As mentioned, different Naïve Bayes (NB) methods have different assumptions regarding the distribution of P(xi|y). In this regard, BernouliNB considers the data to follow a multi-variate Bernoulli distribution. Regardless of the number of features, each is a binary variable. Any other kind of data is made into a boolean variable using the binarize parameter. The decision rule in this case is (write eqn.). In case the input data is in the form of text, word frequency, instead of word count, could be employed when this classifier is being trained and, subsequently, used.

In case the data is multinomially distributed, MultinomialNB could be implemented. It is classically used with text input data represented by word count vectors.unlike BernoulliNB, which penalizes a non-occuring feature, MultinomialNB merely ignores such features. To parameterize the distribution, vectors theta_y = (\theta_{y1},\ldots,\theta_{yn}) are considered for each class y, where n is the number of considered features (for example, the size of the vocabulary in text-input data) and theta-y-I is the probability of having a feature i in a sample of class y. In this context, a smoothened version of maximum likelihood, like counting relative frequencies, could be used to estimate the parameters theta-y (write eqn). Where Ny is the count of all features in a class y and Nyi is the frequency of a given feature i in a sample belonging to class y in the training dataset T.

As mentioned, MultinomialNB ignores non-occuring features and this, the smoothing priors alpha >/= 0 removes zero probabilities from further computations. Laplace smoothing sets alpha to 1 whereas Lidstone smoothing takes alpha to be > 1.

Finally, GuassianNB is suited to classifications where the feature likelihood (Pxi|y) follows a Guassian distribution. Under this assumption (write eqn). Where maximum likelihood is used to give an estimate to each of the parameters (sigma and mu)

[**BernoulliNB**](http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html#sklearn.naive_bayes.BernoulliNB) implements the naive Bayes training and classification algorithms for data that is distributed according to multivariate Bernoulli distributions; i.e., there may be multiple features but each one is assumed to be a binary-valued (Bernoulli, boolean) variable. Therefore, this class requires samples to be represented as binary-valued feature vectors; if handed any other kind of data, a BernoulliNB instance may binarize its input (depending on the binarize parameter).

The decision rule for Bernoulli naive Bayes is based on

(x_i \mid y) = P(i \mid y) x_i + (1 - P(i \mid y)) (1 - x_i)

which differs from multinomial NB’s rule in that it explicitly penalizes the non-occurrence of a feature http://scikit-learn.org/stable/_images/math/df0deb143e5ac127f00bd248ee8001ecae572adc.png that is an indicator for class http://scikit-learn.org/stable/_images/math/276f7e256cbddeb81eee42e1efc348f3cb4ab5f8.png, where the multinomial variant would simply ignore a non-occurring feature.

In the case of text classification, word occurrence vectors (rather than word count vectors) may be used to train and use this classifier. BernoulliNB might perform better on some datasets, especially those with shorter documents. It is advisable to evaluate both models, if time permits.

[**MultinomialNB**](http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html#sklearn.naive_bayes.MultinomialNB) implements the naive Bayes algorithm for multinomially distributed data, and is one of the two classic naive Bayes variants used in text classification (where the data are typically represented as word vector counts, although tf-idf vectors are also known to work well in practice). The distribution is parametrized by vectors theta_y = (\theta_{y1},\ldots,\theta_{yn}) for each class http://scikit-learn.org/stable/_images/math/276f7e256cbddeb81eee42e1efc348f3cb4ab5f8.png, where http://scikit-learn.org/stable/_images/math/e11f2701c4a39c7fe543a6c4150b421d50f1c159.png is the number of features (in text classification, the size of the vocabulary) and theta_{yi} is the probability (x_i \mid y) of feature http://scikit-learn.org/stable/_images/math/df0deb143e5ac127f00bd248ee8001ecae572adc.png appearing in a sample belonging to class http://scikit-learn.org/stable/_images/math/276f7e256cbddeb81eee42e1efc348f3cb4ab5f8.png.

The parameters theta_y is estimated by a smoothed version of maximum likelihood, i.e. relative frequency counting:

hat{\theta}_{yi} = \frac{ N_{yi} + \alpha}{N_y + \alpha n}

where _{yi} = \sum_{x \in T} x_i is the number of times feature http://scikit-learn.org/stable/_images/math/df0deb143e5ac127f00bd248ee8001ecae572adc.png appears in a sample of class http://scikit-learn.org/stable/_images/math/276f7e256cbddeb81eee42e1efc348f3cb4ab5f8.png in the training set http://scikit-learn.org/stable/_images/math/f2d283a2071f9d043c9e0b0f794a8880fa0d3ce9.png, and _{y} = \sum_{i=1}^{|T|} N_{yi} is the total count of all features for class http://scikit-learn.org/stable/_images/math/276f7e256cbddeb81eee42e1efc348f3cb4ab5f8.png.

The smoothing priors alpha \ge 0 accounts for features not present in the learning samples and prevents zero probabilities in further computations. Setting alpha = 1 is called Laplace smoothing, while alpha < 1 is called Lidstone smoothing.

[**GaussianNB**](http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html#sklearn.naive_bayes.GaussianNB) implements the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian:

(x_i \mid y) &= \frac{1}{\sqrt{2\pi\sigma^2_y}} \exp\left(-\frac{(x_i - \mu_y)^2

The parameters sigma_y and mu_y are estimated using maximum likelihood.