Rohit Arora

Summary

The provided content is a comprehensive guide on Naive Bayes, covering

Data Science Interview Preparation Series: Naive Bayes

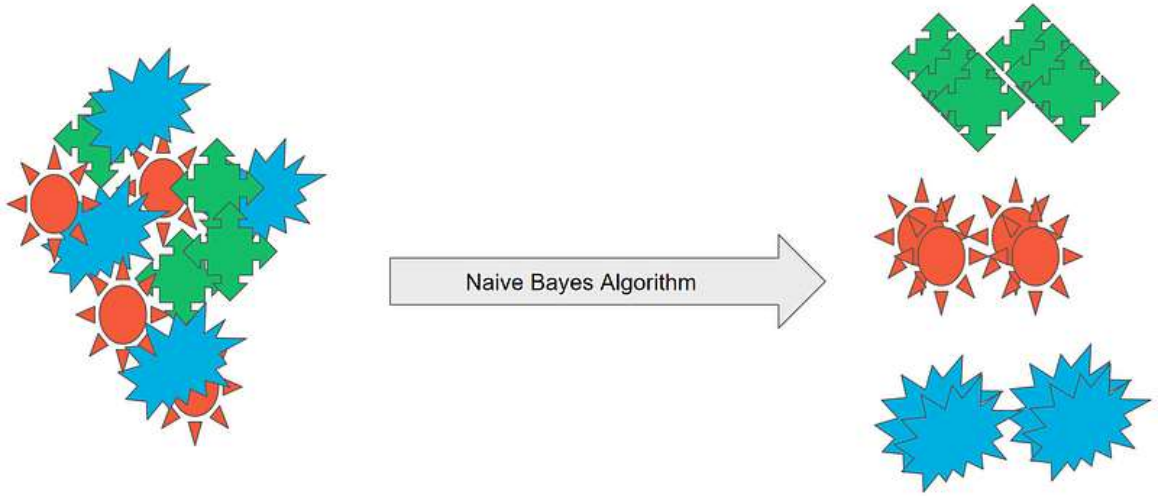
> This is the 2nd part of the Data Science interview preparation series. Here is the link for Part 1.

Taken from Unsplash

This article is structured in a way that is different than most articles. In general, the articles comprise subheadings and a description. In this article, I have accumulated the most fundamental questions that can be asked about Naive Bayes and have answered them in a manner that makes the article exhaustive and self-sufficient. The questions are ordered so that each subsequent question builds upon the previous one, simulating how an interviewer might try to test your knowledge of Naive Bayes. I have also inserted images in between for better understanding and linked references if you further want to feed your curiosity.

What is Naive Bayes? Can you give a brief introduction to the algorithm?

Naive Bayes is a supervised machine learning algorithm that can be trained to classify data into multi-class categories. At the heart of the Naive Bayes algorithm is the probabilistic model that computes the conditional probabilities of the input features and assigns the probability distributions to each possible class.



The image is taken from the Udacity Data Science Nanodegree

You talked about the model being a probabilistic one. Can you elaborate on the mathematical concept behind it and

Let us break the question down into two parts:

- **The mathematical concept used in Naive Bayes, Bayes Theorem**
- **Naive Bayes belongs to the class of models known as Generative Models**

Baye's Theorem:

Naive Bayes is based on the Bayes theorem in statistics. It calculates probabilities independently for each class based on conditions and without conditions and then predicts outcomes. Let's explain it in a bit more detail:

Initially, we start with an event, and this event could be A or B. The probabilities for each are shown as P(A) and P(B). Now, we observe the third event, and that event can either happen or not happen for A and B. R will help us find more exact probabilities for A and B in the following way.

Calculate the probability of R given A (noted as $P(R|A)$), and also the probability of R complement given A (noted as $P(R^c|A)$). And similarly, $P(R|B)$ and $P(R^c|B)$.

The set of scenarios are these four:

- $P(R \cap A)$

- $P(R \cap B)$
- $P(R^c \cap B)$

But since we know R occurred, we know that the second and fourth events are not possible. Our new universe consists of two events, $P(R \cap A)$ and $P(R \cap B)$.

- $P(R \cap A) = P(A)P(R|A)$
- $P(R \cap B) = P(B)P(R|B)$

Because these probabilities do not add to one, we divide them both by their sum so that the new normalized probabilities now do add to one.

Normalized probabilities:

Thus, we get formulas for $P(A|R)$ and $P(B|R)$

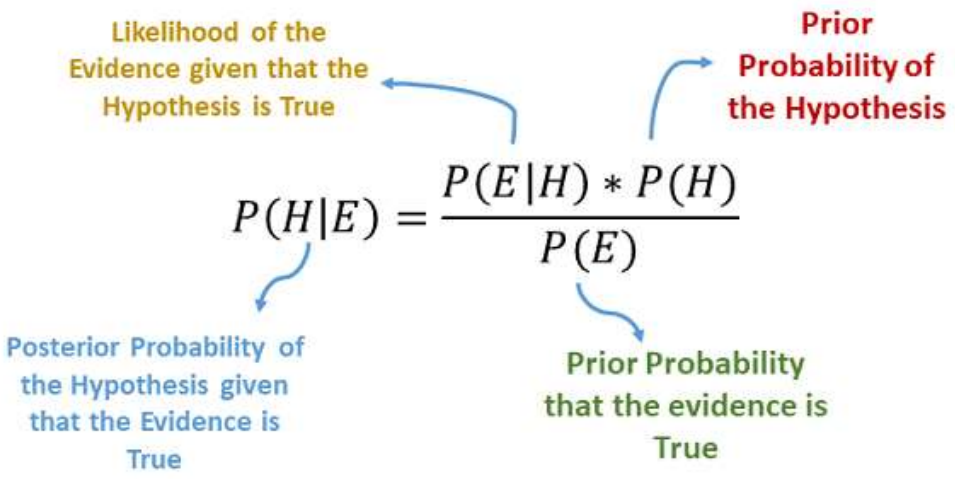
$$P(A|R) = \frac{P(A)\cdot P(R|A)}{P(A)\cdot P(R|A)+P(B)\cdot P(R|B)}$$
$$P(B|R) = \frac{P(B)\cdot P(R|B)}{P(A)\cdot P(R|A)+P(B)\cdot P(R|B)}$$

These are our new and improved probabilities for A and B after we know that R occurred.

$$P(A|R) = \frac{P(A) \cdot P(R|A)}{P(A) \cdot P(R|A) + P(B) \cdot P(R|B)}$$

Generative Models:

Naive Bayes is a generative classifier. It learns from the actual distribution of the dataset by performing operations on it. It does not create a decision boundary to classify data as done in discriminative models. These models use **probability estimates** and **likelihood** to model data points and differentiate between different class labels in a dataset. To know more about generative and discriminative models, refer to this thread I wrote on Twitter:



The image describes Bayes Theorem and is taken from another medium article.

Does Naive Bayes work only for discrete data? If not, what kind of data can it be used for?

Naive Bayes can be adapted to work for discrete, continuous, and data that cannot be represented numerically. We have different types of Naive Bayes classifiers for that:

- Gaussian Naive Bayes
- Bernoulli Naive Bayes
- Multinomial Naive Bayes

Gaussian Naive Bayes:

Gaussian Naive Bayes is a variant of Naive Bayes that follows Gaussian normal distribution and supports continuous data.

The likelihood of the features is assumed to be:

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

Sometimes assume variance

- is independent of Y (i.e., σi),
- or independent of Xi (i.e., σk)
- or both (i.e., σ)

Gaussian Naive Bayes supports continuous-valued features and models, each conforming to a Gaussian (normal) distribution.

An approach to creating a simple model assumes that a Gaussian distribution describes the data with **no co-variance (independent dimensions)**

define such a distribution.

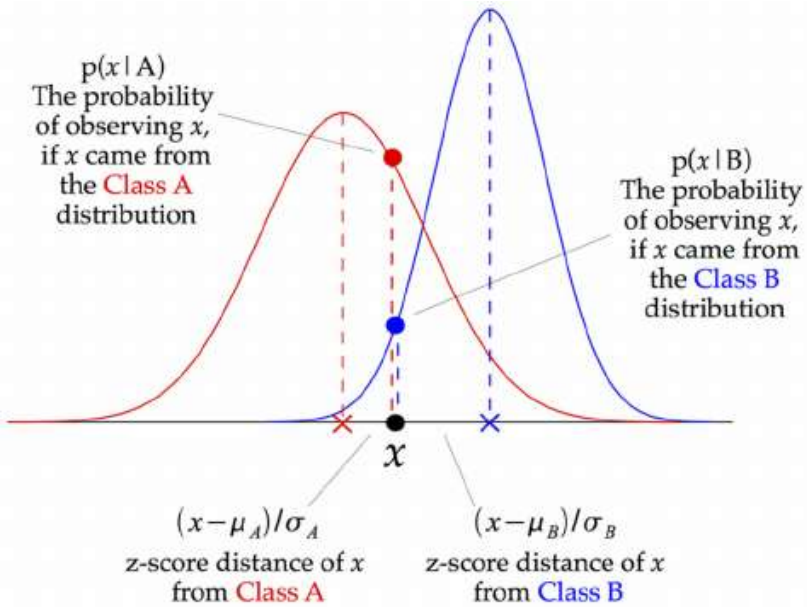


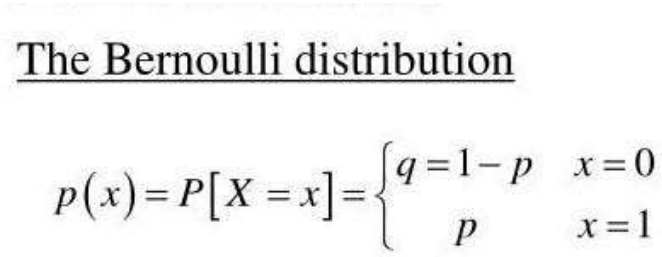
Image is taken from the article on gaussian NB

The above illustration indicates how a Gaussian Naive Bayes (GNB) classifier works. At every data point, the z-score distance between that point and each class mean is calculated, namely the distance from the class mean divided by the standard deviation of that class.

Thus, we see that the Gaussian Naive Bayes has a slightly different approach and can be used efficiently. The above content is taken from this article.

This is used for discrete data and works on the Bernoulli distribution. The main feature of Bernoulli Naive Bayes is that it accepts features only as binary values like true or false, yes or no, success or failure, 0 or 1, and so on. So when the feature values are binary, we know we have to use the Bernoulli Naive Bayes classifier.

As we deal with binary values, let's consider 'p' as the probability of success, 'q' as the probability of failure, and q=1-p. For a random variable 'X' in Bernoulli distribution,



The image is taken from the article on Bernoulli NB

where 'x' can have only two values, either 0 or 1

Bernoulli Naive Bayes Classifier is based on the following rule(likelihood):

The above content is taken from the article. The original article also has shown its working using tabular data.

Multinomial Naive Bayes:

The multinomial model provides an ability to classify data that cannot be represented numerically. Its main advantage is the significantly reduced complexity. It provides an ability to perform the classification using small training sets, not requiring to be continuously re-trained.

The multinomial naïve Bayes is widely used for assigning documents to classes based on the statistical analysis of their contents. It provides an alternative to the "heavy" AI-based semantic analysis and drastically simplifies textual data classification. It is widely used as an alternative to distance-based K-Means clustering and decision tree forests. It deals with probability as the "likelihood" that data belongs to a specific class.

Algorithm:

Let S — an input string, D — a corpus of z-documents, C — a set of m-classes:

Compute the class C_s of sample S as follows:

- Split sample S into a set of n-terms
- For each k-th class C_k k=1..m, do the following:

with which the corresponding i-th term occurred in the documents from C_k.

- Evaluate the prior $p(C_k)$ as the full probability that a document occurred in the documents from class C_k.
- Compute the posterior $\Pr(C_k \mid W)$ by adding prior $p(C_k)$ to the sum of each term's w_i, given C_k, probabilities $p(w_i \mid C_k)$:

$$\Pr(C_k \mid W) = \log p(C_k) + \sum_{i=1}^{n} [\, w_i * \log p(w_i \mid C_k) \,]$$

prior likelihood

The Posterior $\Pr(C_k \mid W)$ Formulae

3. Determine a class C_s of S, for which $\Pr(C_k \mid W)$ k=1..m is the maximum:

$$C_s = \underset{k \in \{1..m\}}{\operatorname{argmax}} \mid \Pr(C_k \mid W) \mid$$

The Class C_s Of S Computation

This algorithm's complexity σ is evaluated as:

$$\sigma = O(nm \times (z+2))$$

z — the total amount of documents in D, n — the number of terms in sample S, m — the number of classes C

The author's above expressions are beautifully derived in the original article here. :)

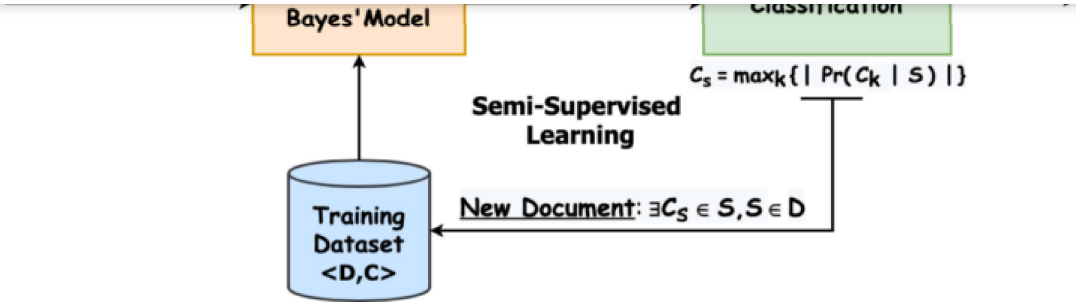
All of the above methods work in a supervised setting. Do you think Naive Bayes can be trained in any other way?

The Naive Bayes algorithm can be adjusted to be trained in a semi-supervised setting. Semi-supervised learning provides an ability to increase the multinomial models' performance. Also, it allows for improving the quality of classification by training the model based on the corpus of documents already classified.

The semi-supervised learning algorithm is rather intuitively simple and formulated, such as:

- Compute class C_s of S using the multinomial model discussed above.
- Append S labeled with C_s to the corpus of documents D.
- Re-evaluate the classification model.

Proceed with the following process to classify each new sample S.



Semi-Supervised Learning Process(Source)

What are some of the benefits of the Naive Bayes algorithm?

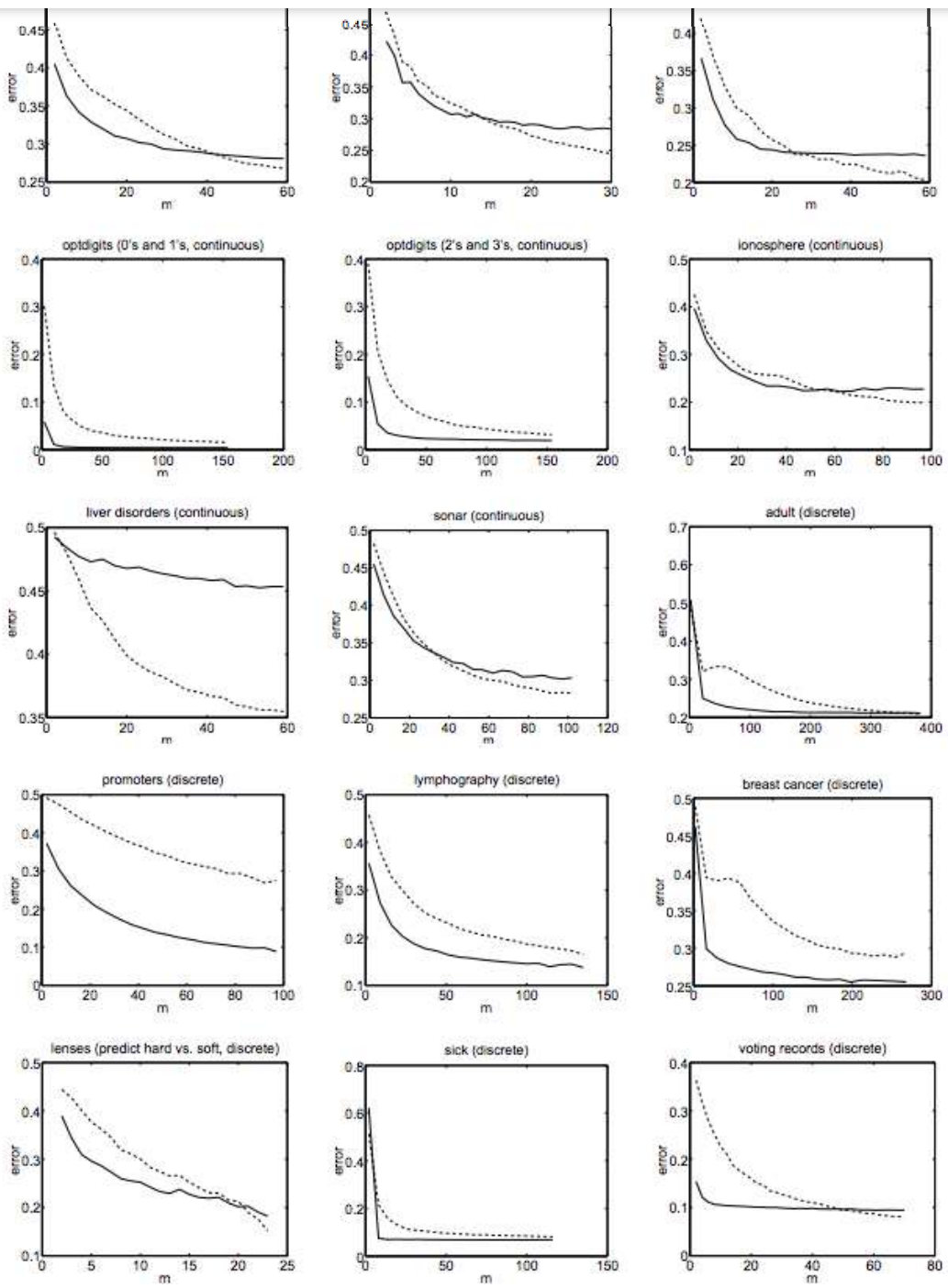
It works better than simple algorithms like logistic regression etc. It also works well with categorical data and with numerical data as well. Additionally, It is very easy and fast to work with the Naive Bayes classifier. Complex and high-dimensional data is well suited for the Naive Bayes classifier. It can also be trained using a small labeled dataset with semi-supervised learning.

In other words:

- It performs well with both clean and noisy data. I couldn't find any good explanation for this behavior other than a few experimental observations, but here is a good answer on quora that might give some insight into this.
- Training takes a few samples, but the fundamental assumption is that the training dataset is a genuine representation of the population.

Is there any definitive reason Naive Bayes works better than simple algorithms like logistic regression? And is it always true, or is there any caveat to it?

This paper by Professor Andrew Ng and Professor Michael I Jordan provides mathematical proof of the error properties of both models. They conclude that when the training size reaches infinity, the discriminative model: logistic regression performs better than the generative model Naive Bayes. However, the generative model reaches its asymptotic faster (O(log n)) than the discriminative model(O (n)), i.e., the generative model (Naive Bayes) reaches the asymptotic solution for fewer training sets than the discriminative model (Logistic Regression).



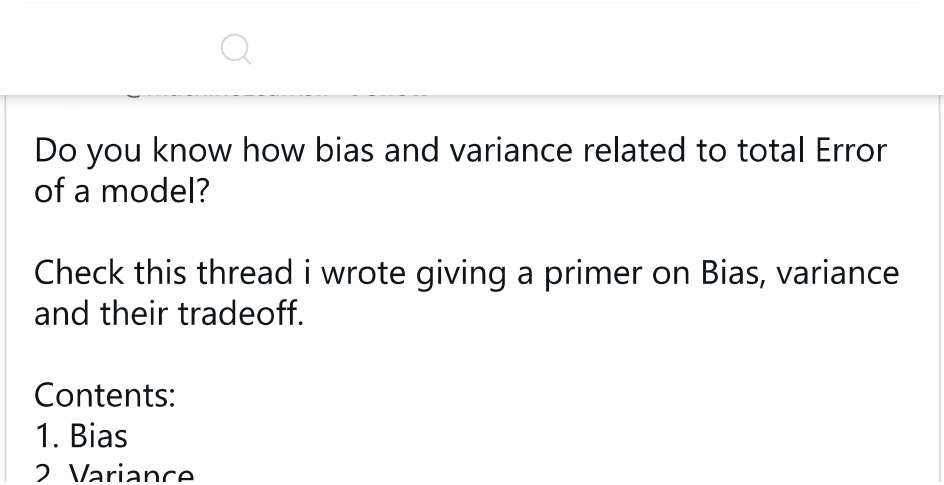
Naive Bayes also assumes that the features are conditionally independent. Real data sets are never perfectly independent, but they can be close. In short, **Naive Bayes has a higher bias but lower variance compared to logistic regression. Naive Bayes will be a better classifier if the data set follows the bias.** Both Naive Bayes and Logistic regression are linear classifiers. Logistic Regression predicts the probability using a direct functional form, whereas Naive Bayes figures out how the data was generated given the results.

Can you elaborate on your understanding of bias and variance?

To answer this question it's better to break it down into more fundamental questions like,

- What are bias and variance?
- How do underfitting and overfitting relate to bias and variance?
- In terms of mathematical formulation, how do bias and variance make up the total error of the model?

All of these questions are answered in this thread I wrote on Twitter ⬇️



Why do you think complex and high-dimensional data suits the Naive Bayes Classifier well?

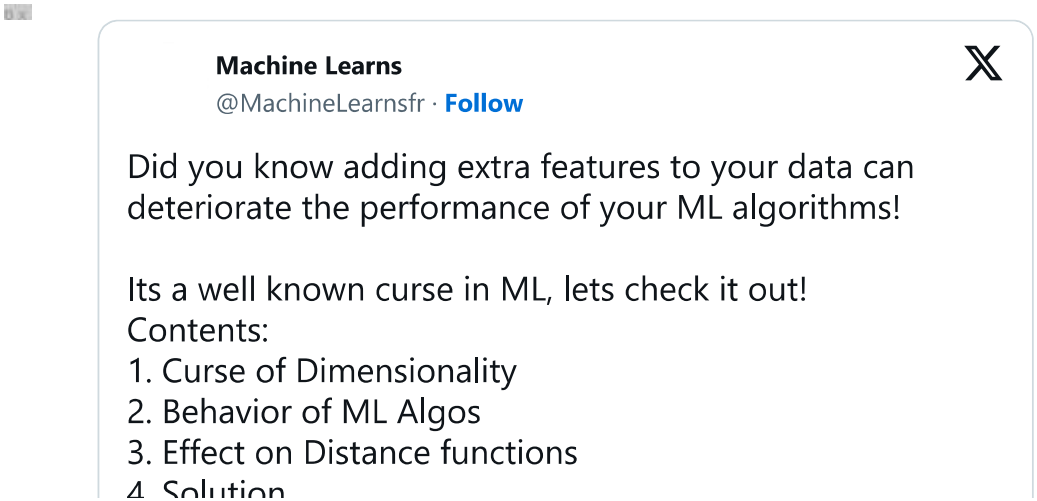
Naive Bayes implicitly treats all features as being independent of one another. Therefore the sorts of curse-of-dimensionality problems typically rear their head when dealing with high-dimensional data do not apply.

If your data has k dimensions, then a fully general ML algorithm that attempts to learn all possible correlations between these features has to deal with 2^k possible feature interactions and therefore needs on the order of 2^k many data points to be performant. However, because Naive Bayes assumes independence between features, it only needs on the order of k many data points, exponentially fewer.

However, this comes at the cost of only being able to capture much simpler mappings between the input variables and the output class, and as such

trained on a large dataset when it comes to tasks like image recognition, although it might perform better on *very* small datasets(but with more features).

Here is a Twitter thread I wrote on the curse of dimensionality. Feel free to refer. :)



What are some cons of the Naive Bayes classifier?

Naive Bayes classifiers suffer from the "Zero Frequency" problem. This happens when a category is not present in the training set. It will give it 0 probability.

features are somehow co-related with each other.

Do you know any solution to the “Zero Frequency” problem?

An approach to overcome this ‘zero-frequency problem’ is to add one to the count for every attribute value-class combination when an attribute value doesn’t occur with every class value.

This will lead to the removal of all the zero values from the classes and, at the same time, will not impact the overall relative frequency of the classes.

In the example above, we increase the value of the word count by 1 for both the spam and the ham emails. Then we calculate the probability of a new email with the words ‘office’, ‘rich,’ ‘rich,’ and ‘money’ as spam or ham.

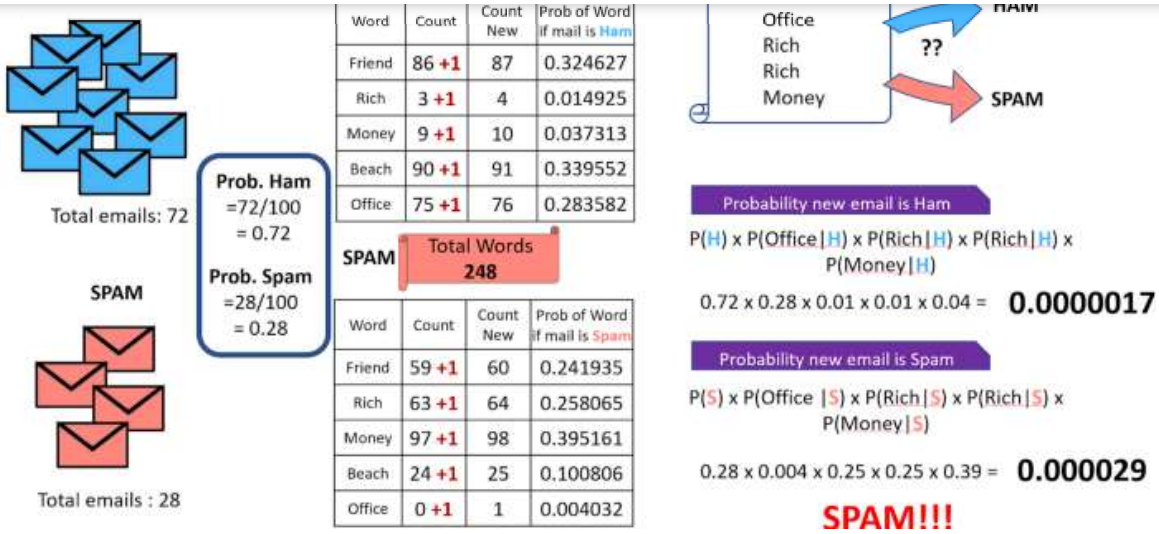


Image showing the effect of adding a constant to solve the zero frequency problem in NB(Souce)

We might encounter zero division error when the probability for a particular scenario in the numerator is zero. To mitigate this, we can use Laplace Smoothing, which adds a number to the numerator and another to the denominator.

The next two questions of this article are not as general as the other questions. They focused on a problem solved using the main algorithm we discussed. I deliberately did this to point out that exploring and investigating are good. ***The interviewer will appreciate it if you explain concepts using an example you have worked on.*** *It will show that you like to dig deep into things; basically, you are a nerd and will be an asset to the company.*

Let's take an example of **spam email classification**. Particular words have particular probabilities of occurring in spam emails and legitimate emails. The filter doesn't know these probabilities in advance and must first be trained so it can build them up. To train the filter, the user must manually indicate whether a new email is a spam or not. For all words in each training email, the filter will adjust the probabilities that each word will appear in the spam or legitimate email in its database.

After training, the word probabilities (also known as likelihood functions) are used to compute the probability that an email with a particular set of words belongs to either category. Each word in the email contributes to the spam probability or only the most interesting words. This contribution is called the posterior probability and is computed using Bayes' theorem. Then, the email's spam probability is computed over all words in the email, and if the total exceeds a certain threshold (say 95%), the filter will mark the email as spam.

If we were to summarize the whole process of classifying a mail as spam or ham, we can do it as follows:

- Compute the spamminess of individual words.
- Combine all the word probabilities to get a score.
- Compare the score to a threshold and give the final verdict.

$$\Pr(S|W) = \frac{\Pr(W|S) \cdot \Pr(S)}{\Pr(W|S) \cdot \Pr(S) + \Pr(W|H) \cdot \Pr(H)}$$

where:

- Pr(S|W) is the probability that a message is spam, knowing that the word is in it;
- Pr(S) is the overall probability that any given message is spam;
- Pr(W|S) is the probability that the word appears in spam messages;
- Pr(H) is the overall probability that any given message is not spam (is “ham”);
- Pr(W|H) is the probability that the word appears in ham messages.

The number Pr(W|S) used in this formula is approximated to the frequency of messages containing a particular word in the messages identified as spam during the learning phase. Similarly, Pr(W|H) is approximated to the frequency of messages containing a particular word in the messages identified as ham during the learning phase. For these approximations to make sense, the set of learned messages needs to be big and representative enough.

consider several words and combine their spamicities to determine a message's overall probability of being spam.

Combine all the word probabilities to get a score

Words present in the message are independent events. This condition is not generally satisfied (for example, in natural languages like English, the probability of finding an adjective is affected by the probability of having a noun). Still, it is a useful idealization, especially since the statistical correlations between individual words are usually not known. On this basis, one can derive the following formula from Bayes' theorem:

$$p = \frac{p_1 p_2 \cdots p_N}{p_1 p_2 \cdots p_N + (1 - p_1)(1 - p_2) \cdots (1 - p_N)}$$

where:

- p is the probability that the suspect message is spam;
- p_1 is the probability $p(W_1|S)$ that the first word (for example, "replica") appears, given that the message is spam;
- p_2 is the probability $p(W_2|S)$ that the second word (for example, "watches") appears, given that the message is spam;

The result *p* is typically compared to a given threshold to decide whether the message is spam or not. **If *p* is lower than the threshold, the message is considered as likely ham; otherwise, it is considered as likely spam.**

Can you tell me about the advantages and disadvantages of using a bayesian based spam filtering model?

Advantages:

The Naive Bayes spam filtering model can be trained on a per-user basis. A user's spam is often related to the online user's activities. For example, a user may have been subscribed to an online newsletter that the user considers spam. This online newsletter is likely to contain words common to all newsletters, such as the name of the newsletter and its originating email address. A Bayesian spam filter will eventually assign a higher probability based on the user's specific patterns.

The word probabilities are unique to each user and can evolve over time with corrective training whenever the filter incorrectly classifies an email. As a result, Bayesian spam filtering accuracy after training is often superior to pre-defined rules.

avoiding false positives, where legitimate email is incorrectly classified as spam. For example, a pre-defined rules filter might reject it outright if the email contains the word “Nigeria,” frequently used in Advance fee fraud spam. A Bayesian filter would mark the word “Nigeria” as a probable spam word but would take into account other important words that usually indicate legitimate e-mail. For example, a spouse's name may strongly indicate the e-mail is not spam, which could overcome the use of the word “Nigeria.”

Disadvantages:

The NB spam filtering model may be susceptible to Bayesian poisoning, a technique used by spammers to degrade the effectiveness of spam filters that rely on Bayesian filtering. A spammer practicing Bayesian poisoning will send emails with large amounts of legitimate text (gathered from legitimate news or literary sources). Spammer tactics include the insertion of random innocuous words that are not normally associated with spam, decreasing the email’s spam score, and making it more likely to slip past a Bayesian spam filter.

One solution to beat this is using Paul Graham’s scheme, where only the most significant probabilities are used. Padding the text with non-spam-related words does not significantly affect the detection probability.

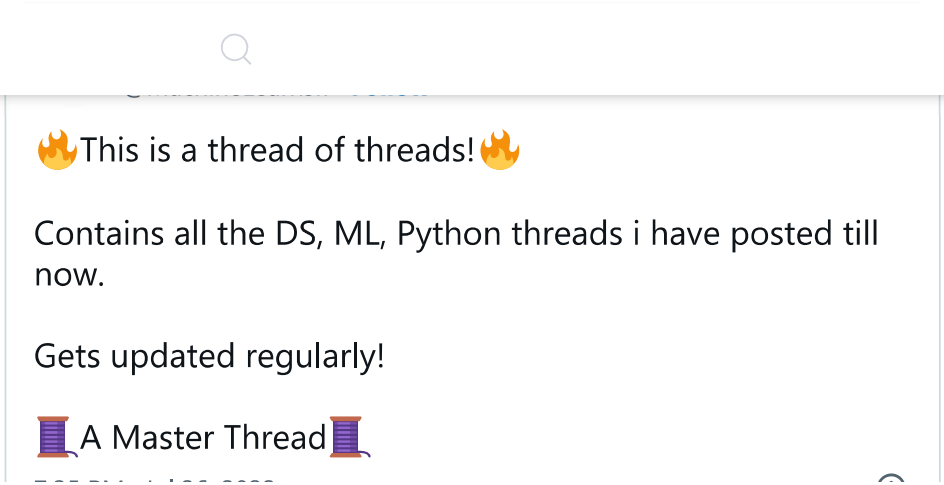
Concluding the whole article

All of the above questions are enough to tackle any discussion on the topic of Naive Bayes when asked in an interview. Of course, if you are going into a research company, you need to have a much deeper understanding of the algorithm and how to use it to solve the research problem you were trying to tackle.

And as a closing remark, I want to announce that I'm **pro nerd**. In fact, I want to become a nerd and would encourage you to become one too!!!!

Being a nerd about something shows that you love what you do and will lead you to live a more satisfying life. Work won't seem like work anymore; that means we will, at some point, if we get good enough at our craft, get paid to have fun! But becoming a nerd is not that easy. In the start, you would have to force yourself to utilize your downtime towards your craft and get obsessed with it! I have written more about this in the **first article** of this series. Please go check it out! :)

And last but not least, you would have noticed that I have routed you guys to various Twitter threads that covered the more basic concepts at many places in the article. I hope it helped you get a quick primer on those topics, and if you want to check out more such threads, here is a link to my 🔥 master thread 🔥 , a collection of all the DS, ML, and python threads I have written till now.



Acknowledgments

This article would not have been possible without the numerous articles and blogs others have written on Naive Bayes and other fundamental concepts I have covered. And I would like wholeheartedly thank the respective authors for doing such a great job!

I have to admit that it takes a significant amount of time to curate the questions and find the best possible answers to those questions. So most answers are directly taken from the article/blog with minor editing or paraphrasing.

References

(Not in any particular order)

Why Naive Bayes is so naive?

Gaussian Naive bayes

Bernoulli Naive Bayes

Multinomial Naive bayes

Comparison between logistic regression and Naive Bayes

Semi supervised learning in Naive Bayes

There are many other important link, for that refer to my rough notes for this article.

Machine Learning | Data Science | Interview

---

6 min read


Samuele Mazzanti

Why “Statistical Significance” Is Pointless

Here’s a better framework for data-driven decision-making

9 min read


Ashu Jha

My Machine Learning Journey: Perfect Roadmap for Beginners

Learning Approach: Code First, Theory Later

6 min read


Be 10x Engineer

Staff Backend Engineer @ Google | Interview Experience

It was a crisp Tuesday morning when I first saw the email. My hands trembled slightly as I read the subject line: “Google Interview...

3 min read


Abdur Rahman

7 min read

Ritesh Gupta

Can You Handle These 25 Toughest Data Science Interview Questions?

The role of a Data Scientist demands a unique blend of skills, including statistics, machine learning, data analysis, and programming. In...

7 min read