CH- Naive Bayes.

Naive Bayes is a classification algorithm which is based on probability. Before understanding Naive Bayes, we need to be familiar with some term.

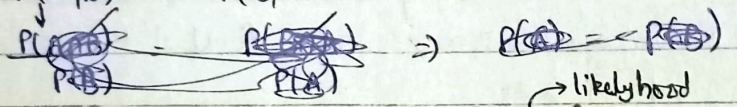
(i) Conditional probability :- $P(A/B)$:- Probability $P(A=a/B=b)$
Probability that $A=a$ given that B is b.

$$P(A/B) = \frac{P(A \cap B)}{P(B)} \quad \left\{ \text{if } P(B) \neq 0 \right.$$

(ii) Independent events :- Two events are said to be independent if $P(A/B) = P(A)$ similarly $P(B/A) = P(B)$

(iii) Mutually exclusive events :- If $P(A/B) = P(B/A) = 0$ then A & B are said to be mutually exclusive.

$P(A/B) = P(B/A) = 0$



(iv) Baye's Theorem :- $P(A/B) = \frac{P(B/A)\,P(A)}{P(B)}$, $P(B) \neq 0$

$P(A/B)$ — Posterior
$P(B/A)$ — likelyhood
$P(A)$ — Prior
$P(B)$ — evidence

Proof :- L.H.S. $P(A/B) = \frac{P(A \cap B)}{P(B)}$

$$= \frac{P(B \cap A)}{P(B)} \quad \text{— (i)}$$

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

$$P(A) * P(B/A) = P(B \cap A) \quad \text{— (ii)}$$

put value of $P(B \cap A)$ in (i)

$$P(A/B) = \frac{P(B/A) * P(A)}{P(B)} = \text{R.H.S}$$

Now let's understand Naive Bayes algorithm.

Naive Bayes is a conditional probability model: given a problem instance to be classified, represented as vector $x = (x_1, x_2 \dots x_n)$ representing n features (independent variables), it assigns to this instance probabilities

$$p(C_K \mid x_1, \dots x_n)$$

for each of K possible outcomes or classes C_K.

The problem with above formula is that if the number of features n is too large or if a feature can take on a large no. 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,

$$P(C_K \mid X) = \frac{P(X \mid C_K)\, P(C_K)}{P(X)}$$

Numerator: $P(X \mid C_K)\, P(C_K) = P(X \cap C_K)$

suppose we have 4 classes. we calculate

$$P(C_1/X) = P(X/C_1) * P(C_1) / P(X)$$
$$P(C_2/X) = P(X/C_2) * P(C_2) / P(X)$$
$$P(C_3/X) = P(X/C_3) * P(C_3) / P(X)$$
$$P(C_4/X) = P(X/C_4) * P(C_4) / P(X)$$

→ max class will choose as class label of X

↳ This is common and output is not dependent on this.

So, let's focus only on numerator for now.

$P(C_K \mid X) \propto P(C_K, X)$ → $\{P(C_K \cap X)\}$

This is called a joint probability model.

$P(C_K, X) \Rightarrow p(C_K, x_1 x_2 x_3 \dots x_n)$

$$p(C_K, x_1, x_2, x_3 \dots x_n) = p(x_1, x_2, x_3 \dots x_n, C_K)$$
$$= p(x_1 \mid x_2, x_3, \dots x_n, C_K)\, p(x_2, x_3 \dots x_n, C_K)$$
$$= p(x_1 \mid x_2, x_3 \dots x_n, C_K)\, p(x_2 \mid x_3 \dots x_n, C_K)\, p(x_3 \dots x_n, C_K)$$
$$= p(x_1 \mid x_2, x_3 \dots x_n, C_K) \dots p(x_n \mid C_K) * P(C_K)$$

Applying Probability chain rule.

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

$$p(x_i | x_{i+1}, \ldots x_n, C_K) = p(x_i | C_K)$$

Thus the joint model can be expressed as.

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

That means under above assumption, the conditional distribution over the class variable C is:

$$p(C_K | x_1, \ldots, x_n) = \frac{p(C_K) \prod_{i=1}^{n} p(x_i | C_K)}{p(X)}$$

→ Constructing a classifier from the above model:
Naive Baye's classifier combines this model with a decision rule. One common rule is to pick the hypothesis that is most probable. This is known as maximum a posteriori or MAP decision rule.

$$\hat{y} = \max_{k \in \{1 \ldots K\}} p(C_K) \prod_{i=1}^{n} p(x_i | C_K)$$

↑ Predicted class label.

Let's take an example:-

	Outlook	Temperature	Humidity	Wind	Response: (Play = YES/NO)
Day 1	Sunny	Hot	High	Weak	NO
Day 2	Sunny	Hot	High	Strong	NO
Day 3	Overcast	Hot	High	Weak	Yes
Day 4	Rain	Mild	High	Weak	Yes
Day 5	Rain	Cool	Normal	Weak	Yes
Day 6	Rain	Cool	Normal	Strong	NO
Day 7	Overcast	Cool	Normal	Strong	Yes
Day 8	Sunny	Mild	High	Weak	No
Day 9	Sunny	Cool	Normal	Weak	Yes
Day 10	Rain	Mild	Normal	Weak	Yes
Day 11	Sunny	Mild	Normal	Strong	Yes
Day 12	Overcast	Mild	High	Strong	Yes
Day 13	Overcast	Hot	Normal	Weak	Yes
Day 14	Rain	Mil	High	Strong	NO

The learning phase :- In learning phase, we compute the table of likelihoods from the training data : They are:

$P(Outlook=o \mid class\ play=b)$ where $o \in [Sunny, overcast, rainy]$, $b \in [Yes, no]$

$P(Temp=t \mid class\ play=b)$ where $t \in [Hot, Mild, cool]$

$P(Humidity=h \mid class\ play=b)$ where $h \in [High, normal]$

$P(Wind=w \mid class\ play=b)$ where $w \in [weak, strong]$

∴ $P(o \mid c)$

	Yes	No	P(o/Yes)	P(o/No)
Sunny	2	3	2/9	3/5
Overca	4	0	4/9	0/5
Rain	3	2	3/9	2/5
	9	5		

Similarly $P(t \mid c)$

	Y	N	P(t/Yes)	P(t/No)
Hot	2	2	2/9	2/5
mild	4	2	4/9	2/5
Cold	3	1	3/9	1/5
	9	5		

P(H/C)

	Yes	No	P(H/Y)	P(H/N)
High	3	4	3/9	4/5
Normal	6	1	6/9	1/5
	9	5		

P(W/C)

	Yes	N	P(S/Y)	P(S/N)
Strong	3	3	3/9	3/5
Weak	6	2	6/9	2/5

$P(\text{class} = \text{Yes}) = 9/14$

$P(\text{class} = \text{No}) = 5/14$.

→ Classification phase: Let's say we get a new instance of the weather condition, $x' = (\text{sunny}, \text{cool}, \text{high}, \text{strong})$ we have to classify it.

$$\hat{y} = \max\left(\{P(\text{Yes}) * P(\text{sunny}/\text{Y}) * P(\text{cool}/\text{Yes}) * P(\text{high}/\text{Yes}) * P(\text{strong}/\text{Yes})\}, \{P(\text{No}) * P(\text{sunny}/\text{N}) * P(\text{cool}/\text{N}) * P(\text{high}/\text{N}) * P(\text{strong}/\text{N})\}\right)$$

$$\hat{y} = \max\left\{\frac{9}{14} * \frac{2}{9} * \frac{3}{9} * \frac{3}{9} * \frac{3}{9}, \frac{5}{14} * \frac{3}{5} * \frac{1}{5} * \frac{4}{5} * \frac{5}{5}\right\}$$

$\hat{y} = \max\{0.0053, 0.0205\}$ → max

$\therefore \hat{y} = \text{No}$

Time complexity : $O(n * d)$ → Training phase

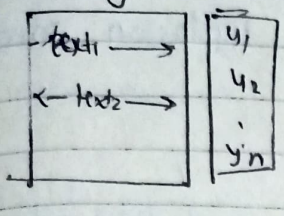
space complexity : $O(d * c)$

Time comp : $O(d * c)$ → Test time

(d → No. of features, c → No. of classes)

* Naive Bayes is much more memory efficient at run time.

Naive Bayes on text data :-



text $\longrightarrow$ {preprocessing} $\longrightarrow$ $\{w_1, w_2, \ldots w_d\}$

$$P(y=1 \mid \text{text}) \propto P(y=1 \mid w_1 w_2 \ldots w_d)$$
$$\propto P(y=1) * P(w_1 \mid y_1=1) * P(w_2 \mid y_2=1) \cdots P(w_d \mid y=1)$$
$$\propto P(y=1) * \prod_{i=1}^{d} P(w_i \mid y=1)$$

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

If $P(y_1=1 \mid \text{text}) > P(y=0 \mid \text{text})$ $\quad \hat{y}=1$

else $\quad \hat{y}=0$

→ Laplace smoothing :- At the end of training a Bayes classifier we have,

$P(y=1)$, $P(y=0)$

$P(w_1 \mid y=1)$ $P(w_2 \mid y=1)$ $\ldots$ $P(w_d \mid y=1)$ } → likelihoods.

$P(w_1 \mid y=0)$ $P(w_2 \mid y=0)$ $\ldots$ $P(w_d \mid y=0)$

Now at Test stage :- $\text{text}_q = (w_1, w_2, w_3, w')$ (w_1, w_2, w_3 → These are old words; w' → new word)

So, what we will do, if we get new word in test data.

$$P(y=1 \mid \text{text}_q) = P(y=1 \mid w_1 w_2 w_3 w')$$
$$= P(y=1) * P(w_1 \mid y=1) * P(w_2 \mid y=1) * P(w_3 \mid y=1) * P(w' \mid y=1)$$

How to find this

$$P(w' \mid y=1) = \frac{P(w', y=1)}{P(y=1)} : \frac{\text{No. of data points s.t } w' \text{ exists and } y=1}{\text{No. of points where } y=1}$$

$= \frac{0}{\text{No. of pts } y=1}$ → 0, because there are no data point which contains w', because w' comes (for the) now)

$= 0$

But we cannot put 0 because we will get $P(y=1 \mid \text{text}_q) = P(y=0 \mid \text{text}_q) = 0$

To remove this we you use
Laplace smoothing (it is Laplace not Laplacian smoothing)

it says:- $P(w'/y=1) = \frac{\alpha}{n + \alpha K}$

↳ K = No. of distinct values w' can take.

(in our case w' can be present or not present
hence w' can have 2 value $\therefore K = 2$)
α can be any value, typically $\alpha = 1$

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

Case let a case where $\alpha = 100000$

$P(w'/y=1) = \frac{100000}{n_1 + 200000}$ let $n_1 = 100 = \frac{100000}{200000} \approx 0.5$

when α is large we get a balanced probability.

So, for each word weather it is present in having our

$$P(w'/y=1) = \frac{\#\text{datapoint with } w' \text{ and } y=1 + \alpha}{\#\text{ data points } y=1 + K\alpha}$$

as $\alpha \uparrow \rightarrow$ moving my likelihood prob to uniform distribution

One more problem is:
we know $P(y=1/w_1 w_2 \ldots w_d)$

$= P(y=1) * P(w_1/y=1) * P(w_2/y=1) \ldots P(w_d/y=1)$

All these values are b/w 0 to 1 and hence if d is very large this number becomes very very small. It is called numerical stability.
In lang like python, C++ etc, we can have numerical underflow (i.e we cannot represent them)

So, solution is take log of the probabilities.

$P(y=1/w_1, w_2 \dots w_d) \propto P(y=1) \prod_{i=1}^{d} P(w_i/y=1)$

$P(y=0/w_1, w_2 \dots w_d) \propto P(y=0) \prod_{i=1}^{d} P(w_i/y=0)$

Instead of comparing these probabilities, we can compare log of these probabilities.

$$\log\{P(y=1/w_1, w_2 \dots w_d)\} \Rightarrow \log(P(y=1)) + \log(P(w_i/y=1)) + \log(P(w_2/y=1)$$

$$:= \log(P(y=1)) + \sum_{i=1}^{d} \log(P(w_i/y=1))$$

Bias - variance tradeoff in Naive Bayes:-

In Naive Bayes, only one parameter i.e. α (laplace smoothing) decides under fitting & overfitting.

Case 1: $\alpha = 0$

$$P(w_i/y=1) = \frac{\#\text{ train datapoint with } w_i \text{ occur \& } y=1}{\#\text{ train point with } y=1}$$

w_i occur only 2 times and there are 1000 train datapoints $y=1$

$$P(w_i/y=1) = \frac{2}{1000}$$

Here even for a rarer word we are assigning some prob. hence we are overfitting in a way. Now suppose $\alpha = 0$

Then $P(w_i/y=1) = \frac{2+0}{1000+2*0} = \frac{2}{1000}$

If we remove these 2 words then probabilities become 0 that mean even for a small change model changes drastically i.e. model have high variance when α is small hence $\alpha = 0$ have high variance problem i.e. overfitting

Case 2: α is very large: $\alpha = 1000$

$$P(w_i/y=1) = \frac{2+1000}{1000+2000} = \frac{1002}{2000} \approx 0.5$$

when α is very large, $P(w_i/y=1) \approx P(w_i/y=0) \approx 0.5$

hence it is under fitting.

hence in high α, every word have approx prob ≈ 0.5 hence this is a case of underfitting. When $\alpha \uparrow$ every time predicted class is mgority class.
Hence we need a balanced α.

How to find the right α. similar to K-NN, finding K we used cross-validation, same here we use cross validation to find α.

→ Feature importance & interpretability:-
In KNN we used feature selection to find feature importance. Now let see this in NB.
In NB, we have likelihood probability
$\forall w_i$, $p(w_i/y_i=1)$
$\forall w_i$, $p(w_i, y=0)$
If we sort the words on the basis $p(w_i/y=1)$ in decreasing order so, top words occur very often because their probability is very high. We can say that, words which have high value of probability $\in$ are the important feature in determining that a data point belongs to corresponding class.
similarly we can sort on basis of $p(w_i/y=0)$.

Feature importance in Naive base is simple to calculate.

Now, what about Interpretability:-
let we have x_q and declare that $y_q = 1$, we can say that $y_q = 1$ because we have pro$(w_i/y=1)$ is high. we can make statements like this. i.e. we can give reasoning for our prediction. Hence NB is a interpretable model.

→ Imbalanced data:- what happen to NB when we have Imbalanced data set.

$n \rightarrow n_1$ +ve, $n \rightarrow n_2$ −ve

$n_1 >> n_2$ { let $n_1 = 90\%$, $n_2 = 10\%$.

$\frac{n_1}{n} = 0.9$ $\frac{n_2}{n} = 0.1$

$$P(y=1 \mid w_1 w_2 \cdots w_d) = P(y=1) \cdot \prod_{i=1}^{d} P(w_i \mid y=1)$$

$$P(y=0 \mid w_1 w_2 \cdots w_d) = P(y=0) \prod_{i=1}^{d} P(w_i \mid y=0)$$

(0.9 and 0.1 above $P(y=1)$ and $P(y=0)$ respectively)

if these values are equal then $y=1$ have a biasness.

∴ (i) because of class priors → majority / dominating class has an advantage.

what is solution?

Soln :- (i) use the standard tech of upsampling or down sampling.

(ii) simply drop the class prior terms.

$$P(y=1 \mid w_1 w_2 \cdots w_d) = \prod_{i=1}^{d} P(w_i \mid y=1)$$

$$P(y=0 \mid w_1 w_2 \cdots w_d) = \prod_{i=1}^{d} P(w_i \mid y=0)$$

(iii) modified NB formulation to account for imbalanced data. But they are not often used.

There is another problem with imbalanced data.

$n \rightarrow n_1$ +ve (90%) → $P(w_i \mid y=1) \Rightarrow$ (0 to 900)

$n \rightarrow n_2$ −ve (10%) $P(w_i \mid y=0) \Rightarrow$ (0 to 100)

This have more possibilities hence it tends to be larger.

And hence if we give α, then α impacts more of minority class and less impact on majority class.

Ex:- let $P(w_i \mid y=0) = \frac{2}{100} = 2\%$ $P(w_i \mid y=1) = \frac{18}{900} = 2\%$

$\alpha = 10$

$\frac{2+10}{120} = 10\%$ (+8%) $\frac{2+10}{920} = 3.04\%$ (+1.04%)

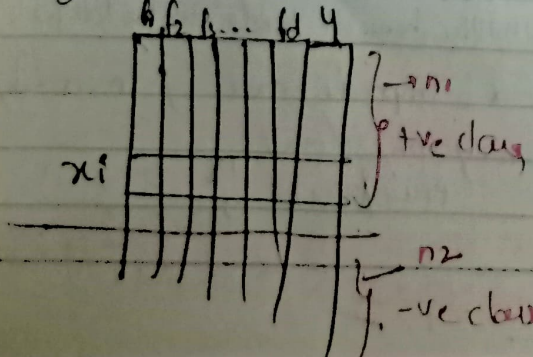
→ Outlier How are outlier handled in naive based.
If we get an outlier at test time, laplace smoothing take care of it.
So, what happens to outlier in training data.
{w1, w2, w3, ... wm} ← set of words in Dtrain

Let w8 occurs very few times in the +ve class.
Then we can say w8 is outlier.
Then soln is: if a word(w_j) occurs fewer than Threshold then just ignore that word. and drop that word.

Second soln is to use laplace smoothing in training data also.

→ Missing value
(i) Case-1: Text-data :- There is no case of missing data in Text-data
(ii) Case-2: categorical feature :- If a categorical feature is missing, then we can assume a another category NaN or null.
(iii) Case-3:- Numerical feature :- If we have numerical feature is missing we can use imputation as discussed previously.

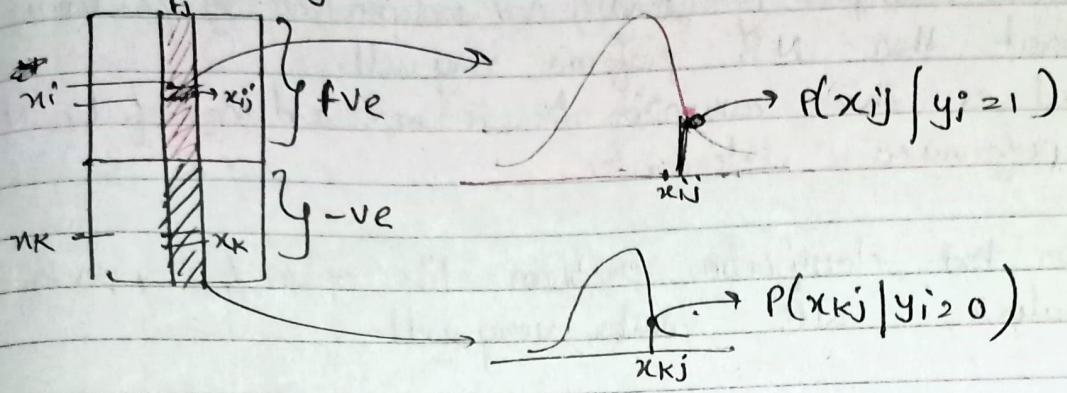
☆ NB with Numerical feature:-
Till now we have seen NB on categorical feature and Binary (text) feature. Now suppose we have feature with numerical data.



let f_j is Real valued.

$$P(y_i=1 \mid x_{i1}, x_{i2}, x_{i3} \ldots x_{id}) \propto \underbrace{P(y_i=1)}_{\text{class prior}} \prod_{j=1}^{d} \cdot P(x_{ij} \mid y_i=1)$$

$= P(y_i=1) \prod_{j=1}^{d} P(x_{ij} \mid y_i=1)$



Hence, we can use PDF to get $P(x_{ij} \mid y_i = k)$
↓ PDF of feature j only on the data points of class K

people usually assume fj in $D'_{\downarrow D^{+ve}}$ → Gaussian distn

" " " fj in $D''_{\downarrow D^{-ve}}$ → " "

And such a model with above 2 assumption, is called Gaussian Naive based.

→ Multiclass classification:- Naive Bae can do multiclass classification.

→ Similarity matrix or distance matrix:-
Can NB do classification if instead of data point distance or similarity matrix is given. In General, NB cannot use distance or similarity matrix. because NB is a probability based model where we need actual data values.

→ Large dimensionality:- NB is extensively used in Text class which is a higher dimension data. Hence NB can handle large dimensional data. only catch is we need to use log-probabilities.

→ Best & worst case for NB

① If our assumption of conditional independence of features is correct then NB performs very well.
But as this assumption become more and more false NB performance deteriorate.

② For text classification problem like spam filter, review analysis, NB works very well.

③ NB also perform well for categorical & Binary value.

④ for real values and numerical value feature NB is not often used.

⑤ NB is very interpretable, we can easily get feature importance.

⑥ Run time complexity is low, training time complexity is low Run time space complexity is also low.

⑦ ** We can easily overfit, if we don't do laplace smoothing and get right α.