Naive Bayes

Conditional Probability :-

$P(A|B)$ = Probability of an event A given that event B has occured.

$$= \frac{P(A \cap B)}{P(B)} \quad ; \quad P(B) \neq 0$$

Independent Events

A, B are said be "independent" if :

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

A : Getting value of 6 in die 1 throw $(D_1 = 6)$.

B : Getting a value of 3 in die 2 throw. $(D_2 = 3)$.

Both the dice are independent of each other ☺

Mutually exclusive events :

If $P(A|B) = P(B|A) = 0$, then A and B are said to be mutually exclusive.

A = 6 appears on die, D1

B = 3 " " " , D1

Bayes Theorem

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

Proof :-

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

$$\Uparrow$$

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

Naive Bayes Algorithm

$X = (x_1, x_2, x_3, x_4, \ldots, x_n)$ → n-dimensional vector of all column values,

It tries to calculate, $p(C_k \mid x_1, x_2, \ldots, x_n)$

$C_k \Rightarrow$ Classes or Target variable

Now we know that,

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

$P(X)$ ⟶ Constant for each C_k

→ To determine this, only numerator is enough as denominator is constant.

$$P(X \cap C_k) = P(C_k, X) = P(C_k \text{ and } X)$$

$P(A, B) = P(A \mid B) \cdot P(B)$

or $P(A, B, C) = P(A \mid B, C)\, P(B, C)$

So, Using Chain rule of probability :-

$$P(C_k, x_1, x_2, x_3, \ldots, x_n) = P(x_1, x_2, \ldots, x_n, C_k)$$

$$= p(x_1 \mid x_2, x_3, \ldots, x_n, C_k) \cdot p(x_2, x_3, \ldots, x_n, C_k)$$

$$= p(x_1 \mid x_2, x_3, \ldots, x_n, C_k) \cdot p(x_2 \mid x_3, x_4, \ldots, x_n, C_k)\, p(x_3, x_4, \ldots, x_n, C_k)$$

....

$$= p(x_1 \mid x_2, x_3, \ldots, x_n, C_k)\, p(x_2 \mid x_3, x_4, \ldots, x_n, C_k) \cdots\cdots p(x_{n-1} \mid x_n, C_k)\, p(x_n \mid C_k)\, p(C_k).$$

Now, here the "Naive" assumption of "Conditional Independence" comes into picture.

$$P(A|B) = P(A) \Rightarrow$$ ~~Compl~~ Independent events

$$P(A|B,C) = P(A|C) \Rightarrow$$ Conditionally Independent.

$$\cancel{p(C_k | x_1, x_2, \ldots, x_n) \propto}$$

So, $p(x_i | x_{i+1}, \ldots x_n, C_k) = p(x_i|C_k)$ [Using Naive assumption]

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

Multiplication

$$P(C_k | x_1, x_2, \ldots x_n) = \frac{1}{Z}\, p(C_k) \prod_{i=1}^{n} p(x_i|C_k) \quad , \quad Z = p(x)$$

Time Complexity :- $O(nd)$

Space Complexity :- $O(d*c) \Rightarrow$ To store $d*c$ likelihood probabilities

$d \rightarrow$ no. of columns

$c \rightarrow$ no. of classes

Naive Bayes is much better ~~as~~ in terms of space complexity at runtime, i.e., $O(d*c)$

Naive Bayes for Text classification

It is used as a Baseline model ~~to~~ in ~~class~~ classification problems.

	Class
← $Text_1$ →	0
← $Text_2$ →	1
	0
	1

Task :-?

$$P(y=0 \mid text_q) = ?$$

$$P(y=1 \mid text_q) = ?$$

After preprocessing of text like :- Stopwords removal, Stemming, n-grams

We get a particular sequence of words :- $\{w_1, w_2, w_3 \ldots, w_n\}$

↓

Suppose we use Binary BOW on this.

$$P(y=1 \mid text) = P(y=1 \mid w_1, w_2, \ldots, w_d)$$

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

$$\text{So, } P(y=1 \mid text) \propto \underbrace{P(y=1) * \prod_{i=1}^{d} P(w_i \mid y=1)}_{\text{Each term is easy to calculate.}}$$

$$P(w_i \mid y=1) \Rightarrow \frac{\text{No. of pts containing } w_i \text{ with } y=1}{\text{No. of pts with } y=1}$$

Laplace Smoothing / Additive Smoothing

During training, we have the following values:-

$$P(y=1),\ P(y=0),\ P(w_1|y=1),\ P(w_1|y=0), \ldots\ \ldots\ P(w_m|y=1),\ P(w_m|y=0)$$

Now, suppose in the test data ~~set~~ we have a $text_q$ containing an unknown word.

$$P(y=1|text_q) = P(y=1|w_1 w_2 w_3 \ldots w')$$
$$= P(y=1) * P(w_1|y=1) * P(w_2|y=1) \ldots * P(w'|y=1)$$

Unknown to model

We can't just simply ignore it as in that case $P(w'|y=1)=1$ and $P(w'|y=0)=0$ will be considered which is logically incorrect.

$$P(w'|y=1) = \frac{P(w', y=1)}{P(y=1)} = \frac{\#\text{ train pts in which } w' \text{ occurs and } y=1}{\#\text{ train pts where } y=1}$$

$$= \frac{0}{n_1}$$

We also can't consider it 0 as the whole expression will then become 0.

So, what happens in Laplace Smoothing is that:-

$$P(w'|y=1) = \cancel{\frac{0+\alpha}{100+K\alpha}} \quad \frac{0+\alpha}{100+K\alpha}$$

$n_1 = 100$

K :- Possible values w' can take $= 2$

So, $K=2$:- if $\alpha=1 \Rightarrow P(w'|y=1) = \frac{1}{102} \neq 0$

If α is large suppose, $\alpha = 10{,}000$.

$$P(w'|y=1) = \frac{10000}{20100} \approx \frac{1}{2}$$

Now here we are considering that $P(w'|y=1) = P(w'|y=0) = \frac{1}{2}$

The w' has equal chance of occuring when $y = 0, 1$.

Even with small α we are getting rid of multiplying with 0.

for all words

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

present in my training data

Log Probabilities for numerical stability

$0.2 \times 0.004 \times 0.003 \times \ldots$ what happens in Python is that for a it start approximating after too many decimal digits which can led to errors.

$$P(y=1|w_1, w_2, \ldots, w_d) = P(y=1) * \prod_{i=1}^{d} P(w_i|y=1)$$

$$P(y=0|w_1, w_2, \ldots, w_d) = P(y=0) * \prod_{i=1}^{d} P(w_i|y=0)$$

Instead of calculate the above expression, their log is calculated

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

Bias - Variance Tradeoff

High Bias → Underfitting
High Variance → Overfitting

{ α in Laplace smoothing is the hyper-parameter that determines underfit and overfit }

Now, let understand this with two extreme cases:-

Case I :- $\alpha = 0$

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

Suppose, $n = 2000 \begin{cases} 1000 \text{ +ve} \\ 1000 \text{ -ve} \end{cases}$ $= \frac{2}{1000}$ → words w_i occurs in only 2 texts

So, overfitting means that small changes in ~~traing~~ training data result in model change dramatically.

Since w_i occurs only in 2 texts out of 2000 cases :-

↳ Small change in D_{train} → Removing the 2 texts containing w_i

↓

Probability, $P(w_i/y=1) = \frac{2}{100}$ to $\frac{0}{100}$

Case II :- α is V. large, $\alpha = 10{,}000$

$$P(w_i | y=1) = \frac{2 + 10{,}000}{1000 + \underset{\substack{\uparrow \\ k=2}}{20000}} \approx \frac{1}{2}$$

↓

Can have 2 values 0 or 1

So, if α = V. large

$$P(y=1 \mid \underbrace{w_1 w_2 \ldots w_d}_{x_q}) = P(y=1) * \underbrace{\prod_{i=1}^{d} P(w_i \mid y=1)}_{(1/2)^d \rightarrow \text{constant}} \quad \leftarrow ①$$

$$P(y=0 \mid w_1, w_2, \ldots w_d) = P(y=0) * \underbrace{\prod_{i=1}^{d} P(w_i \mid y=0)}_{(1/2)^d \rightarrow \text{constant}} \quad - ②$$

So, the final result will be actually decided by $P(y=1)$ and $P(y=0)$

So, the new datapoint will ~~of~~ be assigned with the label of majority class.

If $P(y=1) > P(y=0) \Rightarrow$ All new pts will be assigned with label 1.

Feature ~~Selection~~ Importance and interpretability

Feature importance in Naive Bayes can directly be obtained from the model.

+ve class :- Find words (w_i) /features with highest value of $P(w_i/y=1)$

-ve class :- " " " " " " $P(w_i/y=0)$

Imbalanced Data

$$n \rightarrow \begin{cases} n_1 & +ve & 90\% \\ n_2 & -ve & 10\% \end{cases} \qquad n_1 >> n_2$$

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

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

If these terms are same

Then majority class will decide the test data

Q How to deal with this Problem?

→ Upsampling or downsampling

↳ $n_1 \approx n_2$ $\quad P(y=1) = P(y=0) = \frac{1}{2}$

→ drop $P(y=1)$ & $P(y=0)$

, i.e., $P(y=1) = P(y=0) = 1$

They won't affect the comparison after becoming equal

→ Modified NB to account for class imbalance

↳ Not often used

One more problem with imbalanced data is:-
Minority class gets impacted more with same α as compared to majority class.

~~Max~~ (-ve) : $P(w_i \mid y=0) = \frac{2}{100} = 2\%$
Minority

Significant change of 8%

Let $\alpha = 10$

So, $P(w_i \mid y=0) = \frac{12}{120} = 10\%$

Majority (+ve) : $P(w_i \mid y=1) = \frac{18}{900} = 2\%$

Let $\alpha = 10$

Change of 1.04%

$= \frac{28}{920} = 3.04\%$

Hence, Same α impacts both class differently.

Handling Outliers

During training time, either outliers should be ignored or they can be handled by Laplace smoothing.

Missing Values :-

In case of text data there are no missing features.

But in case of categorical data, missing values can themselves be treated as a separate category.

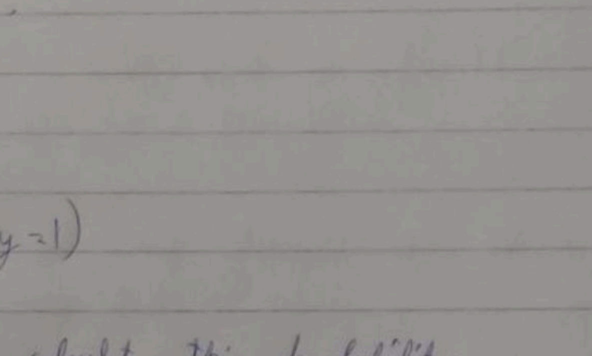
Handling Numerical Features

$$P(\underbrace{f_j}_{\text{numerical}} \mid y=0) \quad \text{or} \quad P(f_j \mid y=1)$$

So, how do we calculate this probability.

	f_1	f_2	... f_j	...	f_d	y
						1
x_i	x_{i1}	x_{i2}	...	...	x_{id}	1
						1
						0
						0
						0

So, for a particular feature f_j, it is considered as Gaussian distribution and its PDF is drawn.



For the whole column where $y=1$ its (μ_{ij}, σ_{ij}) is calculated

To plot this

This is how probability of a particular feature is calculated.

$P(w|y=1)$

This is called Gaussian NB, w is Gaussian.
when $w \in [0,1]$, it's Bernoulli NB.
when w can have multiple values, it is Multinomial NB.

Similarity matrix or Distance Matrix

Naive Bayes can't be used when we are provided with

Large Dimensionality :-

It can definitely handle data with large dimensionality but log-probabilities should be used to bring Numerical Stability

Best and Worst Cases of Naive Bayes

1) ~~If~~ In case of NB, if conditional independence ~~tale~~ is :-
True :- Performs really well.
False :- It degrades and ~~detoro~~ deteriorates

In real world, some features are dependent, in those cases too Naive Bayes performs reasonably well.

2) Text classification → Email Spam, Review Polarity { High Dimensionality data }

NB can be used as a baseline model.

3) Categorical Features ⟶ NB performs good
Real Valued features ⟶ Only sometimes do good.
(As in real world ~~Ew~~ we don't get Gaussian Distribution)

4) ~~Interpretable~~ It is interpretable, feature importance can be easily found.

Runtime Complexity → Low
Traintime Complexity → Low
Run time Space Complexity → Low