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)

D1						
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

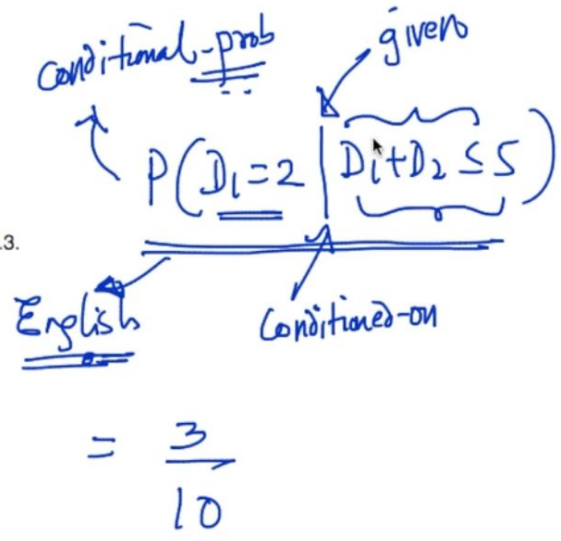
***What is the probability that* D1 = *2* given that D1+D2 ≤ *5* ?**

Table 3 shows that for 3 of these 10 outcomes, *D1* = 2.

Thus, the conditional probability $P(D1=2 \mid D1+D2 \le 5) = {}^{3}/_{10} = 0.3$.

Table 3

+		D2					
		1	2	3	4	5	6
D1	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



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 $D_1 + D_2 \le 5$.

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

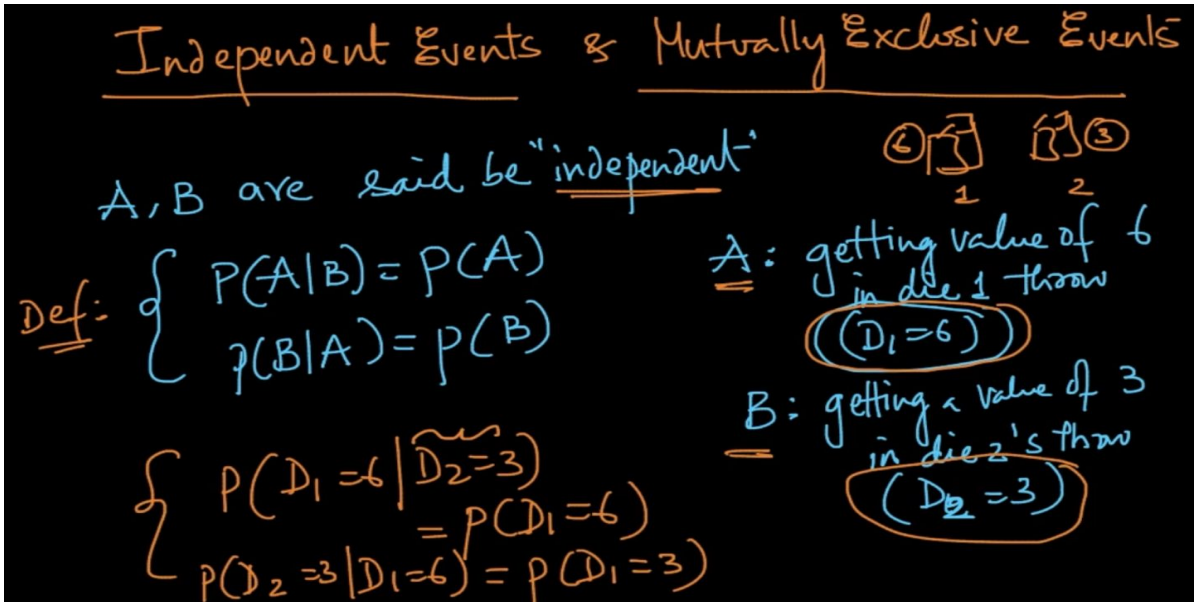
Definition :- P (A | B) = $\frac{P(A \cap B)}{P(B)}$ here A : $D_1 = 2$ and B : $D_1 + D_2 \le 5$

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

Conditional probability - Wikipedia

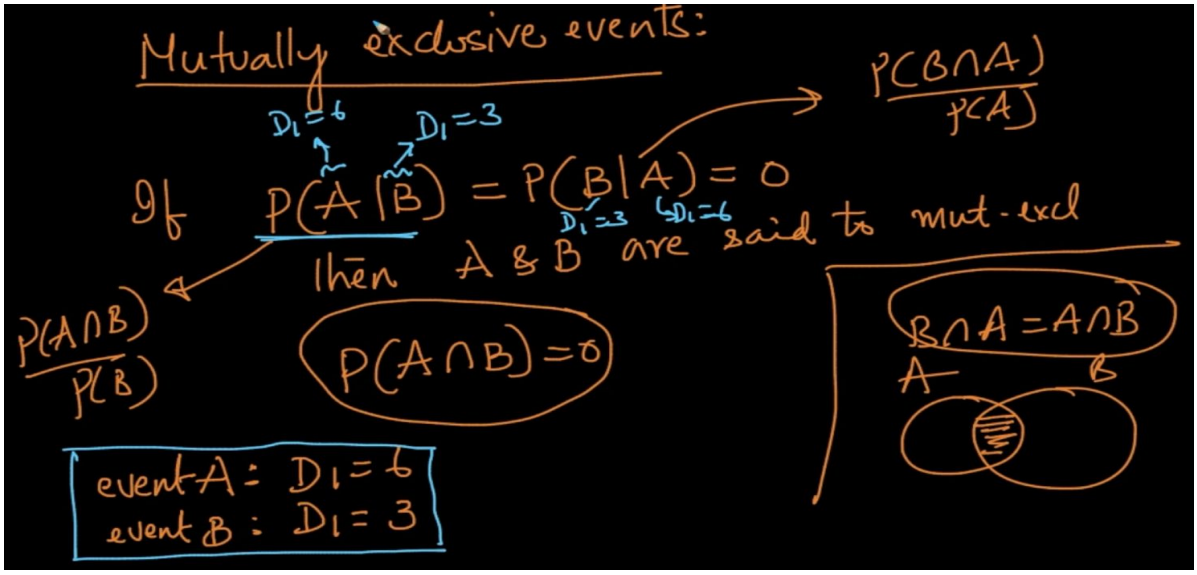
INDEPENDENT AND MUTUALLY EXCLUSIVE EVENTS

INDEPENDENT EVENTS



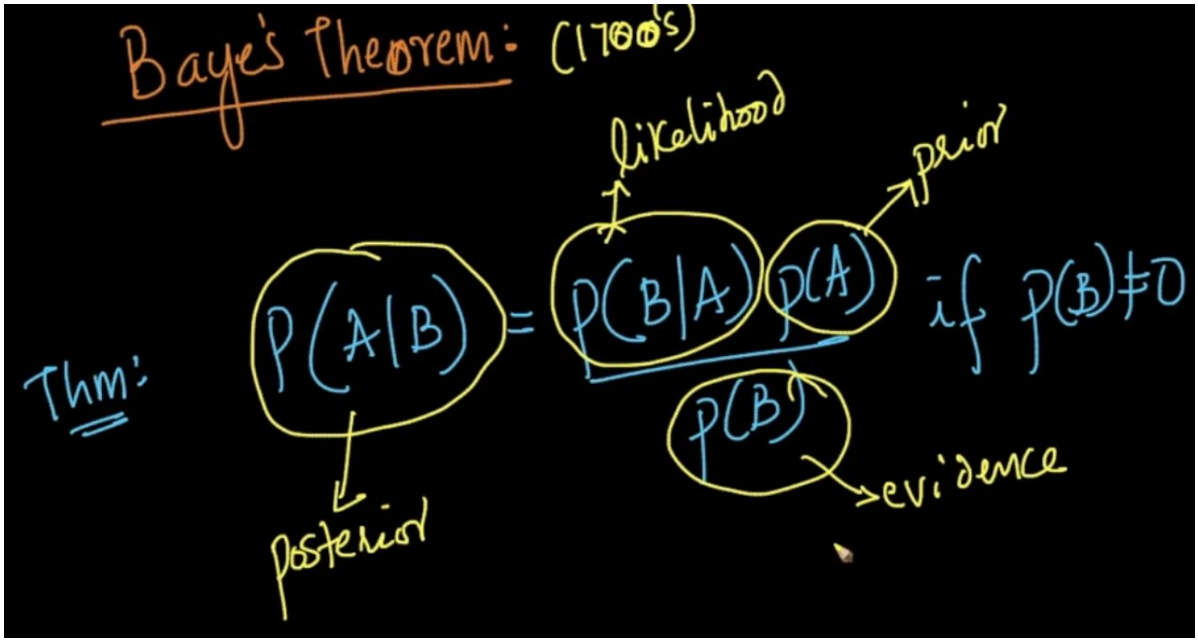
A, B are said to be independent if a probability of event A is not dependent on event B

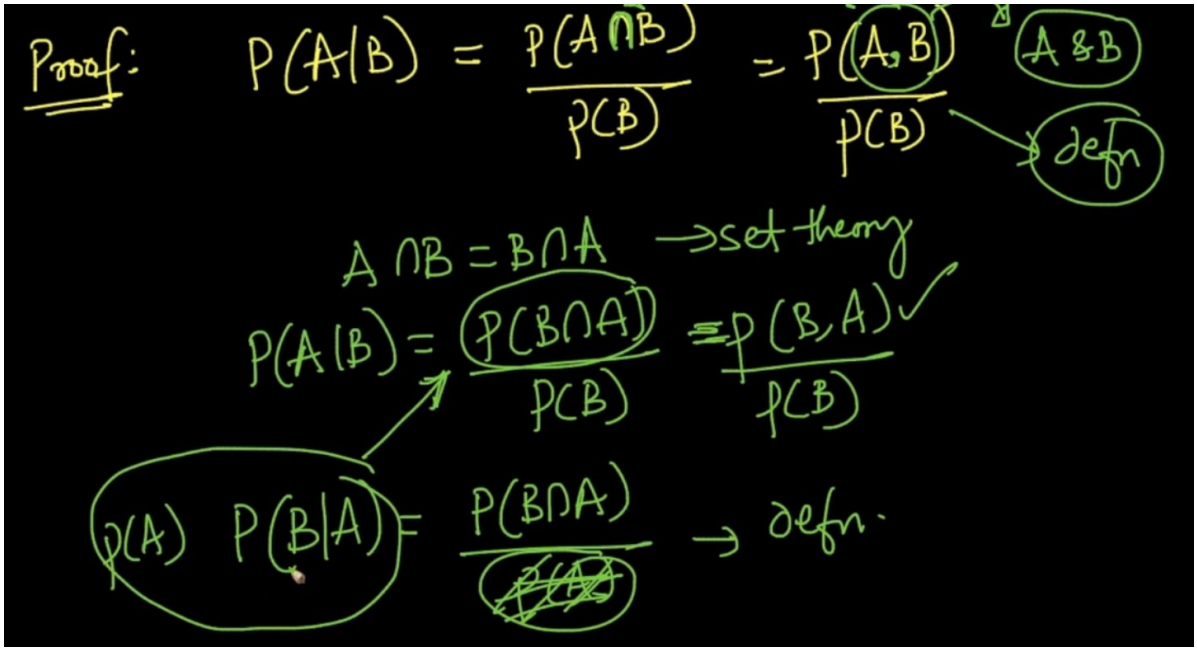
Take the example of 2 dices above thrown.Their outcomes are independent obviously



In P (A | B) = 0 because if it's given D1 = 3 (B) then there's no chance of A : D1 = 6 happening .Vice versa is also true here. So, they are mutually exclusive events P(A ∩ B) =0

BAYES THEOREM

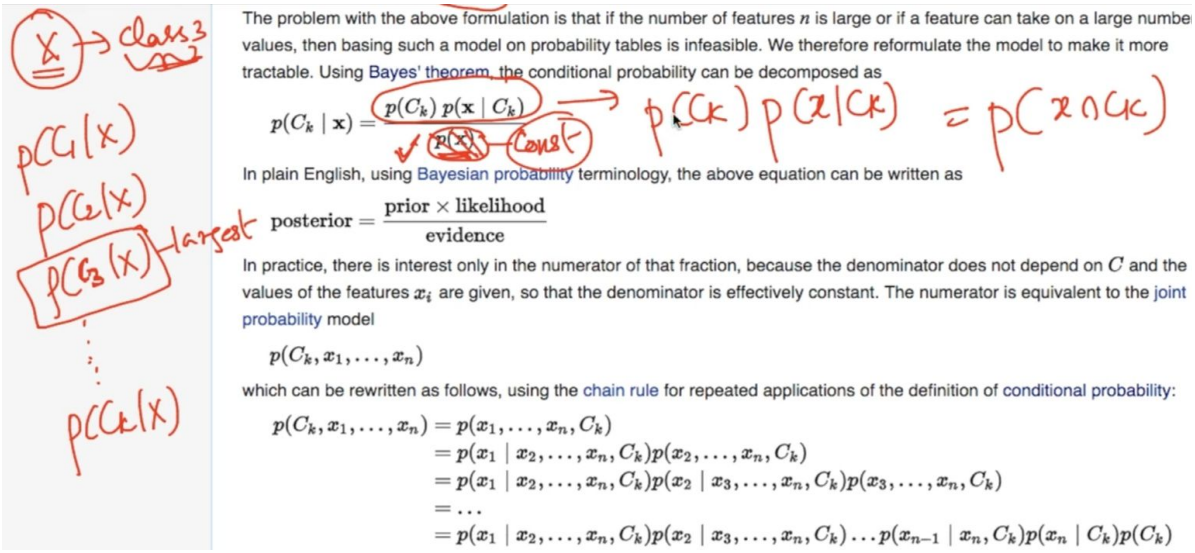




Here, P (A | B) = $\frac{P(B \cap A)}{P(B)}$ and P(B | A) = $\frac{P(B \cap A)}{P(A)}$ and it'll be P(B ∩ A) = P (A). P(B | A)

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

IVE 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 numbe 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

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\ p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$

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

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

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

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

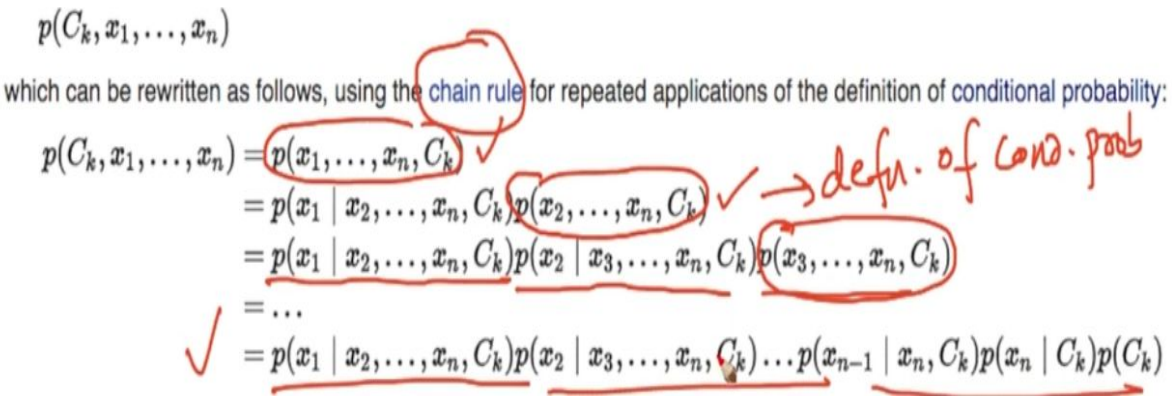
1) Naive Bayes is based on Bayes' theorem . Here we are given x and we want to predict class label C_k i.e P($C_k \mid x$).

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

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

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

p(C_k) p(x | C_k) => p($C_k \cap x$) = p(C_k, x1,x2…..xn)

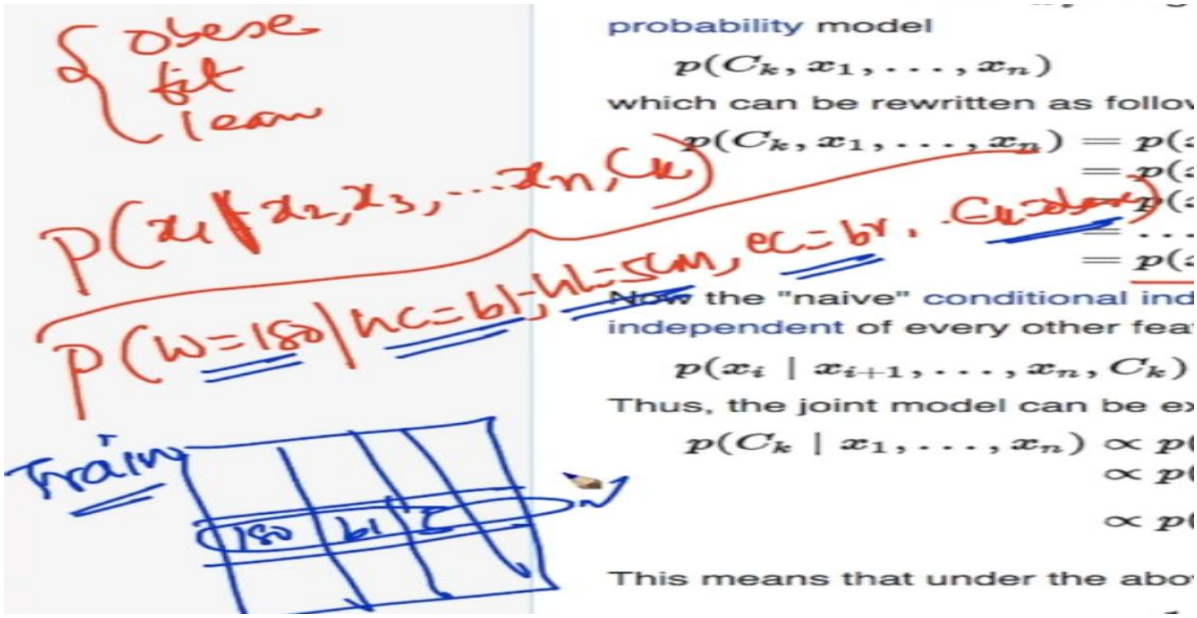


$$p(C_k, x_1, \ldots, x_n)$$

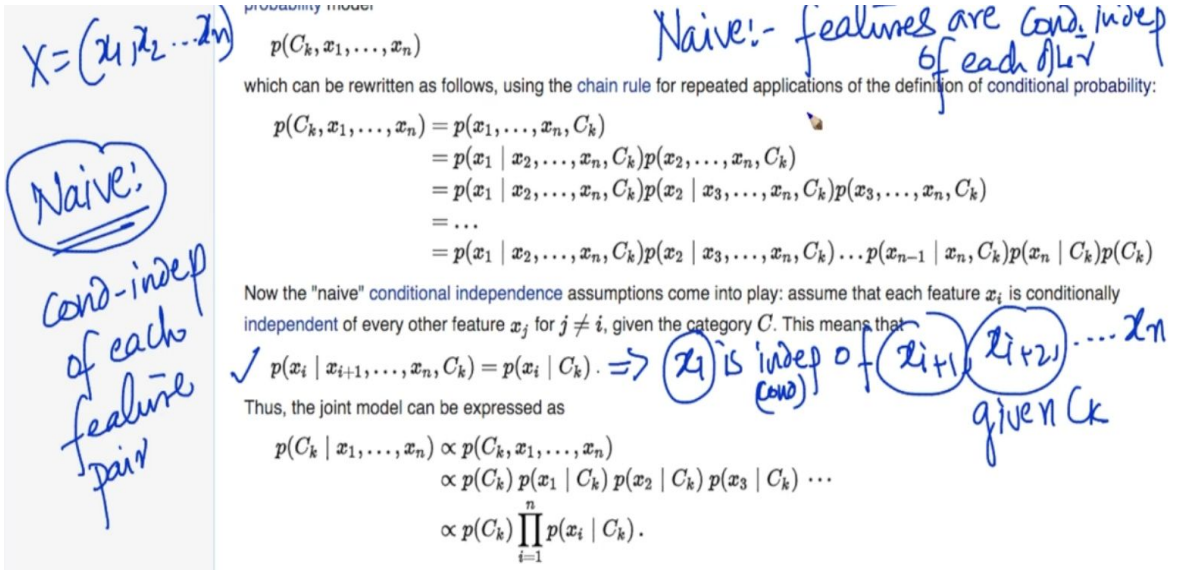
which can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability:

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

For every x probability is calculated



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



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($x_1 | x_2, x_3 .... C_k$) = P($x_1 | C_k$)

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

Thus, the joint model can be expressed as

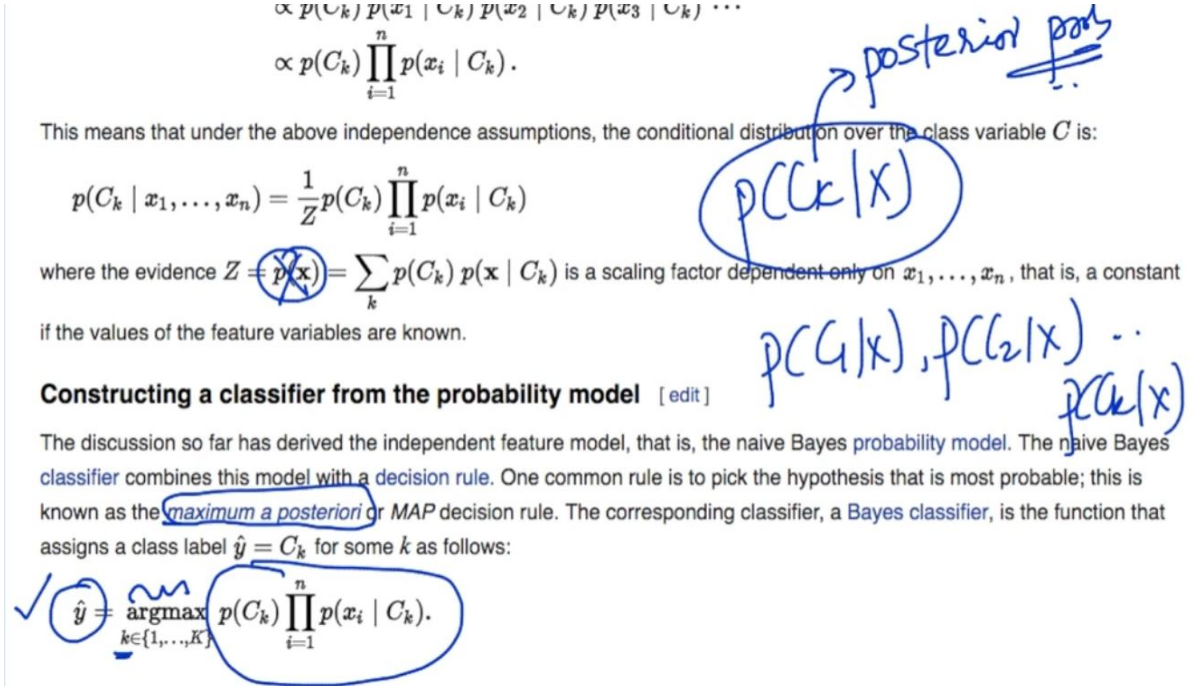
$$\begin{aligned} p(C_k \mid x_1, \ldots, x_n) &\propto p(C_k, x_1, \ldots, x_n) \\ &\propto p(C_k)\, p(x_1 \mid C_k)\, p(x_2 \mid C_k)\, p(x_3 \mid C_k)\, \cdots \\ &\propto p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)\,. \end{aligned}$$

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

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

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

So P($C_k \mid x$) ∝ p(C_k) p(x | C_k) where x = x1,x2,x3...xn can be represented like above $\prod$ stands for multiplication just like $\sum$



$$\propto p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)\,.$$

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

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

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

Constructing a classifier from the probability model [edit]

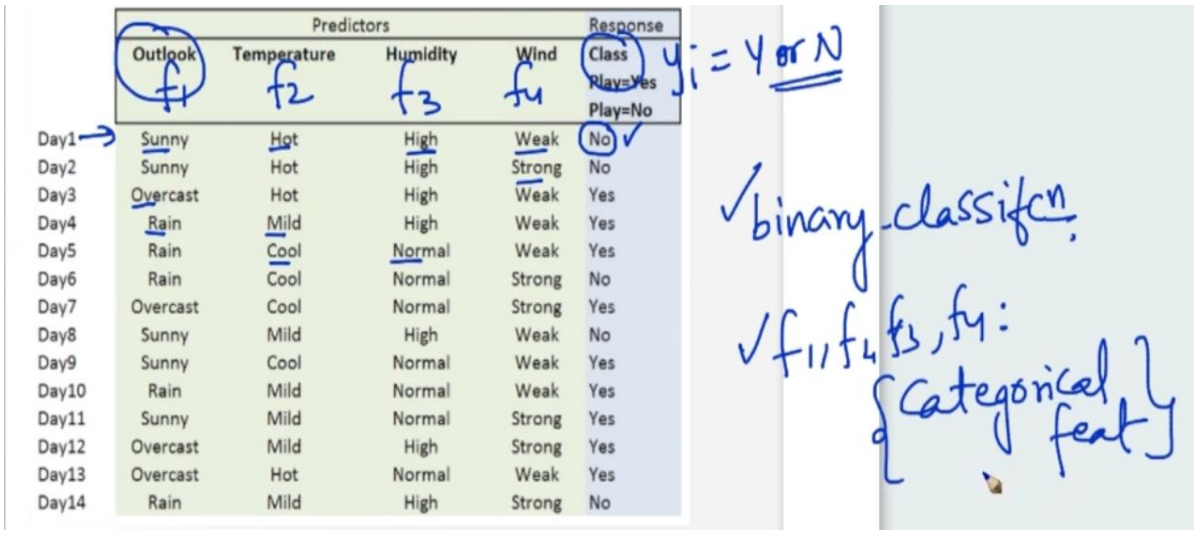
The discussion so far has derived the independent feature model, that is, the naive Bayes probability model. The naive Bayes classifier combines this model with a decision rule. One common rule is to pick the hypothesis that is most probable; this is known as the *maximum a posteriori* or *MAP* decision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

$$\hat{y} = \underset{k \in \{1, \ldots, K\}}{\operatorname{argmax}} \; p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k).$$

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

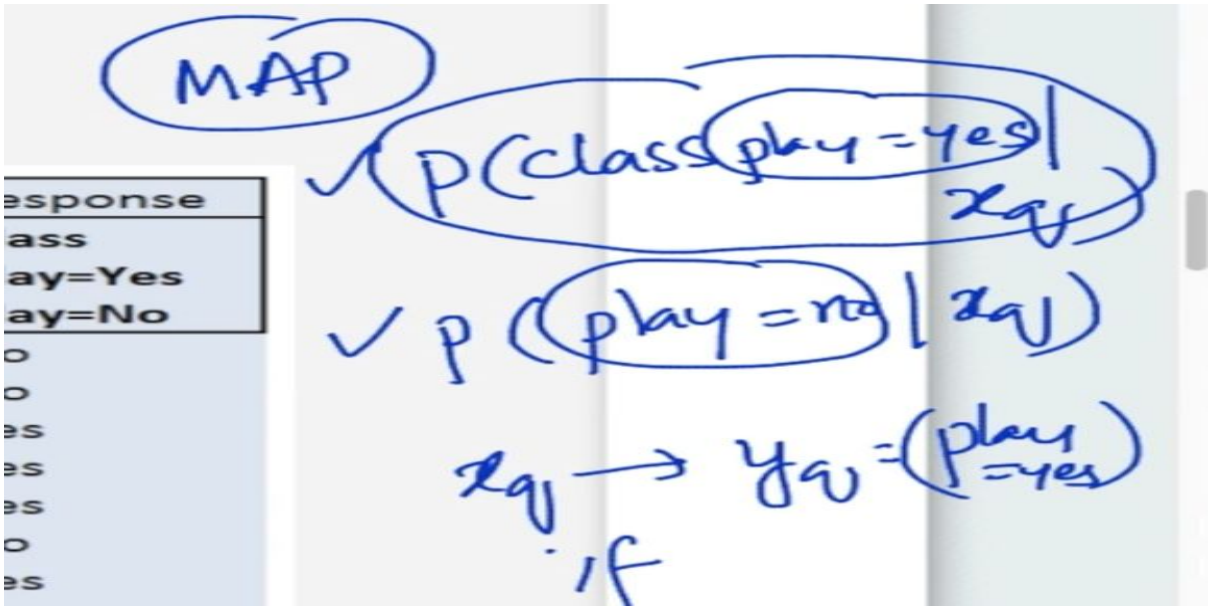
[Naive Bayes classifier - Wikipedia]

TOY EXAMPLE - NAIVE BAYES



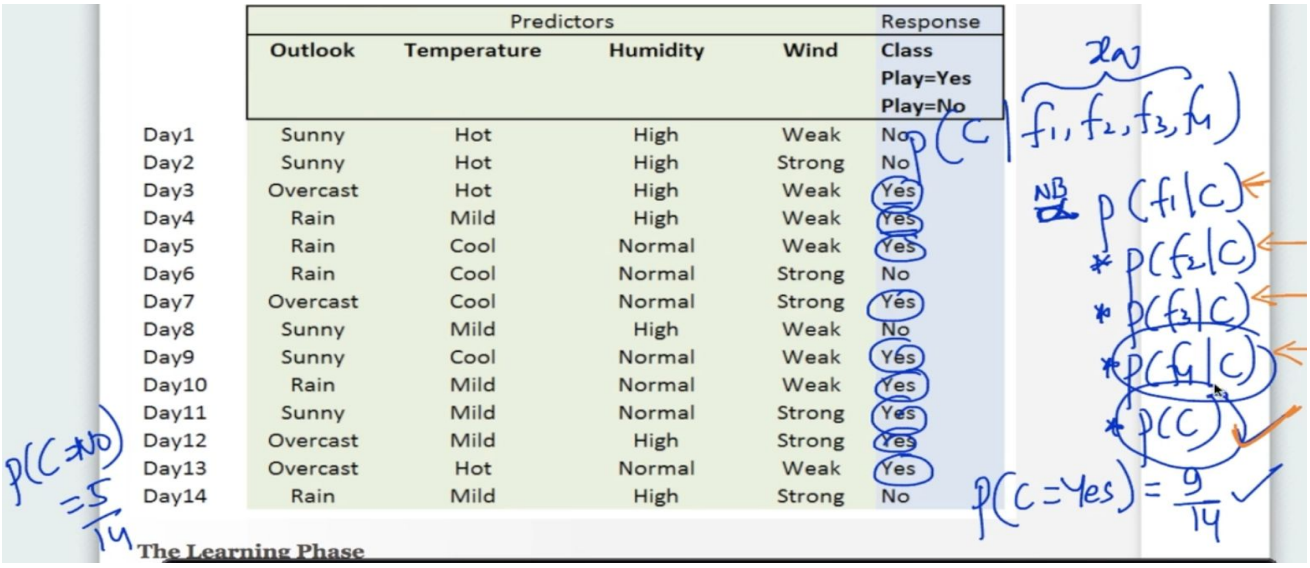
	Predictors				Response
	Outlook (f1)	Temperature (f2)	Humidity (f3)	Wind (f4)	Class (Play=Yes / Play=No)
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	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

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 lihoods?They are probability of a feature given a class label.

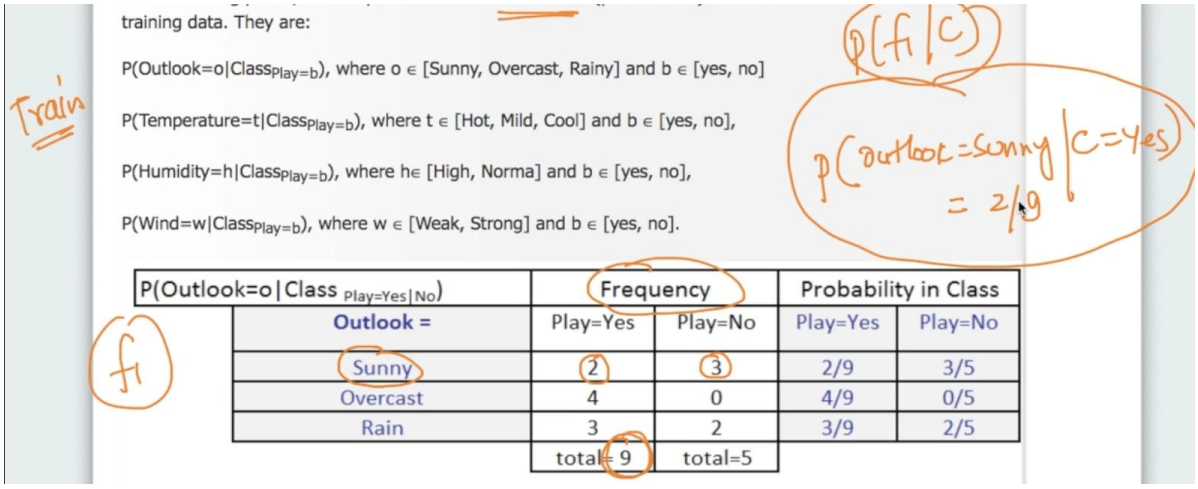


	Predictors				Response
	Outlook	Temperature	Humidity	Wind	Class Play=Yes Play=No
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8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

The Learning Phase

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 = Yes) = Total no. of yes in class / Total no. of class labels = 9/14.



training data. They are:

P(Outlook=o|Class$_{Play=b}$), where o ∈ [Sunny, Overcast, Rainy] and b ∈ [yes, no]

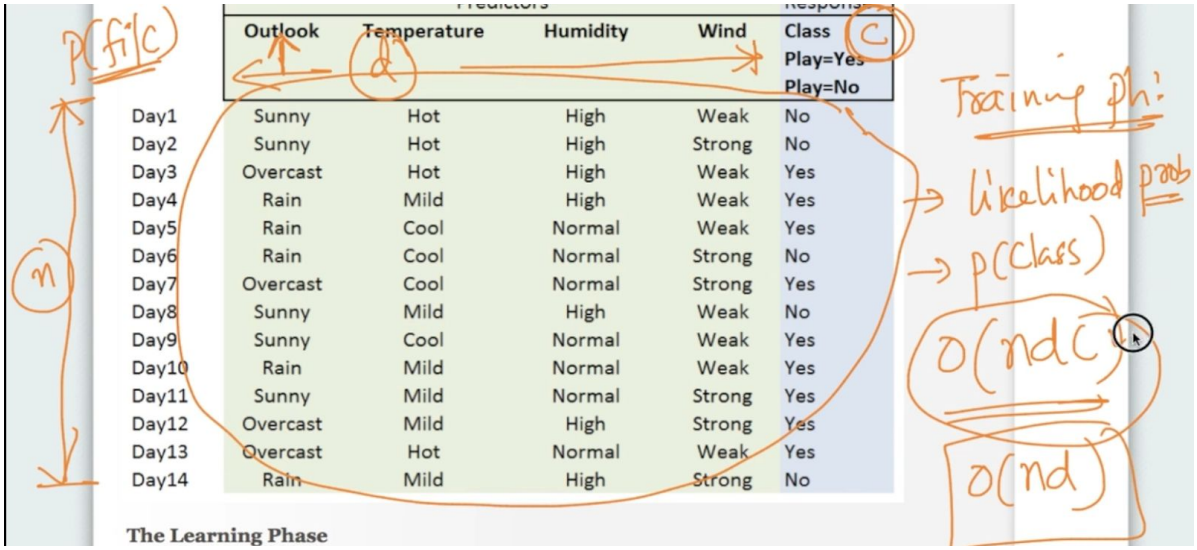
P(Temperature=t|Class$_{Play=b}$), where t ∈ [Hot, Mild, Cool] and b ∈ [yes, no],

P(Humidity=h|Class$_{Play=b}$), where h∈ [High, Norma] and b ∈ [yes, no],

P(Wind=w|Class$_{Play=b}$), where w ∈ [Weak, Strong] and b ∈ [yes, no].

P(Outlook=o|Class $_{Play=Yes\|No}$)	Frequency		Probability in Class	
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		

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

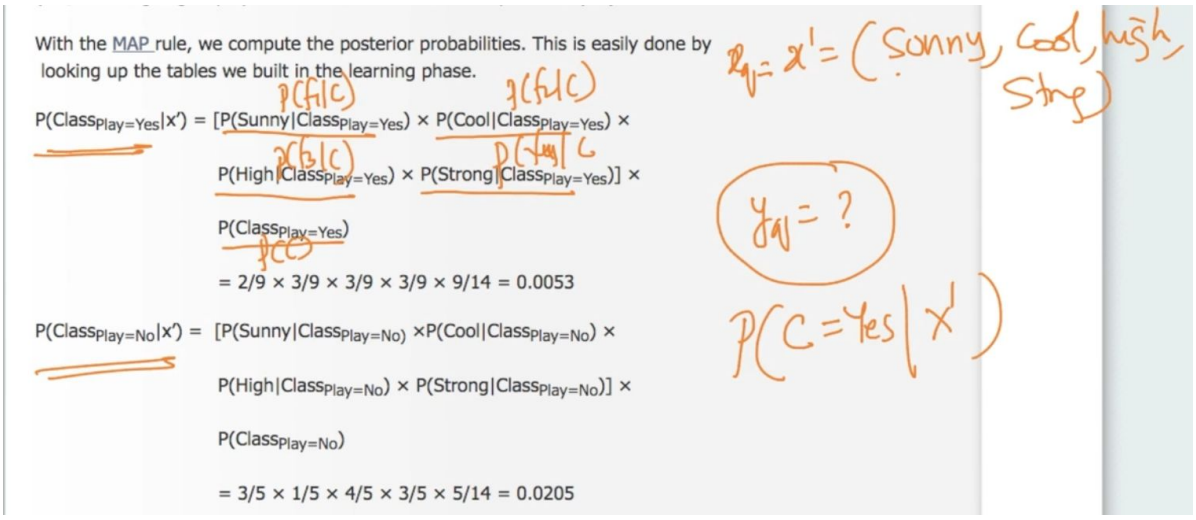


	Outlook	Temperature	Humidity	Wind	Class Play=Yes / Play=No
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	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

The Learning Phase

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.

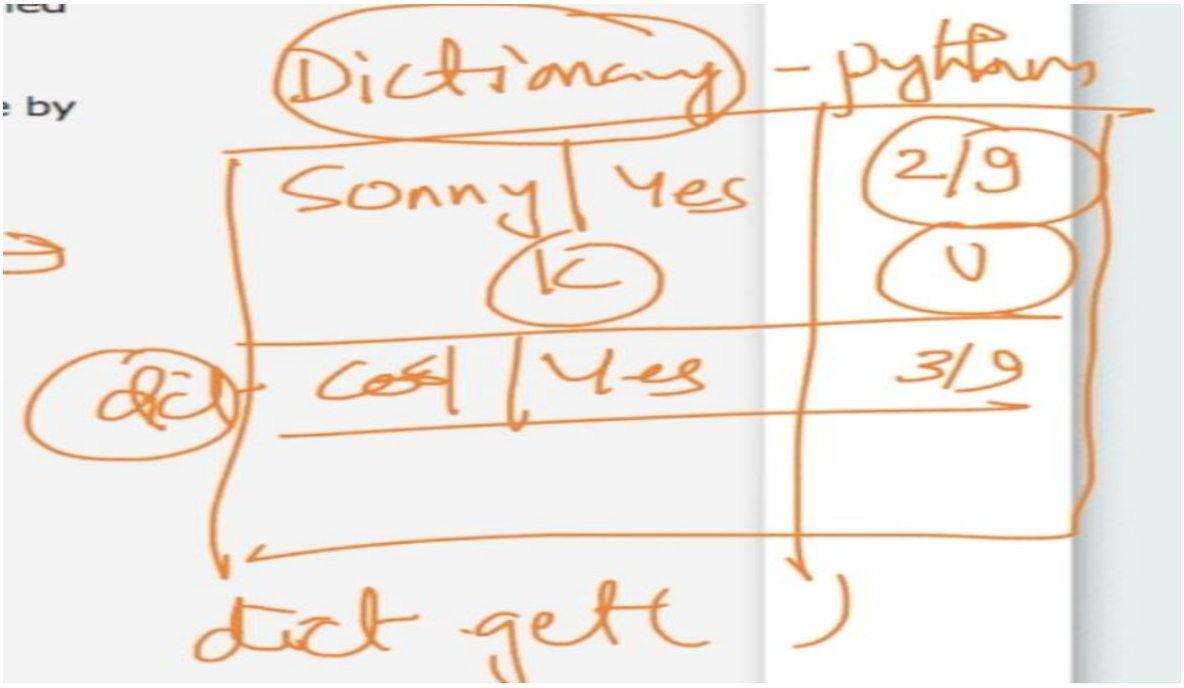
$$P(Class_{Play=Yes}|x') = [P(Sunny|Class_{Play=Yes}) \times P(Cool|Class_{Play=Yes}) \times P(High|Class_{Play=Yes}) \times P(Strong|Class_{Play=Yes})] \times P(Class_{Play=Yes})$$

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

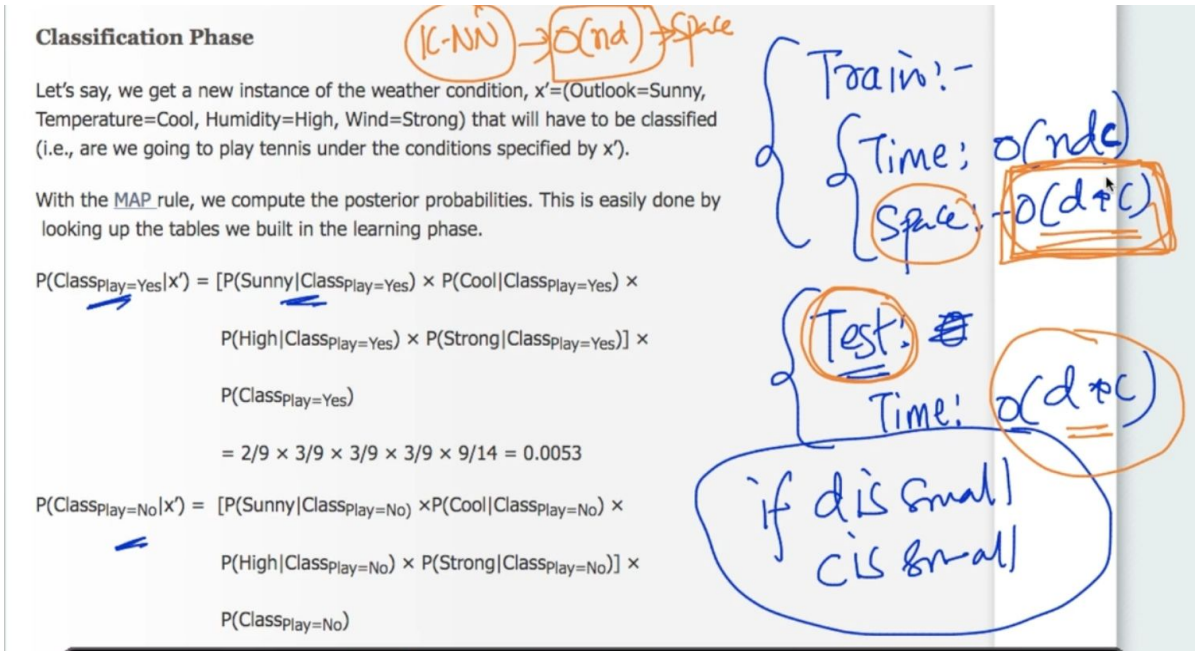
$$P(Class_{Play=No}|x') = [P(Sunny|Class_{Play=No}) \times P(Cool|Class_{Play=No}) \times P(High|Class_{Play=No}) \times P(Strong|Class_{Play=No})] \times P(Class_{Play=No})$$

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

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



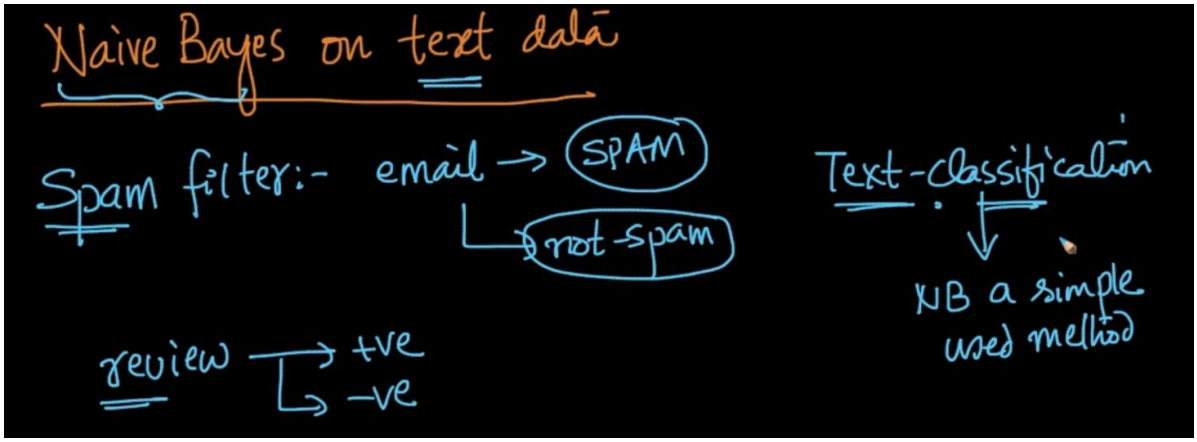
Look up table with key = Query (f | C) and Value = Probability score



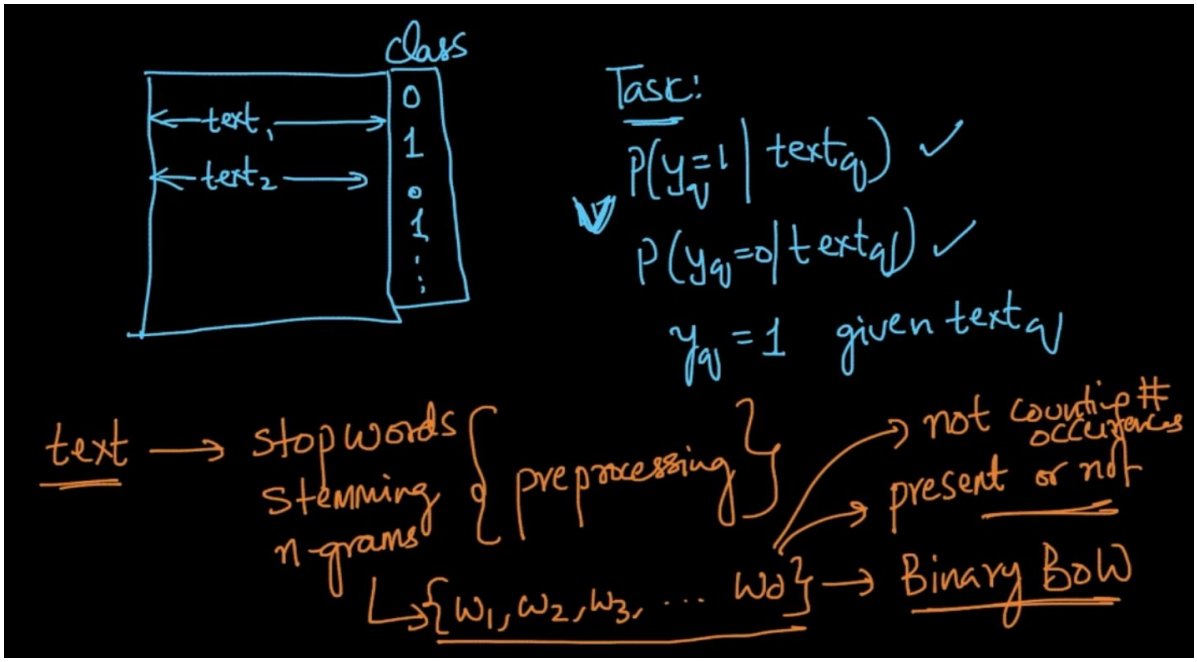
Space Complexity : O(d * C) here d = 4 and classes = 2 so only 8 values in lookup table

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



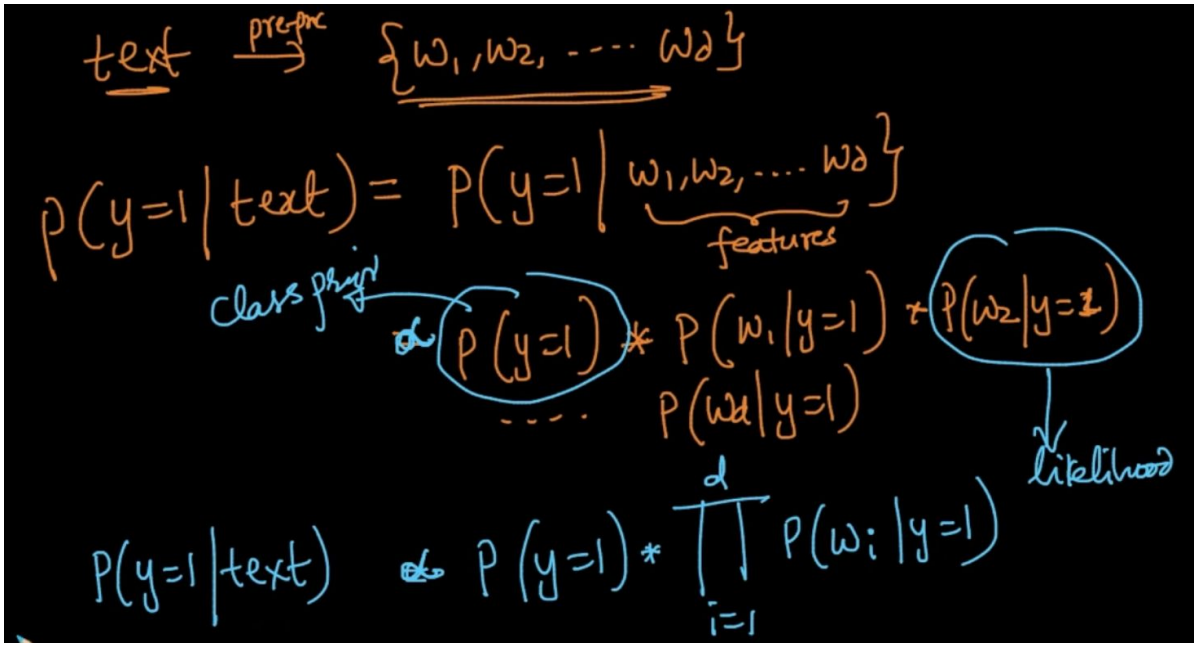
Naive Bayes is popular for text classification like Spam Filter



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($y_q = 1 \mid text_q$) and P($y_q = 0 \mid text_q$)

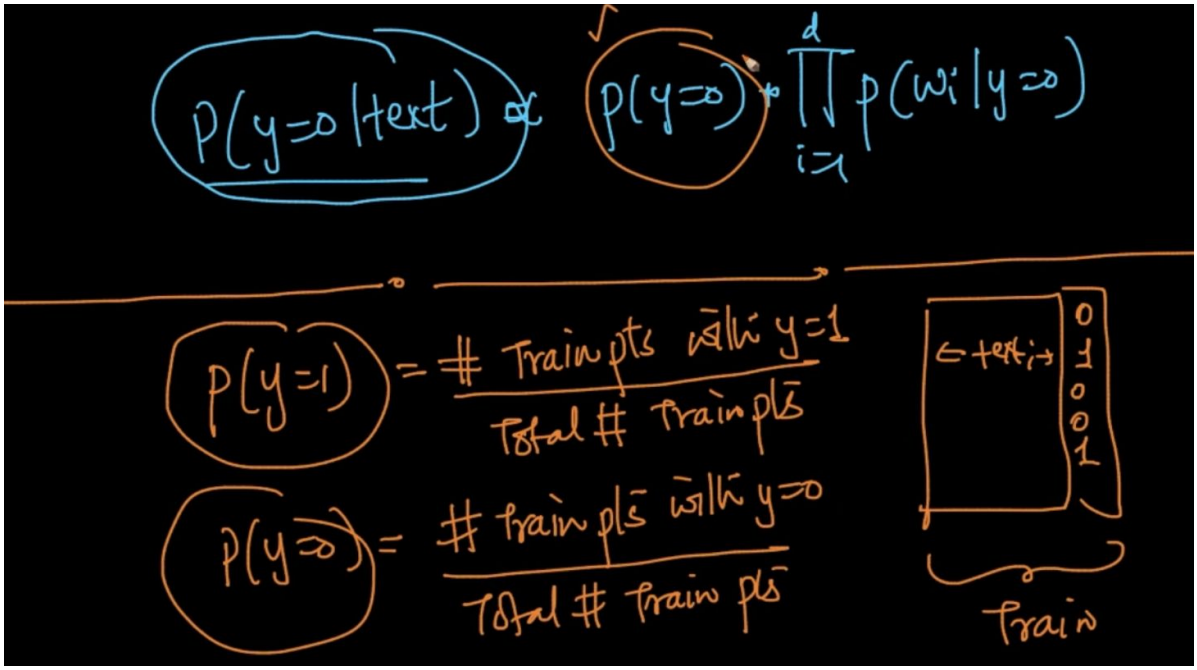
If P($y_q = 1 \mid text_q$) > P($y_q = 0 \mid text_q$) then y_q = 1

We do the pre-processing steps above on a text and get a set of words {w1,w2….wd} in a Binary BOW representation i.e if the word is present or not

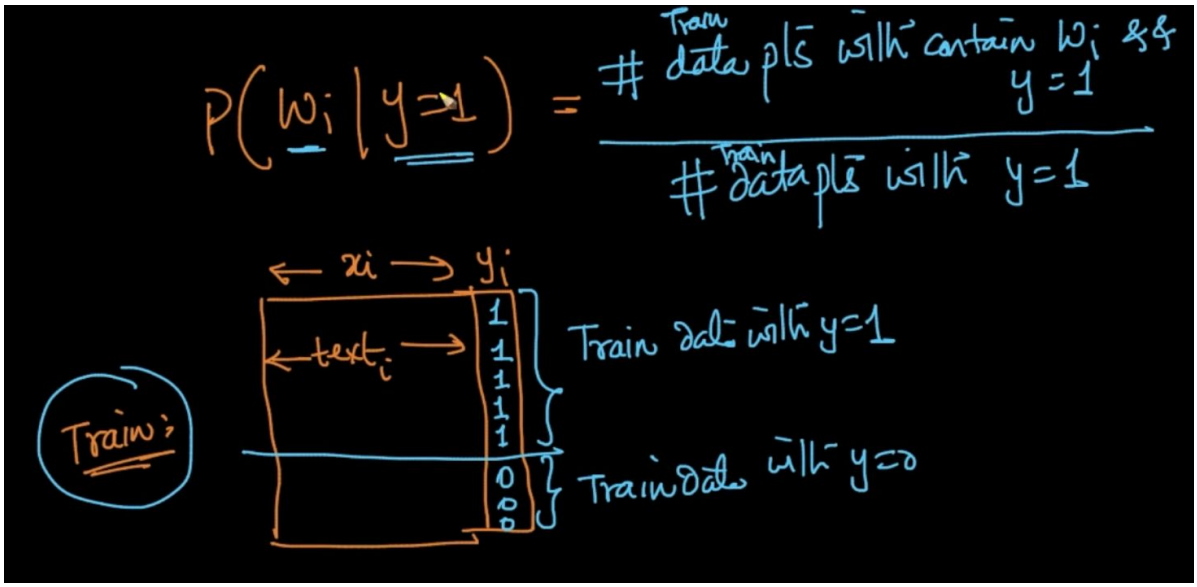


We calculate P(y = 1 | text). Here, P(y = 1) is easy to calculate. Now we want likelihoods i.e

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



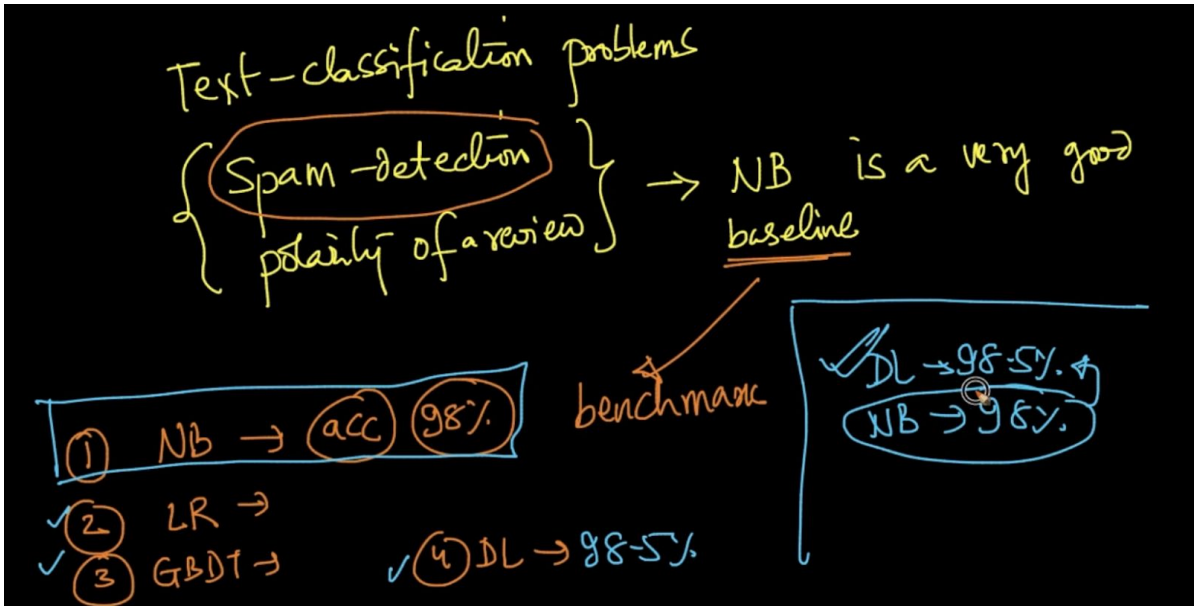
P(y = 1) and P(y = 0) are calculated very easily. Simple counting stuff. We take all the points with y = 1/ Total points



For likelihood calculation, we divide the Train Data with y = 1 and y =0. For P (w | y = 1)

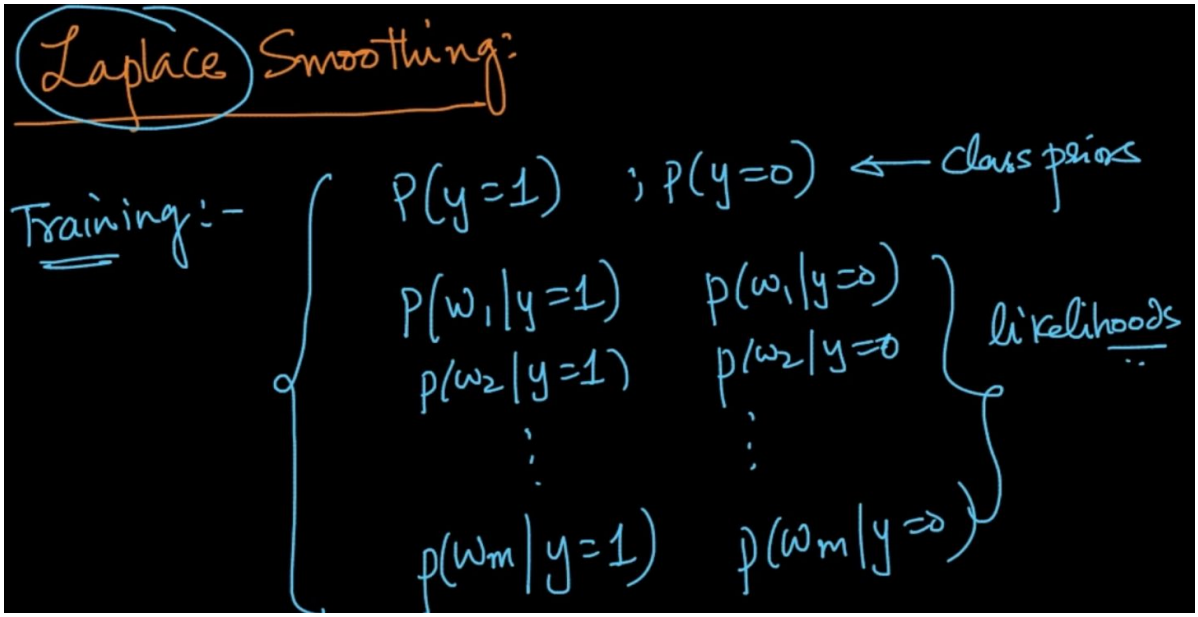
We count all the data points which contain word w and y = 1/ Total Data points with y = 1

Same for y = 0

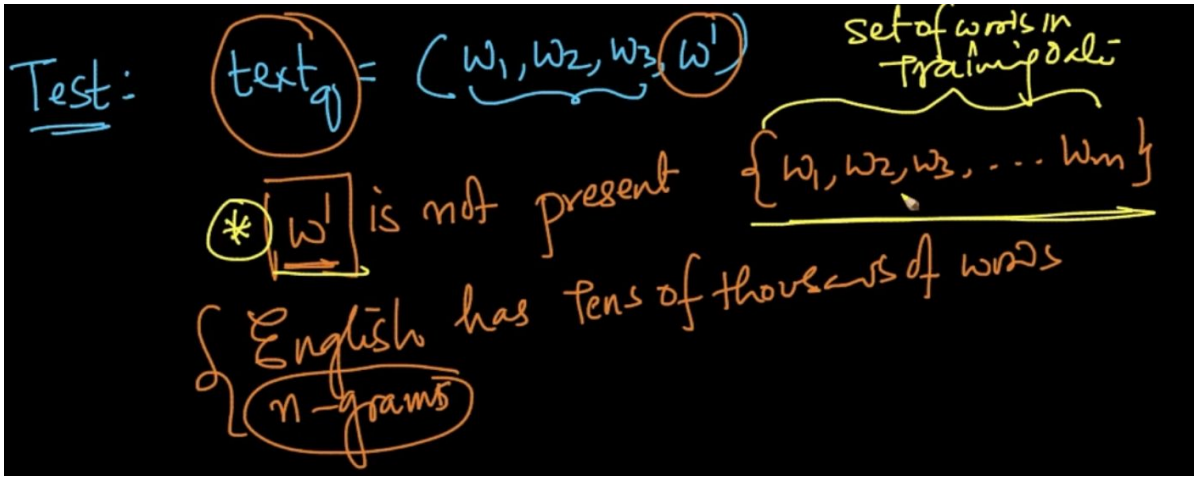


Naive Bayes is a very good baseline and it's used as a benchmark. Suppose if we get 98% on our simple baseline Naive Bayes and 98.5% on a complicated DL model then we say it's only a slight improvement

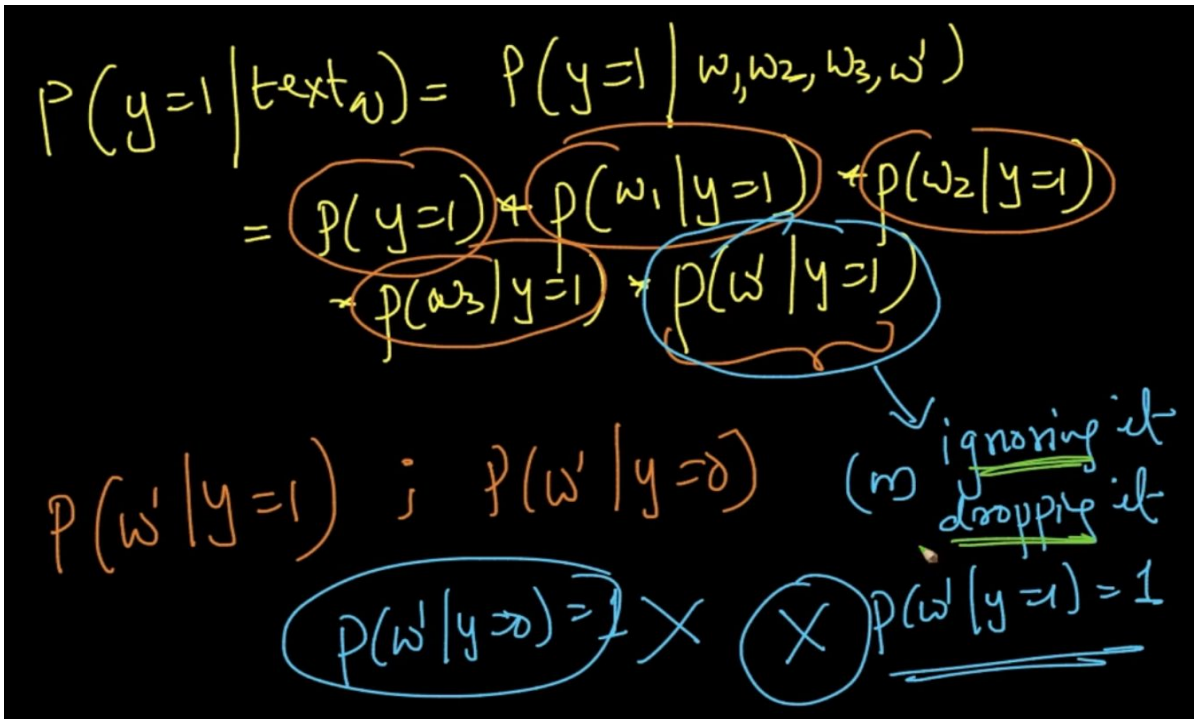
LAPLACE ADDITIVE SMOOTHING



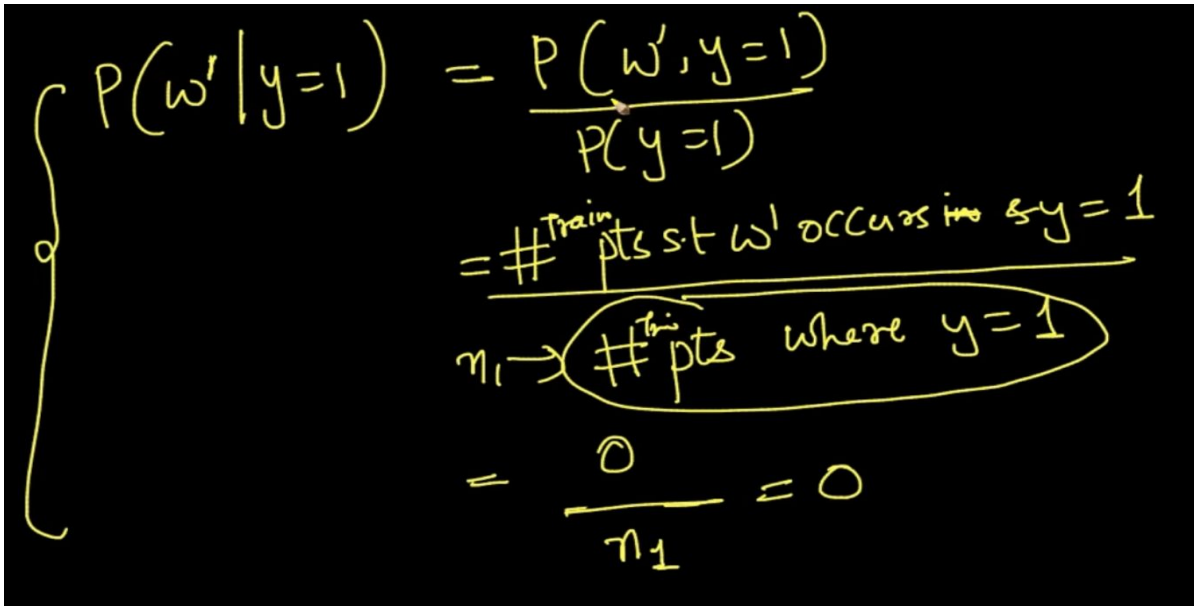
In the training parts, we want to calculate likelihoods and class priors



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?

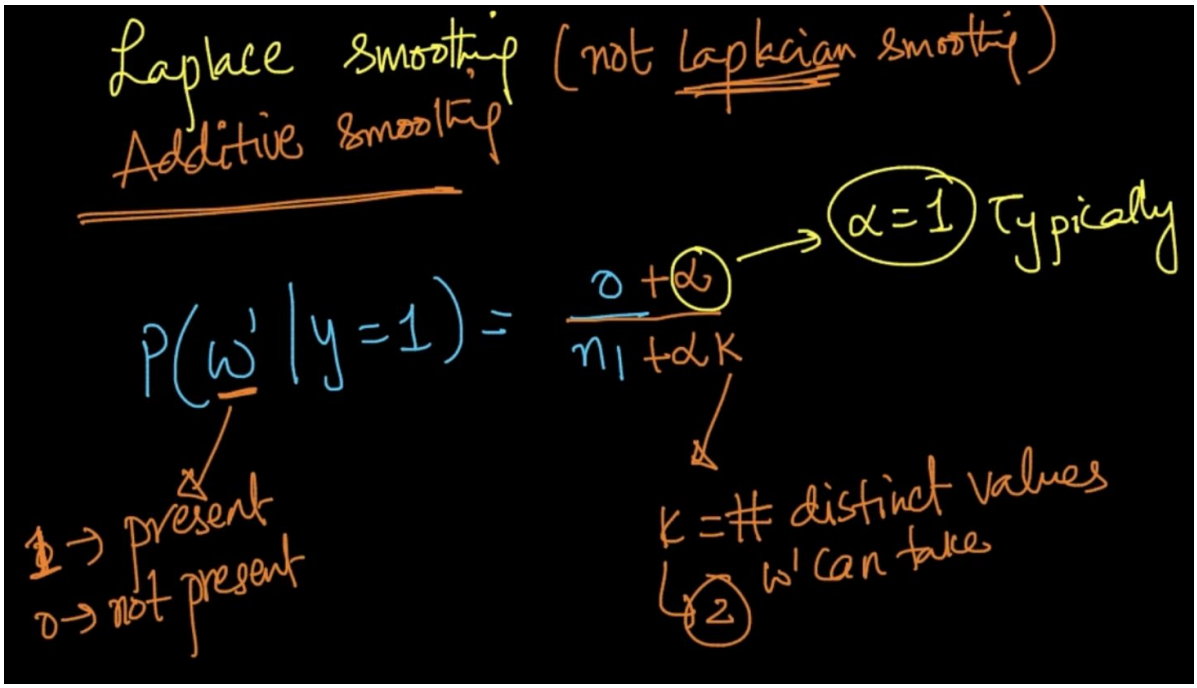


We want to calculate P (w' | y = 1). What if we ignore it ? It'll mean P (w' | y = 1) = 1 and Also P (w' | y = 0) = 1 which cant be the case

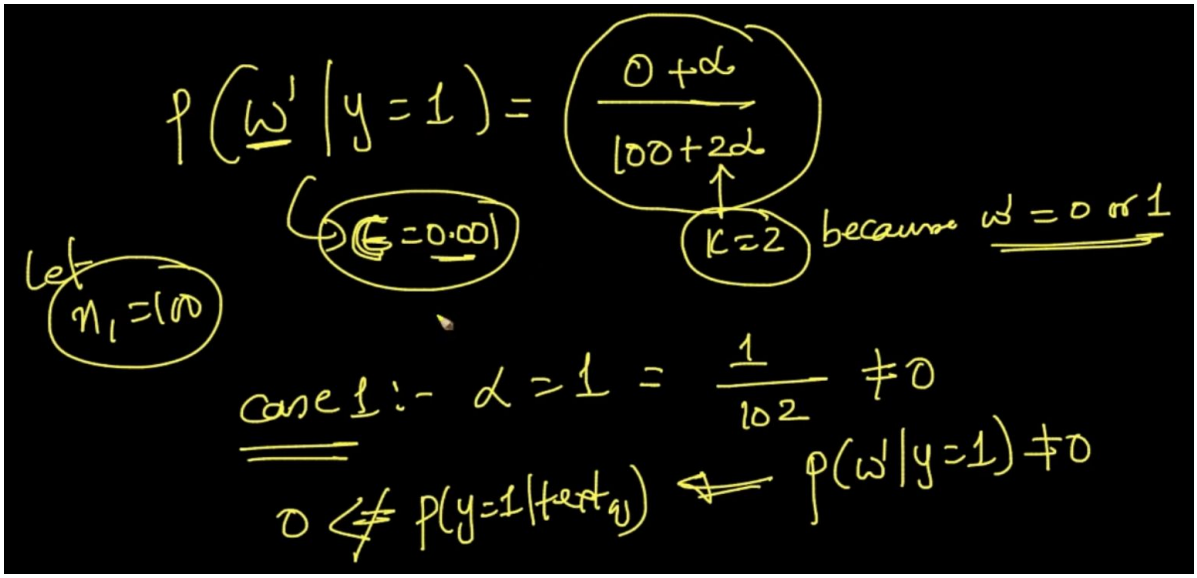


We can't go for our conditional probability method since w' is not present which means

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

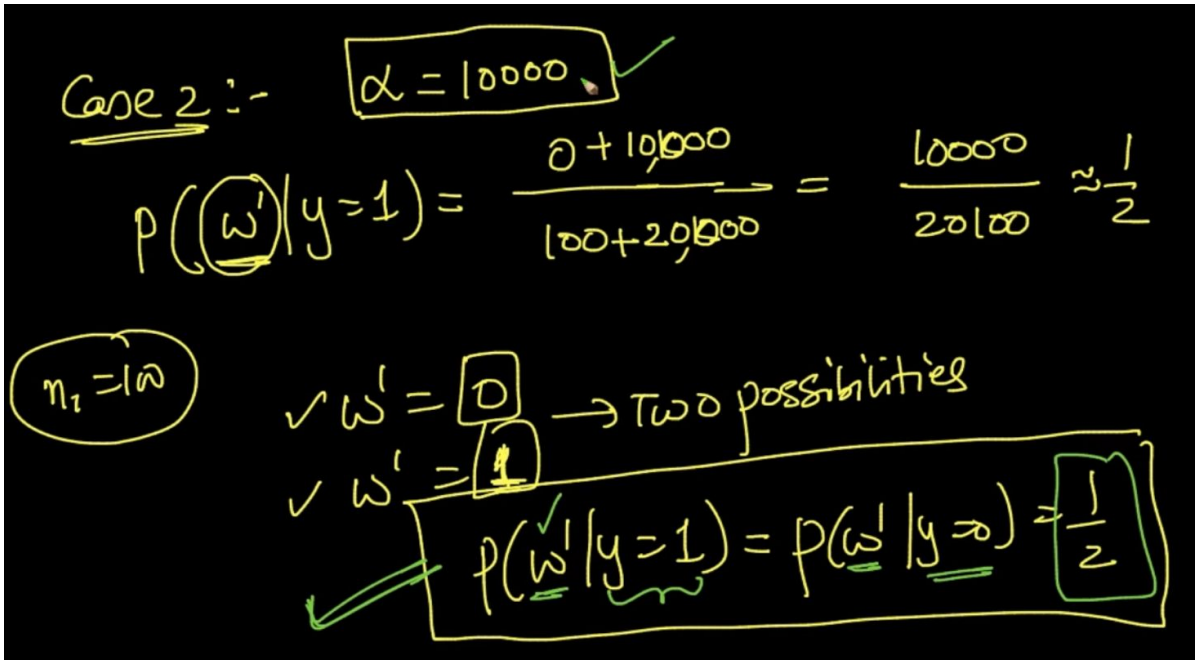


We add α and K for smoothing. K = number of distinct values w'/ feature can take . Here, since it's a Binary BOW it can only take 2 distinct values (0 or 1) therefore K = 2

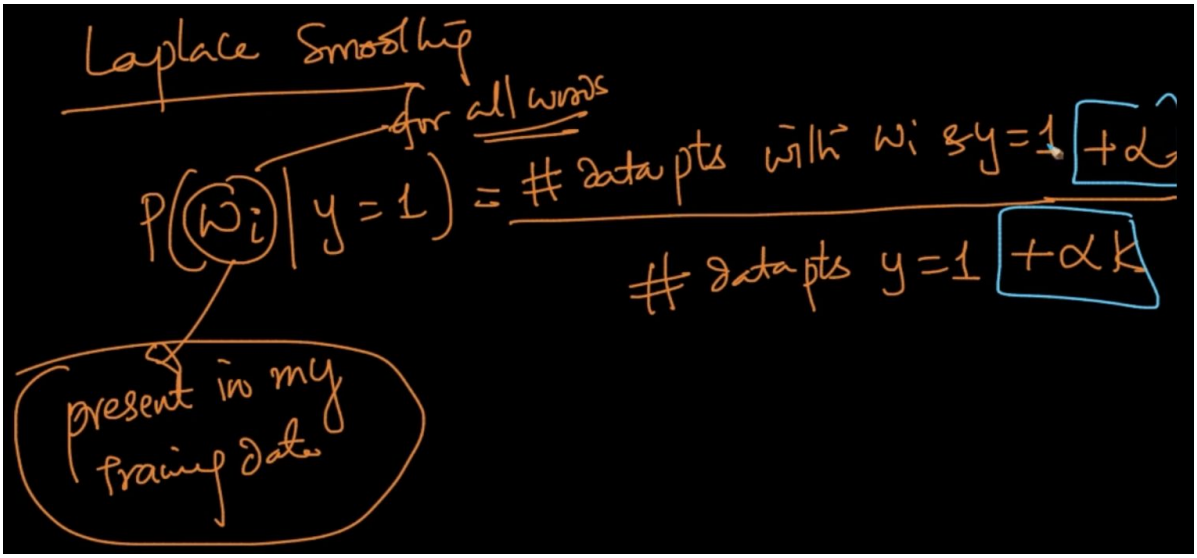


Case 1 : α = 1 . We got rid of the problem of P (w' | y = 1) = 0. It won't be 0 anymore but why using this method. We could've just replaced new word w' probability with a small e.g 0.01

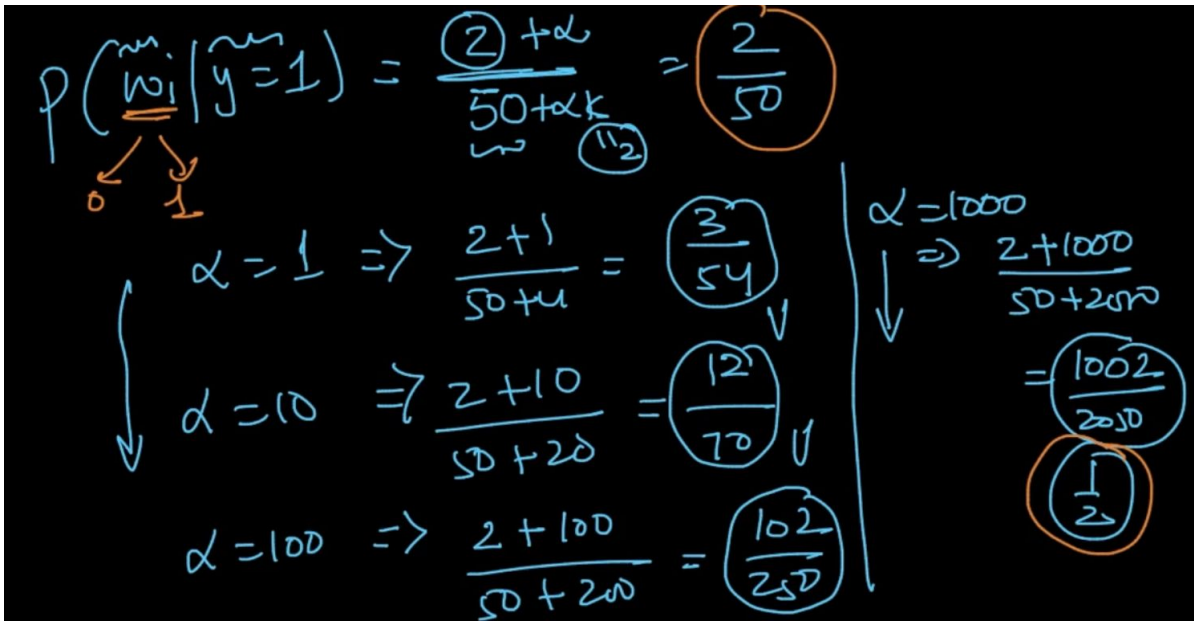
But there are some mathematical insights we can get from this smoothing method



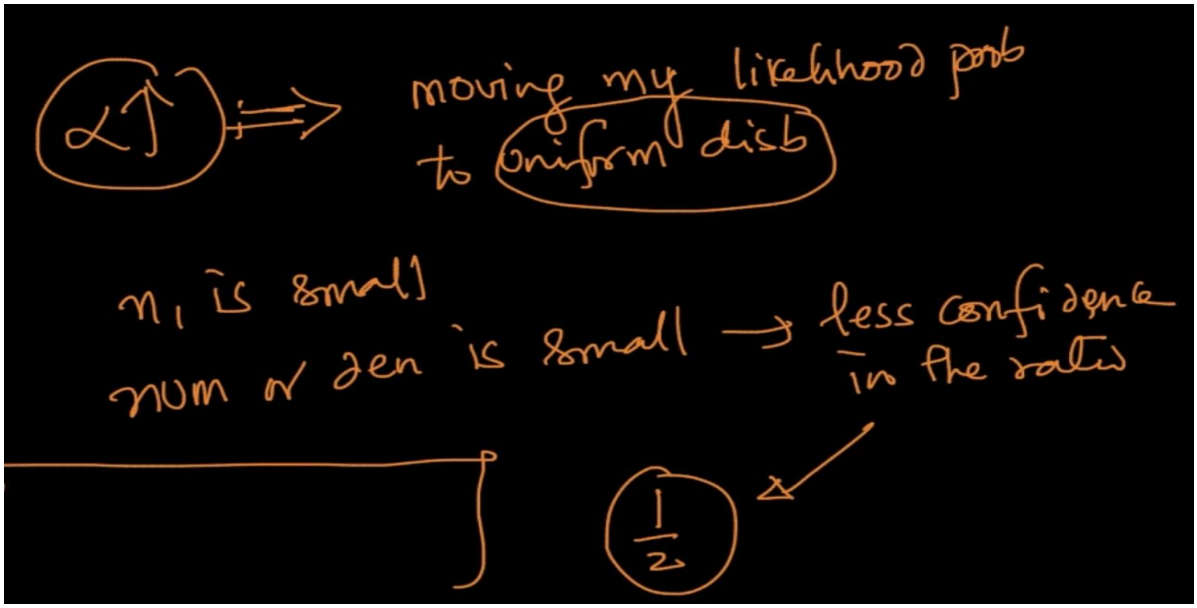
Case 2 : α = 10,000. P $\approx$ 1/2 is a good thing and more logical because we don't know anything about w' and saying that probability of w' occuring in both classes = ½ i.e equiprobable is logical. So what happens in Laplace Smoothing ?



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



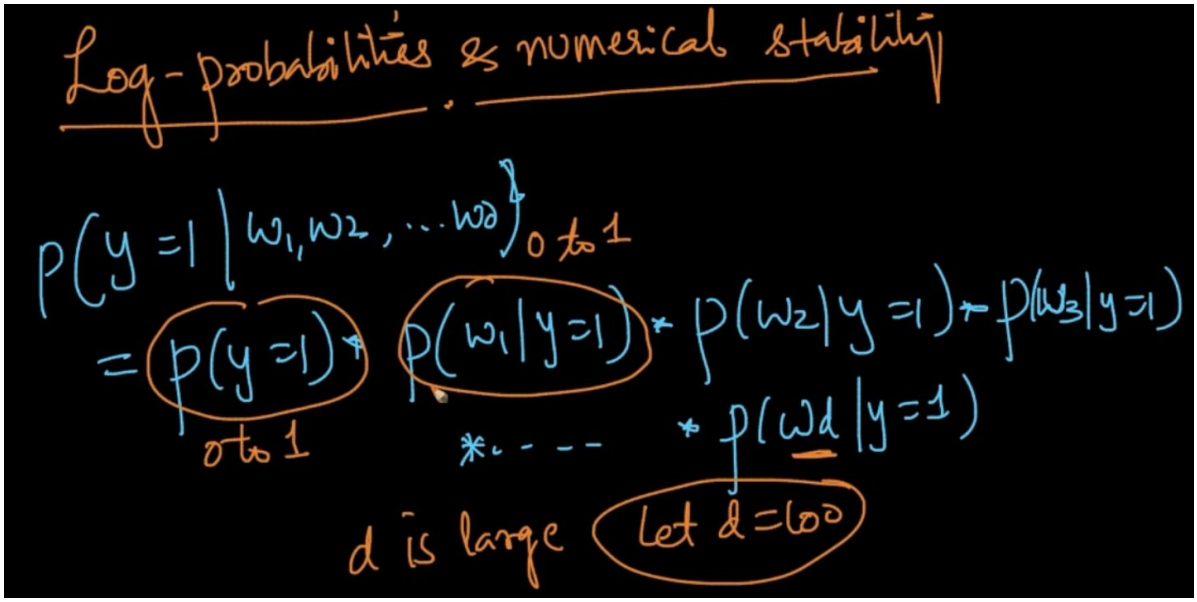
So what is happening when α is increasing ?



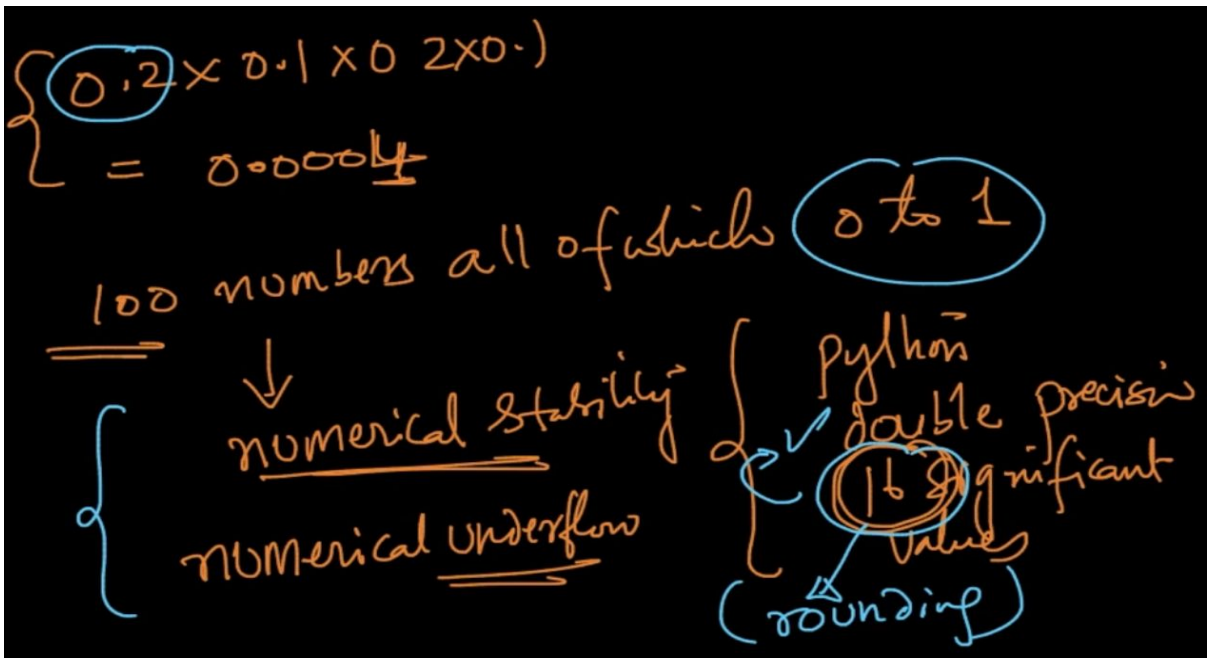
$\alpha \uparrow$, we move our likelihood probabilities to uniform distribution. If our numerator or denominator is small we've less confidence in the ratio. So bu increasing α we are moving

towards a stable confidence. In above example, at α = 1000 it is ½ which makes much sense since w' can only take 2 values (0 or 1) so it has smoothed/ moved to a uniform distribution.

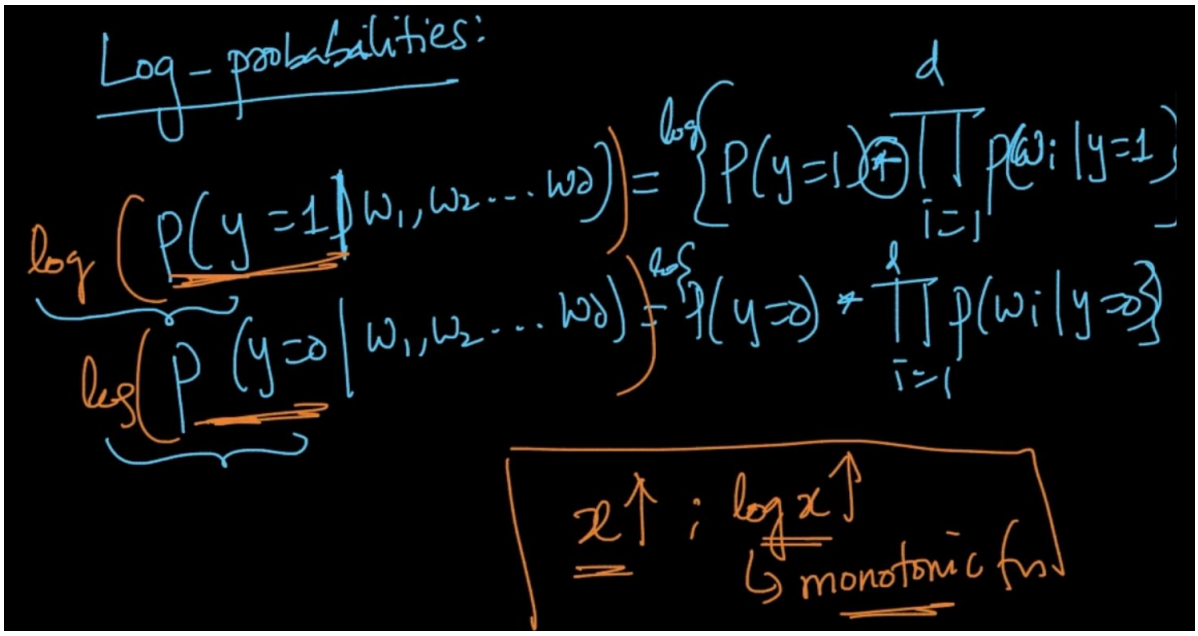
LOG - PROBABILITIES FOR NUMERICAL STABILITY



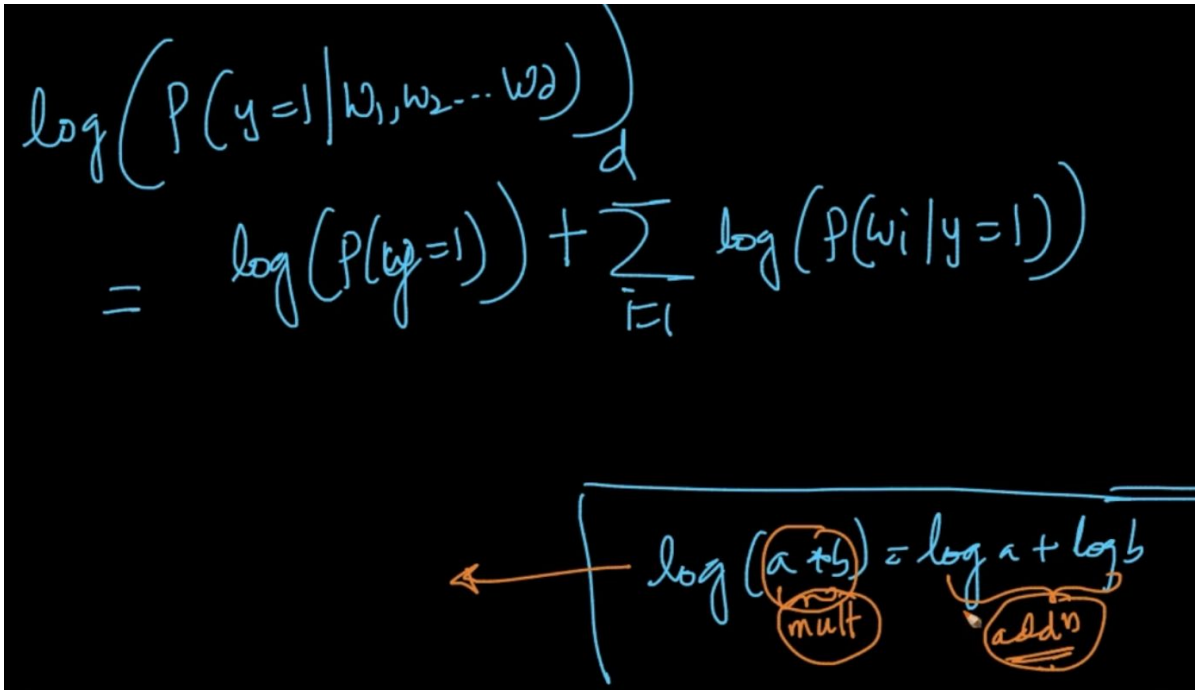
Calculation for prediction of class label with d = 100 features



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

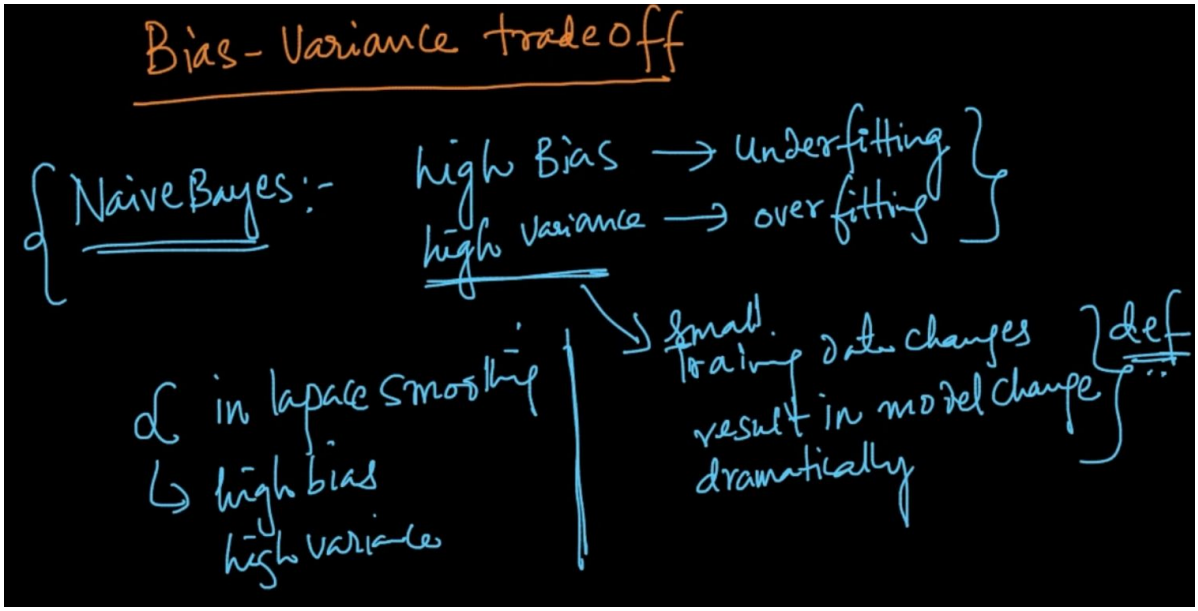


We need to compare P(y = 1| w) and P(y = 0 | w). If we do their log then the calculations will get easier as log is a monotonic function

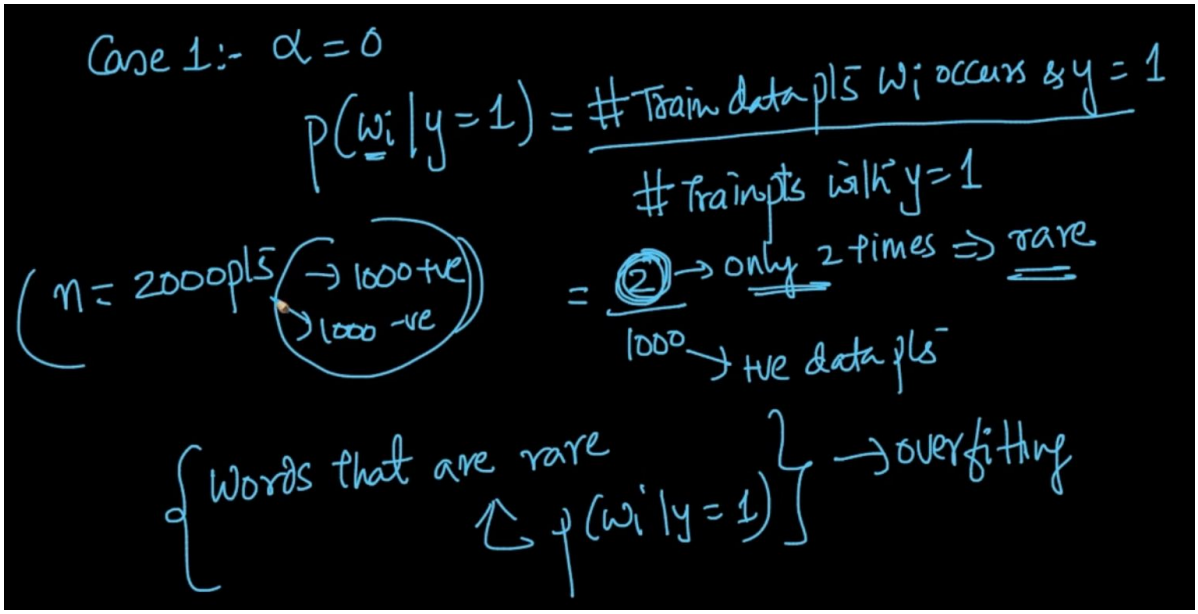


Here it can be seen that calculations are getting easier because multiplication get verted to addition. It also converts exponentiation to multiplication as log(a^b)= b*log a

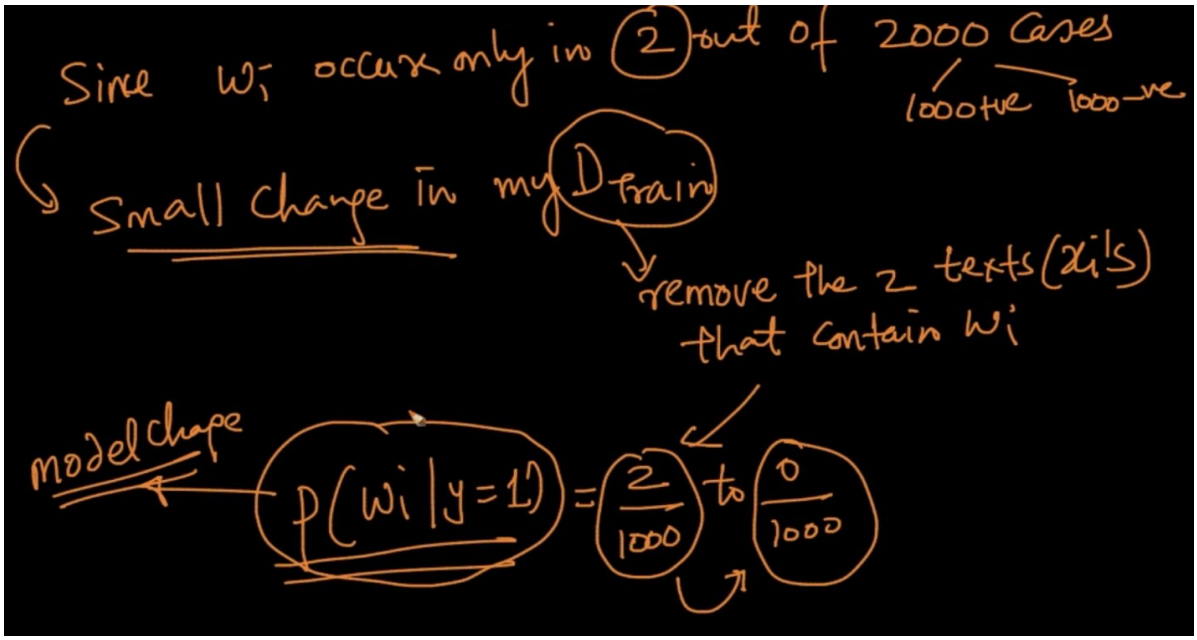
BIAS-VARIANCE TRADEOFF



In Naive Bayes, α in Laplace smoothing determines high bias or high variance

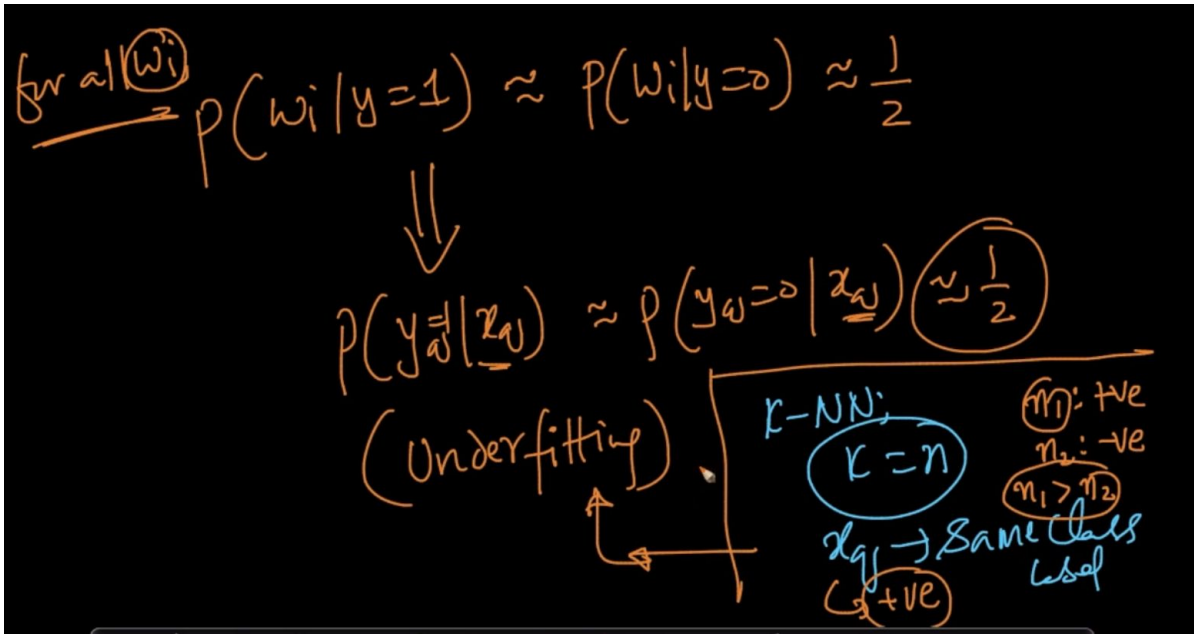


In α = 0 , if there's a word w_i is rare then calculating it's probability is overfitting in a way

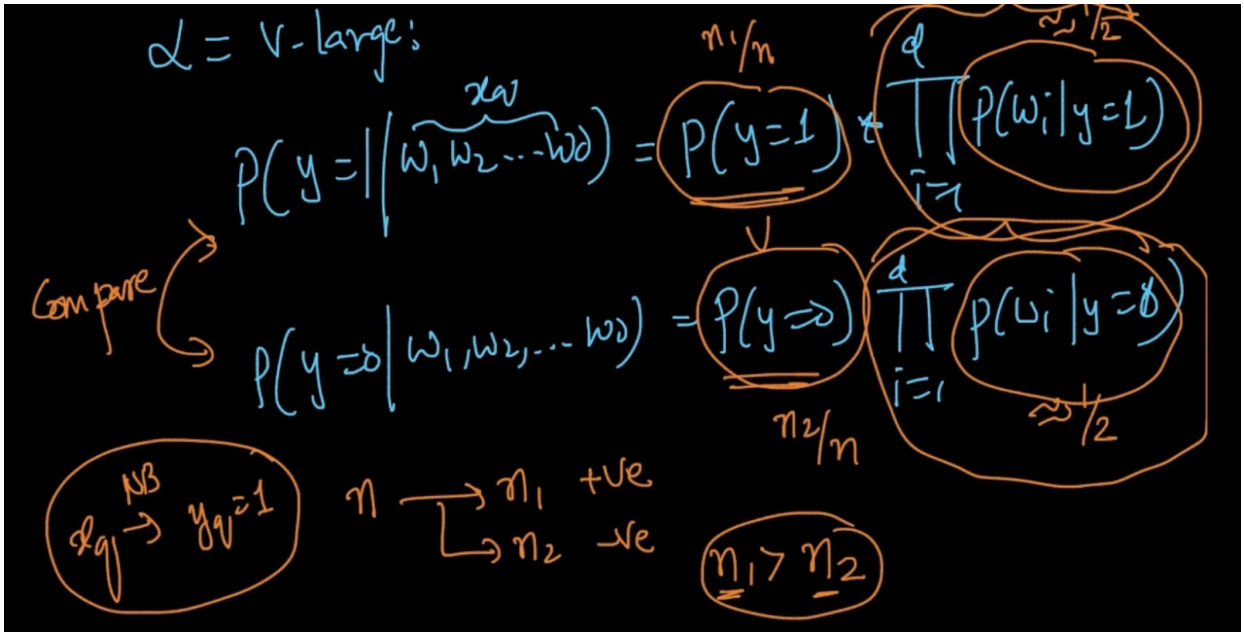


Since there's no laplace smoothing i.e α = 0 we remove the texts that contain word w_i

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

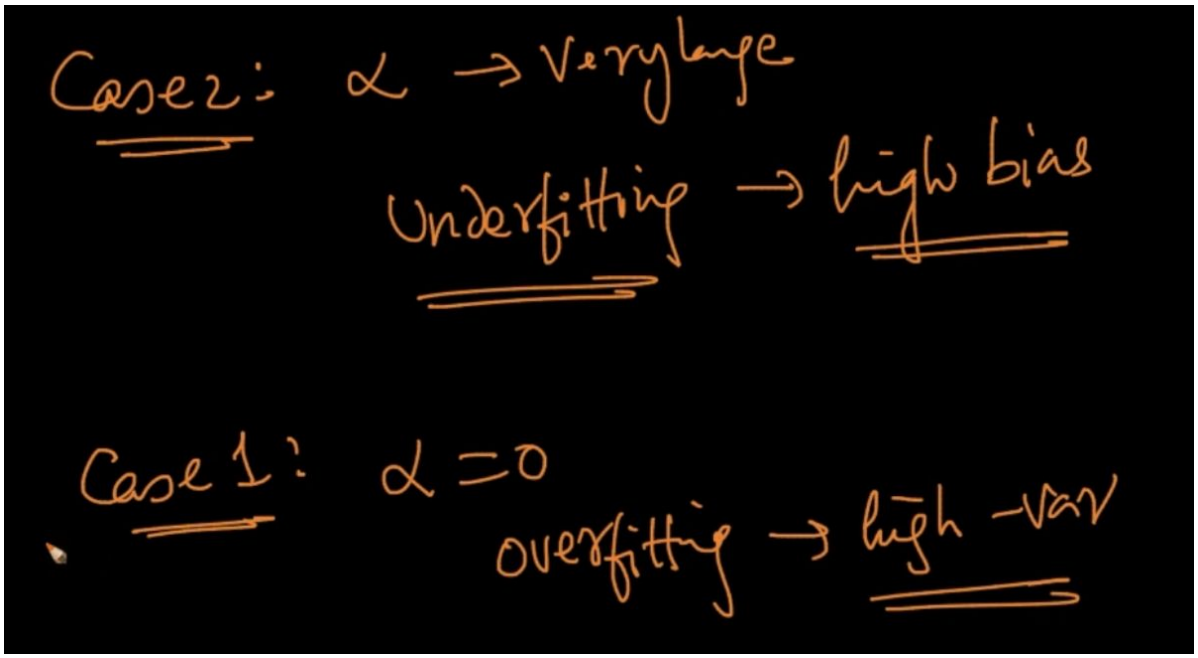


Case 2 : α = Very Large then P (w | y = 1) $\approx$ P(w | y = 0) $\approx$ ½. Our model can't distinguish so it'll be underfitting



Since P (w | y = 1) $\approx$ P(w | y = 0) $\approx$ ½ it's only dependent on P(y = 1) and P(y =0). So, if

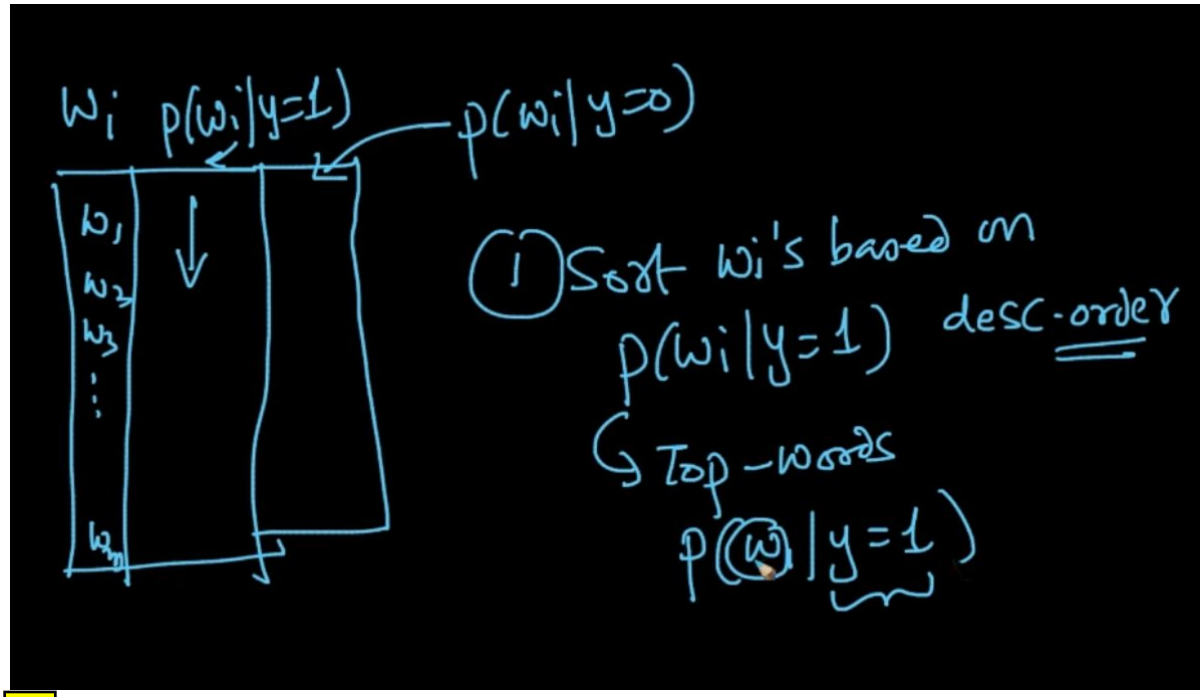
Positive class points > Negative class points then for every query x_q it'll predict y_q = 1 which is the same as underfitting in k-NN



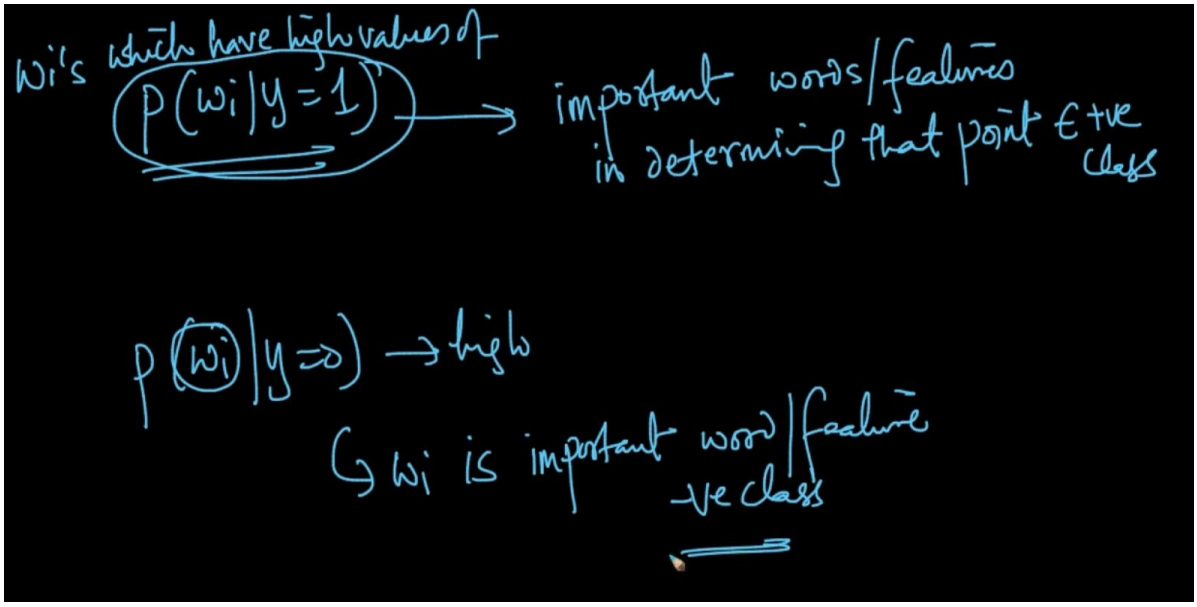
So we need to find the right α. So, to find it we do Cross Validation as α is a hyper-parameter and hyper-parameters are decided using Cross Validation(CV)

FEATURE IMPORTANCE AND INTERPRETABILITY

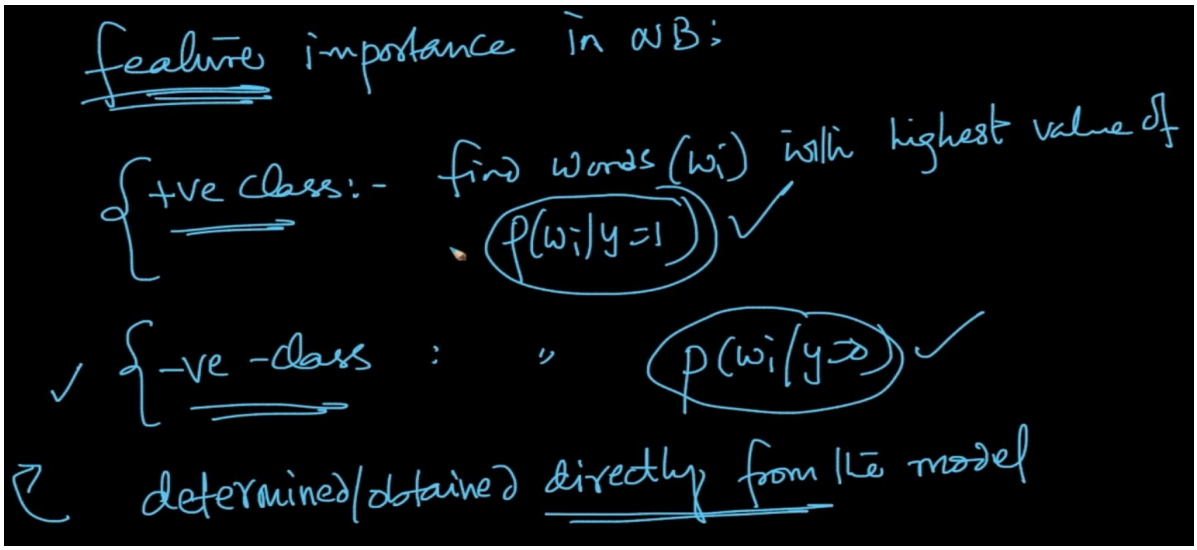
FEATURE IMPORTANCE



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

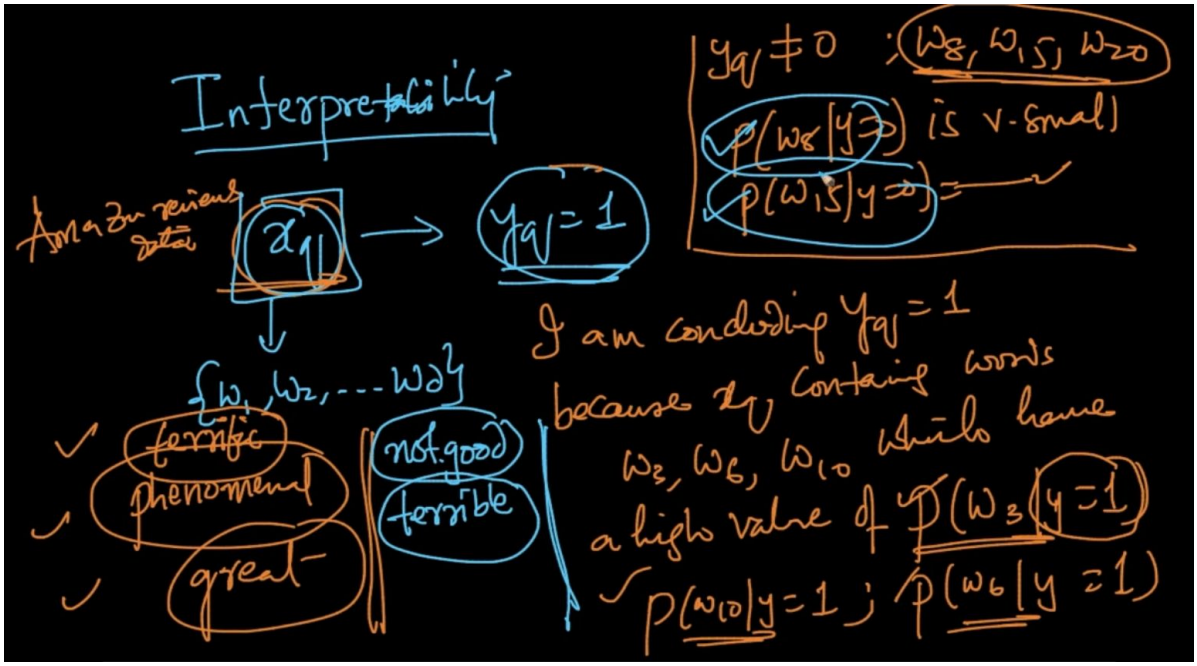


That's how most important features are decided



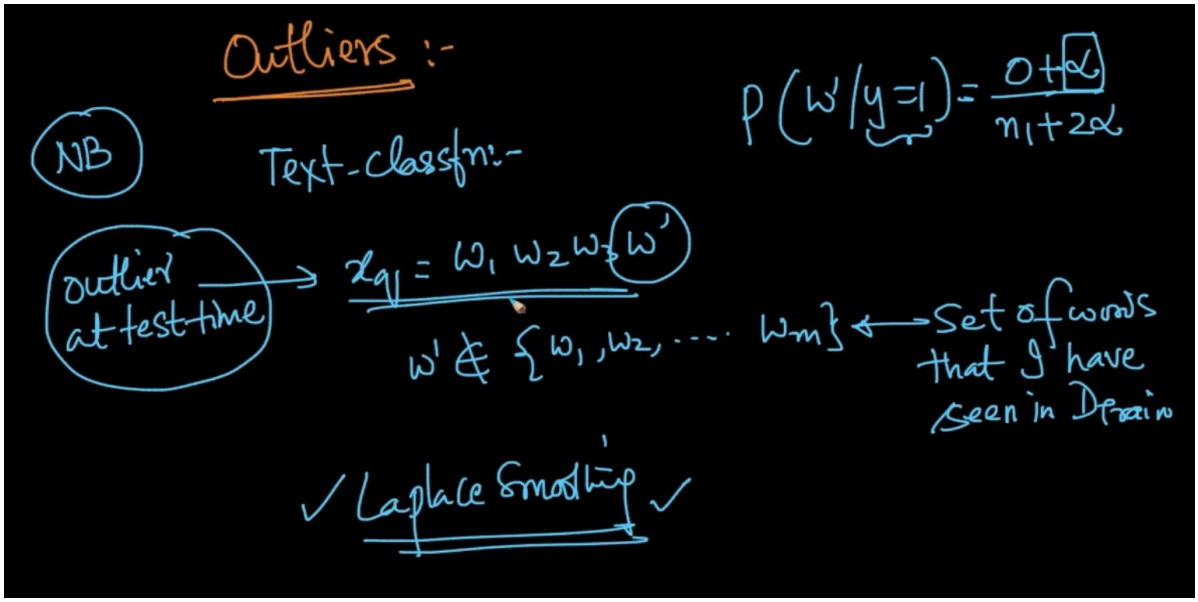
In k-NN, we used Forward Feature selection to select the most important features. Here, we can directly get most important features from model itself since Probabilities are calculated at the learning stage itself

INTERPRETABILITY

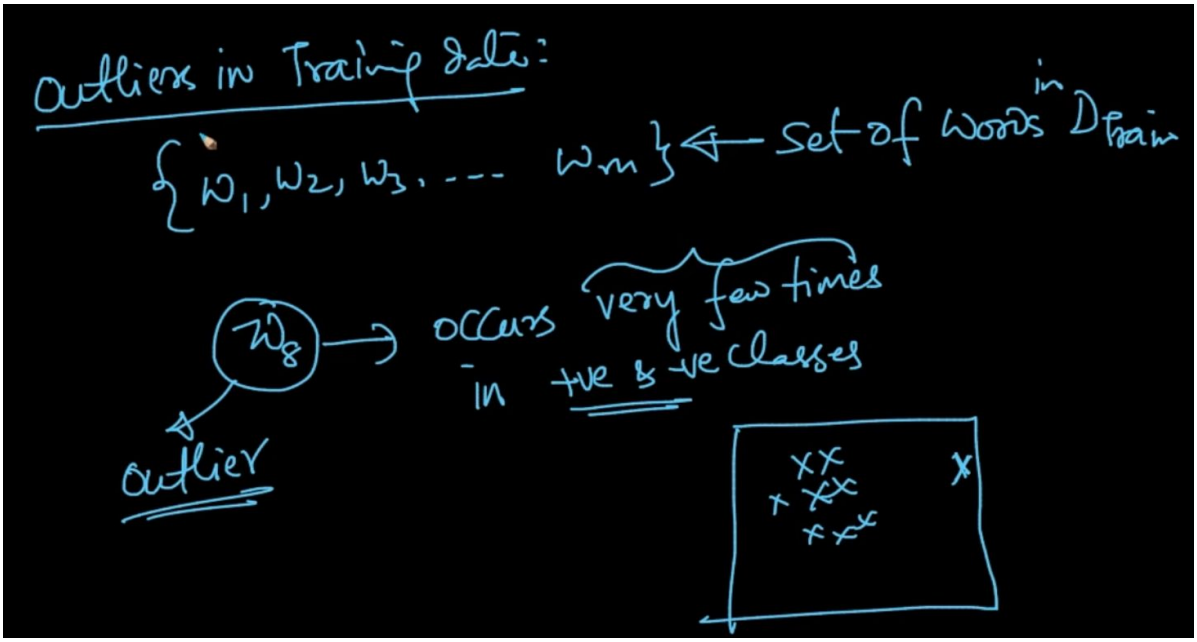


Suppose if we've a positive review. We can conclude it's y = 1 because x_q contains words w3, w5, w8 like *terrific, great* which has high probabity score and also conclude y $\neq 0$ since the probabilty scores of negative words w7, w9 like terrible, not-good are small

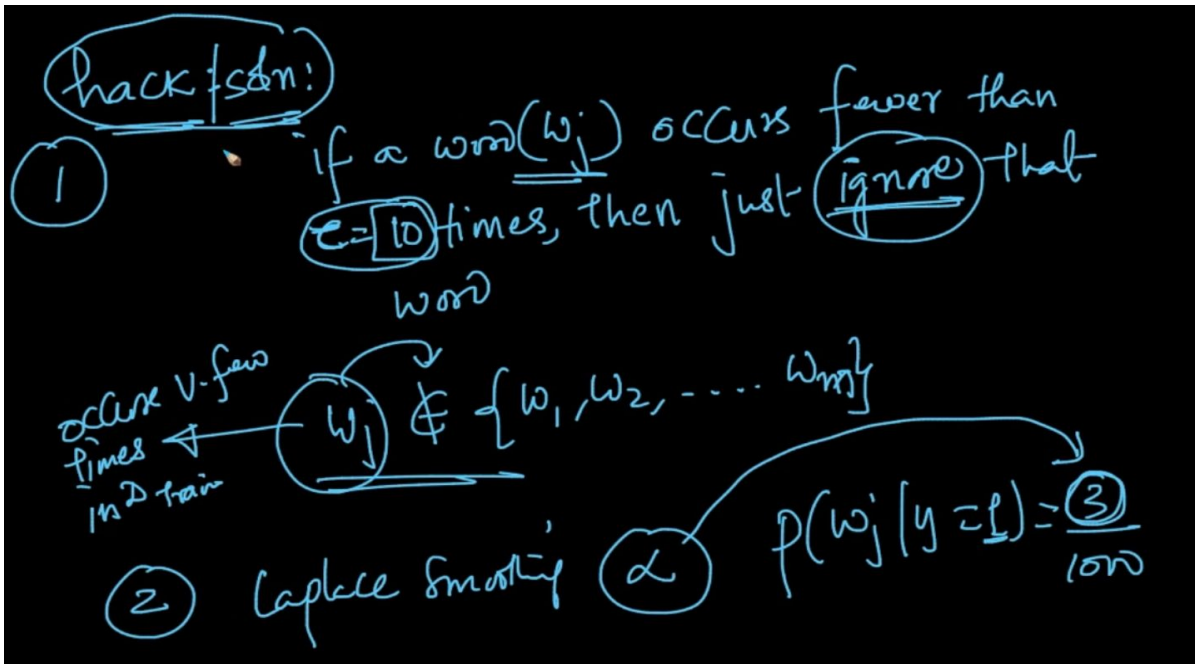
OUTLIERS



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



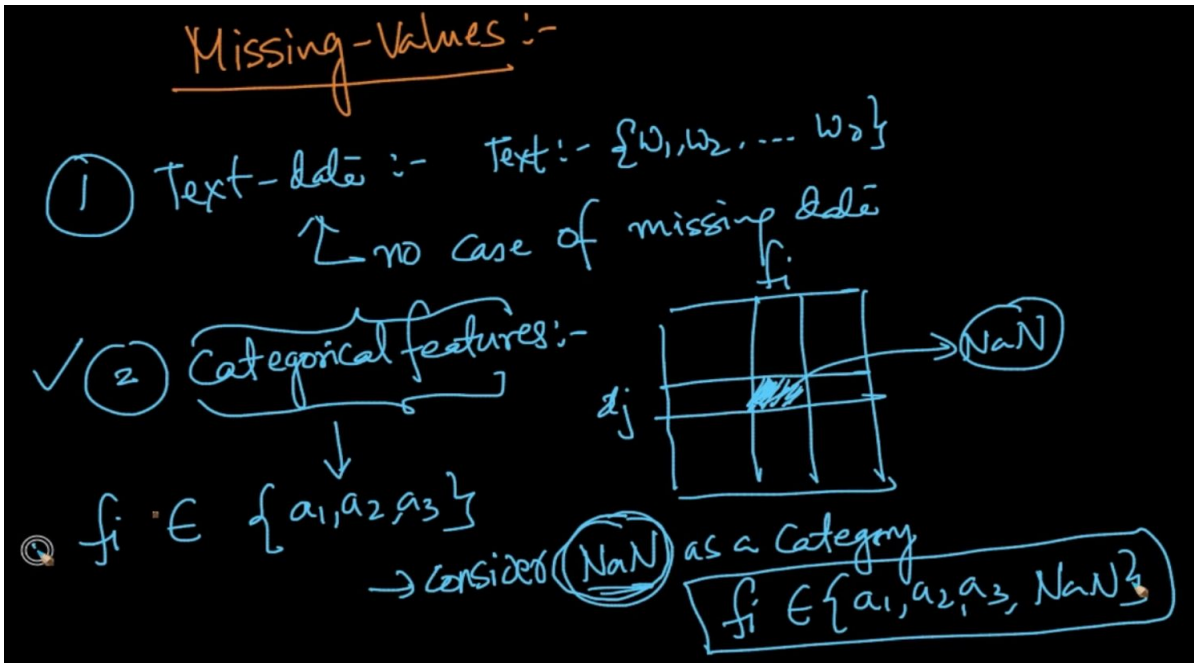
If a word w occurs very few times in our Train data then it's an outlier. How do we deal with it?



1) We can apply a condition/ threshold to just ignore that word as seen above

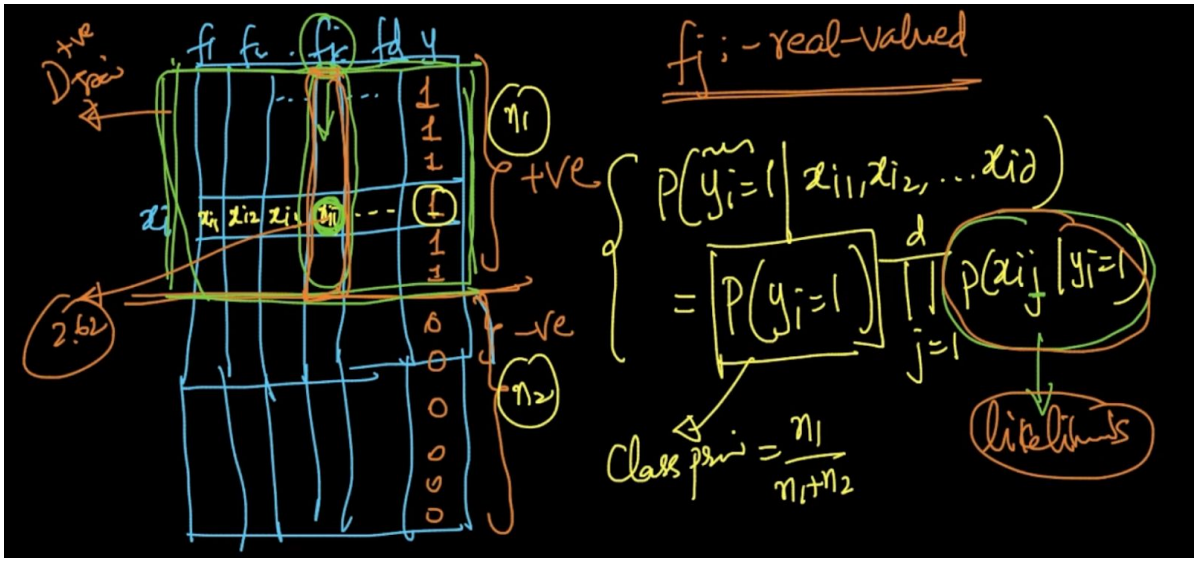
2) We can use Laplace Smoothing in our Train Data as well with the right α

MISSING VALUES

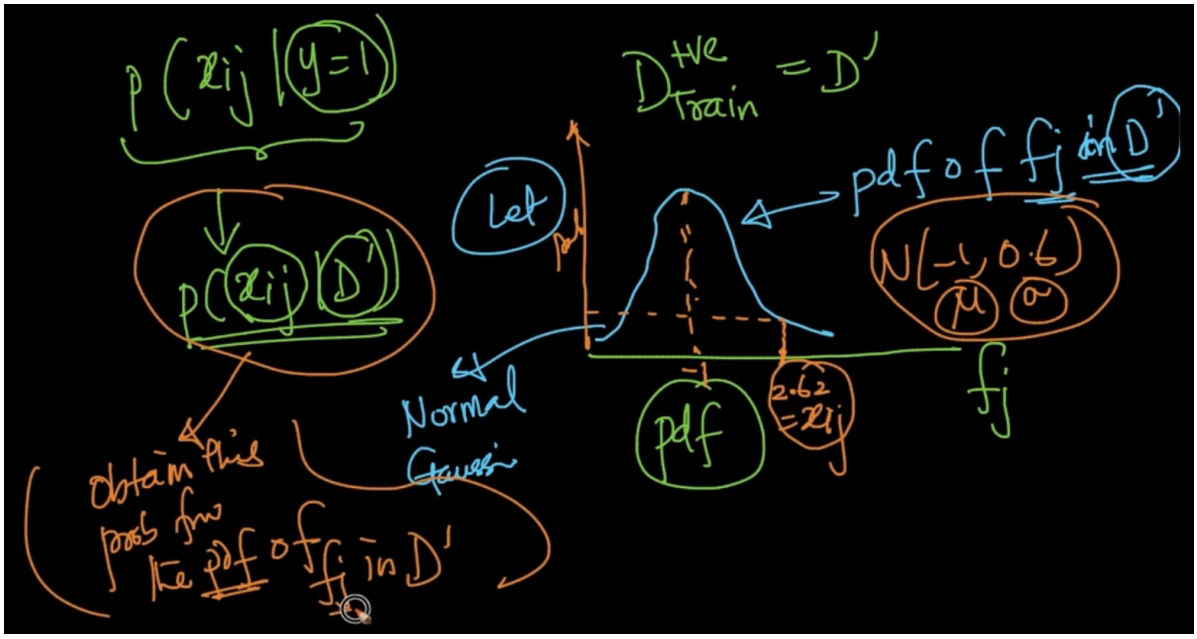


For Categorical features if it doesn't belong to features {a1,a2,a3} e.g NaN then we also consider NaN as a category. If missing values are there in Numerical data then Imputation

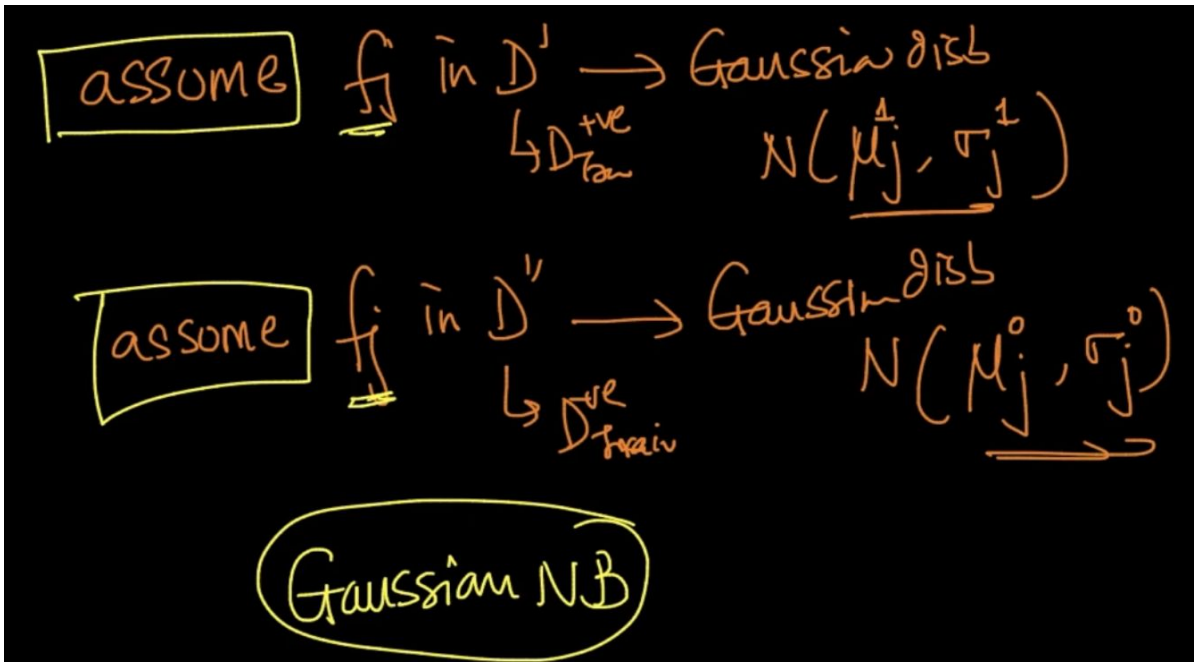
GAUSSIAN NAIVE BAYES



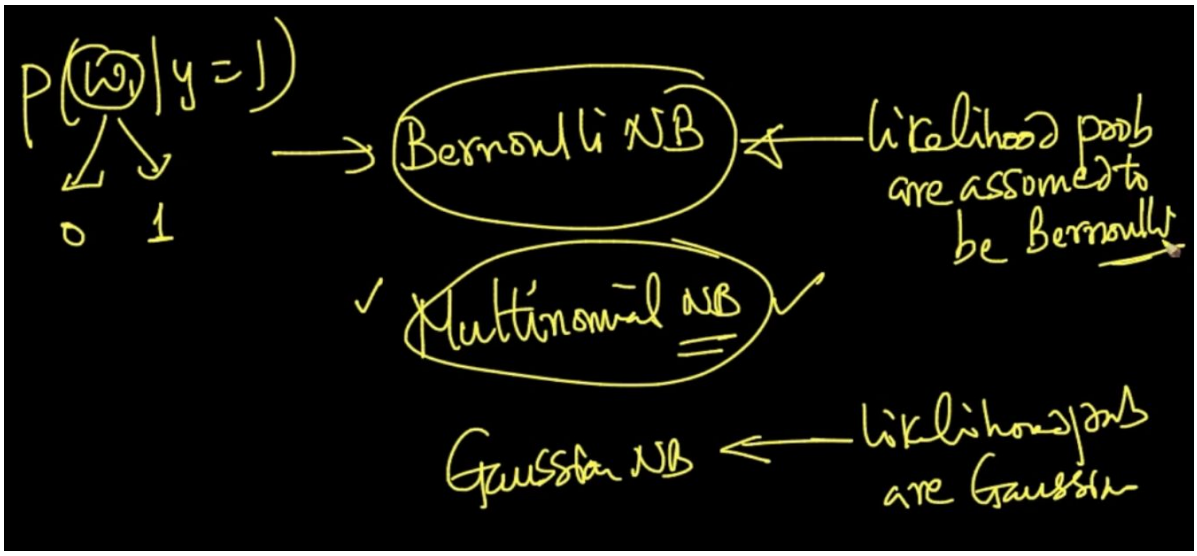
We are seeing Naive Bayes for numerical data. First we'll divide the data into positive and negative classes . P(y = 1) is easy to calculate. P($x_{ij} | y = 1$) is difficult to calculate .It's our likelihood. It is probability of feature j given our class label



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 probabilty value by plotting PDF of j since PDF has Probability scores in y-axis



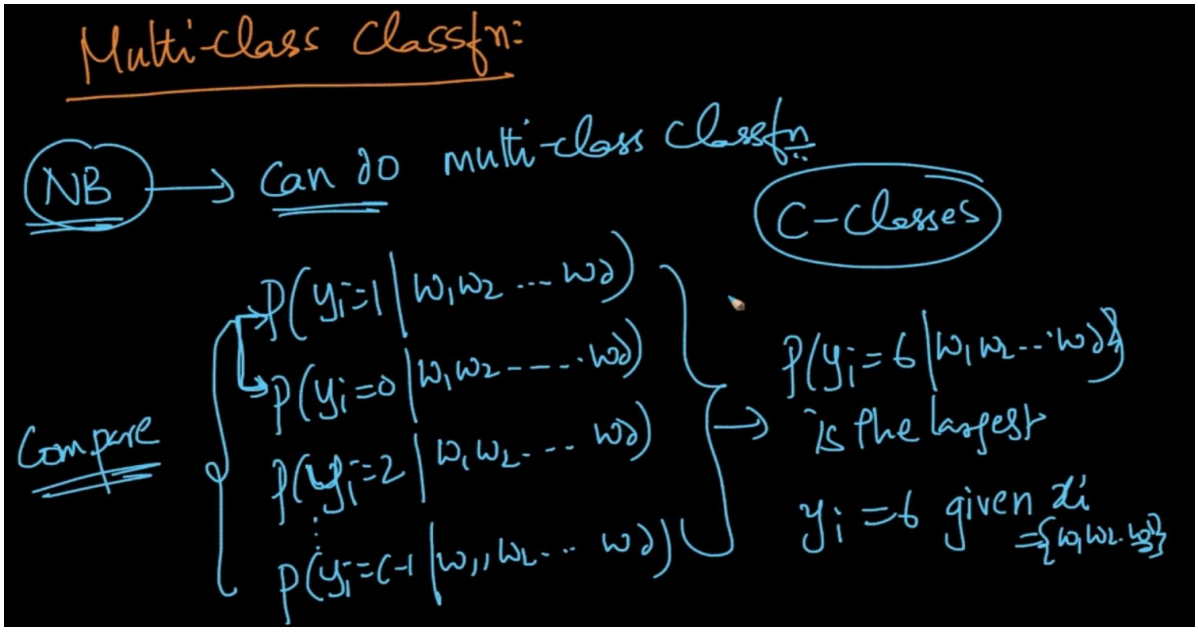
Here we are assuming that our features follows Gaussian Distribution. Therefore, this is called Gaussian Naive Bayes



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

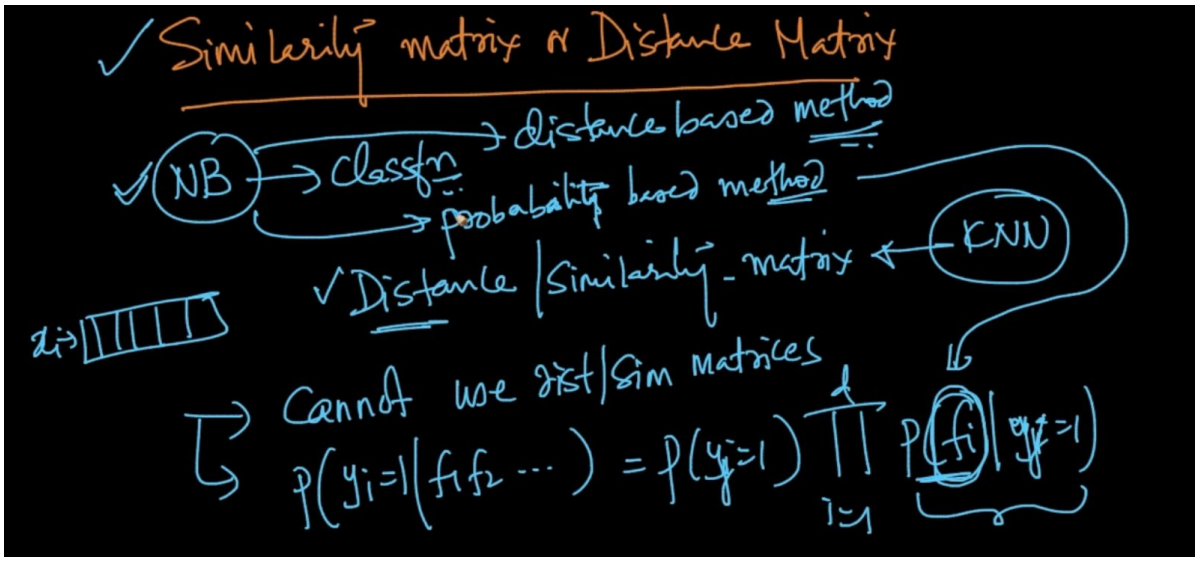
NOTE - Naive Bayes' big assumption is that all the features are conditionally independent i.e the features aren't dependent on each other

MULTICLASS CLASSIFICATION



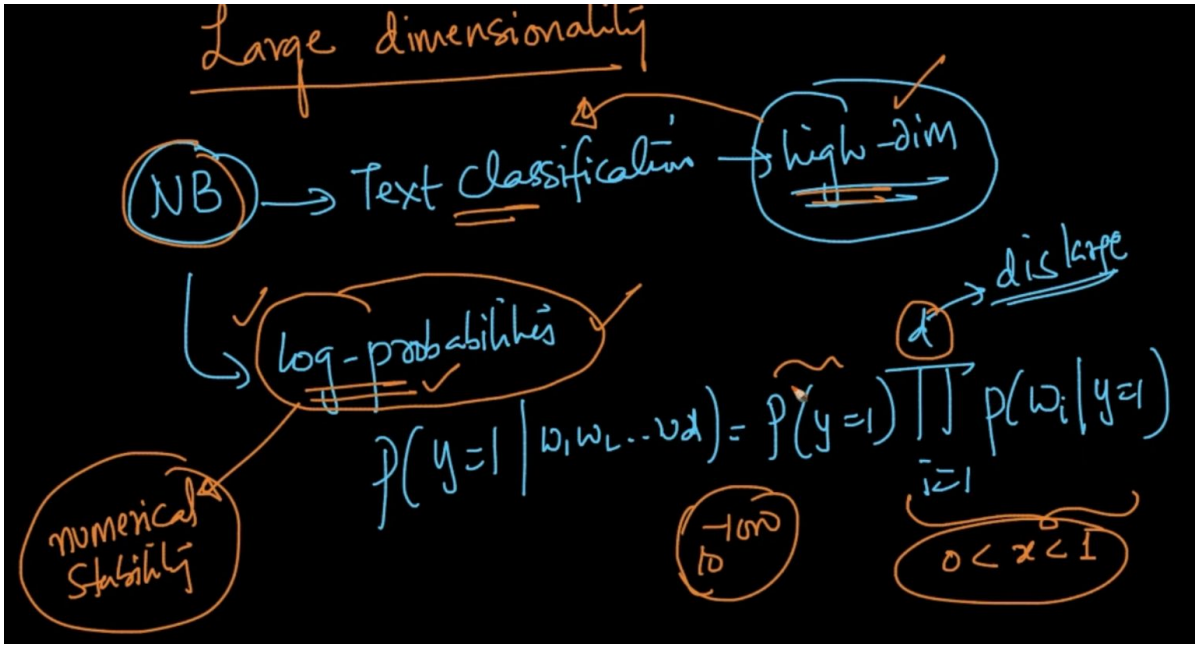
Naive Bayes can do Multiclass Classification pretty well. If we've c-classes then we'll calculate probability scores for all the classes and select the one class which has largest probability in it

SIMILARITY MATRIX OR DISTANCE MATRIX



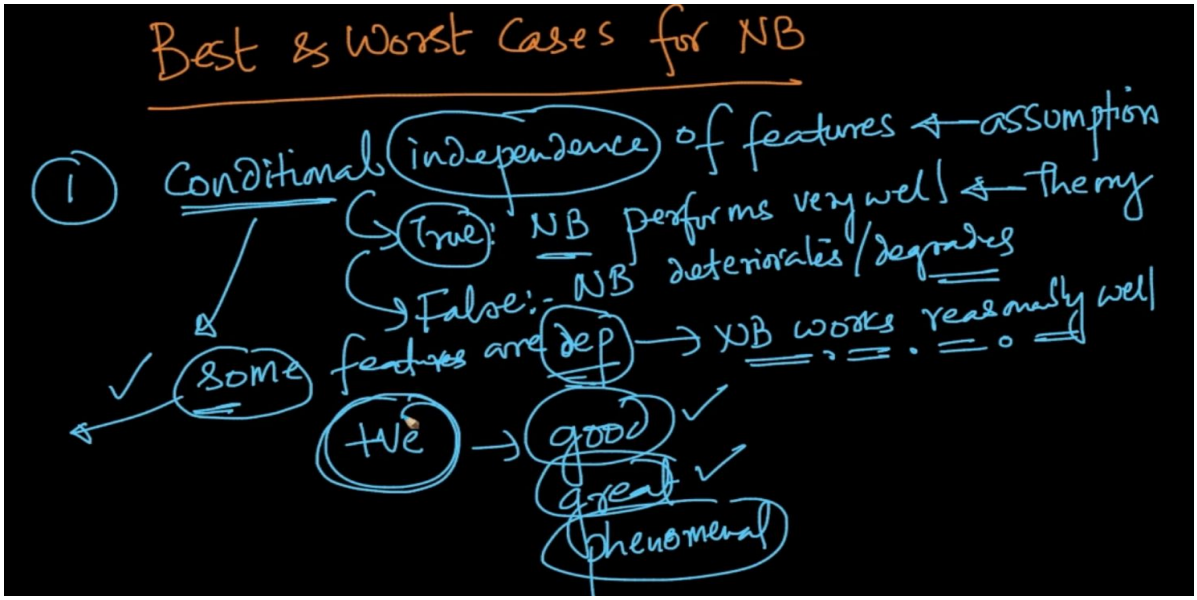
We cannot use distance/similarity matrices in NB as classification is done here on a probabilistic based method.So we need the exact value of the feature unlike k-NN which is a distance-based method which could use Distance matrix to do classification

LARGE DIMENSIONS

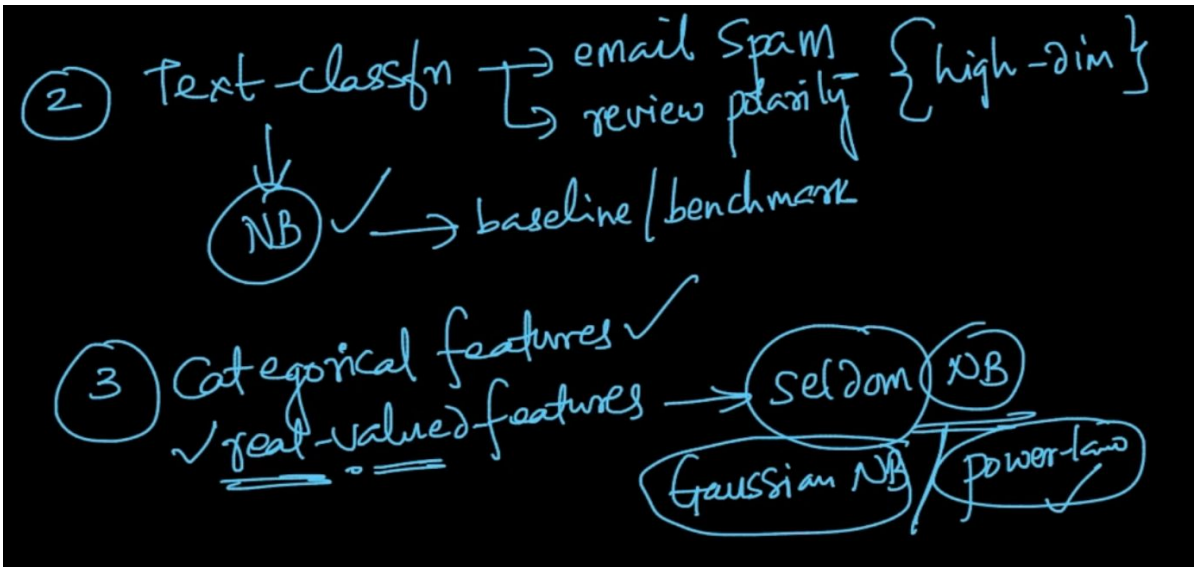


Naive Bayes can do very good in Large Dimensionality but use Log-probabilities instead of normal probabilities to avoid numerical underflow

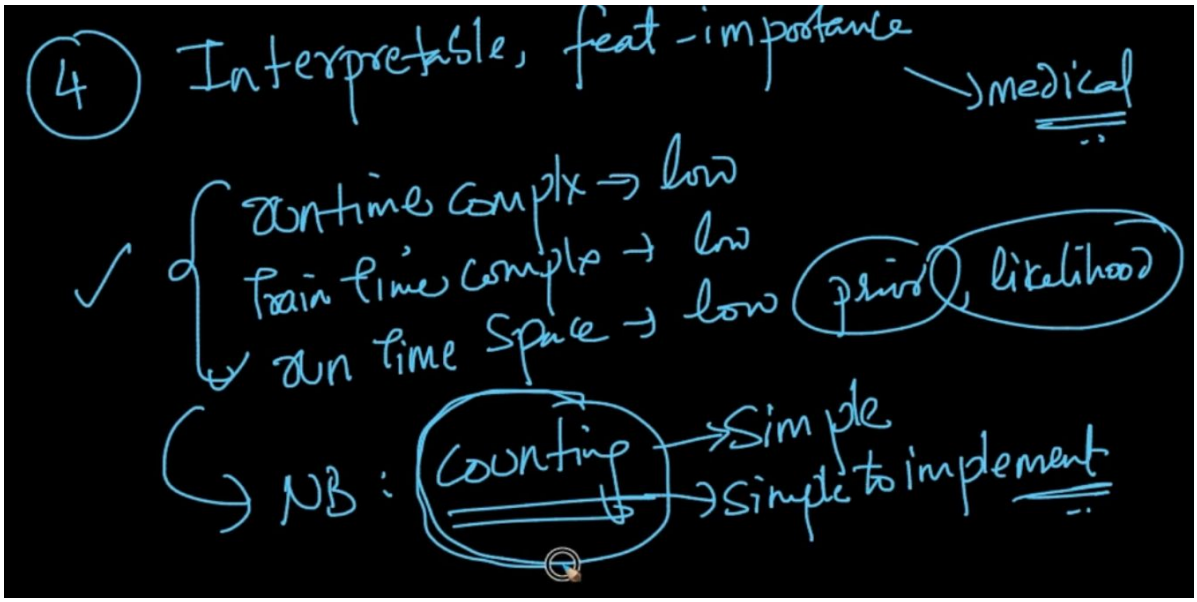
BEST AND WORSE CASES FOR NAIVE BAYES



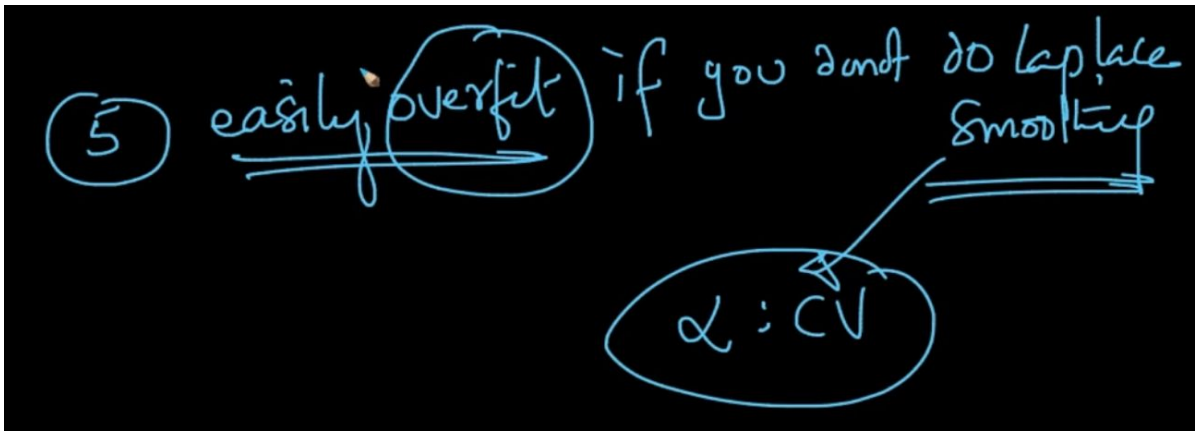
We are assuming that our features are conditional independent. Naive Bayes performs well in conditional independent features but if there is dependence then it deteriorates but if some features are dependent for ex : In review classification words are dependent like if word *good* is there then *great*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



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



Laplace Smoothing is necessary