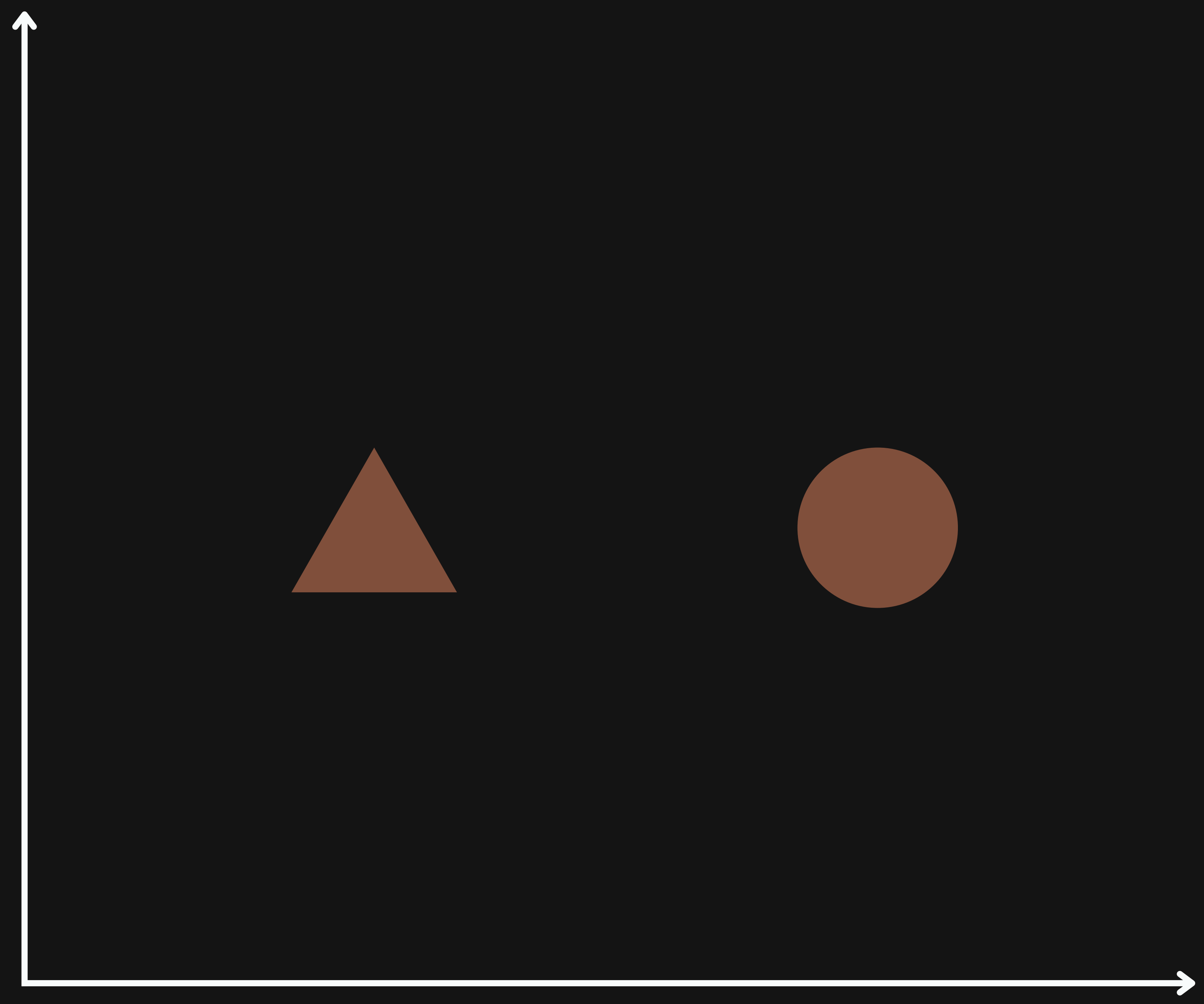
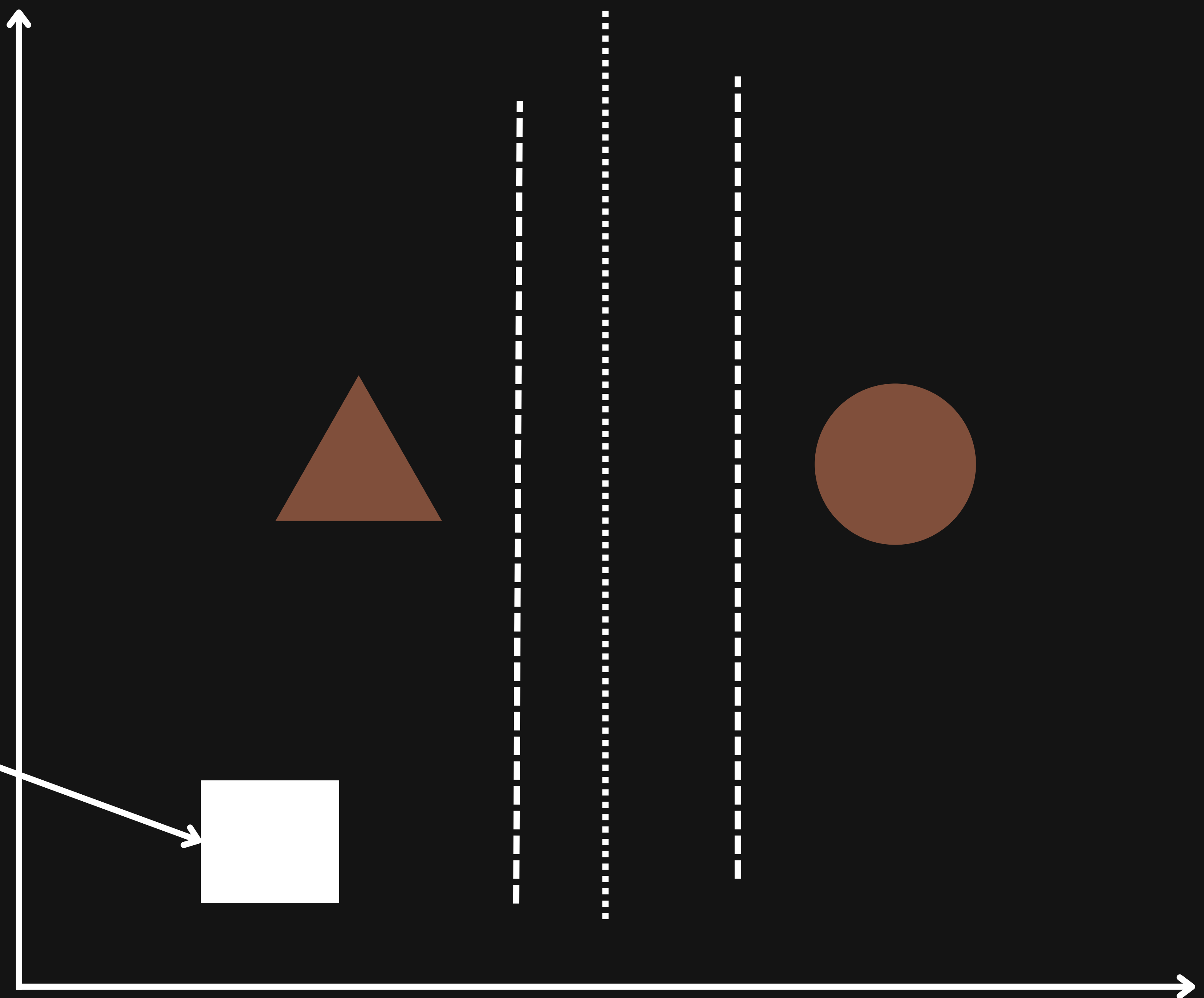
KNN Algo

AI Club
IIT MADRAS

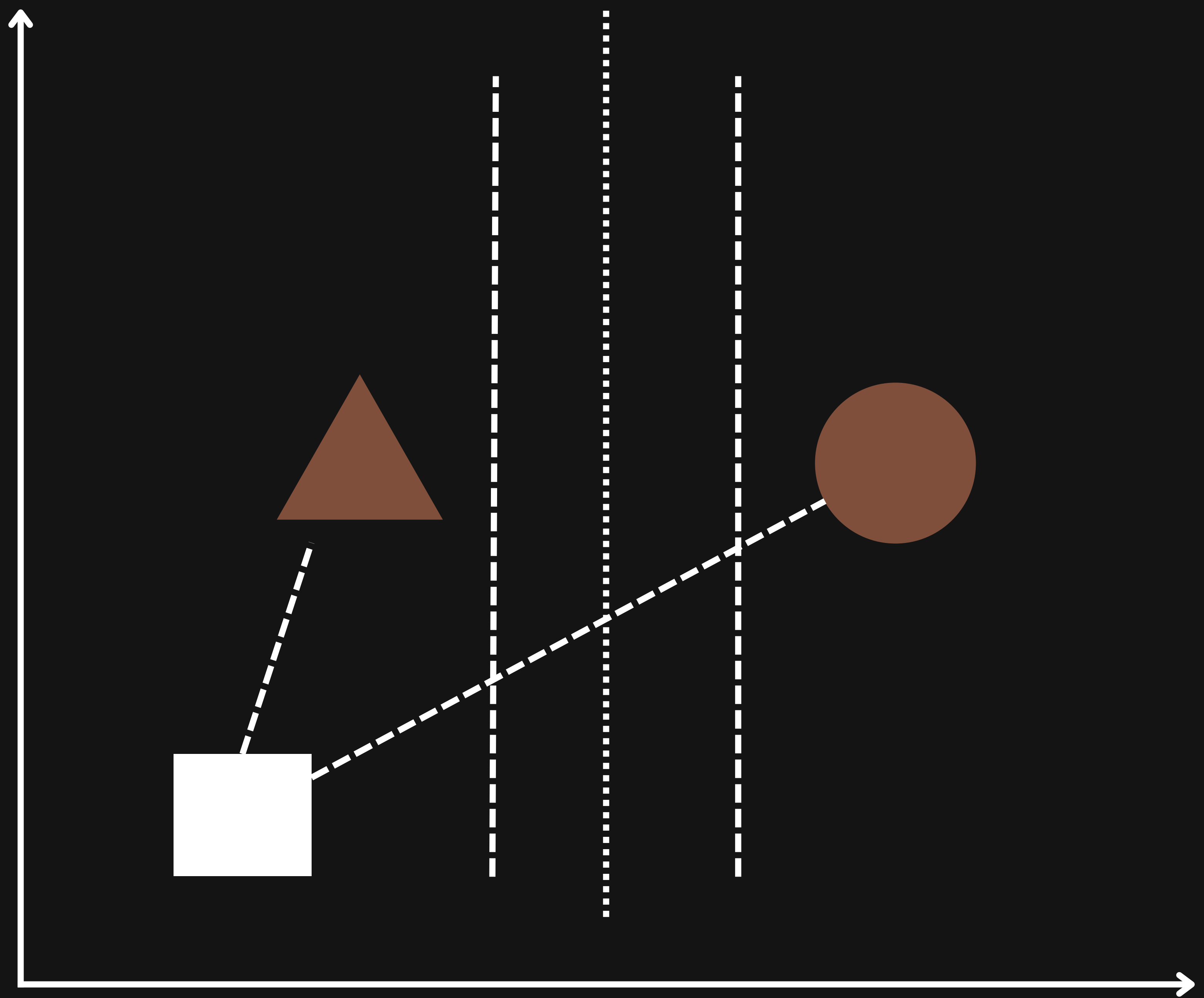


Let us begin with a simplest case of a classification problem.Let us try to classify the instances into   two classes.

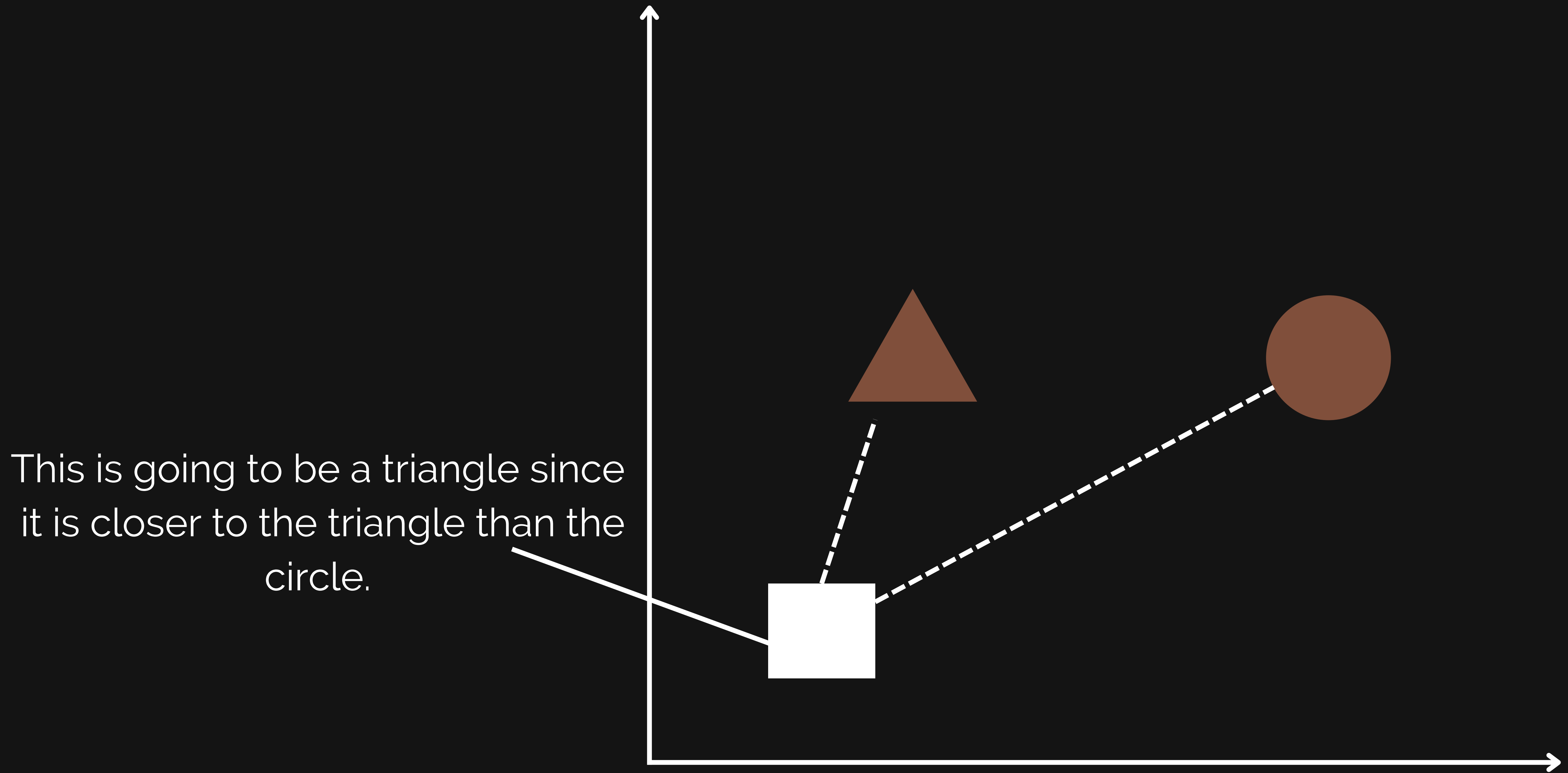
This is going to be a triangle since
it lies left to the decision boundary



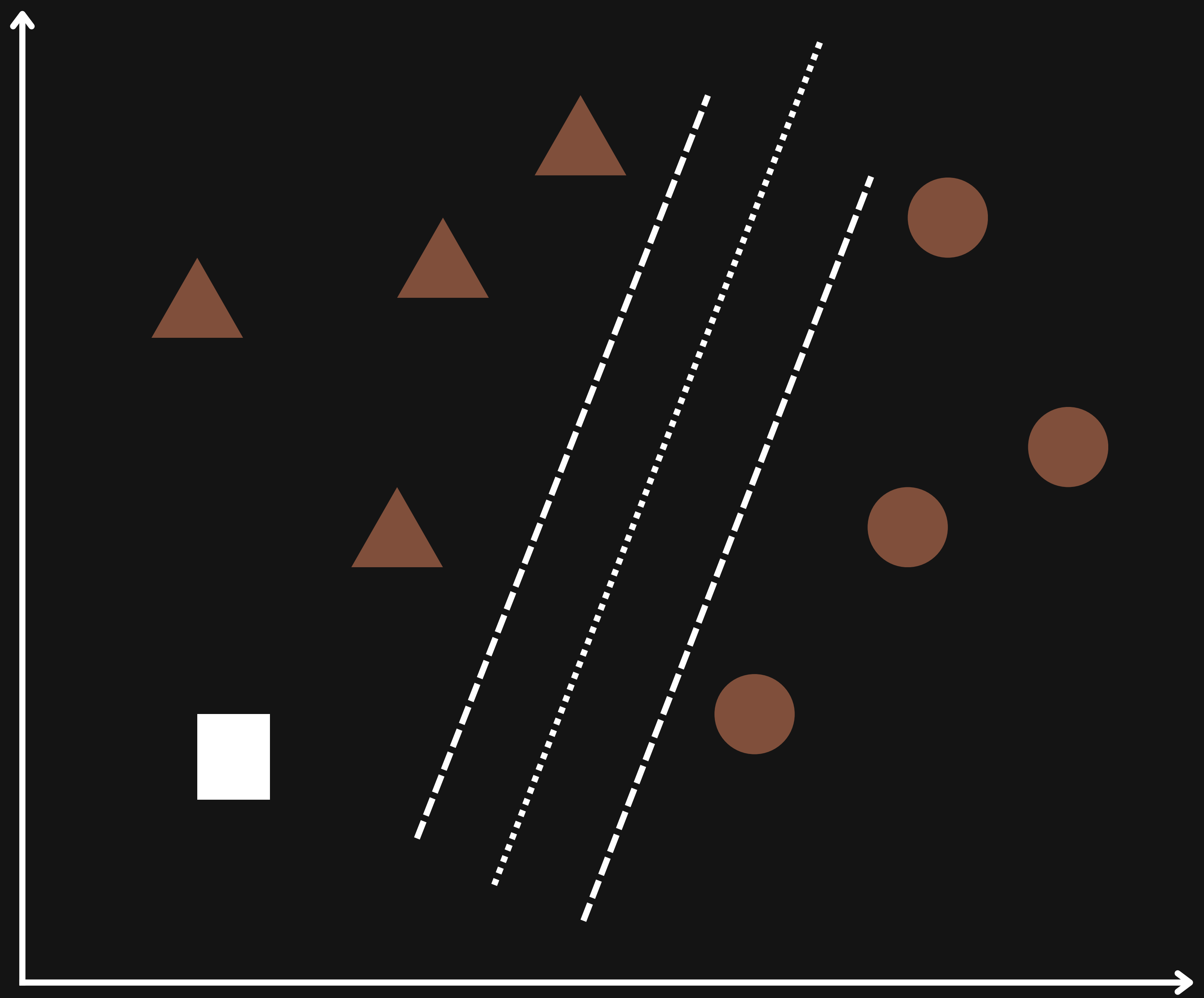
Let us try to classify it using a SVM classifier.Clearly, a line that goes midway between the two figures would be an ideal classifier



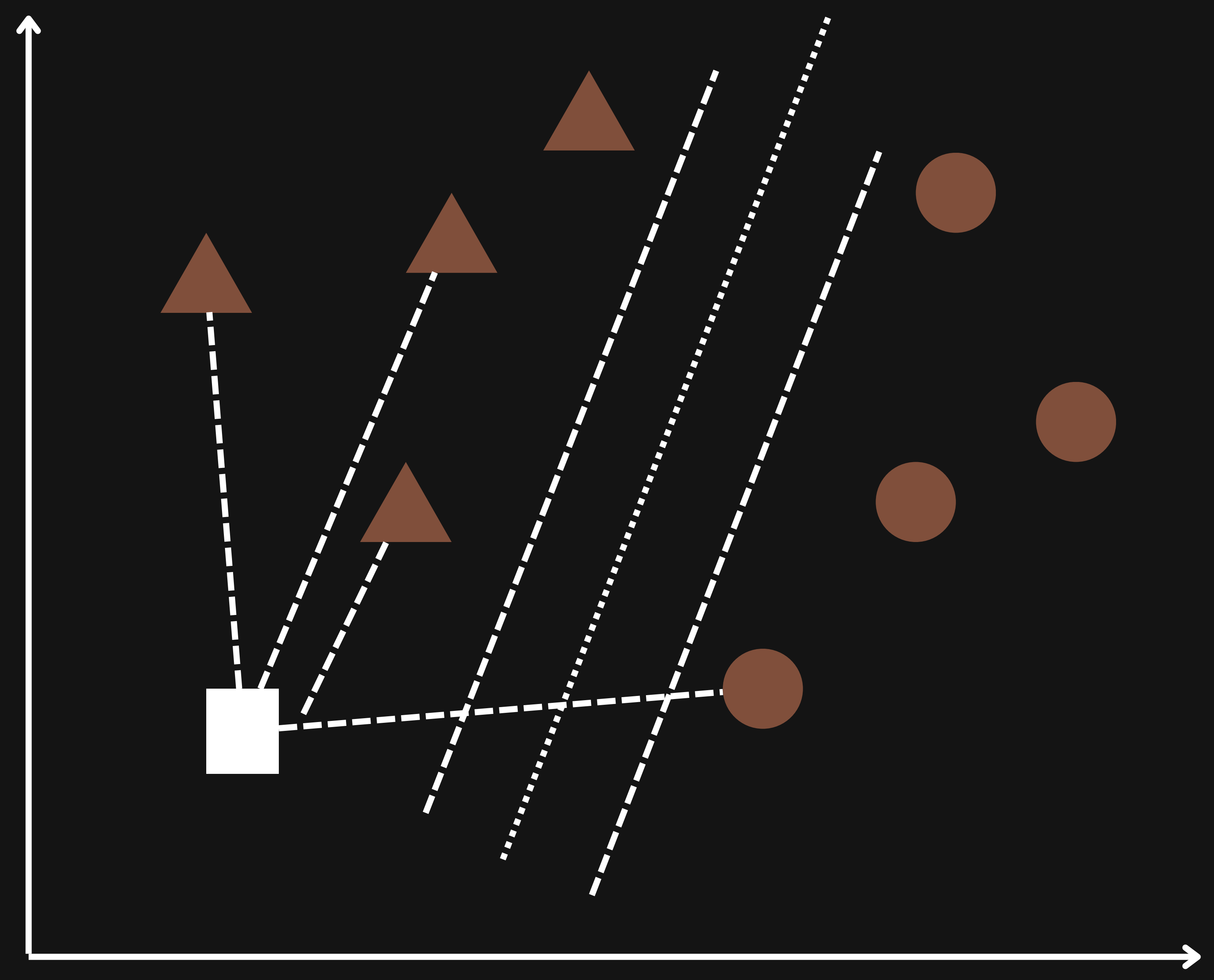
If the new point introduced falls into the left half plane,it is going to be closer to the triangle lying in the left plane.Infact it is intuitively easy to see that all points that lie to the left of the boundary lie closer to the triangle than the circle.



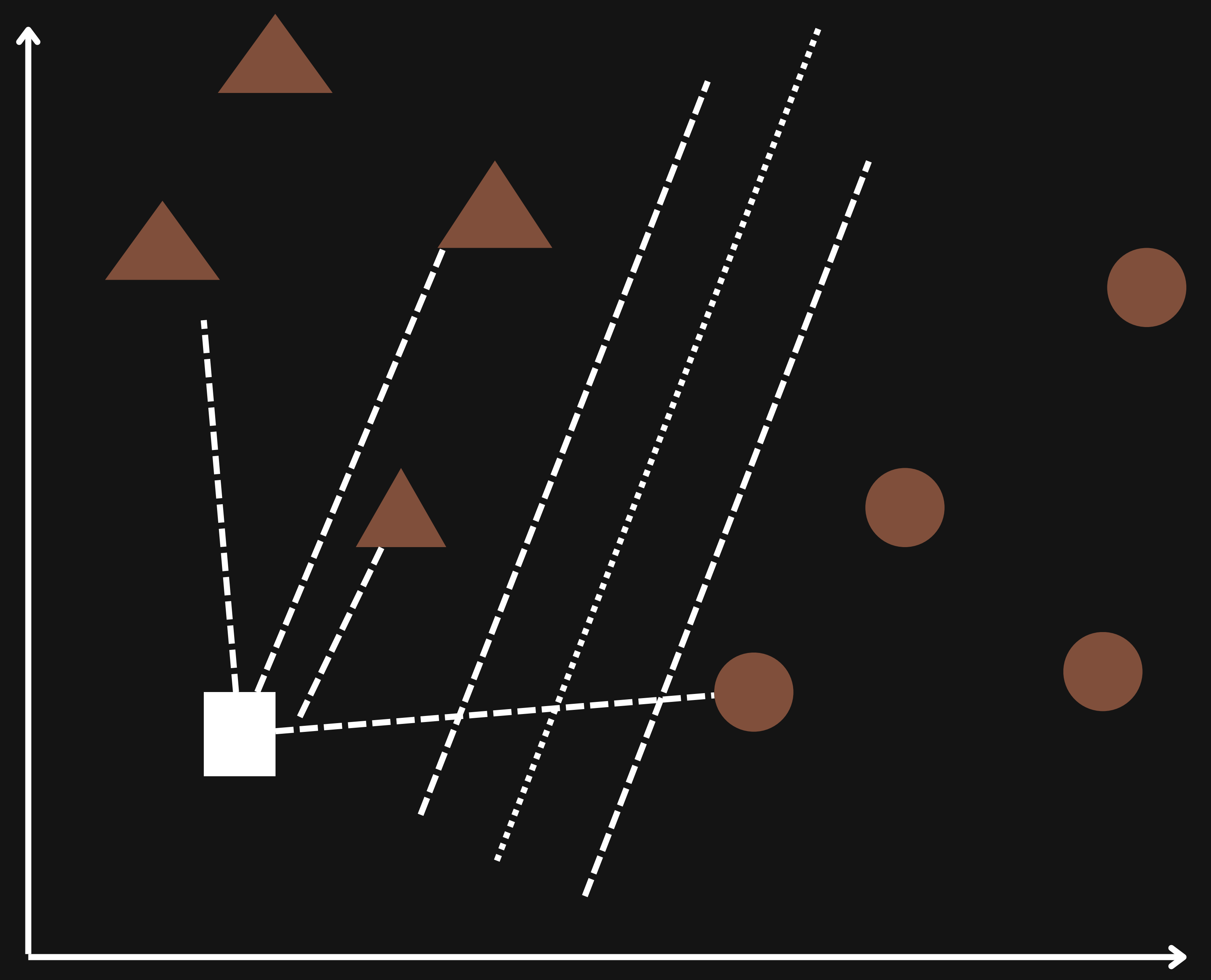
So another way of thinking to classify the unknown object as follows
Find out the nearest known object closer to it;And then classify it under the same label.;



Here we have a slightly more complex example.We clearly can see that the newly introduced point is a triangle,using SVM classifier,but can we use the alternative approach to figure out its label?



Ii should seem to work;Afterall the points on the left of the decision boudary seems to have more no of triangles as its "neighbours" than circles and hence can be classified as a triangle.



But in this case,there is an ambiguity;How many neighbours are we to consider?


U.K
NETHERLANDS
BELGIUM
LUXEMBOURG
GERMANY
POLAND
CZECH REPUBLIC
AUSTRIA
SWITZERLAND
LIECHTENSTEIN
SLOVENIA
CROATIA
ITALY
MONACO
FRANCE
Massif Central
English Channel
Strait of Dover
Channel Tunnel
Bay of Biscay
Adriatic Sea
Guernsey (U.K)
Jersey (U.K)
Belle Île
PARIS
BERLIN
The Eiffel Tower
The Palace of Versailles
Musse d'Orsay Museum
Disneyland
The Louvre Palace
Notre Dame Cathedral
Georges Pompidou centre
Fontainebleau
Le Mont-Saint-Michel
Puy du Fou
La Rochelle
Charente Maritime
Puy de Sancy 1885 m
Mt. Blanc 4807m
Vanoise National Park
Ecrins National Park
Cévennes National Park
Pyrénées National Park
French Riviera
Film Festival
Calais
Lille
Amiens
Le Havre
Rouen
Caen
Reims
Metz
Strasbourg
Rennes
Le Mans
St-Nazaire
Nantes
Angers
Tours
Orléans
Dijon
Besançon
Limoges
Clermont-Ferrand
Lyon
Villeurbanne
St-Étienne
Grenoble
Bordeaux
Bayonne
Toulouse
Pau
Nîmes
Montpellier
Avignon
Aix-en Provence
Marseille
Nice
Cannes
Oldenburg
Pergamon Museum
Potsdam
Hannover
Magdeburg
Bielefeld
Braunschweig
Münster
Dortmund
Duisburg
Essen
Düsseldorf
Cologne Cathedral
Köln (Cologne)
Aachen
Bonn
Kassel
Halle
Leipzig
Barbarossa Cave
Erfurt
Dresden
Zwinger Palace
Dresden Frauenkirche
Elbe Sandstone Mountains
Chemnitz
Moselle Valley
Frankfurt
Frankfurt Book Fair
Mainz
Darmstadt
Mannheim
Rothenburg ob der Tauber
Schloß Heidelberg
Nuremberg
Karlsruhe
Stuttgart
Regensburg
Neue Pinakothek
Augsburg
München (Munich)
Deutsches Museum
Oktoberfest
Neuschwanstein Castle
Zugspitze 2962 m
Wendelstein 1838 m
Berchtesgaden
Kosseine 939 m
Großer Arbér 1456 m
Black Forest
LEGEND

S-MAN

"The man has to be french since he is within the territorial region of france."

K-MAN

"The man has to be french since the most of the guys he is interacting with is french."

Note the two distinct ways in which the two men derive their conclusion;Both could be wrong,both could be correct or only one of them might be accurate.

Now u know the basics of KNN algo!!Well done

KNN Algorithm

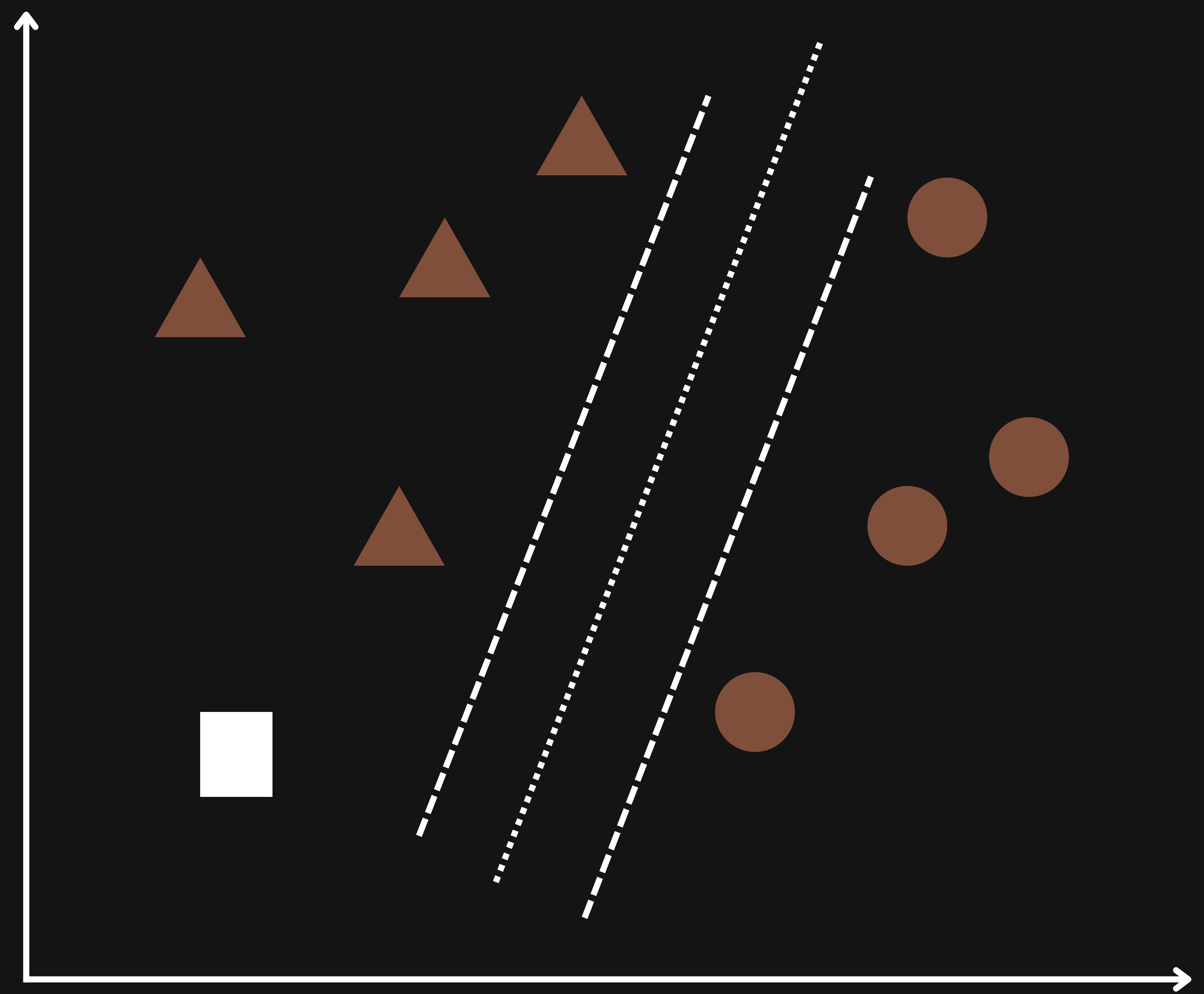
In simple Words KNN look for it's K nearest neighbours in by their distances and then mark the class of the point according to the majority of the neighbours.

Features of KNN

Lazy Learning

Instance-based learning

Non -Parametric:



How do u find the optimal value of K?

Choosing the K value is the hardest part of building a KNN model. As a rule of thumb we can choose K values as sqrt(n) Where 'n' is the no of datapoints . But The best way to do it is make a graph between loss vs K and choosing the correct K value for the implementation of the model.

How do we measure the distance to figure out the closeness of the objects?

$$\text{Minkowski Distance} = \left(\sum_{i=1}^{n} \left|x_i - y_i\right|^p\right)^{1/p}$$

$p = 2$

Euclidean distance :

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}$$

Manhattan distance:

$p = 1$

$$\text{Manhattan Distance} = d(x,y) = \left(\sum_{i=1}^{m} |x_i - y_i| \right)$$

Limitation of KNN

Time Complexity

Space Complexity

Naive Bayes Algo

Suppose we want to classify emails as spams or normal messages.We will do this on the basis of presence of certain words,namely dear,friend,lunch and money.
We collect all the messages we have and we go message by message and check if a particular word is present.
We will try to figure out the probability of the message being a nrml mssge given it has the words-dear and friend.Then we will do the same for spams and will figure out which is larger.

We can say that

P(message being a spam given it consists of dear,friend)

*P(message consisting of dear,friend)

=

P(message being a spam and consists of dear,friend)

=

P(message is a spam)

*P(message consisting of dear,friend given it is a spam)

P(message being a normal mssge given it consists of dear,friend)

*P(message consisting of dear,friend)

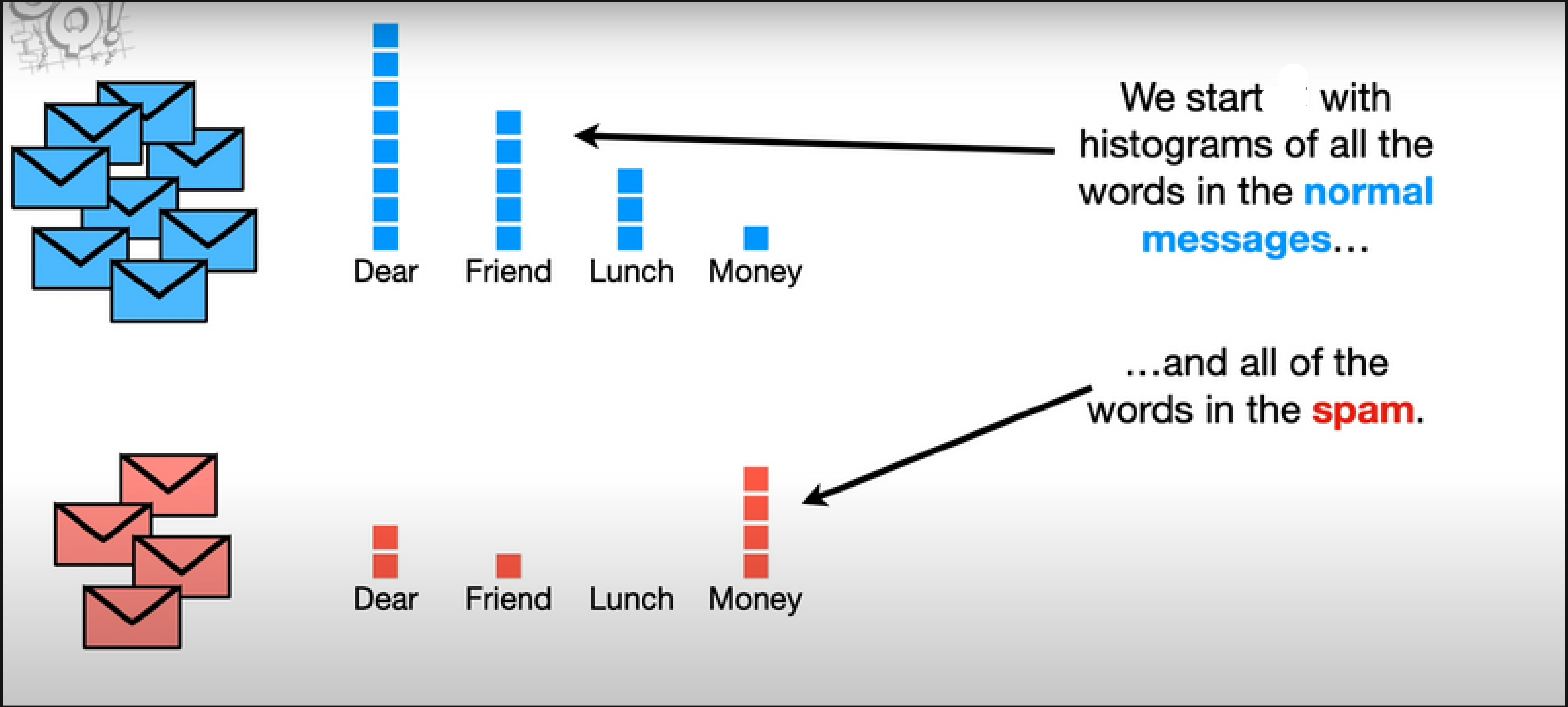
=

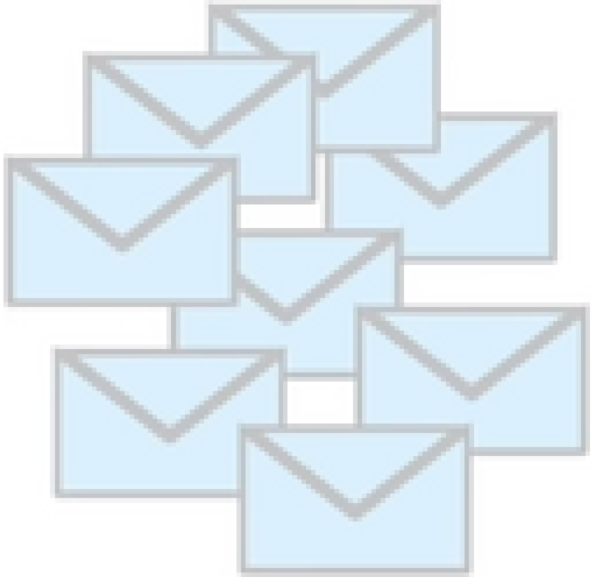
P(message being a nrml mssge and consists of dear,friend)

=

P(message is a nrml mssge)

*P(message consisting of dear,friend given it is a nrml mssge)


Dear
Friend
Lunch
Money
We start with histograms of all the words in the normal messages…
…and all of the words in the spam.
Dear
Friend
Lunch
Money



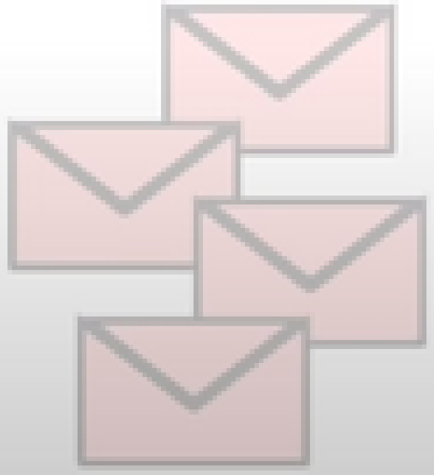
$p(\textbf{Dear} \mid \textbf{N}) = 0.47$

$p(\textbf{Friend} \mid \textbf{N}) = 0.29$

$p(\textbf{Lunch} \mid \textbf{N}) = 0.18$

$p(\textbf{Money} \mid \textbf{N}) = 0.06$

Then we calculate the probabilities of seeing each word, given that we saw the word in either a **normal message** or **spam**.



$p(\textbf{Dear} \mid \textbf{S}) = 0.29$

$p(\textbf{Friend} \mid \textbf{S}) = 0.14$

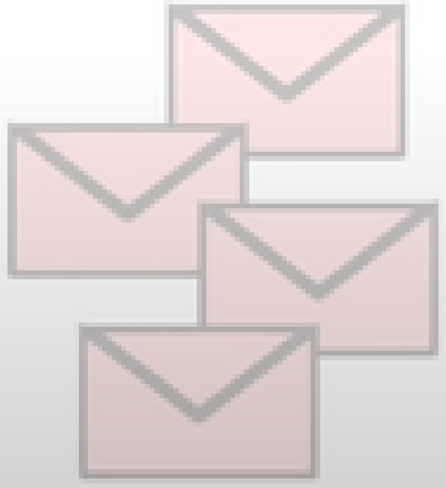
$p(\textbf{Lunch} \mid \textbf{S}) = 0.00$

$p(\textbf{Money} \mid \textbf{S}) = 0.57$

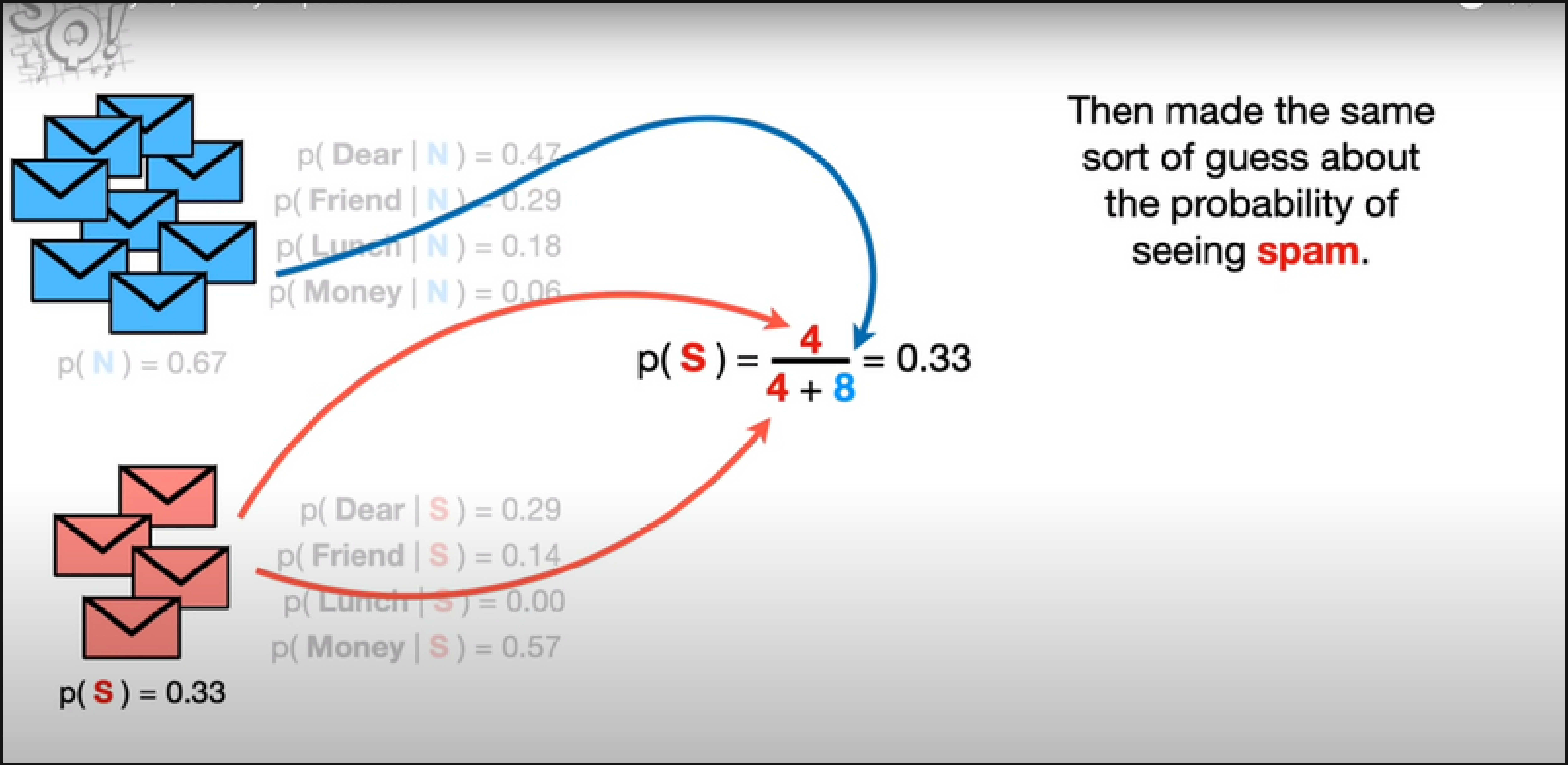
p(**Dear** | **N**) = 0.47
p(**Friend** | **N**) = 0.29
p(**Lunch** | **N**) = 0.18
p(**Money** | **N**) = 0.06

p(**N**) = 0.67

Then we made an initial guess about the probability of seeing a **normal message**.



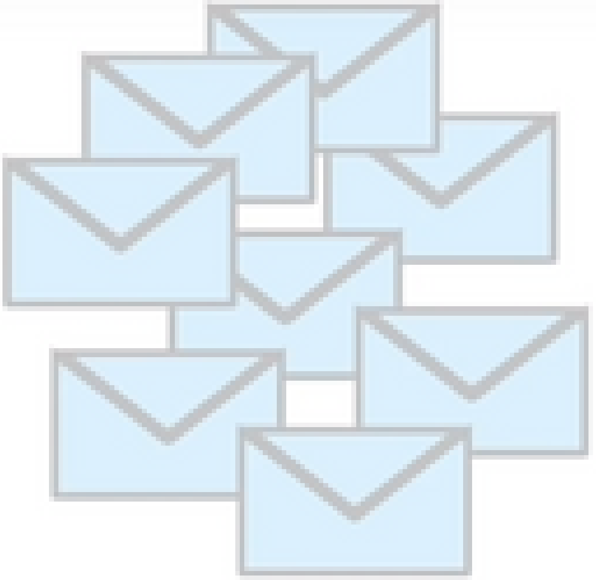
p(**Dear** | **S**) = 0.29
p(**Friend** | **S**) = 0.14
p(**Lunch** | **S**) = 0.00
p(**Money** | **S**) = 0.57



p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(N) = 0.67
p(S) = $\frac{4}{4 + 8}$ = 0.33
Then made the same sort of guess about the probability of seeing spam.
p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57
p(S) = 0.33

Dear Friend 

Then we did the math and decided that **Dear Friend** was a **normal message** because **0.09** > **0.01**.



p(**Dear** | **N**) = 0.47

p(**Friend** | **N**) = 0.29

p(**Lunch** | **N**) = 0.18

p(**Money** | **N**) = 0.06

p(**N**) = 0.67

p(**N**) x p(**Dear** | **N**) x p(**Friend** | **N**) $\boxed{= 0.09}$

p(**S**) x p(**Dear** | **S**) x p(**Friend** | **S**) = 0.01

p(**Dear** | **S**) = 0.29

p(**Friend** | **S**) = 0.14

p(**Lunch** | **S**) = 0.00

p(**Money** | **S**) = 0.57

p(**S**) = 0.33

Bayes Theorem

P (A and B happening)= P(A happening provided B happened)
* P(B happening)

P (A and B happening)= P(B happening provided A happened)
* P(A happening)

P(A happening provided B happened) * P(B happening)
=
P(B happening provided A happened) * P(A happening)

Thus,

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

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

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

So far the feature that we considered(presence of a particular word) can only take 2 possible values-yes or no
This is called mutinomial bayesian classification.
What if we had a feature that took continous values?

Now let us add one more feature into consideration while classifying an email-No of words

No of words	Type
35	SPAM
40	SPAM
70	NORMAL
76	NORMAL
56	SPAM

Suppose we get a new message which has the words Dear(20)money(43)

Now we need to include one more factor into calculations P(message has 63 words given it is a nrml mssge) and P(message has 63 words given it is a spam)

That is,

P(mssge is a spam)*P(mssge consists of 63 words given it is a spam)
*P(mssge consists of word dear(20)money(43)

And

P(mssge is a nrml)*P(mssge consists of 63 words given it is nrml)
*P(mssge consists of word dear(20)money(43) given it is nrml)

No of words	Type
35	SPAM
40	SPAM
70	NORMAL
76	NORMAL
56	SPAM

But we see that there is no data available for letters that are 63 words long.

We cannot expect that either,since no of words is a continous (approximately) quantity and we cannot have data which includes all the possible values that the "no of words" feature can take.

Note that the mean value for the no of words in spams is 43.67

Note that the standard deviation for the no of words in spams is 10.96

Note that the mean value for the no of words in normal msg is 73

Note that the standard deviation for the no of words in normal mssge is 3

Note that the mean value for the no of words in spams is 43.67

Note that the standard deviation for the no of words in spams is 10.96

Note that the mean value for the no of words in normal msg is 73

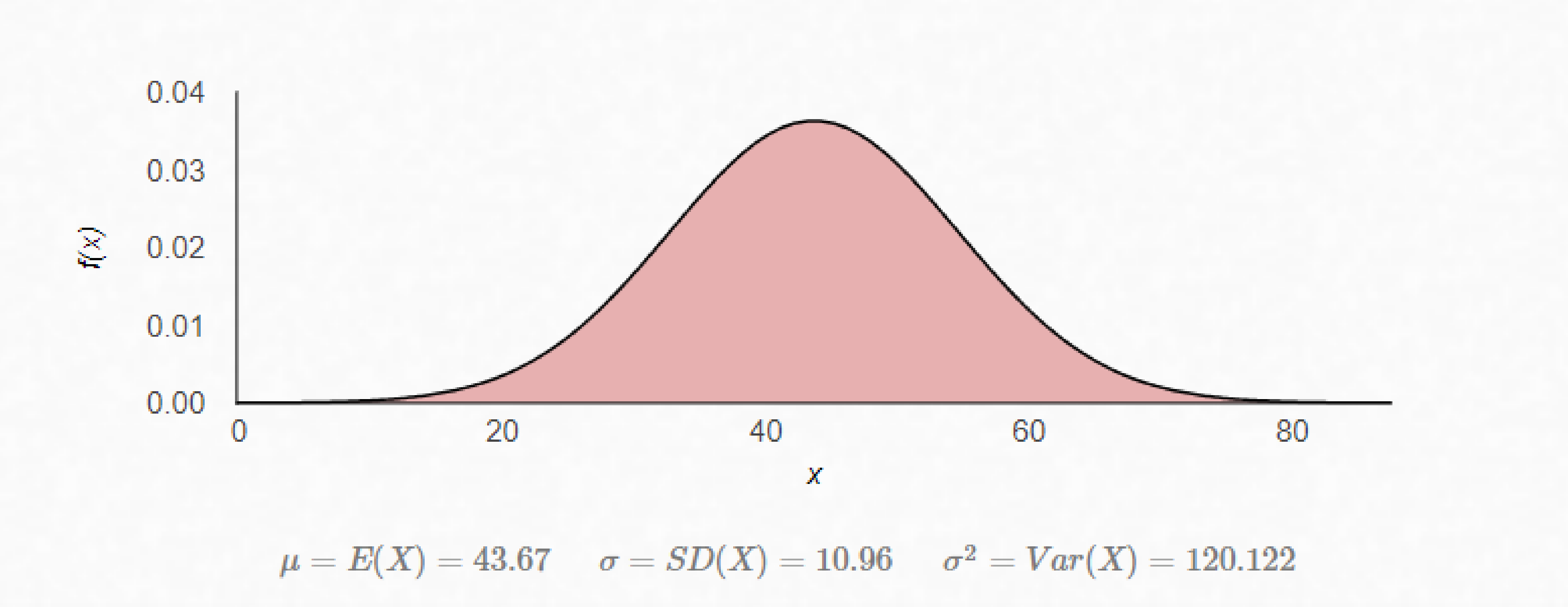
Note that the standard deviation for the no of words in normal mssge is 3

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

σ = standard deviation

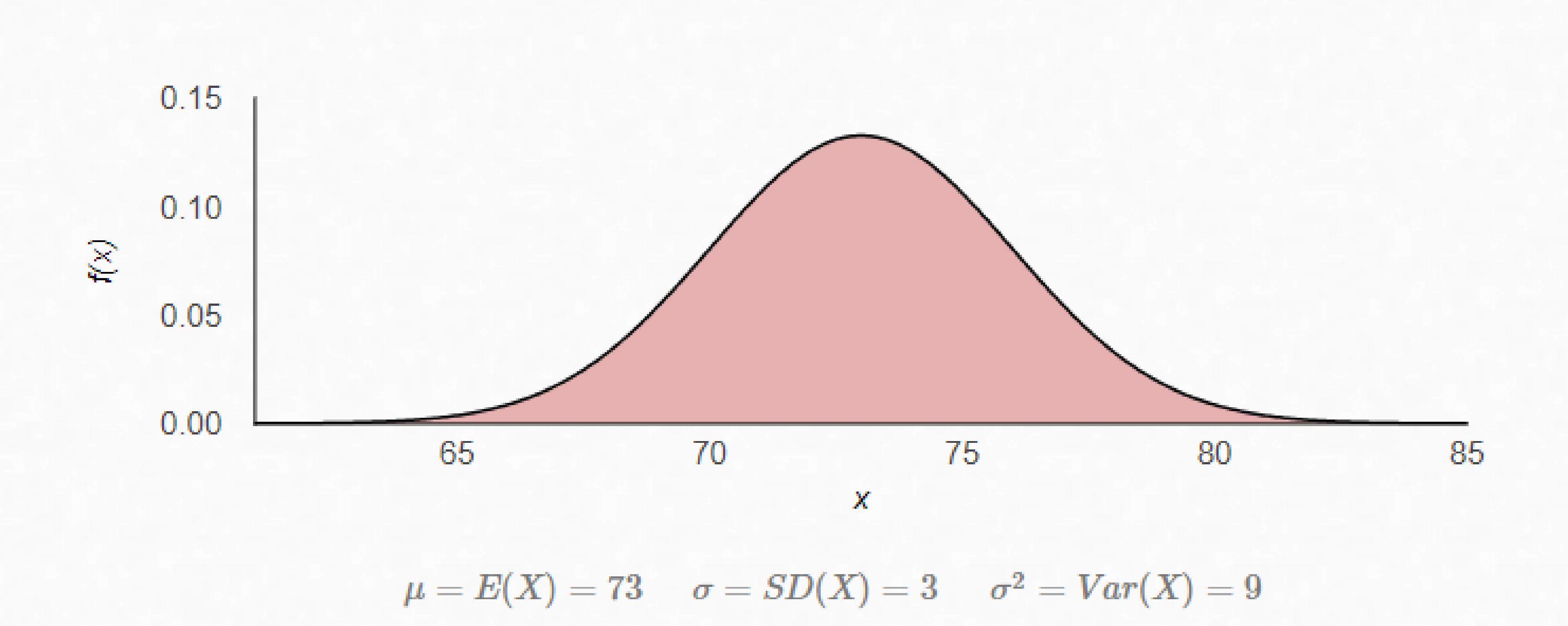
μ = mean

If we plug these values into this formula and plot the respective graphs, Spam

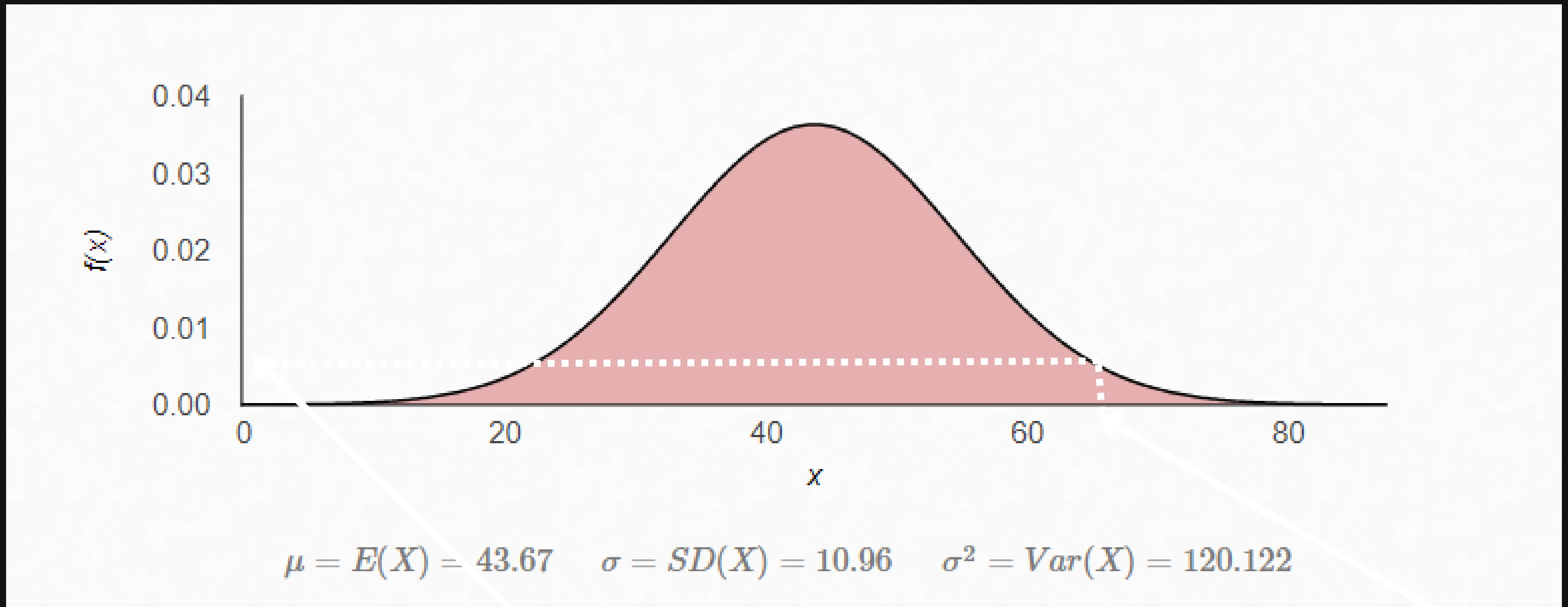


The value in the y axis represents the probability that message has x words given it is a spam.

Nrml msge



Suppose we need the P(mssge has 63 words|spam)

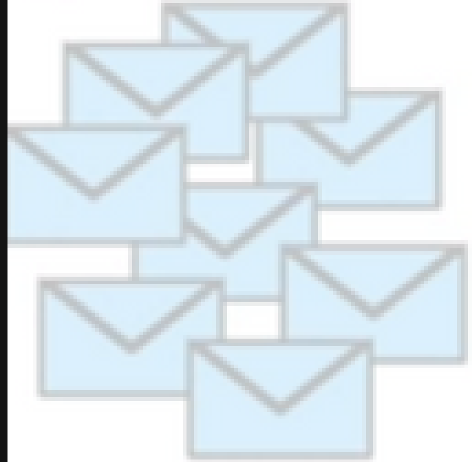


63

P(mssge has 63 words|spam)

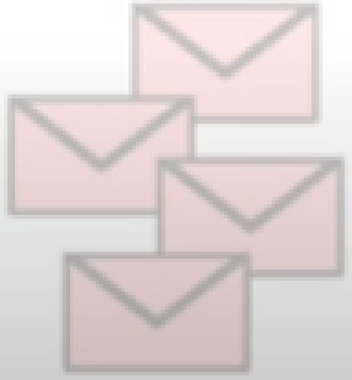
Lunch Money Money Money Money

This time, let's try to classify this message.



p(**Dear** | N) = 0.47
p(**Friend** | N) = 0.29
p(**Lunch** | N) = 0.18
p(**Money** | N) = 0.06

p(N) = 0.67



p(**Dear** | S) = 0.29
p(**Friend** | S) = 0.14
p(**Lunch** | S) = 0.00
p(**Money** | S) = 0.57

p(S) = 0.33

Lunch Money Money Money Money

p(**Dear** | N) = 0.47
p(**Friend** | N) = 0.29
p(**Lunch** | N) = 0.18
p(**Money** | N) = 0.06

p(N) = 0.67

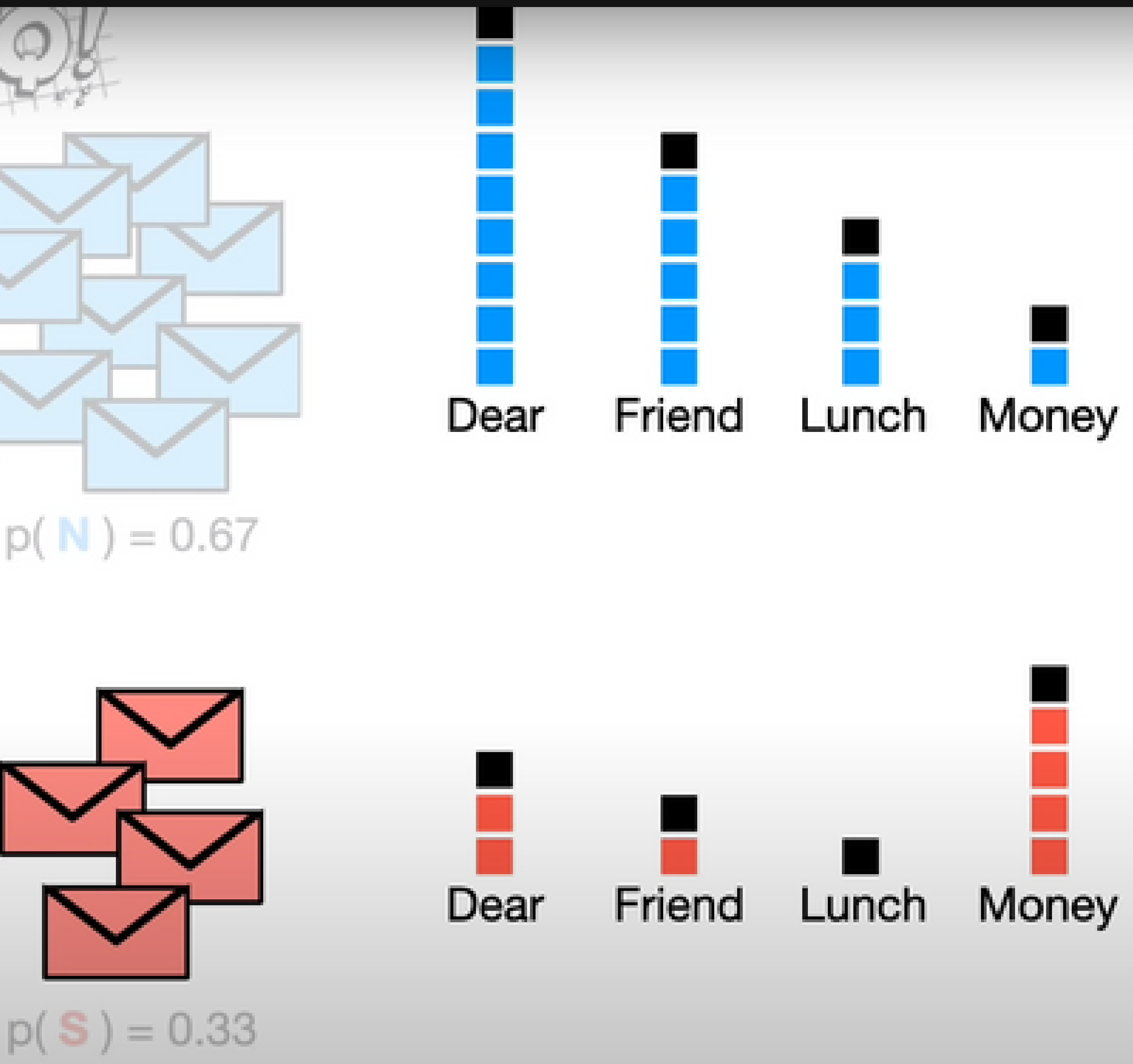
…and that means we will always classify the messages with **Lunch** in them as **normal**, no matter how how many times we see the word **Money**.

$$p(N) \times p(\textbf{Lunch} \mid N) \times p(\textbf{Money} \mid N)^4 = 0.000002$$

$$p(S) \times p(\textbf{Lunch} \mid S) \times p(\textbf{Money} \mid S)^4 = 0$$

p(**Dear** | S) = 0.29
p(**Friend** | S) = 0.14
p(**Lunch** | S) = 0.00
p(**Money** | S) = 0.57

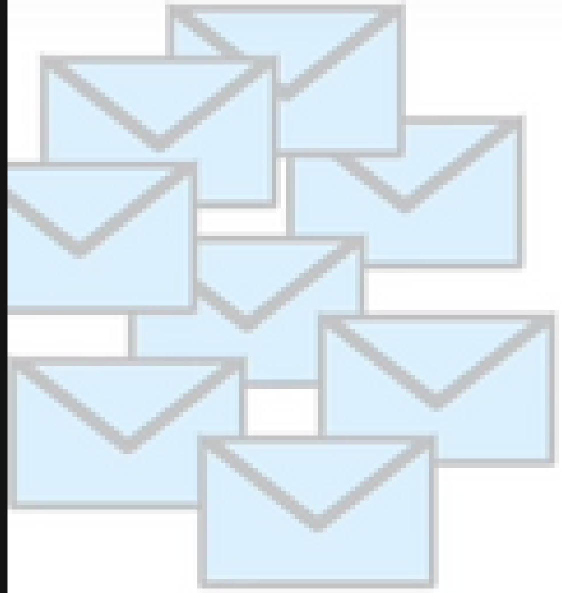
p(S) = 0.33



To avoid this problem we add one more count to each feature,so that none of the Probabilities vanishes.

This is called "smoothing"

Lunch Money Money Money Money



p(**Dear** | **N**) = 0.43

p(**Friend** | **N**) = 0.29

p(**Lunch** | **N**) = 0.19

p(**Money** | **N**) = 0.10

p(**N**) = 0.67

And since the value for **spam** is greater than the one for a **normal message**…

$$p(N) \times p(\textbf{Lunch} \mid N) \times p(\textbf{Money} \mid N)^4 = 0.00001$$

$$p(S) \times p(\textbf{Lunch} \mid S) \times p(\textbf{Money} \mid S)^4 = 0.00122$$

p(**Dear** | **S**) = 0.27

p(**Friend** | **S**) = 0.18

p(**Lunch** | **S**) = 0.09

p(**Money** | **S**) = 0.45

p(**S**) = 0.33

Now let us consider two sentences with the same words

1)" John , you do take care of your family ,hey?"

2)"Hey John ,do you take care of your family?"

Note that 2 sentences constructed with same words

can appear in different contexts.

Since Naive Bayes does not care about the order of the words

It ignores the semantics and context of the sentence.

This is why it is "naive".