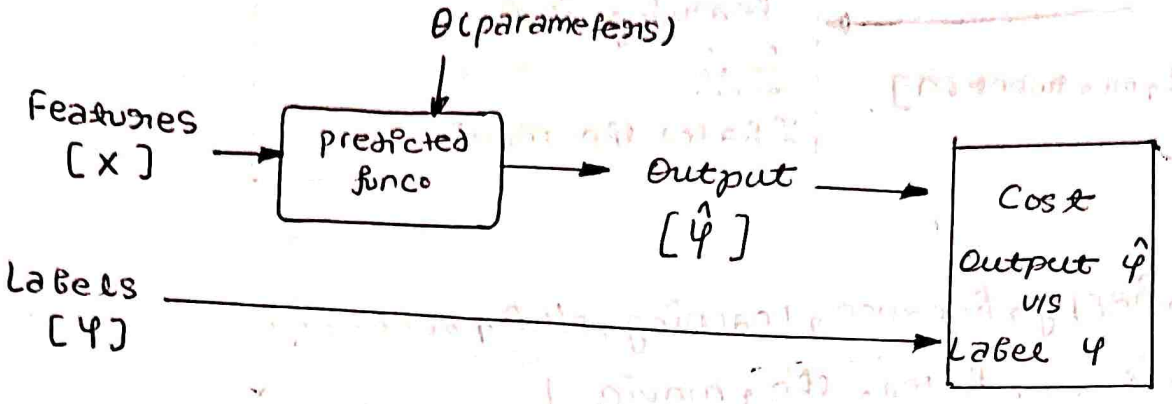
Logistic Regression for sentiment Analysis of tweets:

In supervised ML,

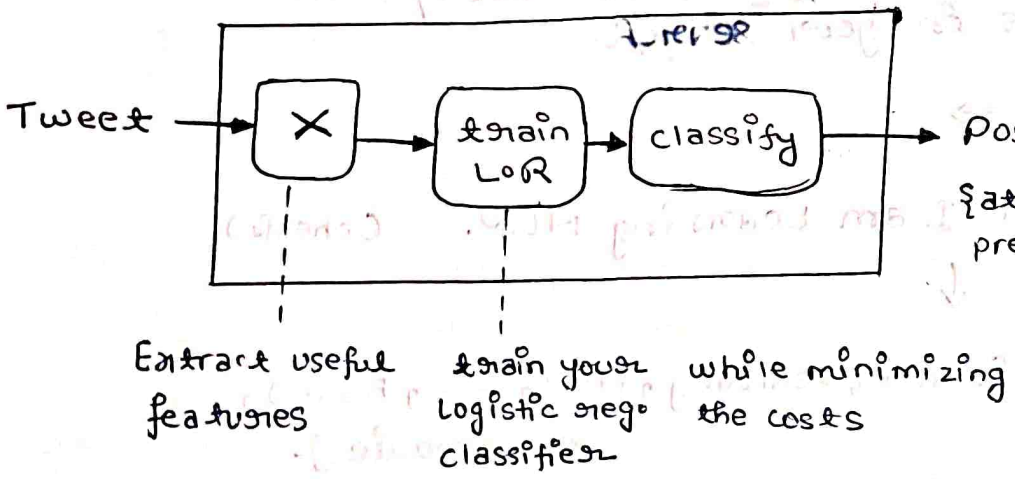


i.e. Whether the tweet have a positive or negative sentiment?

Tweet: I am happy because I am learning NLP

→ positive sentiment: 1

→ Negative sentiment: 0



Step 1: Vocabulary & feature extraction:

How to represent a text as a vector →

first build a vocabulary, that will allow you to encode any text or any tweet as an array of numbers.

∘ Vocabulary

Tweets → [tweet1, tweet2, ..., tweetm]

I am happy because I am learning NLP.
.....
I hated the movie.

V = [I, am, happy, because, learning, NLP, ... hated, the, movie]

- Vocabulary (V) is the list of unique words from your list of tweets.
- To get that list you'll have to go through all the words from ~~the list~~ all your tweets every new word that appears in your search.

∘ Feature Extraction →

I am happy because I am learning NLP. (check)

↓

V = [I, am, happy, because, learning, NLP, ---, hated, the, movie].

↓

- To do so, you will have to check every word from your vocabulary appears in the tweet. you will assign a value of (1) to that feature. If it doesn't appear you assign a value (0).

↓

[I, am, happy, because, learning, NLP, ..., hated, the, movie].

[1, 1, 1, 1, 1, 1, ..., 0, 0, 0].

6 count of 1's, many 0's.

Problems with "Sparse" Representations —

- A lot of zeros → that's a sparse representation.

  i.e. I am happy because I am learning NLP

  ↓

  [1, 1, 1, 1, 1, 1, ..., 0, 0, 0].

- This representation would have a number of features equal to the size of your entire vocabulary
- This would have a lot of features equal to 0 for every tweet
- With the sparse representation, a logistic regression model would have to learn (n+1) parameters, where n = size of your vocabulary.

$n = |V|$ → $[\theta_0, \theta_1, \theta_2, ..., \theta_n]$

Hence, ① Large training time
② Large prediction time.

Positive and Negative frequencies —

Corpus

I am happy because I am learning NLP.
I am happy
I am sad, I am not learning NLP.
I am sad.

Vocabulary

I
am
happy
because
learning
NLP
sad
not

Positive tweets

I am happy because I am learning NLP
I am happy

Negative tweets

I am sad, I am not learning NLP.
I am sad.

Vocabulary	PosFreq(1)	NegFreq(0)
I	3	3
am	3	3
happy	2	0
because	1	0
learning	1	1
NLP	1	1
sad	0	2
not	0	1

- previously we encoded a ~~vector~~ tweet as a vector of dimension (V). Now, we'll learn to encode a tweet or specially represented as a vector of dimension 3.

freqs : dictionary mapping from (word, class) to frequency

$$X_m = [1, \sum_w freqs(w, 1), \sum_w freqs(w, 0)]$$

- X_m : arbitrary tweet m or features of tweet m
- 1 : Bias
- $\sum_w freqs(w, 1)$: sum of pos freqs
- $\sum_w freqs(w, 0)$: sum of neg freqs

(for ~~above~~ example)

Problems with "Sparse" Representations —

- A lot of zeros → that's a sparse representation.

  ie I am happy Because I am Learning NLP

  ↓

  [1, 1, 1, 1, 1, 1, ..., 0, 0, 0].

- This representation would have a number of features equal to the size of your entire vocabulary
- This would have a lot of features equal to 0 for every tweet
- With the sparse representation, a logistic regression model would have to learn (n+1) parameters, where n = size of your vocabulary.

$n = |V|$ → $[\theta_0, \theta_1, \theta_2, ..., \theta_n]$

Hence, ① Large training time
② Large prediction time.

Positive and Negative frequencies —

Corpus

I am happy Because I am Learning NLP.
I am happy
I am sad, I am not Learning NLP.
I am sad.

Vocabulary

I
am
happy
Because
learning
NLP
sad
not

Eg: I am sad, I am not learning NLP.

↓

Vocab ≡ I am ~~happy~~ ~~because~~ learning NLP sad not

(posfreq) 3 + 3 + ~~0~~ + ~~0~~ + 1 + 1 + 0 + 0 ≡ 8

Vocab : I am ~~happy~~ ~~because~~ learning NLP sad not

Negfreq : 3 + 3 + ~~0~~ + ~~0~~ + 1 + 1 + 2 + 1 ≡ 11

$X_m \equiv [1, 8, 11]$.

Putting it all together →

I am Happy Because i am Learning NLP @deeplearning

↓ [preprocessing]

[happy, learn, nlp].

↓ [feature extraction]

bias ← [1, 4, 2]

↑ posfreq ↑ Negfreq

⇓

I am Happy Because i am learning NLP @deeplearning

I am sad not learning NLP

.....

I am sad :(

→ [happy, learn, nlp]

[sad, not, learn, nlp]

.....

[sad]

→ [[1, 40, 20],

[1, 20, 50],

....

[1, 5, 35]].

$$\begin{bmatrix} 1 & X_1^{(1)} & X_2^{(1)} \\ 1 & X_1^{(2)} & X_2^{(2)} \\ \vdots & \vdots & \vdots \\ 1 & X_1^{(m)} & X_2^{(m)} \end{bmatrix}$$

⟷

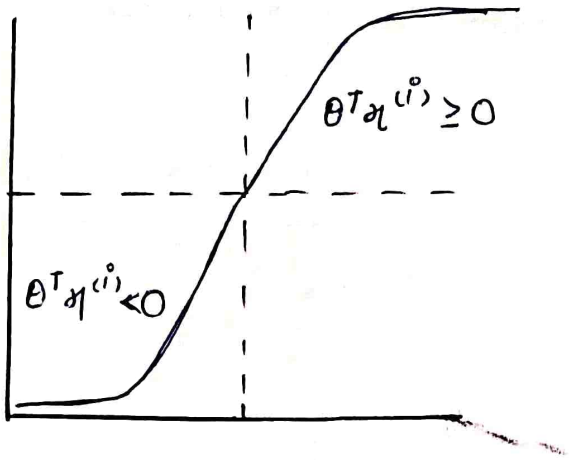
[[1, 40, 20],

[1, 20, 50],

....

[1, 5, 35]].

$$h(x^{(i)}, \theta) = \frac{1}{1+e^{-\theta^T x^{(i)}}}$$



@YMoussi and
@AndrewYNg are tuning a GREAT AI model

↓

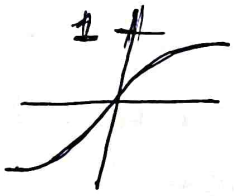
[tun, ai, great, model]

↓

$$x^{(i)} = \begin{bmatrix} 1 \\ 3476 \\ 245 \end{bmatrix}$$

Sum of all pos & neg freqs of all the words in your processed ~~tweets~~

↓

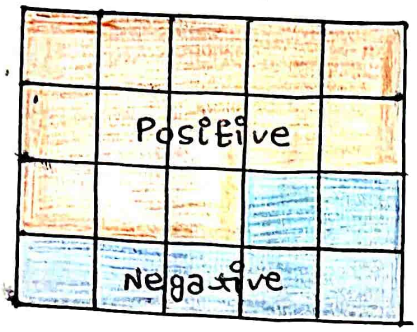


$$\theta = \begin{bmatrix} 0.00003 \\ 0.00150 \\ -0.00120 \end{bmatrix}$$

Outline

- Probabilities
- Baye's Rule (Applied in diff. field, including NLP)
- Build your own Naive Bayes Tweet Classifier

Corpus of Tweets



Tweets containing the word "happy".

A → Positive tweet

$P(A) = N_{pos} / N \rightarrow 13/20$

$\rightarrow 0.65$

$P(Negative) = 1 - P(positive)$

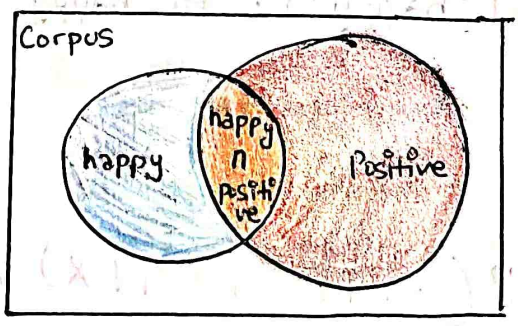
$\rightarrow 1 - 0.65 \rightarrow 0.35$

B → tweet containing "happy"

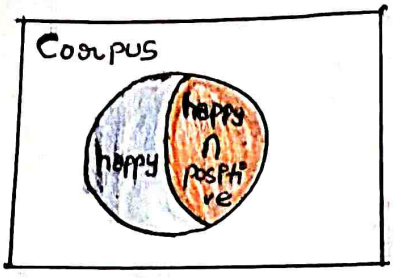
$P(B) = P(happy) = N_{happy} / N$

$P(B) = 4/20 \rightarrow 0.2$

Probability of the intersection



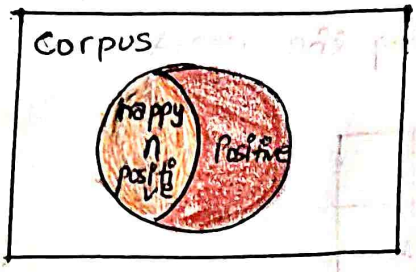
$$P(A \cap B) = P(A, B) = \frac{3}{20} = 0.15$$



$$P(A|B) = P(\text{positive} \mid \text{"happy"})$$

$$= \frac{3}{4} \Rightarrow 0.75$$

prob of tweet is positive given that it contains word happy



$$P(B|A) = P(\text{"happy"} \mid \text{positive})$$

$$= \frac{3}{13} \Rightarrow 0.231$$

Baye's Rule →

$$P(\text{Positive} \mid \text{"happy"}) = \frac{P(\text{Positive} \cap \text{"happy"})}{P(\text{"happy"})}$$

$$P(\text{"happy"} \mid \text{Positive}) = \frac{P(\text{"happy"} \cap \text{Positive})}{P(\text{Positive})}$$

↓

$$P(\text{Positive} \mid \text{"happy"}) = P(\text{"happy"} \mid \text{Positive}) \times \frac{P(\text{Positive})}{P(\text{"happy"})}$$

↓

$$P(x|y) = P(y|x) \times \frac{P(x)}{P(y)}$$

Naive Bayes for Sentiment Analysis —

Positive tweets

I am happy because I am learning NLP.
I am happy, not sad.

Negative tweets.

I am sad, I am not learning NLP.
I am sad, not happy

word
I
am
happy
because
learning
NLP
sad
not

freq

word	Pos	neg
I	3	3
am	3	3
happy	2	1
because	1	0
learning	1	1
NLP	1	1
sad	1	2
not	1	2
Nclass	13	12

↓
helps in computing the conditional probabilities of word in the class.

word	pos	neg
I	$P\left(\frac{I}{pos}\right) = \frac{3}{13} = 0.24$	$P\left(\frac{I}{neg}\right) = \frac{3}{12} = 0.25$
am	$P\left(\frac{am}{pos}\right) = \frac{3}{13} = 0.24$	$P\left(\frac{am}{neg}\right) = \frac{3}{12} = 0.25$
happy	0.15	0.08
because	0.08	0.01
learning	0.08	0.08
NLP	0.08	0.08
sad	0.08	0.17
not	0.08	0.17
Σ	1	1

⇓

$P(W_i | class)$

Eg: predict sentiment of whole tweet

Tweet: I am happy today ; I am Learning

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

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

$$\equiv \frac{0.14}{0.10} \Rightarrow 1.4 > 1$$

(positive sentiment)

a nearly identical conditional probability, words that are equally probable & don't add anything to the sentiment

These are your power words, These carry a lot of weights in determining in your tweet sentiments

conditional probability for the negative class is zero. when this happens no way of comparing b/w positive tweets & negative tweets corpora.

which will become a problem for your calculations

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

Naive Baye's inference conditional-rule for Binary classification

This expression says you are going to take the product across all of the words in your tweets of the probability for each word in positive class divide it by the prob. in neg. class

Naive Bayes Classifier is a supervised learning algorithm based on Baye's theorem.

It's a probabilistic classifier, which mean it predict on the basis of probability of an object.

Bayes theorem →

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \quad \text{—— (1)} \qquad P(B|A) = \frac{P(A \cap B)}{P(A)} \quad \text{—— (2)}$$

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

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

or

$$P(y|X) = \frac{P(X|y)\,P(y)}{P(X)}$$

⇒ Bayes Theorem

where y is class variable and X is dependent feature ~~variable~~ vector

↳ $(x_1, x_2, x_3, \ldots, x_n)$

$$P(y|x_1, x_2, \ldots, x_n) = \frac{P(y) * P(x_1, x_2, x_3, \ldots, x_n|y)}{P(x_1, x_2, x_3, \ldots, x_n)}$$

$$\rightarrow \frac{P(y) * P(x_1|y_1) * P(x_2|y_2) * \ldots * P(x_n|y_n)}{P(x_1) * P(x_2) * P(x_3) * \ldots * P(x_n)}.$$

y = yes or no / 0 or 1 / T or F.

Day	Outlook	Temp	Humidity	Wind	Play
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	yes
4	Rain	Mild	High	Weak	yes
5	Rain	Cool	Normal	Weak	yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	~~No~~ yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	yes
10	Rain	Mild	Normal	Strong	yes
11	Sunny	Mild	Normal	Strong	yes
12	Overcast	Mild	High	Strong	yes
13	Overcast	Hot	Normal	Weak	yes
14	Rain	Mild	High	Strong	No
15	Sunny	Cool	High	Strong	?

Day (15) $P(yes \mid sunny, cool, high, strong)$

↑ class variable

$sunny, cool, high, strong$ ↓ dependent feature vector

$$\Rightarrow \frac{P(yes) * P(sunny, cool, high, strong \mid yes)}{P(sunny, cool, high, strong)}$$

$$\Rightarrow \frac{P(yes) * P(sunny \mid yes) * P(cool \mid yes, sunny) * P(high \mid yes, sunny, cool) * P(strong \mid yes, sunny, cool, high)}{P(sunny, cool, high, strong)}$$

$$\rightarrow \frac{9}{14} * \frac{2/14}{9/14} * \frac{3}{9} * \frac{3}{9} * \frac{4}{9} \Rightarrow \frac{9 * 2 * 3 * 3 * 4}{14 * 9 * 9 * 9 * 9}$$

$P(high \mid yes)$ $P(strong \mid yes)$

$\frac{P(cool \cap yes \cap sunny)}{P(yes \cap sunny)} \rightarrow P(cool \mid yes)$

$\rightarrow 0.007$

$P(No \mid Sunny, Cool, High, Strong)$

- No — class variable
- $Sunny, Cool, High, Strong$ — dependent feature vector

$\Rightarrow P(No) * P(Sunny \mid No) * P(Cool \mid No, Sunny) * P(High \mid \cancel{Yes}\ No, Sunny, Cool) * P(Strong \mid No, Sunny, Cool, High)$

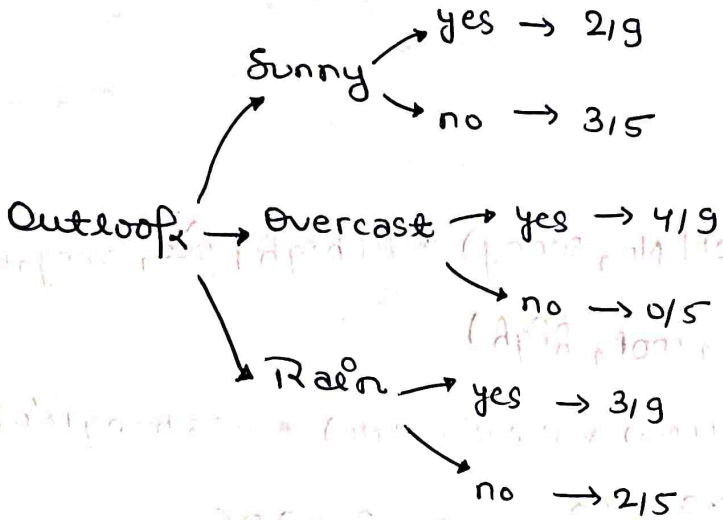
$P(No) * P(Sunny \mid No) * P(Cool \mid No) * P(High \mid No) * P(Strong \mid No)$

$\Rightarrow \frac{5}{14} * \frac{3}{5} * \frac{1}{5} * \frac{4}{5} * \frac{3}{5} \rightarrow \frac{5*3*4*3}{14*5*5*5*5} \rightarrow 0.0206$

Wind
- Strong
  - yes: $P(Strong \mid yes) = 4/9$ → $\frac{P(Strong \cap yes)}{P(yes)}$
  - no: $P(Strong \mid no) = 3/5$ → $\frac{P(Strong \cap no)}{P(no)}$
- Weak
  - yes: $P(weak \mid yes) = 5/9$
  - no: $P(weak \mid no) = 2/5$

Humidity
- High
  - yes: $P(High \mid yes) = \frac{P(High \cap yes)}{P(yes)} \rightarrow 3/9$
  - No: $P(High \mid No) = \frac{P(High \cap No)}{P(No)} \rightarrow 4/5$
- Normal
  - yes: $P(Normal \mid yes) = \frac{P(Normal \cap yes)}{P(yes)} \rightarrow 6/9$
  - No: $P(Normal \mid No) = \frac{P(Normal \cap No)}{P(No)} \rightarrow 1/5$

Temp
- Hot
  - yes → 2/9
  - no → 2/5
- Mild
  - yes → 4/9
  - no → 2/5
- Cold
  - yes → 3/9
  - no → 1/5


Outlook
Sunny
yes → 2/9
no → 3/5
Overcast → yes → 4/9
no → 0/5
Rain → yes → 3/9
no → 2/5

Laplacian Smoothing :

Sometimes you try to calculate the probability of a word happening after a word. To do that you might want to count the nos of times those two words showed up. One after another, divided by the number of times the first word appears. Now what if the ~~two~~ two words never showed up next to each other in the training corpus. You, get a probability of zero, and the probability of an entire sequence might go to zero.

Now, how can we fix this?

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

$class \in \{positive, negative\}$

N_{class} = frequency of all words in class.

V_{class} = nos of unique words in class.

avoids the prob being zero.

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

[Laplacian Smoothing].

word	pos	neg
I	3	3
am	3	3
happy	2	1
Because	1	0
Learning	1	1
NLP	1	1
sad	1	2
not	1	2
Nclass	13	12

→ V = 8

word	pos	neg
I	$P(I\|pos) \Rightarrow \frac{3+1}{13+8} \Rightarrow \boxed{0.19}$	$P(I\|neg) \Rightarrow \frac{3+1}{12+8} \Rightarrow \boxed{0.20}$
am	0.19	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
Σ	1	1

[Laplacian Smoothing]

- Laplacian smoothing to avoid $\prod_{i=1}^{m} \frac{P(w_i|pos)}{P(w_i|neg)}$

$P(w_i|class)$

0

$$\frac{0.19}{0.20} * \frac{0.19}{0.20} * \frac{0.14}{0.10} * \frac{0.10}{0.05} * \frac{0.10}{0.10} * \frac{0.10}{0.10} * \frac{0.10}{0.15} * \frac{0.10}{0.15}$$

$$\Rightarrow \frac{0.000005054}{0.0000045} \Rightarrow 1.12 > 1$$

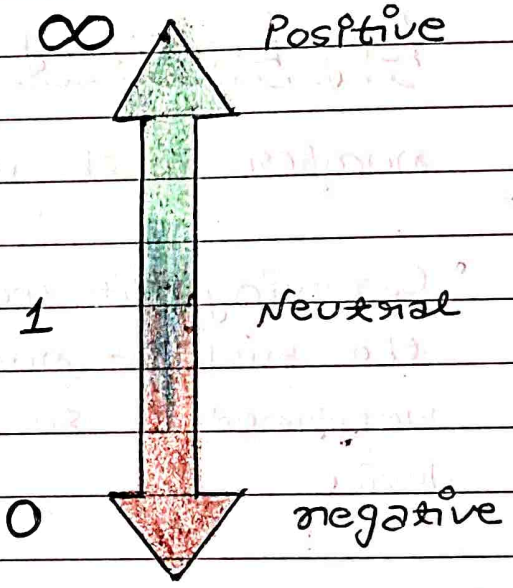
Log Likelihoods

Words can have many shades of emotional meaning, but for the purpose of sentiment classification, they are simplified into three categories : neutral, positive & negative

All can be identified by using their conditional probabilities →

$$\text{ratio}(w_i) = \frac{P(w_i \mid Pos)}{P(w_i \mid Neg)}$$

Word	Pos	neg	ratio
I	0.20	0.20	1
am	0.20	0.20	1
happy	0.14	0.10	1.4
because	0.10	0.10	1
learning	0.10	0.10	1
NLP	0.10	0.10	1
sad	0.10	0.15	0.6
not	0.10	0.15	0.6



- The larger the ratio the more positive the word is going to be.
- Now, on the other hands, negative words have a ratio smaller than 1.

$$\simeq \frac{freq(w_i, 1) + 1}{freq(w_i, 0) + 1}$$

Naive Bayes Inference

Class $\in$ {pos, neg}

w → set of m words in a tweet

$$\underbrace{\prod_{i=1}^{m} \frac{P(w_i|Pos)}{P(w_i|Neg)}}_{\text{likelihood}} > 1 \qquad \underbrace{\frac{P(pos)}{P(neg)}}_{\text{prior ratio}}$$

- A simple, fast and powerful baseline.
- A probabilistic method used for classification

Log Likelihood → • Sentiments probability calculations requires multiplication of many numbers with values between 0 an 1.

- Carrying out such multiplications on a computer run the risk of numerical underflow when the number returned is so small if it can't be stored on your device
- mathematical trick to solve this —

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

- $$\log\left(\frac{P(pos)}{P(neg)} \prod_{i=1}^{n} \frac{P(w_i|Pos)}{P(w_i|Neg)}\right)$$

log (prior multiplied by likelihood)

$$\Rightarrow \log(\text{prior}) + \log(\text{likelihood})$$

or

$$\log \frac{P(pos)}{P(neg)} + \sum_{i=1}^{n} \log \frac{P(w_i|pos)}{P(w_i|neg)}$$

Calculating Lambda —

word	pos	neg	λ
I	0.05	0.05	0
am	0.04	0.04	0
happy	0.09	0.01	2.2
because	0.01	0.01	0
learning	0.03	0.01	1.1
NLP	0.02	0.02	0
sad	0.01	0.09	-2.2
not	0.02	0.03	-0.4

$$\lambda(I) = \log \frac{0.05}{0.05} \Rightarrow 0$$

$$\lambda(am) = \log \frac{0.04}{0.04} \Rightarrow 0$$

$$\lambda(happy) = \log \frac{0.09}{0.01} \rightarrow \log 9 \rightarrow 2.2$$

$$\lambda(w) = \log \frac{P(w|pos)}{P(w|neg)}$$

- word sentiment
  - $ratio(w) = \frac{P(w|pos)}{P(w|neg)}$
  - $\lambda(w) = \log \frac{P(w|pos)}{P(w|neg)}$

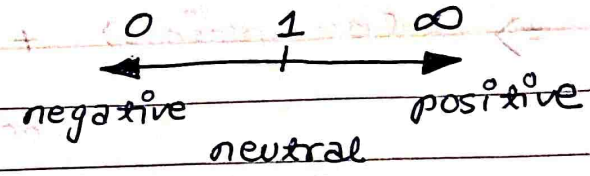
I am happy because I am learning

↓ ↓ ↓ ↓ ↓ ↓ ↓

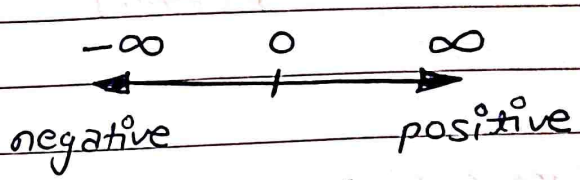
log likelihood → 0 + 0 + 2.2 + 0 + 0 + 0 + 1.1

⇒ 3.3

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



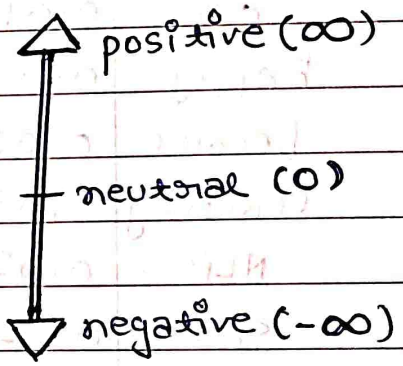
$$\sum_{i=1}^{m} \log \frac{P(w_i|pos)}{P(w_i|neg)} > 0$$



$3 \cdot 3 > 0$

Tweet sentiment,

$$\log \prod_{i=1}^{m} ratio(w_i) = \sum_{i=1}^{m} \lambda(w_i) > 0$$



Step5: Get the log prior

- Trainning Naive Bayes

Step 0: Collect and Annotate Corpus

positive tweets

I am happy because I am learning NLP

I am happy, not sad. @ NLP

Negative tweets

I am sad, I am not learning NLP

I am sad, not happy !!

Step 1: Data - Preprocessing

- Lowercase
- Remove punctuation, urls, names
- Remove Stopwords
- Stemming
- Tokenize sentences

positive tweets

[happi, because, learn, NLP]

[happi, not, sad]

Negative tweets

[sad, not, learn, NLP]

[sad, not, happi].

Step 2: word - count

Once, you have a clean corpus of processed tweets, you will start by computing the vocabulary for each word in class,

freq(w, class)

word	Pos	Neg
happi	2	1
because	1	0
learn	1	1
NLP	1	1
sad	1	2
not	1	2
N_{class}	7	7

Step 3: $P(w|class)$

$V_{class} = 6$

$$\frac{freq(w, class) + 1}{N_{class} + V_{class}}$$

word	Pos	Neg
happi	0.23	0.15
because	0.15	0.07
learn	0.08	0.08
NLP	0.08	0.08
sad	0.08	0.17
not	0.08	0.17

Step 4:
Get Lambda(λ)

$$\lambda(w) = \log \frac{P(w|pos)}{P(w|neg)}$$

word	pos	neg	λ
happi	0.23	0.15	0.43
because	0.15	0.07	0.6
learn	0.08	0.08	0
NLP	0.08	0.08	0
sad	0.08	0.17	-0.75
not	0.08	0.17	-0.75

Step5: Get the log prior

D_{pos} = Nos of positive tweets

D_{neg} = Nos of negative tweets

$$\text{log prior} = \log \frac{D_{pos}}{D_{neg}}$$

If dataset is balanced, $D_{pos} = D_{neg}$ and log prior = 0

Testing Naive Bayes →

- Predict using a Naive Bayes Model
- Using your validation set to compute model accuracy

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

- log-likelihood dictionary

$$\lambda(w) = \log \frac{P(w|pos)}{P(w|neg)}$$

- $\log prior = \log \frac{D_{pos}}{D_{neg}} = 0$

- Tweet : (check)

[I, pass, the, NLP, interview]

↓ ↓ ↓ ↓ ↓

score : $= -0.01 + 0.5 - 0.01 + 0 + 0 + \log prior$

$\Rightarrow 0.48$

pred = score > 0 (positive)

- Xval Yval λ logprior

score = predict (Xval, λ, logprior)

pred = score > 0

$$\begin{bmatrix} 0.5 \\ -1 \\ 1.3 \\ \vdots \\ score_m \end{bmatrix} > 0 \Rightarrow \begin{bmatrix} 0.5>0 \\ -1>0 \\ 1.3>0 \\ \vdots \\ score_m > 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \\ pred_m \end{bmatrix}$$

produces a vector populated with 0's & 1's indicating if the predicted sentiment is negative or positive.

$$\begin{bmatrix} 0 \\ 1 \\ 1 \\ \vdots \\ pred_m \end{bmatrix} == \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ y_{val_m} \end{bmatrix}$$

0 - 0 → 1
1 - 0 → 0
1 - 1 → 1

$$\frac{1}{m}\sum_{i=1}^{m}(pred_i == y_{val_i}) \Rightarrow \begin{bmatrix} 1 \\ 0 \\ 1 \\ \\ pred_m == y_{val_m} \end{bmatrix}$$

Summary • X_{val} y_{val} → performance on unseen data

- predict using λ and logprior for each new tweet
- Accuracy → $\frac{1}{m}\sum_{i=1}^{m}(pred_i == y_{val_i})$
- What about words that don't appear in $\lambda(w)$?

Applications of Naive Bayes :

① Twitter Sentiment Analysis

$P(pos \mid tweet) = P(pos)\, P(tweet \mid pos)$

$P(neg \mid tweet) = P(neg)\, P(tweet \mid neg)$

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

priors — likelihood

estimating the probability for each class by using the joint probability of the words in classes

(2) Identification system

i.e Author Identification : $\frac{P(\text{"shakespeare"} \mid \text{book})}{P(\text{"meadly"} \mid \text{book})}$

If you have two large corporal, each written by two different authors, you could train a model to recognize whether a new document was written by one or another

(3) Word disambiguation : Consider that you have only two possible interpretations of a given word within a text.

Let, say you don't know if the word bank in your reading is reffering to the bank of a river or to a financial institution

To disambiguate your word calculate the score of documents, given that it reffers to each one of the possible meanings.

In this case, if the text refers to the concept of river instead of the concept of money, then the score will be bigger than one.

$\frac{P(\text{river} \mid \text{text})}{P(\text{money} \mid \text{text})}$

Decision Trees!

Day	Outlook	Temp	Humidity	Wind	play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	yes
D4	Rain	Mild	High	Weak	yes
D5	Rain	Cool	Normal	Weak	yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	yes
D10	Rain	Mild	Normal	Weak	yes
D11	Sunny	Mild	Normal	Strong	yes
D12	Overcast	Mild	High	strong	yes
D13	Overcast	Hot	Normal	Weak	yes
D14	Rain	Mild	High	Strong	No

Entropy measures homogeneity of examples,

- Entropy measures the impurity of a collection of examples It depends from the distribution of the random variable p.

→ S is a collection of trainning examples
→ $p_{\oplus}$ the proportion of positive examples in S.
→ $p_{\ominus}$ the proportion of negative examples in S.

- Entropy is the measure of disorder or impurity in a node

$$E = -\sum_{i=1}^{n} p_i \log_2 p_i$$

- node with a more variable composition considered to be more entropy than node with less variable composition.
- Entropy of universal fact = 0;

High Entropy → less homogenous / more impurity
less Entropy → more homogenous / less impurity

Eg Attribute : Outlook
Values (Outlook) = Sunny, Overcast, Rain

$S = [9+, 5-]$ entropy $(S) = \frac{-9}{14} \log_2^{9/14} \frac{-5}{14} \log_2^{5/14} \Rightarrow 0.94$

$S_{sunny} \leftarrow [2+, 3-]$ entropy $_{(sunny)}(S) = \frac{-2}{5} \log_2^{2/5} - \frac{3}{5} \log_2^{3/5} \Rightarrow 0.971$

$S_{overcast} \leftarrow [4+, 0-]$ Entropy $_{(S_{overcast})} = \frac{-4}{4} \log_2^{4/4} - \frac{0}{4} \log_2^{0/4} \Rightarrow 0$

$S_{rain} \leftarrow [3+, 2-]$ Entropy $_{(S_{rain})} = \frac{-3}{5} \log_2^{3/5} - \frac{-2}{5} \log_2^{2/5} \Rightarrow 0.971$

- Information gain, measure of change in Entropy

$$Gain(S, outlook) = Entropy(S) - \sum_{v \in \{sunny, overcast, rain\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S, outlook) = Entropy(S) - \frac{5}{14} Entropy(S_{sunny}) - \frac{4}{14} Entropy(S_{overcast}) - \frac{5}{14} Entropy(S_{rain})$$

$$\Rightarrow 0.94 - \frac{5}{14}(0.971) - \frac{4}{14}(0) - \frac{5}{14}(0.971) \Rightarrow 0.7464$$

Attribute : Temp

Values(Temp) : Hot, Mild, Cold

$S = [9+, 5-]$ $\quad Entropy(S) \rightarrow -\frac{9}{14}\log_2^{9/14} - \frac{5}{14}\log_2^{5/14} \Rightarrow 0.94$

$S_{hot} \leftarrow [2+, 2-]$ $\quad Entropy(S_{hot}) \rightarrow -\frac{2}{4}\log_2^{2/4} - \frac{2}{4}\log_2^{2/4} \Rightarrow 1.00$

$S_{mild} \leftarrow [4+, 2-]$ $\quad Entropy(S_{mild}) \rightarrow -\frac{4}{6}\log_2^{4/6} - \frac{2}{6}\log_2^{2/6} \Rightarrow 0.9183$

$S_{cool} \leftarrow [3+, 1-]$ $\quad Entropy(S_{cool}) \rightarrow -\frac{3}{4}\log_2^{3/4} - \frac{1}{4}\log_2^{1/4} \Rightarrow 0.8132$

$$Gain(S, Temp) \Rightarrow Entropy(S) - \sum_{v \in (Hot, mild, cold)} \frac{|S_v|}{|S|} Entropy(S_v).$$

Gain (S, Temp) ⇒ Entropy (S) − $\frac{4}{15}$ Entropy (Shot) − $\frac{6}{14}$ Entropy(Smild)

− $\frac{4}{14}$ Entropy (Scool)

⇒ 0.94 − $\frac{4}{14}$ (1.0) − $\frac{6}{14}$ (0.9183) − $\frac{4}{14}$ (0.8113) ⇒ 0.0289

Attribute : Humidity

Values (Humidity) = High, Normal

S = [9+, 5−] Entropy(S) = $-\frac{9}{14}\log_2^{9/14} - \frac{5}{14}\log_2^{5/14}$ ⇒ 0.94

S_{high} ← [3+, 4−] Entropy (Shigh) = $\frac{-3}{7}\log_2^{3/7} - \frac{4}{7}\log_2^{4/7}$ = 0.9852

S_{normal} ← [6+, 1−] Entropy (Snormal) = $\frac{-6}{7}\log_2^{6/7} - \frac{1}{7}\log_2^{1/7}$ ⇒ 0.5916

Gain (S, Humidity) = Entropy(S) − $\sum_{v \in \{high, normal\}} \frac{|S_v|}{|S|}$ Entropy(S_v)

Gain (S, Humidity) ⇒ 0.94 − $\frac{7}{14}$ (0.9852) − $\frac{7}{14}$ (0.5916) = 0.1516

Attribute : Wind

Values (Wind) = Strong, Weak

S = [9+, 5-] Entropy(S) => 0.94

S_{Strong} ← [3+, 3-] Entropy (Sstrong) = 1.0

Sweak ← [6+, 2-] Entropy (Sweak) $\Rightarrow -\frac{6}{8}\log_2^{6/8} - \frac{2}{8}\log_2^{2/8} \Rightarrow 0.0113$

$$Gain(S, wind) = Entropy(S) - \sum_{V \in \{strong, weak\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S, wind) = Entropy(S) - \frac{6}{14} Entropy(Sstrong) - \frac{8}{14} Entropy(Sweak).$$

$$Gain(S, wind) = 0.94 - \frac{6}{14}(1.0) - \frac{8}{14}(0.8113) \Rightarrow 0.0478$$

Gain (S, Outlook) → 0.2464

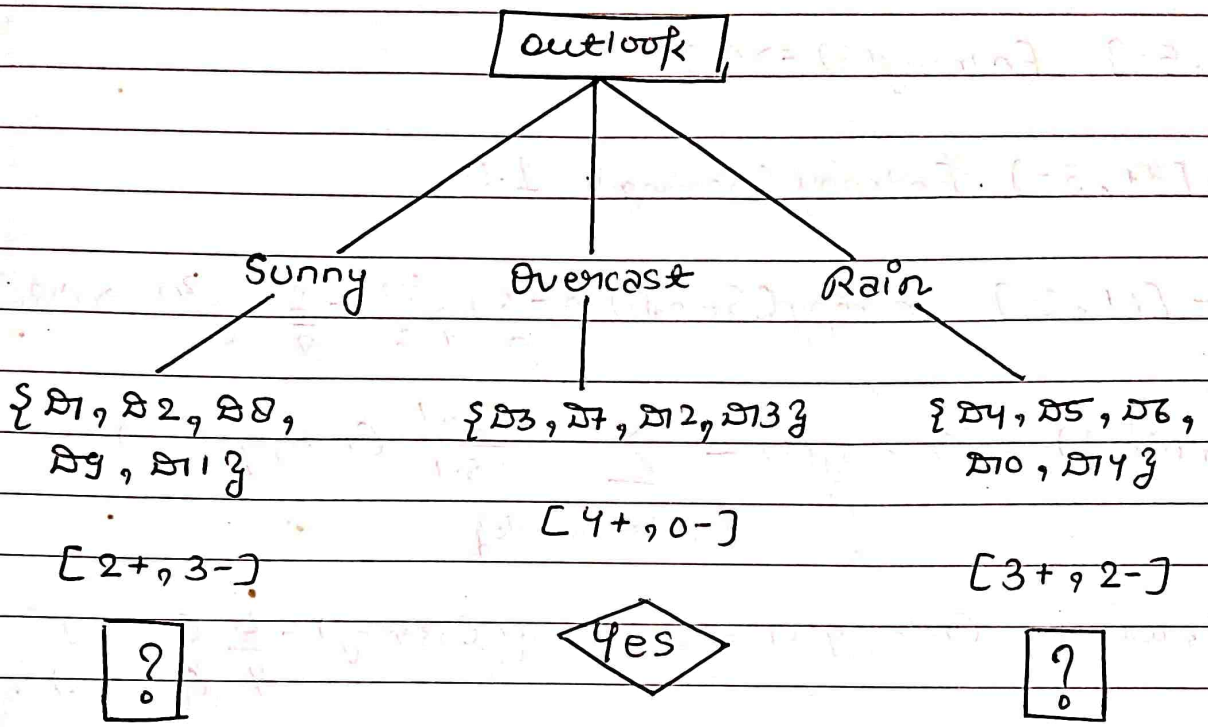
Gain (S, temp) → 0.0289

Gain (S, Humidity) → 0.1516

Gain (S, Wind) → 0.0478

{D1, D2, ..., D14}

[9+, 5-]



Day	Temp	Humidity	Wind	Play
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	yes
D11	Mild	Normal	Strong	yes

Attribute : Temp

Values (Temp) = Hot, Mild, Cool

S_{sunny} = [2+, 3-] Entropy (S_{sunny}) = $-\frac{2}{5}\log\frac{2}{5} - \frac{3}{5}\log\frac{3}{5} \Rightarrow 0.97$

S_{Hot} = [0+, 2-] S_{Mild} = [1+, 1-] S_{cool} = [1+, 0-]

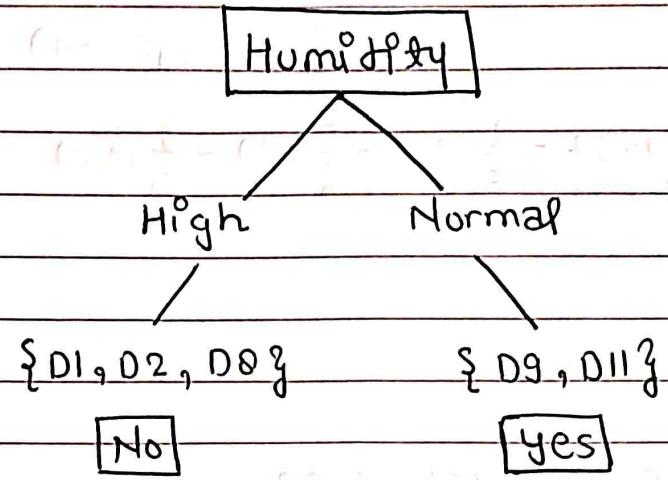
Entropy (S_{Hot}) ↓ Entropy (S_{Mild}) ↓ Entropy (S_{cool}) ↓

0.0 1.0 0.0

Gain (Ssunny, temp) = 0.570

Gain (Ssunny, Humidity) = 0.97 ← node

Gain (Ssunny, Wind) = 0.0192



Day	Temp	Humidity	Wind	play
D4	Mild	High	Weak	yes
D5	Cool	Normal	Weak	yes
D6	Cool	Normal	Strong	No
D10	Mild	Normal	Weak	yes
D14	Mild	High	Strong	No

Attribute : Humidity

Values (Humidity) = High, Normal

S_{sunny} ← [3+, 2-] Entropy ⇒ 0.97 (Ssunny)

Shigh ← [1+, 1-] Entropy (Shigh) ⇒ 1.0

Snormal ← (2+, 1-] Entropy (Snormal) ⇒ 0.9183

Gain (S_{Rain}, humidity) ⇒ 0.0192

$$Gain(S_{sunny}, Temp) = Entropy(S) - \sum_{v \in \{hot, cool, mild\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$= Entropy(S) - \frac{2}{5} Entropy(S_{hot}) - \frac{2}{5} Entropy(S_{mild}) - \frac{1}{5} Entropy(S_{cool})$$

$$Gain(S_{sunny}, Temp) \Rightarrow 0.97 - \frac{2}{5}(0.0) - \frac{2}{5}(1) - \frac{1}{5}(0.0) \Rightarrow 0.570$$

Attribute : Humidity
Values (Humidity) : High, Normal

$S_{sunny} = [2+, 3-]$ $Entropy(S) = 0.97$

$S_{high} \leftarrow [0+, 3-]$ $Entropy(S_{high}) = 0.0$
$S_{normal} \leftarrow [2+, 0-]$ $Entropy(S_{normal}) = 0.0$

$Gain(S_{sunny}, Humidity) \Rightarrow 0.97$

Attribute : Wind
Values (Wind) : Strong, Weak

$S_{sunny} = [2+, 3-]$ $Entropy(S) = 0.97$

$S_{strong} \leftarrow [1+, 1-]$ $Entropy(S_{strong}) = 1.0$

$S_{weak} \leftarrow [1+, 2-]$ $Entropy(S_{weak}) = 0.9183$

$Gain(S_{sunny}, Wind) \Rightarrow 0.0192$

Attribute : Wind

Values (wind) : strong, weak

$S_{Rain} \leftarrow [3+, 2-]$ Entropy (S) = 0.97

$S_{strong} \leftarrow [0+, 2-]$ Entropy (S_{strong}) = 0.0

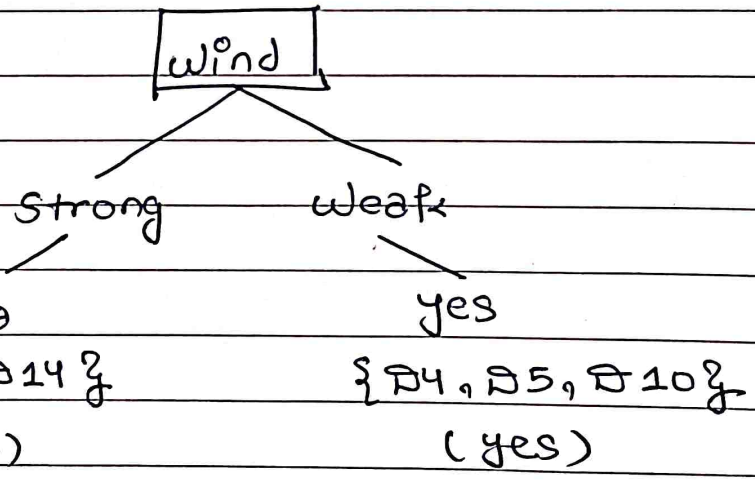
$S_{weak} \leftarrow [3+, 0-]$ Entropy (S_{weak}) = 0.0

Gain (S_{Rain}, Wind) = 0.97

Gain (S_{Rain}, Temp) = 0.0192

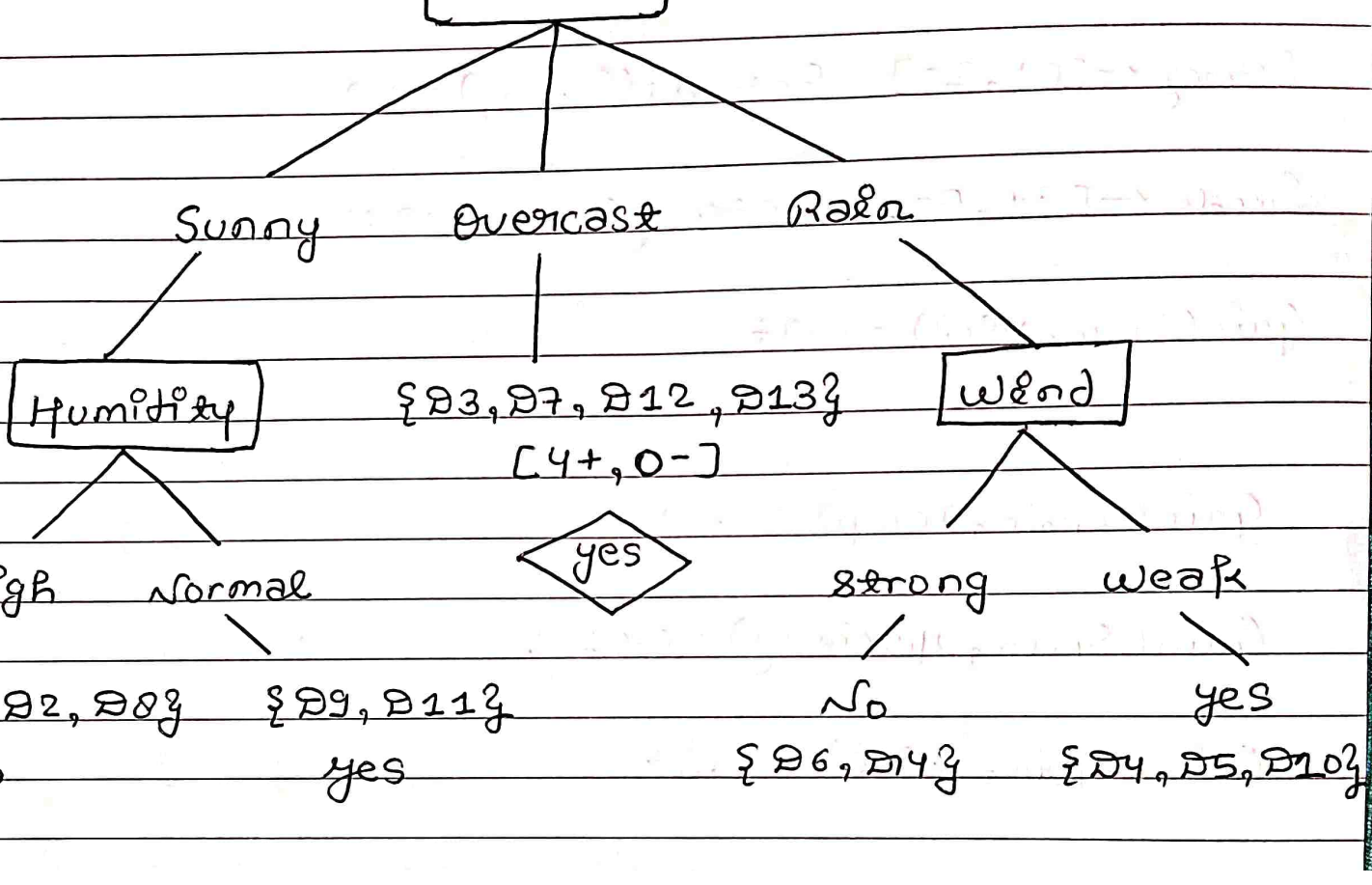
Gain (S_{Rain}, Humidity) = 0.0192

Gain (S_{Rain}, Wind) = 0.97



{D1, D2, ..., D14}

[9+, 5−]


Outlook
Sunny
Overcast
Rain
Humidity
{D3, D7, D12, D13}
[4+, 0−]
Wind
High
Normal
yes
strong
weak
{D1, D2, D8}
No
{D9, D11}
yes
No
{D6, D14}
yes
{D4, D5, D10}