Naive Bayes

A set of supervised learning algorithms based on applying Bayes theorem with the 'naive' assumption of independence between every pair of features.

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) P(x_1, \dots x_n \mid y)}{P(x_1, \dots, x_n)}$$

- Works well in document classification and spam filtering
- Require a small amount of training data to estimate the necessary parameters
- Fast
- Good classifier, bad estimator

Gaussian Naive Bayes

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

```
>>> from sklearn import datasets
>>> iris = datasets.load_iris()
>>> from sklearn.naive_bayes import Gaus-
sianNB
>>> gnb = GaussianNB()
>>> y_pred = gnb.fit(iris.data,
iris.target).predict(iris.data)
>>> print("Number of mislabeled points out
of a total %d points :
...      %d" % (iris.data.shape[0],
(iris.target != y_pred).sum()))
Number of mislabeled points out of a total
150 points : 6
```

Multinomial Naive Bayes

`MultinomialNB` implements the naive Bayes algorithm for multinomially distributed data.

- Used in text classification
- Used for discrete counts

Bernoulli Naive Bayes

`BernoulliNM` implements the naive Bayes training and classification algorithms for data that is distributed according to multivairate Bernoulli distributions

- May be multiple features but each one is assumed to be a binary-valued (Bernoulli, boolean) variable
- Requires samples to be represented as binary-valued feature vectors

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

Differs from multinomial NB's rule:

- Explicitly penalizes the non-occurrence of a feature *i* that is an indicator for class *y*
- Multinomial variant would simply ignore a non-occurring feature
- Perform better on datasets with shorter documents
- Evaluate both models if time permits

Out-of-core Naive Bayes Model Fitting

`MultinomialNB`, `BernoulliNB`, and `GaussianNB` expose a `partial_fit` method that can be used incrementally as done with other classifiers.

All naive Bayes classifiers support *sample weighting*.

Contrary to the `fit` method, the first call to `partial_fit` needs to be passed the list of all the expected class labels.

Calculating Accuracy Score

```
def NBAccuracy(features_train, labels_-
train, features_test, labels_test):
    from sklearn.naive_bayes import Gaus-
sianNB
    from sklearn.metrics import accura-
cy_score

    clf = GaussianNB()
    clf.fit(features_train, labels_train)

    pred = clf.predict(features_test)

    accuracy = accuracy_score(labels_test,
pred)
    return accuracy
```

Bayes Rule (also called Bayes Theorem)

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}$$

Why Naive Bayes naive?

Naive bayes ignores words order, and it just cares about word frequency.