Topics:

- Naive Bayes (contd) → Malh
  - 'practical' problems
    - new-words
    - Laplace smoothing + Bias-variance
    - Underflow & computational challenges
  - Multinomial NB vs Bernouli NB
  - Feature-Importance & Interpretability

- Outliers & treating them
- Gaussian NB for numerical features
- Code-sample for SVM & NB } Spam vs Not ← text
  - SVM
    - Lr-SVM
    - RBF-SVM
  - M-NB

Prev:

Naive Bayes

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

(y: no, text: $w_1 w_2 \cdots w_d$)

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

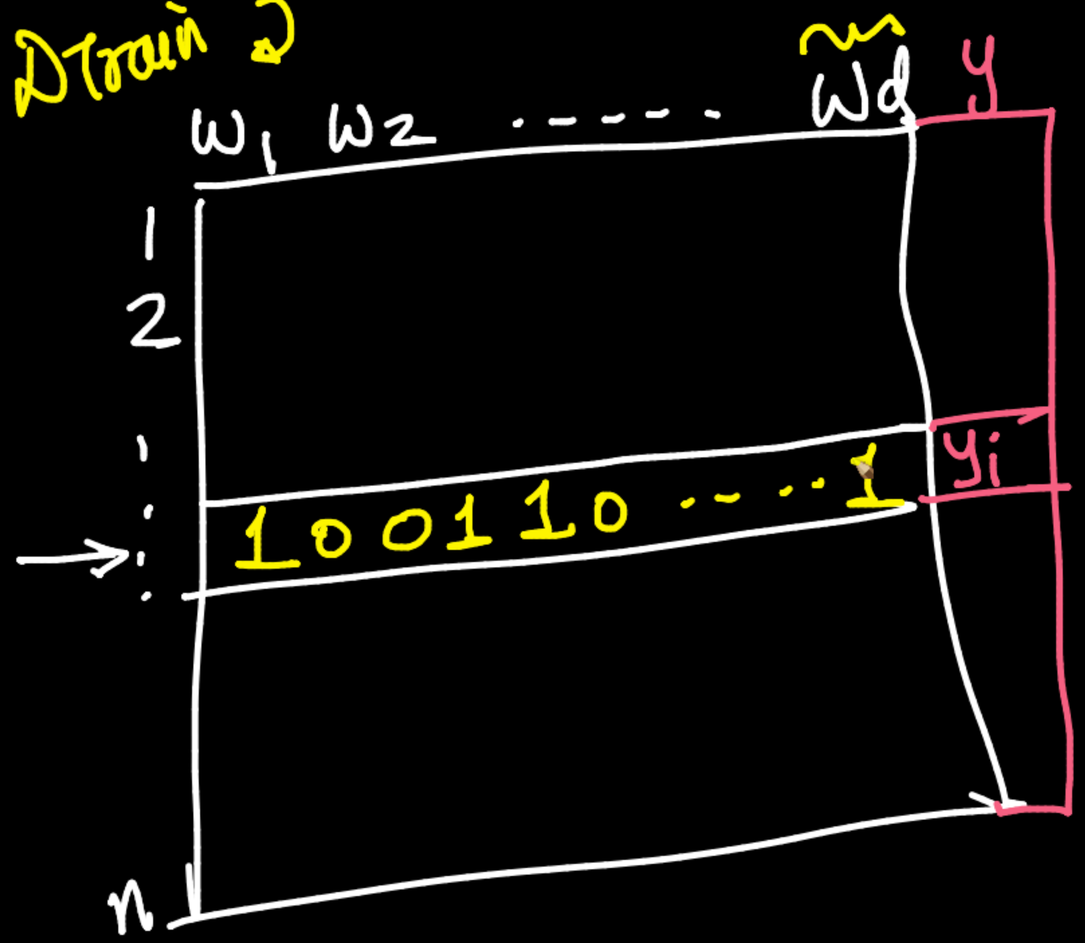
($w_1 w_2 \cdots w_d$: text)

$$P(w_1 w_2 w_3 \ldots w_d \mid y=1)$$

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

↳ independence of each feature/word conditioned on the class

Train:



Class-prior

① $P(y=1) = \frac{\#\text{ of } y_i = 1}{n}$ ✓

$P(y=0) = \frac{\#\text{ of } y_i = 0}{n}$ ✓

② likelihoods:

$\forall j : 1 \to d$

$$P(w_j | y_i = 1) = \frac{P(w_j \cap y_i = 1)}{P(y_i = 1)}$$

$$P(w_j | y_i = 0) = \frac{P(w_j \cap y_i = 0)}{P(y_i = 0)}$$

Run Test-time

$x_q = \overbrace{w_1\ w_3\ w_6\ w_{12}\ w_{15}}^{\text{K-words}}$

$y_q = 1$

$$P(y=1 \mid w_1 w_3 w_6 w_{12} w_{15}) \propto \underbrace{P(y=1)}_{\text{prior}} \times \prod_{j=1}^{k} \underbrace{p(w_j \mid y_i=1)}_{\text{likelihood}}$$

$$\Big\downarrow$$

$$P(y=0 \mid w_1 w_3 w_6 w_{12} w_{15}) \propto \underbrace{p(y=0)}_{\text{prior}} \prod_{j=1}^{K} \underbrace{p(w_j \mid y_i=0)}_{\text{likelihood}}$$

$2K+2$

Q

@ Run-time

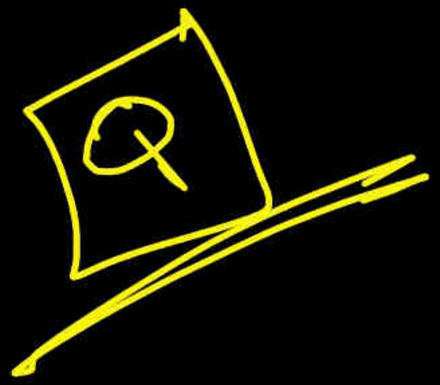
Space-complex of NB = O(d)

likelihoods: 2d

class priors: 2

2d+2 probabilities

total #words in the total dataset



Run time - complx:

xq → k-words

O(1)

O(k)

Q

$$P(y=1 \mid \overbrace{w_1 w_2 w_3 \cdots w_k}^{x_q}) \propto p(y=1) \prod_{j=1}^{k} p(w_j \mid y=1)$$

$$P(w_1 w_2 w_3 \cdots w_k) \times$$

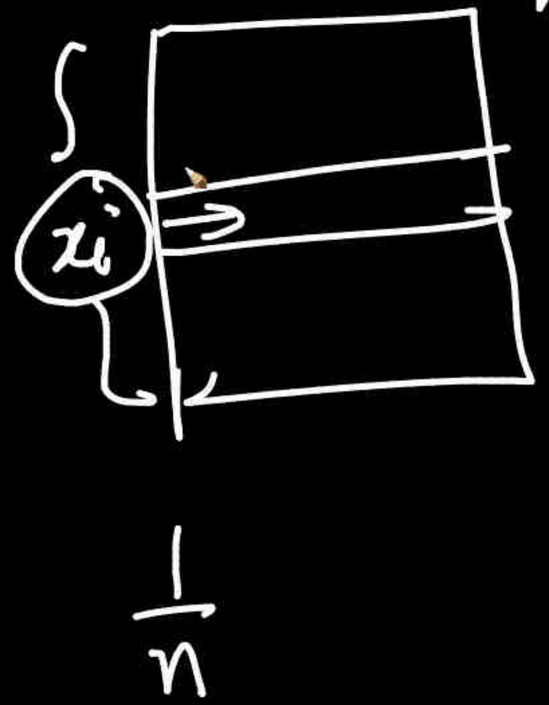
$$P(y=0 \mid w_1 w_2 \cdots w_k) \propto p(y=0) \prod_{j=1}^{k} p(w_j \mid y=0)$$

$$p(w_1 w_2 \cdots w_k) \times$$

popln

sample ⇒ D_{train}

den: $p(\underbrace{w_1, w_2 \cdots w_K}_{x_i})$



$\frac{1}{n}$

Practical - issues

Run-time

$x_q = w_1\ w_2\ w_6\ w_7\ w'$

$w' \notin \{w_1, w_2, \ldots w_3\}$

↓

Set of words in Dtrain

$$P(y=1 \mid w_1 w_2 w_6 w_7 w') = P(y=1) \times P(w_1 \mid y=1) \cdots P(w' \mid y=1)$$

$P(w' \mid y=1) = 0$

$w' \notin \{w_1 \cdots w_d\}_{\text{train}}$

$P(y=1 \mid w_1 w_2 w_6 w_7 w') = 0$

$P(y=0 \mid w_1 w_2 w_6 w_7 w') \neq 0$

Likelihoods

$$P(w_j \mid y=1) = \frac{n_{j1}}{n_1}$$

n_{j1} → Drain

n_1 → Drain

$$P(w' \mid y=1) = \frac{n'_1}{n_1} = \frac{0}{n_1} = 0$$

hack

- ignore ✓
- Laplace Smoothing

$$P(w_j \mid y=1) = \frac{n_{j,1} + \alpha}{n_1 + \alpha k}$$

$n_{j,1} + \alpha$ ← 1

Laplace Smoothing (Additive)

$n_1 + \alpha k$ ← # distinct values than w_j can take

w_j → Bernoulli (2)

Simplicity:

$\alpha = 1$

$k = 2$

$$p(w' \mid y=1) = \frac{n'_1}{n_1} = \frac{0+1}{100+2} = \frac{1}{102}$$ ✓

will not break my math

NB + Laplace-Smoothing

for all likelihoods:

(train time & test time)

$$P(w_j | y=1) = \frac{n_{j1} + \alpha}{n_1 + 2\alpha}$$

(let $\alpha = 1$)

n_{j1}: 1 or 0

n_1: 0

$$P(w_j | y=1) = \frac{1}{2}$$

(∵ we don't any data)

$n_1' = 0$

Case 1:

$\alpha = 1$

$n_1 = 100$

$$p(w' \mid y=1) = \frac{n_1' + \alpha}{n_1 + 2\alpha} = \boxed{\frac{1}{100+2}}$$

n_1': 100

Case 2:

$\alpha = 10{,}000$

$n_1 = 100$

$$p(w' \mid y=1) = \frac{0 + 10000}{100 + 20000} \approx \boxed{\frac{1}{2}}$$

α → 2α

$\alpha_1 \gg n_1$

Chrome File Edit View History Bookmarks Profiles Tab Window Help

Train time

w_j

$n_{j|1} = 10$; $n_j = 100$ (let)

No laplace smoothing :- $\frac{n_{j|1}}{n_1} = 0.1$

Case 1:

$$p(w_j \mid y=1) = \frac{n_{j|1} + \alpha}{n_1 + 2\alpha} = \frac{10+1}{100+2} \approx 0.1$$

$\alpha = 1$

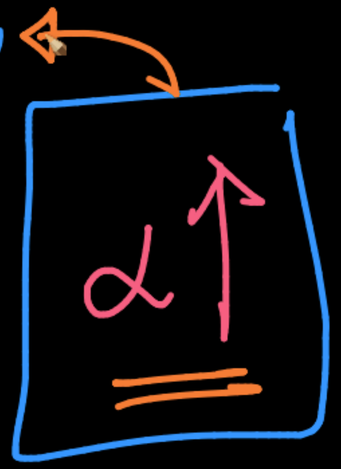
Case 2:

$\alpha = 10{,}000$

$$p(w_j \mid y=1) = \frac{10 + 10000}{100 + 20000} \approx \frac{1}{2}$$

19 / 19

Underfitting



Same as KNN; K=n

$P(y=1) = 0.7$

$P(y=0) = 0.3$

⇓

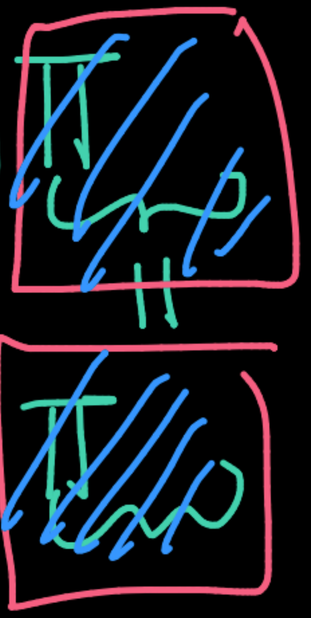
for all x_q: $y_q = 1$

all my likelihoods will be close to 1/2 ($\because K=2$)

⇓

$P(y=1 \mid \underbrace{w_1 \dots w_3}_{x_q}) = P(y=1)$

$P(y=0 \mid \underbrace{w_1 \dots w_3}_{x_q}) = P(y=0)$



$\alpha \downarrow$ overfit

likelihoods $\rightarrow$ $\frac{n_{j1}}{n_1}$ & $\frac{n_{j0}}{n_0}$

$$p(w \mid y=1) = 1 \qquad w' \in \{w_1, \ldots w_3\}$$

X

no hyper-param to play with

Naive Bayes + Laplace Smoothing

↓

Hp: $\begin{cases} \alpha \uparrow & \text{underfit} \\ \alpha \downarrow & \text{overfit} \end{cases}$

(Q)

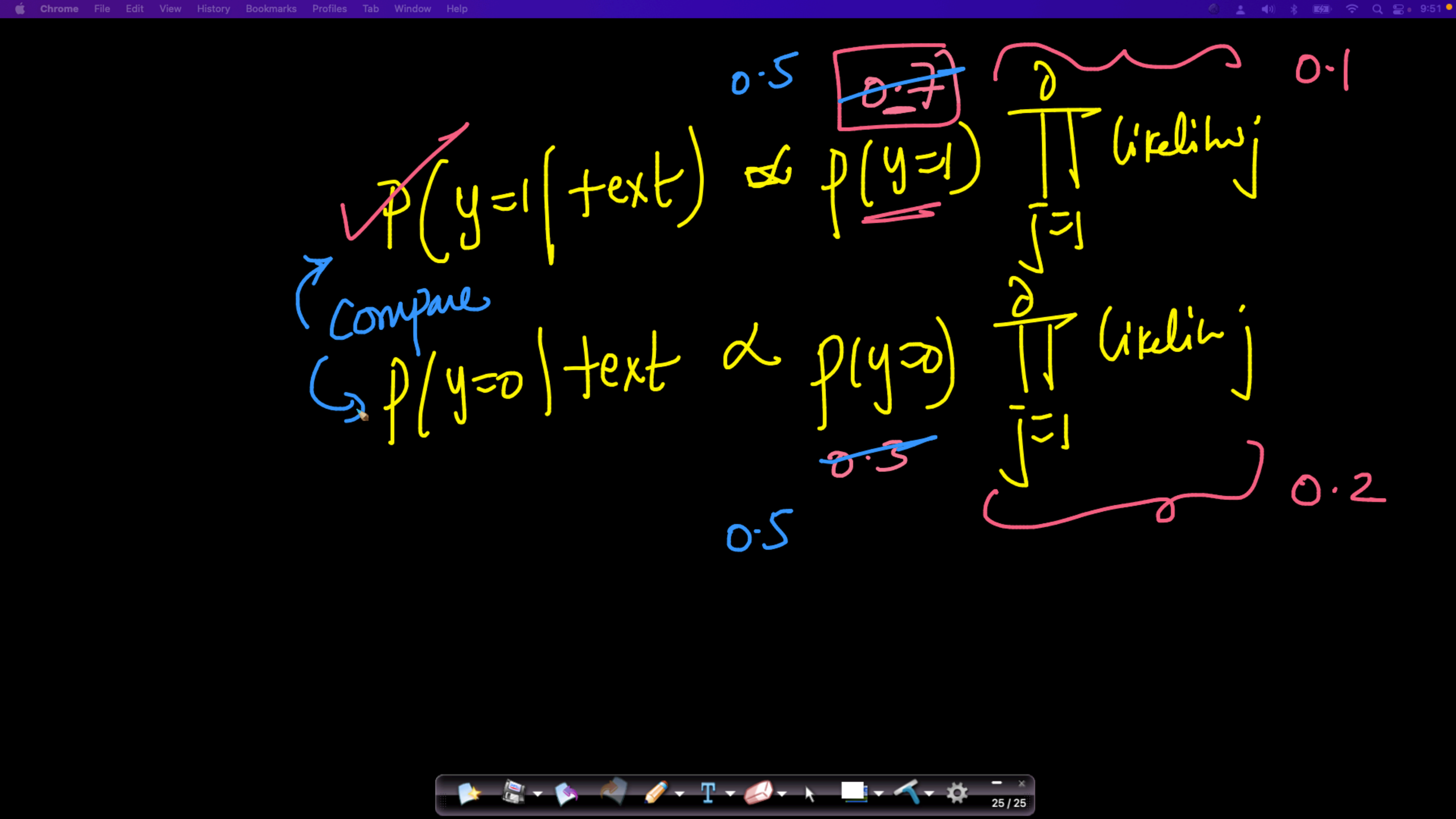
$p(y=1) = 0.7$

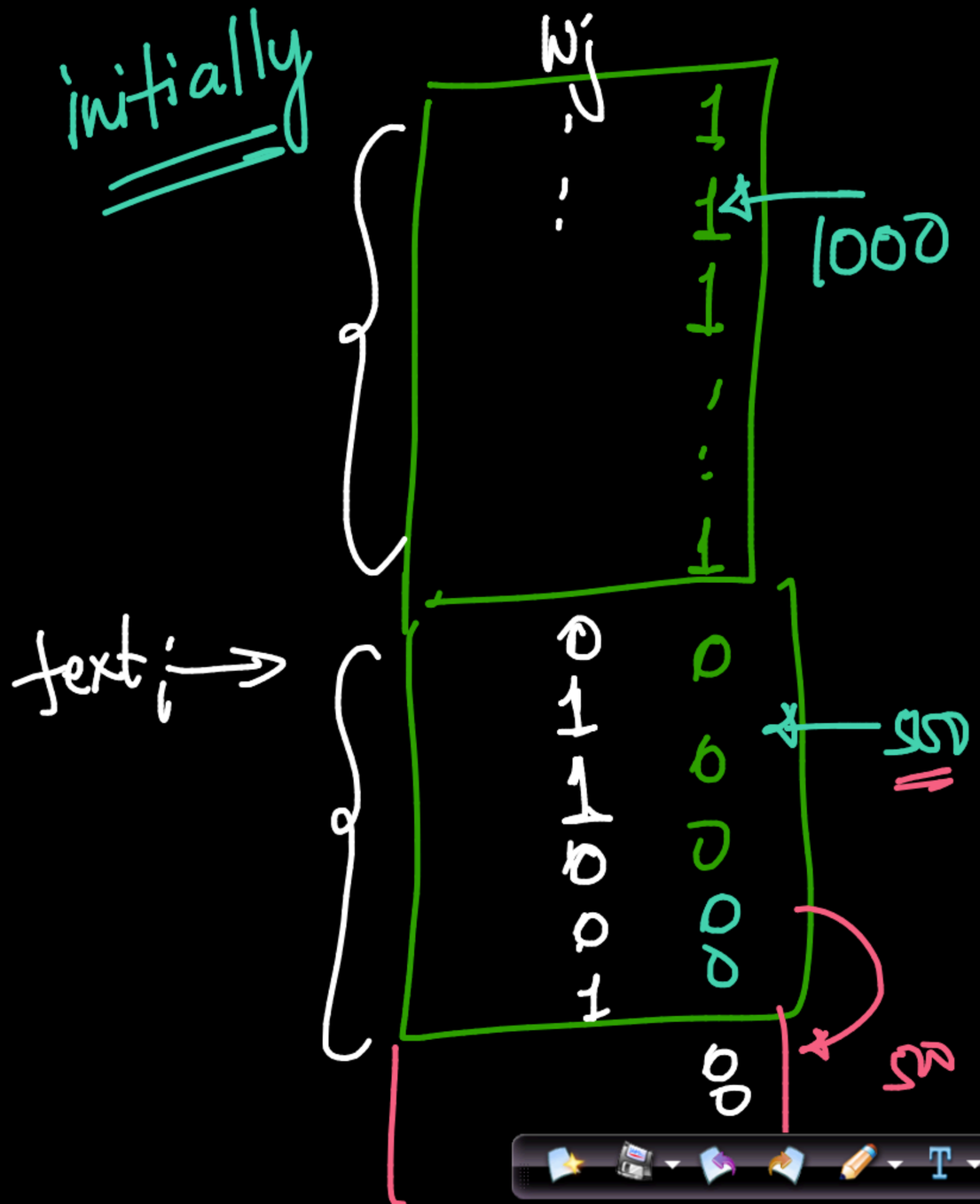
$p(y=0) = 0.3$

} ⇒ imbalance

Yes, it is good to rebalance ✓

rebalance ⟶ Upsampling


0.5
0.7
0.1
$P(y=1 \mid text) \propto P(y=1) \prod_{j=1} likelihood_j$
compare
$P(y=0 \mid text \propto P(y=0) \prod_{j=1} likelihood_j$
0.5
0.5
0.2



before rebalancing :

$$p(w_j | y=0) = \frac{n_{j0}}{n_0} = \frac{50+k}{500+2k}$$

rebalancing by upsampling :

$$p(w_j | y=0) = \frac{n_{j0}}{n_0} = \frac{500+k}{1000+2k}$$

$$P(y=1 \mid w_1 w_2 \cdots w_d) \propto P(y=1) \prod_{j=1}^{d} P(w_j \mid y=1)$$

Computational problem →

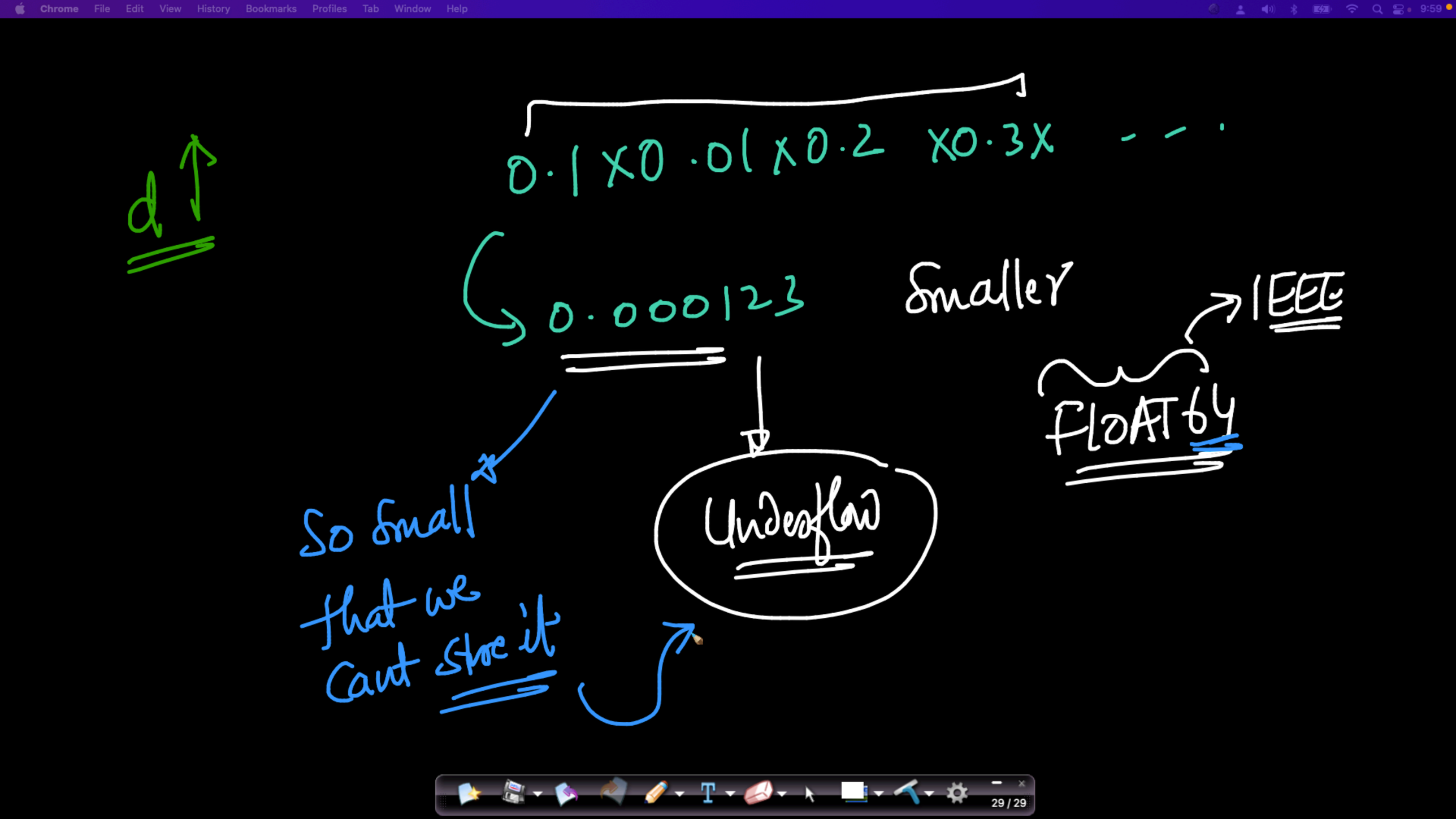
Code

$$P(y=1 \mid w_1 w_2 \cdots w_d) \propto \underbrace{P(y=1)}_{0 \text{ to } 1} \prod_{j=1}^{d} \underbrace{P(w_j \mid y=1)}_{0 \text{ to } 1}$$

d → 100

Computational Problem →

$(d+1)$ $[0 - 1]$


d ↑
0.1 X0.01 X0.2 X0.3X - - -
0.000123
Smaller
IEEE
FloAT64
So small
that we
Cant store it
Underflow
29 / 29

1. reduce d → X

How to solve underflow problem

2. scaling :-

3. Log ✓

$$\log\left(P(y=1 \mid w_1 w_2 \dots w_d)\right) = \log\left[P(y=1) \cdot \prod_{j=1}^{d} P(w_j \mid y=1)\right]$$

$$= \log P(y=1) + \sum_{j=1}^{d} \log P(w_j \mid y=1)$$

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

$\log(P(y=1 \mid w_1 \cdots w_3))$

compare

$\log(P(y=0 \mid w_1 \cdots w_3))$

→ $P(y=1 \mid w_1 \dots w_d)$

↳ Bayes + Naivity

$P(w_1 w_2 \dots \mid y=1) = \prod P(w_i \mid y=1)$ } Recap

Bernoulli NB

$w_j \rightarrow 0$
$\rightarrow 1$

→ likelihoods & Class prior (Train)

→ $P(y=1 \mid x_q) : P(y=0 \mid x_q)$ → NB

→ $p(w_j|y=1)$ $\quad w_j \notin D_{train}$

↳ $\dfrac{n'_j + \alpha}{n_1 + K\alpha}$

↑ 2 as w_j takes 2 values

→ $\alpha \uparrow$ underfit

$\alpha \downarrow$ overfit

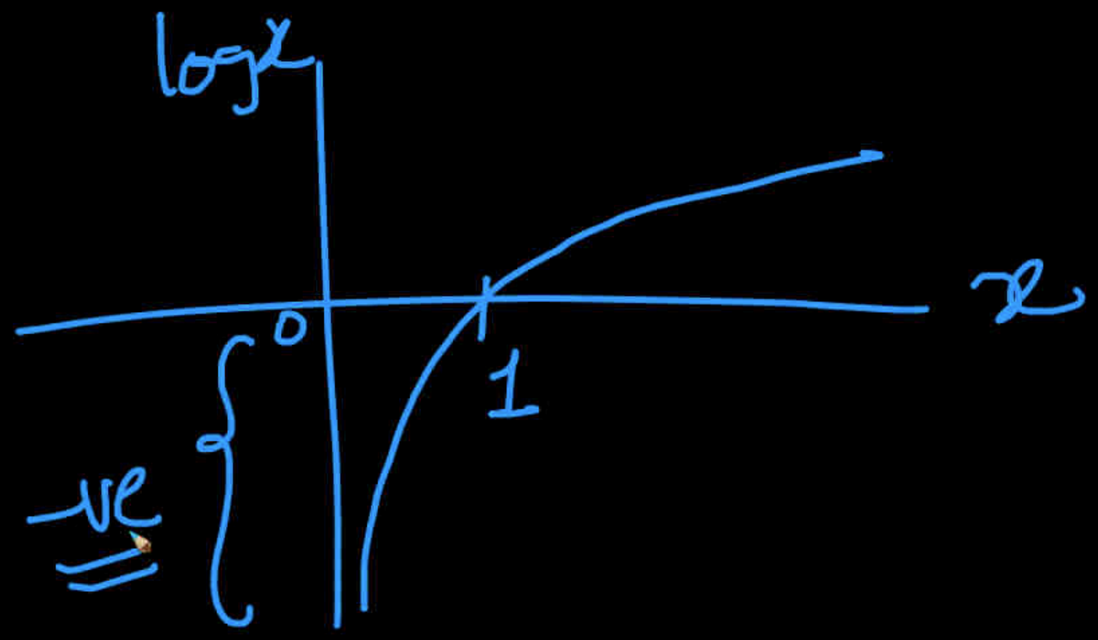
→ log probs → replacing products with Sum (underflow) ✗

0 — 1

-ve value → log(probs..)

-ve

$x \uparrow \quad \log(x) \uparrow$



Chrome File Edit View History Bookmarks Profiles Tab Window Help 10:08

Warning

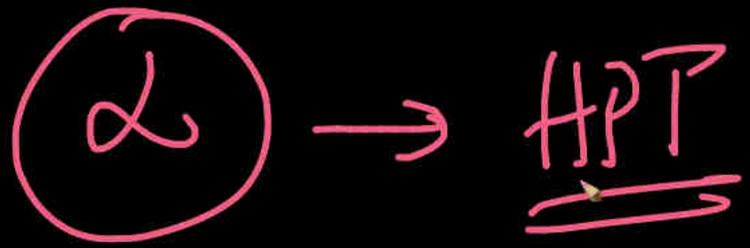
V. Common mistake

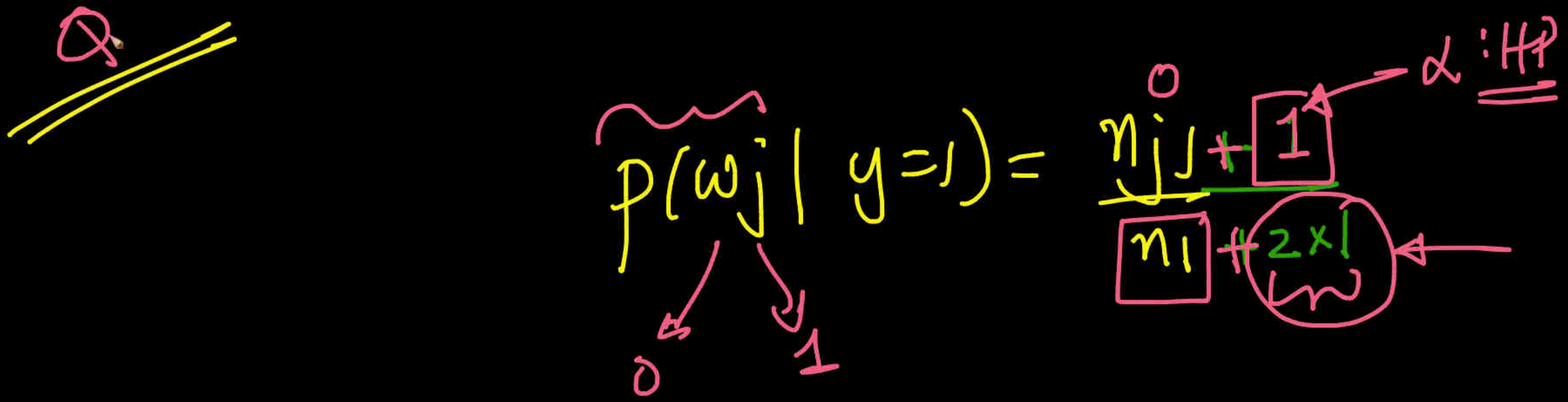
Naive → w_i & w_j are indep of one another

$$P(w_i\, w_j \mid y=1)$$

$$= P(w_i \mid y=1)\, P(w_j \mid y=1)$$

37 / 37


2
HPT

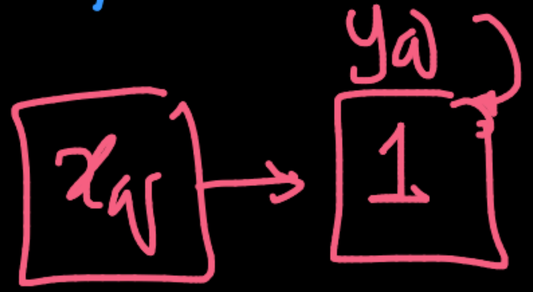

Q
p(wj | y=1) =
0
1
0
nj1 + 1
n1 + 2x1
α : HP

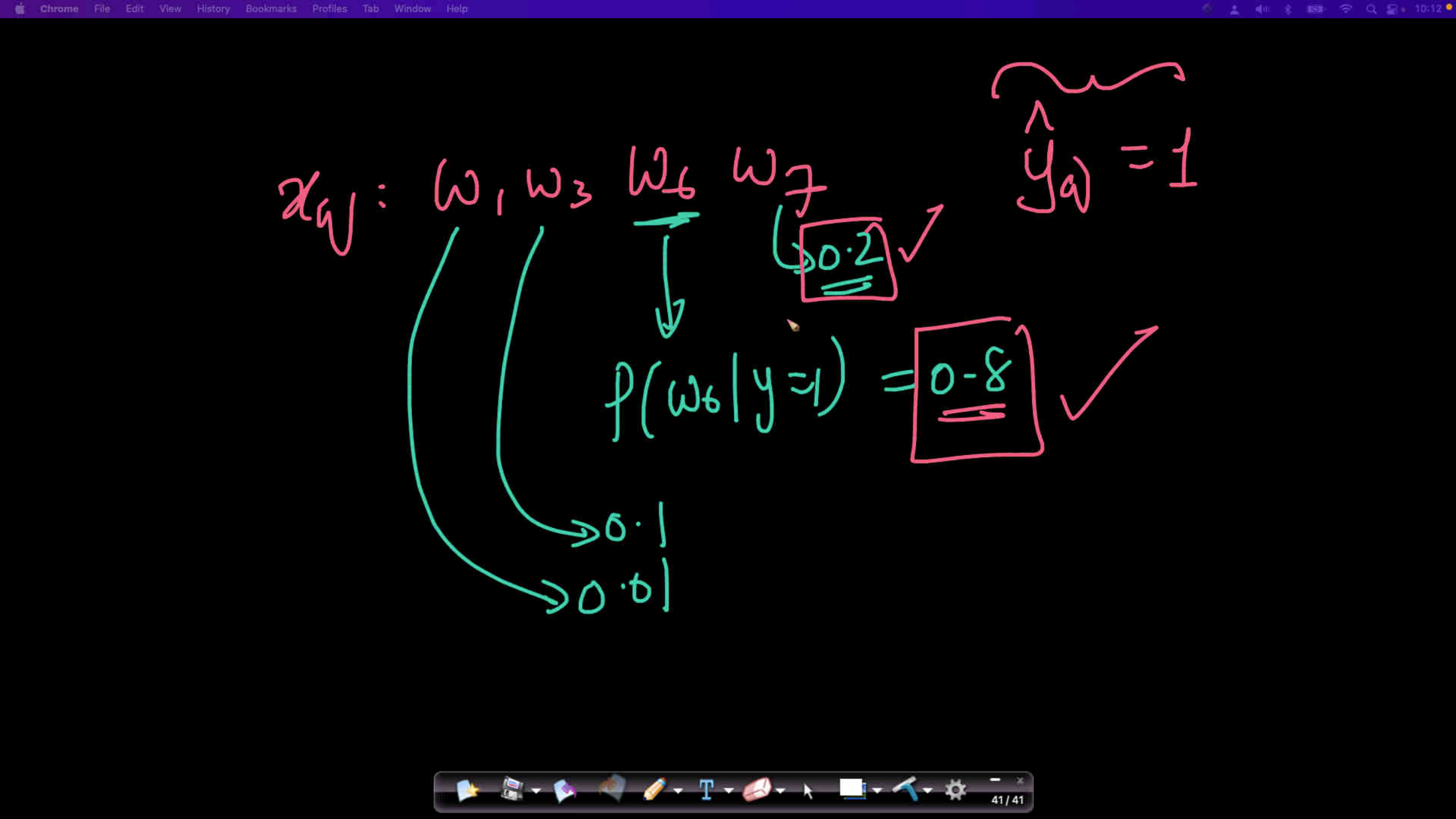
✓ Class-priors → $P(y=1)$, $P(y=0)$

✓ likelihoods → $P(w_j \mid y=1)$, $P(w_j \mid y=0)$ → top likelihoods

F.I & Interpretability

feature Imp & Interpretability




x_q : w_1, w_3 w_6 w_7
$\hat{y}_q = 1$
0.2
$p(w_6 | y=1) = 0.8$
0.1
0.01

Outliers:

@Test time:-

$x_q = w_1 w_3 w_4 w'$ & D_{train}

↳ outlier → mitigated by laplace smoothing

@ Train time

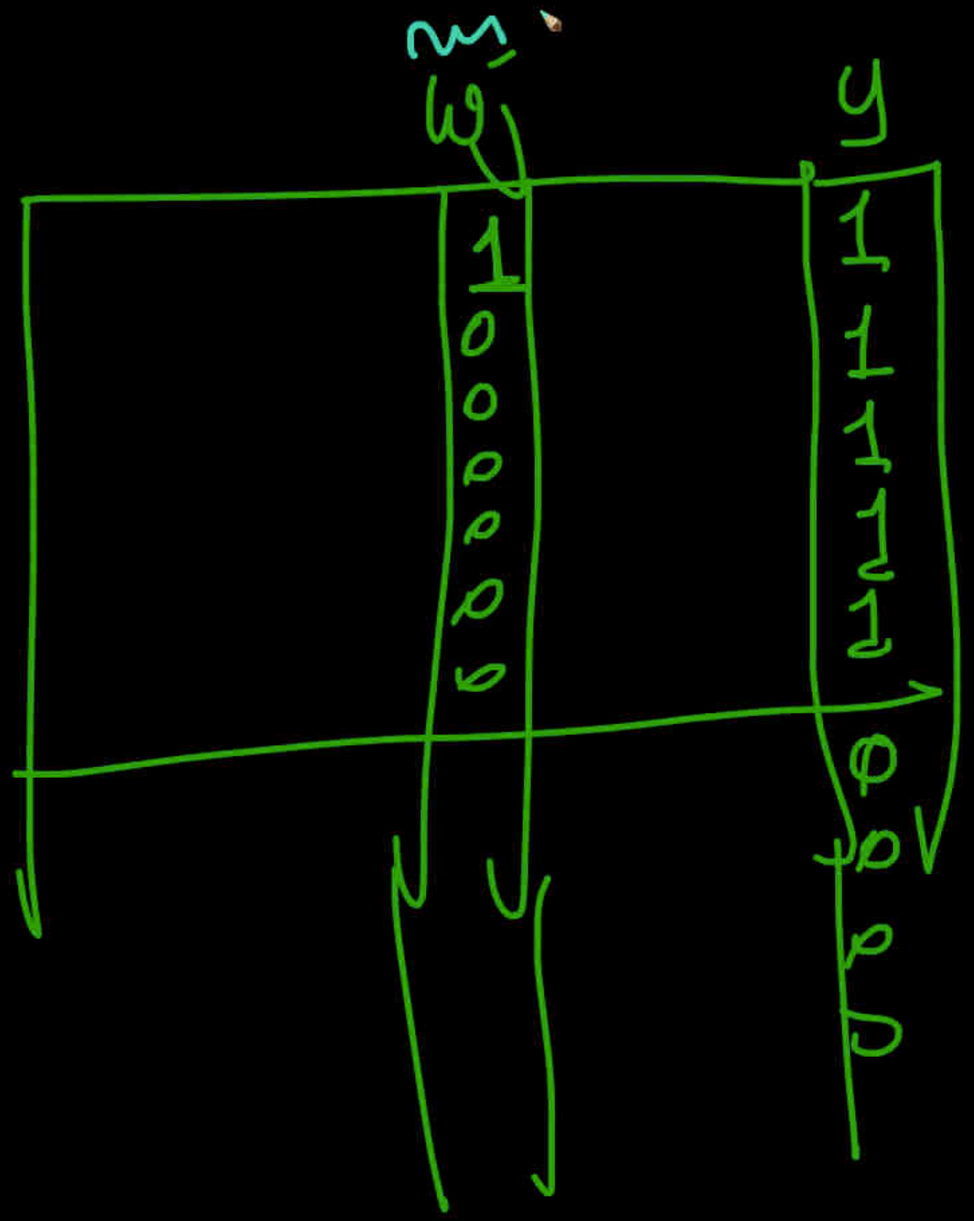
outlier? → rare word / noise

$$P(w_j \mid y=1) = \frac{n_{j1}}{n_1} = \frac{1 + \alpha}{1000 + 2\alpha}$$

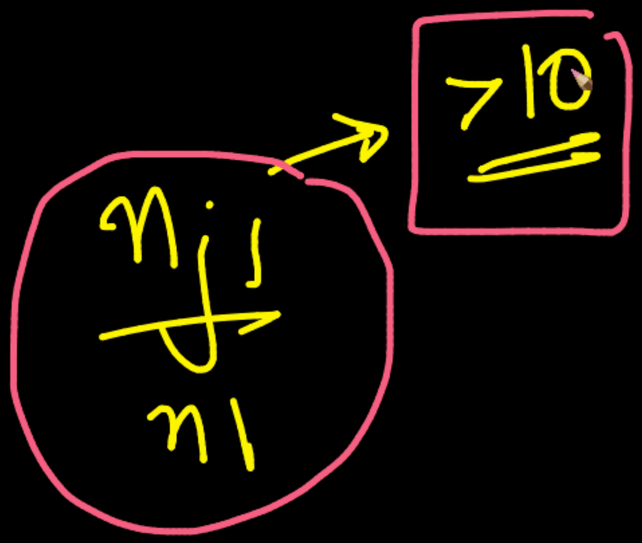
$\alpha \to 100$

{ laplace smoothing

larger α


w
y
1
0 0 0 0 0 0
1 1 1 1 1 1
0 0 0 0 0 0

hack:



only then we will even construct

$w_j x$

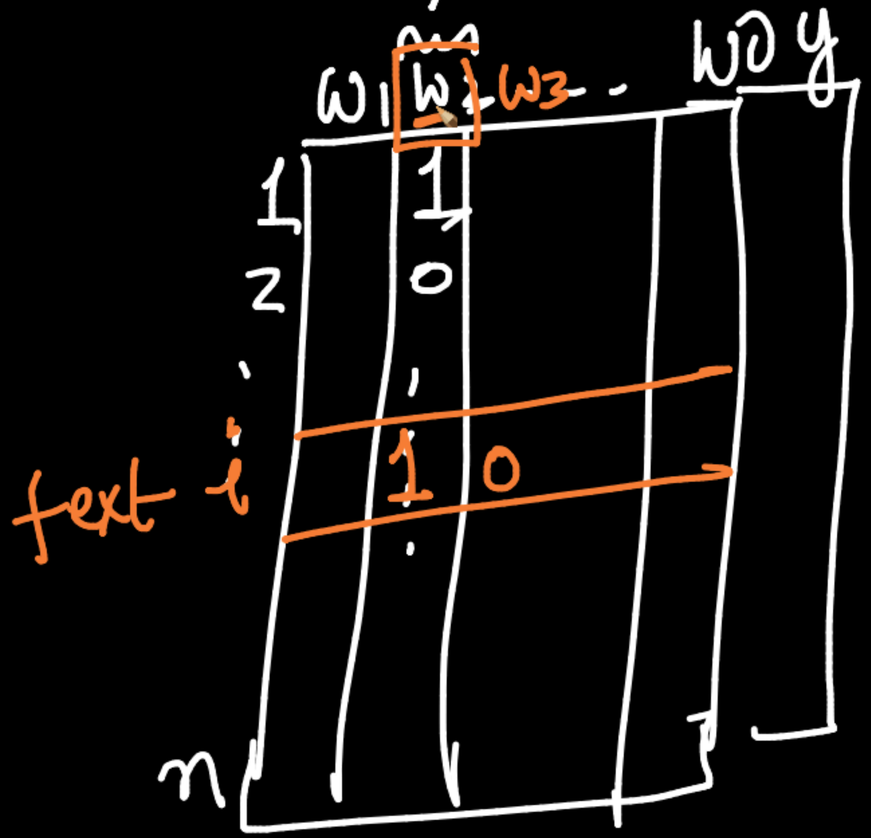
Theoretically: one α per word

Practice → we avoid (one α)

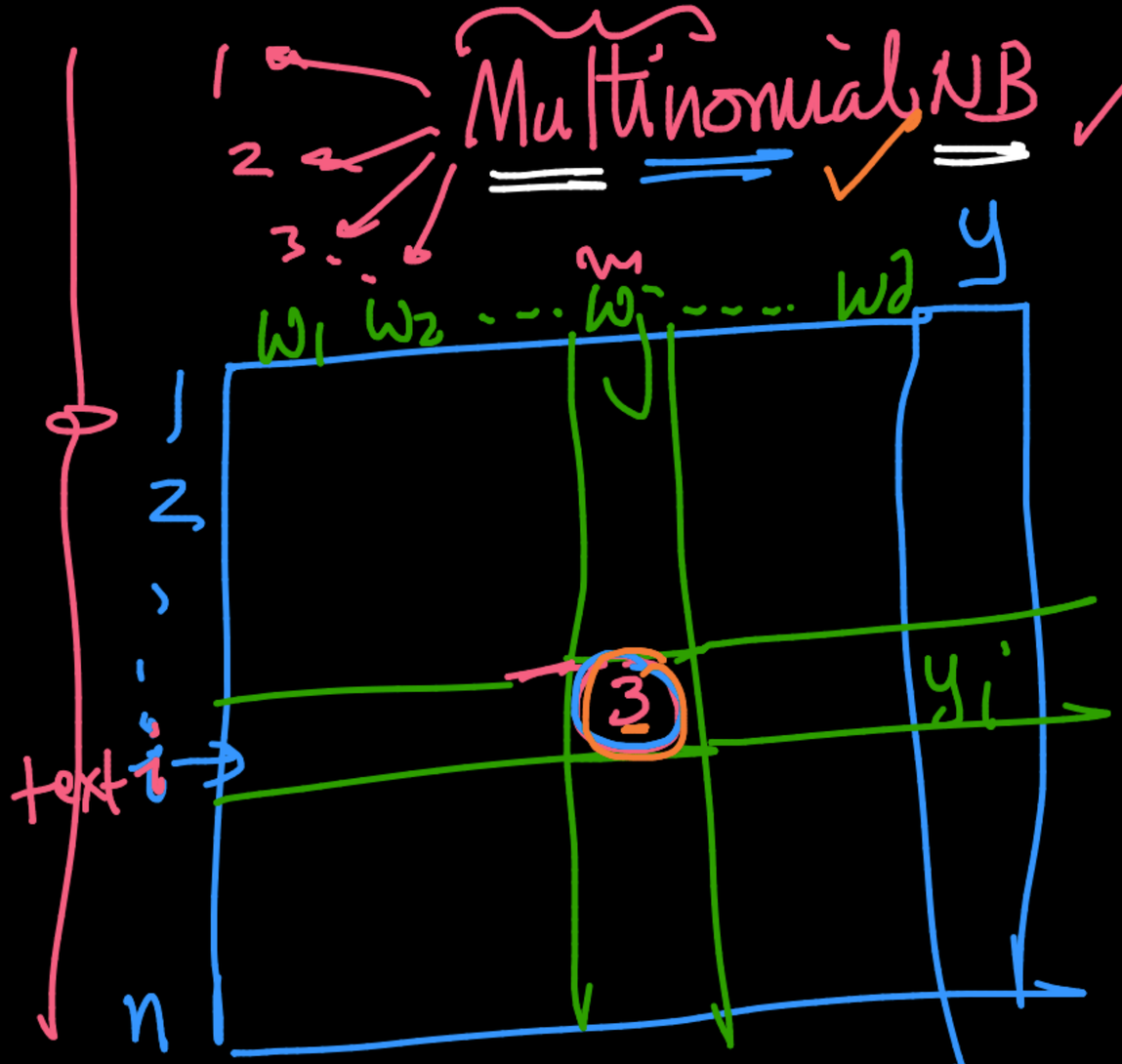
#words: 10,000

$\alpha_1 \alpha_2 \cdots \alpha_{10,000}$ → V. hard to HPT

Bernouli NB → 0 → 1



Multinomial NB



max-Info ✓

$P(w_j = 3 \mid y = 1)$

Bernoulli(p)

$0 \rightarrow 1-p = q$

$1 - p$

Multinomial (K-values)

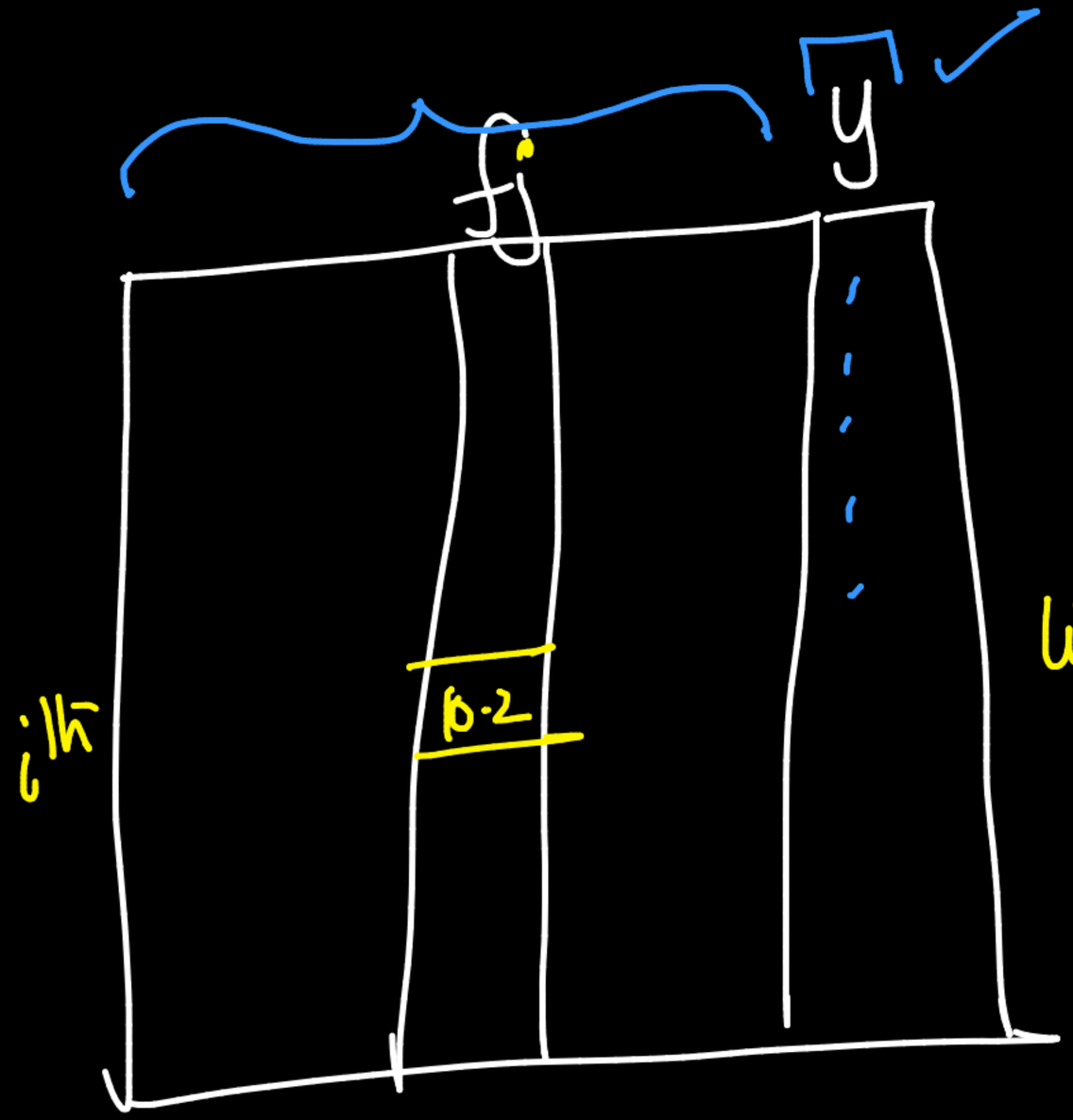
$p_1, p_2 \cdots p_{K-1}$

NB → Class priors → $P(y=1)$ $P(y=0)$

→ Likelihoods → $P(w_j = c \mid y=1)$

$P(w_j = c \mid y=0)$

Non text



fj : real-valued

P(y=1) & P(y=0) ✓

likelihoods

$P(f_j = 10.2 \mid y=1)$

$P(f_j = 10.2 \mid y=0)$

Continous r.V

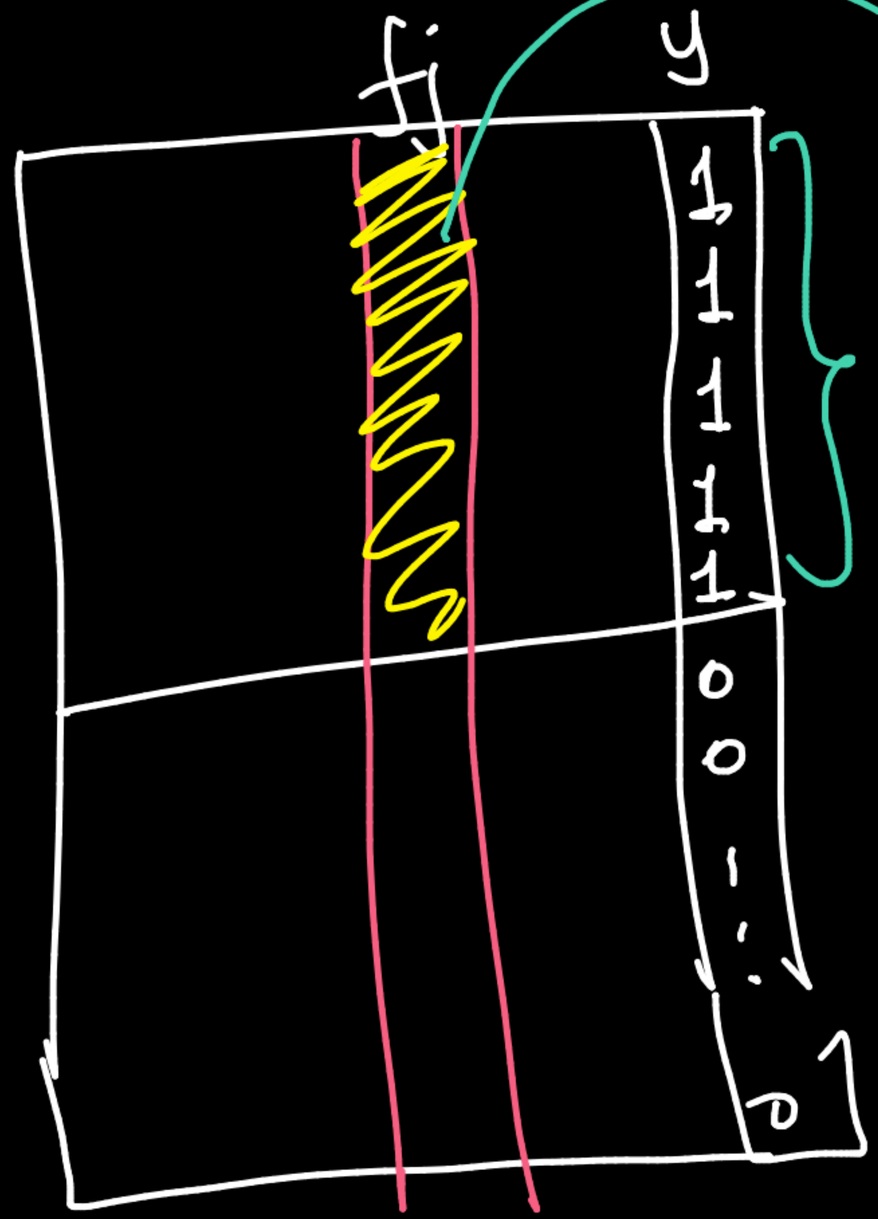
$P(f_j = 10.2 \mid y = 1)$

Conditional PDF

↳ pdf of $w_j \mid y = 1$

✓ $P(f_j)$ = PDF of f_j

Dtrain



$x_1, x_2 \dots x_{n1}$: observed values

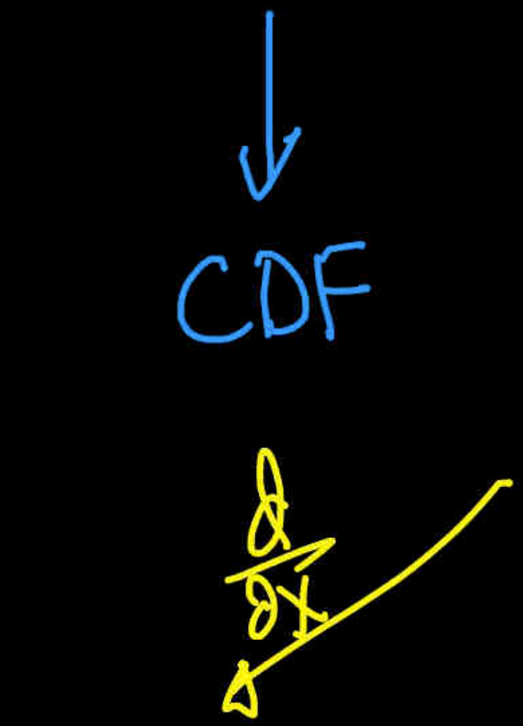
$\downarrow$

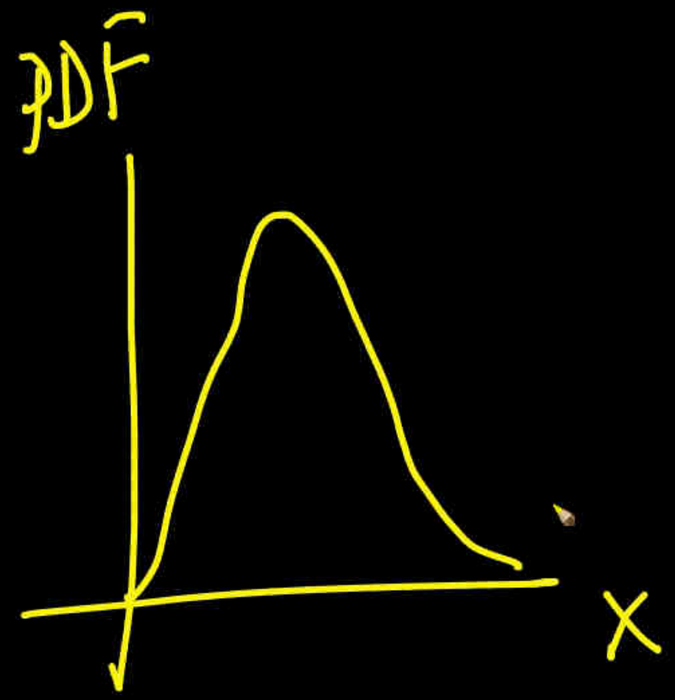
PDF of $\widetilde{fj} \mid \widetilde{y=1}$

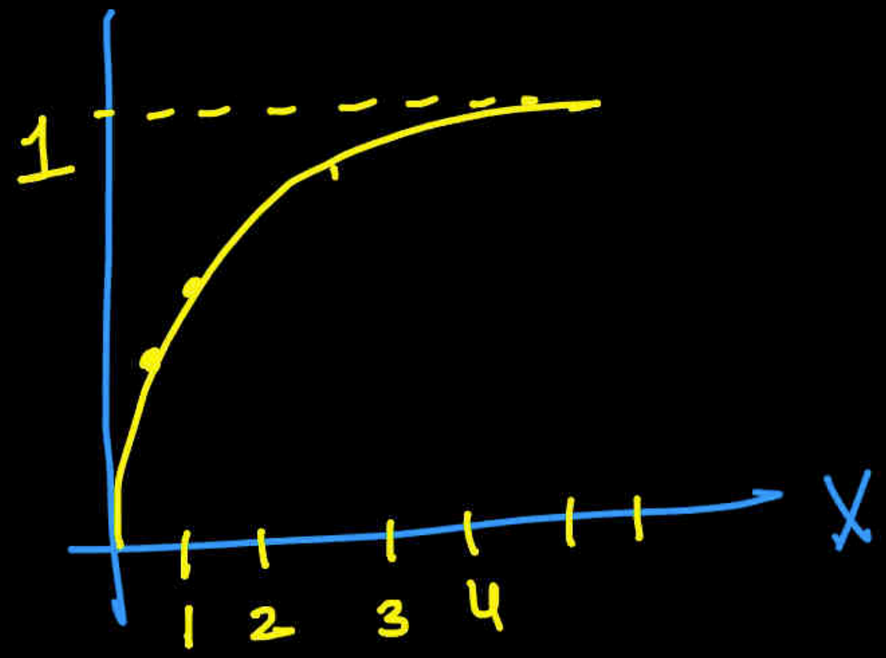
$\downarrow$

G

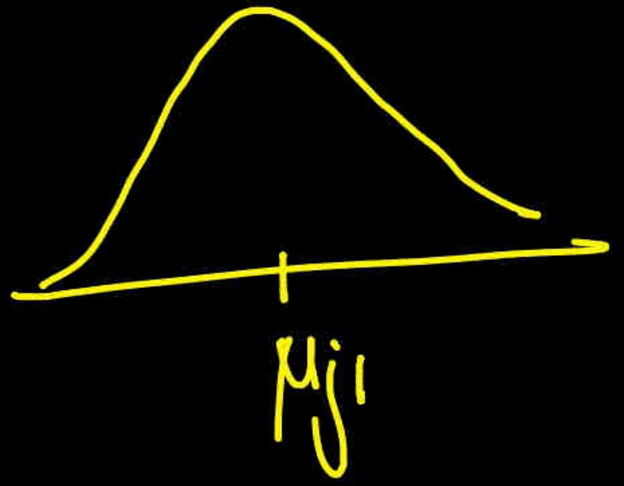
$x_1, x_2 \cdots x_{n1}$ → emperical PDF



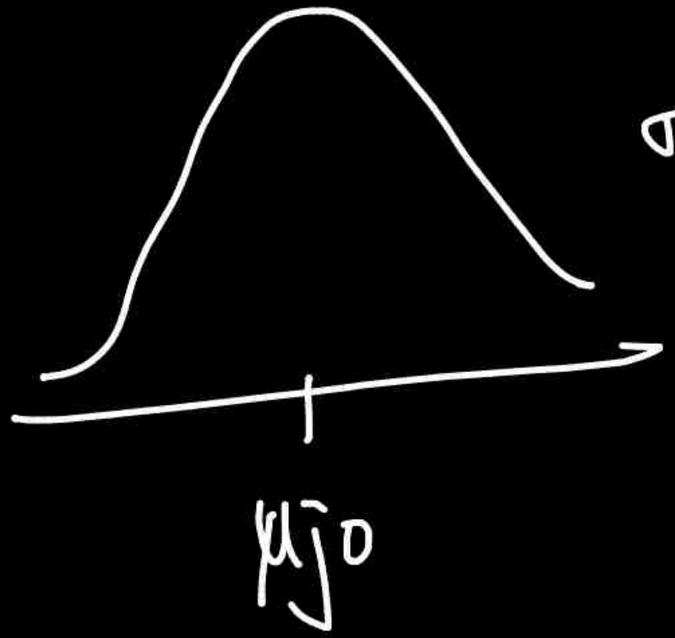




$x_j :-$ $f_j | y=1$ →



$x_j :-$ $f_j | y=0$



if all conditionals PDFs are gaussian

Gaussian NB

Gaussian NB

Train:

Class priors: $P(y=1)$ & $P(y=0)$

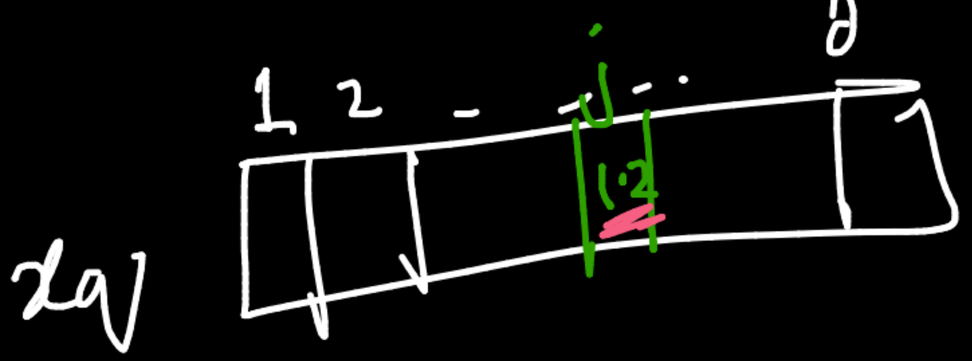
likelihoods!

$f_j | y=1 \rightarrow N(\mu_{j1}, \sigma_{j1})$

$f_j | y=0 \rightarrow N(\mu_{j0}, \sigma_{j0})$

QB-pIA/SW/AD/KS tests

Test time

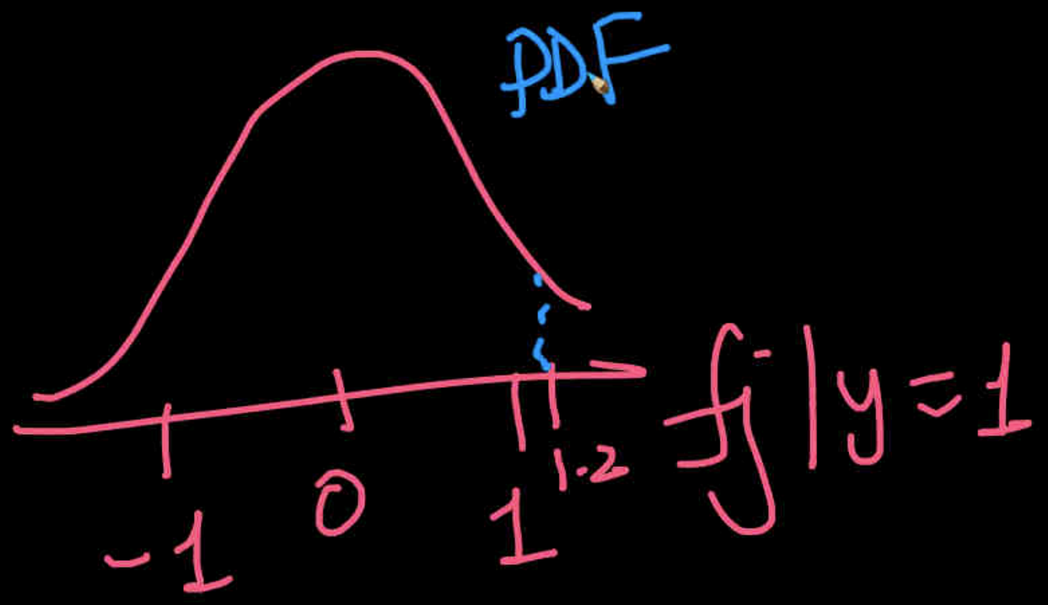


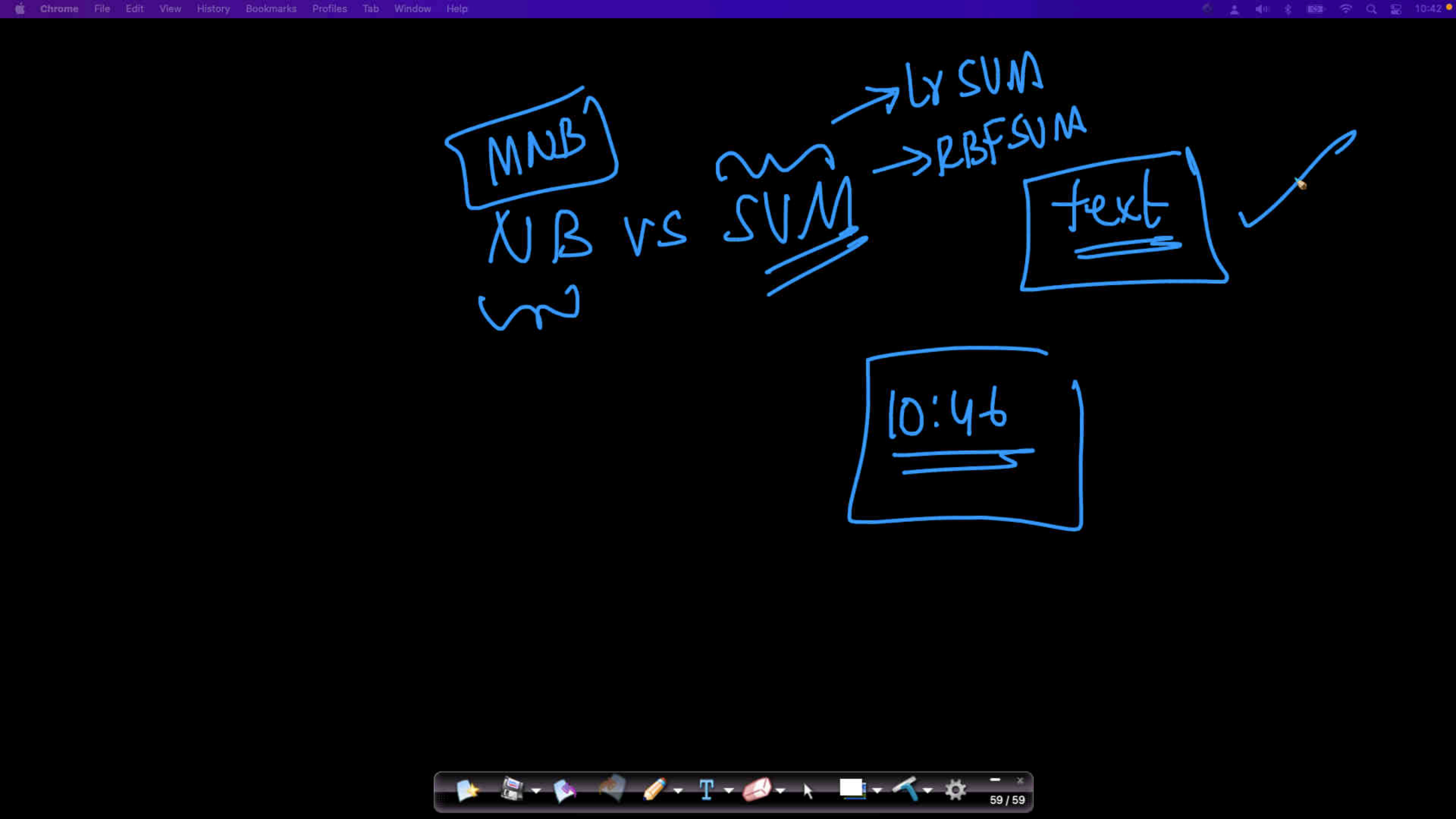
$\sim N(\mu_{j1}, \sigma_{j1})$

$P(w_j = 1.2 \mid y=1)$

$$P(y=1 \mid x_{(q)}) \propto P(y=1) \prod_{j=1}^{d} P(w_j \mid y=1)$$

$$P(y=0 \mid x_{(q)}) \propto P(y=0) \prod_{j=1}^{d} P(w_j \mid y=0)$$




MNB
NB vs SVM
LV SVM
RBFSVM
text
10:46

```
# Importing libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from collections import Counter
from sklearn import feature_extraction, model_selection, naive_bayes, metrics, svm
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC

# Libraries for text processing
import re, nltk
nltk.download('punkt')
nltk.download('stopwords')
from nltk import word_tokenize, sent_tokenize
from nltk.corpus import stopwords

import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt
```

```
[ ]  from sklearn.model_selection import train_test_split, cross_val_score
     from sklearn.preprocessing import StandardScaler

     from sklearn.svm import SVC

     # Libraries for text processing
     import re, nltk
     nltk.download('punkt')
     nltk.download('stopwords')
     from nltk import word_tokenize, sent_tokenize
     from nltk.corpus import stopwords

     import warnings
     warnings.filterwarnings('ignore')
     %matplotlib inline
```

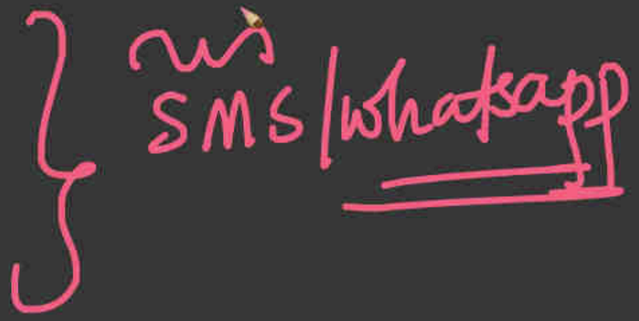
```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
[ ]  #Data: https://drive.google.com/file/d/1UEc9IY2HgIAYOsm4qpXdqUO5a1ZyeyTV/view?usp=sharing
     id = "1UEc9IY2HgIAYOsm4qpXdqUO5a1ZyeyTV"
     path = "https://docs.google.com/uc?export=download&id=" + id
     print(path)
```

```
2022-05-27 14:32:11 (121 MB/s) - 'spam_clean.csv' saved [483640/483640]
```

```
df = pd.read_csv('./spam_clean.csv', encoding='latin-1')
df.head()
```

	type	message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...




```
freq = pd.value_counts(df["type"], sort= True)
freq.plot(kind= 'bar', color= ["red", "green"])
plt.title('Distribution of Dependent feature')
plt.show()
```

Chrome File Edit View History Bookmarks Profiles Tab Window Help 10:49

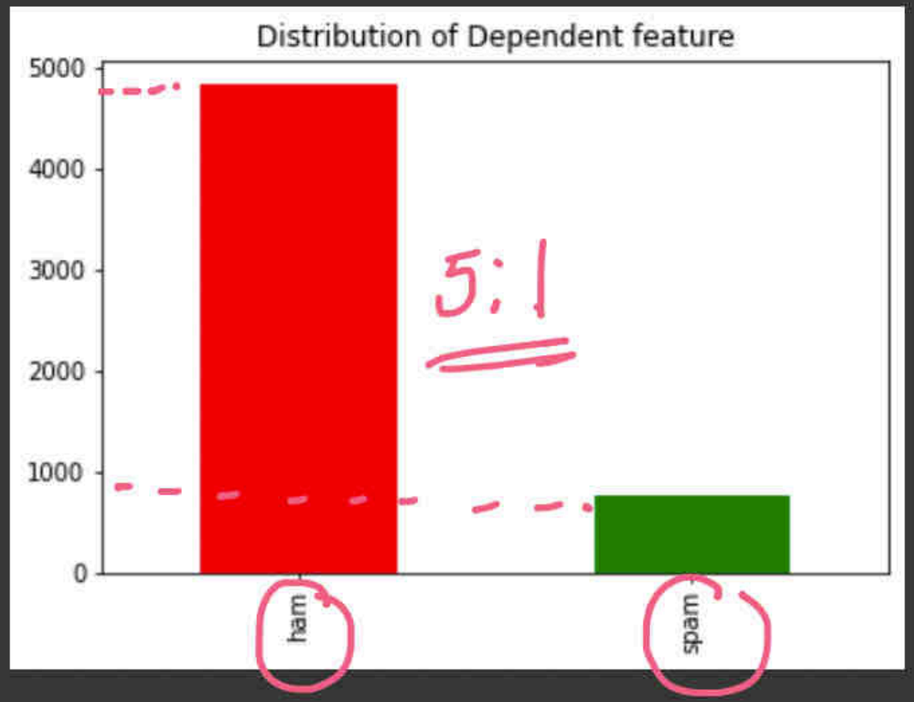
SVM_NB.ipynb - Colaboratory | sklearn.preprocessing.Standa | sklearn.svm.SVC — scikit-lear | sklearn.naive_bayes.Bernoulli | sklearn.naive_bayes.Multinom | 1.9. Naive Bayes — scikit-learn | dsml-course/SVM Business Ca

colab.research.google.com/drive/1_V0nqZq54rk_XVtlC_j1ez7vPaKDaZVM#scrollTo=XJPNgzCaFvX_

+ Code + Text Reconnect

4	ham	Nah I don't think he goes to usf, he lives aro...

```
freq = pd.value_counts(df["type"], sort= True)
freq.plot(kind= 'bar', color= ["red", "green"])
plt.title('Distribution of Dependent feature')
plt.show()
```



imbalanced data

63 / 63

Text Cleaning & Preprocessing

w_1 w_2 w_3

```
[ ] def clean_tokenized_sentence(s):
        """Performs basic cleaning of a tokenized sentence"""
        cleaned_s = ""  # Create empty string to store processed sentence.
        words = nltk.word_tokenize(s)
        for word in words:
            # Convert to lowercase #
            c_word = word.lower()
            # Remove punctuations #
            c_word = re.sub(r'[^\w\s]', '', c_word)
            # Remove stopwords #
            if c_word != '' and c_word not in stopwords.words('english'):
                cleaned_s = cleaned_s + " " + c_word    # Append processed words to new list.
        return(cleaned_s.strip())
```

```
df["cleaned_message"] = df["message"].apply(clean_tokenized_sentence)
df.head(10)
```

	type	message	cleaned_message
0	ham	Go until jurong ... great ...	

Text Cleaning & Preprocessing

```
[ ] def clean_tokenized_sentence(s):
        """Performs basic cleaning of a tokenized sentence"""
        cleaned_s = ""  # Create empty string to store processed sentence.
        words = nltk.word_tokenize(s)
        for word in words:
            # Convert to lowercase #
            c_word = word.lower()
            # Remove punctuations #
            c_word = re.sub(r'[^\w\s]', '', c_word)
            # Remove stopwords #
            if c_word != '' and c_word not in stopwords.words('english'):
                cleaned_s = cleaned_s + " " + c_word    # Append processed words to new list.
        return(cleaned_s.strip())
```

```
df["cleaned_message"] = df["message"].apply(clean_tokenized_sentence)
df.head(10)
```

	type	message	cleaned_message
0	ham	Go until jurong ... great ...	

Text Cleaning & Preprocessing

```
[ ]  def clean_tokenized_sentence(s):
         """Performs basic cleaning of a tokenized sentence"""
         cleaned_s = ""  # Create empty string to store processed sentence.
         words = nltk.word_tokenize(s)
         for word in words:
             # Convert to lowercase #
             c_word = word.lower()
             # Remove punctuations #
             c_word = re.sub(r'[^\w\s]', '', c_word)
             # Remove stopwords #
             if c_word != '' and c_word not in stopwords.words('english'):
                 cleaned_s = cleaned_s + " " + c_word    # Append processed words to new list.
         return(cleaned_s.strip())
```

```
df["cleaned_message"] = df["message"].apply(clean_tokenized_sentence)
df.head(10)
```

	type	message	cleaned_message
0	ham	Go until jurong ...	... great ...

Text Cleaning & Preprocessing

```
[ ] def clean_tokenized_sentence(s):
        """Performs basic cleaning of a tokenized sentence"""
        cleaned_s = ""  # Create empty string to store processed sentence.
        words = nltk.word_tokenize(s)
        for word in words:
            # Convert to lowercase #
            c_word = word.lower()
            # Remove punctuations #
            c_word = re.sub(r'[^\w\s]', '', c_word)
            # Remove stopwords #
            if c_word != '' and c_word not in stopwords.words('english'):
                cleaned_s = cleaned_s + " " + c_word    # Append processed words to new list.
        return(cleaned_s.strip())
```

```
df["cleaned_message"] = df["message"].apply(clean_tokenized_sentence)
df.head(10)
```

	type	message	cleaned_message
0	ham	Go until jurong ...	... great ...

```
[ ]  def clean_tokenized_sentence(s):
         """Performs basic cleaning of a tokenized sentence"""
         cleaned_s = ""  # Create empty string to store processed sentence.
         words = nltk.word_tokenize(s)
         for word in words:
             # Convert to lowercase #
             c_word = word.lower()
             # Remove punctuations #
             c_word = re.sub(r'[^\w\s]', '', c_word)
             # Remove stopwords #
             if c_word != '' and c_word not in stopwords.words('english'):
                 cleaned_s = cleaned_s + " " + c_word    # Append processed words to new list.
         return(cleaned_s.strip())
```

```
df["cleaned_message"] = df["message"].apply(clean_tokenized_sentence)
df.head(10)
```

	type	message	cleaned_message
0	ham	Go until jurong point, crazy.. Available only ...	go jurong point crazy available bugis n great ...
1	ham	Ok lar... Joking wif u oni...	ok lar joking wif u oni
2	spam	Free entry in 2 a w	al tkts 2...

```
[ ]  def clean_tokenized_sentence(s):
         """Performs basic cleaning of a tokenized sentence"""
         cleaned_s = ""  # Create empty string to store processed sentence.
         words = nltk.word_tokenize(s)
         for word in words:
             # Convert to lowercase #
             c_word = word.lower()
             # Remove punctuations #
             c_word = re.sub(r'[^\w\s]', '', c_word)
             # Remove stopwords #
             if c_word != '' and c_word not in stopwords.words('english'):
                 cleaned_s = cleaned_s + " " + c_word    # Append processed words to new list.
         return(cleaned_s.strip())
```

```
df["cleaned_message"] = df["message"].apply(clean_tokenized_sentence)
df.head(10)
```

	type	message	cleaned_message
0	ham	Go until jurong point, crazy.. Available only ...	go jurong point crazy available bugis n great ...
1	ham	Ok lar... Joking wif u oni...	ok lar joking wif u oni
2	spam	Free entry in 2 a w...	...al tkts 2...

69 / 69

```
[ ]  def clean_tokenized_sentence(s):
         """Performs basic cleaning of a tokenized sentence"""
         cleaned_s = ""  # Create empty string to store processed sentence.
         words = nltk.word_tokenize(s)
         for word in words:
             # Convert to lowercase #
             c_word = word.lower()
             # Remove punctuations #
             c_word = re.sub(r'[^\w\s]', '', c_word)
             # Remove stopwords #
             if c_word != '' and c_word not in stopwords.words('english'):
                 cleaned_s = cleaned_s + " " + c_word    # Append processed words to new list.
         return(cleaned_s.strip())
```

Go go

```
df["cleaned_message"] = df["message"].apply(clean_tokenized_sentence)
df.head(10)
```

	type	message	cleaned_message
0	ham	Go until jurong point, crazy.. Available only ...	go jurong point crazy available bugis n great ...
1	ham	Ok lar... Joking wif u oni...	ok lar joking wif u oni
2	spam	Free entry in 2 a w...	...al tkts 2...

```
[ ]  def clean_tokenized_sentence(s):
         """Performs basic cleaning of a tokenized sentence"""
         cleaned_s = ""  # Create empty string to store processed sentence.
         words = nltk.word_tokenize(s)
         for word in words:
             # Convert to lowercase #
             c_word = word.lower()
             # Remove punctuations #
             c_word = re.sub(r'[^\w\s]', '', c_word)
             # Remove stopwords #
             if c_word != '' and c_word not in stopwords.words('english'):
                 cleaned_s = cleaned_s + " " + c_word     # Append processed words to new list.
         return(cleaned_s.strip())
```

{ is the a of for an until } ;

```
df["cleaned_message"] = df["message"].apply(clean_tokenized_sentence)
df.head(10)
```

	type	message	cleaned_message
0	ham	Go until jurong point, crazy.. Available only ...	go jurong point crazy available bugis n great ...
1	ham	Ok lar... Joking wif u oni...	ok lar joking wif u oni
2	spam	Free entry in 2 a w...	...al tkts 2...

```python
# Importing libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from collections import Counter
from sklearn import feature_extraction, model_selection, naive_bayes, metrics, svm
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC

# Libraries for text processing
import re, nltk
nltk.download('punkt')
nltk.download('stopwords')
from nltk import word_tokenize, sent_tokenize
from nltk.corpus import stopwords

import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading pa
```

```
[ ] def clean_tokenized_sentence(s):
      """Performs basic cleaning of a tokenized sentence"""
      cleaned_s = ""  # Create empty string to store processed sentence.
      words = nltk.word_tokenize(s)
      for word in words:
        # Convert to lowercase #
        c_word = word.lower()
        # Remove punctuations #
        c_word = re.sub(r'[^\w\s]', '', c_word)
        # Remove stopwords #
        if c_word != '' and c_word not in stopwords.words('english'):
            cleaned_s = cleaned_s + " " + c_word    # Append processed words to new list.
      return(cleaned_s.strip())
```

```
df["cleaned_message"] = df["message"].apply(clean_tokenized_sentence)
df.head(10)
```

	type	message	cleaned_message
0	ham	Go until jurong point, crazy.. Available only ...	go jurong point crazy available bugis n great ...
1	ham	Ok lar... Joking wif u oni...	ok lar joking wif u oni
2	spam	Free entry in 2 a w...	...al tkts 2...

```
[ ] def clean_tokenized_sentence(s):
    """Performs basic cleaning of a tokenized sentence"""
    cleaned_s = ""  # Create empty string to store processed sentence.
    words = nltk.word_tokenize(s)
    for word in words:
        # Convert to lowercase #
        c_word = word.lower()
        # Remove punctuations #
        c_word = re.sub(r'[^\w\s]', '', c_word)
        # Remove stopwords #
        if c_word != '' and c_word not in stopwords.words('english'):
            cleaned_s = cleaned_s + " " + c_word    # Append processed words to new list.
    return(cleaned_s.strip())
```

```
df["cleaned_message"] = df["message"].apply(clean_tokenized_sentence)
df.head(10)
```

	type	message	cleaned_message
0	ham	Go until jurong point, crazy.. Available only ...	go jurong point crazy available bugis n great ...
1	ham	Ok lar... Joking wif u oni...	ok lar joking wif u oni
2	spam	Free entry in 2 a w	al tkts 2...

```
    for word in words:
        # Convert to lowercase #
        c_word = word.lower()
        # Remove punctuations #
        c_word = re.sub(r'[^\w\s]', '', c_word)
        # Remove stopwords #
        if c_word != '' and c_word not in stopwords.words('english'):
            cleaned_s = cleaned_s + " " + c_word    # Append processed words to new list.
    return(cleaned_s.strip())
```

```
df["cleaned_message"] = df["message"].apply(clean_tokenized_sentence)
df.head(10)
```

	type	message	cleaned_message
0	ham	Go until jurong point, crazy.. Available only ...	go jurong point crazy available bugis n great ...
1	ham	Ok lar... Joking wif u oni...	ok lar joking wif u oni
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	free entry 2 wkly comp win fa cup final tkts 2...
3	ham	U dun say so early hor... U c already then say...	u dun say early hor u c already say
4	ham	Nah I don't think he goes to usf, he lives aro...	nah nt think goes usf lives around though
5	spam	FreeMsg Hey there darling it's been 3 week's n...	freemsg hey darling 3 week word back like fun ...

```
[ ]         c_word = re.sub(r'[^\w\s]', '', c_word)
            # Remove stopwords #
            if c_word != '' and c_word not in stopwords.words('english'):
                cleaned_s = cleaned_s + " " + c_word    # Append processed words to new list.
        return(cleaned_s.strip())
```

```
[ ] df["cleaned_message"] = df["message"].apply(clean_tokenized_sentence)
    df.head(10)
```

	type	message	cleaned_message
0	ham	Go until jurong point, crazy.. Available only ...	go jurong point crazy available bugis n great ...
1	ham	Ok lar... Joking wif u oni...	ok lar joking wif u oni
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	free entry 2 wkly comp win fa cup final tkts 2...
3	ham	U dun say so early hor... U c already then say...	u dun say early hor u c already say
4	ham	Nah I don't think he goes to usf, he lives aro...	nah nt think goes usf lives around though
5	spam	FreeMsg Hey there darling it's been 3 week's n...	freemsg hey darling 3 week word back like fun ...
6	ham	Even my brother is not like to speak with me. ...	even brother like speak treat like aids patent
7	ham	As per your request 'Melle Melle (Oru Minnamin...	per request melle melle oru minnaminunginte nu...
8	spam	WINNER!! As a valued network customer you have	winner valued network customer selected receiv...

Most common words in Spam and Ham

```
[ ]  Counter(" ".join(df[df['type']=='ham']["cleaned_message"]).split())
```

```
'luv': 30,
'kettoda': 4,
'manda': 4,
'ups': 1,
'3days': 1,
'shipping': 3,
'takes': 12,
'2wks': 1,
'usps': 1,
'lag': 2,
'bribe': 1,
'nipost': 1,
'lem': 8,
'necessarily': 2,
'expect': 4,
'headin': 2,
'mmm': 4,
'jolt': 2,
'suzy': 1,
```

Most common words in Spam and Ham

```
[ ] Counter(" ".join(df[df['type']=='ham']["cleaned_message"]).split())
```

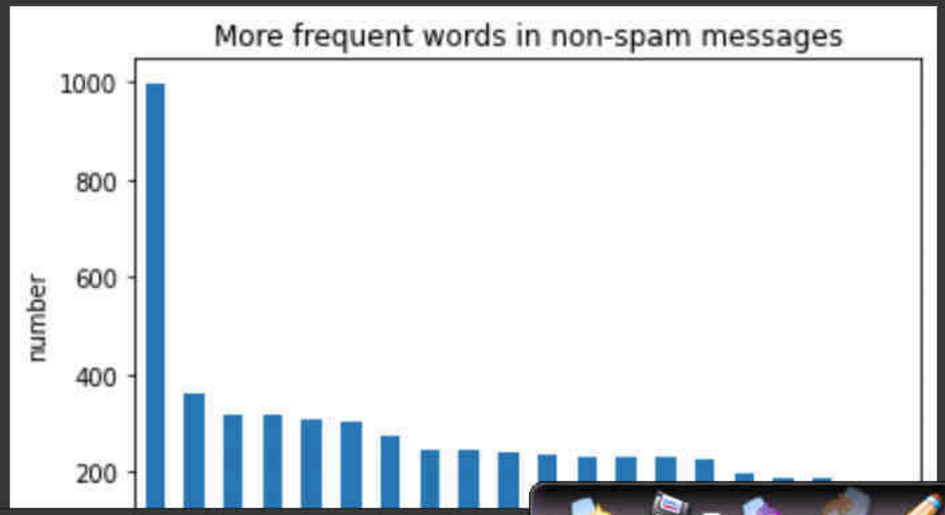
```
'luv': 30,
'kettoda': 4,
'manda': 4,
'ups': 1,
'3days': 1,
'shipping': 3,
'takes': 12,
'2wks': 1,
'usps': 1,
'lag': 2,
'bribe': 1,
'nipost': 1,
'lem': 8,
'necessarily': 2,
'expect': 4,
'headin': 2,
'mmm': 4,
'jolt': 2,
'suzy': 1,
```

→ non-spam msgs
- Cleaned-message
- splitting →

w_1: 100
w_2: 20
w_3: 26
⋮

```
'b': 56,
```

```
counter_ham = Counter(" ".join(df[df['type']=='ham']["cleaned_message"]).split()).most_common(20)
df_ham = pd.DataFrame.from_dict(counter_ham)
df_ham = df_ham.rename(columns={0:"words in non-spam", 1:"count"})

df_ham.plot.bar(legend = False)
y_pos = np.arange(len(df_ham["words in non-spam"]))
plt.xticks(y_pos, df_ham["words in non-spam"])
plt.title('More frequent words in non-spam messages')
plt.xlabel('words')
plt.ylabel('number')
plt.show()
```



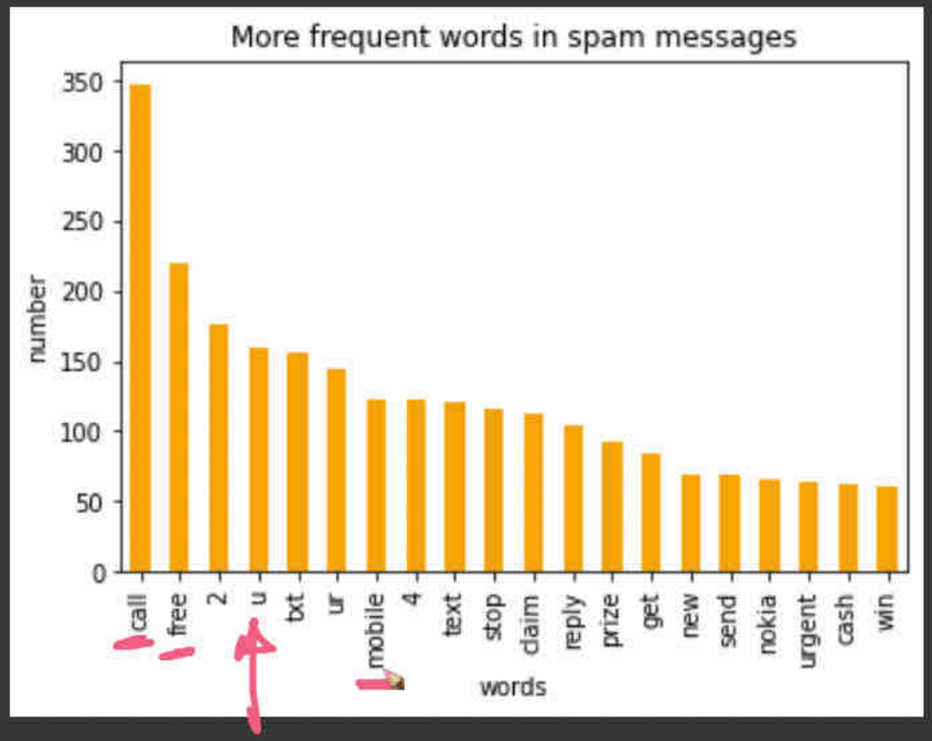
```
plt.ylabel('number')
plt.show()
```

EDA

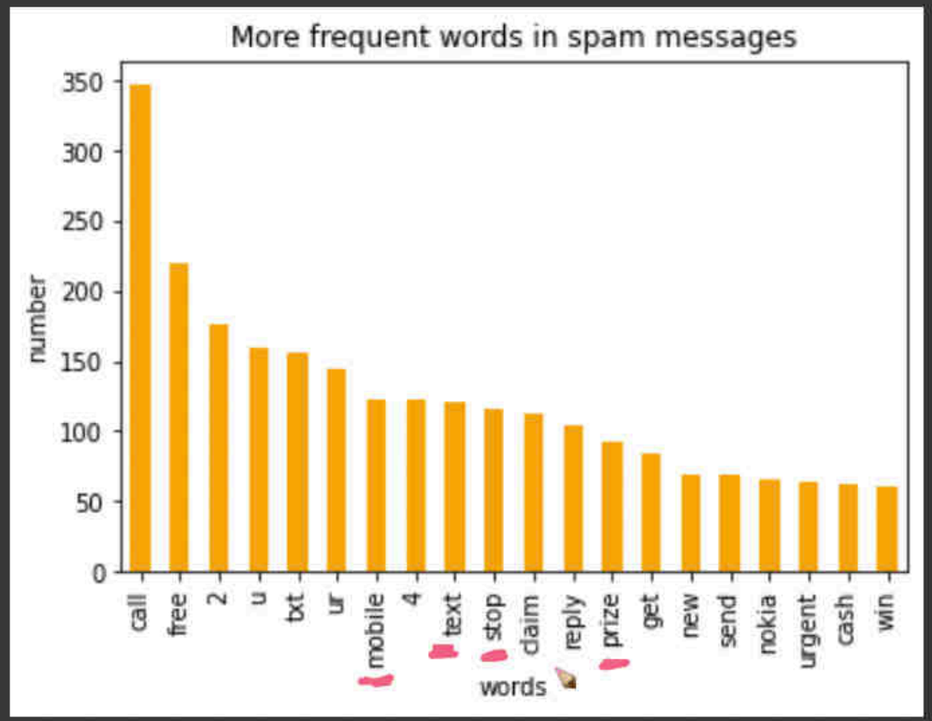
```
counter_spam = Counter(" ".join(df[df['type']=='spam']["cleaned_message"]).split()).most_common(20)
df_spam = pd.DataFrame.from_dict(counter_spam)
df_spam = df_spam.rename(columns={0:"words in spam", 1:"count_"})

df_spam.plot.bar(legend = False, color = 'orange')
y_pos = np.arange(len(df_s
```

80 / 80

```
[ ]  df_spam.plot.bar(legend = False, color = 'orange')
     y_pos = np.arange(len(df_spam["words in spam"]))
     plt.xticks(y_pos, df_spam["words in spam"])
     plt.title('More frequent words in spam messages')
     plt.xlabel('words')
     plt.ylabel('number')
     plt.show()
```



```
df_spam.plot.bar(legend = False, color = 'orange')
y_pos = np.arange(len(df_spam["words in spam"]))
plt.xticks(y_pos, df_spam["words in spam"])
plt.title('More frequent words in spam messages')
plt.xlabel('words')
plt.ylabel('number')
plt.show()
```



[ ]

Generate Datasets

```
df["type"] = df["type"].map({'spam':1,'ham':0})
```

```
[ ] df_X_train, df_X_test, y_train, y_test = train_test_split(df['cleaned_message'], df['type'], test_size=0.25, random
    print([np.shape(df_X_train), np.shape(df_X_test)])
```

```
[(4179,), (1393,)]
```

```
[ ]
    #Count Vectorizer
    f = feature_extraction.text.CountVectorizer()

    X_train = f.fit_transform(df_X_train)
    X_test = f.transform(df_X_test)
```

[ ]

Generate Datasets

K-fold CV

75% Train

25% Test

```
df["type"] = df["type"].map({'spam':1,'ham':0})
```

[ ]
```
df_X_train, df_X_test, y_train, y_test = train_test_split(df['cleaned_message'], df['type'], test_size=0.25, random
print([np.shape(df_X_train), np.shape(df_X_test)])
```

```
[(4179,), (1393,)]
```

[ ]
```
#Count Vectorizer
f = feature_extraction.text.CountVectorizer()

X_train = f.fit_transform(df_X_train)
X_test = f.transform(df_X_test)
```

[ ]

Generate Datasets

```
df["type"] = df["type"].map({'spam':1,'ham':0})
```

[ ]
```
df_X_train, df_X_test, y_train, y_test = train_test_split(df['cleaned_message'], df['type'], test_size=0.25, random
print([np.shape(df_X_train), np.shape(df_X_test)])
```

```
[(4179,), (1393,)]
```

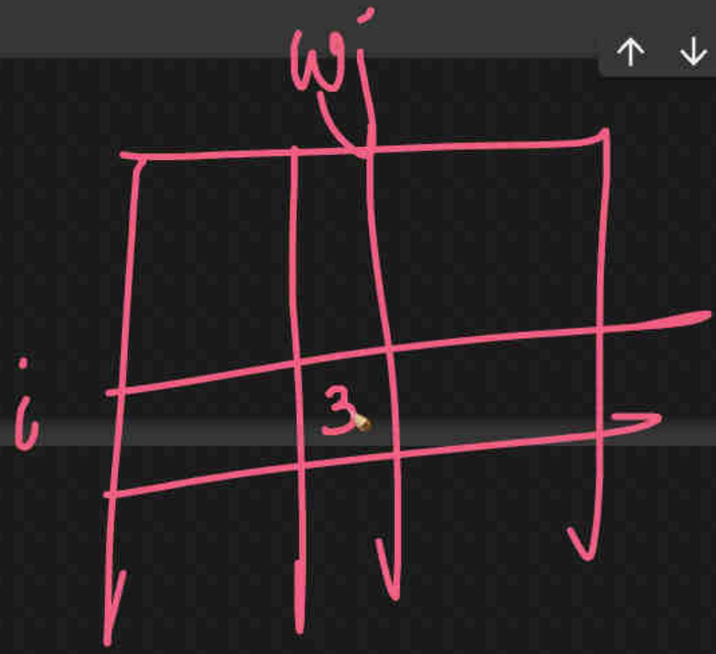
[ ]
```
#Count Vectorizer
f = feature_extraction.text.CountVectorizer()

X_train = f.fit_transform(df_X_train)
X_test = f.transform(df_X_test)
```

```
[ ] df["type"] = df["type"].map({'spam':1,'ham':0})
```

```
[ ] df_X_train, df_X_test, y_train, y_test = train_test_split(df['cleaned_message'], df['type'], test_size=0.25, random
    print([np.shape(df_X_train), np.shape(df_X_test)])
```

```
[(4179,), (1393,)]
```

```
#Count Vectorizer
f = feature_extraction.text.CountVectorizer()

X_train = f.fit_transform(df_X_train)
X_test = f.transform(df_X_test)
```



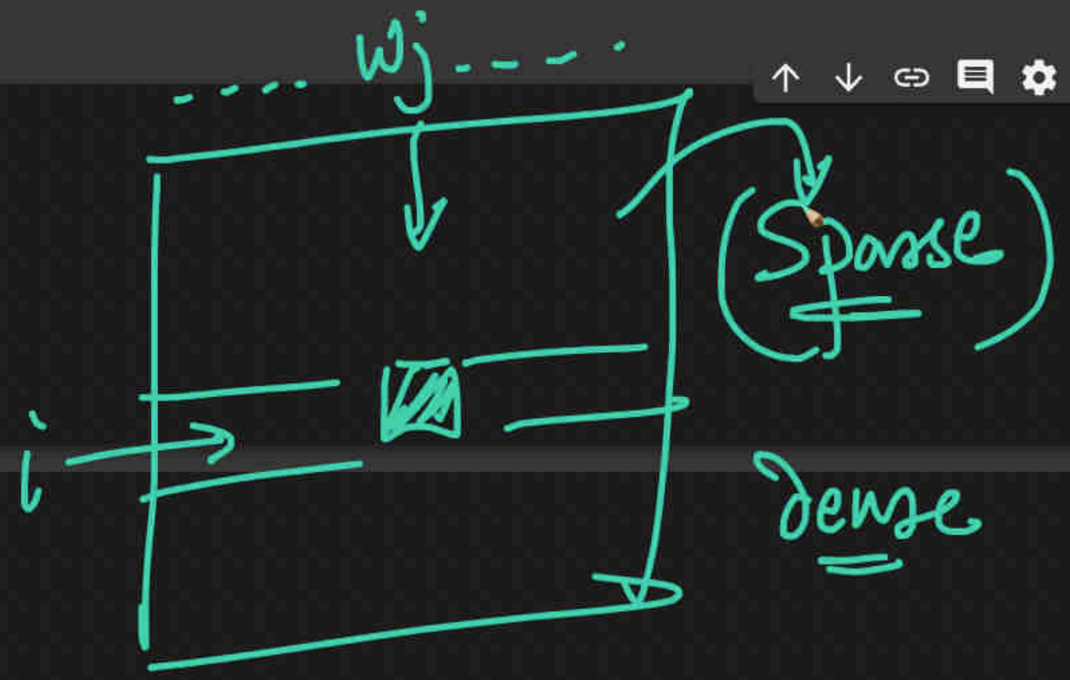

```
# Standard Scaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-46-ae0768e59d91> in <module>()
```

```
[ ] df["type"] = df["type"].map({'spam':1,'ham':0})
```

```
[ ] df_X_train, df_X_test, y_train, y_test = train_test_split(df['cleaned_message'], df['type'], test_size=0.25, random
    print([np.shape(df_X_train), np.shape(df_X_test)])
```

```
[(4179,), (1393,)]
```




```
#Count Vectorizer
f = feature_extraction.text.CountVectorizer()

X_train = f.fit_transform(df_X_train)
X_test = f.transform(df_X_test)
```

```
# Standard Scaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-46-ae0768e59d91> in <module>()
```

```
[ ] df["type"] = df["type"].map({'spam':1,'ham':0})
```

```
[ ] df_X_train, df_X_test, y_train, y_test = train_test_split(df['cleaned_message'], df['type'], test_size=0.25, random
    print([np.shape(df_X_train), np.shape(df_X_test)])
```

```
[(4179,), (1393,)]
```

```
#Count Vectorizer
f = feature_extraction.text.CountVectorizer()

X_train = f.fit_transform(df_X_train)
X_test = f.transform(df_X_test)
```

```
# Standard Scaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-46-ae0768e59d91> in <module>()
```

```
[ ] df["type"] = df["type"].map({'spam':1,'ham':0})
```

```
[ ] df_X_train, df_X_test, y_train, y_test = train_test_split(df['cleaned_message'], df['type'], test_size=0.25, random
    print([np.shape(df_X_train), np.shape(df_X_test)])
```

```
[(4179,), (1393,)]
```

```
#Count Vectorizer
f = feature_extraction.text.CountVectorizer()

X_train = f.fit_transform(df_X_train)
X_test = f.transform(df_X_test)
```

```
# Standard Scaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-46-ae0768e59d91> in <module>()
```

```
[ ] df["type"] = df["type"].map({'spam':1,'ham':0})
```

```
[ ] df_X_train, df_X_test, y_train, y_test = train_test_split(df['cleaned_message'], df['type'], test_size=0.25, random
    print([np.shape(df_X_train), np.shape(df_X_test)])
```

```
[(4179,), (1393,)]
```

```
#Count Vectorizer
f = feature_extraction.text.CountVectorizer()

X_train = f.fit_transform(df_X_train)
X_test = f.transform(df_X_test)
```

```
# Standard Scaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-46-ae0768e59d91> in <module>()
```

```
[ ] df["type"] = df["type"].map({'spam':1,'ham':0})
```

```
[ ] df_X_train, df_X_test, y_train, y_test = train_test_split(df['cleaned_message'], df['type'], test_size=0.25, random
    print([np.shape(df_X_train), np.shape(df_X_test)])
```

```
[(4179,), (1393,)]
```

```
#Count Vectorizer
f = feature_extraction.text.CountVectorizer()

X_train = f.fit_transform(df_X_train)
X_test = f.transform(df_X_test)
```

```
# Standard Scaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-46-ae0768e59d91> in <module>()
```

Chrome File Edit View History Bookmarks Profiles Tab Window Help 11:08

SVM_NB.ipynb - Colaboratory | sklearn.preprocessing.Standa | sklearn.svm.SVC — scikit-lear | sklearn.naive_bayes.Bernoulli | sklearn.naive_bayes.Multinom | 1.9. Naive Bayes — scikit-learn | dsml-course/SVM Business Ca

colab.research.google.com/drive/1_V0nqZq54rk_XVtlC_j1ez7vPaKDaZVM#scrollTo=o3yHdbvxyYcY

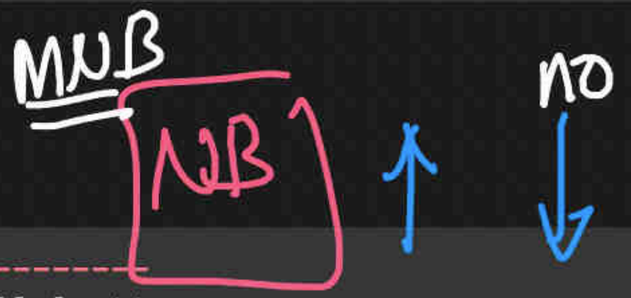
+ Code + Text Reconnect

```
[ ] print([np.shape(df_X_train), np.shape(df_X_test)])
```

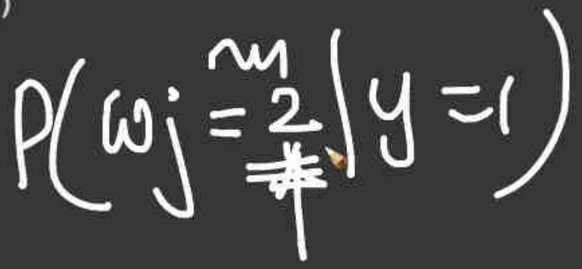
```
[(4179,), (1393,)]
```

```
#Count Vectorizer
f = feature_extraction.text.CountVectorizer()

X_train = f.fit_transform(df_X_train)
X_test = f.transform(df_X_test)
```

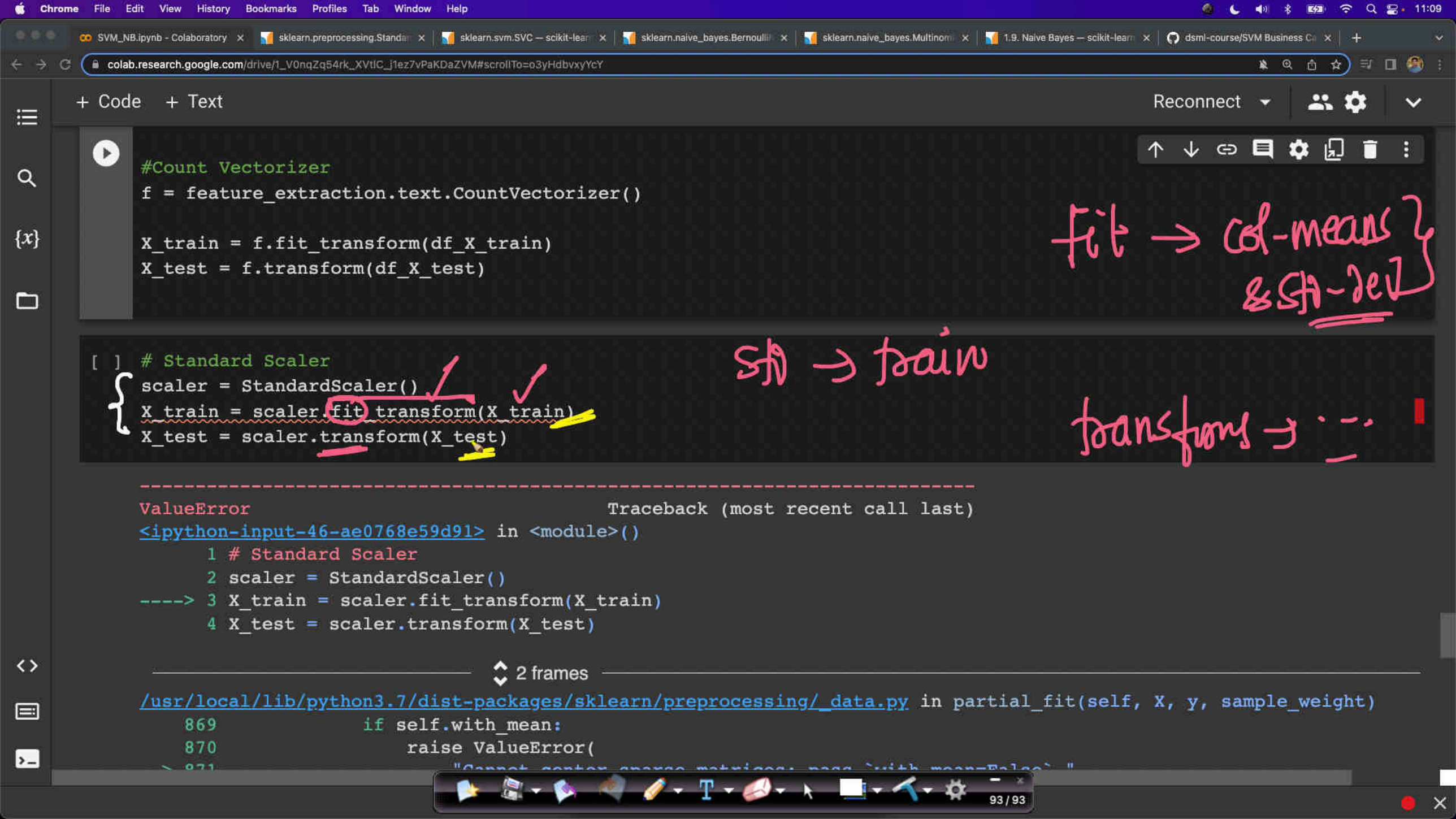

```
# Standard Scaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```




```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-46-ae0768e59d91> in <module>()
      1 # Standard Scaler
      2 scaler = StandardScaler()
----> 3 X_train = scaler.fit_transform(X_train)
      4 X_test = scaler.transform(X_test)
```



92 / 92


Chrome File Edit View History Bookmarks Profiles Tab Window Help
11:09
SVM_NB.ipynb - Colaboratory
sklearn.preprocessing.Standa
sklearn.svm.SVC — scikit-lear
sklearn.naive_bayes.Bernoulli
sklearn.naive_bayes.Multinom
1.9. Naive Bayes — scikit-learn
dsml-course/SVM Business Ca
colab.research.google.com/drive/1_V0nqZq54rk_XVtlC_j1ez7vPaKDaZVM#scrollTo=o3yHdbvxyYcY
+ Code
+ Text
Reconnect
#Count Vectorizer
f = feature_extraction.text.CountVectorizer()
X_train = f.fit_transform(df_X_train)
X_test = f.transform(df_X_test)
fit → col-means & std-dev
[ ]
# Standard Scaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
std → train
transform → ...
ValueError Traceback (most recent call last)
<ipython-input-46-ae0768e59d91> in <module>()
1 # Standard Scaler
2 scaler = StandardScaler()
----> 3 X_train = scaler.fit_transform(X_train)
4 X_test = scaler.transform(X_test)
2 frames
/usr/local/lib/python3.7/dist-packages/sklearn/preprocessing/_data.py in partial_fit(self, X, y, sample_weight)
869 if self.with_mean:
870 raise ValueError(
93 / 93

```
[ ] X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-46-ae0768e59d91> in <module>()
      1 # Standard Scaler
      2 scaler = StandardScaler()
----> 3 X_train = scaler.fit_transform(X_train)
      4 X_test = scaler.transform(X_test)

                                  2 frames

/usr/local/lib/python3.7/dist-packages/sklearn/preprocessing/_data.py in partial_fit(self, X, y, sample_weight)
    869             if self.with_mean:
    870                 raise ValueError(
--> 871                     "Cannot center sparse matrices: pass `with_mean=False` "
    872                     "instead. See docstring for motivation and alternatives."
    873                 )

ValueError: Cannot center sparse matrices: pass `with_mean=False` instead. See docstring for motivation and
alternatives.
```

SEARCH STACK OVERFLOW

```
[ ] # Standard Scaler
    scaler = StandardScaler(with_mean=False) # Problems with dense matrix
```

```
    870                 raise ValueError(
--> 871                     "Cannot center sparse matrices: pass `with_mean=False` "
    872                     "instead. See docstring for motivation and alternatives."
    873                 )

ValueError: Cannot center sparse matrices: pass `with_mean=False` instead. See docstring for motivation and alternatives.
```

SEARCH STACK OVERFLOW

```
# Standard Scaler
scaler = StandardScaler(with_mean=False) # Problems with dense matrix
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test.todense())
```

```
print([np.shape(X_train), np.shape(X_test)])
```

```
[(4179, 7692), (1393, 7692)]
```

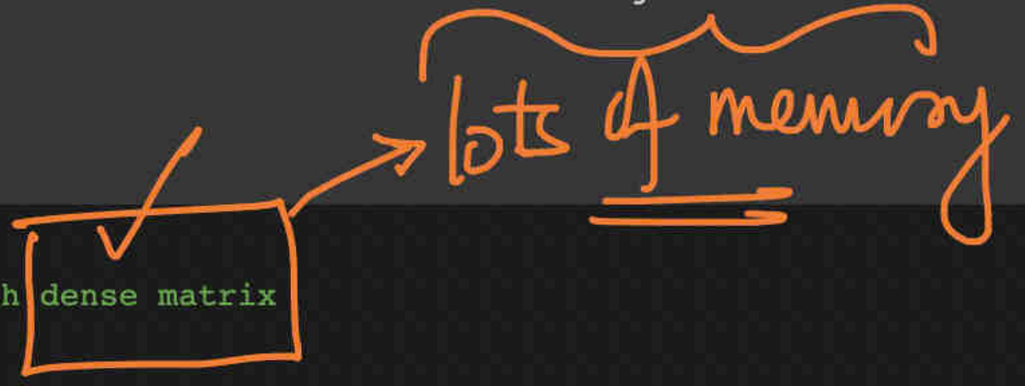
```
type(X_train)
```

```
scipy.sparse.csr.csr_matrix
```

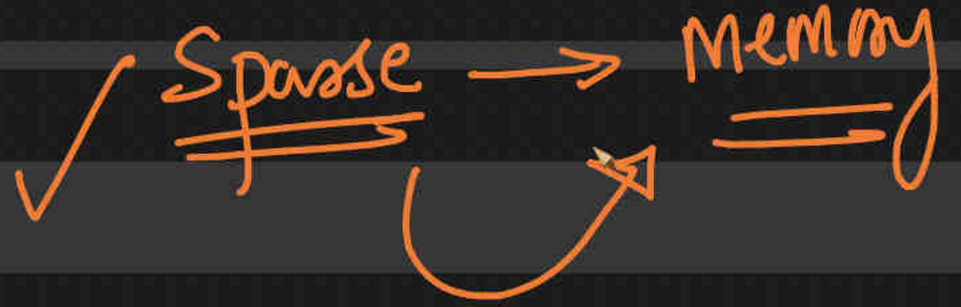
```
[ ]      870                 raise ValueError(
--->     871                     "Cannot center sparse matrices: pass `with_mean=False` "
         872                     "instead. See docstring for motivation and alternatives."
         873                 )

ValueError: Cannot center sparse matrices: pass `with_mean=False` instead. See docstring for motivation and
alternatives.
```

SEARCH STACK OVERFLOW

```
[ ] # Standard Scaler
    scaler = StandardScaler(with_mean=False) # Problems with dense matrix
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test.todense())
```




```
[ ] print([np.shape(X_train), np.shape(X_test)])

[(4179, 7692), (1393, 7692)]
```




```
[ ] type(X_train)

scipy.sparse.csr.csr_matrix
```

Chrome File Edit View History Bookmarks Profiles Tab Window Help 11:12

SVM_NB.ipynb - Colaboratory | sklearn.preprocessing.Standa | sklearn.svm.SVC — scikit-lear | sklearn.naive_bayes.Bernoulli | sklearn.naive_bayes.Multinom | 1.9. Naive Bayes — scikit-lear | dsml-course/SVM Business Ca

colab.research.google.com/drive/1_V0nqZq54rk_XVtlC_j1ez7vPaKDaZVM#scrollTo=o3yHdbvxyYcY

+ Code + Text Reconnect

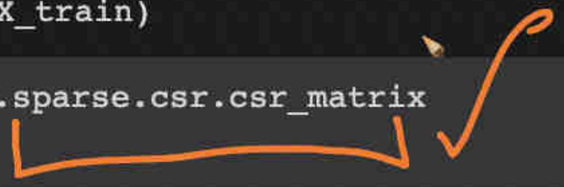
```
[ ]
      870                 raise ValueError(
--> 871                     "Cannot center sparse matrices: pass `with_mean=False` "
      872                     "instead. See docstring for motivation and alternatives."
      873                 )

ValueError: Cannot center sparse matrices: pass `with_mean=False` instead. See docstring for motivation and alternatives.
```

SEARCH STACK OVERFLOW

```
[ ] # Standard Scaler
    scaler = StandardScaler(with_mean=False) # Problems with dense matrix
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test.todense())
```

```
[ ] print([np.shape(X_train), np.shape(X_test)])

[(4179, 7692), (1393, 7692)]
```

```
[ ] type(X_train)

scipy.sparse.csr.csr_matrix
```



97 / 97

Chrome File Edit View History Bookmarks Profiles Tab Window Help 11:13

SVM_NB.ipynb - Colaboratory | sklearn.preprocessing.Standa | sklearn.svm.SVC — scikit-lear | sklearn.naive_bayes.Bernoulli | sklearn.naive_bayes.Multinom | 1.9. Naive Bayes — scikit-lear | dsml-course/SVM Business Ca

colab.research.google.com/drive/1_V0nqZq54rk_XVtlC_j1ez7vPaKDaZVM#scrollTo=o3yHdbvxyYcY

+ Code + Text Reconnect

```
[ ]
      870                 raise ValueError(
--> 871                     "Cannot center sparse matrices: pass `with_mean=False` "
      872                     "instead. See docstring for motivation and alternatives."
      873                 )

ValueError: Cannot center sparse matrices: pass `with_mean=False` instead. See docstring for motivation and alternatives.
```

SEARCH STACK OVERFLOW

```
[ ] # Standard Scaler
    scaler = StandardScaler(with_mean=False) # Problems with dense matrix
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test.todense())
```

```
[ ] print([np.shape(X_train), np.shape(X_test)])

    [(4179, 7692), (1393, 7692)]
```

```
[ ] type(X_train)

    scipy.sparse.csr.csr_matrix
```

BernoulliNB
MultinomialNB
noscaling
scaling/noscaling
GaussianNB
fi N(

98 / 98

Chrome File Edit View History Bookmarks Profiles Tab Window Help 11:13

SVM_NB.ipynb - Colaboratory | sklearn.preprocessing.Standa | sklearn.svm.SVC — scikit-lear | sklearn.naive_bayes.Bernoulli | sklearn.naive_bayes.Multinom | 1.9. Naive Bayes — scikit-lear | dsml-course/SVM Business Ca

colab.research.google.com/drive/1_V0nqZq54rk_XVtlC_j1ez7vPaKDaZVM#scrollTo=o3yHdbvxyYcY

+ Code + Text Reconnect

```
[ ]
    870                 raise ValueError(
--> 871                     "Cannot center sparse matrices: pass `with_mean=False` "
    872                     "instead. See docstring for motivation and alternatives."
    873                 )

ValueError: Cannot center sparse matrices: pass `with_mean=False` instead. See docstring for motivation and alternatives.
```

SEARCH STACK OVERFLOW

```
[ ] # Standard Scaler
    scaler = StandardScaler(with_mean=False) # Problems with dense matrix
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test.todense())
```

```
[ ] print([np.shape(X_train), np.shape(X_test)])

[(4179, 7692), (1393, 7692)]
```

```
[ ] type(X_train)

scipy.sparse.csr.csr_matrix
```

GNB:-

$f_j | y=1 \sim N(\mu_{j1}, \sigma_{j1})$

99 / 99

Chrome File Edit View History Bookmarks Profiles Tab Window Help 11:14

SVM_NB.ipynb - Colaboratory | sklearn.preprocessing.Standa | sklearn.svm.SVC — scikit-lear | sklearn.naive_bayes.Bernoulli | sklearn.naive_bayes.Multinom | 1.9. Naive Bayes — scikit-lear | dsml-course/SVM Business Ca

colab.research.google.com/drive/1_V0nqZq54rk_XVtlC_j1ez7vPaKDaZVM#scrollTo=o3yHdbvxyYcY

+ Code + Text Reconnect

```
[ ]  --> 871                    "Cannot center sparse matrices: pass `with_mean=False` "
         872                    "instead. See docstring for motivation and alternatives."
         873                )
```

```
ValueError: Cannot center sparse matrices: pass `with_mean=False` instead. See docstring for motivation and
alternatives.
```

SEARCH STACK OVERFLOW

```
[ ] # Standard Scaler
    scaler = StandardScaler(with_mean=False) # Problems with dense matrix
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test.todense())
```

```
[ ] print([np.shape(X_train), np.shape(X_test)])
```

```
[(4179, 7692), (1393, 7692)]
```

```
[ ] type(X_train)
```

```
scipy.sparse.csr.csr_matrix
```

Linear SVM Models

100 / 100

```
scipy.sparse.csr.csr_matrix
```

Linear SVM Models

$$\min_{w,b} \frac{1}{2}\|w\| + C\sum_{i=1}^{n}\xi_i$$

```
[ ]  # SVC

     from sklearn.model_selection import GridSearchCV

     params = {
             'C': [1e-4,  0.001, 0.01, 0.1, 1,10] # which hyper-param vlaue of C do you think will work well?
             }
     svc = SVC(class_weight={ 0:0.1, 1:0.5 }, kernel='linear')
     clf = GridSearchCV(svc, params, scoring = "f1", cv=3)

     clf.fit(X_train, y_train)
```

```
GridSearchCV(cv=3,
             estimator=SVC(class_weight={0: 0.1, 1: 0.5}, kernel='linear'),
             param_grid={'C': [0.0001, 0.001, 0.01, 0.1, 1, 10]}, scoring='f1')
```

```
[ ]  = clf.cv_results_
```

```
scipy.sparse.csr.csr_matrix
```

Linear SVM Models

```
[ ] # SVC

from sklearn.model_selection import GridSearchCV

params = {
      'C': [1e-4,  0.001, 0.01, 0.1, 1,10] # which hyper-param vlaue of C do you think will work well?
      }
svc = SVC(class_weight={ 0:0.1, 1:0.5 }, kernel='linear')
clf = GridSearchCV(svc, params, scoring = "f1", cv=3)

clf.fit(X_train, y_train)
```

```
GridSearchCV(cv=3,
             estimator=SVC(class_weight={0: 0.1, 1: 0.5}, kernel='linear'),
             param_grid={'C': [0.0001, 0.001, 0.01, 0.1, 1, 10]}, scoring='f1')
```

```
[ ]  = clf.cv_results_
```

```
clf = GridSearchCV(svc, params, scoring = "f1", cv=3)

clf.fit(X_train, y_train)
```

```
GridSearchCV(cv=3,
             estimator=SVC(class_weight={0: 0.1, 1: 0.5}, kernel='linear'),
             param_grid={'C': [0.0001, 0.001, 0.01, 0.1, 1, 10]}, scoring='f1')
```

```
res = clf.cv_results_

for i in range(len(res["params"])):
  print(f"Parameters:{res['params'][i]} Mean_score: {res['mean_test_score'][i]} Rank: {res['rank_test_score'][i]}")
```

```
Parameters:{'C': 0.0001} Mean_score: 0.6497817092458725 Rank: 6
Parameters:{'C': 0.001} Mean_score: 0.7839609357761174 Rank: 1
Parameters:{'C': 0.01} Mean_score: 0.7814333814333815 Rank: 2
Parameters:{'C': 0.1} Mean_score: 0.7746267554214574 Rank: 3
Parameters:{'C': 1} Mean_score: 0.7746267554214574 Rank: 3
Parameters:{'C': 10} Mean_score: 0.7746267554214574 Rank: 3
```

0.7839

RBF SVM

1 cell hidden

```
            'C': [1e-4,  0.001, 0.01, 0.1, 1,10] # which hyper-param vlaue of C do you think will work well?
[ ]         }
    svc = SVC(class_weight={ 0:0.1, 1:0.5 }, kernel='linear')
    clf = GridSearchCV(svc, params, scoring = "f1", cv=3)

    clf.fit(X_train, y_train)
```

```
GridSearchCV(cv=3,
             estimator=SVC(class_weight={0: 0.1, 1: 0.5}, kernel='linear'),
             param_grid={'C': [0.0001, 0.001, 0.01, 0.1, 1, 10]}, scoring='f1')
```

```
res = clf.cv_results_

for i in range(len(res["params"])):
  print(f"Parameters:{res['params'][i]} Mean_score: {res['mean_test_score'][i]} Rank: {res['rank_test_score'][i]}")
```

```
Parameters:{'C': 0.0001} Mean_score: 0.6497817092458725 Rank: 6
Parameters:{'C': 0.001} Mean_score: 0.7839609357761174 Rank: 1
Parameters:{'C': 0.01} Mean_score: 0.7814333814333815 Rank: 2
Parameters:{'C': 0.1} Mean_score: 0.7746267554214574 Rank: 3
Parameters:{'C': 1} Mean_score: 0.7746267554214574 Rank: 3
Parameters:{'C': 10} Mean_score: 0.7746267554214574 Rank: 3
```

RBF SVM

```
            'C': [1e-4,  0.001, 0.01, 0.1, 1,10] # which hyper-param vlaue of C do you think will work well?
[ ]         }
    svc = SVC(class_weight={ 0:0.1, 1:0.5 }, kernel='linear')
    clf = GridSearchCV(svc, params, scoring = "f1", cv=3)

    clf.fit(X_train, y_train)
```

```
GridSearchCV(cv=3,
             estimator=SVC(class_weight={0: 0.1, 1: 0.5}, kernel='linear'),
             param_grid={'C': [0.0001, 0.001, 0.01, 0.1, 1, 10]}, scoring='f1')
```

```
res = clf.cv_results_

for i in range(len(res["params"])):
  print(f"Parameters:{res['params'][i]} Mean_score: {res['mean_test_score'][i]} Rank: {res['rank_test_score'][i]}")
```

```
Parameters:{'C': 0.0001} Mean_score: 0.6497817092458725 Rank: 6
Parameters:{'C': 0.001} Mean_score: 0.7839609357761174 Rank: 1
Parameters:{'C': 0.01} Mean_score: 0.7814333814333815 Rank: 2
Parameters:{'C': 0.1} Mean_score: 0.7746267554214574 Rank: 3
Parameters:{'C': 1} Mean_score: 0.7746267554214574 Rank: 3
Parameters:{'C': 10} Mean_score: 0.7746267554214574 Rank: 3
```

dim is high

RBF SVM

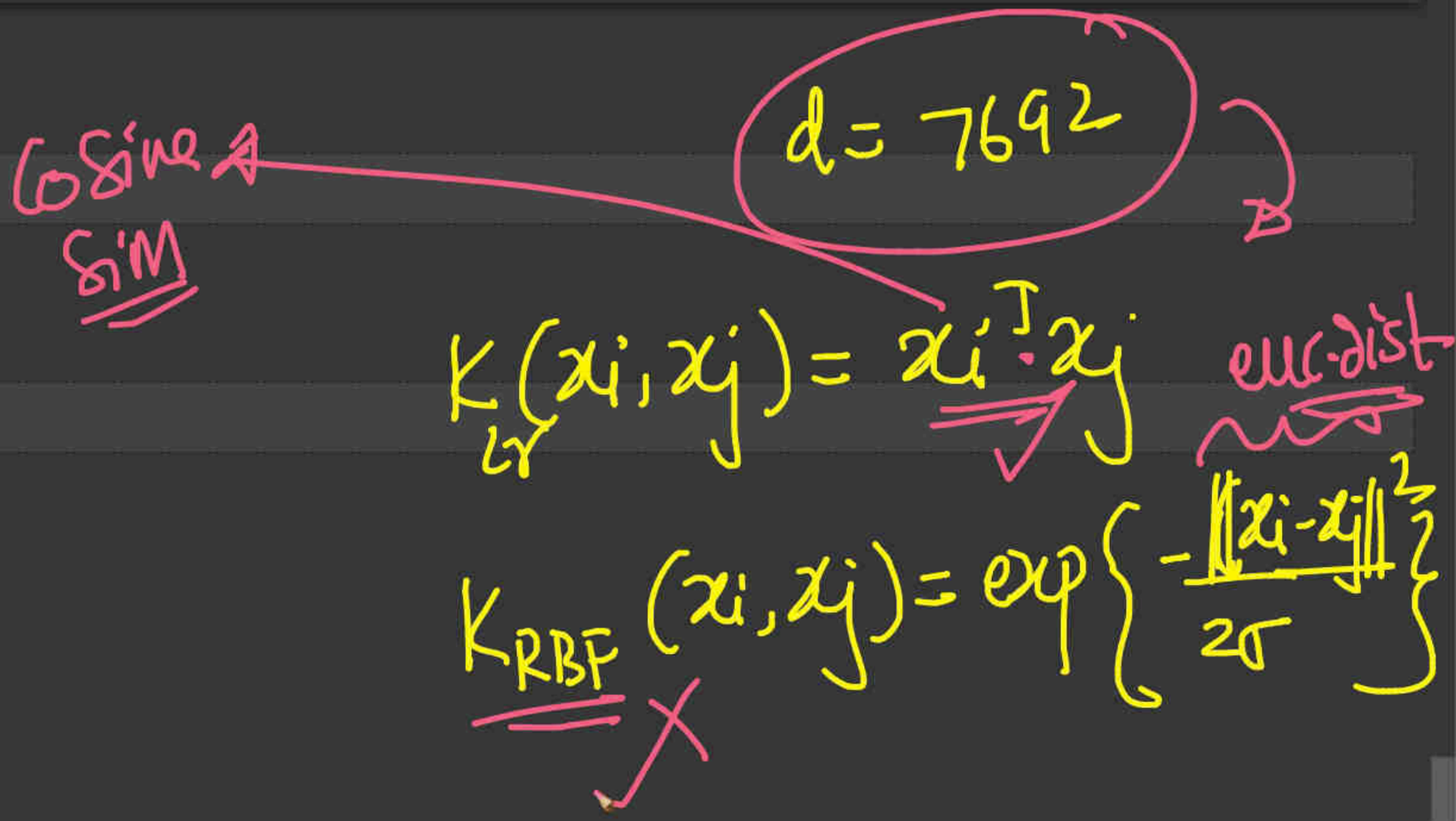
```
Parameters:{'C': 0.01} Mean_score: 0.7814333814333815 Rank: 2
Parameters:{'C': 0.1} Mean_score: 0.7746267554214574 Rank: 3
Parameters:{'C': 1} Mean_score: 0.7746267554214574 Rank: 3
Parameters:{'C': 10} Mean_score: 0.7746267554214574 Rank: 3
```

RBF SVM

[ ] ↳ 1 cell hidden

Multinomial NB

[ ] ↳ 1 cell hidden



```
[ ]  # RBF-SVC

     from sklearn.model_selection import GridSearchCV

     params = {
             'C': [0.1, 1,10],
             'gamma':[1e-3, 0.01, 0.1, 1]
             }
     svc = SVC(class_weight={ 0:0.1, 1:0.5 }, kernel='rbf')
     clf = GridSearchCV(svc, params, scoring = "f1", cv=3)

     clf.fit(X_train, y_train)

     res = clf.cv_results_

     for i in range(len(res["params"])):
       print(f"Parameters:{res['params'][i]} Mean_score: {res['mean_test_score'][i]} Rank: {res['rank_test_score'][i]}")

     # Euclidean distance in high dimensions in RBF calculation.
```

$\gamma = \frac{1}{\sigma}$

```
Parameters:{'C': 0.1, 'gamma': 0.001} Mean_score: 0.0 Rank: 9
Parameters:{'C': 0.1, 'gamma': 0.01} Mean_score: 0.0 Rank: 9
Parameters:{'C': 0.1, 'gamma': 0.1} Mean_score: 0.0 Rank: 9
Parameters:{'C': 0.1, 'gamma': 1} Mean_score: 0.0 Rank: 9
Parameters:{'C': 1, 'gamma': 0.001} Mean_score: 0.3400185328811167 Rank: 1
```

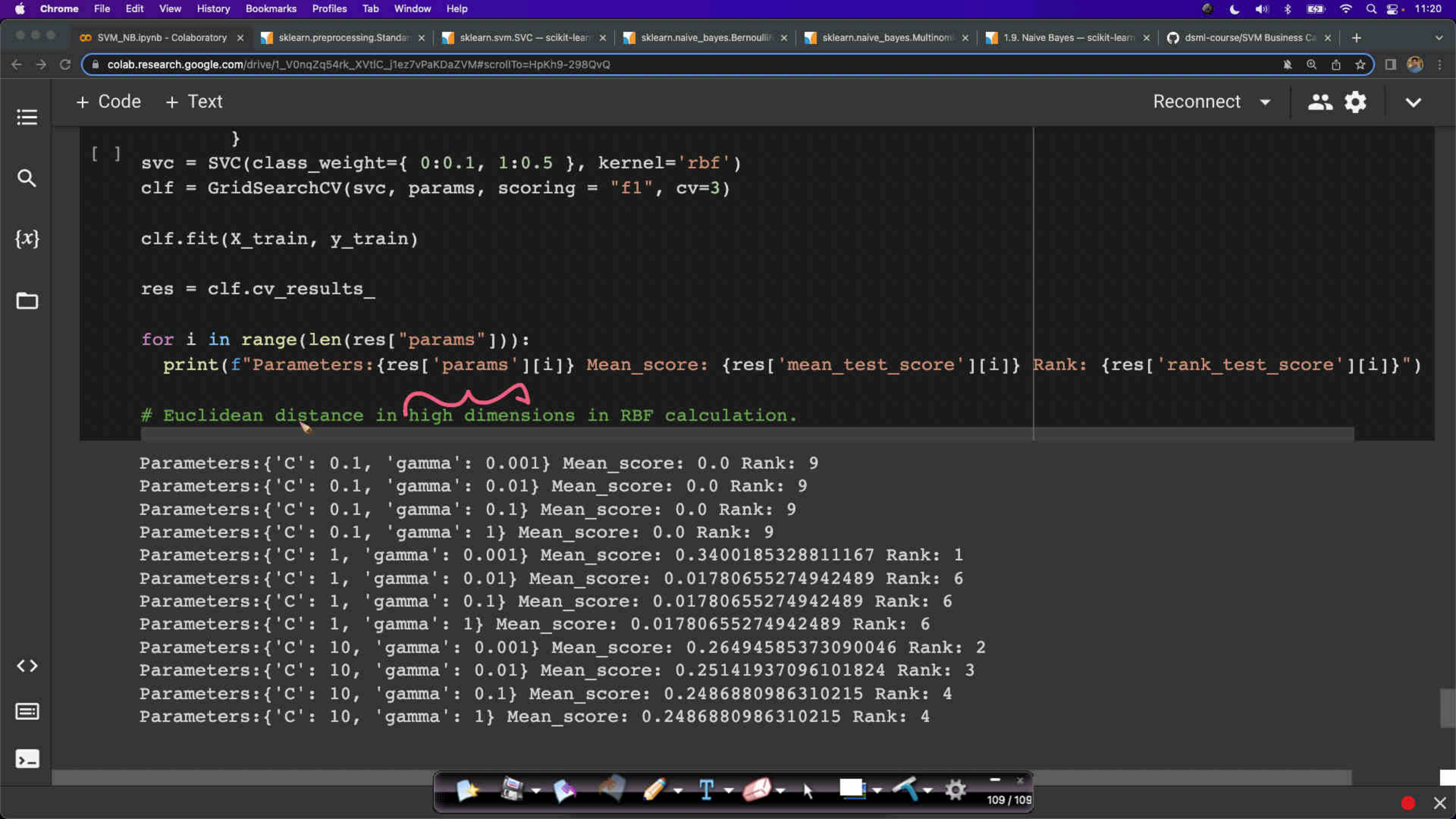
```
clf = GridSearchCV(svc, params, scoring = "f1", cv=3)

clf.fit(X_train, y_train)

res = clf.cv_results_

for i in range(len(res["params"])):
  print(f"Parameters:{res['params'][i]} Mean_score: {res['mean_test_score'][i]} Rank: {res['rank_test_score'][i]}")

# Euclidean distance in high dimensions in RBF calculation.
```

```
Parameters:{'C': 0.1, 'gamma': 0.001} Mean_score: 0.0 Rank: 9
Parameters:{'C': 0.1, 'gamma': 0.01} Mean_score: 0.0 Rank: 9
Parameters:{'C': 0.1, 'gamma': 0.1} Mean_score: 0.0 Rank: 9
Parameters:{'C': 0.1, 'gamma': 1} Mean_score: 0.0 Rank: 9
Parameters:{'C': 1, 'gamma': 0.001} Mean_score: 0.34001853288111167 Rank: 1
Parameters:{'C': 1, 'gamma': 0.01} Mean_score: 0.01780655274942489 Rank: 6
Parameters:{'C': 1, 'gamma': 0.1} Mean_score: 0.01780655274942489 Rank: 6
Parameters:{'C': 1, 'gamma': 1} Mean_score: 0.01780655274942489 Rank: 6
Parameters:{'C': 10, 'gamma': 0.001} Mean_score: 0.26494585373090046 Rank: 2
Parameters:{'C': 10, 'gamma': 0.01} Mean_score: 0.25141937096101824 Rank: 3
Parameters:{'C': 10, 'gamma': 0.1} Mean_score: 0.2486880986310215 Rank: 4
Parameters:{'C': 10, 'gamma': 1} Mean_score: 0.2486880986310215 Rank: 4
```

f1:

f1: 0.34

Multinomial NB


Chrome File Edit View History Bookmarks Profiles Tab Window Help
11:20
SVM_NB.ipynb - Colaboratory
sklearn.preprocessing.Standa
sklearn.svm.SVC — scikit-lear
sklearn.naive_bayes.Bernoulli
sklearn.naive_bayes.Multinom
1.9. Naive Bayes — scikit-learn
dsml-course/SVM Business Ca
colab.research.google.com/drive/1_V0nqZq54rk_XVtlC_j1ez7vPaKDaZVM#scrollTo=HpKh9-298QvQ
+ Code
+ Text
Reconnect
}
svc = SVC(class_weight={ 0:0.1, 1:0.5 }, kernel='rbf')
clf = GridSearchCV(svc, params, scoring = "f1", cv=3)
clf.fit(X_train, y_train)
res = clf.cv_results_
for i in range(len(res["params"])):
print(f"Parameters:{res['params'][i]} Mean_score: {res['mean_test_score'][i]} Rank: {res['rank_test_score'][i]}")
# Euclidean distance in high dimensions in RBF calculation.
Parameters:{'C': 0.1, 'gamma': 0.001} Mean_score: 0.0 Rank: 9
Parameters:{'C': 0.1, 'gamma': 0.01} Mean_score: 0.0 Rank: 9
Parameters:{'C': 0.1, 'gamma': 0.1} Mean_score: 0.0 Rank: 9
Parameters:{'C': 0.1, 'gamma': 1} Mean_score: 0.0 Rank: 9
Parameters:{'C': 1, 'gamma': 0.001} Mean_score: 0.3400185328811167 Rank: 1
Parameters:{'C': 1, 'gamma': 0.01} Mean_score: 0.01780655274942489 Rank: 6
Parameters:{'C': 1, 'gamma': 0.1} Mean_score: 0.01780655274942489 Rank: 6
Parameters:{'C': 1, 'gamma': 1} Mean_score: 0.01780655274942489 Rank: 6
Parameters:{'C': 10, 'gamma': 0.001} Mean_score: 0.26494585373090046 Rank: 2
Parameters:{'C': 10, 'gamma': 0.01} Mean_score: 0.25141937096101824 Rank: 3
Parameters:{'C': 10, 'gamma': 0.1} Mean_score: 0.2486880986310215 Rank: 4
Parameters:{'C': 10, 'gamma': 1} Mean_score: 0.2486880986310215 Rank: 4
109 / 109

↳ 2 cells hidden

RBF SVM

[ ] ↳ 1 cell hidden

Multinomial NB

[ ] ↳ 1 cell hidden

RBF - SVM

MNB

LY SVM

Multinomial NB

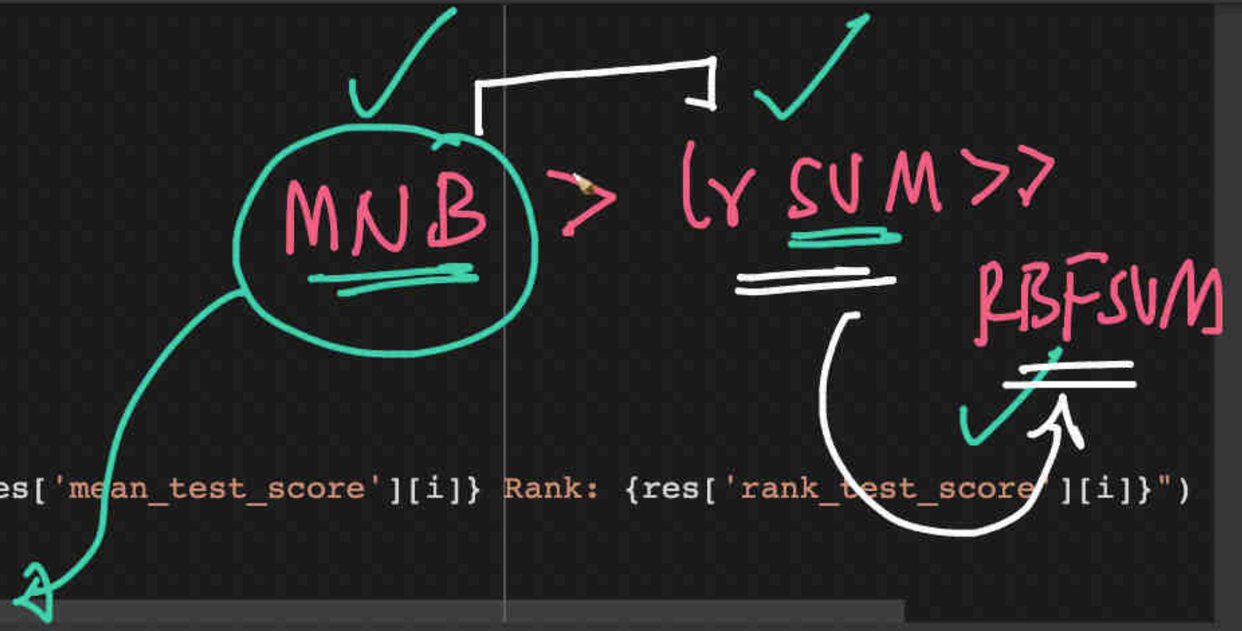
```
[ ] df_X_train, df_X_test, y_train, y_test = train_test_split(df['cleaned_message'], df['type'], test_size=0.25, random
    print([np.shape(df_X_train), np.shape(df_X_test)])

    #Count Vectorizer
    f = feature_extraction.text.CountVectorizer()

    X_train = f.fit_transform(df_X_train)
    X_test = f.transform(df_X_test)

    # No need of scaling

    # Multinomial NB

    from sklearn.model_selection import GridSearchCV

    params = {
            'alpha':[0.01, 0.1, 1, 10]
            }
    mnb = naive_bayes.MultinomialNB()
    clf = GridSearchCV(mnb, params, scoring = "f1", cv=3)
```

```
X_train = f.fit_transform(df_X_train)
X_test = f.transform(df_X_test)

# No need of scaling


# Multinomial NB

from sklearn.model_selection import GridSearchCV

params = {
        'alpha':[0.01, 0.1, 1, 10]
        }
mnb = naive_bayes.MultinomialNB()
clf = GridSearchCV(mnb, params, scoring = "f1", cv=3)

clf.fit(X_train, y_train)

res = clf.cv_results_

for i in range(len(res["params"])):
  print(f"Parameters:{res['params'][i]} Mean_score: {res['mean_test_score'][i]} Rank: {res['rank_test_score'][i]}")
```

```
X_train = f.fit_transform(df_X_train)
X_test = f.transform(df_X_test)

# No need of scaling


# Multinomial NB

from sklearn.model_selection import GridSearchCV

params = {
      'alpha':[0.01, 0.1, 1, 10]
        }
mnb = naive_bayes.MultinomialNB()
clf = GridSearchCV(mnb, params, scoring = "f1", cv=3)

clf.fit(X_train, y_train)

res = clf.cv_results_

for i in range(len(res["params"])):
  print(f"Parameters:{res['params'][i]} Mean_score: {res['mean_test_score'][i]} Rank: {res['rank_test_score'][i]}")
```

```
[(4179, ), (1393, )]
```

```
[ ]          'alpha':[0.01, 0.1, 1, 10]
             }
    mnb = naive_bayes.MultinomialNB()
    clf = GridSearchCV(mnb, params, scoring = "f1", cv=3)

    clf.fit(X_train, y_train)

    res = clf.cv_results_

    for i in range(len(res["params"])):
      print(f"Parameters:{res['params'][i]} Mean_score: {res['mean_test_score'][i]} Rank: {res['rank_test_score'][i]}")
```




```
[(4179,), (1393,)]
Parameters:{'alpha': 0.01} Mean_score: 0.8921625905385584 Rank: 2
Parameters:{'alpha': 0.1} Mean_score: 0.8883633779156167 Rank: 3
Parameters:{'alpha': 1} Mean_score: 0.9006541826507674 Rank: 1
Parameters:{'alpha': 10} Mean_score: 0.8580328947757173 Rank: 4
```

Chrome File Edit View History Bookmarks Profiles Tab Window Help

SVM_NB.ipynb - Colaboratory | sklearn.preprocessing.Standa | sklearn.svm.SVC — scikit-lear | sklearn.naive_bayes.Bernoulli | sklearn.naive_bayes.Multinomi | 1.9. Naive Bayes — scikit-learn | dsml-course/SVM Business Ca

colab.research.google.com/drive/1_V0nqZq54rk_XVtlC_j1ez7vPaKDaZVM#scrollTo=64NrpQPil0Wz

SVM_NB.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text Reconnect Editing

Data

```
# Importing libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from collections import Counter
from sklearn import feature_extraction, model_selection, naive_bayes, metrics, svm
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC

# Libraries for text processing
import re, nltk
nltk.download('punkt')
nltk.download('stopwords')
```

115 / 115

0s completed at 20:14