is difficult to comprehend how model scores and cross-validation techniques operate due to many levels of abstraction. Writing the code for the confusion matrices, cross-validation, and model accuracy helped me get a significantly more nuanced understanding of how these algorithms work and are implemented in other packages.

6 Related Work

Although the versions of Naïve Bayes implemented in this project were simple, academic research has demonstrated that these event models can still have many practical, real-world applications. An abundance of research has been conducted over Naïve Bayes, analyzing the problem instances where the models perform best and discovering new ways to fine-tune the algorithm to generate more accurate results. Researchers at Politeknik Pos Indonesia evaluated the multinomial event model's effectiveness in classifying user intent in online chatbot software services (Setyawan, Awangga, & Efendi, 2018). Similar to the sentiment classification problem, this paper examines how Naïve Bayes performs in natural language processing tasks, extrapolating the results to produce chatbots capable of interacting with human users on the software. Applications of this chatbot could be very beneficial, helping automate parts of customer service operations and fulfilling user requests efficiently.

In another paper published at Gazi University in Turkey, the authors demonstrated that the Naïve Bayes classifier could predict daily total photovoltaic energy generation by solar power with high accuracy (Bayindir, Yesilbudak, Colak, & Genc, 2017). Previous research on this topic utilized far more advanced machine learning models for a similar purpose, applying versions of ensemble learning (including random forests and gradient-boosted trees), convolutional neural networks, and support vector machines. The author's findings showed that the Naïve Bayes classifier exemplified robust prediction capabilities despite being a far simpler model. The classifier performed nearly as well as the other machine learning algorithms used in previous research while requiring substantially less computational resources.

Although these previous examples utilized the versions of Naïve Bayes studied in this paper, other studies aim to find new ways of improving the classifier. A paper published in the Applied Soft Computing Journal attempts to remedy one of the issues with assuming conditional feature independence when applying Naïve Bayes (Arar & Ayan, 2017). The authors created a "Feature Dependent Naïve Bayes" (FDNB) classification model, which includes all features as pairs to simulate dependence. When performing calculations, the FDNB classifier determines feature probabilities by examining the instances where both feature pairs are present in the dataset. Applying this model to a software defect prediction problem, FDNB outperformed traditional Naïve Bayes classifiers and displayed competitive performance with other popular feature-weighting procedures.

7 Conclusions

Two versions of the Naïve Bayes classifier were implemented in this paper and tested on text classification and digit recognition datasets. The multivariate Bernoulli and multinomial event models displayed very similar classification accuracy for the review sentiment problem.

Both models classified reviews with around 80% accuracy, demonstrating good performance for this task. Although the multinomial model is generally accepted to perform better in text classification, the dataset utilized may not have allowed the model to perform optimally due to limited word repetitions in the reviews.

In contrast to the first experiment, the models exhibited a higher disparity in accuracy when tested on the digit recognition dataset. The multinomial model significantly outperformed the Bernoulli, scoring almost 15% higher in classification accuracy. The difference in probability parameter calculations produced this discrepancy in results, revealing the comparative advantage of the multinomial model in this instance. Despite Naïve Bayes algorithms not performing quite as well as more advanced learning methods, one of the biggest strengths of the classifier is its ability to generate predictions in a computationally efficient manner. For this reason, Naïve Bayes is considered the 'model of first resort' when approaching classification problems.

References

Arar, Ö. F., & Ayan, K. (2017). A feature dependent naive bayes approach and its application to the software defect prediction problem. *Applied Soft Computing*, *59*, 197–209.

Bayindir, R., Yesilbudak, M., Colak, M., & Genc, N. (2017). A novel application of naive bayes classifier in photovoltaic energy prediction. In *2017 16th ieee international conference on machine learning and applications (icmla)* (pp. 523–527).

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, *29*(6), 141–142.

Kotzias, D., Denil, M., De Freitas, N., & Smyth, P. (2015). From group to individual labels using deep features. In *Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining* (pp. 597–606).

Setyawan, M. Y. H., Awangga, R. M., & Efendi, S. R. (2018). Comparison of multinomial naive bayes algorithm and logistic regression for intent classification in chatbot. In *2018 international conference on applied engineering (icae)* (pp. 1–5).