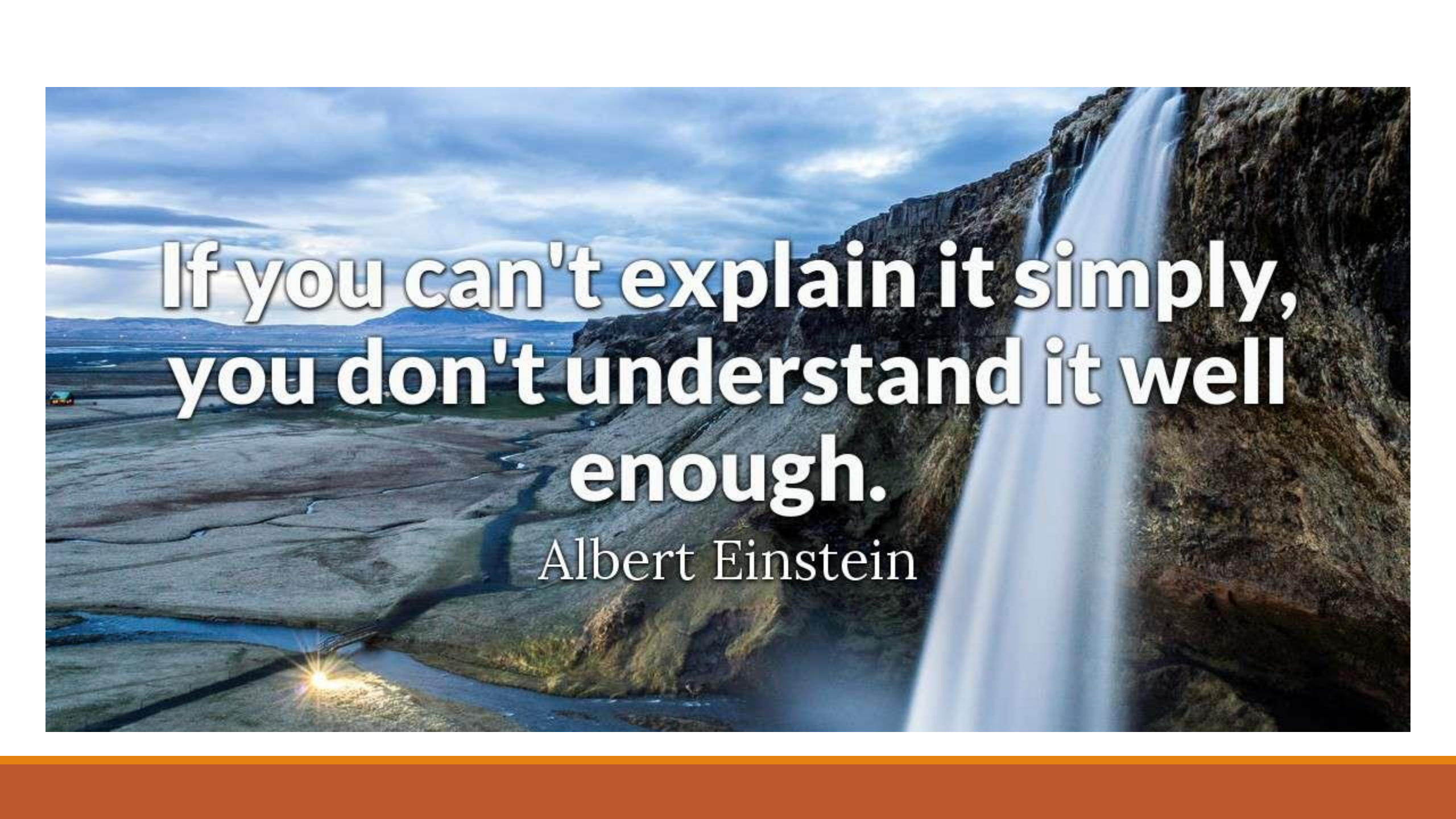


Naïve Bayes & SVM

CLASSIFICATION | GAUSSIAN | BERNOULLI | KERNEL

A solid orange horizontal bar at the bottom of the slide.



**If you can't explain it simply,
you don't understand it well
enough.**

Albert Einstein

Approach to the course

Applications of AI / ML

Image Processing

- Image tagging / Image Recognition
- OCR or Optical Character Recognition
- Self-driving cars

Text Analysis

- Spam Filtering
- Sentiment Analysis
- Information Extraction

Data Mining

- Anomaly Detection
- Association Rules
- Grouping
- Predictions

Healthcare

- Medical Diagnosis
- Imaging Diagnosis
- Oncology
- Drug Trials

Video Games

- Reinforcement Learning

Robotics

- Industrial tasks
- Human simulations



Cost function

Logistic regression:

$$J(\theta) = -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)})) \right] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

Neural network:

$\rightarrow h_{\Theta}(x) \in \mathbb{R}^K$ $(h_{\Theta}(x))_i = i^{th}$ output

$$\rightarrow J(\Theta) = -\frac{1}{m} \left[\sum_{i=1}^m \sum_{k=1}^K y_k^{(i)} \log(h_{\Theta}(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_{\Theta}(x^{(i)}))_k) \right]$$

$\frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\Theta_{ji}^{(l)})^2$

$\Theta_{ji}^{(l)} = \omega_{ji}^{(l)} + b_j^{(l)}$

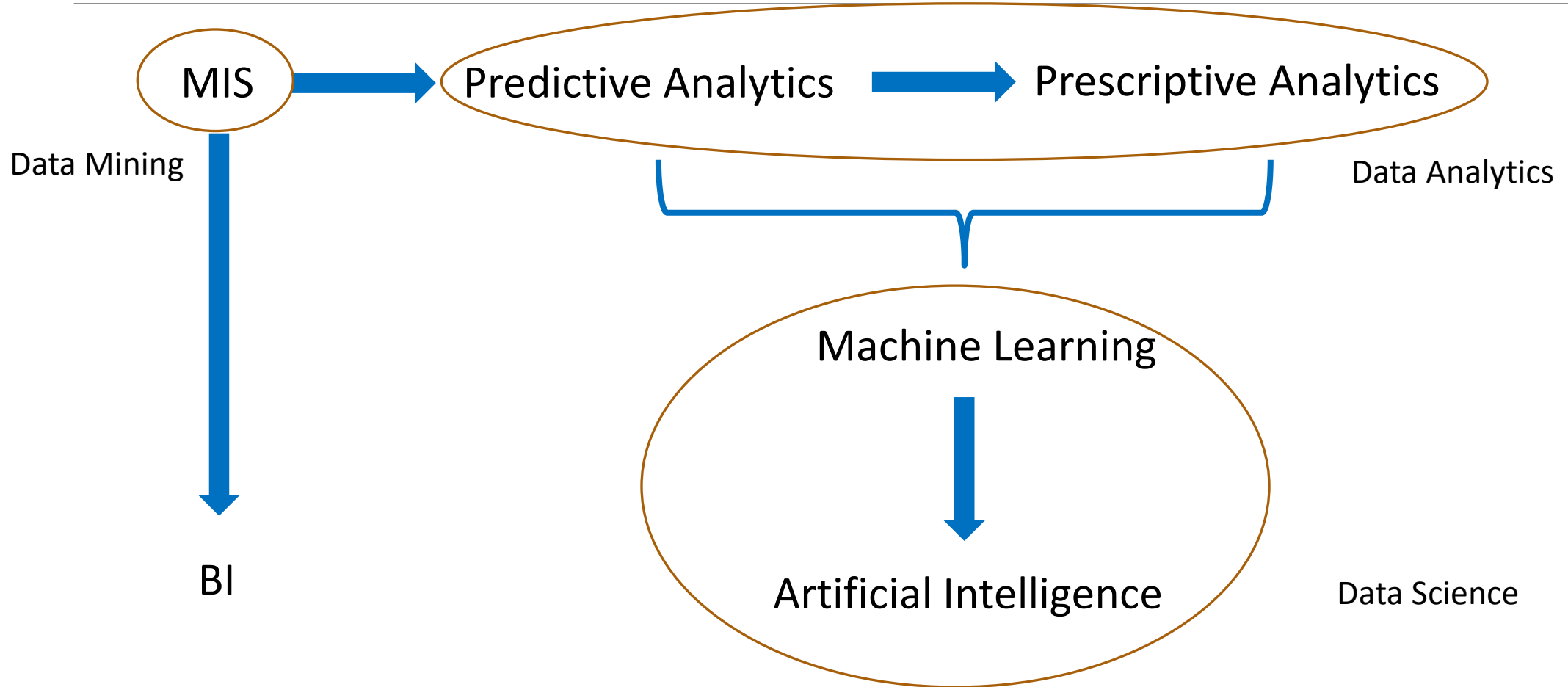
$\omega_{ji}^{(l)} = \omega_{ji}^{(l)} x_i + \dots$

$y_k = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_K \end{bmatrix}$

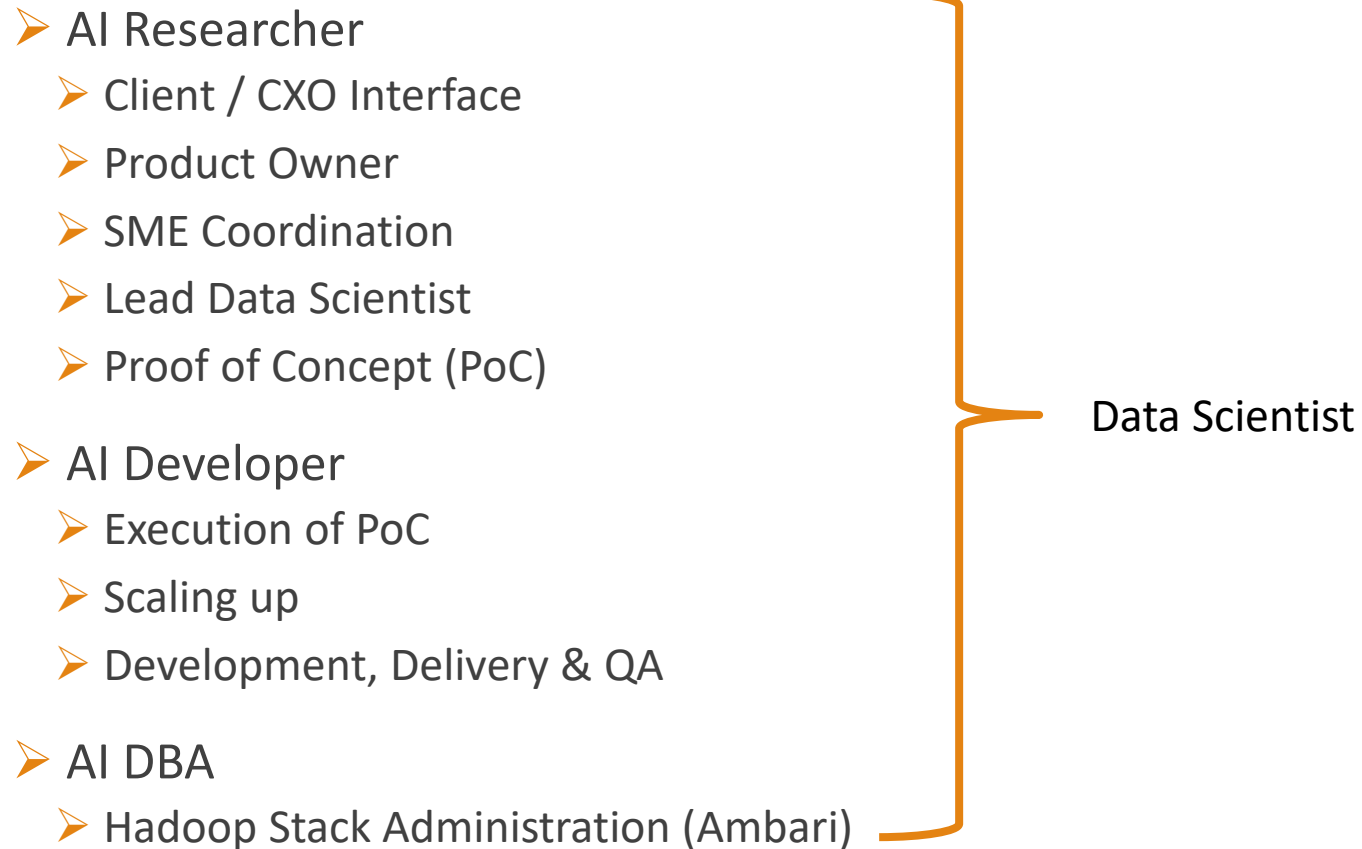
Andrew Ng



The Progression



Roles / Profiles / Designation



Why AI? Let us Intuit.

➤ Would you cross this bridge?

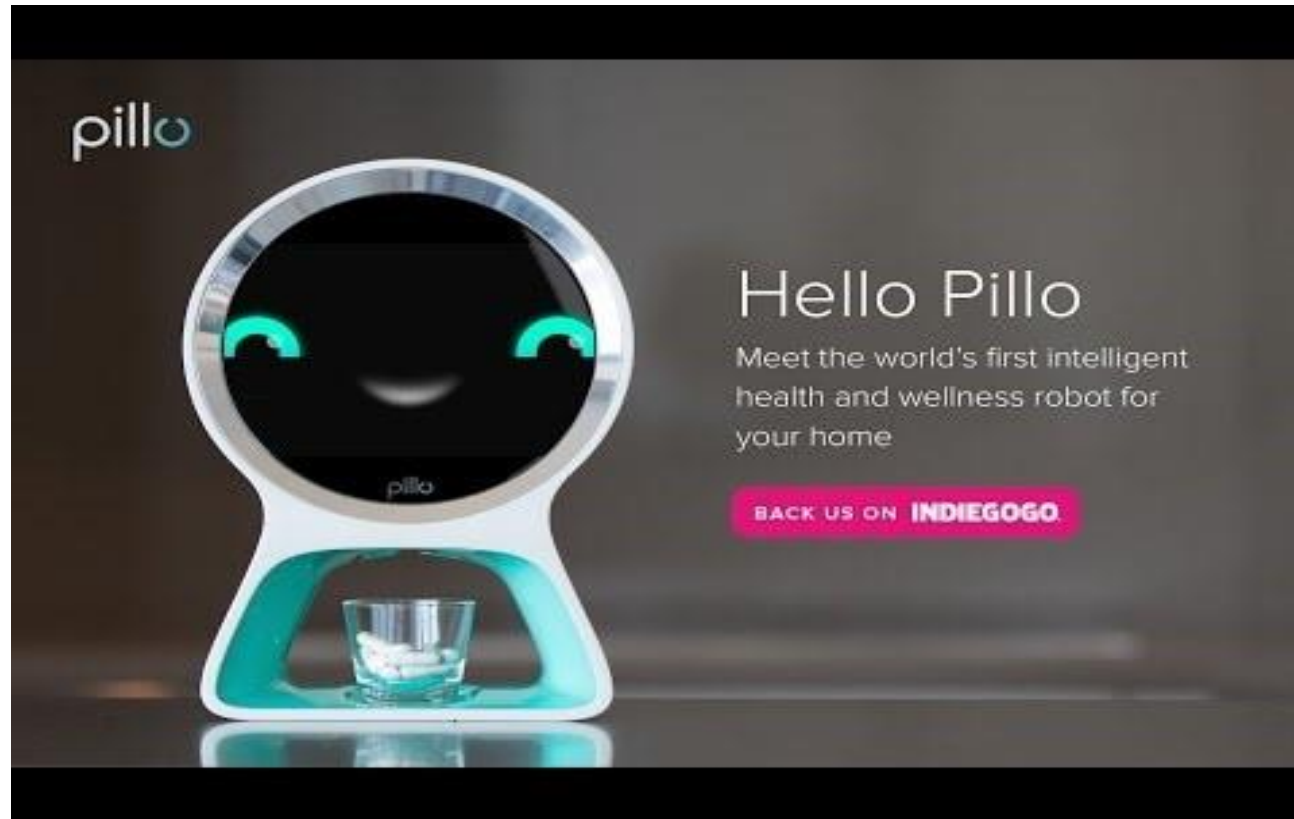


Examples of AI Systems - BitBite



https://www.youtube.com/watch?v=qU2w_qiP4Ck

Examples of AI Systems - Pillo



<https://www.youtube.com/watch?v=GfjGPTKBFB4&t=1s>

Examples of AI Systems - Cozmo



<https://www.youtube.com/watch?v=cjFA531qJNE>

AI in Application

```
In [50]: evidences = ["The patient has lower jaw pain."]
          hypotheses = ["The patient has heart attack."]

          sentence1 = [fit_to_size(np.vstack(sentence2sequence(evidence)[0]),
                                   (30, 50)) for evidence in evidences]

          sentence2 = [fit_to_size(np.vstack(sentence2sequence(hypothesis)[0]),
                                   (30, 50)) for hypothesis in hypotheses]

          prediction = sess.run(classification_scores, feed_dict={hyp: (sentence1 * N),
                                                                evi: (sentence2 * N),
                                                                y: [[0,0,0]]*N})
          print(["Positive", "Neutral", "Negative"][np.argmax(prediction[0])] +
                " entailment")
```

Negative entailment

AI in Application

```
In [49]: evidences = ["The patient has diabetes and lower jaw pain."]
          hypotheses = ["The patient has heart attack."]

          sentence1 = [fit_to_size(np.vstack(sentence2sequence(evidence)[0]),
                                   (30, 50)) for evidence in evidences]

          sentence2 = [fit_to_size(np.vstack(sentence2sequence(hypothesis)[0]),
                                   (30,50)) for hypothesis in hypotheses]

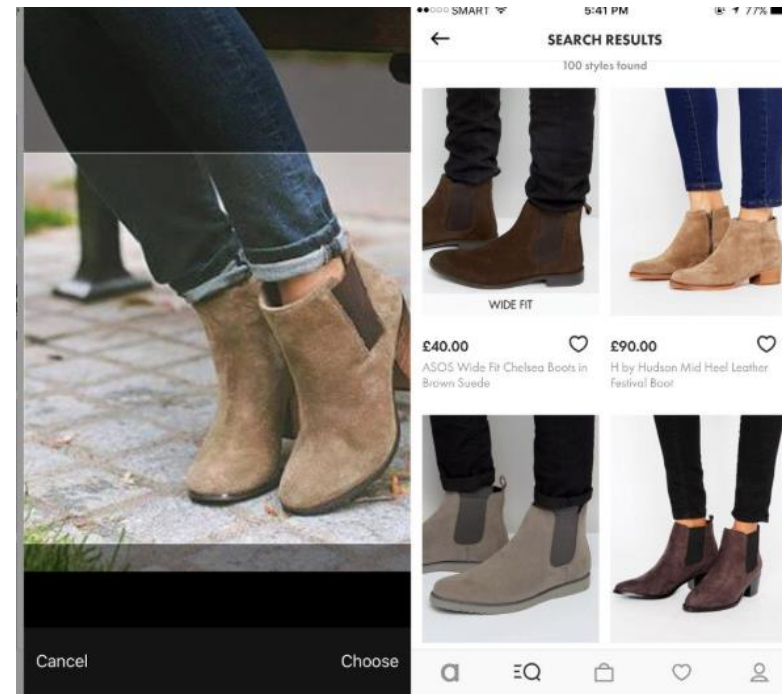
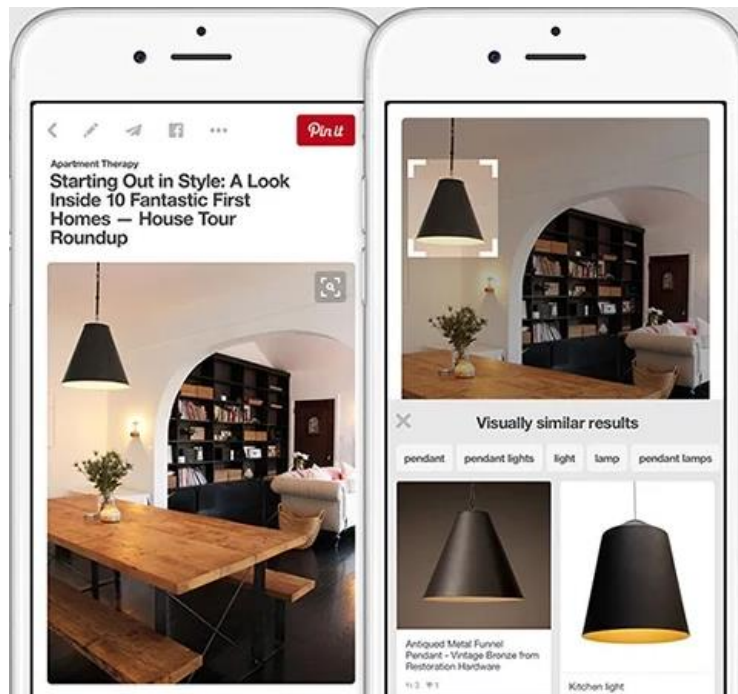
          prediction = sess.run(classification_scores, feed_dict={hyp: (sentence1 * N),
                                                                evi: (sentence2 * N),
                                                                y: [[0,0,0]]*N})

          print(["Positive", "Neutral", "Negative"][np.argmax(prediction[0])] +
                " entailment")
```

Positive entailment

AI Use Case – e Commerce

- App that suggests recipes using the inventory of your fridge – Chefling
- Custom Search – Pinterest.



AI Use Case – e Commerce

- Chatbots
- Virtual Assistants
 - By using Alexa on Amazon's Echo device, customers can discover local gigs for the upcoming weekend through StubHub, arrange transport to and from the event via Uber, or even order pre-event dinner from Domino's (and track the order status in real time).
- Wish
 - On the spot personal pricing
- Tackle fake reviews
 - www.fakespot.com
- Inventory Management & Sales Forecasting
- Generating product descriptions in any writing style

Imagine Real Time Modules

- Deception detection in speech
- Image verification – KYC – document photo and selfie
- Reading handwritten forms / applications (any language)
- Intuitive client risk profiling (social media data + financial data)
- Voice based initial screening of loan applicants - https://www.youtube.com/watch?v=oUf9_rS1fYg
- Transactional bots - digital assistants helping users navigate their finance plans, savings, and spending based on web footprints.
- OCR can be used to digitize hard copy documents. An NLP model with layered business logic can then interpret, record, and correct contracts at high speed.

Transactional Bots

➤ Automated Claims Processes –

```
> Hi
Hi, how can I help?
> I'd like to submit a claim
Sure. Which area is this regarding: Home, Car or Health?
> Home
Ok. I'm going to ask you a series of questions to help you fill the claim.
What is the nature of the incident?
> █
```

- It then moves on to the adjustment model where it provides a range of values for payout. Once all data is set, human intervention can be included for auditing purposes. The bot can at this point calculate and propose payout amounts, based on a payout predictor model it has been trained on.

Transactional Bots

- AI assistance with claims settlement is fast and more successful.
 - And fast means — 3 seconds from claim submission to payment. This is the result of the independent work of Lemonade's AI - Jim.
 - Lemonade is a New York based Insurance startup.
-
- What AI can do is to assist, advise, and create a better customer experience. At the moment, AI complements human work and makes it more efficient, but it isn't aimed to replace a human.

Story so far... and more

- Understanding the Requirements (think about a few logical parameters required)
- Understanding the data available (identify the gaps)
- Data Preparation
 - (EDA, missing values, outliers, feature engineering, scaling etc.)
 - Define data type
 - Train Test Split
- Train & tune the model (hyperparameter tuning)
- Test / Validate
- Saving the model
- Automate the pipeline
- Deploy (API)
- Refurbish

Bayes' Theorem

- How does computers learn about humans?
- An internet search for "movie automatic shoe laces" brings up "Back to the future".
- Has the search engine watched the movie?
- No, but it knows from lots of other searches what people are **probably** looking for.
- And it calculates that probability using Bayes' Theorem.

Bayes' Theorem

- Bayes' Theorem is a way of finding a probability when we know certain other probabilities.
- The formula is:

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

- Which tells us: how often A happens given that B happens, $P(A|B)$
- When we know: how often B happens given that A happens, $P(B|A)$
- and how likely A is to happen independently, $P(A)$
- and how likely B is to happen independently, $P(B)$

Example

- Shall we go for a picnic?
- You are planning a picnic today, but the morning is cloudy
- Oh no! 50% of all rainy days start off cloudy!
- But cloudy mornings are common (about 40% of days start cloudy)
- And this is usually a dry month (only 3 of 30 days tend to be rainy, or 10%)
- **What is the chance of rain during the day?**

Example

- We will use Rain to mean rain during the day, and Cloud to mean cloudy morning.
- The chance of Rain given Cloud is written $P(\text{Rain} | \text{Cloud})$
- So let's put that in the formula:

$$P(\text{Rain} | \text{Cloud}) = \frac{P(\text{Rain}) P(\text{Cloud} | \text{Rain})}{P(\text{Cloud})}$$

Example

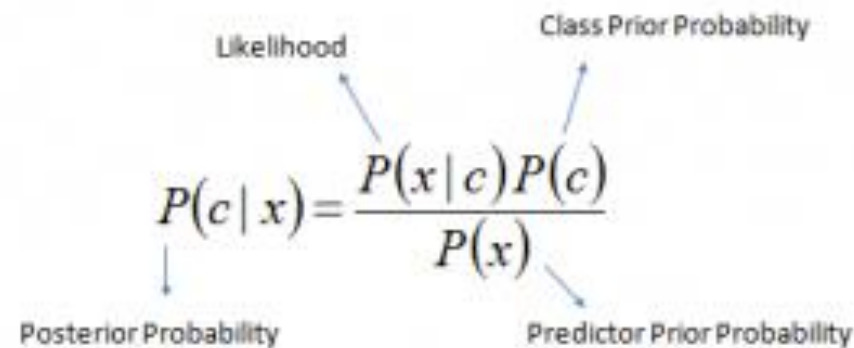
- $P(\text{Rain})$ is Probability of Rain = 10%
- $P(\text{Cloud} | \text{Rain})$ is Probability of Cloud, given that Rain happens = 50%
- $P(\text{Cloud})$ is Probability of Cloud = 40%

$$P(\text{Rain} | \text{Cloud}) = \frac{0.1 \times 0.5}{0.4} = .125$$

- Or a 12.5% chance of rain. Not too bad, let's have a picnic!

Summary

- Bayes theorem provides a way of calculating posterior probability $P(c|x)$ from $P(c)$, $P(x)$ and $P(x|c)$.



A diagram showing the Bayes' theorem formula $P(c|x) = \frac{P(x|c)P(c)}{P(x)}$ with arrows pointing from descriptive labels to the corresponding parts of the formula. The label 'Likelihood' points to $P(x|c)$, 'Class Prior Probability' points to $P(c)$, 'Posterior Probability' points to $P(c|x)$, and 'Predictor Prior Probability' points to $P(x)$.

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

Likelihood

Class Prior Probability

Posterior Probability

Predictor Prior Probability

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c)$$

- $P(c|x)$ is the posterior probability of *class* (c , *target*) given *predictor* (x , *attributes*).
- $P(c)$ is the prior probability of *class*.
- $P(x|c)$ is the likelihood which is the probability of *predictor* given *class*.
- $P(x)$ is the prior probability of *predictor*.

Naïve Bayes

- Robust classifier
- Not affected by irrelevant features
- Assumption – Independent Variables are not correlated, hence the name
- Doesn't care about distribution (except Gaussian NB), skewness, bias in data

Example

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
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

Learning Phase

Outlook	Play=Yes	Play=No
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Temperature	Play=Yes	Play=No
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Humidity	Play=Yes	Play=No
High	3/9	4/5
Normal	6/9	1/5

Wind	Play=Yes	Play=No
Strong	3/9	3/5
Weak	6/9	2/5

$$P(\text{Play=Yes}) = 9/14 \quad P(\text{Play=No}) = 5/14$$

Testing Phase

- Given a new instance,

$\mathbf{x}' = (\text{Outlook}=\text{Sunny}, \text{Temperature}=\text{Cool}, \text{Humidity}=\text{High}, \text{Wind}=\text{Strong})$

- Look up tables

$P(\text{Outlook}=\text{Sunny} \mid \text{Play}=\text{Yes}) = 2/9$	$P(\text{Outlook}=\text{Sunny} \mid \text{Play}=\text{No}) = 3/5$
$P(\text{Temperature}=\text{Cool} \mid \text{Play}=\text{Yes}) = 3/9$	$P(\text{Temperature}=\text{Cool} \mid \text{Play}=\text{No}) = 1/5$
$P(\text{Humidity}=\text{High} \mid \text{Play}=\text{Yes}) = 3/9$	$P(\text{Humidity}=\text{High} \mid \text{Play}=\text{No}) = 4/5$
$P(\text{Wind}=\text{Strong} \mid \text{Play}=\text{Yes}) = 3/9$	$P(\text{Wind}=\text{Strong} \mid \text{Play}=\text{No}) = 3/5$
$P(\text{Play}=\text{Yes}) = 9/14$	$P(\text{Play}=\text{No}) = 5/14$

- MAP rule

$P(\text{Yes} \mid \mathbf{x}')$: $[P(\text{Sunny} \mid \text{Yes})P(\text{Cool} \mid \text{Yes})P(\text{High} \mid \text{Yes})P(\text{Strong} \mid \text{Yes})]P(\text{Play}=\text{Yes}) =$
0.0053

$P(\text{No} \mid \mathbf{x}')$: $[P(\text{Sunny} \mid \text{No})P(\text{Cool} \mid \text{No})P(\text{High} \mid \text{No})P(\text{Strong} \mid \text{No})]P(\text{Play}=\text{No}) =$
0.0206

Given the fact $P(\text{Yes} \mid \mathbf{x}') < P(\text{No} \mid \mathbf{x}')$, we label \mathbf{x}' to be “No”.

MAP Rule

- MAP = Maximum A Posteriori
- Maximum a Posteriori or MAP for short is a Bayesian-based approach to estimating a distribution and model parameters that best explain an observed dataset.
- Maximum a Posteriori estimation is a probabilistic framework for solving the problem of density estimation.
- MAP involves calculating a conditional probability of observing the data given a model weighted by a prior probability or belief about the model.
- MAP provides an alternate probability framework to maximum likelihood estimation for machine learning.

The Zero Frequency Problem

- If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as Zero Frequency.
- To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.

Types of NB

- **Gaussian**: It is used in classification and it assumes that features follow a normal distribution.
- **Multinomial**: It is used for discrete counts. For example, let's say, we have a text classification problem. Here we can consider Bernoulli trials which is one step further and instead of "word occurring in the document", we have "count how often word occurs in the document", you can think of it as "number of times outcome number x_i is observed over the n trials".
- **Bernoulli**: The binomial model is useful if your feature vectors are binary (i.e. zeros and ones). One application would be text classification with 'bag of words' model where the 1s & 0s are "word occurs in the document" and "word does not occur in the document" respectively.

Advantages

- Naïve Bayes based on the independence assumption
 - Training is very easy and fast; just requiring considering each attribute in each class separately
 - Test is straightforward; just looking up tables or calculating conditional probabilities with normal distributions
 - Works well with less data!
- A popular generative model
 - Performance competitive to most of state-of-the-art classifiers even in presence of violating independence assumption
 - Many successful applications, e.g., spam mail filtering
 - A good candidate of a base learner in ensemble learning

Applications

- Text Classification
- Spam Filtering
- Hybrid Recommender System
 - Recommender Systems apply machine learning and data mining techniques for filtering unseen information and can predict whether a user would like a given resource
- Online Application
 - Simple Emotion Modeling

Text Classification

- Learning which articles are of interest
- Classify web pages by topic
- Information extraction
- Internet filters

Examples of Text Classification

- CLASSES=BINARY

- “spam” / “not spam”

- CLASSES =TOPICS

- “finance” / “sports” / “politics”

- CLASSES =OPINION

- “like” / “hate” / “neutral”

- CLASSES =TOPICS

- “AI” / “Theory” / “Graphics”

- CLASSES =AUTHOR

- “Shakespeare” / “Marlowe” / “Ben Jonson”

Naïve Bayes Approach

- Build the Vocabulary as the list of all distinct words that appear in all the documents of the training set.
- Remove stop words and markings
- The words in the vocabulary become the attributes, assuming that classification is independent of the positions of the words
- Each document in the training set becomes a record with frequencies for each word in the Vocabulary.
- Train the classifier based on the training data set, by computing the prior probabilities for each class and attributes.
- Evaluate the results on Test data

Representing Text: List of Words

$f(\text{[raw text snippet]}) = y$

$f(\text{[refined text snippet]}) = y$

- Common Refinements: Remove Stop Words, Symbols

Text Classification

$$\hat{P}(t|c) = \frac{T_{ct} + 1}{\sum_{t' \in V} (T_{ct'} + 1)} = \frac{T_{ct} + 1}{(\sum_{t' \in V} T_{ct'}) + B'}$$

- T_{ct} — Number of particular word in particular class
- $T_{ct'}$ — Number of total words in particular class
- B' — Number of distinct words in all class

Example

► Table 13.1 Data for parameter estimation examples.

	docID	words in document	in $c = \textit{China}$?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

$$\hat{P}(\textit{Chinese}|c)$$

$$\hat{P}(\textit{Tokyo}|c) = \hat{P}(\textit{Japan}|c)$$

$$\hat{P}(\textit{Chinese}|\vec{c})$$

$$\hat{P}(\textit{Tokyo}|\vec{c}) = \hat{P}(\textit{Japan}|\vec{c})$$

Example

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

$$\hat{P}(c|d_5) \propto 3/4 \cdot (3/7)^3 \cdot 1/14 \cdot 1/14 \approx 0.0003.$$

$$\hat{P}(\bar{c}|d_5) \propto 1/4 \cdot (2/9)^3 \cdot 2/9 \cdot 2/9 \approx 0.0001.$$

○Probability of Yes Class is more than that of No Class, Hence this document will go into Yes Class.

Naïve Bayes Results

- F1 score should be as close to 1 as possible
- Kappa Value (cohen_kappa_score) should be at least 0.9
- Precision and Recall
- Accuracy via confusion matrix

Tips to improve NB model

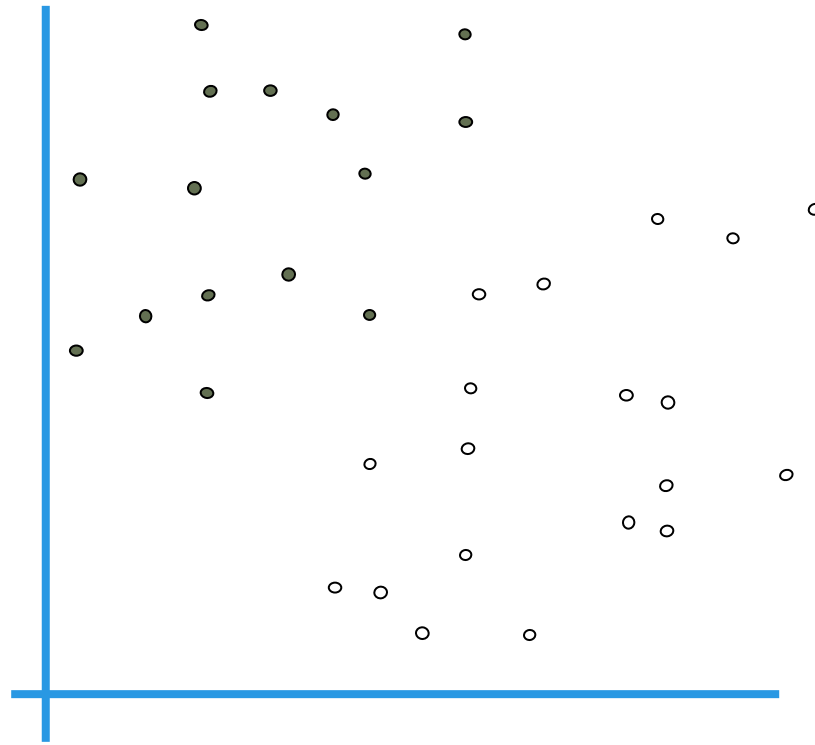
- If continuous features do not have normal distribution, we should use transformation or different methods to convert it in normal distribution.
- If test data set has zero frequency issue, apply smoothing techniques “Laplace Smoothing” to predict the class of test data set.
- Remove correlated features, as the highly correlated features are voted twice in the model and it can lead to over inflating importance.
- Naive Bayes classifiers has limited options for parameter tuning like `alpha=1` for smoothing, `fit_prior=[True|False]` to learn class prior probabilities or not and some other options. Focus on your pre-processing of data and the feature selection.
- You might think to apply some *classifier combination technique like* ensembling, bagging and boosting but these methods would not help. Actually, “ensembling, boosting, bagging” won’t help since their purpose is to reduce variance. Naive Bayes has no variance to minimize.

Hands on

- <https://github.com/2796gaurav/Naive-bayes-explained/tree/master/Naive%20bayes>
- <https://www.datacamp.com/community/tutorials/naive-bayes-scikit-learn>
- https://chrisalbon.com/machine_learning/naive_bayes/multinomial_naive_bayes_classifier/
- <https://towardsdatascience.com/comparing-a-variety-of-naive-bayes-classification-algorithms-fc5fa298379e>

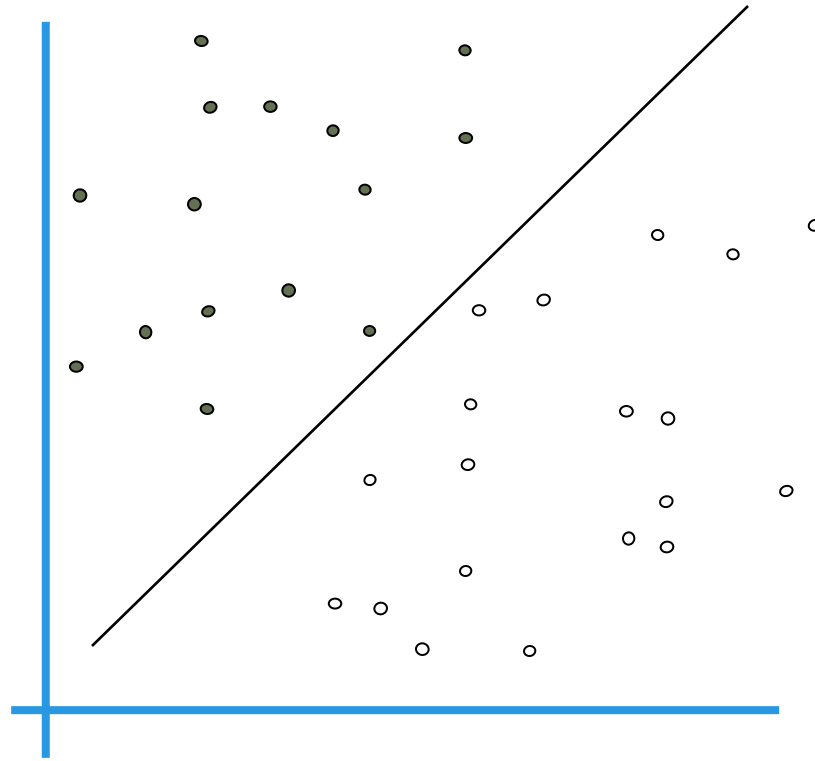
Support Vector Machines

➤ Say your dataset looks like this –



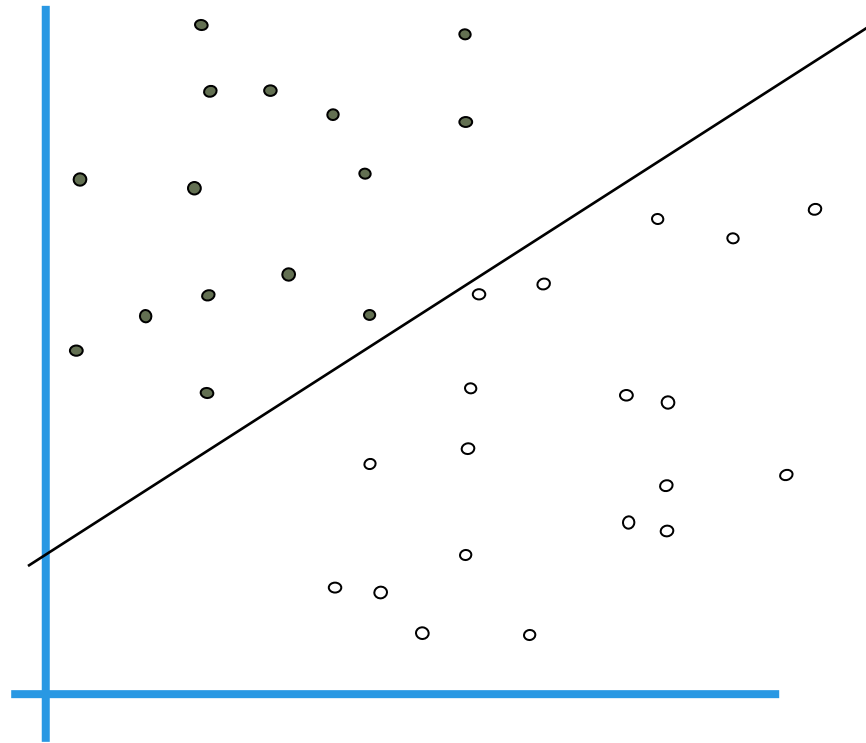
Support Vector Machines

➤ How can we separate this?



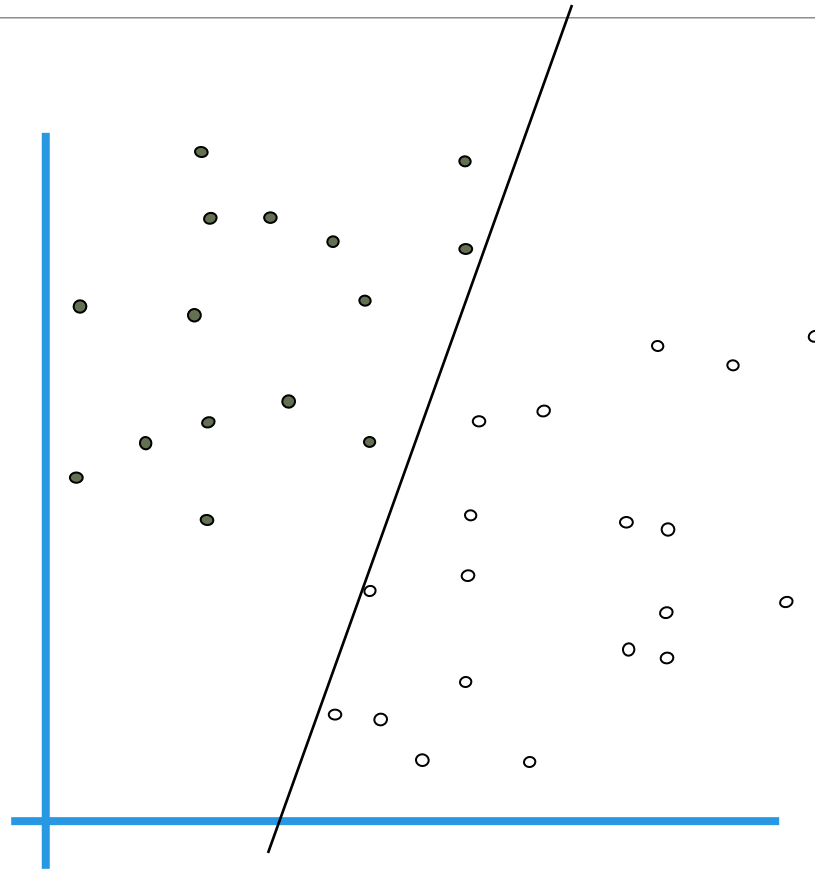
Support Vector Machines

➤ How about this?



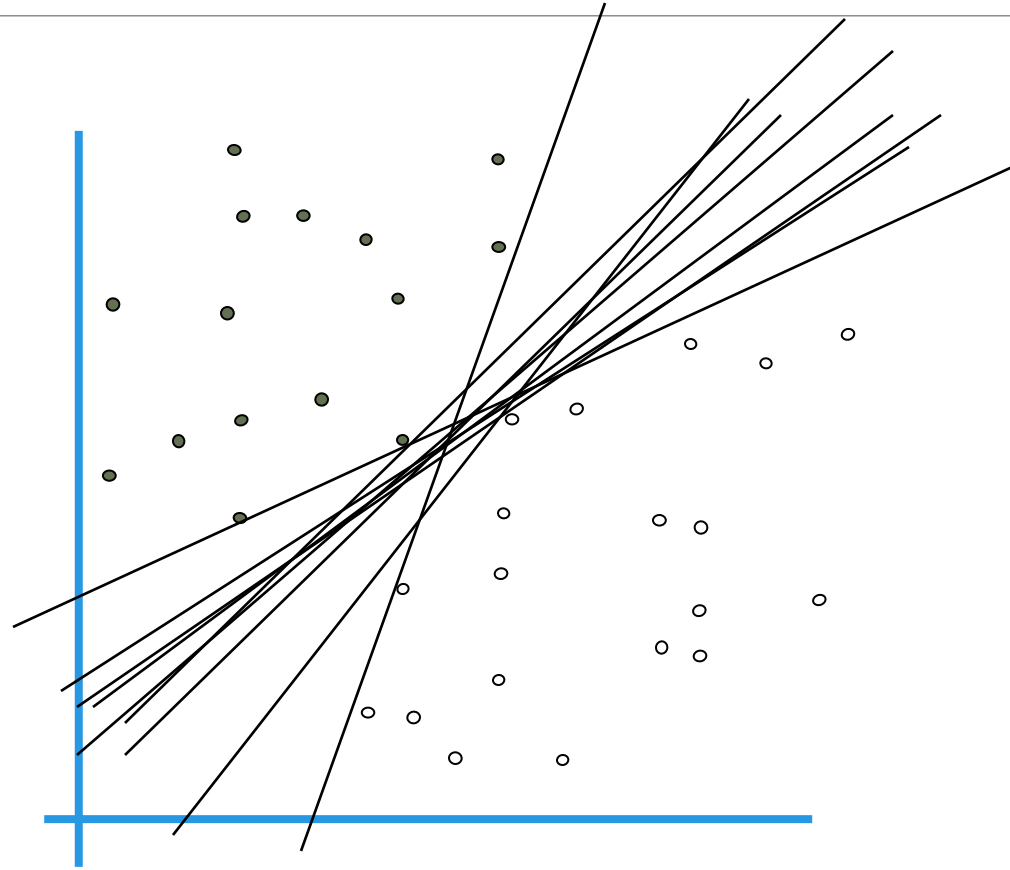
Support Vector Machines

➤ Or this?



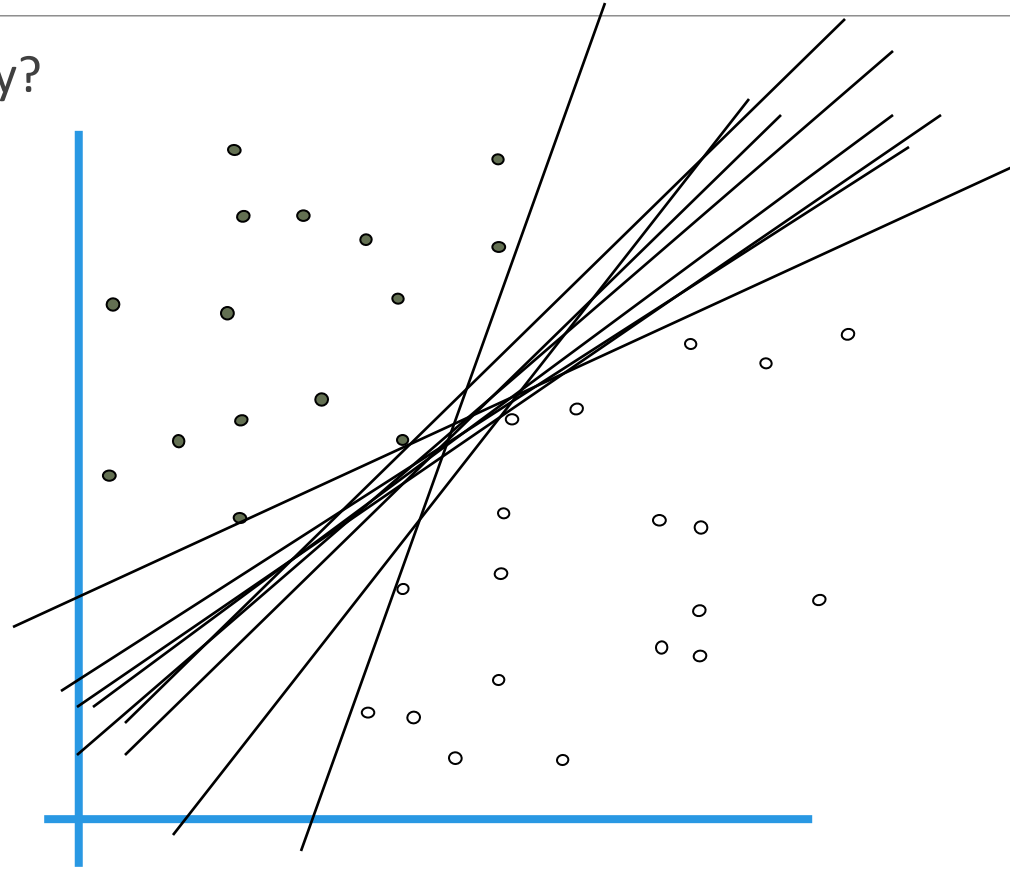
Support Vector Machines

➤ Infinite ways!



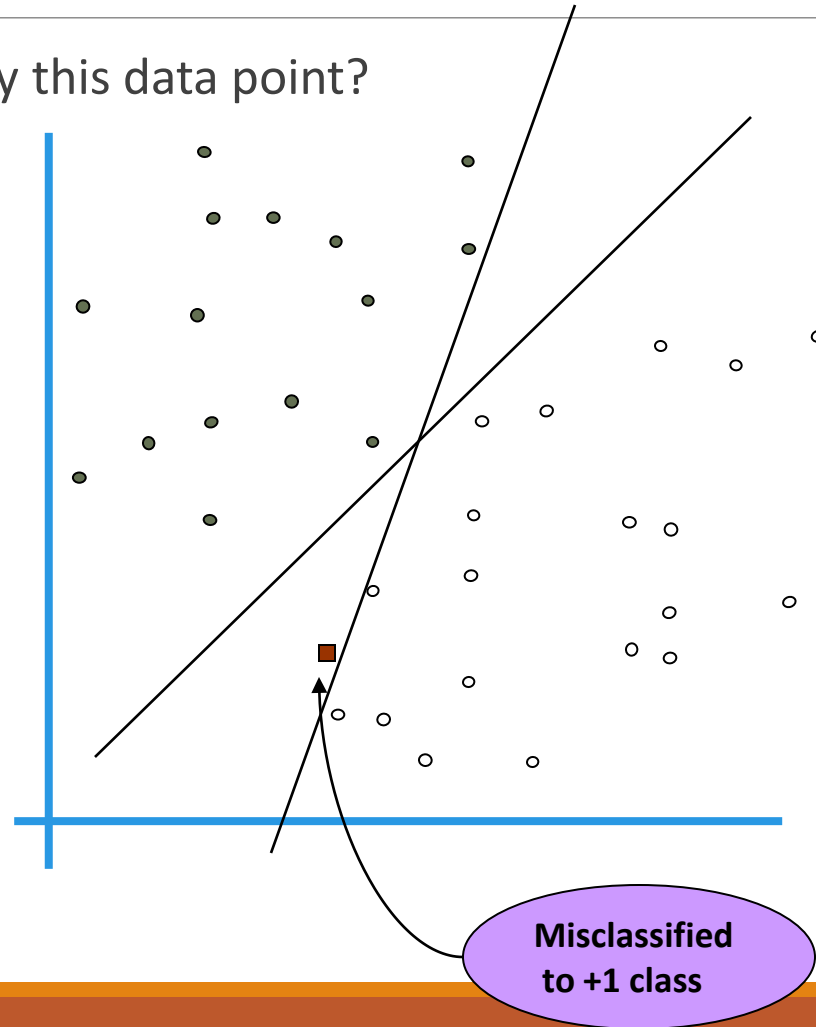
Support Vector Machines

➤ Which is the best way?



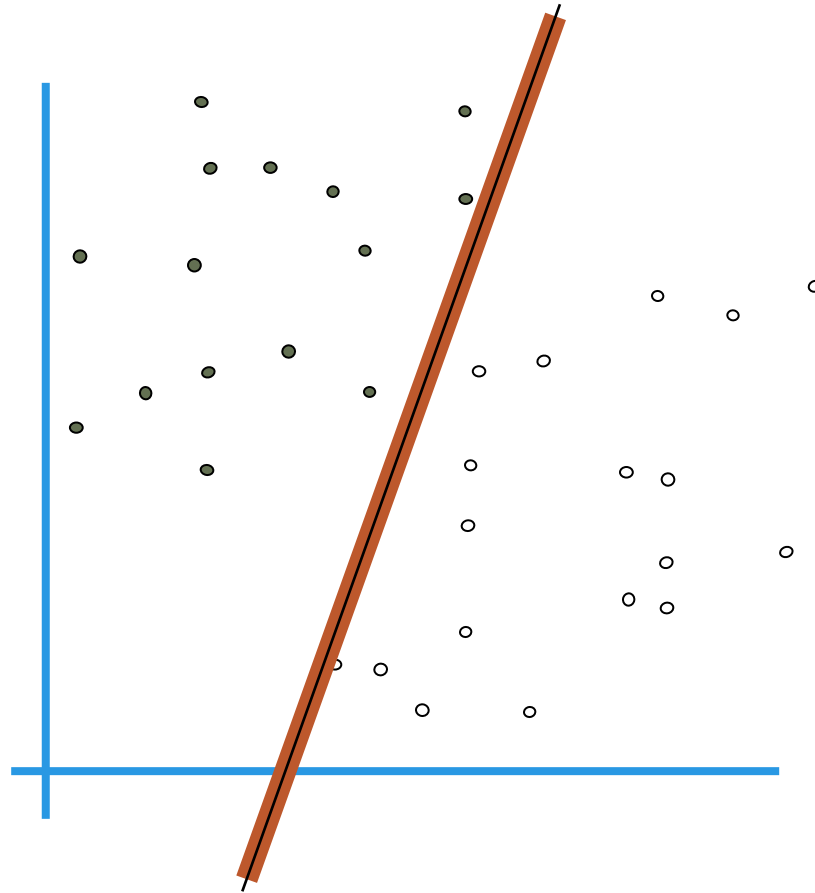
Support Vector Machines

➤ How would you classify this data point?



Support Vector Machines

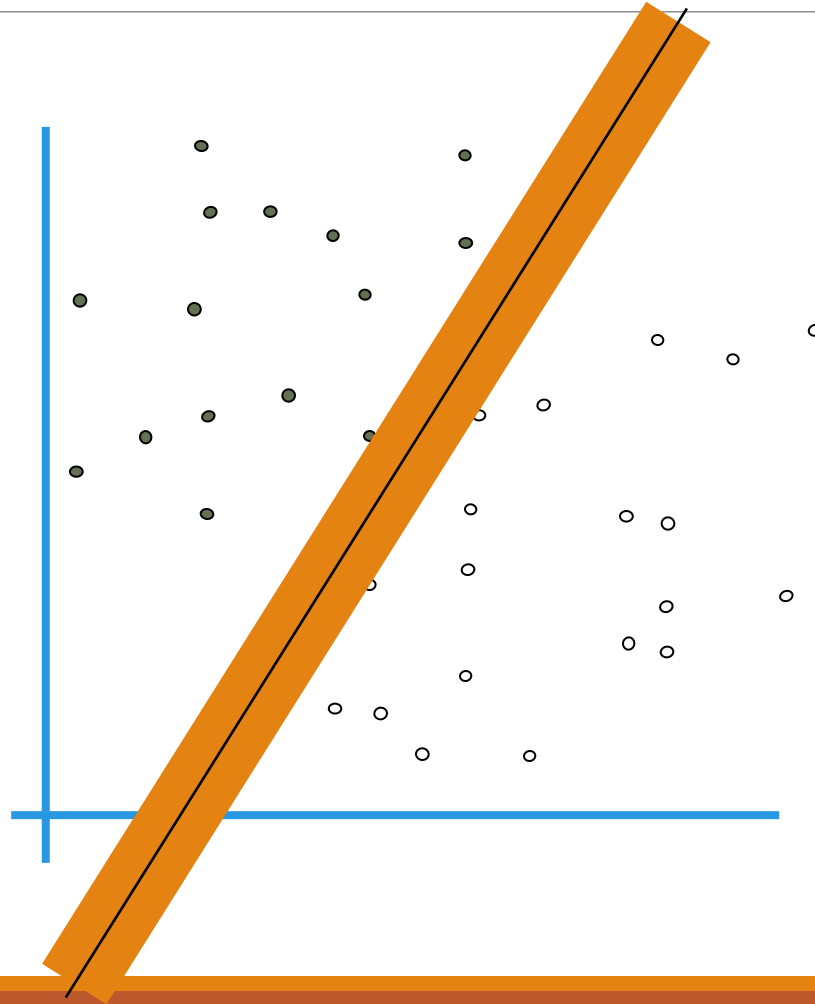
- Define the **margin** of a linear classifier as the width that the boundary could be increased by before hitting a data point.



Support Vector Machines

➤ Maximum Margin

1. Maximizing the margin is good
2. Implies that only support vectors are important; other training examples are ignorable.

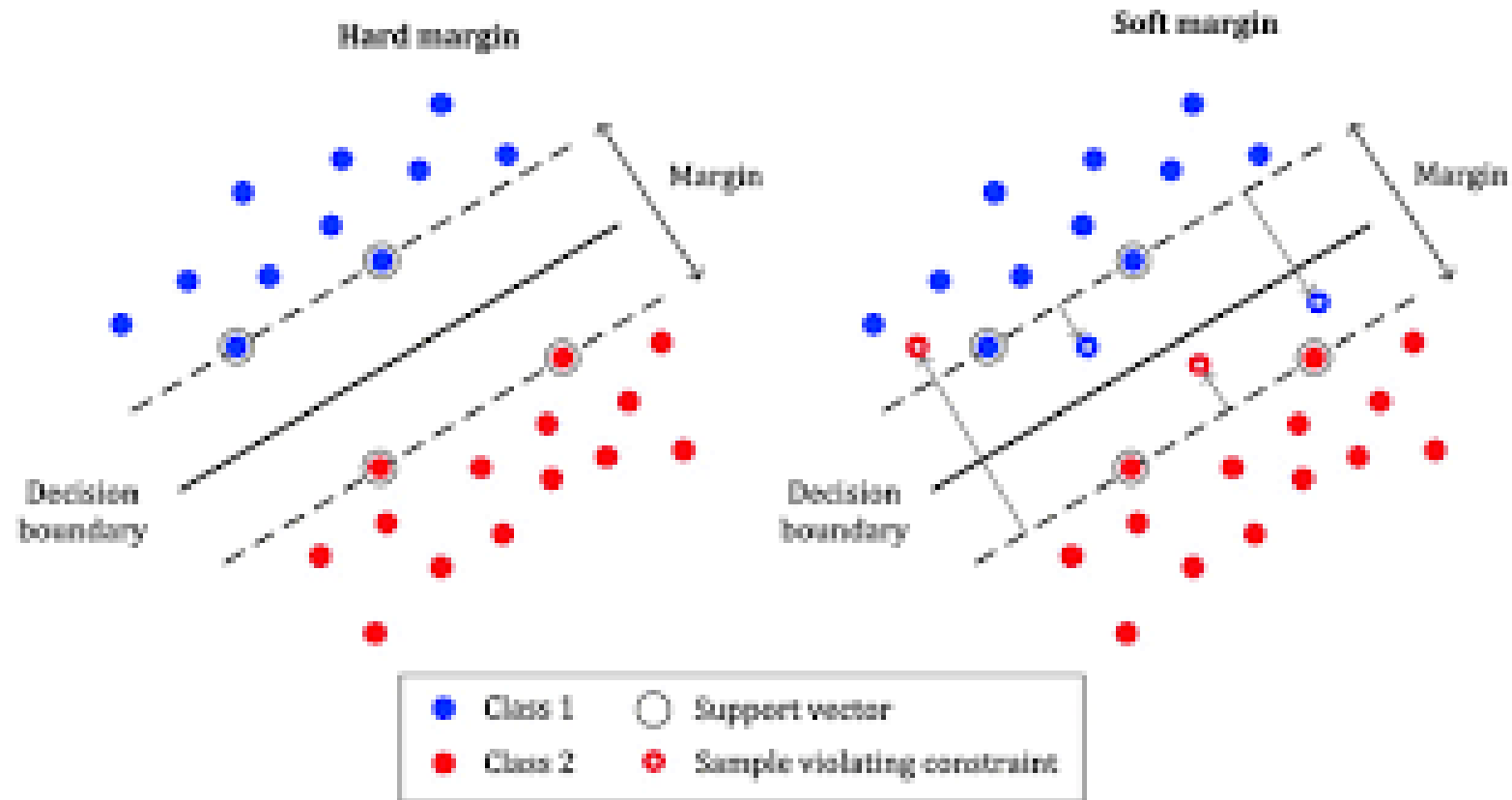


Support Vectors are those datapoints that the margin pushes up against

The **maximum margin linear classifier** is the linear classifier with the, um, maximum margin.

This is the simplest kind of SVM (called an LSVM)

Hard Margin Vs. Soft Margin

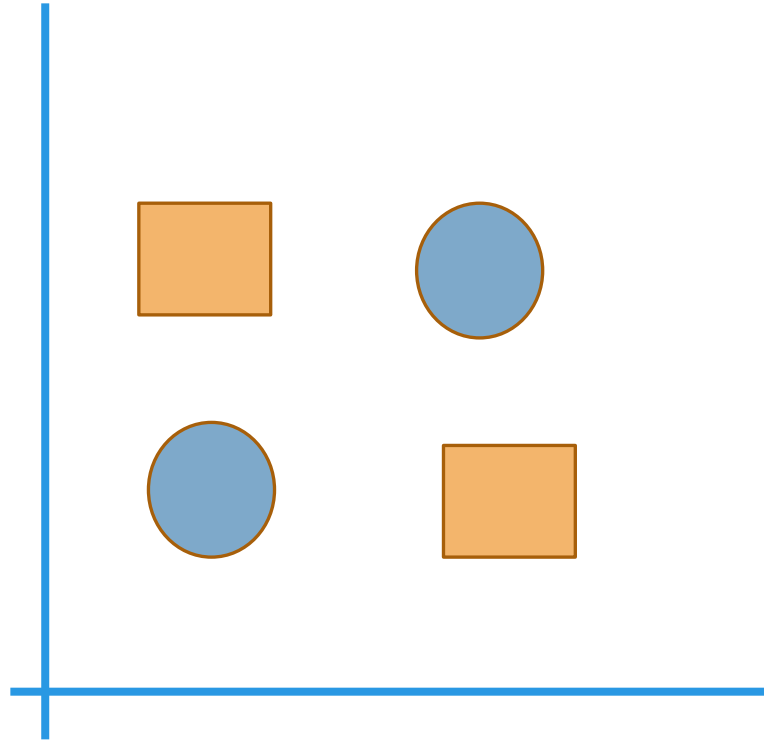


Hard Margin Vs. Soft Margin

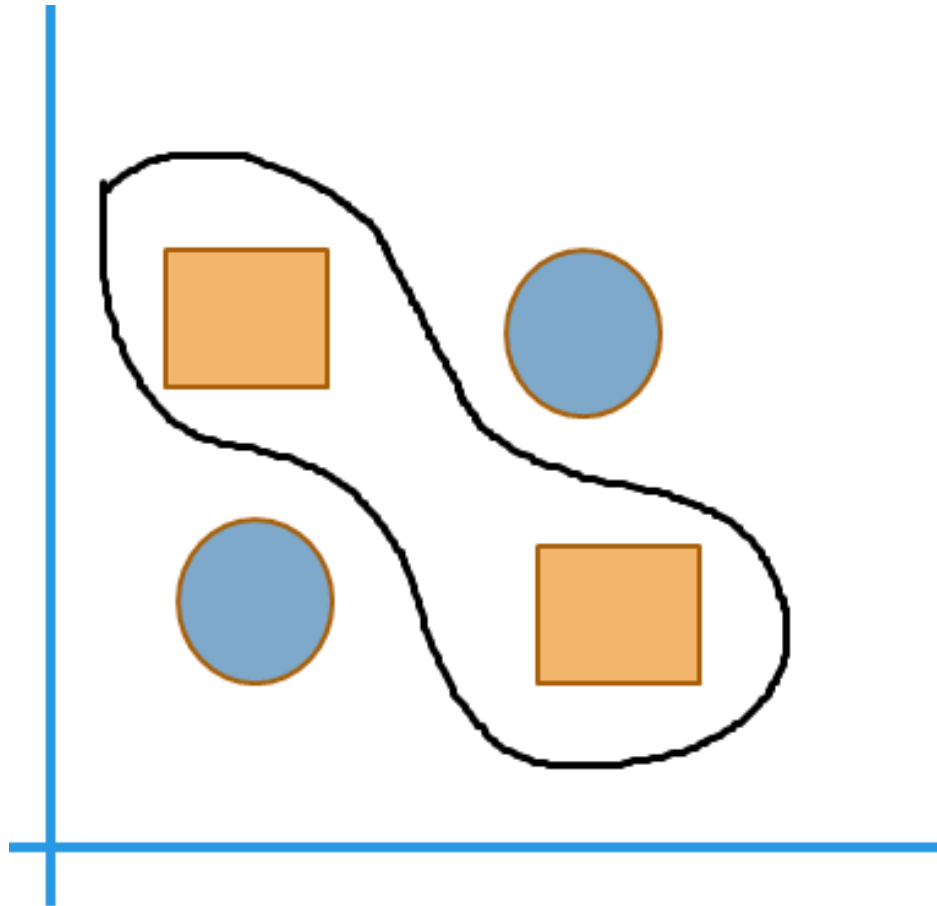
- **Hard margin SVM** can work only when data is completely linearly separable without any errors (noise or outliers).
- In case of errors either the **margin** is smaller or **hard margin SVM** fails.
- On the other hand **soft margin SVM** solves this problem by introducing slack variables.

Kernels

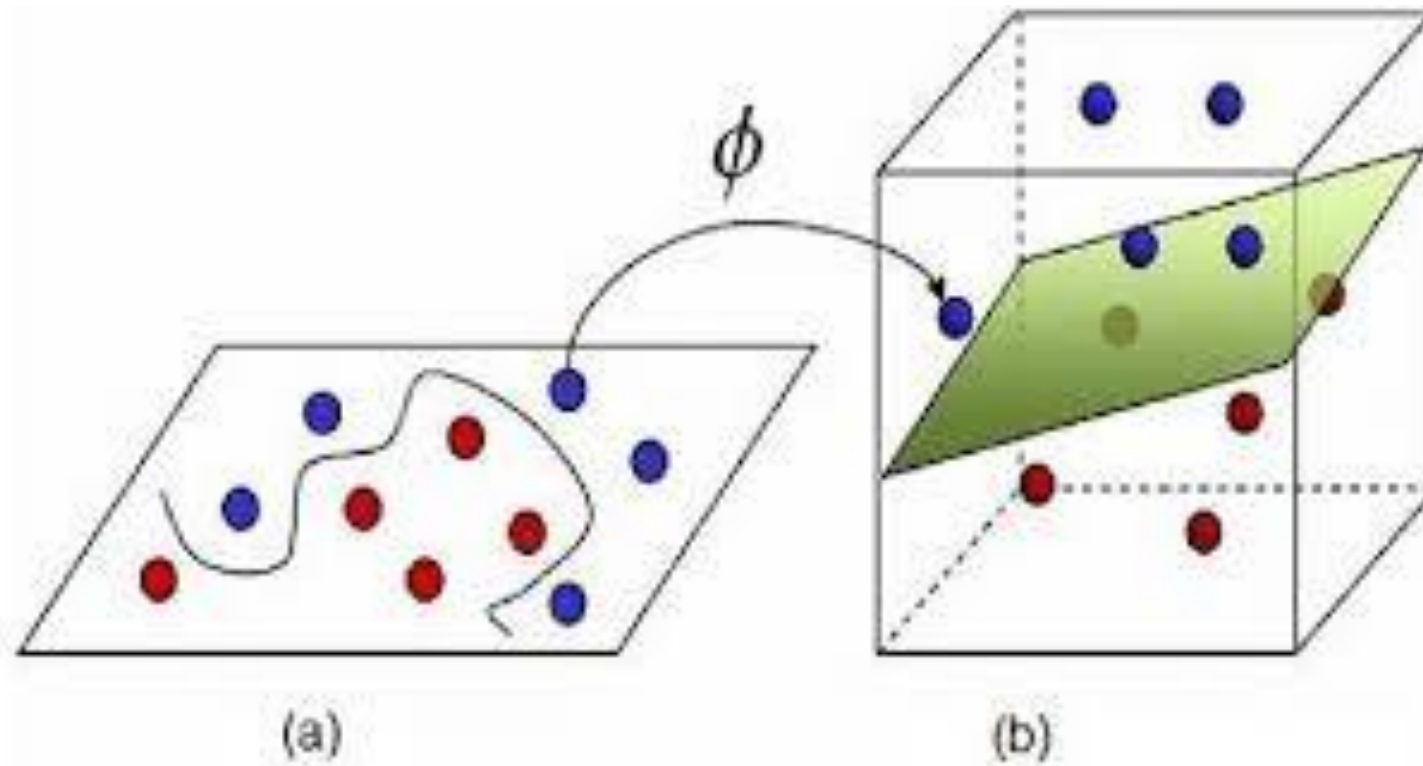
- Feature Projections to higher dimension
- What??
- Say your data is like this –
- How can you classify?



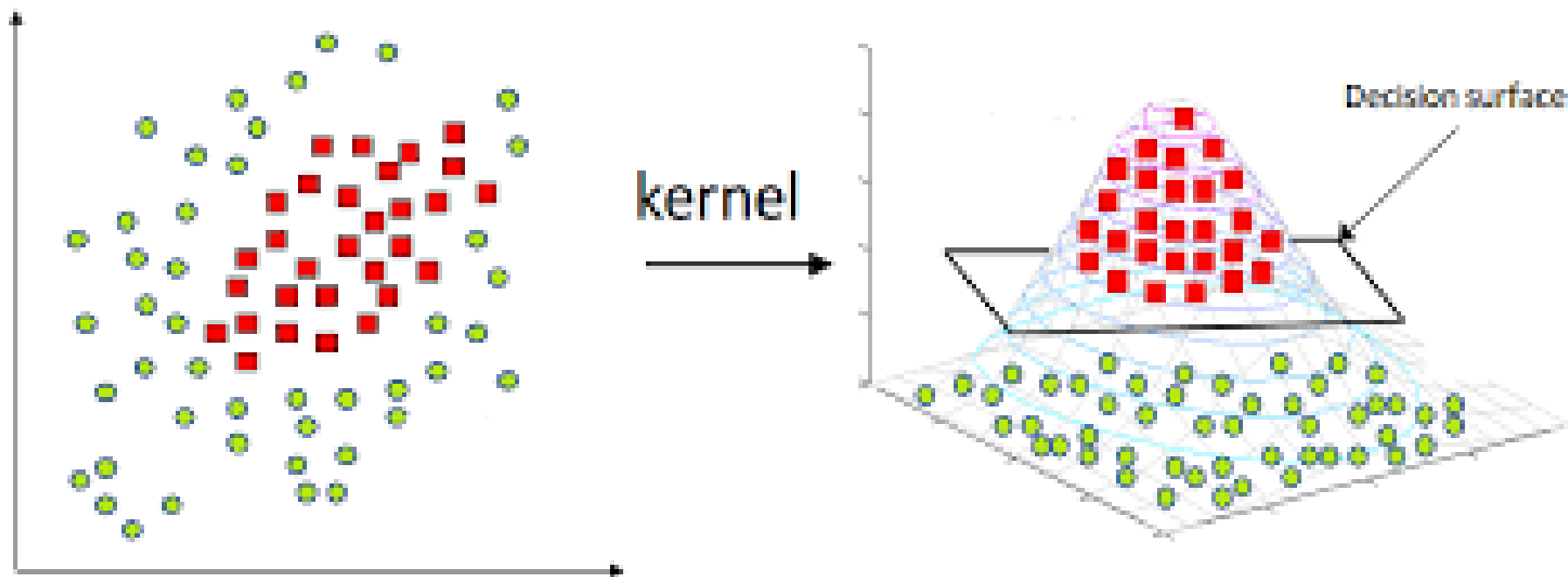
Kernels



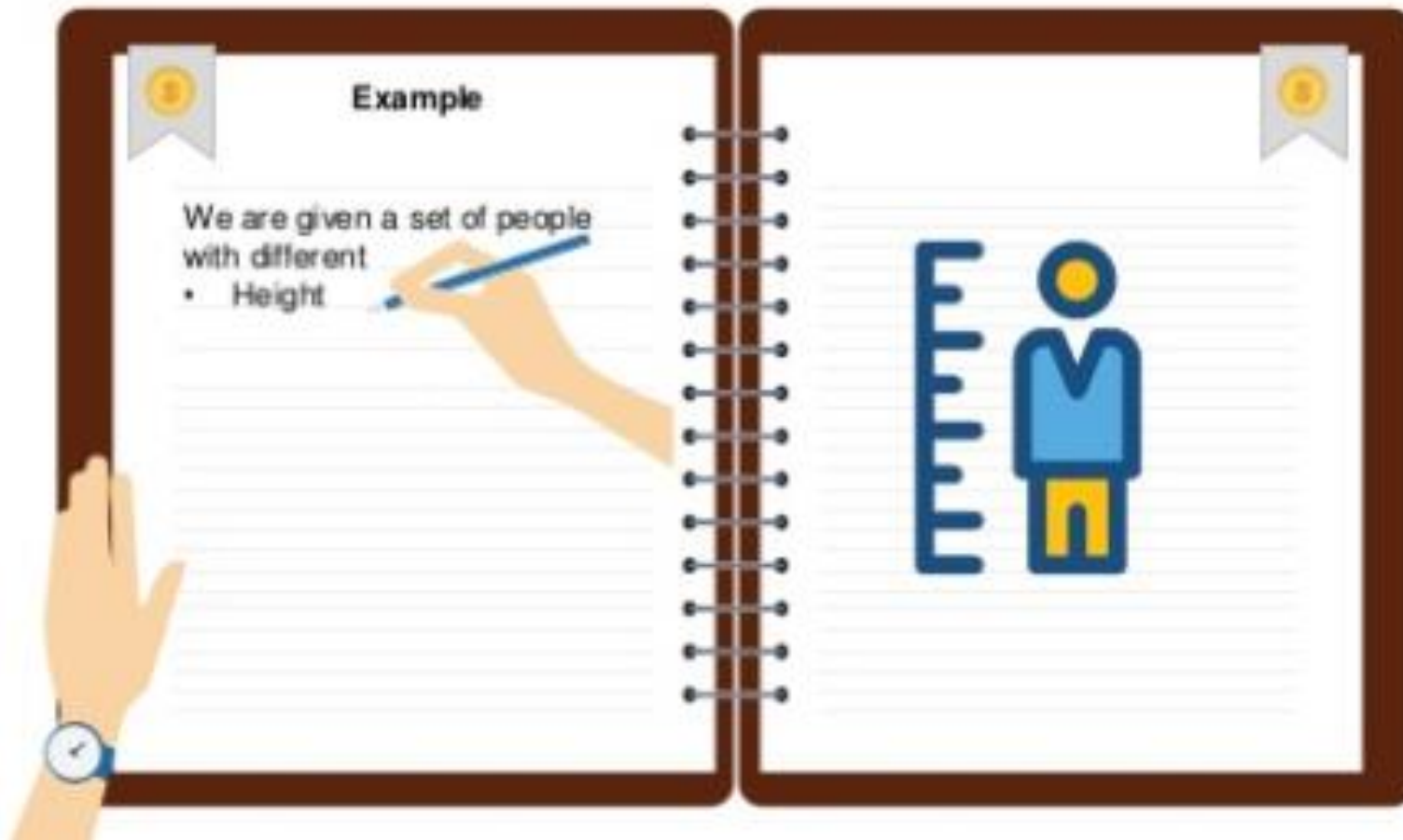
Kernels



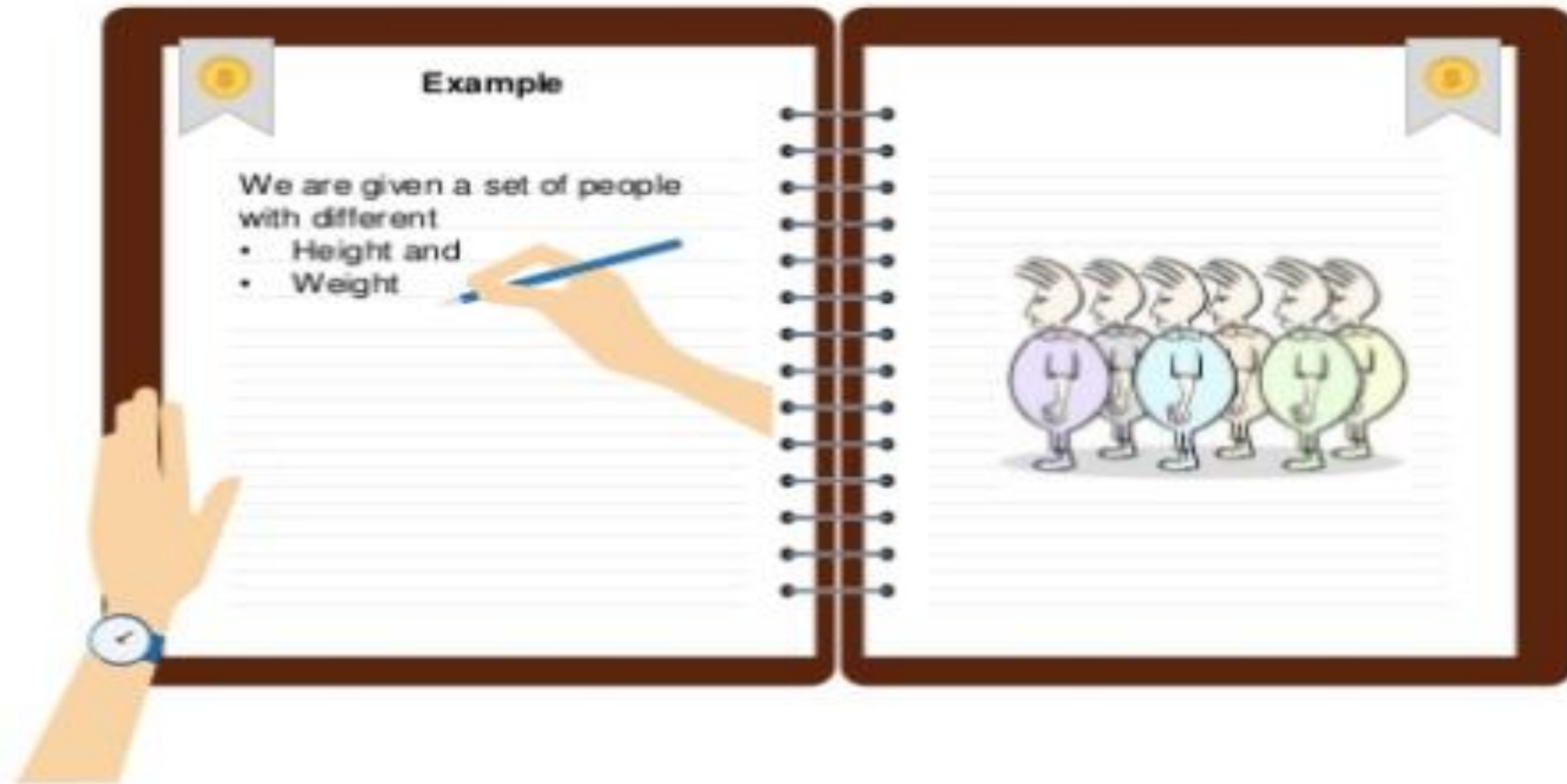
Kernels



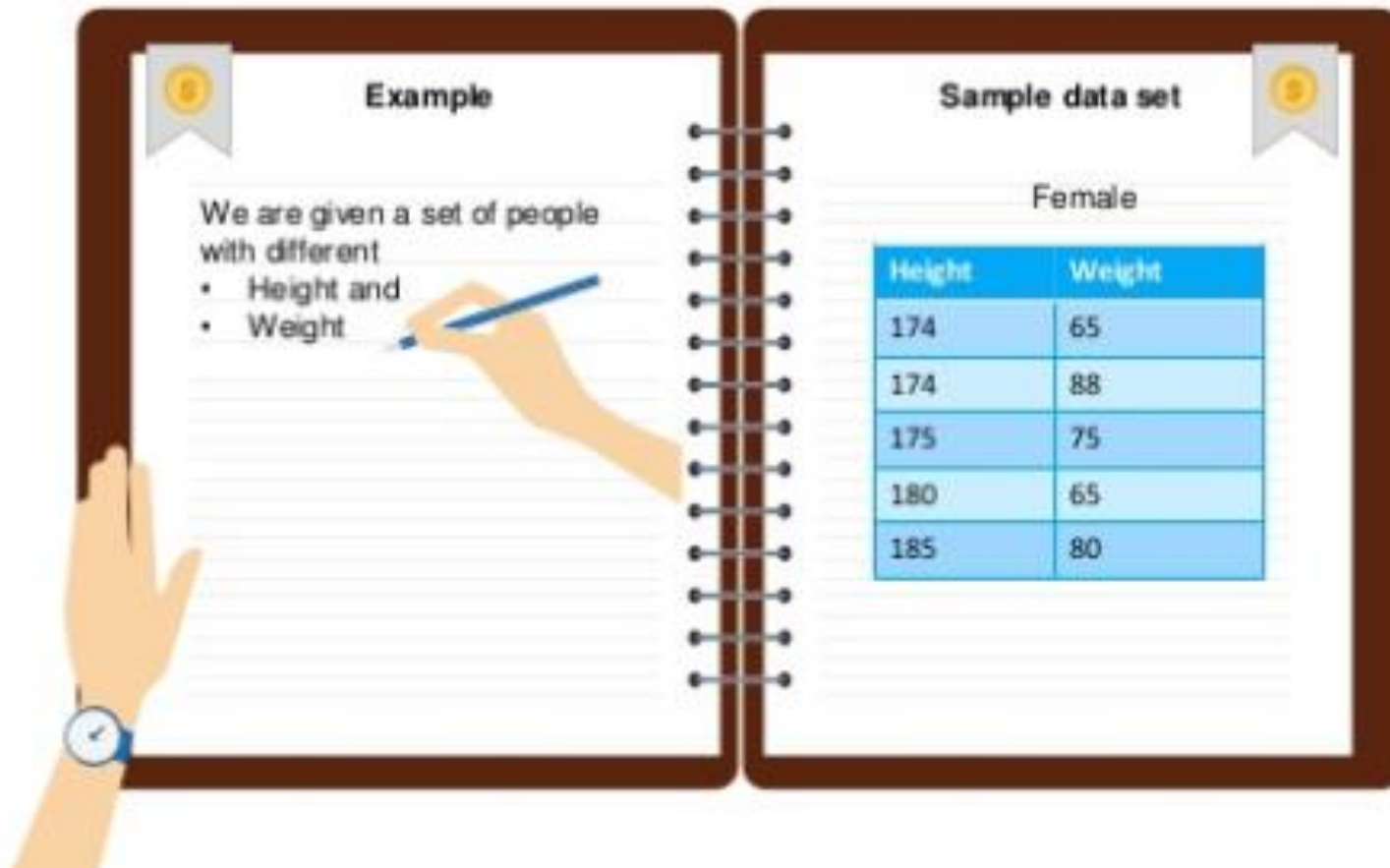
SVM – Another try



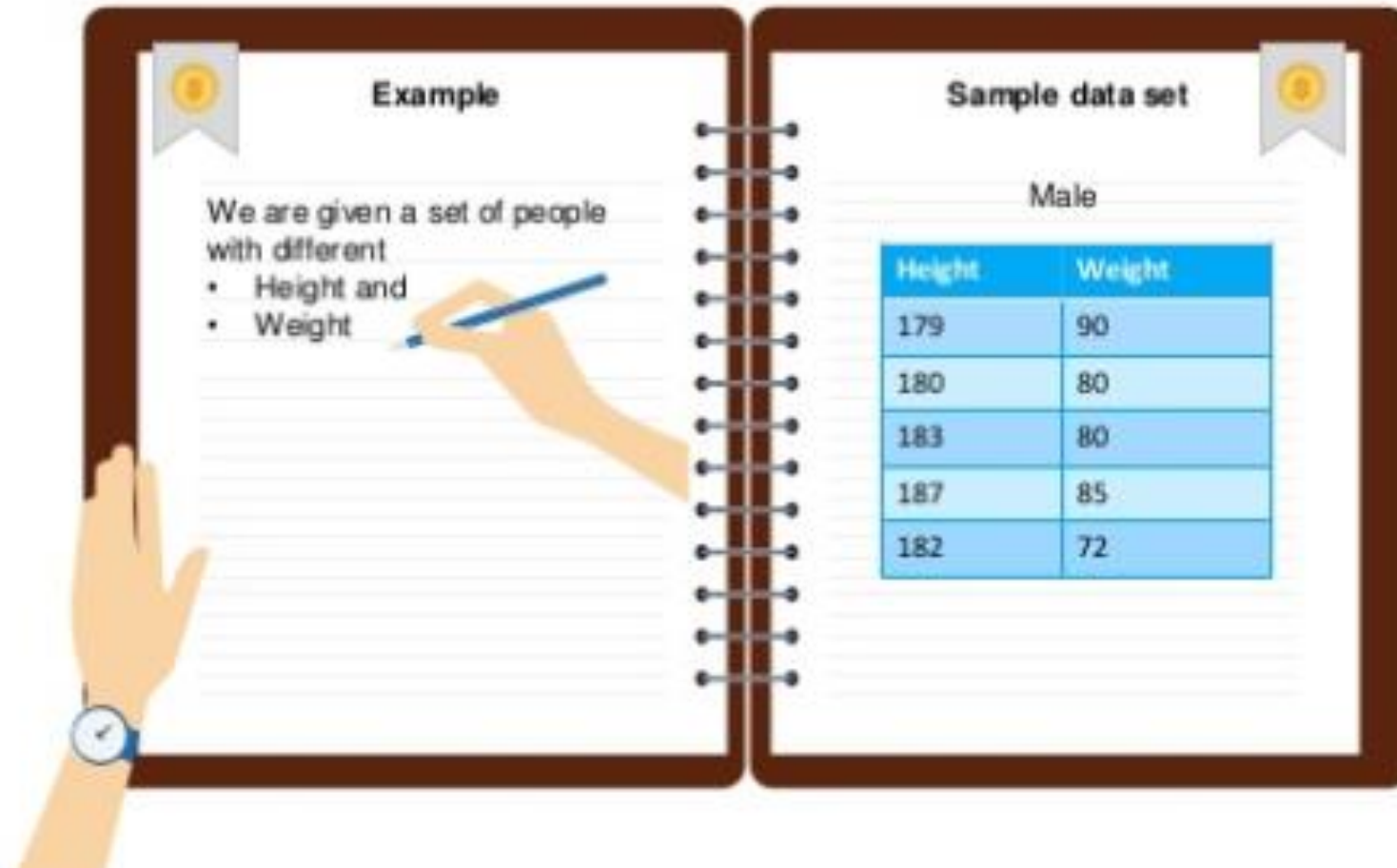
SVM – Another try



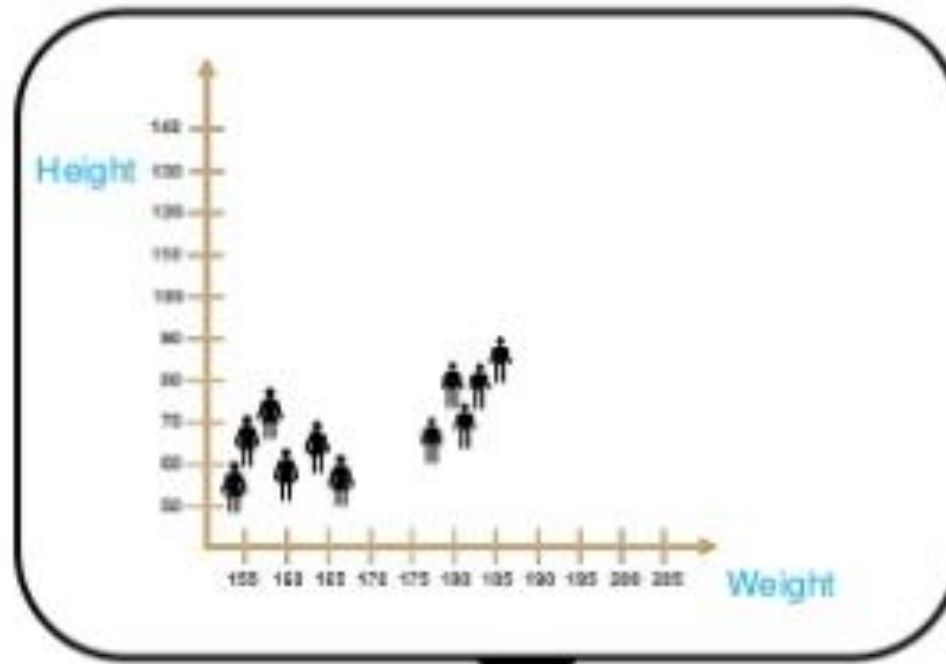
SVM – Another try



SVM – Another try



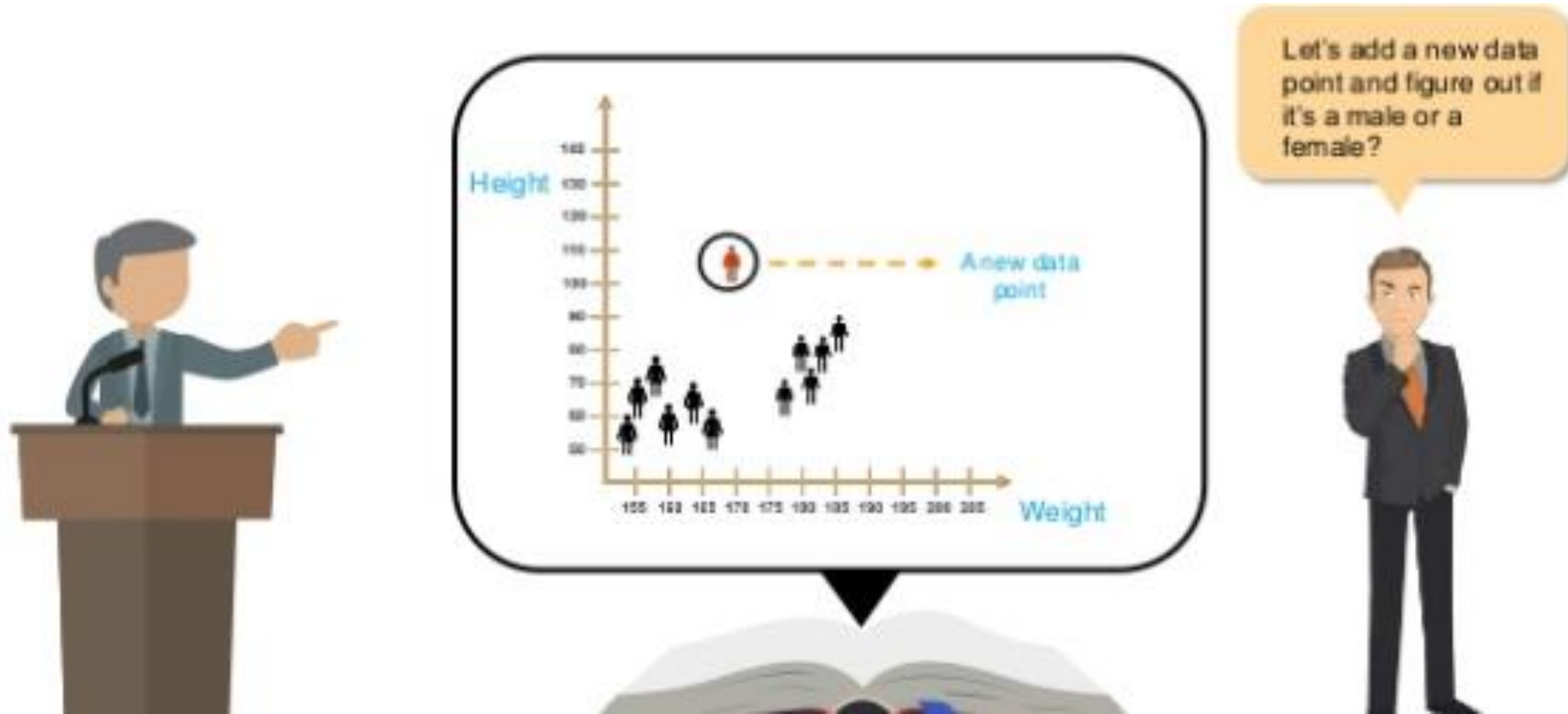
SVM – Another try



Let's add a new data point and figure out if it's a male or a female?

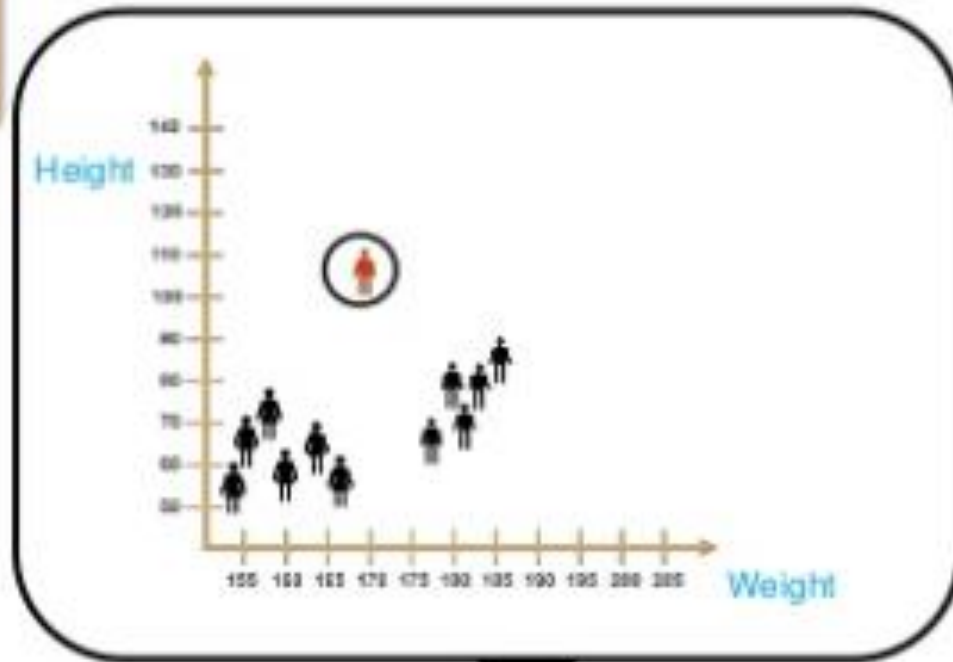


SVM – Another try



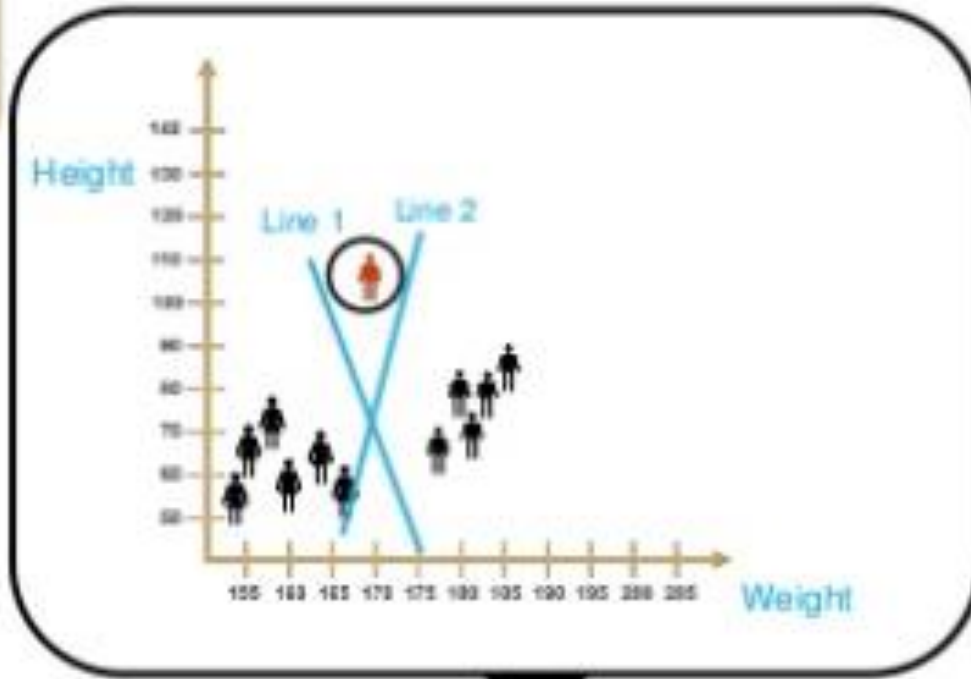
SVM – Another try

Sure.. For this task, we need to split our data first



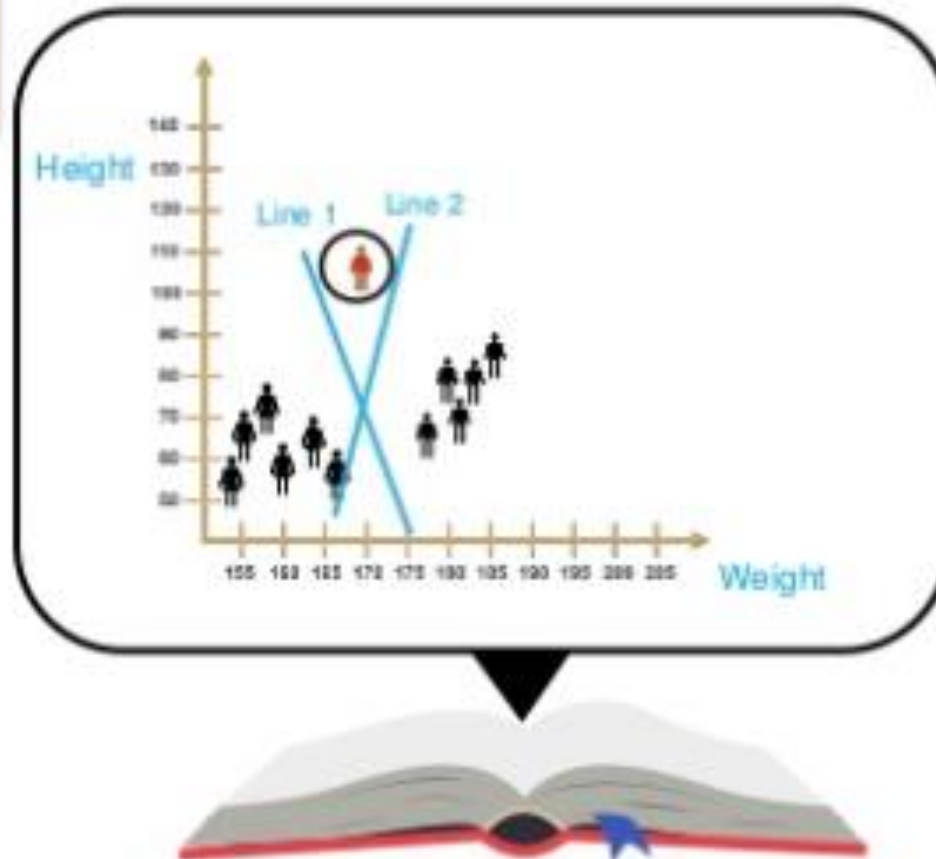
SVM – Another try

We can split our data by choosing any of these lines



SVM – Another try

But to predict the gender of a new data point we should split the data in the best possible way



SVM – Another try

Then I would say, this line best splits the data

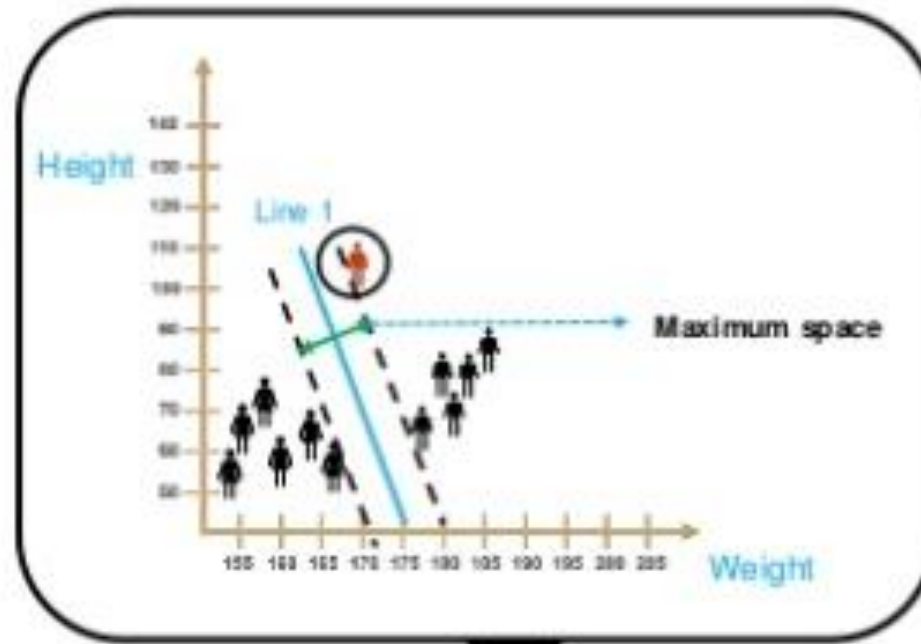


Why do you say it's the best split??



SVM – Another try

This line has the maximum space that separates the two classes

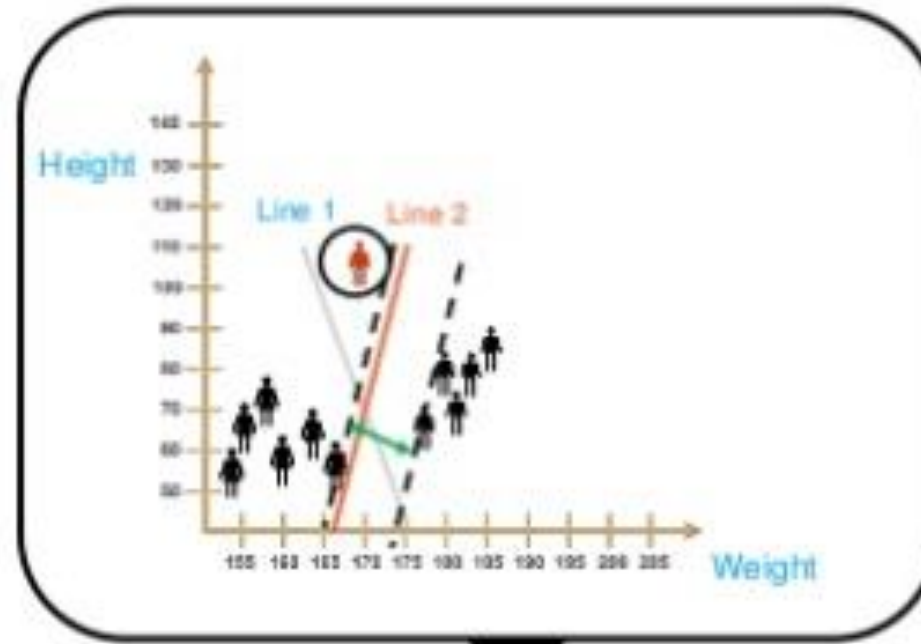


Why do you say it's the best split?



SVM – Another try

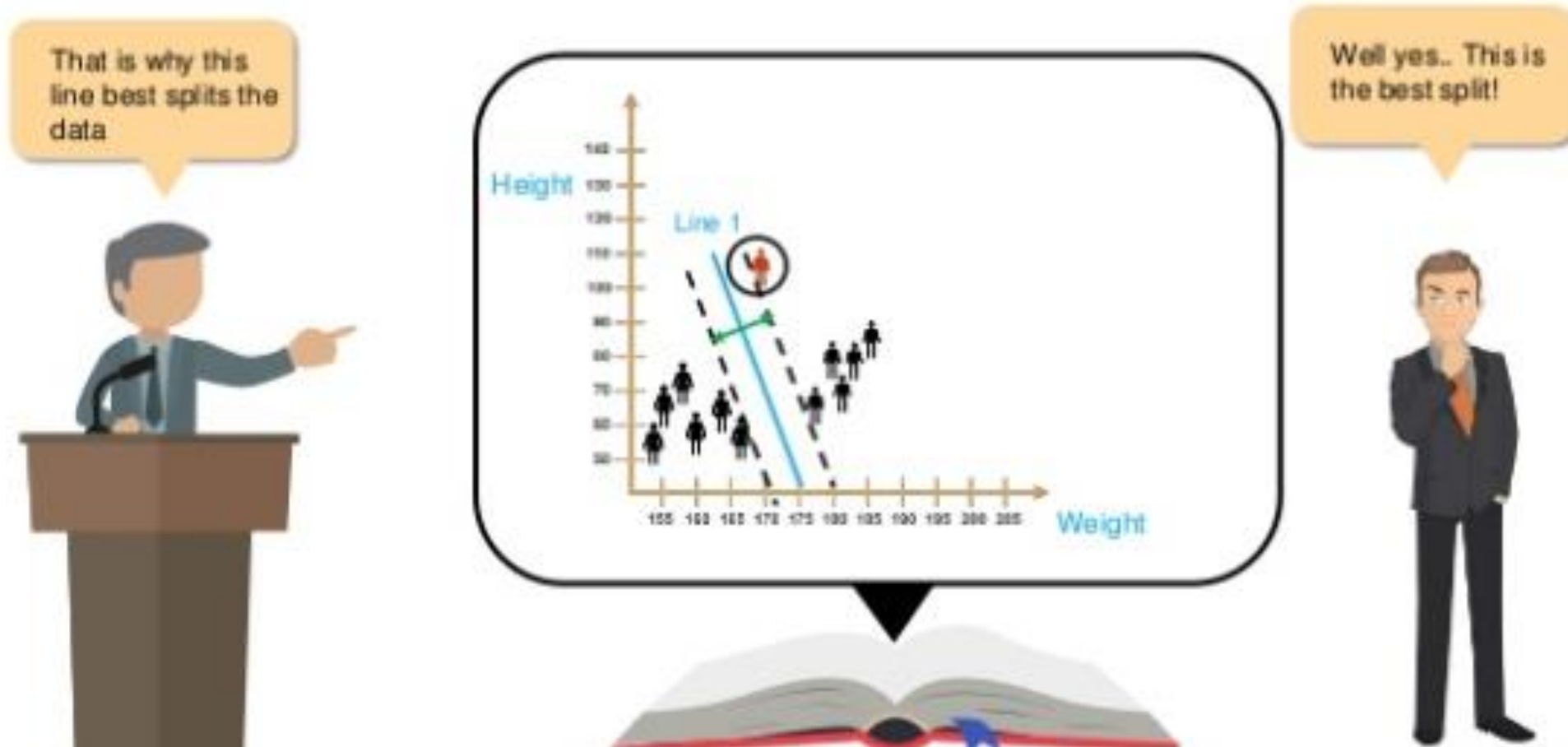
While the other line doesn't have the maximum space that separates the two classes



Why do you say it's the best split??

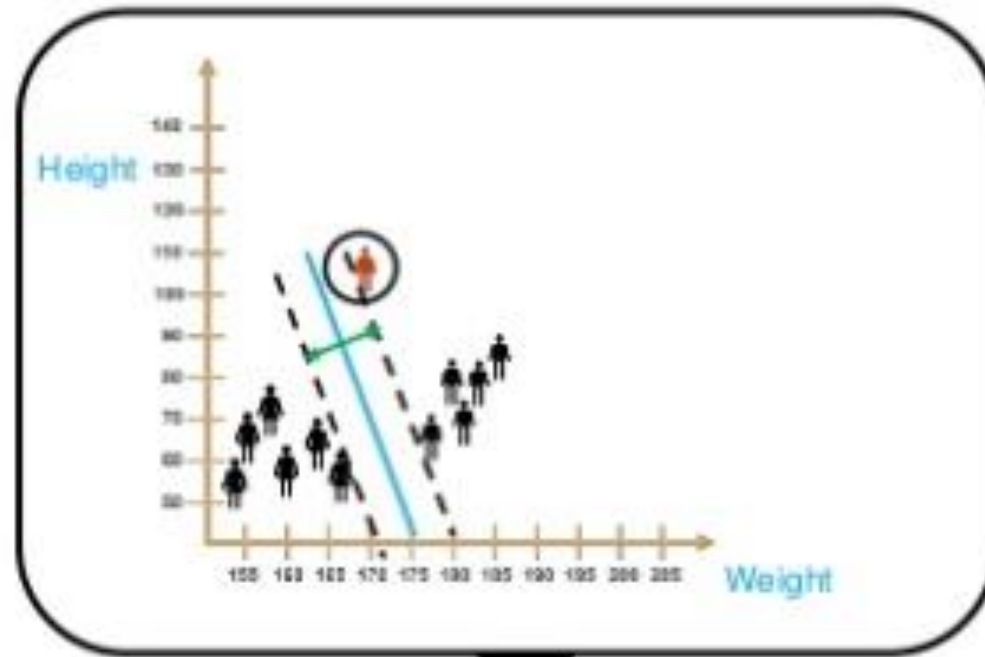


SVM – Another try



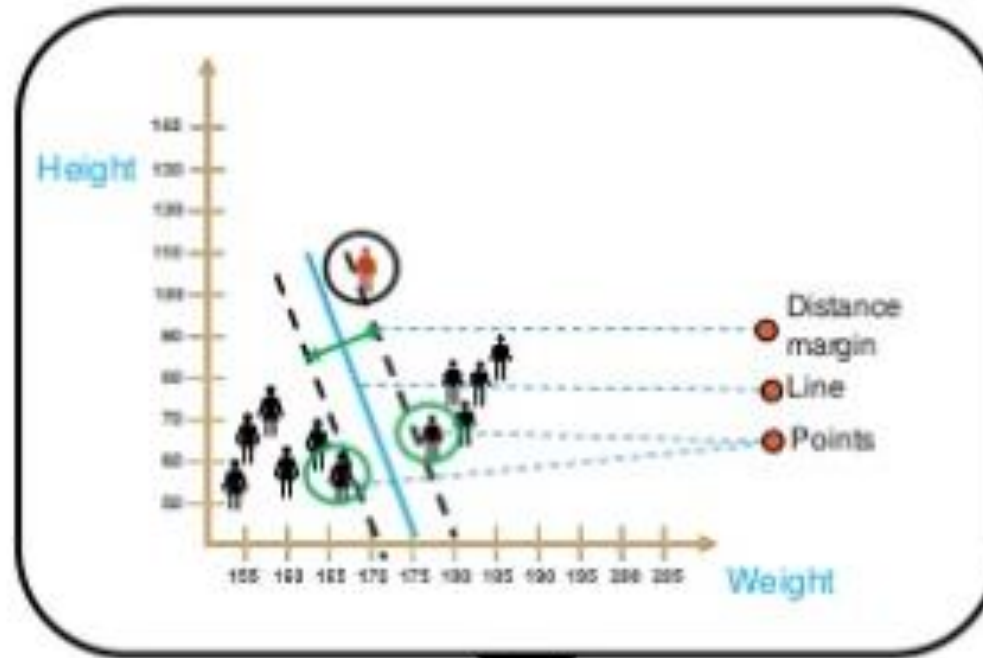
SVM – Another try

Now, Let me add some technical terms to this



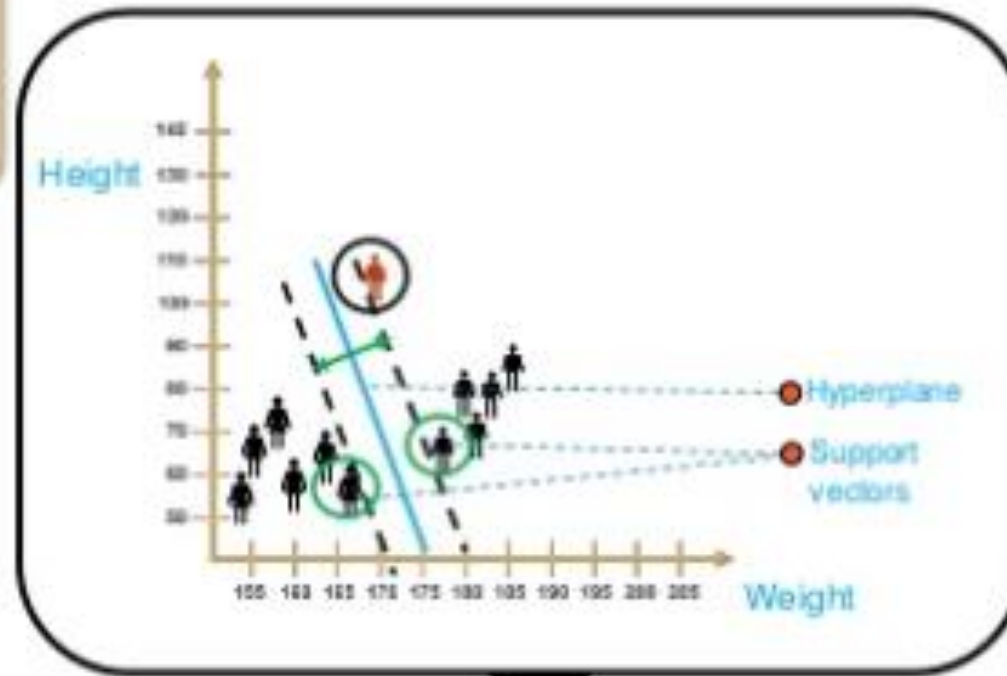
SVM – Another try

We can also say that the distance between the points and the line should be far as possible



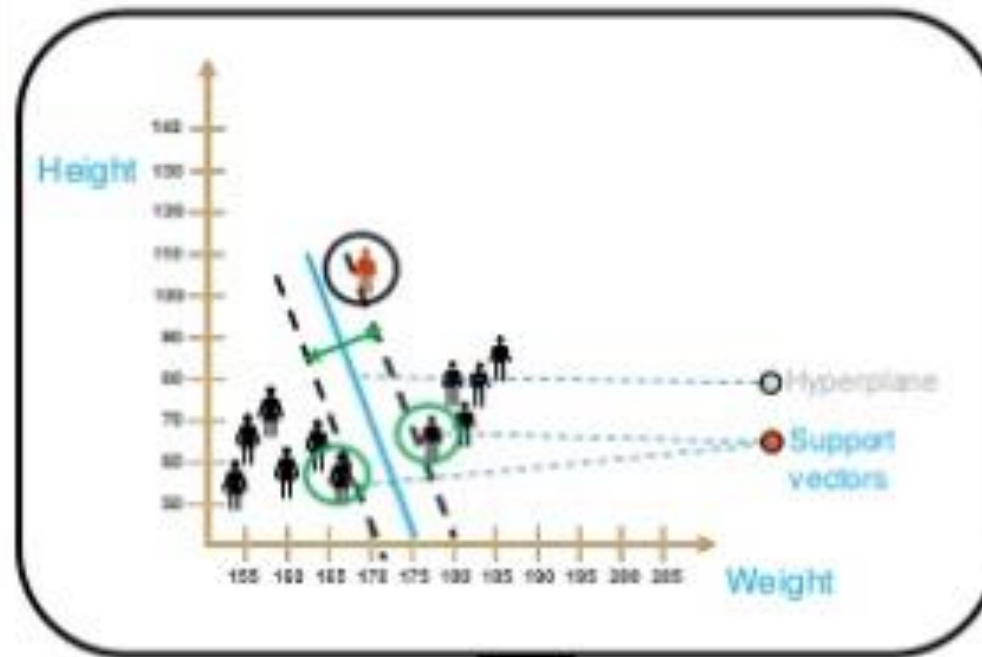
SVM – Another try

In technical terms we can say, the distance between the support vector and the hyperplane should be far as possible



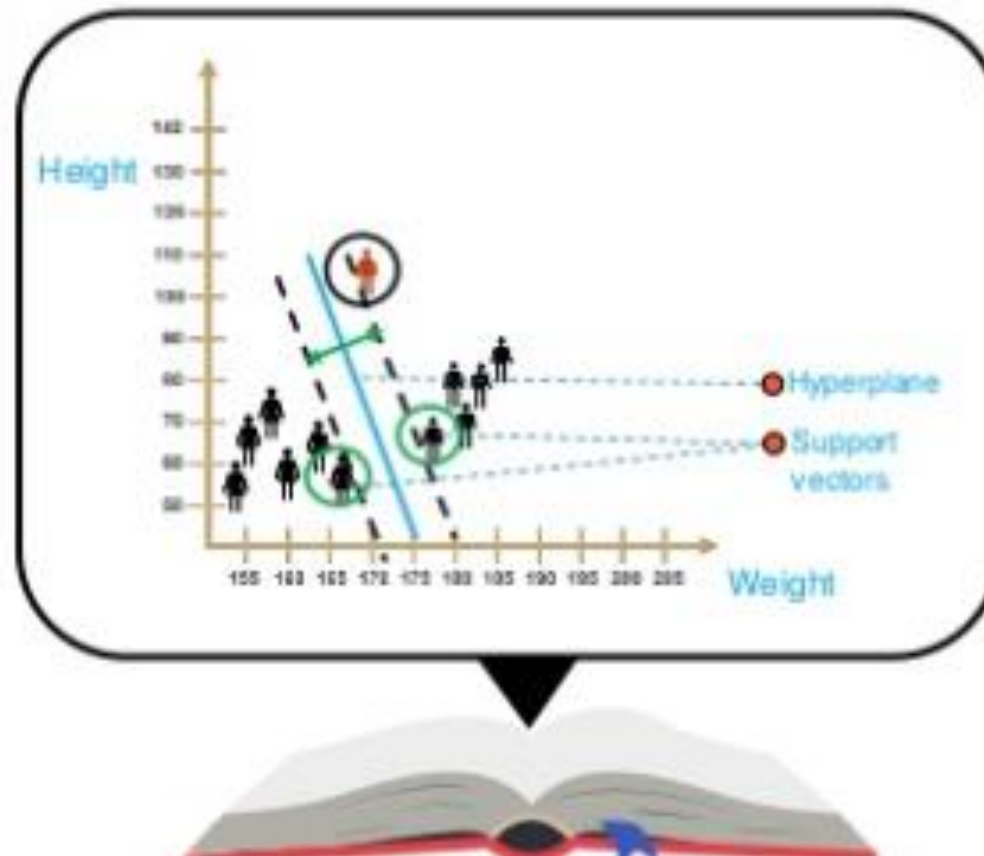
SVM – Another try

Where support vectors are the extreme points in the datasets



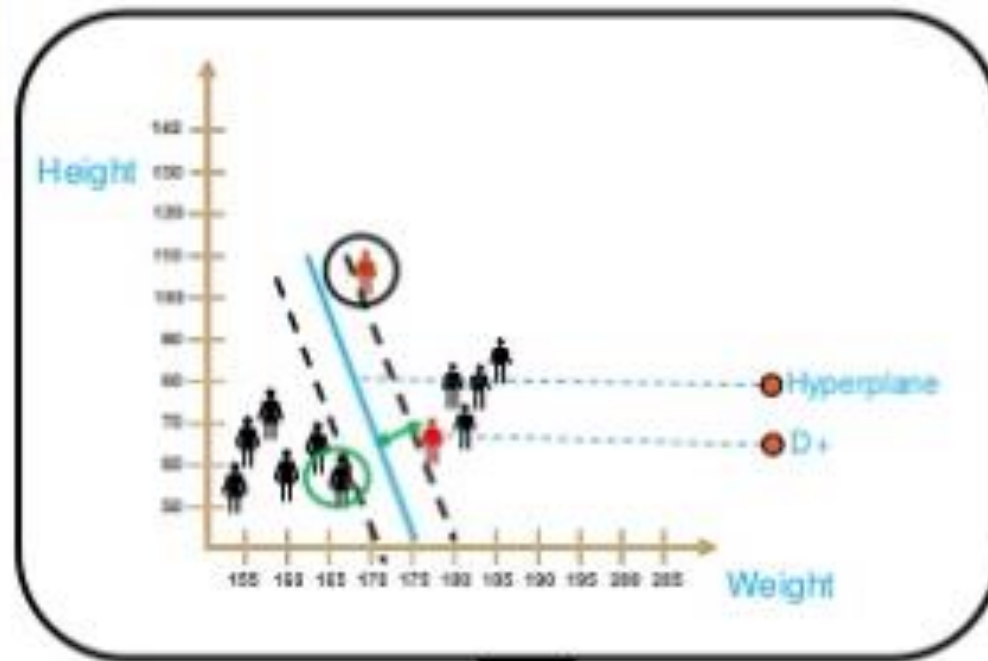
SVM – Another try

And a hyperplane has the maximum distance to the support vectors of any class



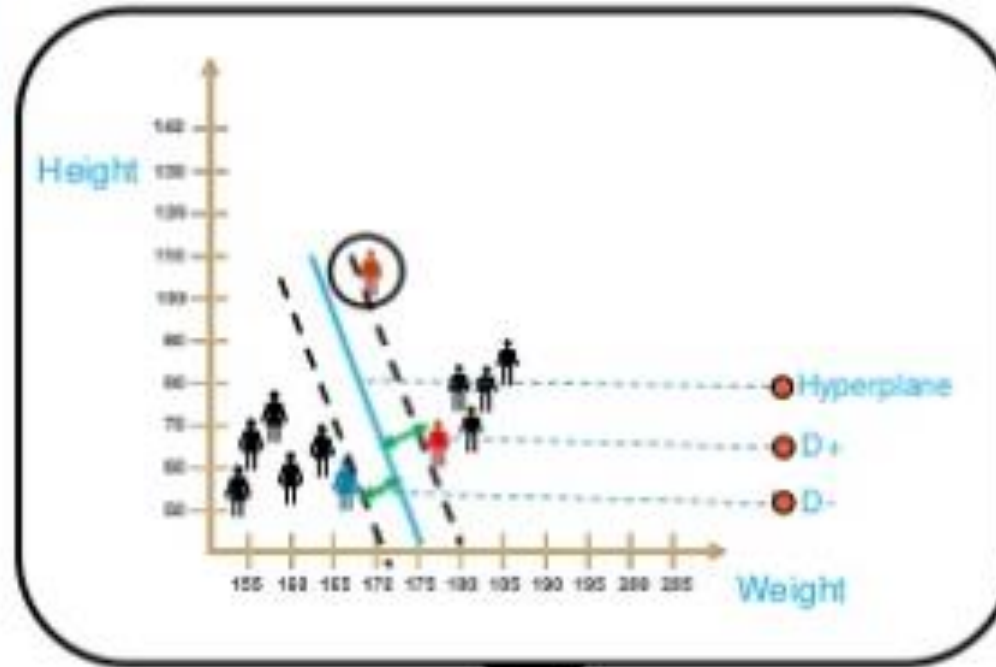
SVM – Another try

Here, D_+ is the shortest distance to the closest positive point



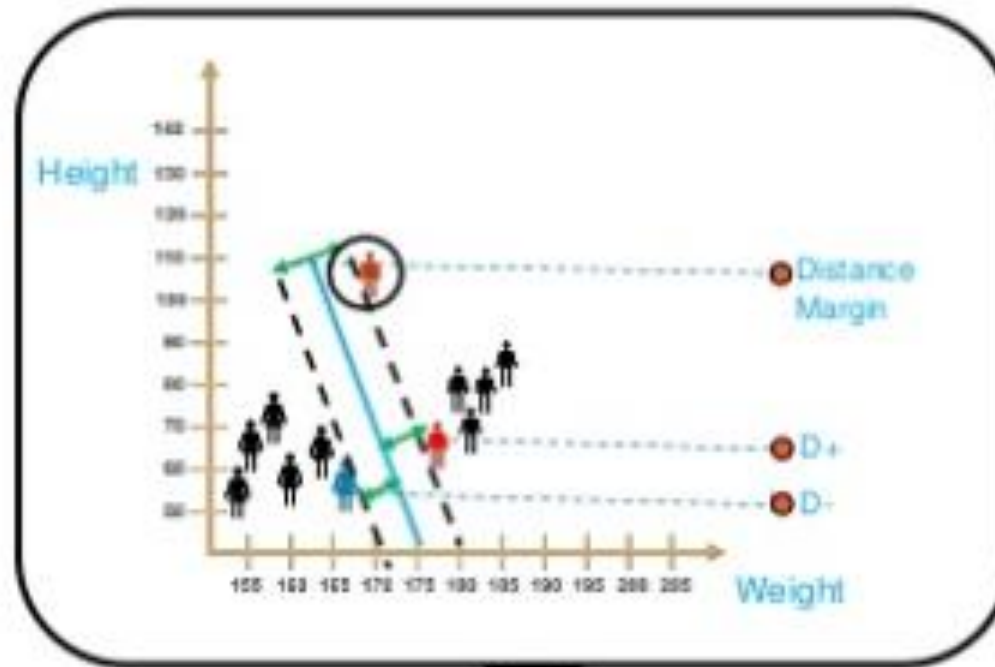
SVM – Another try

And D^- is the shortest distance to the closest negative point



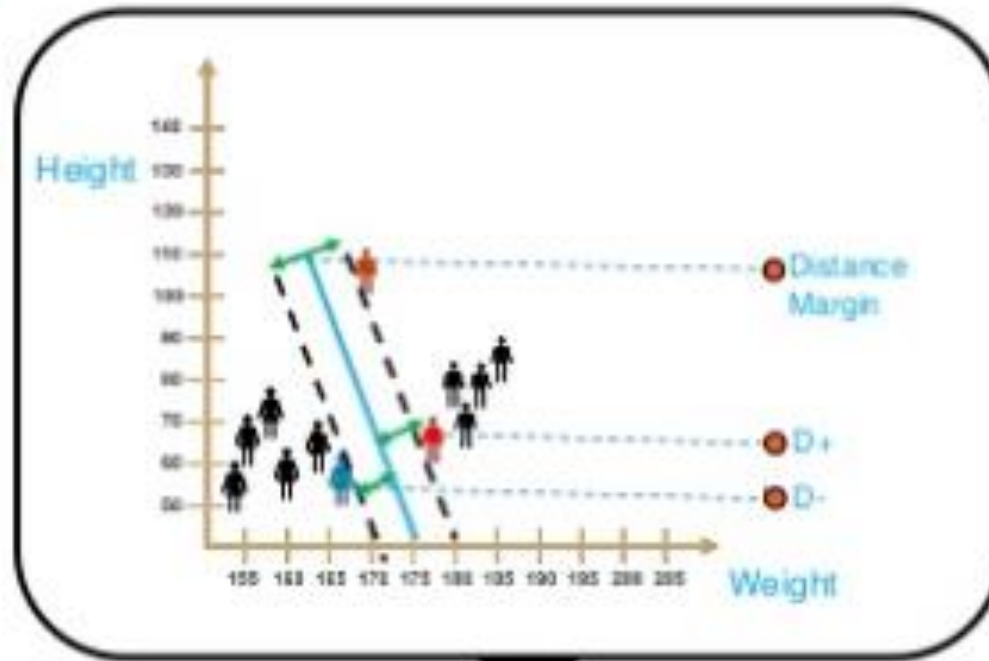
SVM – Another try

Sum of D_+ and D_- is called the distance margin



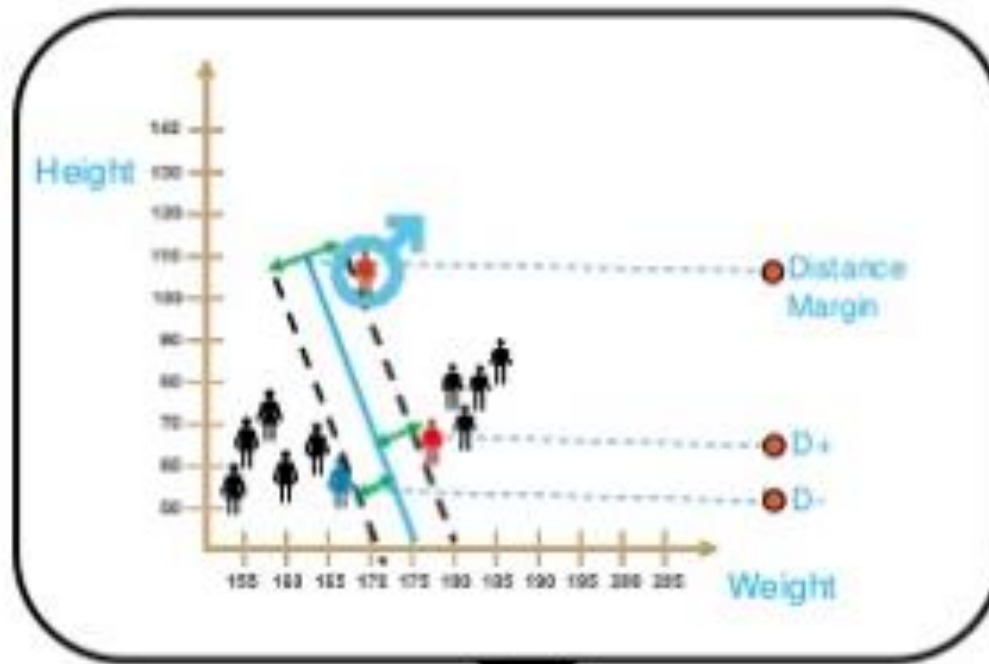
SVM – Another try

From the distance margin, we get an optimal hyperplane

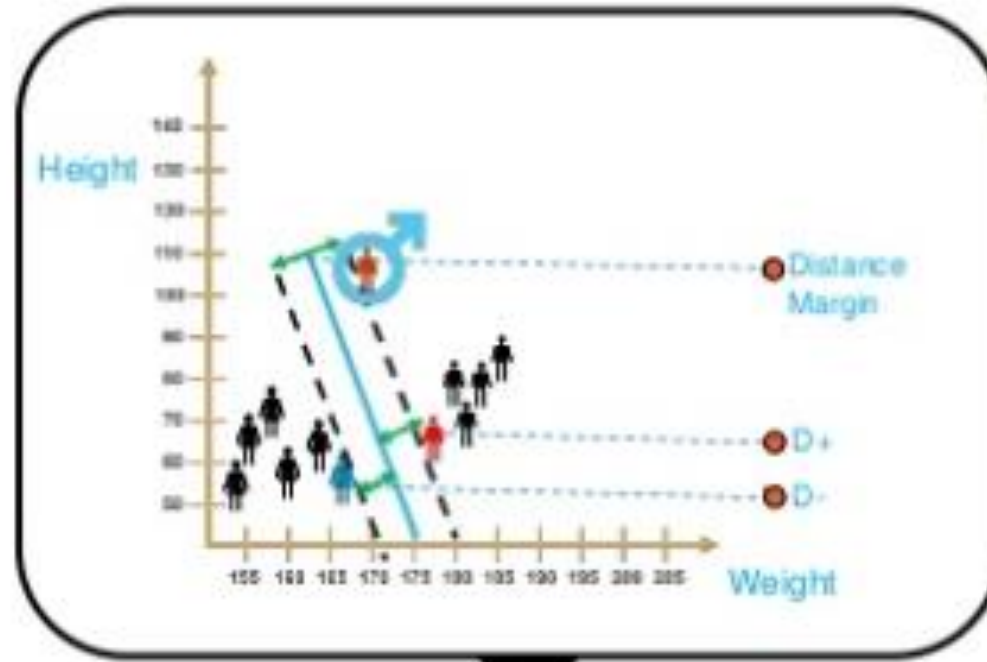


SVM – Another try

Based on the hyperplane,
we can say the new data point
belongs to male gender



SVM – Another try

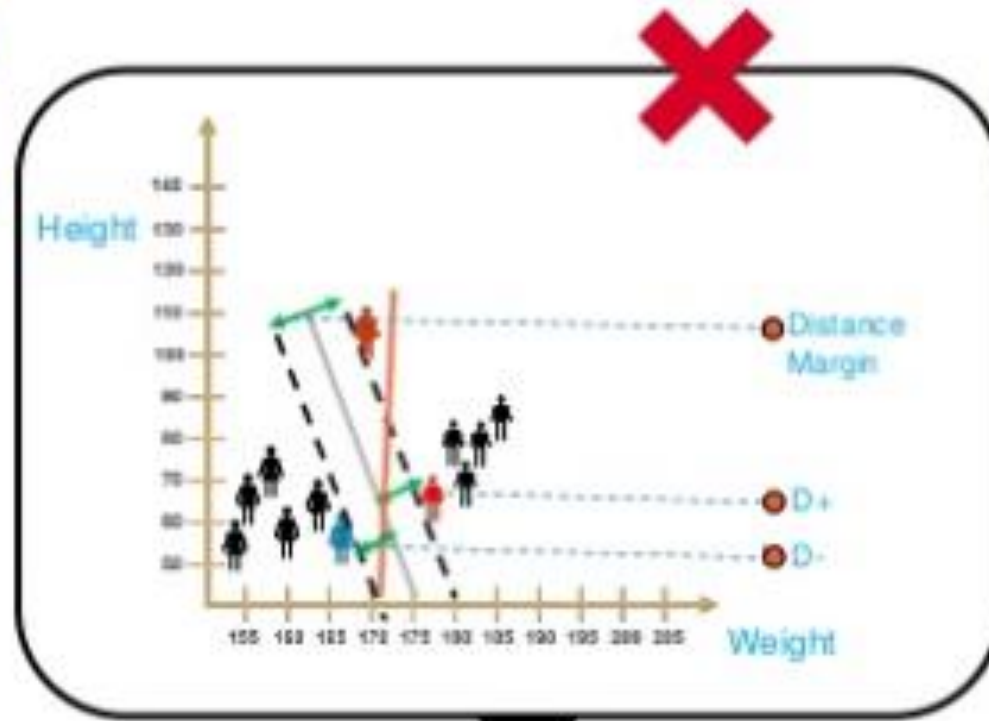


But what happens if a hyperplane is not optimal?



SVM – Another try

If we select a hyperplane having low margin then there is high chance of misclassification

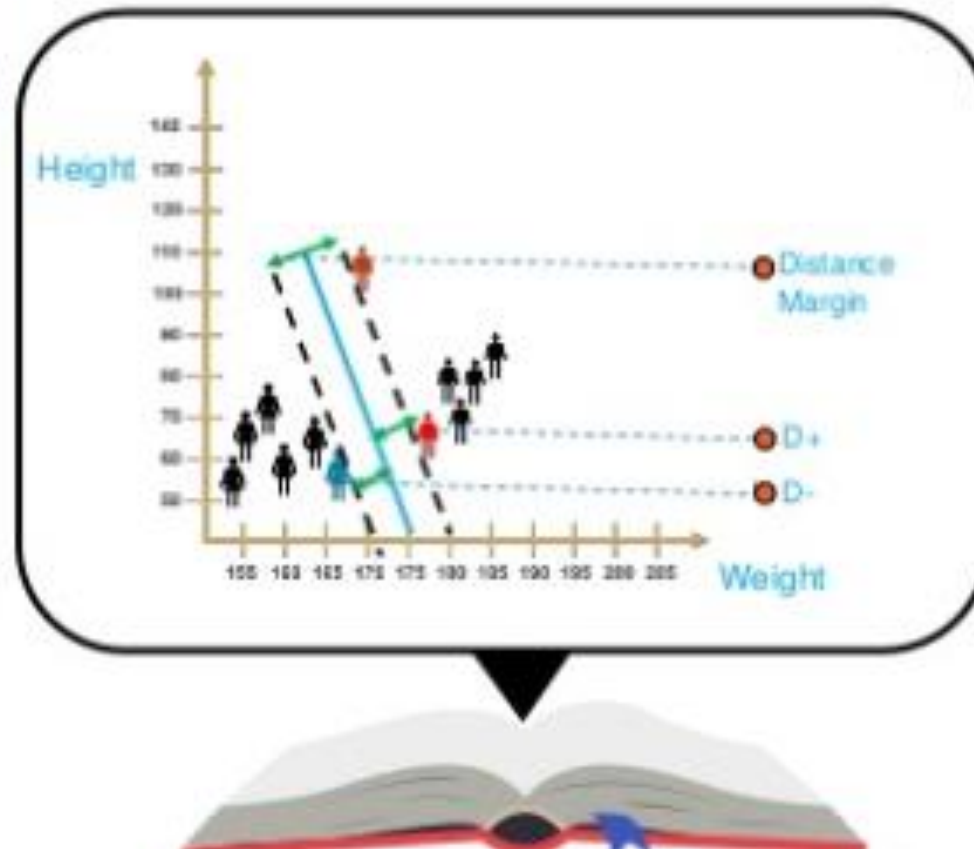


But what happens if a hyperplane is not optimal?



SVM – Another try

What we discussed so far, is also called as **LSVM**



But what happens if a hyperplane is not optimal?



SVM – Another try

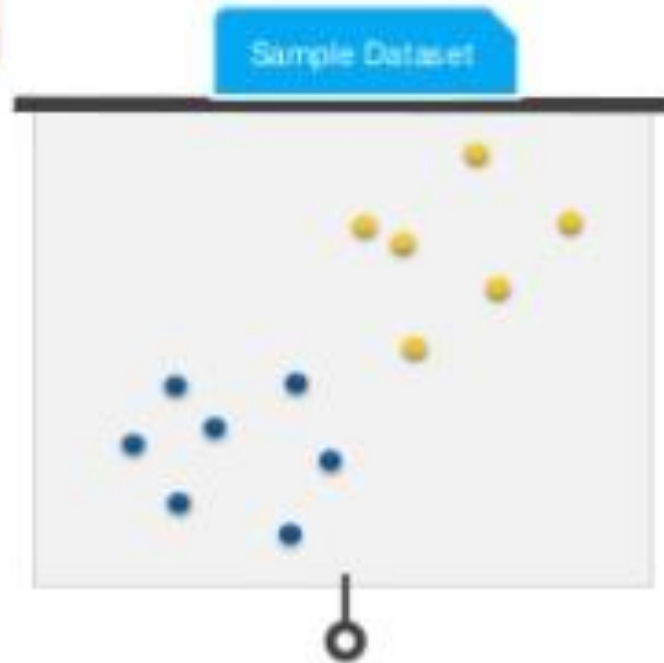
Well, so far it is clear



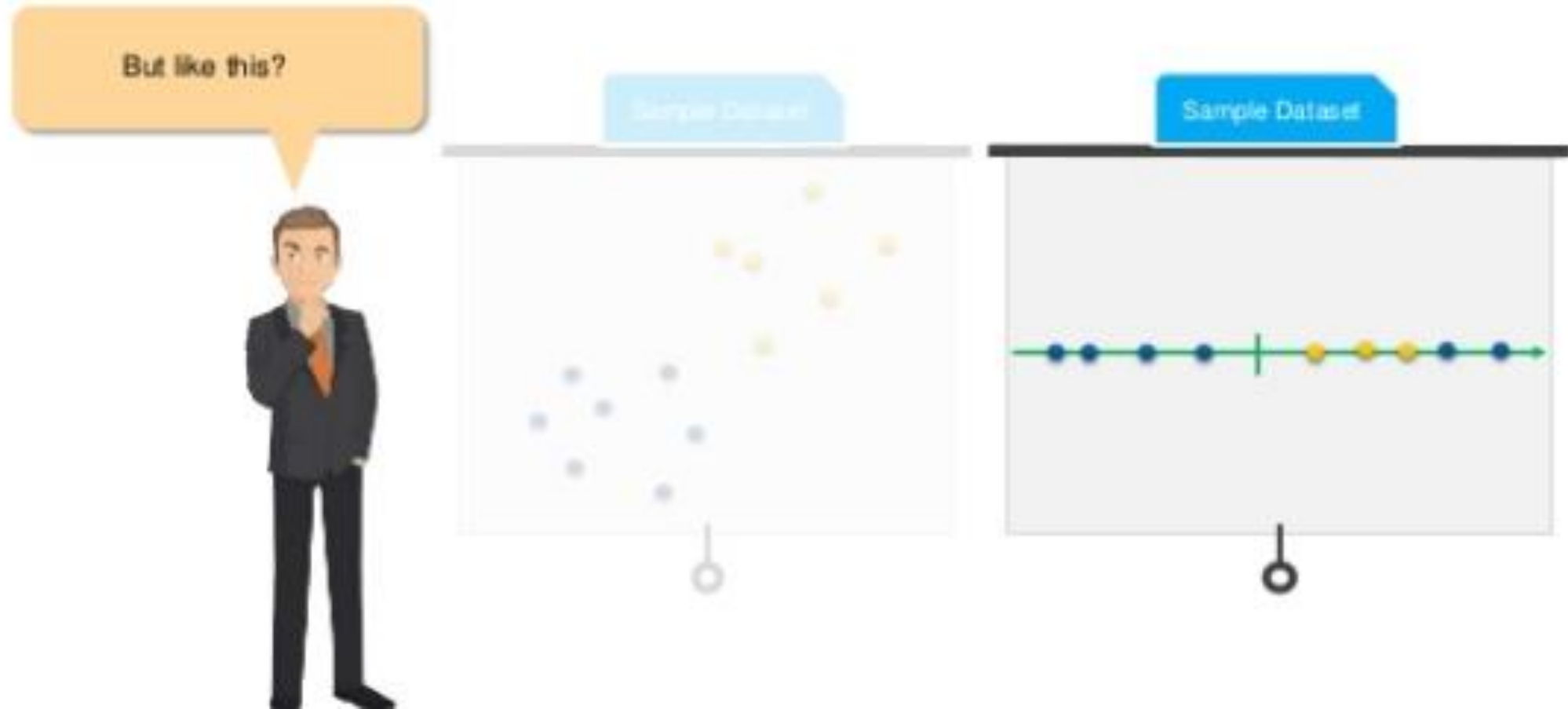
SVM – Another try

What if my data was not like this

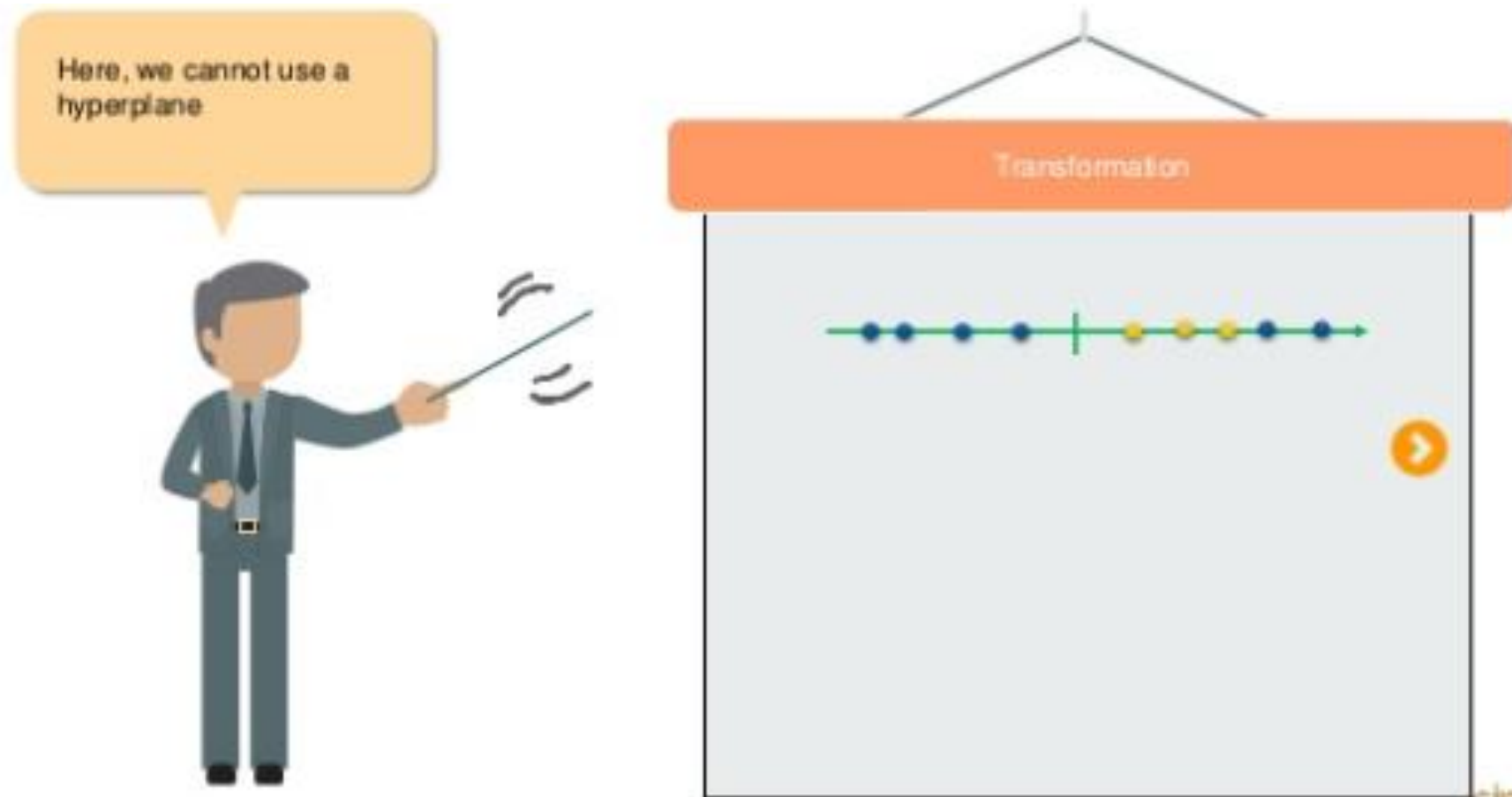
Sample Dataset



SVM – Another try

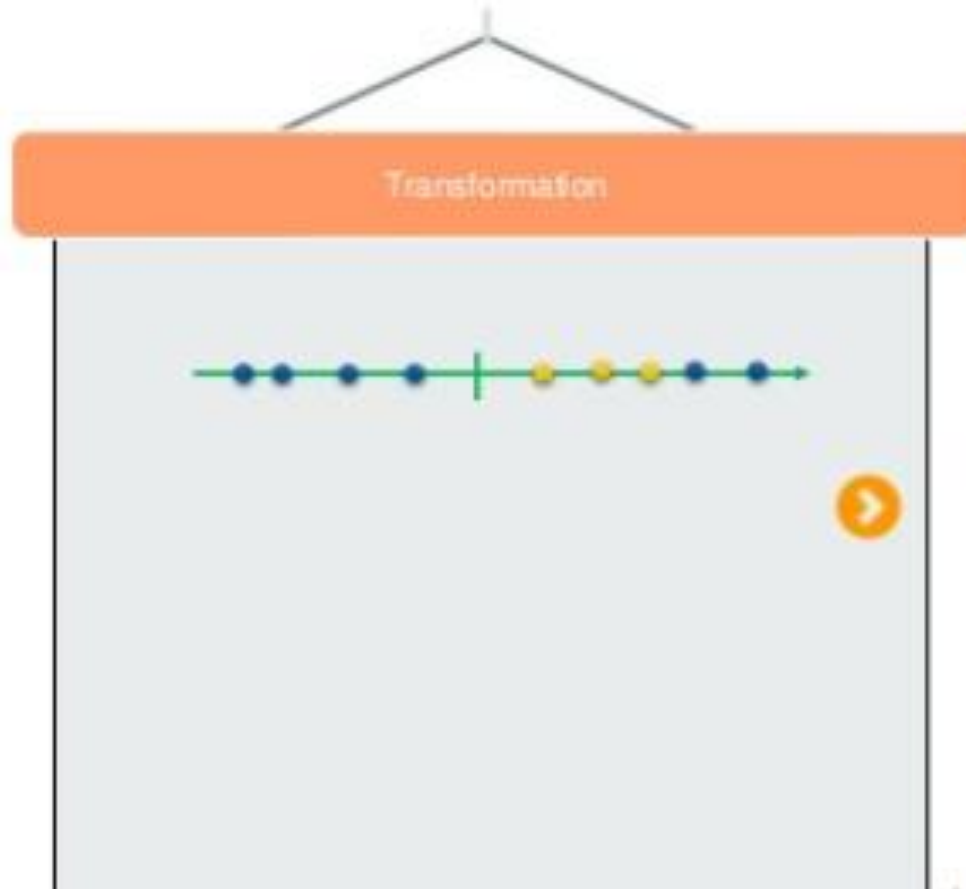


SVM – Another try

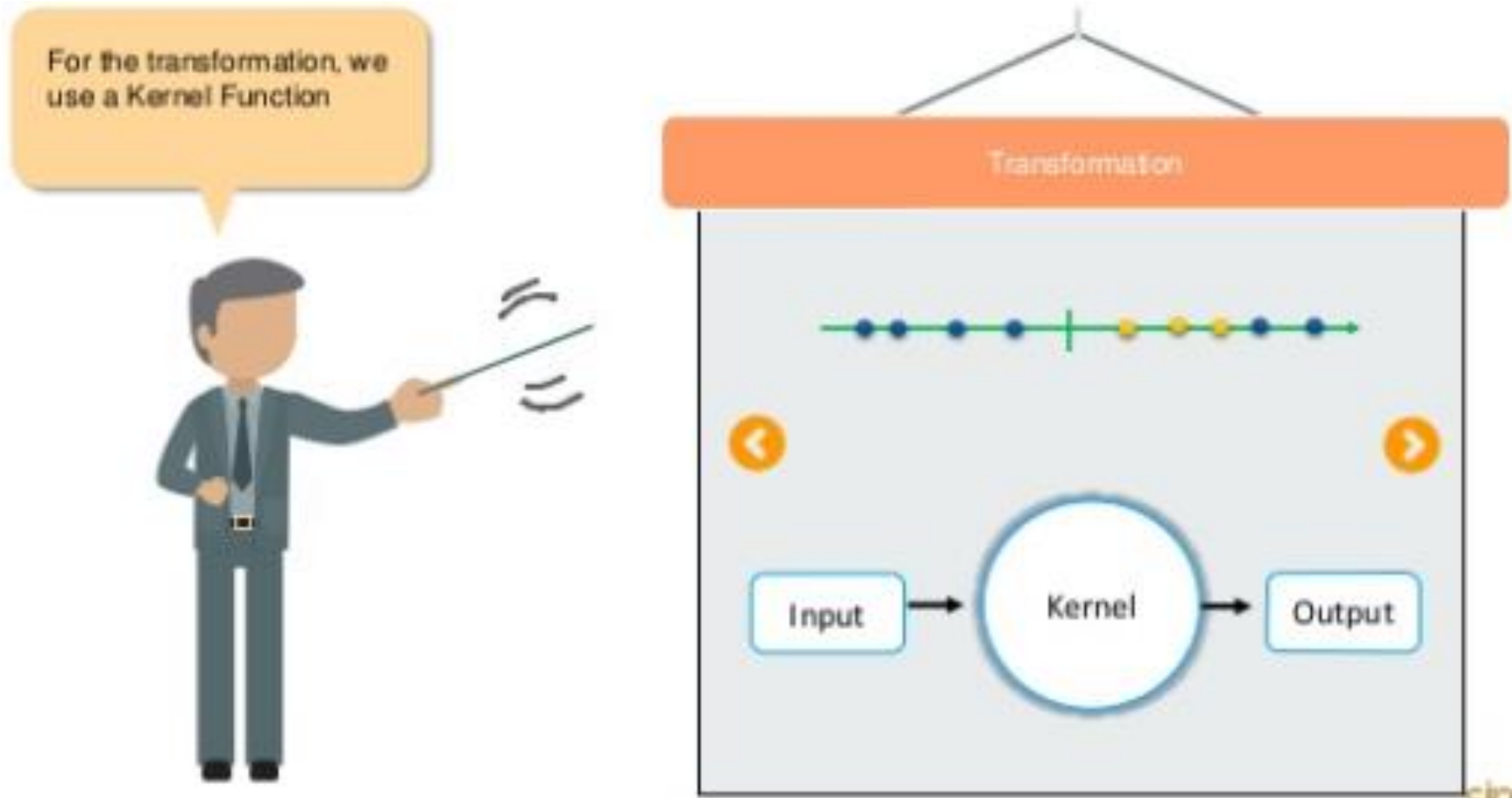


SVM – Another try

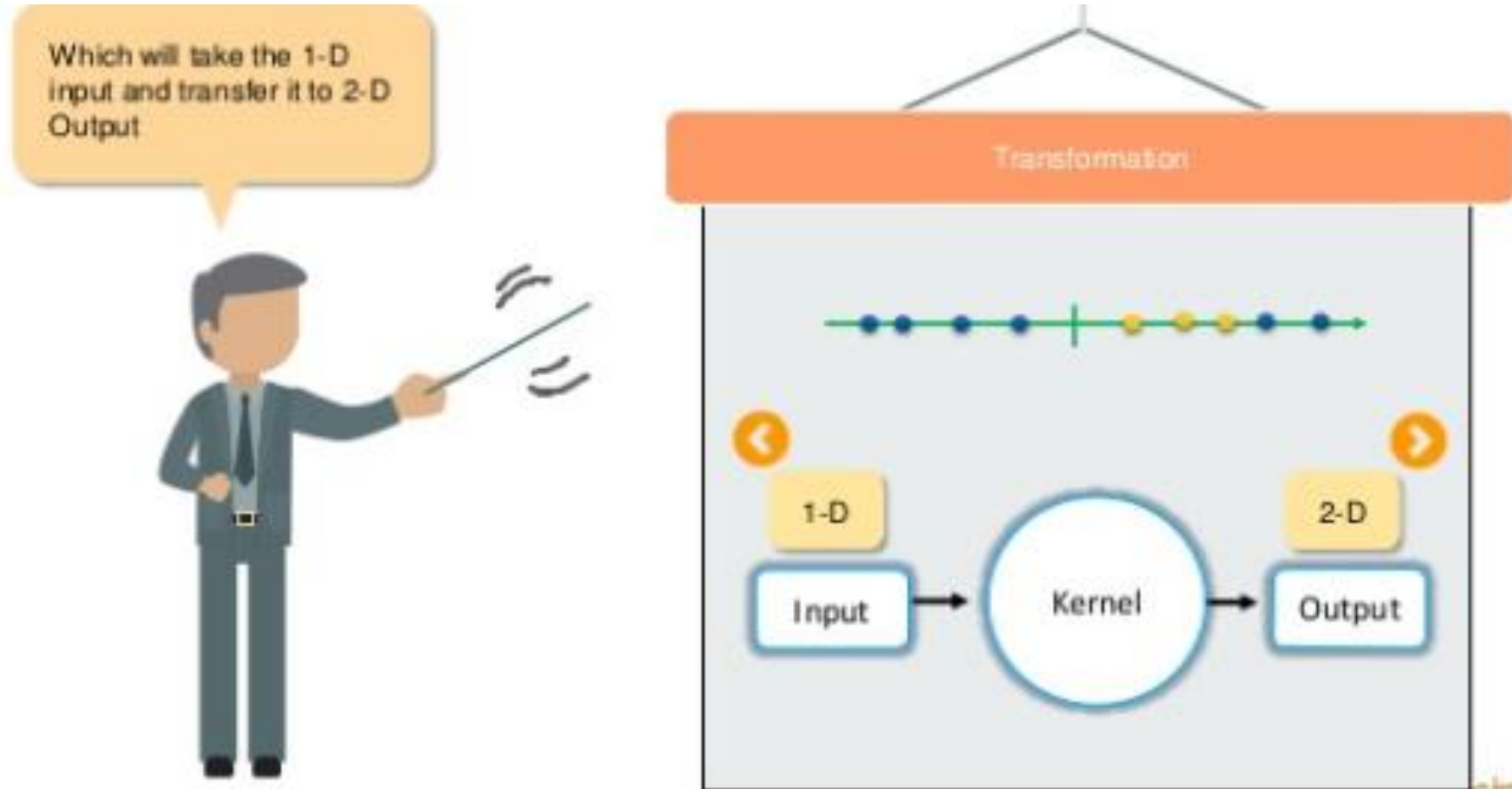
So, it's necessary to move away from a 1-D view of the data to a 2-D view



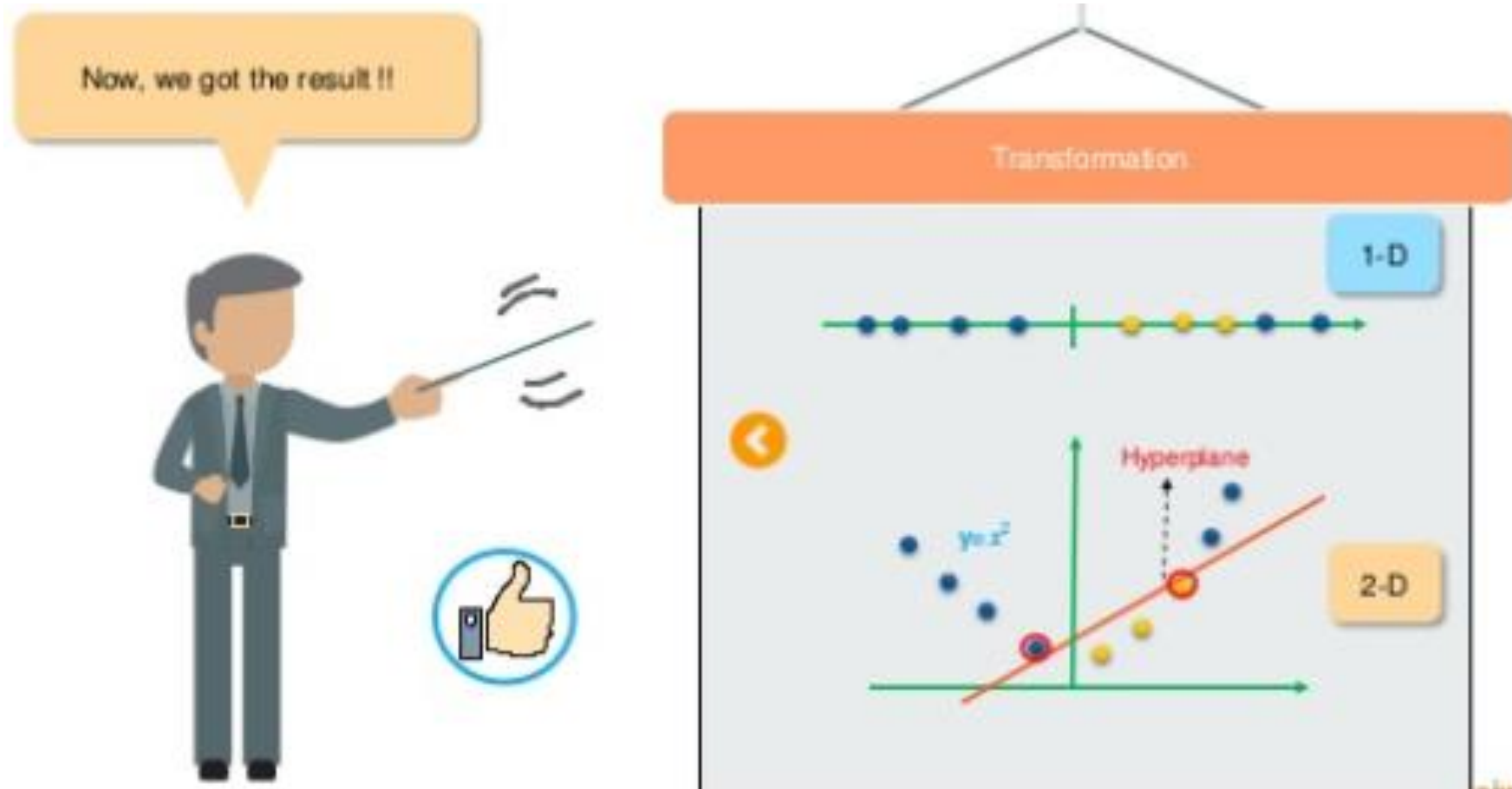
SVM – Another try



SVM – Another try



SVM – Another try

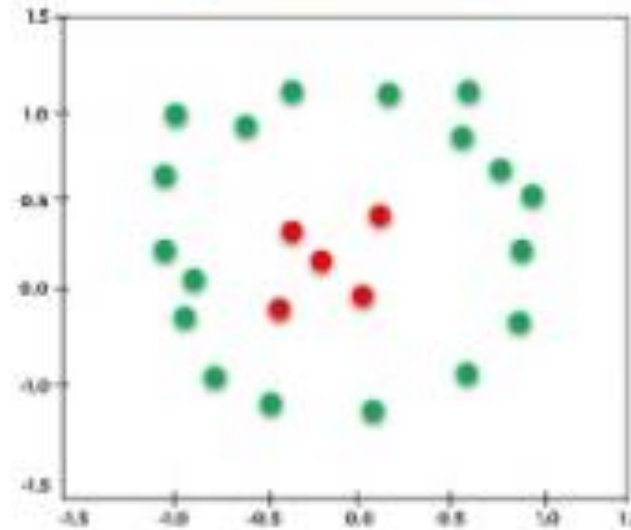


SVM – Another try

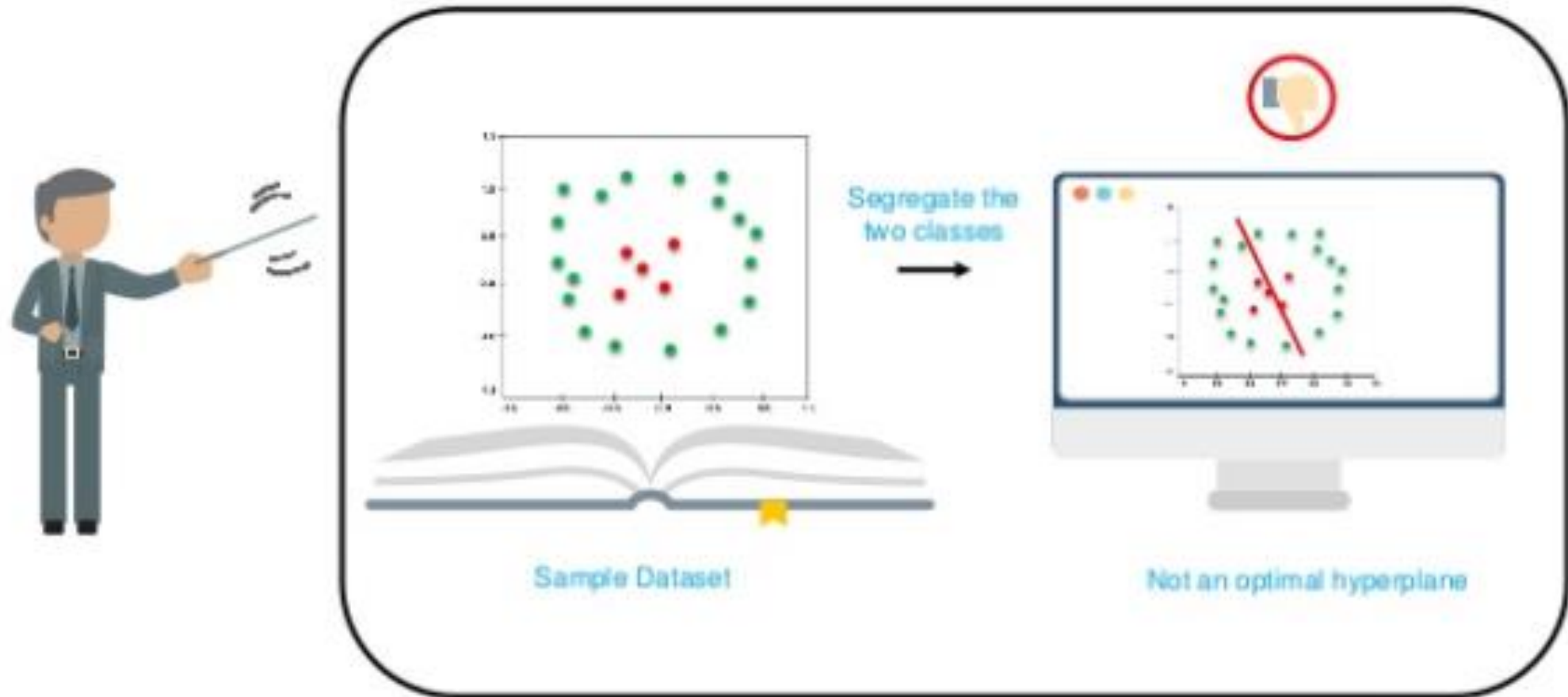
How to perform SVM
for this type of dataset?



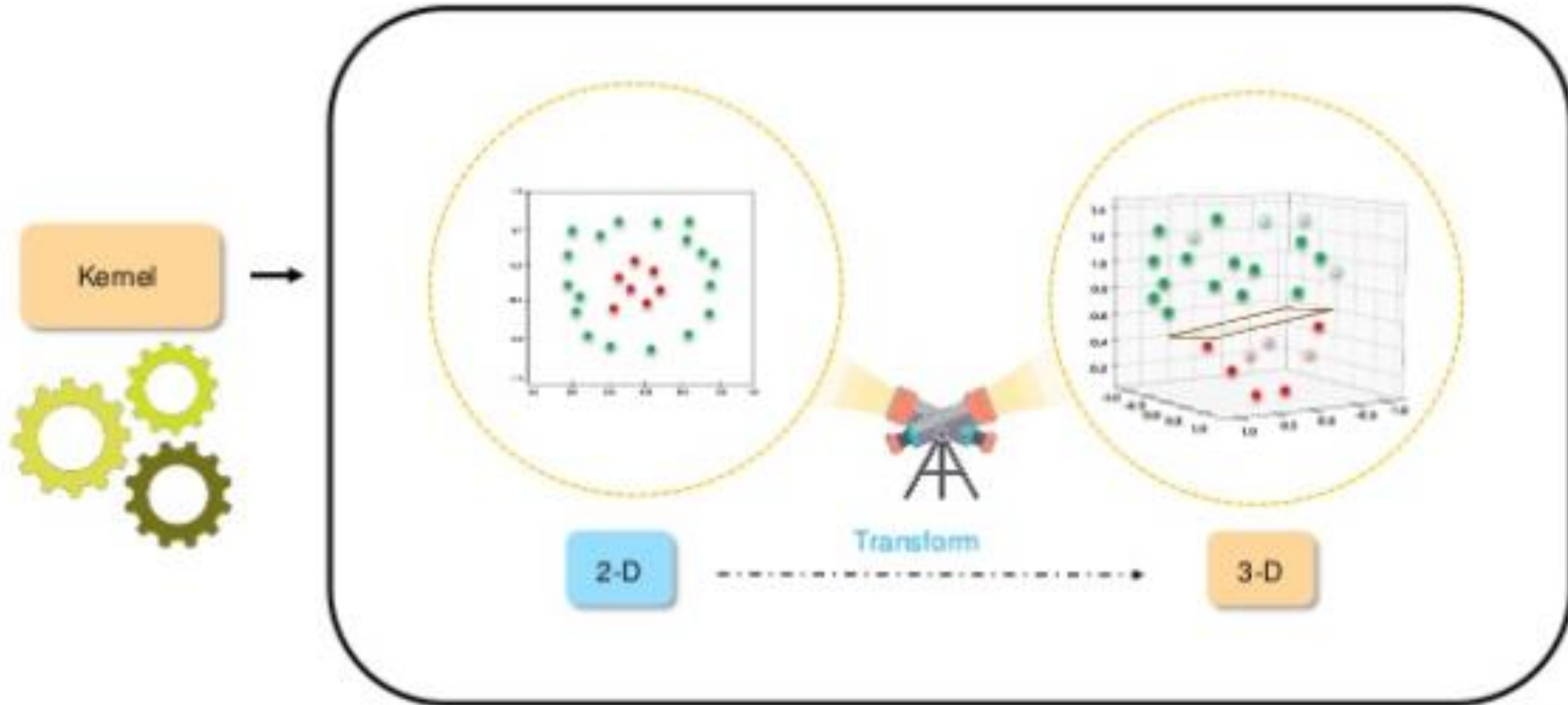
Sample Dataset



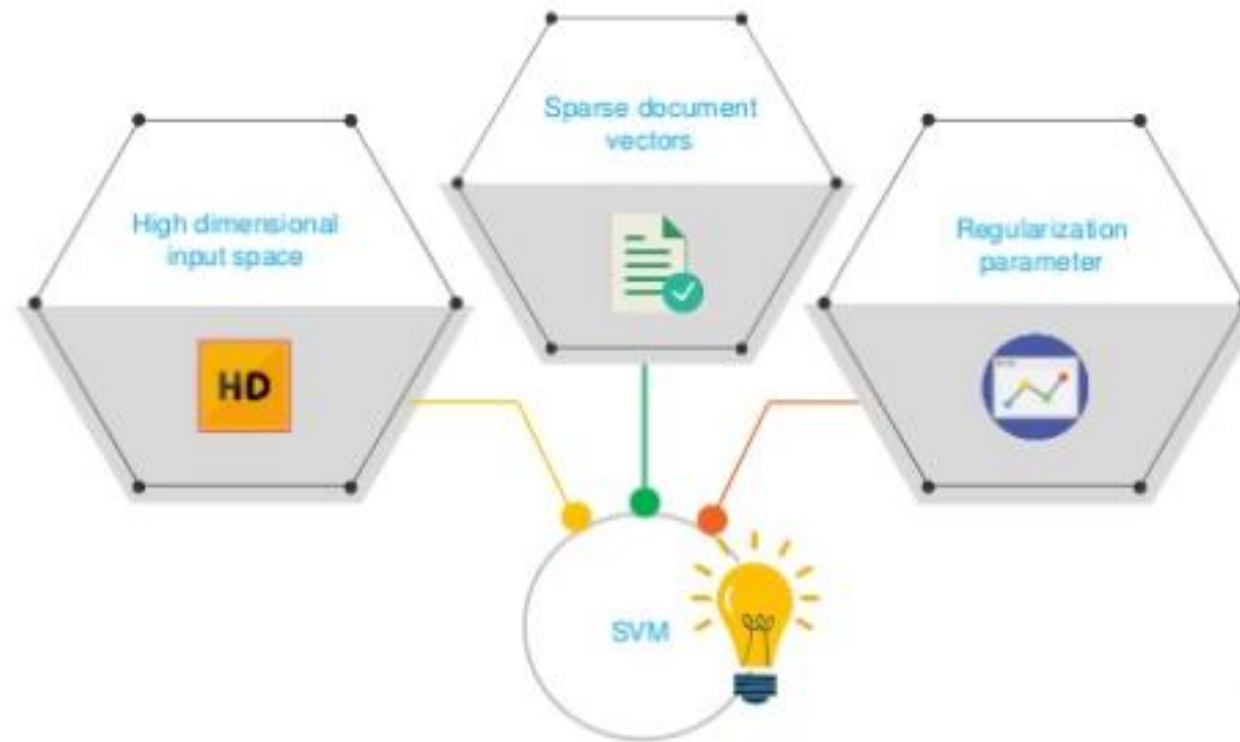
SVM – Another try



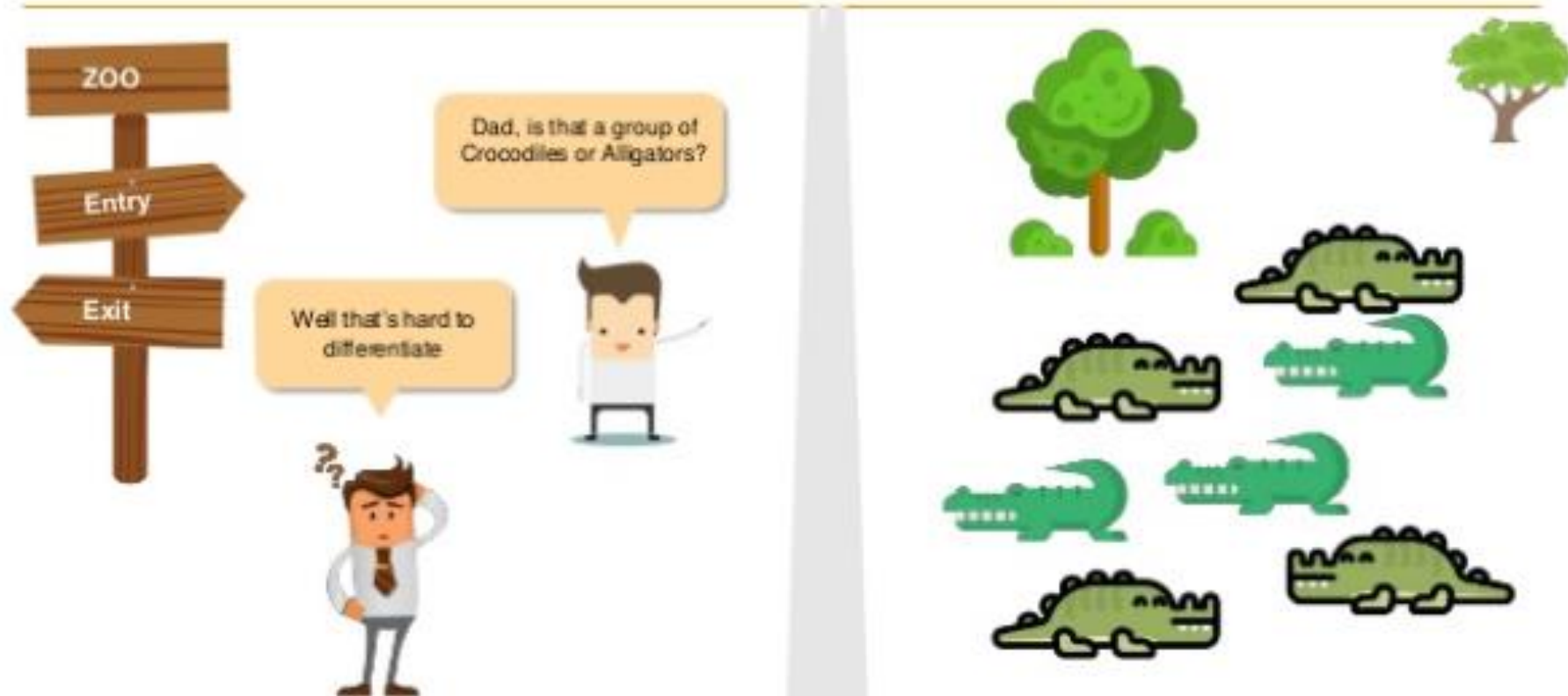
SVM – Another try



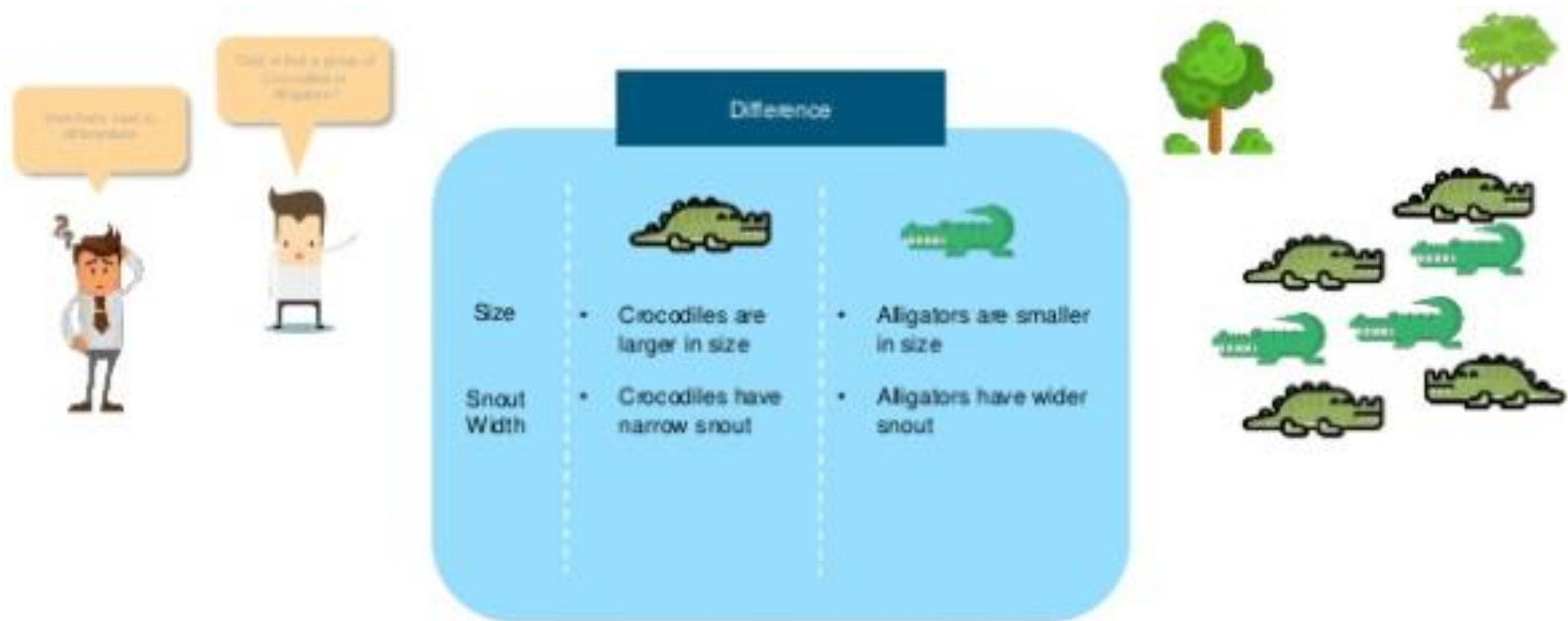
SVM – Another try



SVM – Another try



SVM – Another try



SVM – Another try



SVM – Another try

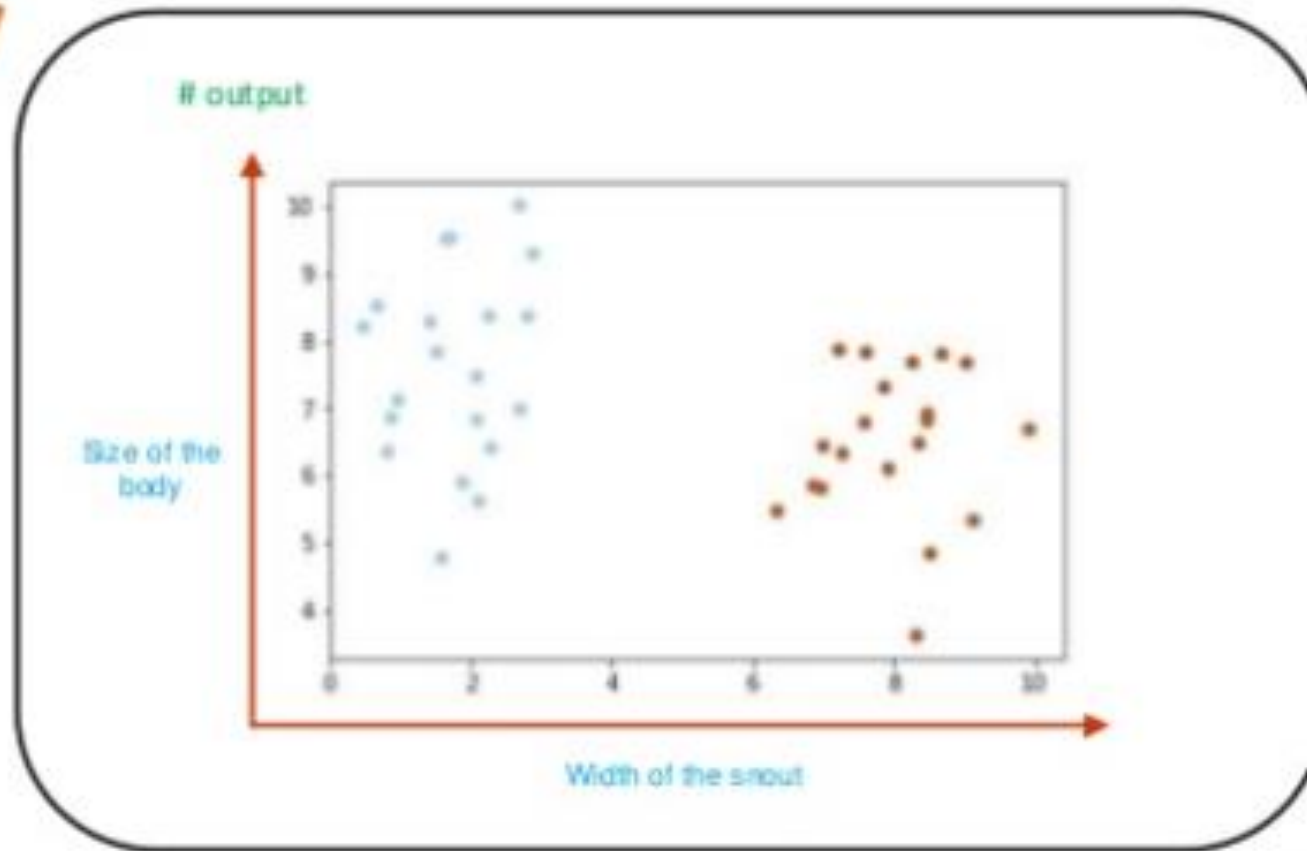


```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.datasets.samples_generator import make_blobs
# we create 40 separable points
X, y = make_blobs(n_samples=40, centers=2, random_state=20)

# fit the model, don't regularize for illustration purposes
clf = svm.SVC(kernel='linear', C=1000)
clf.fit(X, y)

plt.scatter(X[:, 0], X[:, 1], c=y, s=30, cmap=plt.cm.Paired)
```

SVM – Another try



SVM – Another try



```
# fit the model, don't regularize for illustration purposes
clf = svm.SVC(kernel='linear', C=1000)
clf.fit(X, y)

plt.scatter(X[:, 0], X[:, 1], c=y, s=30, cmap=plt.cm.Paired)
plt.show

# plot the decision function
ax = plt.gca()
xlim = ax.get_xlim()
ylim = ax.get_ylim()

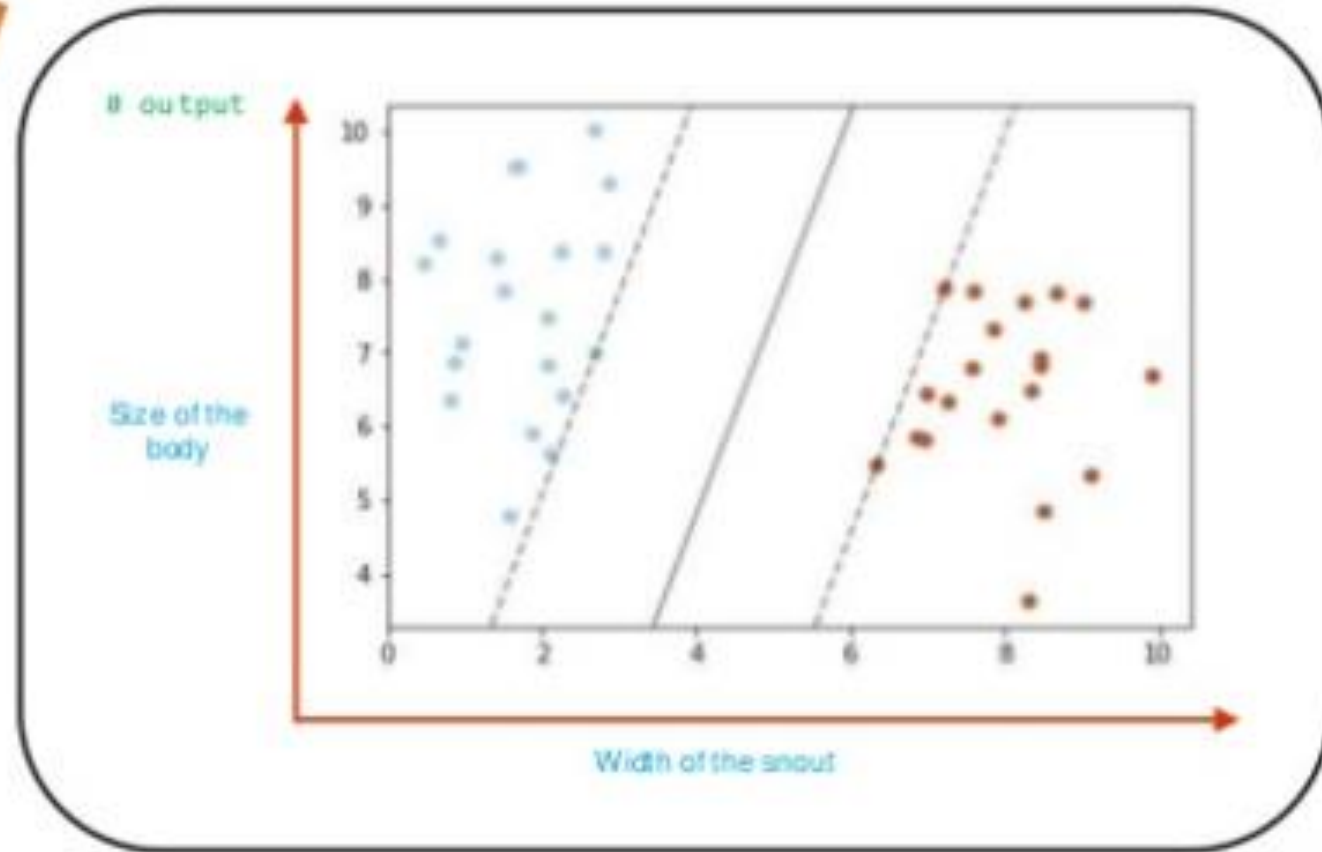
# create grid to evaluate model
xx = np.linspace(xlim[0], xlim[1], 30)
yy = np.linspace(ylim[0], ylim[1], 30)
YY, XX = np.meshgrid(yy, xx)
xy = np.vstack([XX.ravel(), YY.ravel()]).T
Z = clf.decision_function(xy).reshape(XX.shape)
```

SVM – Another try

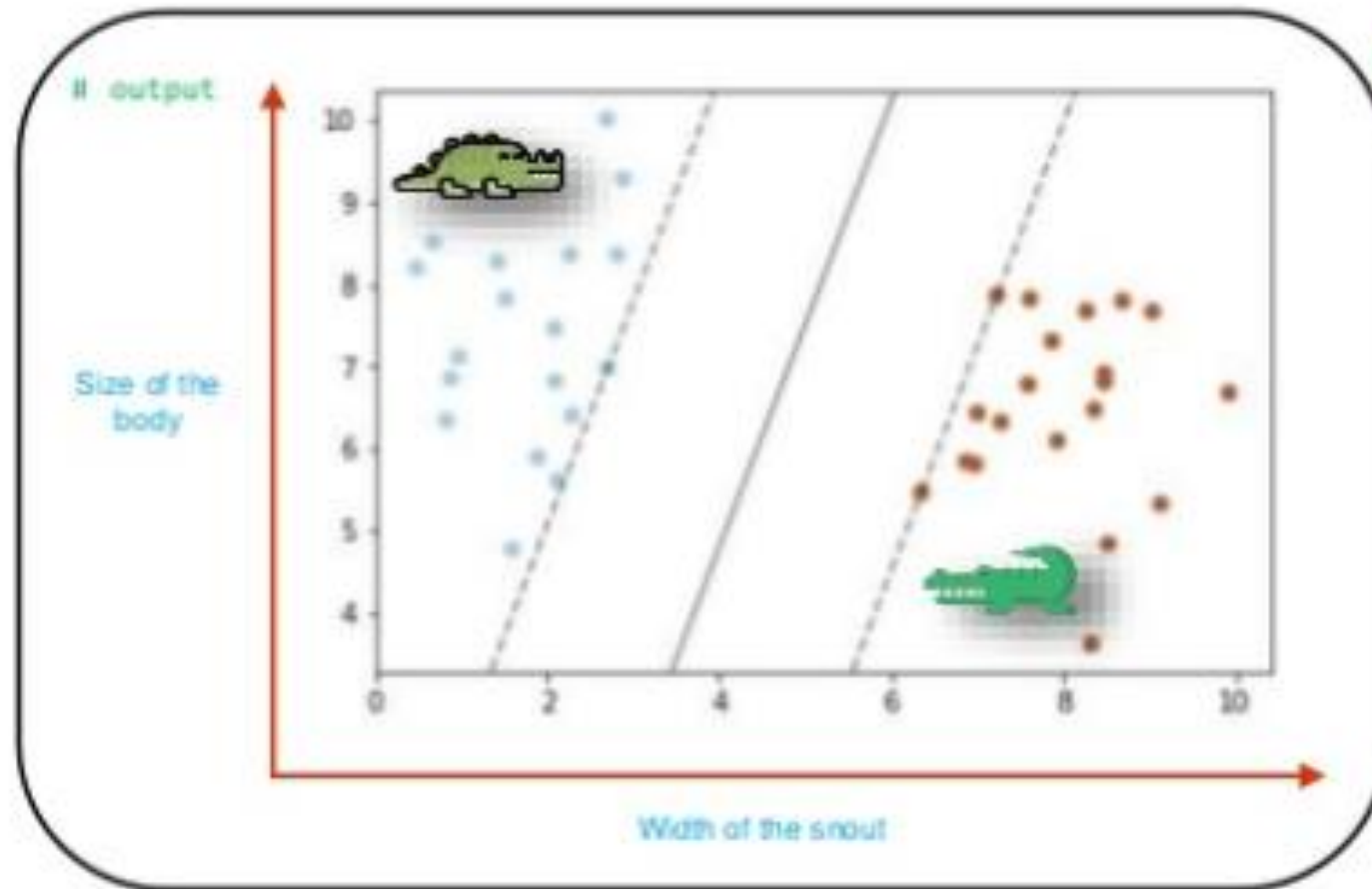


```
# plot decision boundary and margins
ax.contour(XX, YY, Z, colors='k', levels=[-1, 0, 1], alpha=0.5,
           linestyles=['--', '-', '--'])
# plot support vectors
ax.scatter(clf.support_vectors_[0], clf.support_vectors_[1],
          s=100, linewidth=1, facecolors='none')
plt.show()
```

SVM – Another try



SVM – Another try



A Cautionary Example



Image classification of tanks. Autofire when an enemy tank is spotted.

Input data: Photos of own and enemy tanks.

Worked really good with the training set used.

In reality it failed completely.

Reason: All enemy tank photos taken in the morning. All own tanks in dawn.
The classifier couldn't recognize dusk from dawn!!!!

Tips to tune SVM

- **Kernel:** The main function of the kernel is to transform the given dataset input data into the required form. There are various types of functions such as linear, polynomial, and radial basis function (RBF). Polynomial and RBF are useful for non-linear hyperplane.
- **Regularization:** Regularization parameter in python's Scikit-learn C parameter used to maintain regularization. Here C is the penalty parameter, which represents misclassification or error term. The misclassification or error term tells the SVM optimization how much error is bearable. This is how you can control the trade-off between decision boundary and misclassification term. A smaller value of C creates a small-margin hyperplane and a larger value of C creates a larger-margin hyperplane.
- **Gamma:** A lower value of Gamma will loosely fit the training dataset, whereas a higher value of gamma will exactly fit the training dataset, which causes over-fitting. In other words, you can say a low value of gamma considers only nearby points in calculating the separation line, while the a value of gamma considers all the data points in the calculation of the separation line.

SVM Results

- F1 score should be as close to 1 as possible
- Precision and Recall
- Accuracy via confusion matrix

Hands on

- <https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python>
- <https://www.kaggle.com/halien/simple-image-classifer-with-svm>
- <https://github.com/whimian/SVM-Image-Classification>
- <https://kapernikov.com/tutorial-image-classification-with-scikit-learn/>

Questions?

Thank you!