Natural Language Processing with Classification and Vector Spaces || Week — 02 (Naive Bayes)

medium.com/@aakashgoel12/natural-language-processing-with-classification-and-vector-spaces-week-02-naive-bayes-5f702065d209

July 21, 2020

Called Naive → Features used for classification are all independent which in reality is the case.

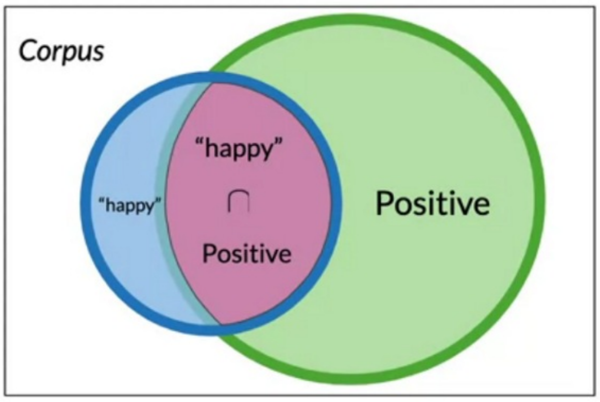
A → Prob. of tweet being positive

B → Prob. of tweet containing happy

P(A,B) = P(A∩B) = No. of times tweet is +ve as well as contain happy / Total no of tweets

Derive Bayes Rule

Conditional probabilities



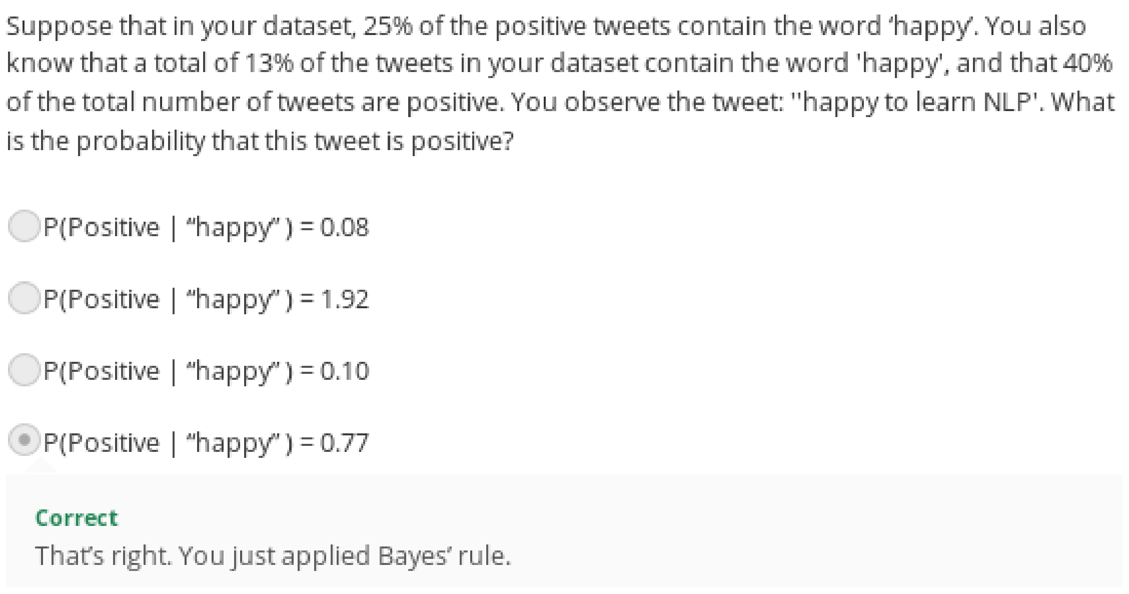
$$P(\text{Positive}|\text{“happy”}) = \frac{P(\text{Positive} \cap \text{“happy”})}{P(\text{“happy”})}$$

STEP 01: P(A/B) can be written as P(A∩B)/P(B)

$$P(\text{Positive}|\text{“happy”}) = \frac{\boxed{P(\text{Positive} \cap \text{“happy”})}}{P(\text{“happy”})}$$

$$P(\text{“happy”}|\text{Positive}) = \frac{\boxed{P(\text{“happy”} \cap \text{Positive})}}{P(\text{Positive})}$$

STEP 02: Replace Numerator of 1st using 2nd eqn.



Suppose that in your dataset, 25% of the positive tweets contain the word ‘happy’. You also know that a total of 13% of the tweets in your dataset contain the word 'happy', and that 40% of the total number of tweets are positive. You observe the tweet: ''happy to learn NLP'. What is the probability that this tweet is positive?

- P(Positive | “happy”) = 0.08
- P(Positive | “happy”) = 1.92
- P(Positive | “happy”) = 0.10
- [✓ P(Positive | “happy”) = 0.77]

Correct
That’s right. You just applied Bayes’ rule.

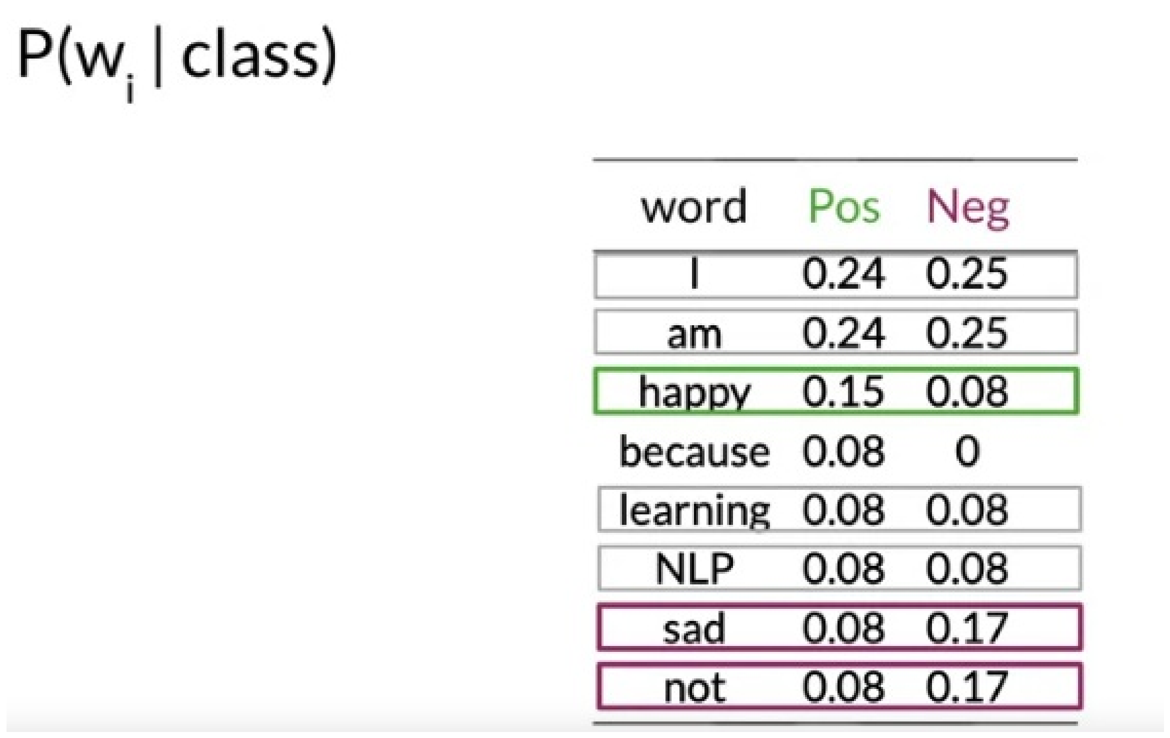
Question on Bayes Rule

P(POS/”happy”) = { P(“happy”/POS) * P(POS) }/ (P(“happy”))

P(POS/”happy”) = P(POS,”happy”)/p(“happy”)
0.77 = P(POS,”happy”)/(0.13)

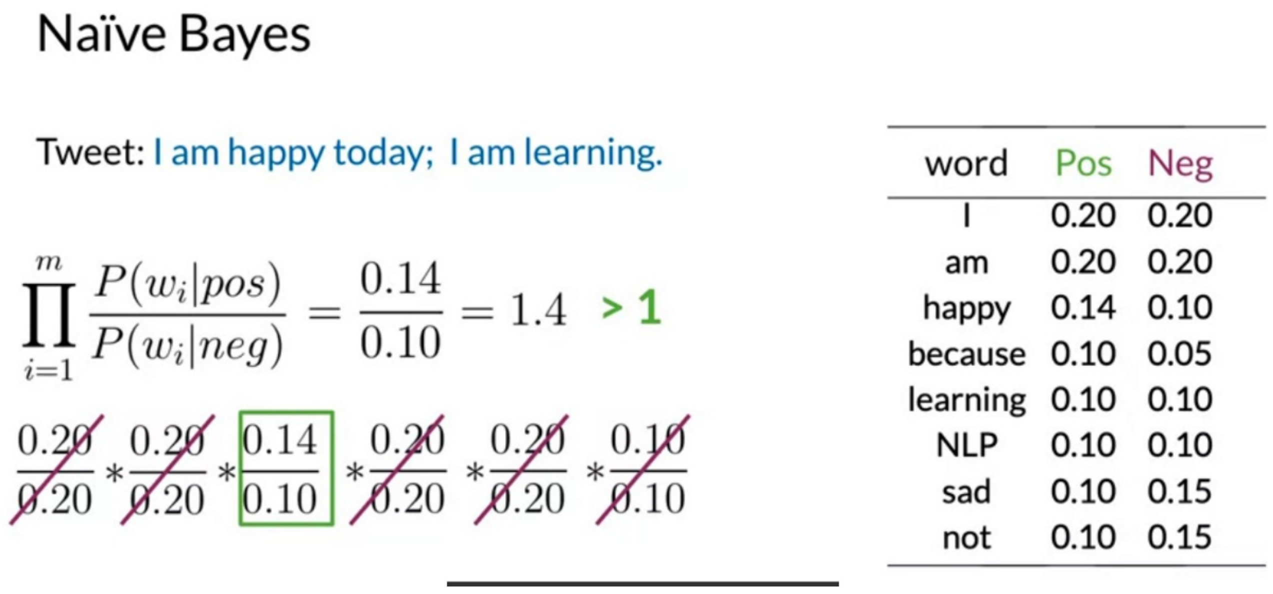
Impt. How to find power words i.e. signifying positive and negative sentiment word.

Answer. See difference or ratio of Prob. of that words for Positive and negative sentiment using frequency of that word in positive tweet/freq. of all words in positive tweet.. If ratio>1, positive word ; ratio<1, negative word



P(w_i | class)

word	Pos	Neg
I	0.24	0.25
am	0.24	0.25
happy	0.15	0.08
because	0.08	0
learning	0.08	0.08
NLP	0.08	0.08
sad	0.08	0.17
not	0.08	0.17



Naïve Bayes

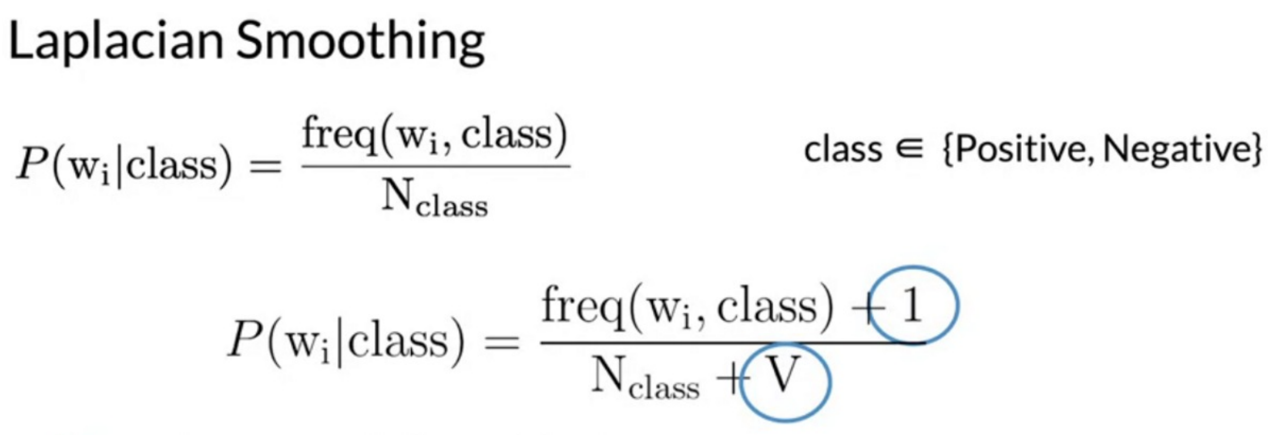
Tweet: I am happy today; I am learning.

$$\prod_{i=1}^{m} \frac{P(w_i|pos)}{P(w_i|neg)} = \frac{0.14}{0.10} = 1.4 \quad > 1$$

$$\frac{\cancel{0.20}}{\cancel{0.20}} * \frac{\cancel{0.20}}{\cancel{0.20}} * \boxed{\frac{0.14}{0.10}} * \frac{\cancel{0.20}}{\cancel{0.20}} * \frac{\cancel{0.20}}{\cancel{0.20}} * \frac{\cancel{0.10}}{\cancel{0.10}}$$

word	Pos	Neg
I	0.20	0.20
am	0.20	0.20
happy	0.14	0.10
because	0.10	0.05
learning	0.10	0.10
NLP	0.10	0.10
sad	0.10	0.15
not	0.10	0.15

Inference Rule ..



Laplacian Smoothing

$$P(\mathrm{w_i}|\mathrm{class}) = \frac{\mathrm{freq}(\mathrm{w_i}, \mathrm{class})}{\mathrm{N_{class}}}$$

class ∈ {Positive, Negative}

$$P(\mathrm{w_i}|\mathrm{class}) = \frac{\mathrm{freq}(\mathrm{w_i}, \mathrm{class}) + 1}{\mathrm{N_{class}} + \mathrm{V}}$$

N_{class} = frequency of all words in class

V = number of unique words in vocabulary

Laplacian Smoothing to avoid zero prob..

Prior ratio and Likelihood

If we had no specific information and blindly picked a tweet out of the population set, what is the probability that it will be positive versus that it will be negative? That is the "prior".

$$\boxed{\frac{P(pos)}{P(neg)}}\boxed{\prod_{i=1}^{m}\frac{P(w_i|pos)}{P(w_i|neg)}}$$

Prior ratio & Likelihood

Calculate the likelihoods for each tweet

For each tweet, we have calculated the likelihood of the tweet to be positive and the likelihood to be negative. We have calculated in different columns the numerator and denominator of the likelihood ratio introduced previously.

$$\log\frac{P(tweet|pos)}{P(tweet|neg)} = \log(P(tweet|pos)) - \log(P(tweet|neg))$$

$$positive = \log(P(tweet|pos)) = \sum_{i=0}^{n}\log P(W_i|pos)$$

$$negative = \log(P(tweet|neg)) = \sum_{i=0}^{n}\log P(W_i|neg)$$

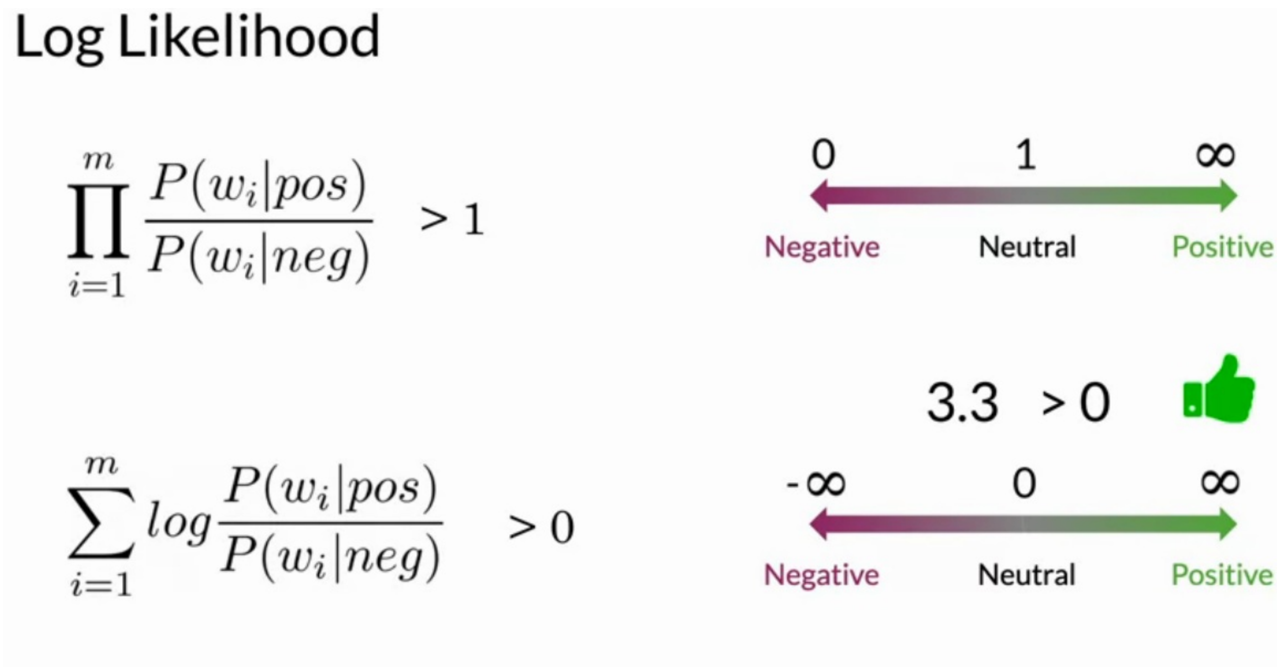
Importance of Log

Log Likelihood

$$\frac{P(pos)}{P(neg)}\prod_{i=1}^{m}\frac{P(w_i|pos)}{P(w_i|neg)}$$

- Products bring risk of underflow
- $log(a*b) = log(a) + log(b)$
- $log(\frac{P(pos)}{P(neg)}\prod_{i=1}^{n}\frac{P(w_i|pos)}{P(w_i|neg)}) \Rightarrow \underbrace{log\frac{P(pos)}{P(neg)}}_{\text{log prior}} + \underbrace{\sum_{i=1}^{n}log\frac{P(w_i|pos)}{P(w_i|neg)}}_{\text{log likelihood}}$

Log Likelihood



Log Likelihood
$\prod_{i=1}^{m} \frac{P(w_i|pos)}{P(w_i|neg)} > 1$
0
1
∞
Negative
Neutral
Positive
3.3 > 0
$\sum_{i=1}^{m} log \frac{P(w_i|pos)}{P(w_i|neg)} > 0$
-∞
0
∞
Negative
Neutral
Positive

Log Likelihood Sentiment Scale

$$\text{logprior} = log\left(\frac{P(D_{pos})}{P(D_{neg})}\right) = log\left(\frac{D_{pos}}{D_{neg}}\right)$$

Log prior

We'll use these to compute the positive and negative probability for a specific word using this formula:

$$P(W_{pos}) = \frac{freq_{pos} + 1}{N_{pos} + V}$$
$$P(W_{neg}) = \frac{freq_{neg} + 1}{N_{neg} + V}$$

Notice that we add the "+1" in the numerator for additive smoothing. This wiki article explains more about additive smoothing.

Log likelihood

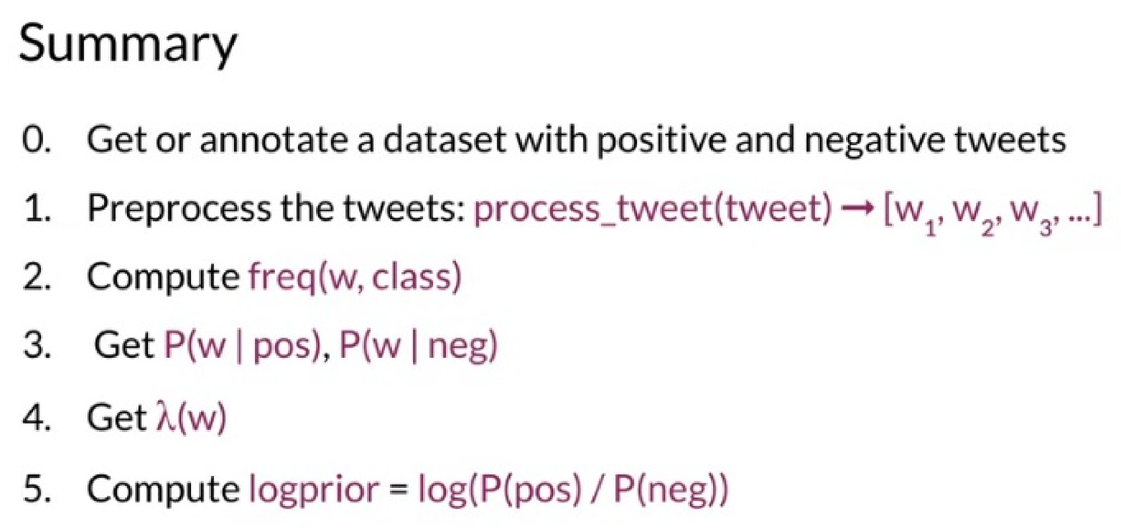
To compute the loglikelihood of that very same word, we can implement the following equations:

$$\text{loglikelihood} = \log\left(\frac{P(W_{pos})}{P(W_{neg})}\right)$$

Log Likelihood

$$p = logprior + \sum_{i}^{N}(loglikelihood_i)$$

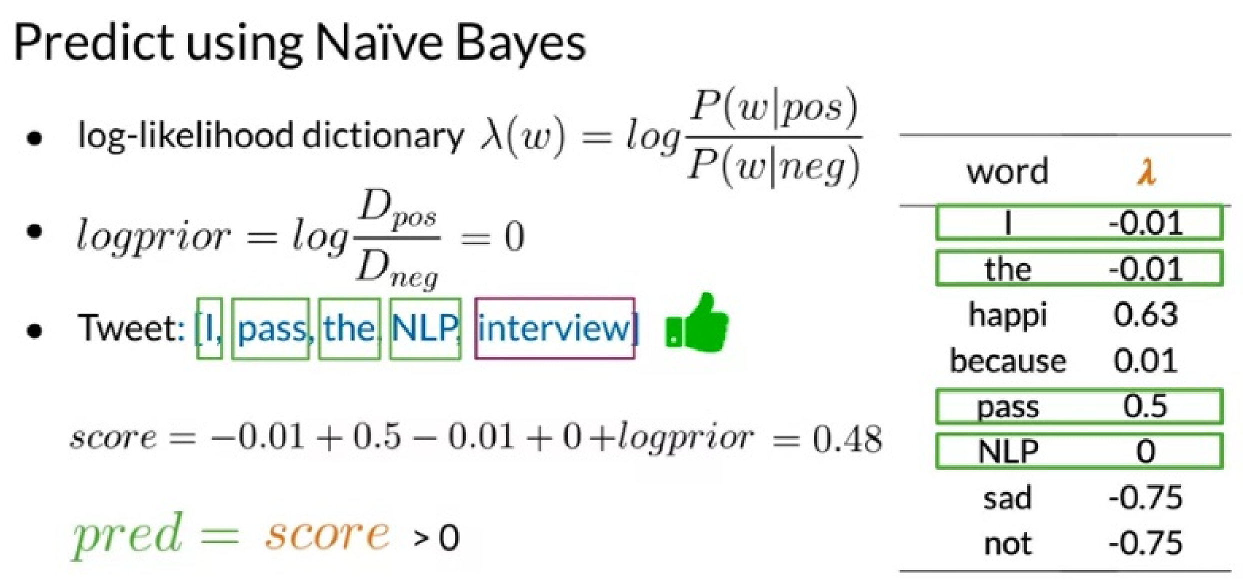
Final prediction



Summary

0. Get or annotate a dataset with positive and negative tweets
1. Preprocess the tweets: process_tweet(tweet) → [w_1, w_2, w_3, ...]
2. Compute freq(w, class)
3. Get P(w | pos), P(w | neg)
4. Get λ(w)
5. Compute logprior = log(P(pos) / P(neg))

Prediction



Predict using Naïve Bayes

- log-likelihood dictionary $\lambda(w) = log\frac{P(w|pos)}{P(w|neg)}$
- $logprior = log\frac{D_{pos}}{D_{neg}} = 0$
- Tweet: [I, pass, the, NLP, interview]

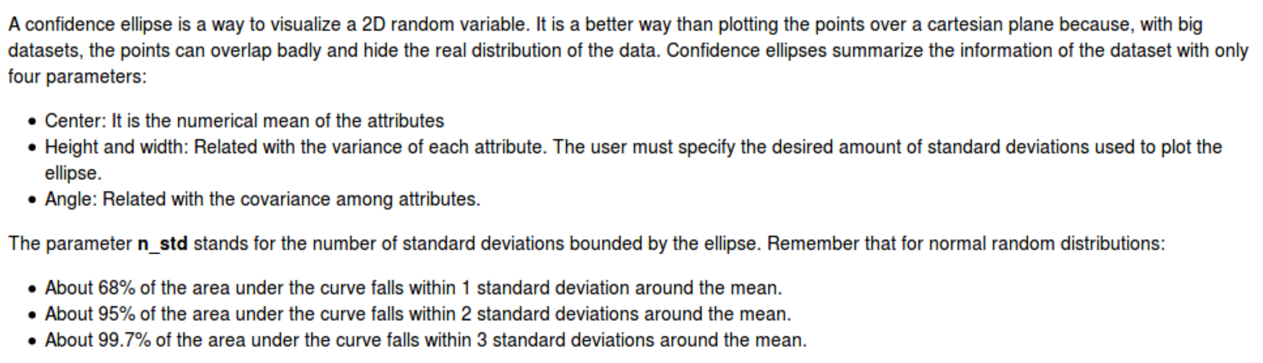
$$score = -0.01 + 0.5 - 0.01 + 0 + logprior = 0.48$$

$$pred = score > 0$$

word	λ
I	-0.01
the	-0.01
happi	0.63
because	0.01
pass	0.5
NLP	0
sad	-0.75
not	-0.75

Prediction example

Refer **Jupyter NB** for visualization of NB (Confidence Ellipse Method — Visualization of Distribution)

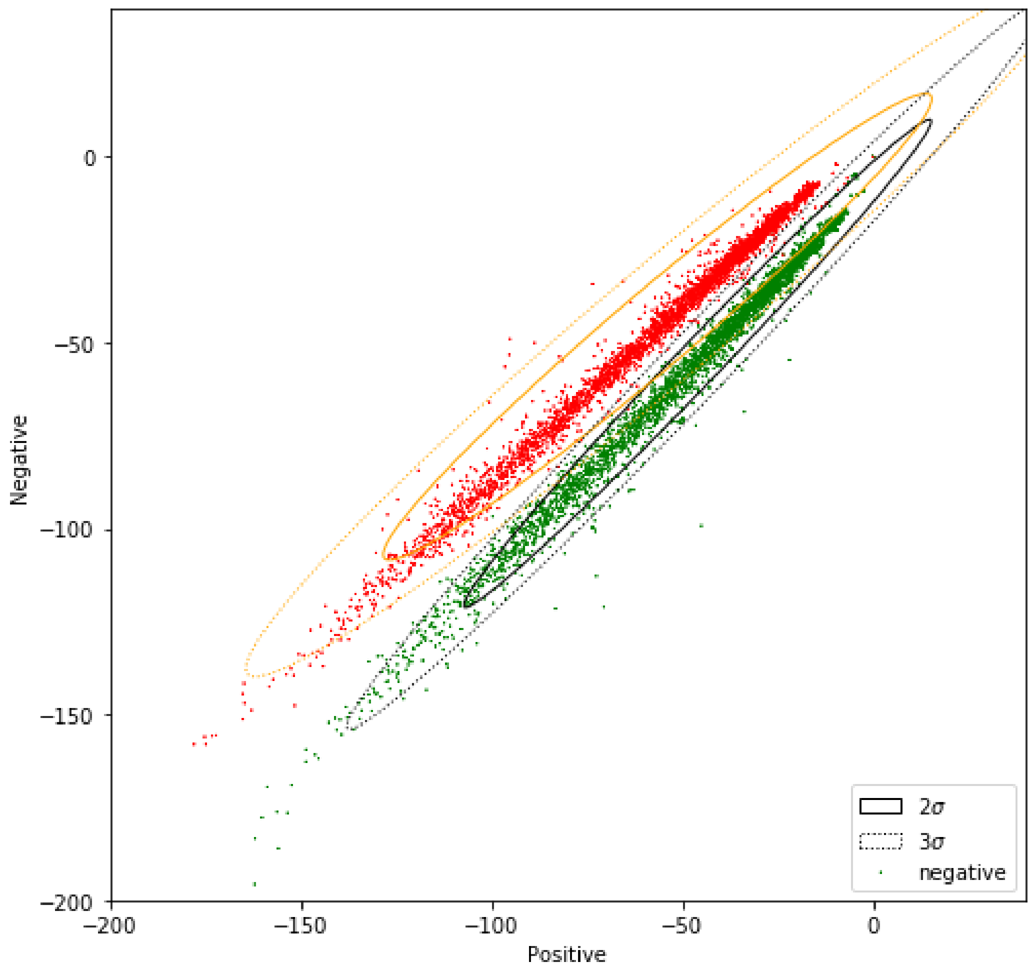


A confidence ellipse is a way to visualize a 2D random variable. It is a better way than plotting the points over a cartesian plane because, with big datasets, the points can overlap badly and hide the real distribution of the data. Confidence ellipses summarize the information of the dataset with only four parameters:

- Center: It is the numerical mean of the attributes
- Height and width: Related with the variance of each attribute. The user must specify the desired amount of standard deviations used to plot the ellipse.
- Angle: Related with the covariance among attributes.

The parameter **n_std** stands for the number of standard deviations bounded by the ellipse. Remember that for normal random distributions:

- About 68% of the area under the curve falls within 1 standard deviation around the mean.
- About 95% of the area under the curve falls within 2 standard deviations around the mean.
- About 99.7% of the area under the curve falls within 3 standard deviations around the mean.



Confidence Ellipse

Application of NB

- Information retrieval
- Word Disambiguation
- Text Classification (Sentiment, Spam, Author Identification)

Assumption of NB

- **Independence of words in a sentence** — It is sunny and hot in Sahara dessert. Sunny and hot usually appear together. If together taken, it might describe beach or dessert but if taken independent which is case in NB, very difficult to describe. **Overestimate** the conditional prob. of individual words "It's always cold and snowy in ______" spring ?? summer ?? fall ?? **winter**
- Relative frequencies in corpus i.e. P(class) might not align with real world data

Source of Error

- **Preprocessing** (Punctuation, stopword)
- **Word order** : S1 — I am happy because I did not go. (positive) S2 — I am not happy because I did go. (negative) … Order of not matters a lot here ..
- **Adversarial attacks:** Sarcasm, Irony, Euphemisms