# Supervised Machine Learning: The Naïve Bayes Classifier Applied in the Multivariate Bernoulli Event and Multinomial Event Models in Image and Text Data

**Jayson C. Garrison**

**The University of Tulsa**
**Tandy School of Computer Science**
**CS-5333-01 (22/SP)**
**January 31, 2022**

https://github.com/jayson-garrison/ML-Naive-Bayes

### Abstract

Object classification problems occur frequently and are complex problems that often involve a large amount of features. Machine learning algorithms are used to effectively handle classification problems and are highly effective. One basic supervised machine learning technique is the Naïve Bayes classifier that is applied in the multi-variate Bernoulli event and multinomial models. The two different models are, in general, improved by Laplace smoothing and perform differently based on the importance of repeat feature occurrences. The efficiency of the simple NB classification approach is verified by testing the model on two different data sets involving images and text. Furthermore, the adaptation of the NB classifier into larger fields involving sensitive data reinforces the usefulness of related machine learning algorithms. Despite its direct statistical feature and label association, the Naïve Bayes classifier, due to its validated training and test accuracy in image and text data, stands as the base line for machine learning algorithms when considering more advanced methods.

## 1 Introduction

Classifying objects in the field of computer science, in general, is a difficult task. However if large data sets consisting of similar objects to classify are available, a predictive model can be obtained using various techniques. As the problem of object classification became increasingly relevant in computer applications, developments in the field of **Machine Learning (ML)**, a sub-field of **Artificial Intelligence (AI)**, allowed for the construction of **Machine Learning Algorithms** that efficiently predict, categorize, and classify distinct objects based on different features and their association to classification labels.
One method to solve the object classification problem is to use **Supervised Machine Learning** which constructs accurate predictive models given a data set with their associated

labels. In this approach of machine learning, features are correlated with their associated probabilities to the labels in the training phase. In data were classifications are known given a set of features, such as diagnosis pertaining to different medical conditions, image recognition, and mail sorting, the supervised machine learning approach effectively produces a statistically reliable model produced by a function on a training data set.

# 2 Related Work

With new applications of computer algorithms in fields that often require the solution to object classification problems such as medicine, defense contracting, transportation, and retail, machine learning techniques drastically increase safety, consumer products, and health due to the use of machine learning to solve critical problems. Furthermore, as the field of machine learning expands, new learning techniques are implemented to further increase model accuracy and reliability. The **Naïve Bayes (NB) classifier** method is one simple but effective example of how supervised machine learning algorithms proficiently solve complex problems.

Often data that are analyzed and learned using ML techniques are sensitive. In other words, the data set, such as medical records, security footage, and financial information, that predictive models are trained on cannot be accessible to the end user. This problem concerns supervised ML models such as the NB classifier. Efforts to secure distributed classifier information in models [1] such as the NB classifier allows for the safe use of predictive algorithms on sensitive data. Secure training data used by ML algorithms allows for cancer diagnosis based on important biomarkers [2] as described by Wu *et al.* Allowing effective ML programs to utilize large sets of important data further increases the relevance of using models such as the NB classifier to solve classification problems.

# 3 Model Explanation

The Naïve Bayes classification algorithm requires a large mathematical assumption fittingly named the **Naïve Bayes assumption**. The algorithm therefore assumes that all related conditional probabilities associated with feature occurrence given a certain label are independent of one another. This yields in a simple yet effective classifier routine that has different model adaptations and interpretations. Two well known models often used with the NB classifier are the **multi-variate Bernoulli event model** and the **multinomial event model**. This increases the adaptability of the NB classification approach as data sets often differ, such as whether or not a feature can occur more than once, and require different considerations in constructing a predictive function.

## 3.1 Naïve Bayes Classification

The Naïve Bayes classifier is a probabilistic supervised learning method and utilizes a posterior probability table to ascertain the correct labels given an object's set of features. Using the classification equation by Dr. Sen [3], defining a hypothesis function as $h$ and a data set

$D$ with $n$ data points the NB classifier is illustrated by:

$$h = \text{argmax}_h P(h|D)$$

Which can be expanded by the application of Bayes' rule and eliminating terms independent of $h$ to obtain:

$$h = \text{argmax}_h P(D|h)P(h)$$

Using the Naïve Bayes assumption and applying a monotonic function, such as the natural logarithm, to prevent underflow error the classifier is shown by:

$$h = \text{argmax}_h \left[ \ln(P(h)) + \sum_{i=1}^{n} \ln(P(D_i|h)) \right]$$

With this classifier, labels and features are treated as random variables and are represented by two different posterior probability calculations that best fit a given data set.

## 3.2 Multi-variate Bernoulli Event Model

If a data set were to have a limited vocabulary size, such as a boolean value, then the multi-variate Bernoulli event model would yield a solid posterior probability calculation. Based on Andrew Ng's explanation [4] a function $B$ is defined for a classification $y_c$ and feature $x_f$:

$$B_{y_c,x_f} = \frac{\sum_{i=1}^{m} 1\{x_f = 1 \wedge y = y_c\}}{\sum_{i=1}^{m} 1\{y = y_c\}}$$

With this calculation, the posterior table is constructed without the consideration of repeat feature occurrences known as the "bag of words" approach in natural language processing problems. Due to this generalization, depending on the data set, some important information is lost if repeat features are particularly significant. Despite this possible issue, data sets without repeats or where they are of little to no importance regarding classification association, such as features based on pixel locations in an image, this calculation works well with the NB classifier.

## 3.3 Multinomial Event Model

If, on the other hand, repeat feature occurrences were present and significant, the NB classifier model is still effective with a modification to the posterior probability calculations. The multinomial event model considers repeat feature occurrences and is accurate for data sets with a large vocabulary. Also based on Ng's interpretation [4], $M$ is defined for the classification $y_c$ and feature $x_f$:

$$M_{y_c,x_f} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n_i} 1\{x = x_f \wedge y = y_c\}}{\sum_{i=1}^{m} 1\{y = y_c\} n_i}$$

Now, with this new interpretation of the posterior table calculation, the denominator reflects the number of feature occurrences in any given classification which is not considered in the multi-variate calculation. Furthermore, within the programmatic implementation of this model, the numerator also differs as the features are now calculated probabilistically with respect to number repetitions. These new attributes in the multinomial model allow the NB classifier to predict labels in data were multiple features are present more than once, such as in email classification.

### 3.4 Use of Laplace Smoothing

Often when using the NB classification in the held-out data set there are unseen features. When evaluated, using both multi-variate and multinomial models, the result is a zero probability or an undefined divide by zero calculation. To prevent this, an additive smoothing technique known as Laplace smoothing eliminates the possibility of a divide by zero error by assuming each feature is seen an additional $k$ times. Modifying the multi-variate and multinomial models respectively we achieve:

$$B_{y_c,x_f} = \frac{\sum_{i=1}^{m} 1\{x_f = 1 \land y = y_c\} + k}{\sum_{i=1}^{m} 1\{y = y_c\} + k|V|}$$

and,

$$M_{y_c,x_f} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n_i} 1\{x = x_f \land y = y_c\} + k}{\sum_{i=1}^{m} 1\{y = y_c\}n_i + k|V|}$$

The simple addition of the parameter $k$ and dividing by the vocabulary represented by $|V|$ ”smooths” the two adaptations of the model.

## 4 Data Sets Used

In order to analyze the efficiency of the NB classification using the multi-variate Bernoulli event model and multinomial model, data sets from ML repositories were used to analyze and validate the effects of the two different adaptations. In order to compare the different models two data sets were used with different sizes of labels, features, and data points. Furthermore, the data sets used were sizeable in order to obtain a comprehensive model to predict correct labels.

### 4.1 Image Data

The first data set was a partition of the MNIST repository [5] and consisted of 42,000 28 pixel by 28 pixel images of handwritten digits with ten associated classification labels. The data were booleanized to create a vocabulary length of 2 representing pixel presence and organized into a composite list of row vectors. Letting $P$ denote the data set of images:

$$P = \begin{bmatrix} \begin{bmatrix} 0 & \dots & 0 & \dots & 1 & \dots & 0 & \dots & 0 \end{bmatrix} \\ \vdots \\ \begin{bmatrix} 0 & \dots & 0 & \dots & 1 & \dots & 1 & \dots & 0 \end{bmatrix} \\ \vdots \\ \begin{bmatrix} 0 & \dots & 1 & \dots & 1 & \dots & 0 & \dots & 0 \end{bmatrix} \end{bmatrix}$$

Where,

$$|P_i| = \text{number of features} = 784$$

$$|P| = \text{number of images} = 42,000$$

In this example, image features were defined to be the positions of the pixels independent of each other instead of pixel intensities. Despite the number of features increasing with the resolution of the image data set, the definition of features as each individual pixel, on or off in our booleanized data set, generalizes well with sizeable data sets. On the other hand, pixel intensities, often resulting in a feature size of 256, still produce accurate classification statistics based on the multi-variate Bernoulli adaptation of the NB classification algorithm. Despite this accuracy with features defined as pixel intensity, under fitting may result if the number of pixel intensity levels are sufficiently small compared to the number of labels. However, when using pixel positional features, if significantly large images are classified with this approach, the model might lead to over fitting due to the potentially large amount of features based on this analysis. Both methods of defining features in image data have their advantages and drawbacks.

### 4.2 Word Based Data Set

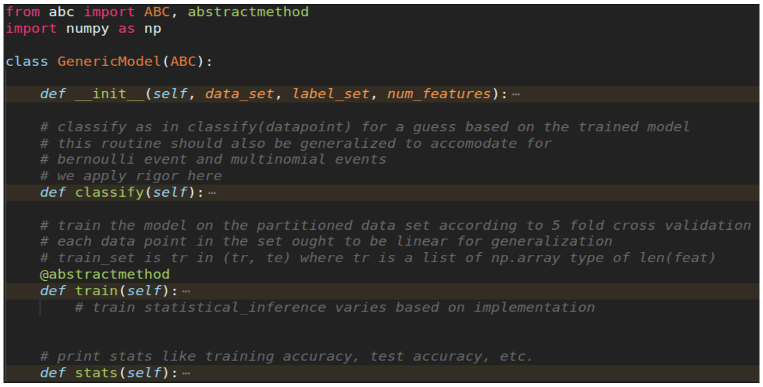
The second data set used consisted of emails consisting of English words that were vectorized [6] where the size of the vocabulary was uniform across all email data points resulting in the length of the vocabulary and length of features for each email to be equal. The organization of the word based data into vectors allowed for this data set to be used by the same programmatic implementation as the image data set streamlining the data parsing phase and enabled methods to be used across both sets. Letting $E$ denote the data set of emails then:

$$|E_i| = \text{number of features} = 3001$$

$$|E| = \text{number of emails} = 5170$$

## 5 Implementation

To determine the effectiveness of the NB classification algorithm, the multi-variate Bernoulli event model was implemented to predict labels in the image data set and both the multi-variate and multinomial models were used to predict labels on the word data set. Due to the array organization of the data and vector mathematics, the models were implemented in Python and NumPy was used to efficiently perform array operations. A generic model was implemented that included common functionalities with abstract methods allowing for two different sub-classes to provide different implementations for the `train()` method reflective of the two different posterior table calculations.



```
from abc import ABC, abstractmethod
import numpy as np

class GenericModel(ABC):

    def __init__(self, data_set, label_set, num_features):⋯

    # classify as in classify(datapoint) for a guess based on the trained model
    # this routine should also be generalized to accomodate for
    # bernoulli event and multinomial events
    # we apply rigor here
    def classify(self):⋯

    # train the model on the partitioned data set according to 5 fold cross validation
    # each data point in the set ought to be linear for generalization
    # train_set is tr in (tr, te) where tr is a list of np.array type of len(feat)
    @abstractmethod
    def train(self):⋯
        # train statistical_inference varies based on implementation


    # print stats like training accuracy, test accuracy, etc.
    def stats(self):⋯
```

Figure 1: Generic model implementation where multi-variate and multinomial models provide their own calculations in `train()`.

## 5.1 Algorithms

In machine learning, several different algorithms are used to construct a model. In the NB classification supervised ML technique, the `train()` method is responsible for building the predictive model and the `classify()` method executes classification on a new object using the model built by the `train()` algorithm. There are other algorithms used in the `driver` file but are used for data processing and are, in general, not important for analyzing the effectiveness of the NB classification technique for object classification.

### 5.1.1 Training

The NB classification algorithms requires the construction of a posterior probability table to make predictions. The `train()` method, using either the multi-variate or multinomial approach, calculates the associated conditional probabilities.

**Algorithm 1:** Training algorithm for NB classification

```
Data: data_set
def Train(training_set, p_table):
    class_occurrences ← class occurrences indexed by label
    feature_instances ← vector of feature instances indexed by label
    for data_point in training_set do
        class_occurrences[label] += 1
        feature_instances[label] += data_point
    end
    p_table ← construct posterior table according to the two models)
```

The training routine is much faster than the classification complexity due to the vector addition provided by NumPy. $\mathcal{O}(n) + c \times \mathcal{O}(1)$ describes the complexity of the `train()` method where $c$ is defined by $f \times l$ such that $f$ is the number of features and $l$ is the number of labels. Since the portion of the algorithm that depends on the size of the data set does not have a multiplier of $f$ or $l$, the training method is, in this implementation, always faster than the classification method.

### 5.1.2 Classification

The NB classification routine is utilized by both models and is illustrated by the following algorithm:

**Algorithm 2:** Classification algorithm for NB classification

```
Data: data_set
def Classify(data_set, p_table):
    correct_guess ← 0
    possibilities ← dictionary with keys of labels initialized to null
    for data_point in data_set do
        possibilities ← initialize with $\ln P(h)$
        for feature in data_point do
            for label in possibility do
                if data_point[feature] = 0 then
                    possibilities[label] += 1 - p_table[label][feature]
                else
                    possibilities[label] += p_table[label][feature]
            end
        end
        most_likely
        if most_likely = actual then
            correct_guess += 1
    end
    accuracy = correct_guess / data_set.length()
```

The algorithmic complexity of the classification routine is $c \times \mathcal{O}(n)$ where $c$ is defined by $f \times l$ as in the training method. In the data sets used the number of labels, $l$, is never sufficiently close to $n$, the number of data points in the data set. However, sometimes $f$ can be sufficiently close to $n$ and thus yields in a worst case complexity of $c \times \mathcal{O}(n^2)$ where $c$ is only defined by $l$. One example where the worst case complexity is reached is in word based data represented in row vectors where the number of features, often defined by a vocabulary consisting of all possible words such that each data point has the same number of *possible* features, are close to the number of data points. In our word based example, the number of features is only about half the size of the number of email samples so the best case complexity of $c \times \mathcal{O}(n)$ is preserved.

# 6 Results and Analysis

Using the two adaptations of the NB classification algorithm the two data sets were used to evaluate the training and testing accuracies. To verify that the training and test accuracies were not statistical anomalies, the two data sets were shuffled and partitioned using five fold cross validation to define the training set and test set on different sectors of the data. In addition to altering the training and test sets, different values of $k$ were used to determine the effect of Laplace smoothing on classification accuracy.

## 6.1 Model Performance on Image Data

Using the multi-variate model algorithm combined with the NB classifier, the results displayed the accuracies over different values of $k$. Based on this data, the addition of $k$ does not positively affect the accuracy after $k = 1$. Interestingly, the model does work with a $k$ value of 0 which should yield in a run time error due to a divide by zero computation. However, due to exception handling and relative numeric comparison provided by NumPy, the results were still obtained for $k = 0$. Therefore, $k = 0$ yields the best average accuracy based on this implementation. If NumPy were not used, $k = 0$ would not work resulting in a $k$ value of 1 as the most optimal value of $k$. Furthermore, the multi-variate model proves its robustness based on the average training and testing accuracies only differing over at most 2% for large values of $k$.

| k | Train%, Test% | | | | | Test%, Train% Average |
|---|---|---|---|---|---|---:|
| 0 | 83.7, 83.4 | 83.8, 83.4 | 83.6, 83.0 | 83.8, 83.2 | 83.7, 83.4 | **83.8, 83.3** |
| 1 | 83.4, 83.4 | 83.5, 83.3 | 83.3, 83.2 | 83.5, 83.3 | 83.4, 83.4 | **83.4, 83.3** |
| 2 | 83.4, 83.4 | 83.5, 83.2 | 83.3, 83.1 | 83.5, 83.3 | 83.3, 83.3 | **83.4. 83.3** |
| 50 | 82.3, 83.2 | 82.5, 81.6 | 82.5, 82.1 | 82.7, 82.4 | 82.5, 83.0 | **82.5, 82.4** |
| 100 | 81.8, 82.3 | 82.1, 81.7 | 81.9, 81.5 | 81.9, 81.8 | 81.9, 81.7 | **81.9, 81.8** |

Figure 2: Multi-variate Bernoulli event model performance with different values of k using five fold cross validation for image classification.

## 6.2 Model Performance on Text Data

Using the multi-variate and multinomial model algorithm combined with the NB classifier performed strongly with the best value of $k$ at 93.2% and 97.0% testing accuracy respectively.

| k | Train%, Test% | | | | | Test%, Train% Average |
|---|---|---|---|---|---|---:|
| 0 | 93.0, 90.9 | 93.0, 91.4 | 93.1, 90.2 | 93.3, 91.1 | 93.7, 90.4 | **93.2, 90.8** |
| 1 | 87.2, 88.1 | 87.3, 87.6 | 87.6, 86.6 | 87.2, 85.3 | 88.1, 88.0 | **87.5, 87.1** |
| 2 | 86.7, 87.3 | 86.7, 87.2 | 87.1, 86.0 | 86.9, 84.6 | 87.5, 87.5 | **87.0, 86.5** |
| 50 | 75.1, 75.2 | 75.1, 75.3 | 75.0, 75.4 | 75.1, 75.8 | 75.5, 73.3 | **75.2, 75.0** |
| 100 | 73.7, 72.7 | 73.7, 72.5 | 73.1, 75.2 | 73.7, 72.6 | 73.4, 74.1 | **73.5, 73.4** |

Figure 3: Multi-variate Bernoulli event model performance with different values of k using five fold cross validation for email classification.

The affects of increasing values of $k$ are more evident in the email data set. The accuracy of the multinomial model is better than the multi-variate due to the significance of repeat feature occurrences. The analysis of $k = 0$ still holds as NumPy is used across all implementations. However, the multinomial model is more robust than the multi-variate in text data as k increases due to the strength of the associate between repeat features and labels.

| k | Train%, Test% | | | | | Test%, Train% Average |
|---|---|---|---|---|---|---|
| 0 | 97.0, 94.2 | 96.9, 95.2 | 96.9, 94.5 | 97.0, 94.9 | 97.2, 94.2 | **97.0, 94.6** |
| 1 | 95.6, 95.3 | 95.7, 95.0 | 95.5, 95.6 | 95.6, 95.1 | 95.6, 95.4 | **95.6, 95.3** |
| 2 | 95.4, 95.2 | 95.5, 94.7 | 95.3, 95.6 | 95.4, 94.8 | 95.5, 95.3 | **95.5, 95.3** |
| 50 | 92.4, 90.8 | 92.6, 91.7 | 91.9, 93.5 | 91.2, 93.1 | 92.4, 91.6 | **92.1, 92.1** |
| 100 | 90.0, 90.2 | 90.2, 90.4 | 90.4, 90.6 | 90.8, 89.3 | 90.8, 90.9 | **90.5, 90.2** |

Figure 4: Multinomial event model performance with different values of k using five fold cross validation for email classification.

## 6.3 Comparing Models

Both the multi-variate and multinomial models, in this implementation, yielded acceptable accuracies. Though the multinomial model was only used on one data set, the increased accuracy compared to the multi-variate counterpart suggests that, in the image data set, the multinomial model out performs the multi-variate in cases where repeat features are especially significant. Between the two different data sets, the image data set yielded an 83.8% testing accuracy which was worse than both model implementations for the text data. The image data set used 5 times as many labels and, due to this, was harder to classify. In addition to this, the emails had a larger count of possible features and combined with a label size of 2 was easier to classify. In both data sets the NB classification algorithm proved over 80% accurate and thus supports the reason to use the NB classifier to compare to other ML techniques.

Using five fold cross validation was useful in this implementation since, for both data sets and models, the accuracies were consistent based on the different partitions. The multinomial model used on the email data set required the validation method since in cases where $k = 0, 1, 2$ produced accuracies that were over 95%. Moreover, five fold cross validation also proved useful in supporting the deterioration of the model accuracy as the values of $k$ increased suggesting that there is an optimal value of $k$.

## 6.4 Effects of Laplace Smoothing

Varying the value of $k$, in this implementation, always decreased the accuracy in both models. In the absence of NumPy, $k = 1$ offers the optimal accuracy for both adaptations of the NB classification. The decrease in accuracy is due to the strength of the prior, $k$, inflating the posterior probability calculation removing strong feature indicators in aspects of an object that are significant. In other words, as $k$ becomes large, the probabilities in the posterior calculation become uniform with one another, generalizing the prediction and reducing the accuracy. Despite the drawbacks of large values of $k$, the small modification of Laplace smoothing, for a value of $k = 1$, greatly improves the model and is in most cases required due to division by zero caused by unseen features.

# 7 Conclusion

Machine learning algorithms are extremely efficient at solving object classification problems. Among machine learning techniques is the Naïve Bayes classification which is an accurate model where other techniques are often compared. The NB classification relies on a strong independent event assumption, fittingly called naïve, and still yields statistically accurate training and testing accuracies verified with five fold cross validation. Combined with the effectiveness and recent advancements to apply machine learning methods, such as the NB classifier approach, to sensitive data while maintaining security furthers the application of such methods in areas where difficult classification problems occur.

By using two data sets comprised of images and text, the NB classification performance proves useful in solving related problems. Furthermore, the simplistic technique of the NB classification can be adapted in two different posterior probability calculations, the multivariate Bernoulli event and multinomial event model. Based on different data sets, particularly whether or not repeat features are significant, the NB classification routine performs well on different problem domains. Is is due to this algorithmically efficient and simplistic probabilistic approach that the NB algorithm is used in several problem spaces and compared against more advances machine learning implementations to evaluate their effectiveness.

# References

[1] J. Vaidya, M. Kantarcıoğlu, and C. Clifton, "Privacy-preserving naïve bayes classification," *The VLDB Journal*, vol. 17, no. 4, p. 879–898, jul 2008. [Online]. Available: https://doi.org/10.1007/s00778-006-0041-y

[2] M.-Y. Wu, D.-Q. Dai, Y. Shi, H. Yan, and X.-F. Zhang, "Biomarker identification and cancer classification based on microarray data using laplace naive bayes model with mean shrinkage," *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, vol. 9, no. 6, p. 1649–1662, nov 2012. [Online]. Available: https://doi.org/10.1109/TCBB.2012.105

[3] S. Sen, "Supervised learning: Naïve bayes," Available in TU Harvey course content.

[4] A. Ng, "Cs229 lecture notes: Generative learning algorithms," Available at https://see.stanford.edu/materials/aimlcs229/cs229-notes2.pdf.

[5] "Digit recognizer." [Online]. Available: https://www.kaggle.com/c/digit-recognizer/data?select=train.csv

[6] B. Biswas, "Email spam classification dataset csv," Mar 2020. [Online]. Available: https://www.kaggle.com/balaka18/email-spam-classification-dataset-csv