

# NAIVE BAYES

[CONDITIONAL PROBABILITY](#)

[INDEPENDENT AND MUTUALLY EXCLUSIVE EVENTS](#)

[BAYES THEOREM](#)

[NAIVE BAYES](#)

[TOY EXAMPLE - NAIVE BAYES](#)

[NAIVE BAYES ON TEXT DATA](#)

[LAPLACE ADDITIVE SMOOTHING](#)

[LOG - PROBABILITIES FOR NUMERICAL STABILITY](#)

[BIAS-VARIANCE TRADEOFF](#)

[FEATURE IMPORTANCE AND INTERPRETABILITY](#)

[OUTLIERS](#)

[MISSING VALUES](#)

[GAUSSIAN NAIVE BAYES](#)

[BEST AND WORSE CASES FOR NAIVE BAYES](#)

## CONDITIONAL PROBABILITY

Naive Bayes is a classification algorithm based on the basics of Probability

We have two dices  $D_1$  and  $D_2$ . So together they can take 36 values. One dice has 6 outcomes. So their combined outcome is  $6 * 6 = 36$ . Eg - (1,1), (1,2), (1,3)...(2,1), (2,2)...(6,6)

<b>D1</b>	3	4	5	6	7	8	9	10
4	5	6	7	8	9	10	11	12
5	6	7	8	9	10	11	12	
6	7	8	9	10	11	12		

✓ What is the probability that  $D_1 = 2$  given that  $D_1 + D_2 \leq 5$  ?

Table 3 shows that for 3 of these 10 outcomes,  $D_1 = 2$ .

Thus, the conditional probability  $P(D_1=2 | D_1+D_2 \leq 5) = \frac{3}{10} = 0.3$ .

+		D2					
		1	2	3	4	5	6
<b>D1</b>	1	2	3	4	5	6	7
	2	3	4	5	6	7	8
	3	4	5	6	7	8	9
	4	5	6	7	8	9	10
	5	6	7	8	9	10	11
	6	7	8	9	10	11	12

conditional-prob
given
  
P(D<sub>1</sub>=2 |
D<sub>1</sub>+D<sub>2</sub> ≤ 5)
  
English   Conditioned-on
  
 $= \frac{3}{10}$

The above equation is the equation of conditional probability. Let's try to interpret it in English, it says that what is the Probability that  $D_1 = 2$  given  $D_1 + D_2 \leq 5$  .

So,  $P(D_1 + D_2 \leq 5) = 10/36$  and  $P(D_1 = 2) = 3/36$

Definition :-  $P(A | B) = \frac{P(A \cap B)}{P(B)}$  here A :  $D_1 = 2$  and B :  $D_1 + D_2 \leq 5$

Using this equation we can get our answer which is 3/10

[Conditional probability - Wikipedia](#)

## INDEPENDENT AND MUTUALLY EXCLUSIVE EVENTS

### INDEPENDENT EVENTS

Independent Events & Mutually Exclusive Events

A, B are said to be "independent"

Def:  $\begin{cases} P(A|B) = P(A) \\ P(B|A) = P(B) \end{cases}$

$\begin{cases} P(D_1=6 | D_2=3) = P(D_1=6) \\ P(D_2=3 | D_1=6) = P(D_2=3) \end{cases}$

$D_1 = 6$  in die 1 throw  
 $(D_1=6)$

$D_2 = 3$  in die 2's throw  
 $(D_2=3)$

Events are said to be independent if a probability of event A is not dependent on event B

Take the example of 2 dices above thrown. Their outcomes are independent obviously

Mutually exclusive events:

If  $P(A|B) = P(B|A) = 0$  then A & B are said to mut-excl

$P(A \cap B) = 0$

$\frac{P(A \cap B)}{P(B)}$

event A:  $D_1 = 6$   
 event B:  $D_1 = 3$

$P(B \cap A) = P(A \cap B)$

$B \cap A = A \cap B$

In  $P(A|B) = 0$  because if it's given  $D_1 = 3$  (B) then there's no chance of A :  $D_1 = 6$  happening. Vice versa is also true here. So, they are mutually exclusive events  $P(A \cap B) = 0$

## YES THEOREM

Bayes's Theorem: (1760's)

Thm:  $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$  if  $P(B) \neq 0$

Proof:  $P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A, B)}{P(B)}$

$A \cap B = B \cap A \rightarrow \text{set theory}$

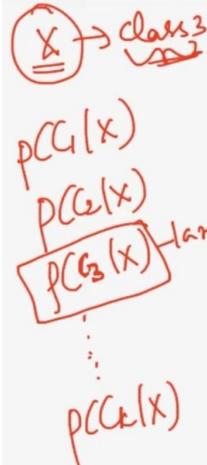
$P(A|B) = \frac{P(B \cap A)}{P(B)} = \frac{P(B, A)}{P(B)}$

$P(A) P(B|A) = \frac{P(B \cap A)}{P(A)}$  → defn.

Here,  $P(A|B) = \frac{P(B \cap A)}{P(B)}$  and  $P(B|A) = \frac{P(B \cap A)}{P(A)}$  and it'll be  $P(B \cap A) = P(A) \cdot P(B|A)$

Therefore,  $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$  which is  'es' theorem

## NAIVE BAYES



The problem with the above formulation is that if the number of features  $n$  is large or if a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable. Using Bayes' theorem, the conditional probability can be decomposed as

$$p(C_k | x) = \frac{p(C_k) p(x | C_k)}{p(x)} \rightarrow p(C_k) p(x | C_k) = p(x \cap C_k)$$

In plain English, using Bayesian probability terminology, the above equation can be written as

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

In practice, there is interest only in the numerator of that fraction, because the denominator does not depend on  $C$  and the values of the features  $x_i$  are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model

$$p(C_k, x_1, \dots, x_n)$$

which can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability:

$$\begin{aligned} p(C_k, x_1, \dots, x_n) &= p(x_1, \dots, x_n, C_k) \\ &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2, \dots, x_n, C_k) \\ &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2 | x_3, \dots, x_n, C_k) p(x_3, \dots, x_n, C_k) \\ &= \dots \\ &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2 | x_3, \dots, x_n, C_k) \dots p(x_{n-1} | x_n, C_k) p(x_n | C_k) p(C_k) \end{aligned}$$

1 ) Naive Bayes is based on Bayes' theorem . Here we are given  $x$  and we want to predict class label  $C_k$  i.e  $P(C_k | x)$ .

2 ) We are given  $x$  and we are predicting class  $C_k$  from multiclass. Whichever class' probability is largest is selected

3) In  $P(C_k | x) = \frac{p(C_k) p(x | C_k)}{p(x)}$  numerator can be ignored because  $x$  is given so it's a constant

So,  $P(C_k | x) \propto p(C_k) p(x | C_k)$ . This numerator is called joint probability model.

$$p(C_k) \quad p(x | C_k) \Rightarrow p(C_k \cap x) = p(C_k, x_1, x_2, \dots, x_n)$$

$$p(C_k, x_1, \dots, x_n)$$

which can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability:

$$\begin{aligned} p(C_k, x_1, \dots, x_n) &= p(x_1, \dots, x_n, C_k) \\ &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2, \dots, x_n, C_k) \xrightarrow{\text{defn. of cond. prob.}} \\ &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2 | x_3, \dots, x_n, C_k) p(x_3, \dots, x_n, C_k) \\ &= \dots \\ &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2 | x_3, \dots, x_n, C_k) \dots p(x_{n-1} | x_n, C_k) p(x_n | C_k) p(C_k) \end{aligned}$$

For every  $x$  probability is calculated

{ obese  
 fit  
 lean

$p(x_1 | x_2, x_3, \dots, x_n, C_k)$   
 $\cancel{p(w=180) | hc=bl, ht=sm, ec=br} \cdot \cancel{\text{constant}}$

Now the "naive" conditional independence assumption means that each feature  $x_i$  is conditionally independent of every other feature given  $C_k$ .

$p(x_i | x_{i+1}, \dots, x_n, C_k)$   
 Thus, the joint model can be expressed as

$$\begin{aligned}
 p(C_k | x_1, \dots, x_n) &\propto p(C_k) \\
 &\propto p(C_k) \\
 &\propto p(C_k)
 \end{aligned}$$

This means that under the above assumptions, the joint probability is proportional to the product of the class prior and the product of the conditional probabilities of each feature given its class.

We want to predict body type given body features. Here, it is extremely rare to find class labels with exact features given above.

$X = (x_1, x_2, \dots, x_n)$  probability model  
 $p(C_k, x_1, \dots, x_n)$

which can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability:

$$\begin{aligned}
 p(C_k, x_1, \dots, x_n) &= p(x_1, \dots, x_n, C_k) \\
 &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2, \dots, x_n, C_k) \\
 &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2 | x_3, \dots, x_n, C_k) p(x_3, \dots, x_n, C_k) \\
 &= \dots \\
 &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2 | x_3, \dots, x_n, C_k) \dots p(x_{n-1} | x_n, C_k) p(x_n | C_k) p(C_k)
 \end{aligned}$$

Now the "naive" conditional independence assumptions come into play: assume that each feature  $x_i$  is conditionally independent of every other feature  $x_j$  for  $j \neq i$ , given the category  $C$ . This means that

$$p(x_i | x_{i+1}, \dots, x_n, C_k) = p(x_i | C_k) \Rightarrow (x_1) \text{ is indep of } (x_{i+1}, x_{i+2}, \dots, x_n) \text{ given } C_k$$

Thus, the joint model can be expressed as

$$\begin{aligned}
 p(C_k | x_1, \dots, x_n) &\propto p(C_k, x_1, \dots, x_n) \\
 &\propto p(C_k) p(x_1 | C_k) p(x_2 | C_k) p(x_3 | C_k) \dots \\
 &\propto p(C_k) \prod_{i=1}^n p(x_i | C_k).
 \end{aligned}$$

Naive: features are cond. indep of each other

Naive: cond-indep of each feature pair

If  $P(A | B, C) = P(A | C)$  the A is conditionally independent from B. This approach is used in Naive Bayes which makes the math easier i.e  $P(x_1 | x_2, x_3, \dots, C_k) = P(x_1 | C_k)$

$$p(x_i | x_{i+1}, \dots, x_n, C_k) = p(x_i | C_k).$$

Thus, the joint model can be expressed as

$$\begin{aligned} p(C_k | x_1, \dots, x_n) &\propto p(C_k, x_1, \dots, x_n) \\ &\propto p(C_k) p(x_1 | C_k) p(x_2 | C_k) p(x_3 | C_k) \dots \\ &\propto p(C_k) \prod_{i=1}^n p(x_i | C_k). \end{aligned}$$

This means that under the above independence assumptions, the conditional distribution over the class variable  $C$  is:

$$\underline{p(C_k | x_1, \dots, x_n)} = \frac{1}{Z} p(C_k) \prod_{i=1}^n p(x_i | C_k)$$

where the evidence  $Z = p(\mathbf{x}) = \sum_k p(C_k) p(\mathbf{x} | C_k)$  is a scaling factor dependent only on  $x_1, \dots, x_n$ , that is, a constant if the values of the feature variables are known.

So  $P(C_k | x) \propto p(C_k) p(x | C_k)$  where  $x = x_1, x_2, x_3, \dots, x_n$  can be represented like above  $\prod$

stands for multiplication just like  $\sum$

$$\begin{aligned} &\propto p(C_k) p(x_1 | C_k) p(x_2 | C_k) p(x_3 | C_k) \dots \\ &\propto p(C_k) \prod_{i=1}^n p(x_i | C_k). \end{aligned}$$

*posterior prob*

This means that under the above independence assumptions, the conditional distribution over the class variable  $C$  is:

$$p(C_k | x_1, \dots, x_n) = \frac{1}{Z} p(C_k) \prod_{i=1}^n p(x_i | C_k)$$

*p(Ck | x)*

where the evidence  $Z = p(\mathbf{x}) = \sum_k p(C_k) p(\mathbf{x} | C_k)$  is a scaling factor dependent only on  $x_1, \dots, x_n$ , that is, a constant if the values of the feature variables are known.

*p(C1 | x), p(C2 | x), ..., p(Ck | x)*

### Constructing a classifier from the probability model [edit]

The discussion so far has derived the independent feature model, that is, the naive Bayes probability model. The naive Bayes classifier combines this model with a decision rule. One common rule is to pick the hypothesis that is most probable; this is known as the maximum a posteriori or MAP decision rule. The corresponding classifier, a Bayes classifier, is the function that assigns a class label  $\hat{y} = C_k$  for some  $k$  as follows:

$$\checkmark \quad \hat{y} = \underset{k \in \{1, \dots, K\}}{\operatorname{argmax}} p(C_k) \prod_{i=1}^n p(x_i | C_k).$$

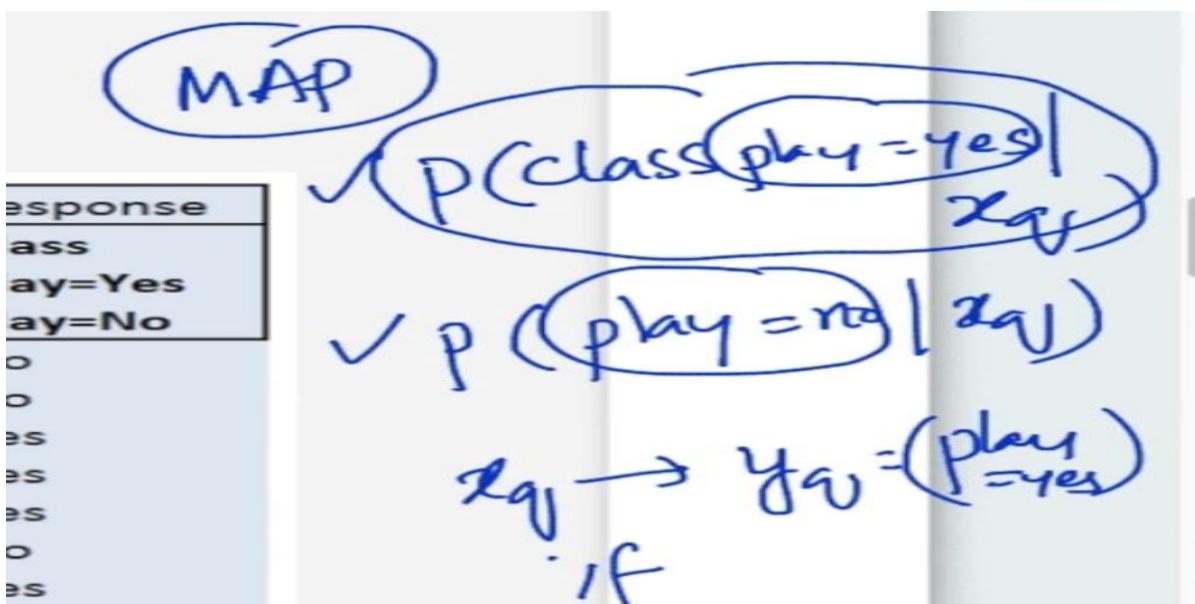
To predict class label  $\hat{y}$  we want the maximum of the above equation

[Naive Bayes classifier - Wikipedia](#)

## TOY EXAMPLE - NAIVE BAYES

	Predictors				Response
Outlook	Temperature	Humidity	Wind	Class	$y_i = Y \text{ or } N$
Day1 →	Sunny	Hot	High	Weak	No ✓
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

This is Binary Classification with the help of categorical feaures .



We want to predict if match can be played using class Plays given a new query  $x_q$  with features.

We are in the learning phase, and we want to calculate likelihoods . What are likelihoods? They are probability of a feature given a class label.

	Predictors				Response
	Outlook	Temperature	Humidity	Wind	Class Play=Yes Play=No
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

$p(C \text{=} \text{No}) = 5/14$

$p(C \text{=} \text{Yes}) = 9/14$

$p(C | f_1, f_2, f_3, f_4) = p(f_1 | C) * p(f_2 | C) * p(f_3 | C) * p(f_4 | C)$

$p(C | f_1, f_2, f_3, f_4) = p(C) * p(f_1 | C) * p(f_2 | C) * p(f_3 | C) * p(f_4 | C)$

$p(C | f_1, f_2, f_3, f_4) = 9/14$

We want to calculate  $P(C | f_1, f_2, f_3, f_4)$  where C is class label and f is feature of a query  $x_q$ . It is proportionate to  $P(f_i | C) * P(C)$ .  $P(C)$  is easy to calculate.  $P(C = \text{Yes}) = \text{Total no. of yes in class} / \text{Total no. of class labels} = 9/14$ .

training data. They are:

$P(\text{Outlook}=o|\text{ClassPlay}=b)$ , where  $o \in [\text{Sunny}, \text{Overcast}, \text{Rainy}]$  and  $b \in [\text{yes}, \text{no}]$

$P(\text{Temperature}=t|\text{ClassPlay}=b)$ , where  $t \in [\text{Hot}, \text{Mild}, \text{Cool}]$  and  $b \in [\text{yes}, \text{no}]$ ,

$P(\text{Humidity}=h|\text{ClassPlay}=b)$ , where  $h \in [\text{High}, \text{Normal}]$  and  $b \in [\text{yes}, \text{no}]$ ,

$P(\text{Wind}=w|\text{ClassPlay}=b)$ , where  $w \in [\text{Weak}, \text{Strong}]$  and  $b \in [\text{yes}, \text{no}]$ .

$f_1$

$P(\text{Outlook}=o \text{Class Play=Yes No})$	Frequency		Probability in Class	
$\text{Outlook} =$	Play=Yes	Play=No	Play=Yes	Play=No
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0/5
Rain	3	2	3/9	2/5
	total=9	total=5		

$P(\text{outlook} = \text{Sunny} | C = \text{Yes}) = 2/9$

In the training phase ,we calculate the frequencies. Above for feature  $f_1 = \text{Outlook}$  we are calculating  $P( f_1 | C )$ .  $P(\text{outlook} = \text{Sunny} | C = \text{Yes}) = 2/9$  since  $\text{Outlook} = \text{Sunny}$  is occurring 2 times in total Yes which is 9. We need to calculate it for every feature

$P(f | c)$

The Learning Phase

	Predictors				Response	
	Outlook	Temperature	Humidity	Wind	Class	C
Day1	Sunny	Hot	High	Weak	No	
Day2	Sunny	Hot	High	Strong	No	
Day3	Overcast	Hot	High	Weak	Yes	
Day4	Rain	Mild	High	Weak	Yes	
Day5	Rain	Cool	Normal	Weak	Yes	
Day6	Rain	Cool	Normal	Strong	No	
Day7	Overcast	Cool	Normal	Strong	Yes	
Day8	Sunny	Mild	High	Weak	No	
Day9	Sunny	Cool	Normal	Weak	Yes	
Day10	Rain	Mild	Normal	Weak	Yes	
Day11	Sunny	Mild	Normal	Strong	Yes	
Day12	Overcast	Mild	High	Strong	Yes	
Day13	Overcast	Hot	Normal	Weak	Yes	
Day14	Rain	Mild	High	Strong	No	

Time Complexity :  $O(ndc)$  where n- Number of inputs, d- dimensions ,C - No. of classes

Space Complexity :  $O(d * c)$  because for every d- dimensional feature we need to store

$P(f | c)$

With the MAP rule, we compute the posterior probabilities. This is easily done by looking up the tables we built in the learning phase.

$$P(\text{ClassPlay}=\text{Yes}|x') = \frac{P(\text{Sunny}|\text{ClassPlay}=\text{Yes})}{\cancel{x'_{\text{Sunny}}}} \times \frac{P(\text{Cool}|\text{ClassPlay}=\text{Yes})}{\cancel{x'_{\text{Cool}}}} \times \frac{P(\text{High}|\text{ClassPlay}=\text{Yes})}{\cancel{x'_{\text{High}}}} \times \frac{P(\text{Strong}|\text{ClassPlay}=\text{Yes})}{\cancel{x'_{\text{Strong}}}} \times P(\text{ClassPlay}=\text{Yes})$$

$$= 2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0053$$

$$P(\text{ClassPlay}=\text{No}|x') = [P(\text{Sunny}|\text{ClassPlay}=\text{No}) \times P(\text{Cool}|\text{ClassPlay}=\text{No}) \times P(\text{High}|\text{ClassPlay}=\text{No}) \times P(\text{Strong}|\text{ClassPlay}=\text{No})] \times P(\text{ClassPlay}=\text{No})$$

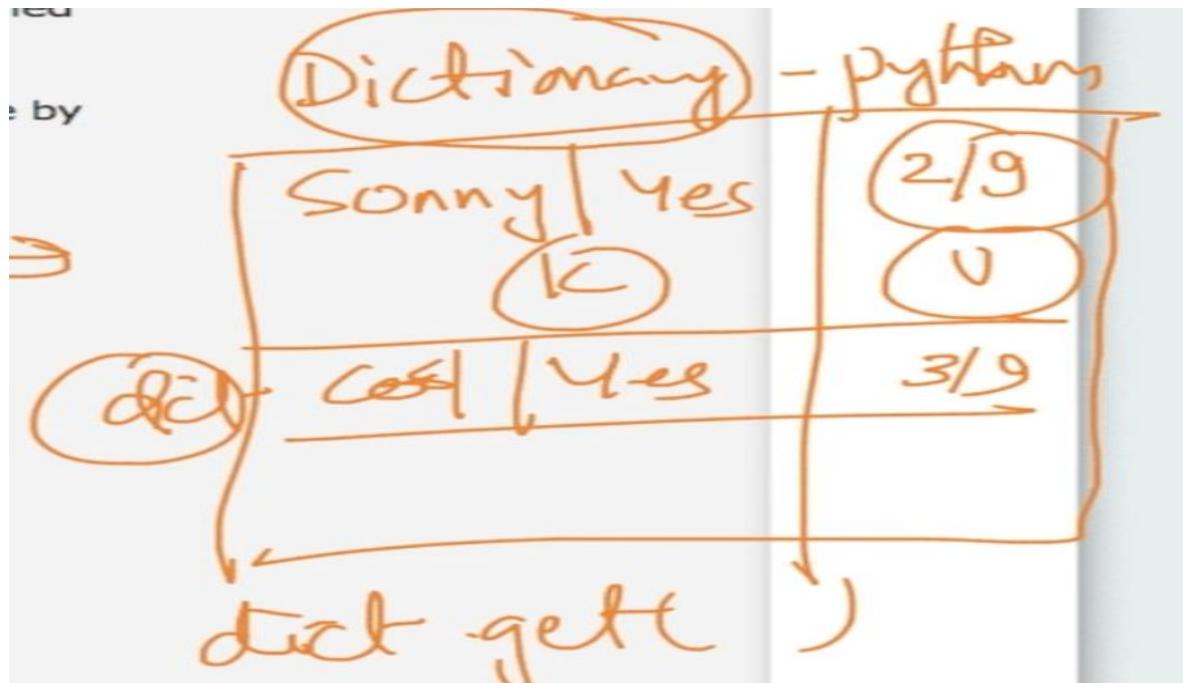
$$= 3/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14 = 0.0205$$

$x'_f = x^1 = (\text{Sunny}, \text{Cool}, \text{High}, \text{Strong})$

$y_q = ?$

$P(C=\text{Yes}|x')$

If we are given query  $x_q$  our  $y_q$  can be calculated very easily since all of the probability features we need are stored in a lookup table (Dictionary). Storing the Query as Key and Probability in value in key- value pair



Look up table with key = Query ( $f \mid C$ ) and Value = Probability score

**Classification Phase**

$(k\text{-NN}) \rightarrow O(nd) \rightarrow \text{Space}$

Let's say, we get a new instance of the weather condition,  $x' = (\text{Outlook}=\text{Sunny}, \text{Temperature}=\text{Cool}, \text{Humidity}=\text{High}, \text{Wind}=\text{Strong})$  that will have to be classified (i.e., are we going to play tennis under the conditions specified by  $x'$ ).

With the MAP rule, we compute the posterior probabilities. This is easily done by looking up the tables we built in the learning phase.

$$\begin{aligned} P(\text{Class}_{\text{Play}}=\text{Yes}|x') &= [P(\text{Sunny}|\text{Class}_{\text{Play}}=\text{Yes}) \times P(\text{Cool}|\text{Class}_{\text{Play}}=\text{Yes}) \times \\ &\quad P(\text{High}|\text{Class}_{\text{Play}}=\text{Yes}) \times P(\text{Strong}|\text{Class}_{\text{Play}}=\text{Yes})] \times \\ &\quad P(\text{Class}_{\text{Play}}=\text{Yes}) \\ &= 2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0053 \end{aligned}$$

$$\begin{aligned} P(\text{Class}_{\text{Play}}=\text{No}|x') &= [P(\text{Sunny}|\text{Class}_{\text{Play}}=\text{No}) \times P(\text{Cool}|\text{Class}_{\text{Play}}=\text{No}) \times \\ &\quad P(\text{High}|\text{Class}_{\text{Play}}=\text{No}) \times P(\text{Strong}|\text{Class}_{\text{Play}}=\text{No})] \times \\ &\quad P(\text{Class}_{\text{Play}}=\text{No}) \end{aligned}$$

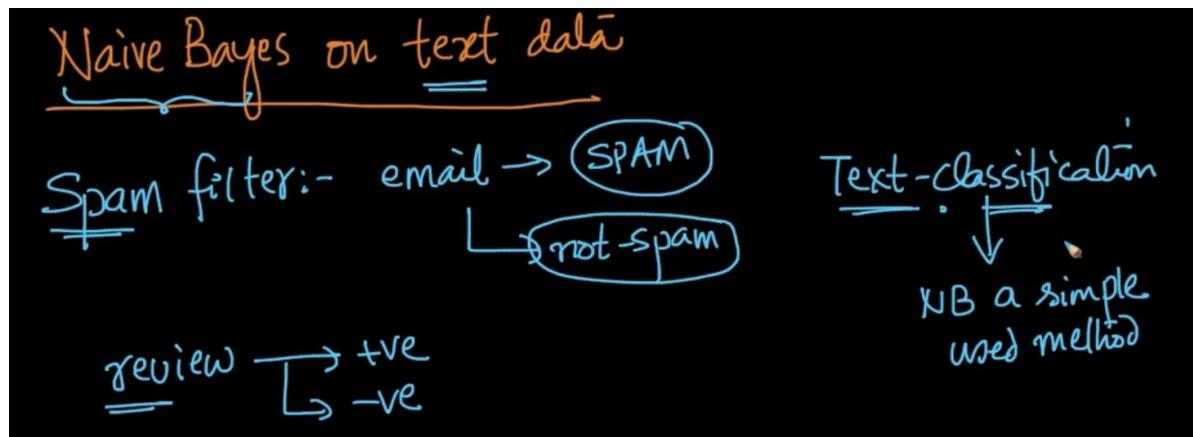
**Train:** - Time:  $O(ndc)$  Space:  $O(d+c)$

**Test:** - Time:  $O(d+c)$  if  $d$  is small  
if  $c$  is small

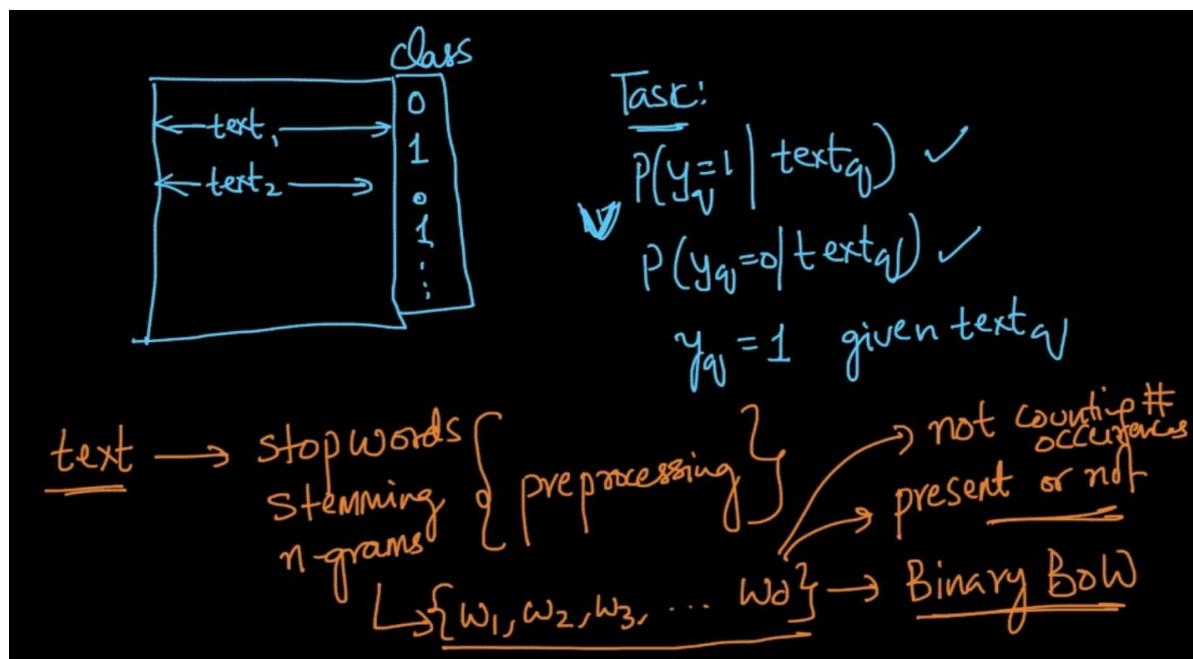
Space Complexity :  $O(d * C)$  here  $d = 4$  and classes = 2 so only 8 values in lookup table

much better than k-NN which has Space Complexity  $O(nd)$ . We don't need to bother our points as we can directly look at the lookup table

## NAIVE BAYES ON TEXT DATA



Naive Bayes is popular for text classification like Spam Filter



Suppose we have a text data with its class labels . Our task is to predict  $y_q$  with text query  $text_q$  . We calculate  $P( y_q = 1 | text_q )$  and  $P( y_q = 0 | text_q )$

If  $P( y_q = 1 | text_q ) > P( y_q = 0 | text_q )$  then  $y_q = 1$

We do the pre-processing steps above on a text and get a set of words  $\{w_1, w_2, \dots, w_d\}$  in a Binary BOW representation i.e if the word is present or not

$$\text{text} \xrightarrow{\text{preproc}} \{w_1, w_2, \dots, w_d\}$$

$$P(y=1 | \text{text}) = P(y=1 | \underbrace{w_1, w_2, \dots, w_d}_{\text{features}})$$

class prior  $\propto$   $P(y=1) * P(w_1 | y=1) * P(w_2 | y=1)$   
 ...  $P(w_d | y=1)$  likelihood

$$P(y=1 | \text{text}) \propto P(y=1) * \prod_{i=1}^d P(w_i | y=1)$$

We calculate  $P(y = 1 | \text{text})$ . Here,  $P(y = 1)$  is easy to calculate. Now we want likelihoods i.e

$P(w | y = 1)$ . Same things for  $P(y = 0 | \text{text})$ , just replace  $y = 1$  with  $y = 0$

$$P(y=0 | \text{text}) \propto P(y=0) * \prod_{i=1}^d P(w_i | y=0)$$


---


$$P(y=1) = \frac{\# \text{Train pts with } y=1}{\text{Total # Train pts}}$$

← text →	0
	1
	0
	0
	1

$$P(y=0) = \frac{\# \text{train pts with } y=0}{\text{Total # train pts}}$$

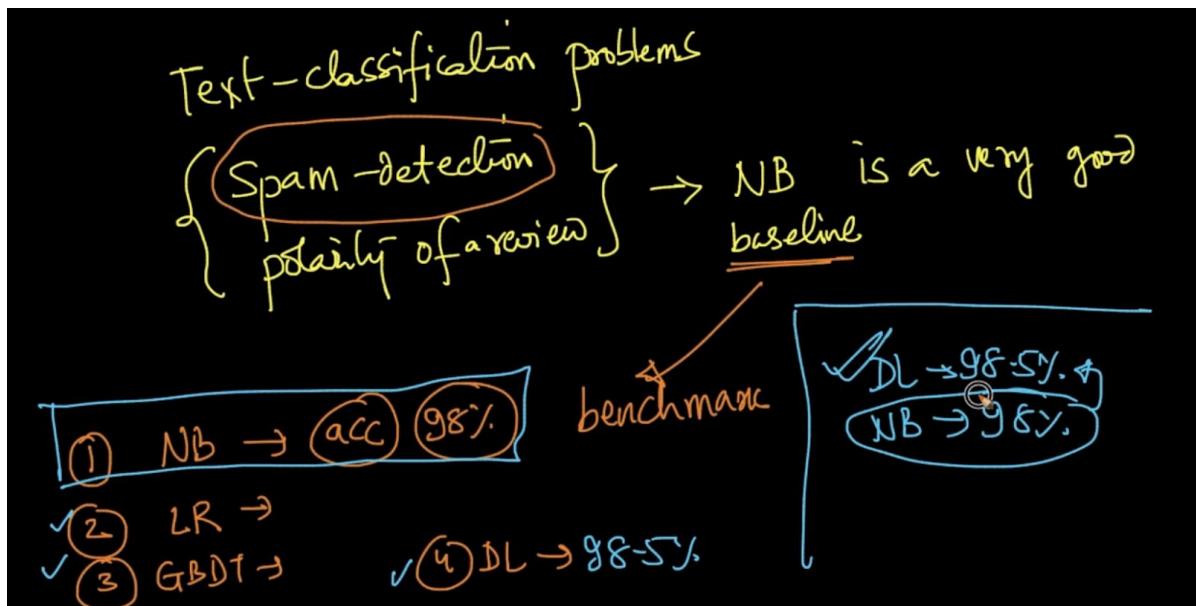
Train

$P(y = 1)$  and  $P(y = 0)$  are calculated very easily. Simple counting stuff. We take all the points with  $y = 1$  / Total points

$$P(w_i | y=1) = \frac{\# \text{Train data pts with contain } w_i \text{ & } y=1}{\# \text{Train data pts with } y=1}$$

Train data with  $y=1$   
 Train data with  $y=0$

For likelihood calculation, we divide the Train Data with  $y = 1$  and  $y = 0$ . For  $P(w | y = 1)$  We count all the data points which contain word  $w$  and  $y = 1$  / Total Data points with  $y = 1$   
 Same for  $y = 0$



Naive Bayes is a very good baseline and it's used as a benchmark. Suppose if we get 98% on our simple baseline Naive Bayes and 98.5% on a complicated DL model then we say it's only a slight improvement

## LAPLACE ADDITIVE SMOOTHING

Laplace Smoothing:

Training:-

$$\left\{ \begin{array}{ll} p(y=1) & ; p(y=0) \leftarrow \text{class priors} \\ p(w_1|y=1) & p(w_1|y=0) \\ p(w_2|y=1) & p(w_2|y=0) \\ \vdots & \vdots \\ p(w_m|y=1) & p(w_m|y=0) \end{array} \right\} \text{likelihoods}$$

In the training parts, we want to calculate likelihoods and class priors

Test:  $\text{text}_q = (w_1, w_2, w_3, w')$  set of words in Training data -

$w'$  is not present  $\{w_1, w_2, w_3, \dots, w_m\}$

English has tens of thousands of words  
n-grams

In the testing part if there occurs a word  $w'$  which is not present in set of words of Training data then what we should be doing?

$$\begin{aligned}
 P(y=1 | \text{text}_w) &= P(y=1 | w_1, w_2, w_3, w') \\
 &= P(y=1) * P(w_1 | y=1) * P(w_2 | y=1) \\
 &\quad * P(w_3 | y=1) * P(w' | y=1) \\
 P(w' | y=1) ; P(w' | y=0) &\quad (\text{ignoring it}) \\
 P(w' | y=0) = 1 &\quad \times P(w' | y=1) = 1
 \end{aligned}$$

We want to calculate  $P(w' | y = 1)$ . What if we ignore it? It'll mean  $P(w' | y = 1) = 1$  and  
Also  $P(w' | y = 0) = 1$  which can't be the case

$$\left\{
 \begin{aligned}
 P(w' | y=1) &= \frac{P(w', y=1)}{P(y=1)} \\
 &= \frac{\# \text{train pts s.t } w' \text{ occurs in } y=1}{n_1} \\
 n_1 &\rightarrow \# \text{ train pts where } y=1 \\
 &= \frac{0}{n_1} = 0
 \end{aligned}
 \right.$$

We can't go for our conditional probability method since  $w'$  is not present which means

$P(w' \cap y = 1) = 0$ . Therefore all the probability will go to 0. To tackle this we use Laplace smoothing

Laplace smoothing (not ~~Laplacian~~ smoothing)  
Additive smoothing

$$P(\underline{w'} | y = 1) = \frac{o + \alpha}{n_1 + \alpha K} \rightarrow \alpha = 1 \text{ Typically}$$

$\begin{cases} 1 \rightarrow \text{present} \\ 0 \rightarrow \text{not present} \end{cases}$

$K = \# \text{ distinct values}$   
 $\begin{cases} 1 & w' \text{ can take} \end{cases}$

We add  $\alpha$  and  $K$  for smoothing.  $K$  = number of distinct values w/ feature can take . Here, since it's a Binary BOW it can only take 2 distinct values (0 or 1) therefore  $K = 2$

$$P(\underline{w'} | y = 1) = \frac{o + \alpha}{n_1 + \alpha K}$$

Let  $n_1 = 100$ ,  $\alpha = 0.001$ ,  $K = 2$  because  $w' = 0 \text{ or } 1$

case 1:-  $\alpha = 1 = \frac{1}{10^2} \neq 0$

$0 \neq P(y=1 | \text{text}_y) \Leftrightarrow P(w' | y=1) \neq 0$

Case 1 :  $\alpha = 1$ . We got rid of the problem of  $P(w' | y = 1) = 0$ . It won't be 0 anymore but why using this method. We could've just replaced new word  $w'$  probability with a small e.g 0.01

But there are some mathematical insights we can get from this smoothing method

Case 2 :-  $\alpha = 10,000$

$$P(\underline{w} | y=1) = \frac{0 + 10,000}{100 + 20,000} = \frac{10,000}{20,100} \approx \frac{1}{2}$$

$n_i = 100$

$\checkmark w^1 = 0 \rightarrow \text{two possibilities}$

$\checkmark w^1 = 1$

$$P(w^1 | y=1) = P(w^1 | y=0) = \frac{1}{2}$$

Case 2 :  $\alpha = 10,000$ .  $P \approx 1/2$  is a good thing and more logical because we don't know anything about  $w'$  and saying that probability of  $w'$  occurring in both classes =  $1/2$  i.e equiprobable is logical. So what happens in Laplace Smoothing ?

Laplace Smoothing

for all words

$$P(w_i | y=1) = \frac{\# \text{data pts with } w_i \text{ & } y=1 + \alpha}{\# \text{data pts } y=1 + \alpha k}$$

$w_i$  present in my training data

It can be done for words which are present in Training data (set of words) not just new words

$$P(w_i | y=1) = \frac{2 + \alpha}{50 + \alpha} = \frac{2}{50}$$

$\alpha = 1 \Rightarrow \frac{2+1}{50+1} = \frac{3}{54}$

$\alpha = 10 \Rightarrow \frac{2+10}{50+20} = \frac{12}{70}$

$\alpha = 100 \Rightarrow \frac{2+100}{50+200} = \frac{102}{250}$

$\alpha = 1000 \Rightarrow \frac{2+1000}{50+2000} = \frac{1002}{2050} = \frac{1}{2}$

So what is happening when  $\alpha$  is increasing ?

$(\alpha \uparrow) \Rightarrow$  moving my likelihood prob to uniform dist

$n_i$  is small  
num or den is small  $\rightarrow$  less confidence in the ratio

$\left. \begin{array}{c} \\ \\ \end{array} \right\} \quad \frac{1}{2}$

$\alpha \uparrow$ , we move our likelihood probabilities to uniform distribution. If our numerator or denominator is small we've less confidence in the ratio. So by increasing  $\alpha$  we are moving

towards a stable confidence. In above example, at  $\alpha = 1000$  it is  $\frac{1}{2}$  which makes much sense since  $w'$  can only take 2 values (0 or 1) so it has smoothed/ moved to a uniform distribution.

## LOG - PROBABILITIES FOR NUMERICAL STABILITY

Log-probabilities vs numerical stability

$$P(y=1 \mid w_1, w_2, \dots, w_d)_{0 \text{ to } 1}$$

$$= \underbrace{p(y=1)}_{0 \text{ to } 1} * \underbrace{p(w_1 \mid y=1)}_{* \dots *} * p(w_2 \mid y=1) * p(w_3 \mid y=1) * \dots * p(w_d \mid y=1)$$

$d$  is large      Let  $d=100$

Calculation for prediction of class label with  $d = 100$  features

$$\left\{ \begin{array}{l} 0.2 \times 0.1 \times 0.2 \times \dots \\ = 0.00004 \end{array} \right.$$

100 numbers all of which 0 to 1

numerical stability

numerical underflow

Python double precision  
16 significant digits  
(rounding)

If we multiply 100 numbers that are between 0 - 1 numerical underflow issue arises i.e Python gives only 16 significant values. So with that multiplication it'll round off to 16 which will give us errors. If we apply log. E.g  $\log(0.0004) = -3.39$  things will be easier

Log-probabilities:

$$\log \left( P(y=1 | w_1, w_2, \dots, w_d) \right) = \underbrace{\log P(y=1)}_{\text{log}} + \underbrace{\log \prod_{i=1}^d P(w_i | y=1)}_{\text{addn}}$$

$$\log \left( P(y=0 | w_1, w_2, \dots, w_d) \right) = \underbrace{\log P(y=0)}_{\text{log}} + \underbrace{\log \prod_{i=1}^d P(w_i | y=0)}_{\text{addn}}$$

$x \uparrow ; \frac{\log x \uparrow}{\text{monotonic fn}}$

We need to compare  $P(y=1 | w)$  and  $P(y=0 | w)$ . If we do their log then the calculations will get easier as log is a monotonic function

$$\log \left( P(y=1 | w_1, w_2, \dots, w_d) \right) = \log(P(y=1)) + \sum_{i=1}^d \log(P(w_i | y=1))$$

$\log(a+b) = \log a + \log b$ 

mult
addn

Here it can be seen that calculations are getting easier because multiplication get converted to addition. It also converts exponentiation to multiplication as  $\log(a^b) = b \cdot \log a$

## BIAS-VARIANCE TRADEOFF

Bias-Variance tradeoff

{ Naive Bayes :-

<ul style="list-style-type: none"> <li>↳ in laplace smoothing</li> <li>↳ high bias</li> <li>↳ high variance</li> </ul>	<p>high Bias → underfitting</p> <hr/> <p>high Variance → overfitting</p>	<p>small training data changes</p> <p>result in model change</p> <p>dramatically</p>
--	--	--

In Naive Bayes,  $\alpha$  in Laplace smoothing determines high bias or high variance

$$\begin{aligned}
 \text{Case 1:- } \alpha &= 0 \\
 p(w_i | y=1) &= \frac{\# \text{Train data pts where } w_i \text{ occurs \& } y=1}{\# \text{Train pts with } y=1} \\
 &= \frac{\textcircled{2} \rightarrow \text{only } 2 \text{ times}}{1000 \rightarrow \text{true data pts}} \Rightarrow \underline{\underline{rare}}
 \end{aligned}$$

$(n = 2000 \text{ pts})$   $\left( \begin{array}{l} \rightarrow 1000 \text{ true} \\ \rightarrow 1000 \text{ -ve} \end{array} \right)$

In  $\alpha = 0$ , if there's a word  $w_i$  is rare then calculating it's probability is overfitting in a way

Since  $w_i$  occurs only in 2 out of 2000 cases  
↳ Small change in my D-train  
remove the 2 texts ( $x_i$ 's)  
that contain  $w_i$

model shape

$$p(w_i | y=1) = \frac{2}{1000} \text{ to } \frac{0}{1000}$$

Since there's no laplace smoothing i.e  $\alpha = 0$  we remove the texts that contain word  $w_i$

So if we remove those 2 texts our P will change to 0 from 2/1000. It can be said that small changes in Train data result in huge change in model => High variance => Over-fitting

$$\text{for } \alpha \approx 1 \quad p(w_i | y=1) \approx p(w_i | y=0) \approx \frac{1}{2}$$

$$p(y=1 | x_w) \approx p(y_w=1 | x_w) \approx \frac{1}{2}$$

(Underfitting)

$K-NN; K=n$

$x_q \rightarrow \text{Same Class Label} (+ve)$

$n_1: +ve$   
 $n_2: -ve$   
 $n_1 > n_2$

Case 2 :  $\alpha = \text{Very Large}$  then  $P(w | y = 1) \approx P(w | y = 0) \approx \frac{1}{2}$ . Our model can't distinguish so it'll be underfitting

$$\alpha = \text{Very Large:}$$

$$p(y=1 | w_1, w_2, \dots, w_d) = p(y=1) + \prod_{i=1}^d p(w_i | y=1)$$

$$p(y=0 | w_1, w_2, \dots, w_d) = p(y=0) + \prod_{i=1}^d p(w_i | y=0)$$

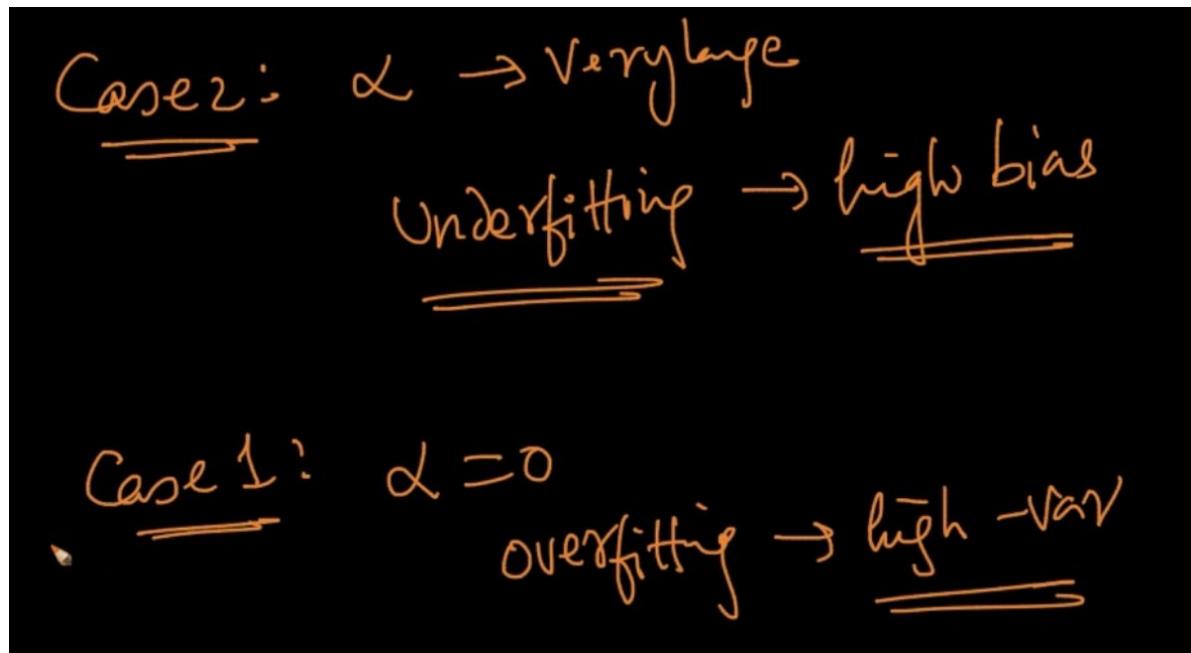
$x_q \rightarrow y_q = 1$

$n \rightarrow n_1: +ve$   
 $n \rightarrow n_2: -ve$

$n_1 > n_2$

Since  $P(w | y = 1) \approx P(w | y = 0) \approx \frac{1}{2}$  it's only dependent on  $P(y = 1)$  and  $P(y = 0)$ . So, if

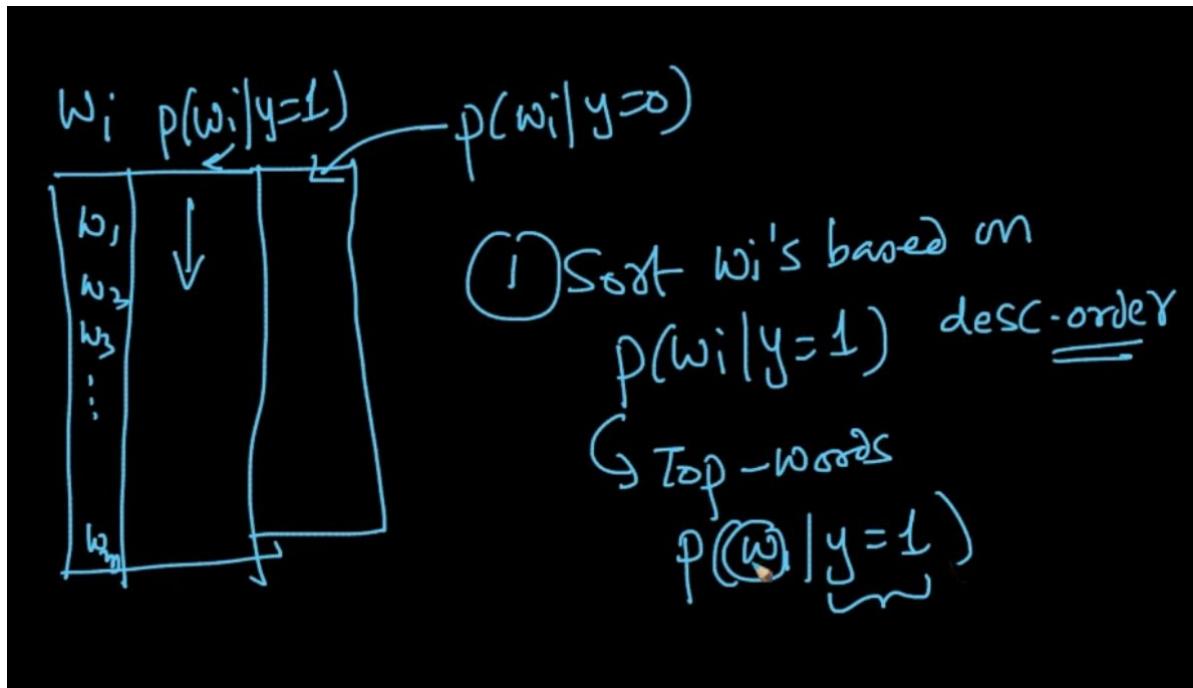
Positive class points > Negative class points then for every query  $x_q$  it'll predict  $y_q = 1$   
which is the same as underfitting in k-NN



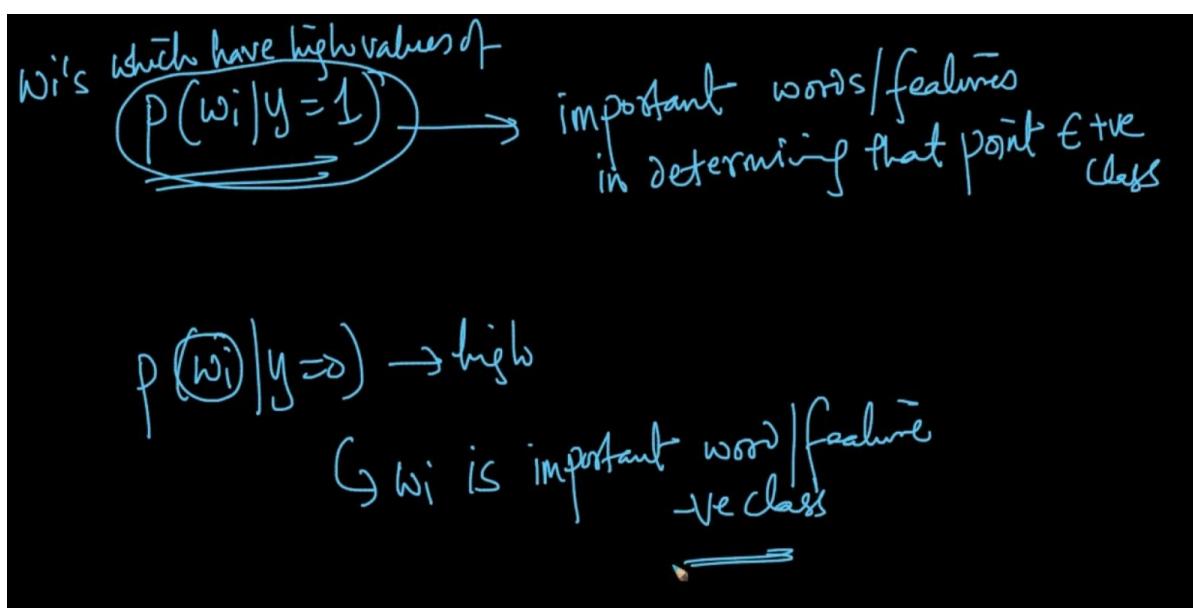
So we need to find the right  $\alpha$ . So, to find it we do Cross Validation as  $\alpha$  is a  
hyper-parameter and hyper-parameters are decided using Cross Validation(CV)

## FEATURE IMPORTANCE AND INTERPRETABILITY

### FEATURE IMPORTANCE



decide the most important features by Sorting w based on their probability scores



That's how most important features are decided

feature importance in NB:

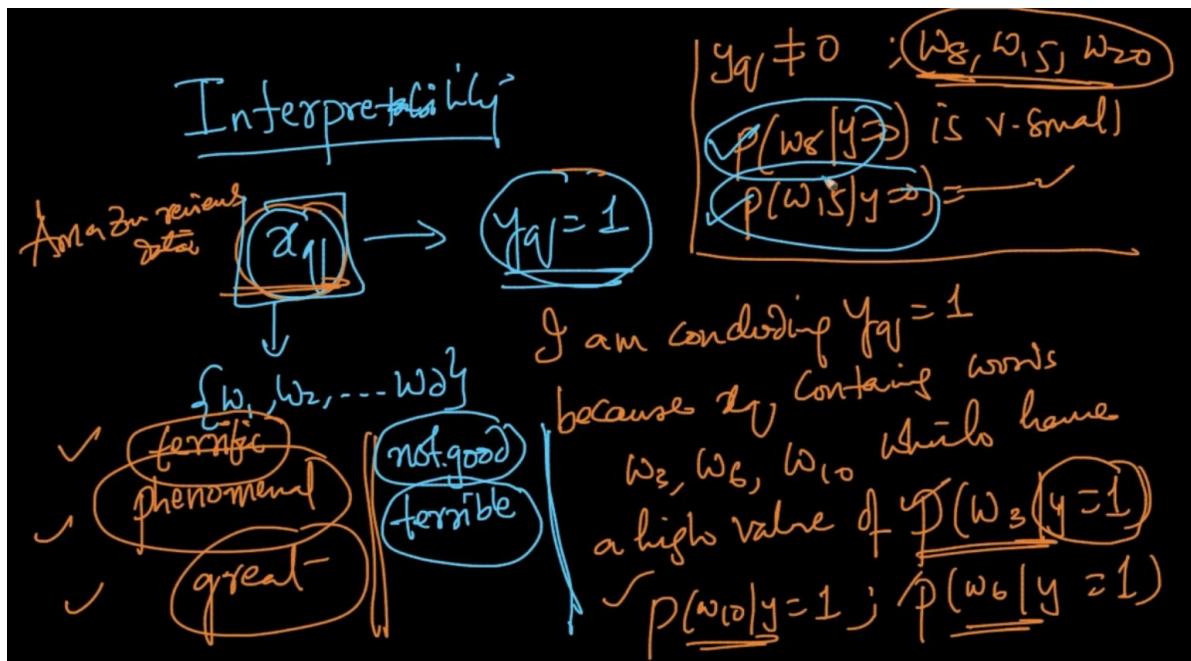
{ +ve class :- find words ( $w_i$ ) with highest value of  
 $P(w_i | y=1)$  ✓

✓ { -ve class : "  $P(w_i | y=0)$  ✓

② determined/obtained directly from the model

In k-NN, we used Forward Feature selection to select the most important features. Here, we can directly get most important features from model itself since Probabilities are calculated at the learning stage itself

### INTERPRETABILITY



Suppose if we've a positive review. We can conclude it's  $y = 1$  because  $x_q$  contains words  $w_3, w_5, w_8$  like *terrific, great* which has high probability score and also conclude  $y \neq 0$   
the probability scores of negative words  $w_7, w_9$  like *terrible, not-good* are small

## OUTLIERS

Outliers :-

NB Text-classf'n:-

outlier at test time  $\rightarrow \frac{\lambda_{qj} = w_1, w_2, w_3, w'}{w' \notin \{w_1, w_2, \dots, w_m\}}$   $\leftarrow$  Set of words that I have seen in D<sub>train</sub>

Laplace Smoothing

$$P(w/y=1) = \frac{o+\alpha}{n_1+2\alpha}$$

If a word  $w'$  occurs at test time which wasn't seen in Train Data then we do Laplace smoothing to deal with it. It's an outlier

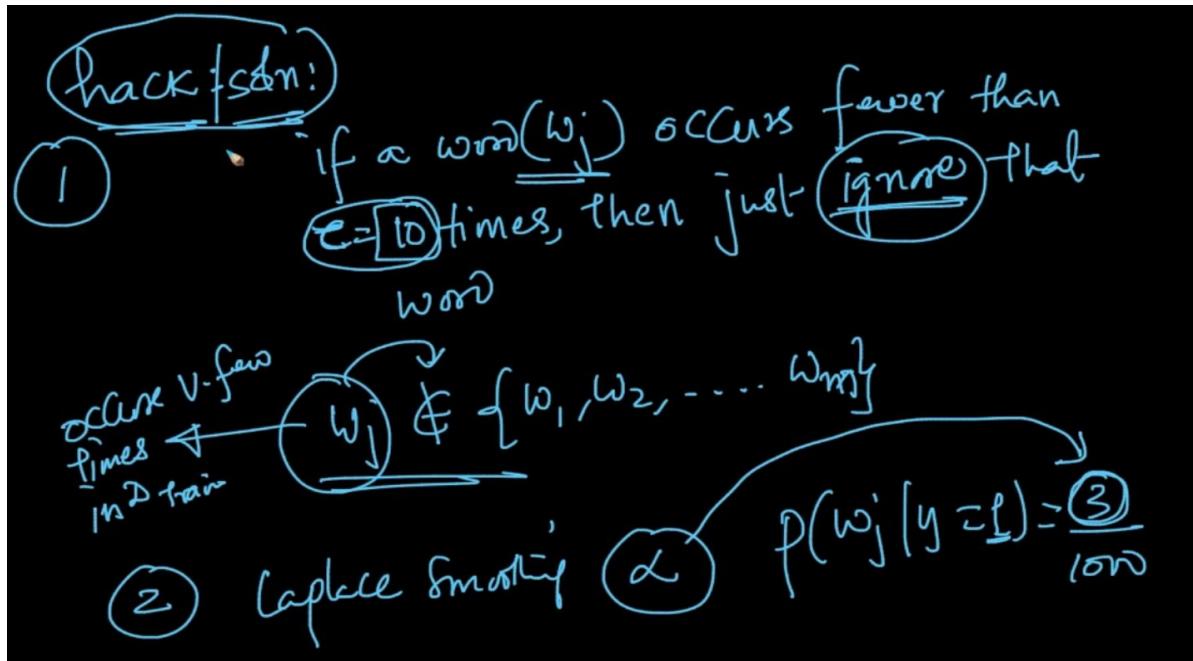
Outliers in Training Data:

$\{w_1, w_2, w_3, \dots, w_m\} \leftarrow$  Set of words in D<sub>train</sub>

$w_8 \rightarrow$  occurs very few times in +ve & -ve classes

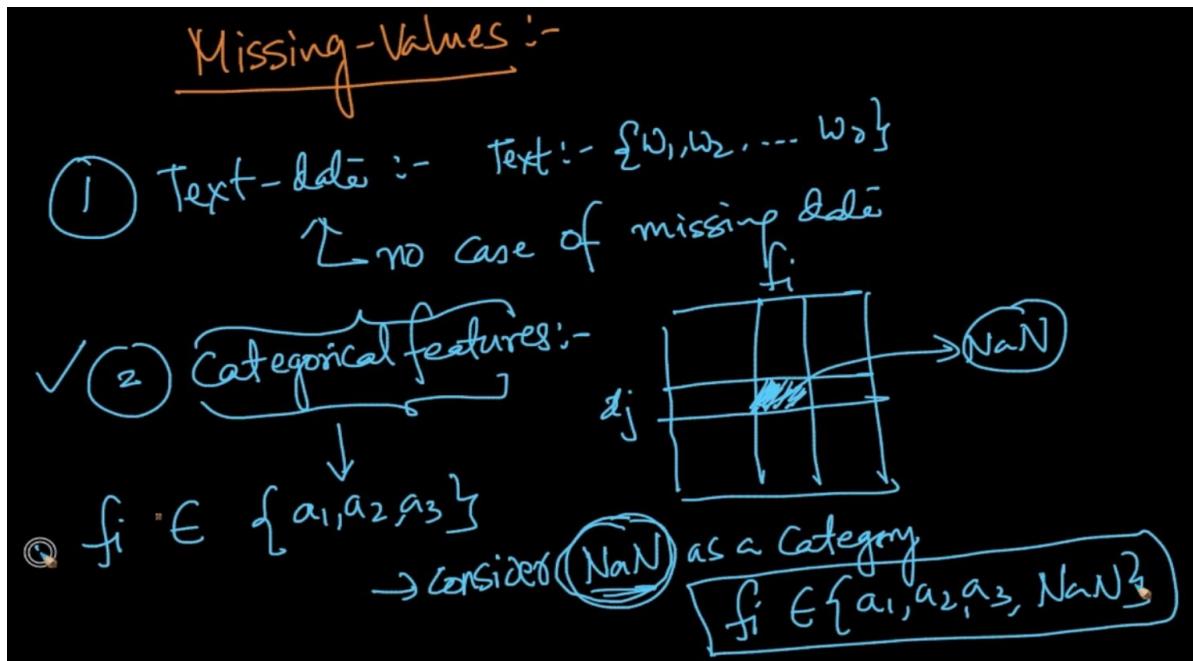
outlier

If a word  $w$  occurs very few times in our Train data then it's an outlier. How do we deal with it?



- 1) We can apply a condition/ threshold to just ignore that word as seen above
- 2) We can use Laplace Smoothing in our Train Data as well with the right  $\alpha$

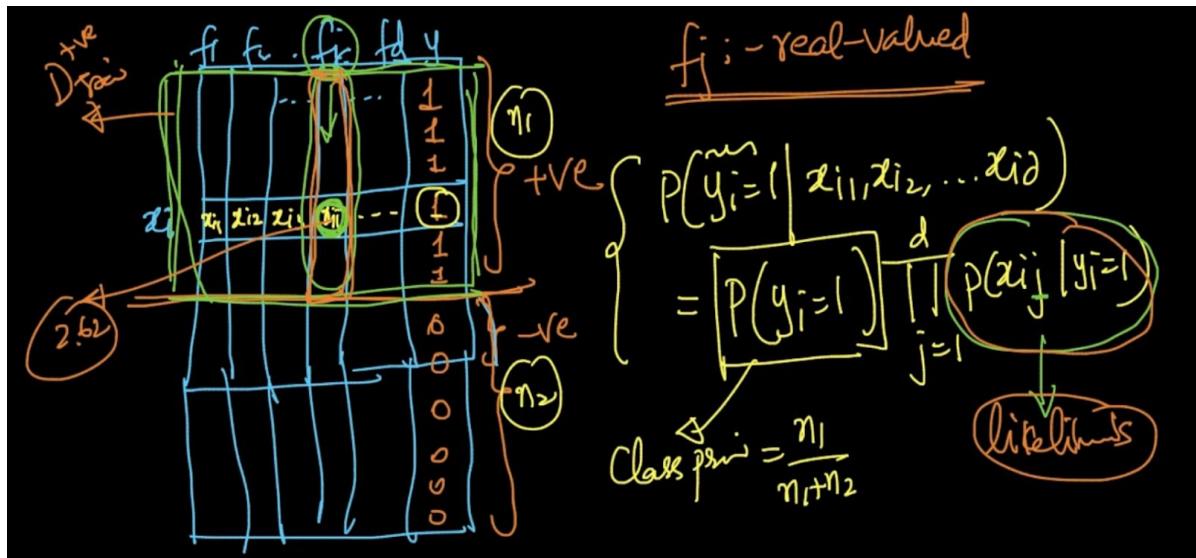
## MISSING VALUES



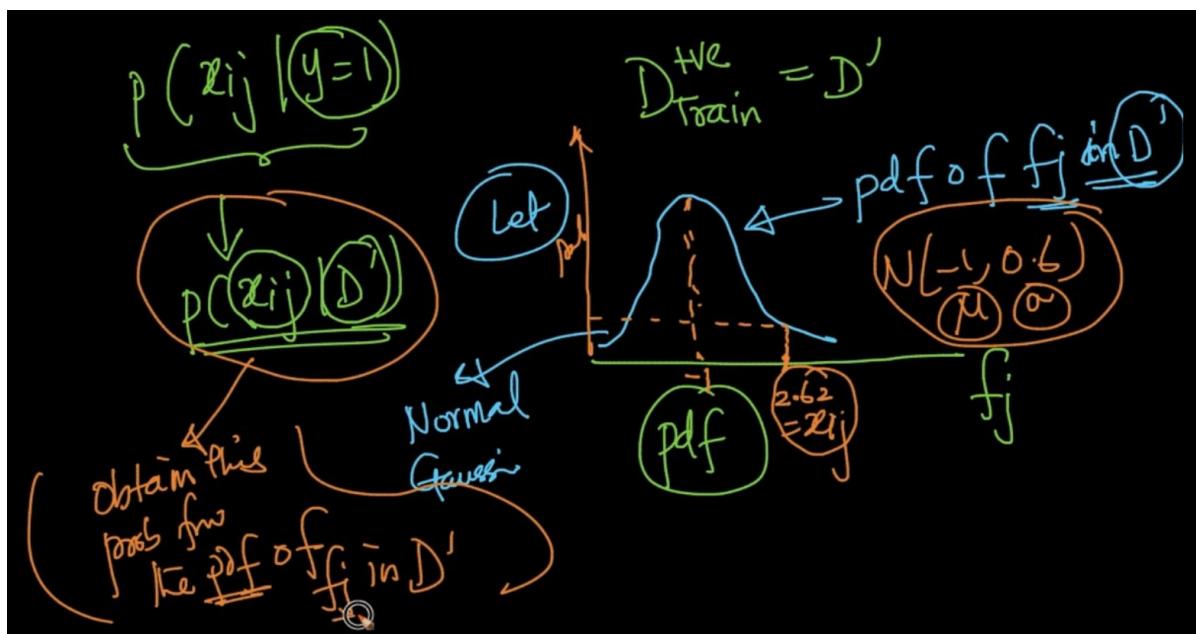
For Categorical features if it doesn't belong to features  $\{a_1, a_2, a_3\}$  e.g  $\text{NaN}$  then we also consider  $\text{NaN}$  as a category. If missing values are there in Numerical data then Imputation



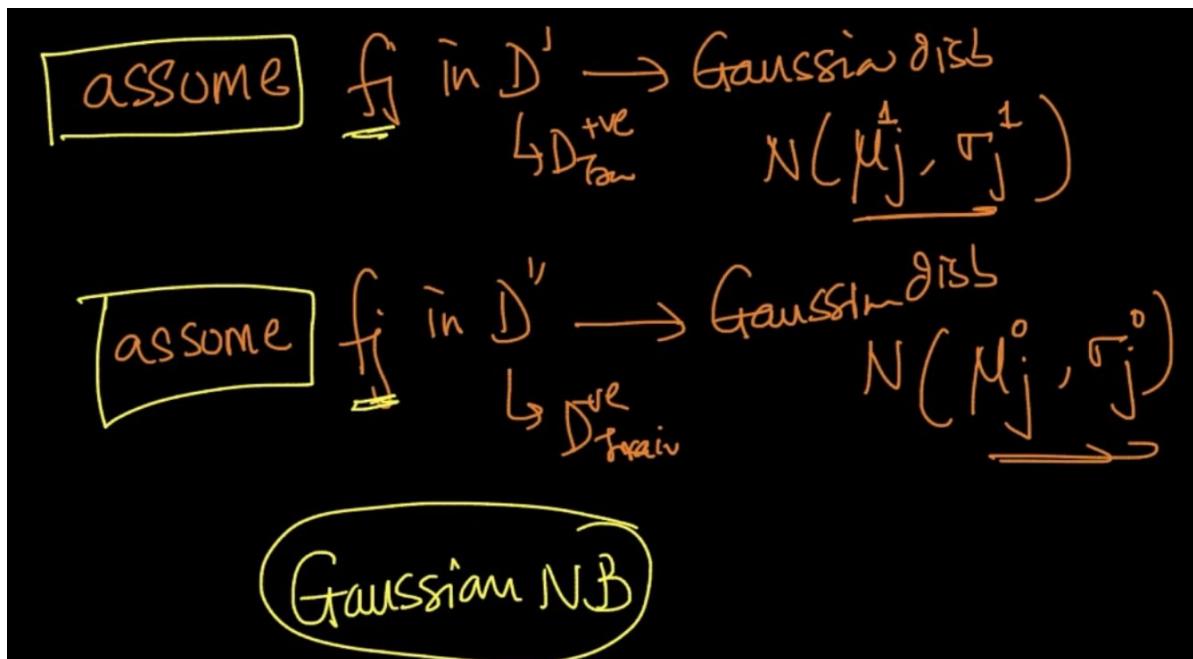
## RUSSIAN NAIVE BAYES



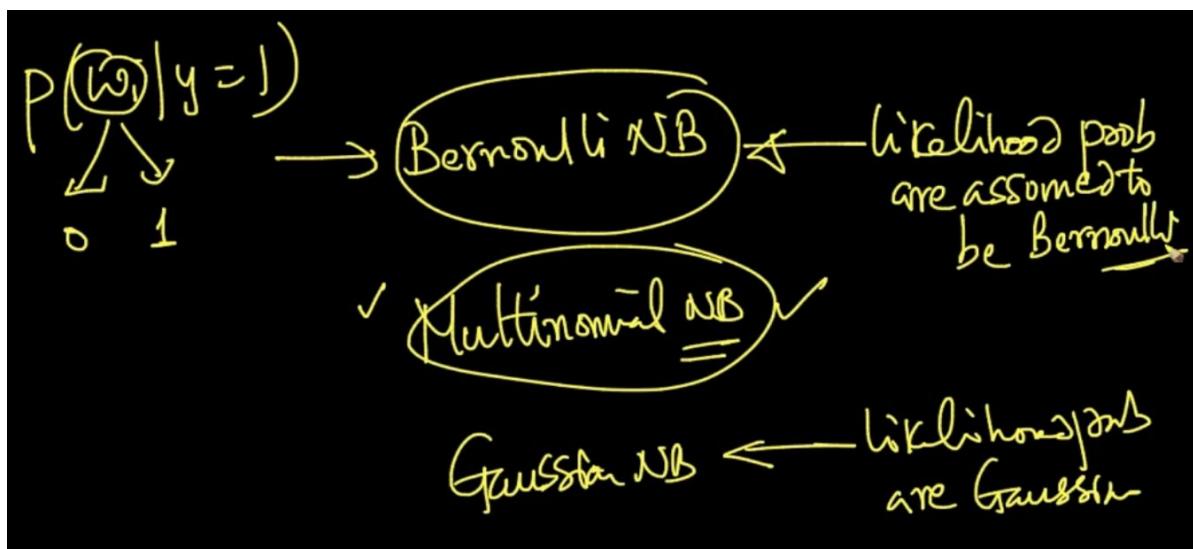
We are seeing Naive Bayes for numerical data. First we'll divide the data into positive and negative classes .  $P(y = 1)$  is easy to calculate.  $P(x_{ij} | y = 1)$  is difficult to calculate .It's our likelihood. It is probability of feature j given our class label



We can predict it by plotting the PDF of a feature j with  $y = 1$ . Ex :  $x_{ij} = 2.62$  then we can get the probability value by plotting PDF of j since PDF has Probability scores in y-axis



Here we are assuming that our features follows Gaussian Distribution. Therefore, this is called Gaussian Naive Bayes



There can be other Naive Bayes as well. Ex: If our random variable  $x$  has binary values then it's following Bernoulli Distribution

NOTE - Naive Bayes' big assumption is that all the features are conditionally independent i.e the features aren't dependent on each other

## MULTICLASS CLASSIFICATION

Multi-class Classfn:

NB → can do multi-class classfn c-classes

Compare  $\left\{ \begin{array}{l} p(y_i=1 | w_1, w_2, \dots, w_d) \\ p(y_i=0 | w_1, w_2, \dots, w_d) \\ p(y_i=2 | w_1, w_2, \dots, w_d) \\ \vdots \\ p(y_i=c-1 | w_1, w_2, \dots, w_d) \end{array} \right\}$  →  $p(y_i=b | w_1, w_2, \dots, w_d)$  is the largest  $y_i = b$  given  $x_i = \{w_1, w_2, \dots, w_d\}$

Naive Bayes can do Multiclass Classification pretty well. If we've c-classes then we'll calculate probability scores for all the classes and select the one class which has largest probability in it

## SIMILARITY MATRIX OR DISTANCE MATRIX

✓ Similarity matrix or Distance Matrix

NB → classfn → distance-based method  
→ probability based method

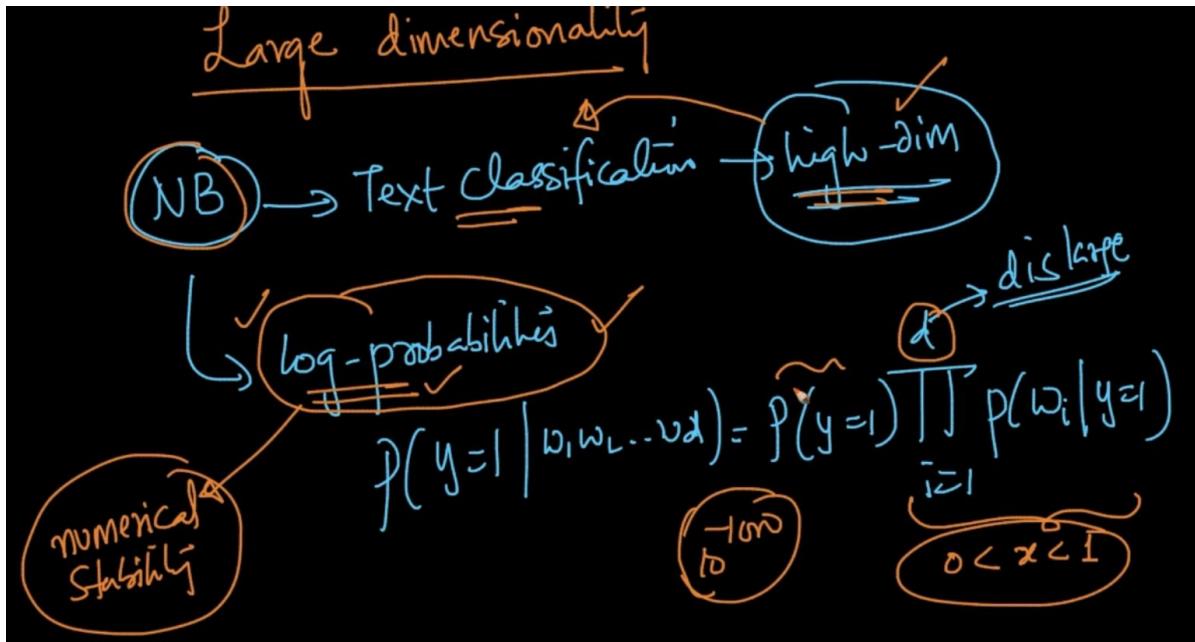
$x_i \rightarrow \boxed{\boxed{\boxed{\quad}}}$  → Distance / Similarity - matrix ← KNN

→ Cannot use dist / sim matrices

$\hookrightarrow p(y_i=1 | f_1, f_2, \dots) = p(y_i=1) \prod_{j=1}^n p(f_j | y_i=1)$

We cannot use distance/similarity matrices in NB as classification is done here on a probabilistic based method. So we need the exact value of the feature unlike k-NN which is a distance-based method which could use Distance matrix to do classification

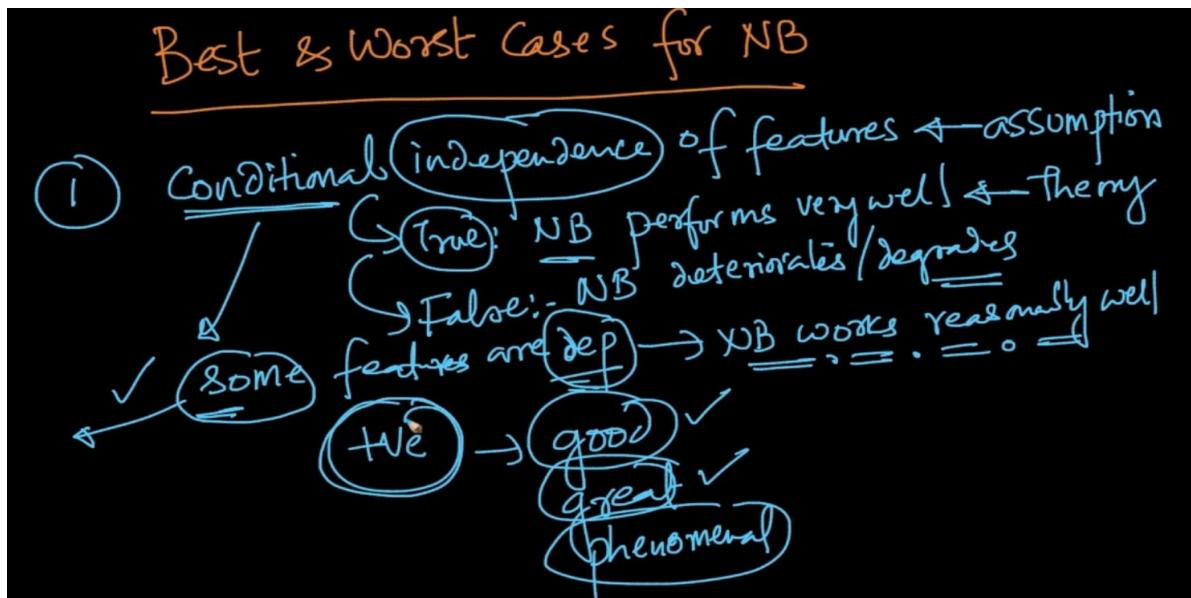
## LARGE DIMENSIONS



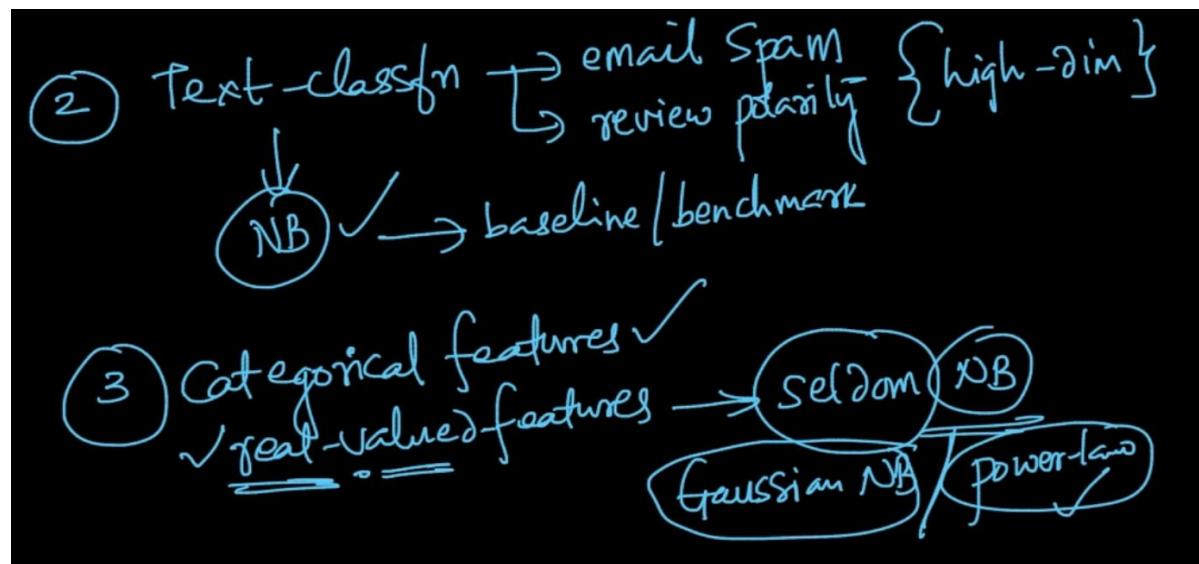
Naive Bayes can do very good in Large Dimensionality but use Log-probabilities instead of normal probabilities to avoid numerical underflow



## BEST AND WORST CASES FOR NAIVE BAYES

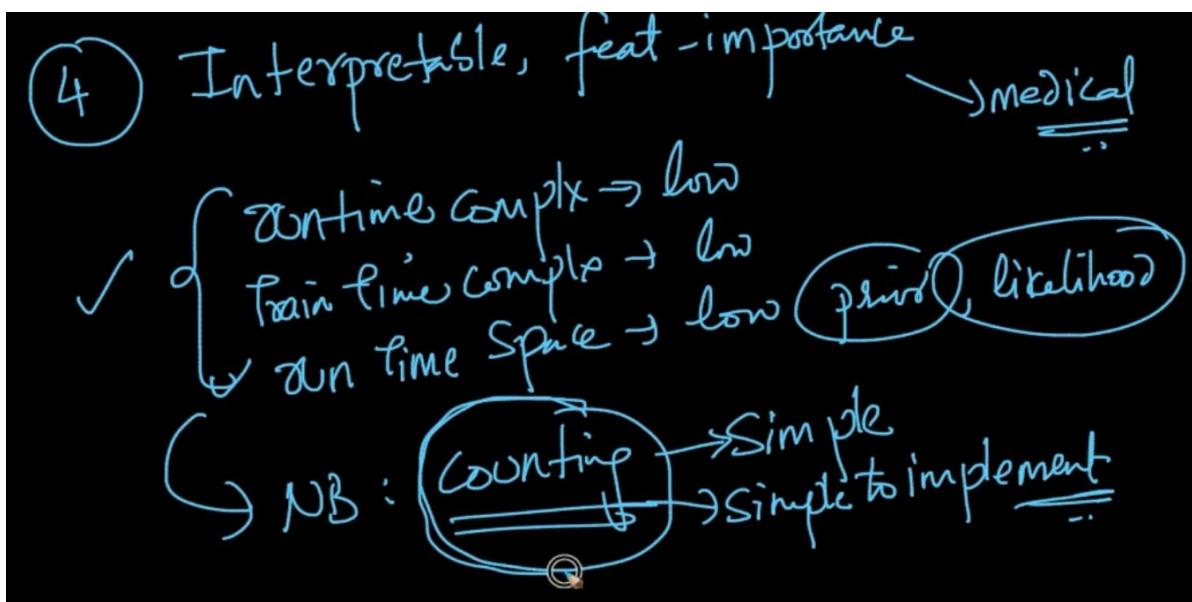


We are assuming that our features are conditional independent. Naive Bayes performs well in conditional independent features but if there is dependence then it deteriorates but if some features are dependent for ex : In review classification words are dependent like if word *good* is there then *great* word can also occur in that NB performs reasonably good



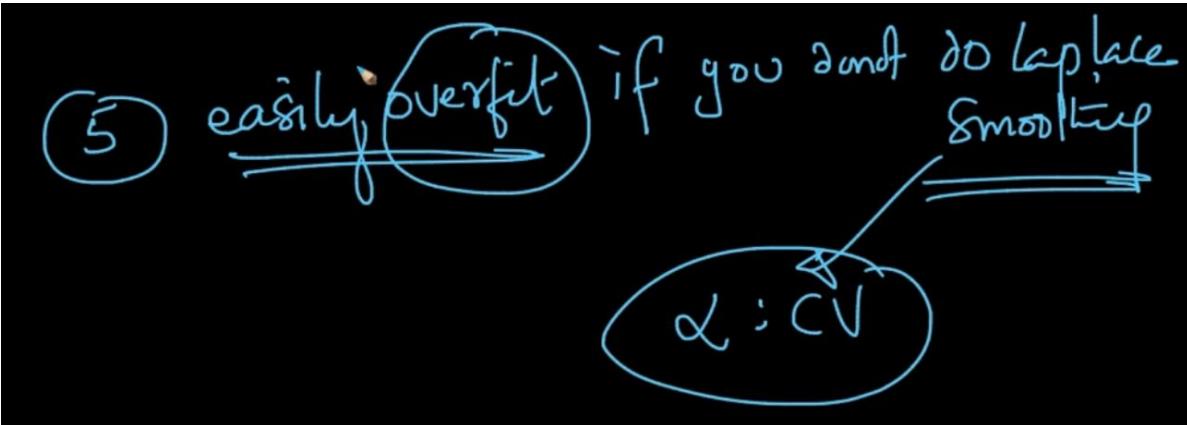
NB performs very well in Text Classification. For categorical features as well but seldom used for real valued features

④



It is very interpretable. Since it's just counting it is simple to implement and all the complexities are low.

⑤



Laplace Smoothing is necessary