ML ASSIGNMENT 03

PRAKRUTHI B GOPAL

21BDA25

* Write the difference between the following:Gaussian Naive Bayes,Multinomial Naive Bayes, Complement Naive Bayes,Bernoulli Naive Bayes, Categorical Naive Bayes, Out-of-core naive Bayes model fitting

[**Gaussian Naive Bayes**](https://iq.opengenus.org/gaussian-naive-bayes/)

It is a variant of [Naive Bayes](https://iq.opengenus.org/text-classification-naive-bayes/) that follows Gaussian normal distribution and supports continuous data. When working with continuous data, an assumption often taken is that the continuous values associated with each class are distributed according to a normal (or Gaussian) distribution. Gaussian Naive Bayes supports continuous valued features and models each as conforming to a Gaussian (normal) distribution. An approach to create a simple model is to assume that the data is described by a Gaussian distribution with no co-variance (independent dimensions) between dimensions. This model can be fit by simply finding the mean and standard deviation of the points within each label, which is all what is needed to define such a distribution.

**Multinomial Naive Bayes**

The Multinomial Naive Bayes algorithm is a Bayesian learning approach popular in Natural Language Processing (NLP). The program guesses the tag of a text, such as an email or a newspaper story, using the Bayes theorem. It calculates each tag's likelihood for a given sample and outputs the tag with the greatest chance.

**Complement Naive Bayes**

Complement Naive Bayes is particularly suited to work with imbalanced datasets. In complement Naive Bayes, instead of calculating the probability of an item belonging to a certain class, we calculate the probability of the item belonging to all the classes. This is the literal meaning of the word, **complement**and hence is called Complement Naive Bayes.

**Bernoulli Naive Bayes**

Bernoulli Naïve Bayes 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.

**Categorical Naive Bayes**

The categorical Naive Bayes classifier is suitable for classification with discrete features that are categorically distributed. The categories of each feature are drawn from a categorical distribution.

**Out-of-core naive Bayes model fitting**

Naive Bayes models can be used to tackle large scale classification problems for which the full training set might not fit in memory. To handle this case, [Multinomial NB](https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html#sklearn.naive_bayes.MultinomialNB), [Bernoulli NB](https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html#sklearn.naive_bayes.BernoulliNB), and [Gaussian NB](https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html#sklearn.naive_bayes.GaussianNB) expose a partial\_fit method that can be used incrementally as done with other classifiers as demonstrated in [Out-of-core classification of text documents](https://scikit-learn.org/stable/auto_examples/applications/plot_out_of_core_classification.html#sphx-glr-auto-examples-applications-plot-out-of-core-classification-py). 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.

* What is Jaccard and Cosine Similarity?

[Jaccard similarity](https://en.wikipedia.org/wiki/Jaccard_index) or intersection over union is defined as **size of intersection divided by size of union of two sets.** In order to calculate similarity using Jaccard similarity, we will first perform **lemmatization**to reduce words to the same root word.

[Cosine similarity](https://en.wikipedia.org/wiki/Cosine_similarity) calculates similarity by measuring **the cosine of angle between two vectors. With cosine similarity, we need to convert sentences into vectors.** One way to do that is to use **bag of words with either TF**(term frequency**) or TF-IDF** (term frequency- inverse document frequency). The choice of TF or [TF-IDF](https://en.wikipedia.org/wiki/Tf%E2%80%93idf) depends on application and is immaterial to how cosine similarity is actually performed — which just needs vectors.