Question-2

a. Write the difference between the following:

i. Gaussian Naive Bayes,

ii. Multinomial Naive Bayes,

iii. Complement Naive Bayes,

iv. Bernoulli Naive Bayes,

v. Categorical Naive Bayes,

vi. Out-of-core naive Bayes model fitting

Naive Bayes algorithm is one of the well-known supervised classification algorithms. It is based on the Bayes theorem.
Is a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable.

- very fast
- very popular classification algorithm
- good enough for text classification.
- most simple algorithm
- As the name suggests, here this algorithm makes an assumption as all the variables in the dataset are "Naive" i.e not correlated to each other.
- mostly used to get the base accuracy of the dataset.

When to use

- Text Classification
- when dataset is huge
- When you have small training set

Applications

- **Real time Prediction:** Naive Bayes is an eager learning classifier and it is sure fast. Thus, it could be used for making predictions in real time.
- **Multi class Prediction:** This algorithm is also well known for multi class prediction features. Here we can predict the probability of multiple classes of target variables.
- **Text classification/ Spam Filtering/ Sentiment Analysis:** Naive Bayes classifiers mostly used in text classification (due to better result in multi class problems and independence rule) have higher success rate as compared to other algorithms. As a result, it is widely used in Spam

filtering (identify spam email) and Sentiment Analysis (in social media analysis, to identify positive and negative customer sentiments)

- **Recommendation System:** Naive Bayes Classifier and Collaborative Filtering together builds a Recommendation System that uses machine learning and data mining techniques to filter unseen information and predict whether a user would like a given resource or not.

Advantages of Naïve Bayes Classifier:

- Naïve Bayes is one of the fast and easy ML algorithms to predict a class of datasets.
- It can be used for Binary as well as Multi-class Classifications.
- It performs well in Multi-class predictions as compared to the other Algorithms.
- It is the most popular choice for **text classification problems**.

Disadvantages of Naïve Bayes Classifier:

- Naive Bayes assumes that all features are independent or unrelate

Types of Naive Bayes model

Gaussian Naive Bayes

In a Gaussian Naive Bayes, the predictors take a continuous value assuming that it has been sampled from a Gaussian Distribution. It is also called a Normal Distribution

This approach is built on the assumption of a normal distribution of probabilities. It means, that spam and not-spam classes of messages have frequencies of the words from vocabulary distributed by the Gaussian law:

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

Multinomial classification suits best for the discrete values like word counts. So we expect it to show the best accuracy. In this case distribution of probabilities for each event bases on the formula:

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

Complement Naive Bayes

This is basically an adaptation of the multinomial naive bayes that is particularly suited for imbalanced datasets.

This approach is almost the same as the Multinomial, though now we count the occurrences of a word in the complement to the class. For example, for the spam message we will count the repetitions of each word in all the non-spam messages:

$$\hat{\theta}_{\tilde{c}i} = \frac{N_{\tilde{c}i} + \alpha_i}{N_{\tilde{c}} + \alpha}$$

Bernoulli Naive Bayes

This classifier is also analogous to multinomial naive bayes but instead of words, the predictors are Boolean values. The parameters used to predict the class variable accepts only yes or no values, for example, if a word occurs in the text or not.

Bernoulli formula is close to the multinomial one, though the input is the set of boolean values (the word is present in the message or not) instead of the set of frequencies.

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

Categorical Naive Bayes

Categorical Naive Bayes is suitable for the categorical values — if the example has the set of features or not. In our case, it means, that the vocabulary is treated as the set of features, and the occurrence of a word in the message is treated as the matching with the feature. All formulas are the same as for the multinomial approach but with the occurrences instead of repetitions.
Out-of-Core Naive Bayes — Gaussian Naive Bayes — In a Gaussian Naive Bayes, the predictors take a continuous value assuming that it has been sampled from a Gaussian Distribution. It is also called a Normal Distribution.

#
https://towardsdatascience.com/all-about-naive-bayes-8e13cef044cf

b. Define which text preprocessing and text transformation steps did you use for the above.
(Reference:
https://www.kdnuggets.com/2019/04/text-preprocessing-nlp-machine-learning.html)

There are different ways to preprocess your text. Here are some of the approaches that you should know about and I will try to highlight the importance of each.
Lowercasing
Lowercasing ALL your text data, although commonly overlooked, is one of the simplest and most effective form of text preprocessing. It is applicable to most text mining and NLP problems and can help in cases where your dataset is not very large and significantly helps with consistency of expected output.

Stemming
Stemming is the process of reducing inflection in words (e.g. troubled, troubles) to their root form (e.g. trouble). The “root” in this case may not be a real root word, but just a canonical form of the original word.

Lemmatization

Lemmatization on the surface is very similar to stemming, where the goal is to remove inflections and map a word to its root form. The only difference is that, lemmatization tries to do it the proper way. It doesn't just chop things off, it actually transforms words to the actual root. For example, the word "better

c. What is Jaccard and Cosine Similarity

(Reference:https://studymachinelearning.com/jaccard-similarity-text-similarity-metric-in-nlp/**,** https://towardsdatascience.com/calculate-similarity-the-most-relevant-metrics-in-a-nutshell-9a43564f533e **)**

In Natural Language Processing, we often need to estimate text similarity between text documents. There are many text similarity matrices such as **Jaccard Similarity, Cosine similarity,** and **Euclidean Distance** measurement. All these text similarity metrics have different behaviour.

Jaccard Similarity is defined as an intersection of two documents divided by the union of those two documents that refer to the number of common words over a total number of words. Here, we will use the set of words to find the intersection and union of the document.

The mathematical representation of the Jaccard Similarity is:

$$J(doc_1, doc_2) = \frac{doc_1 \cap doc_2}{doc_1 \cup doc_2}$$

That is, the size of the intersection divided by the size of the union of two sets.
Jaccard Similarity is also known as the Jaccard index and Intersection over Union. Jaccard Similarity metric used to determine the similarity between two text documents means how the two text documents close to each other in terms of their context, that is how many common words exist over total words.

Cosine similarity is one of the metrics to measure the text-similarity between two documents irrespective of their size in Natural language Processing. A word is represented into a vector form. The text documents are represented in n-dimensional vector space.

Mathematically, the Cosine similarity metric measures the cosine of the angle between two n-dimensional vectors projected in a multi-dimensional space. The Cosine similarity of two documents will range from 0 to 1. If the Cosine similarity score is 1, it means two vectors have the same orientation. The value closer to 0 indicates that the two documents have less similarity.

The mathematical equation of Cosine similarity between two non-zero vectors is:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}},$$