LECTURE 8

WHAT WE HAVE LEARNT?

- Regression
- Classification

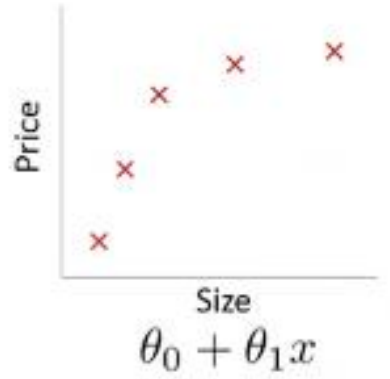
AGENDA FOR TODAY

- Problem of Overfitting & Regularization
- Deciding what to try next
- Model selection
- Bias vs Variance
- Learning Curves
- Performance Metrics
- Confusion Matrix
- Stochastic Gradient Descent and Mini- Batch Gradient Descent
- Unsupervised Learning

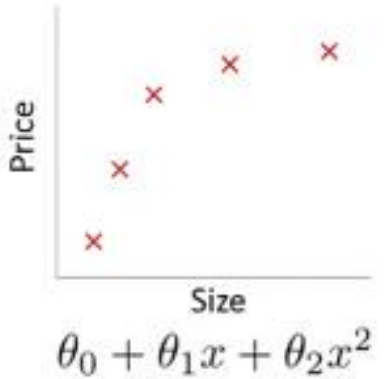
PROBLEM OF OVERFITTING

REGRESSION EXAMPLE

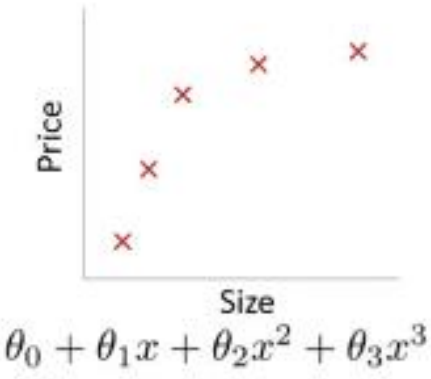
Example: Linear regression (housing prices)



$\theta_0 + \theta_1 x$

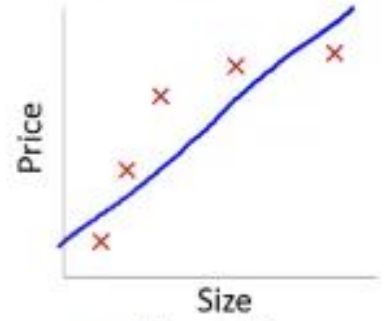


$\theta_0 + \theta_1 x + \theta_2 x^2$



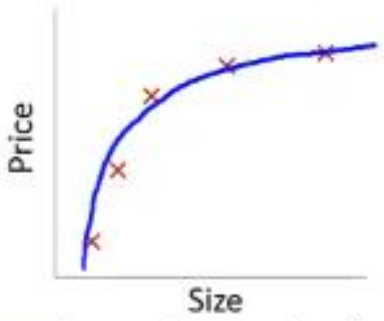
$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

Example: Linear regression (housing prices)



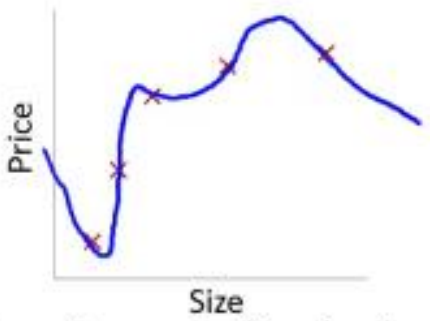
$\rightarrow \theta_0 + \theta_1 x$

"Underfit" "High bias"



$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2$

"Just right"



$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

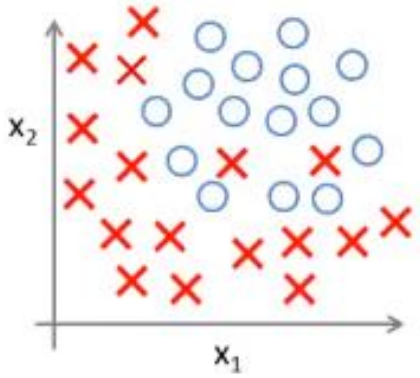
"Overfit" "High variance"

OVERFITTING

- If we have too many features, the learned hypothesis may fit the training set very well, but fail to generalise to new examples

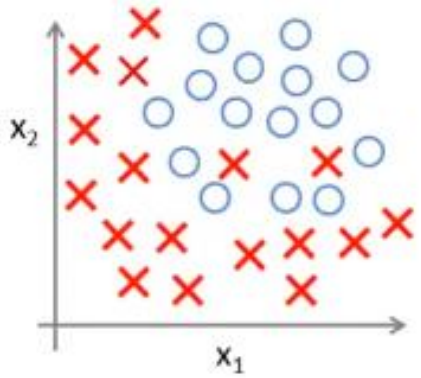
CLASSIFICATION EXAMPLE

Example: Logistic regression

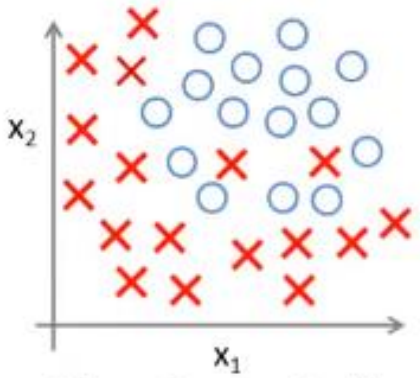


$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

(g = sigmoid function)

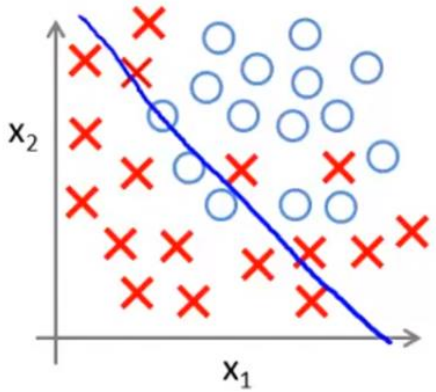


$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$



$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \ldots)$

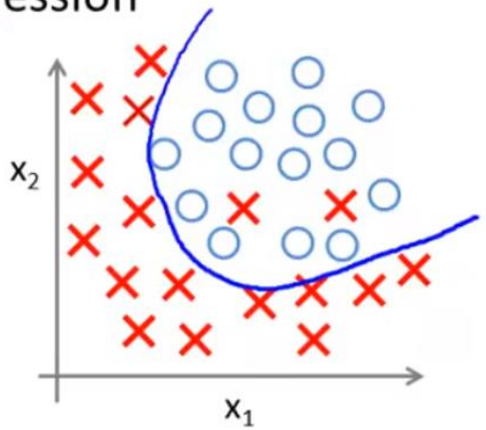
Example: Logistic regression



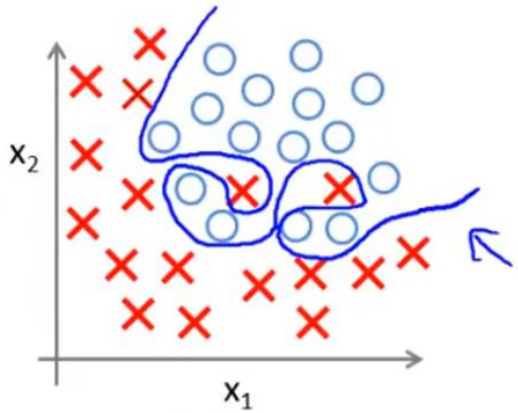
$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

(g = sigmoid function)

"Underfit"



$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$



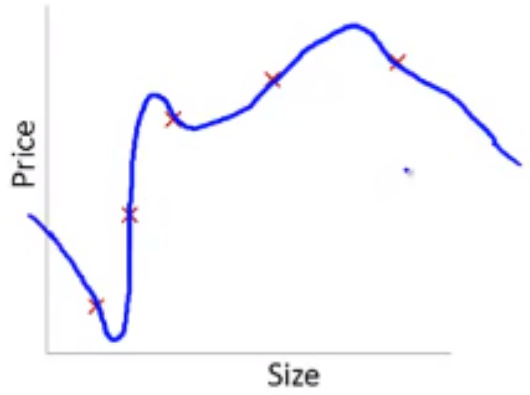
$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots)$

"Overfit"

Usually occurs due to high number of features

Addressing overfitting:

$x_1 =$ size of house
$x_2 =$ no. of bedrooms
$x_3 =$ no. of floors
$x_4 =$ age of house
$x_5 =$ average income in neighborhood
$x_6 =$ kitchen size
$\vdots$
x_{100}



ADDRESSING OVERFITTING

- Option1: Reduce number of features

✓ Usually not advisable. Some complex techniques can be used for feature engineering

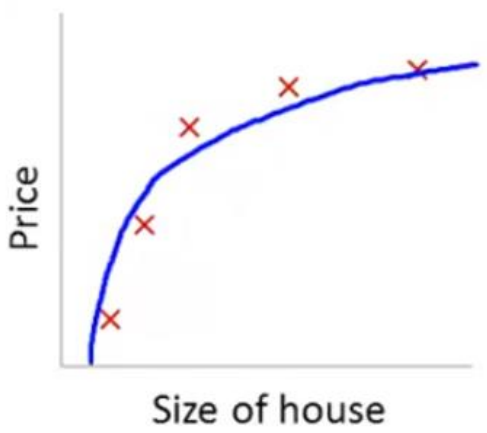
- Option2: Regularization

✓ Keep all the features, but reduce magnitude/values of parameters

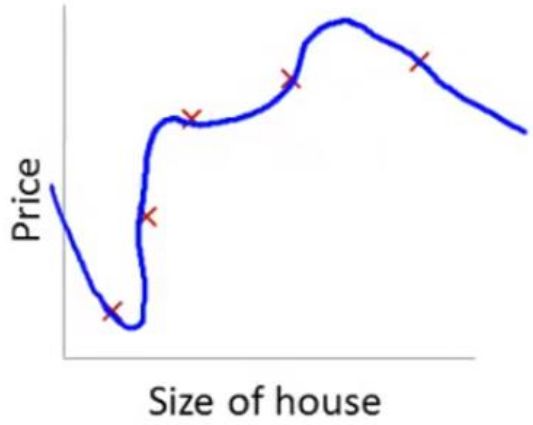
✓ Works well when we have a lot of features, each of which contributes a bit to predicting output

REGULARIZATION AND COST FUNCTION

Intuition



$\theta_0 + \theta_1 x + \theta_2 x^2$



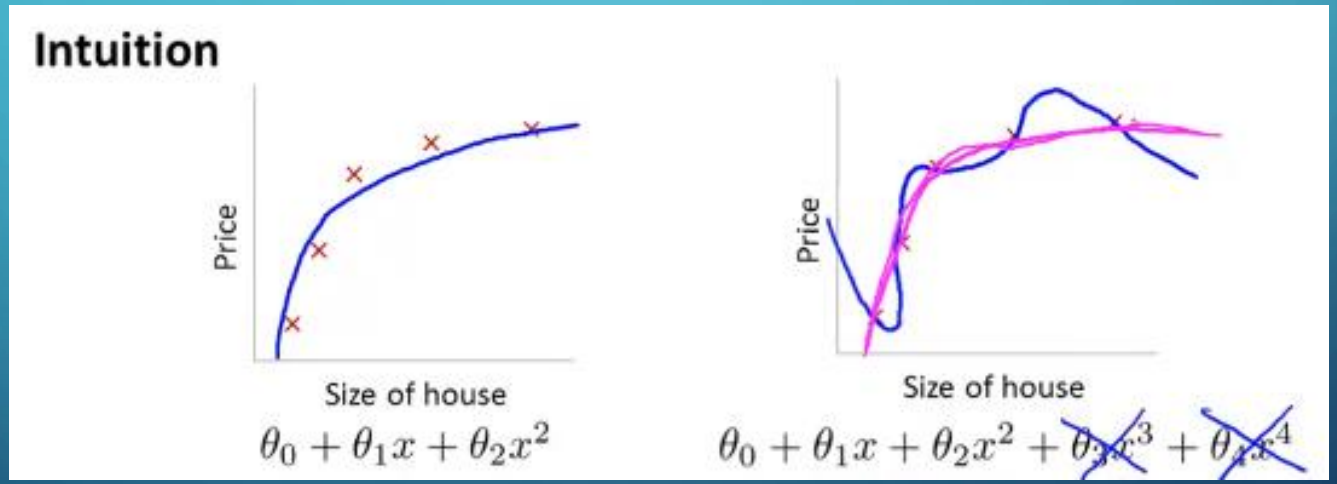
$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

Suppose we penalize and make θ_3, θ_4 really small.

$$\min_\theta \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + 1000\,\theta_3^2 + 1000\,\theta_4^2$$

- Then $\theta 3\ and\ \theta 4$ will have to be reduced to almost zero in order to minimise this function



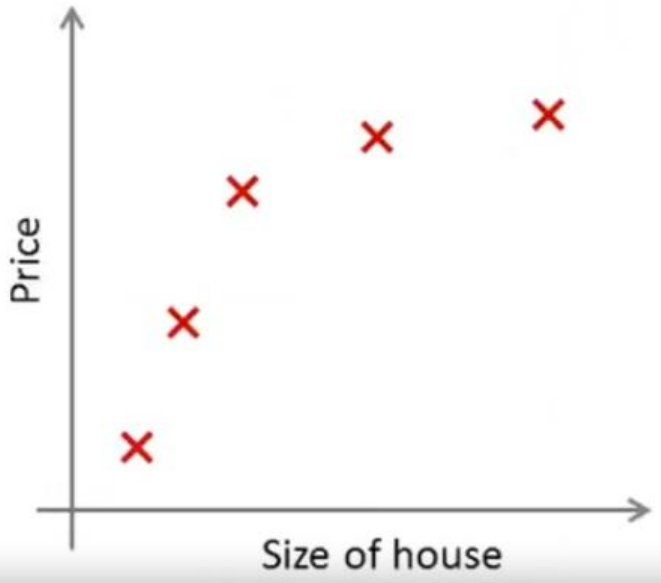
REGULARIZATION

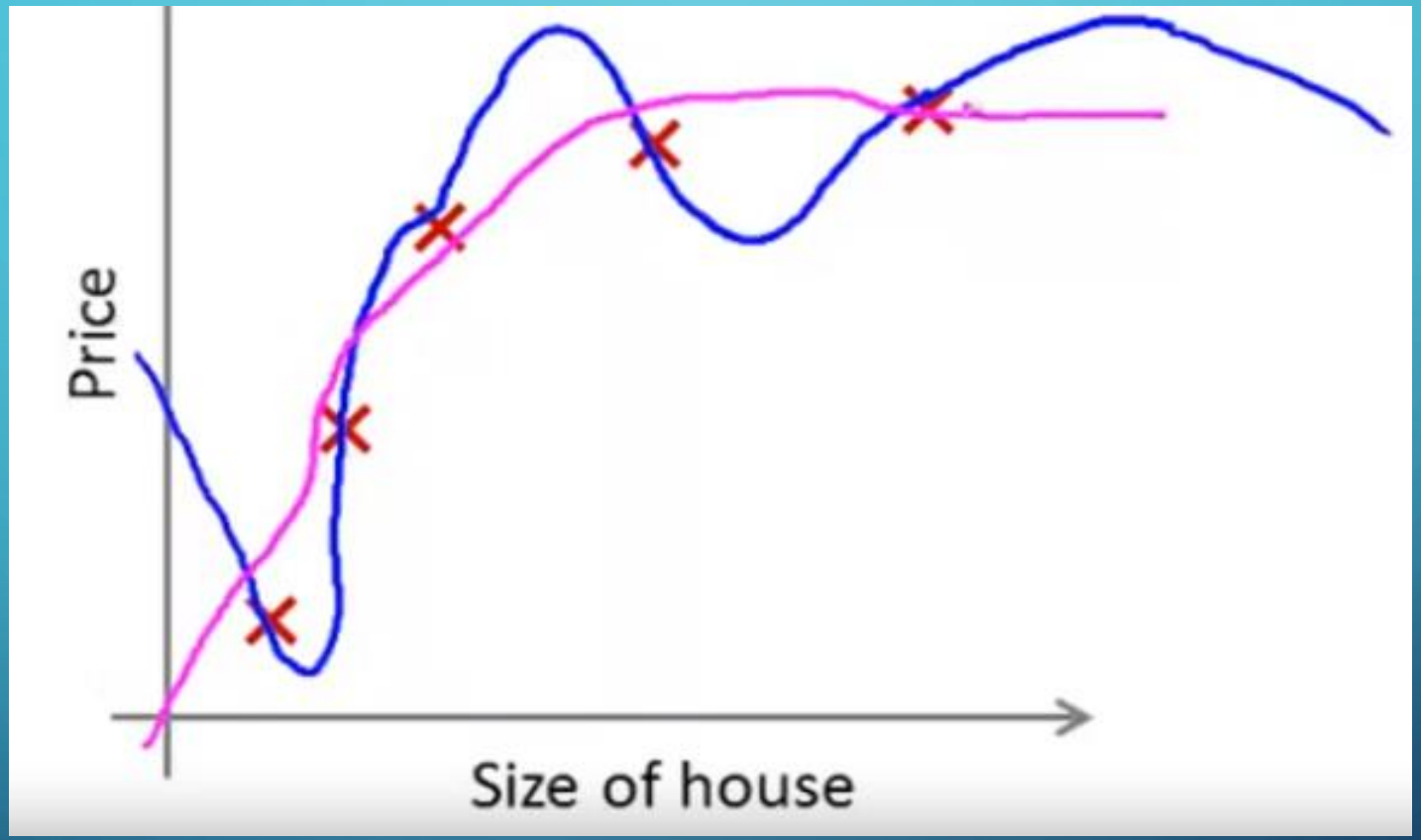
- “Simpler” hypothesis
- Less prone to overfitting

Regularization.

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

$$\min_\theta J(\theta)$$




Price
Size of house

What if λ is set to a very large value? (Say $\lambda = 10^{10}$)

$h\theta(x) = \theta_0$ *and other parameteres are reduced to zero*

REGULARIZATION IN LINEAR REGRESSION

COST FUNCTION

Regularized linear regression

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

$$\min_\theta J(\theta)$$

GRADIENT DESCENT

EARLIER:

Gradient descent

Repeat {

$$\theta_j := \theta_j - \alpha \quad \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$(j = 0, 1, 2, 3, \ldots, n)$$

}

NOW: STEP 1

Gradient descent

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$(j = 0, 1, 2, 3, \ldots, n)$$

}

NOW: STEP2

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} \theta_j \right]$$

$(j = 1, 2, 3, \ldots, n)$

}

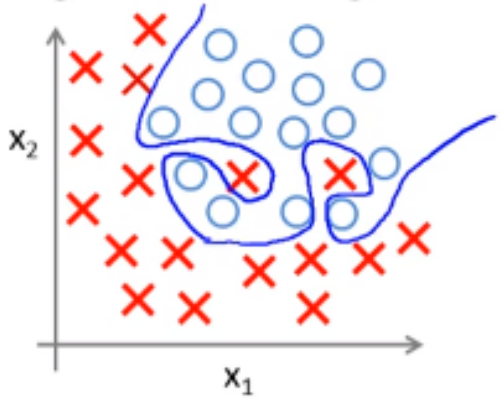
$\frac{\partial}{\partial \theta_j} J(\theta)$ regularized

NOW: STEP3

$$\theta_j := \theta_j(1 - \alpha\frac{\lambda}{m}) - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

REGULARIZATION IN LOGISTIC REGRESSION

Regularized logistic regression.

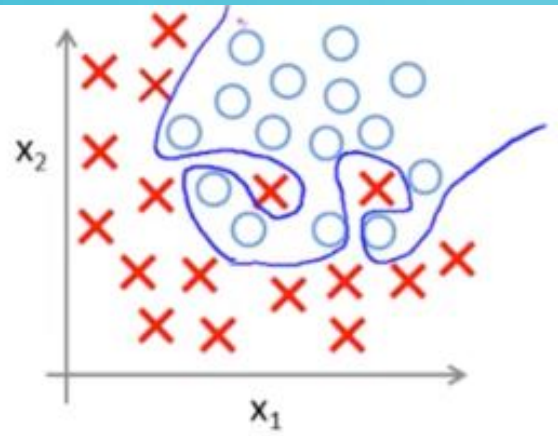


$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \dots)$$

Cost function:

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

COST FUNCTION



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \dots)$$

Cost function:

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

$\theta_1, \theta_2, \dots, \theta_n$

GRADIENT DESCENT

- As we saw earlier, it is exactly same Linear Regression:

$$\theta_j := \theta_j(1 - \alpha\frac{\lambda}{m}) - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

DECIDING WHAT TO TRY NEXT

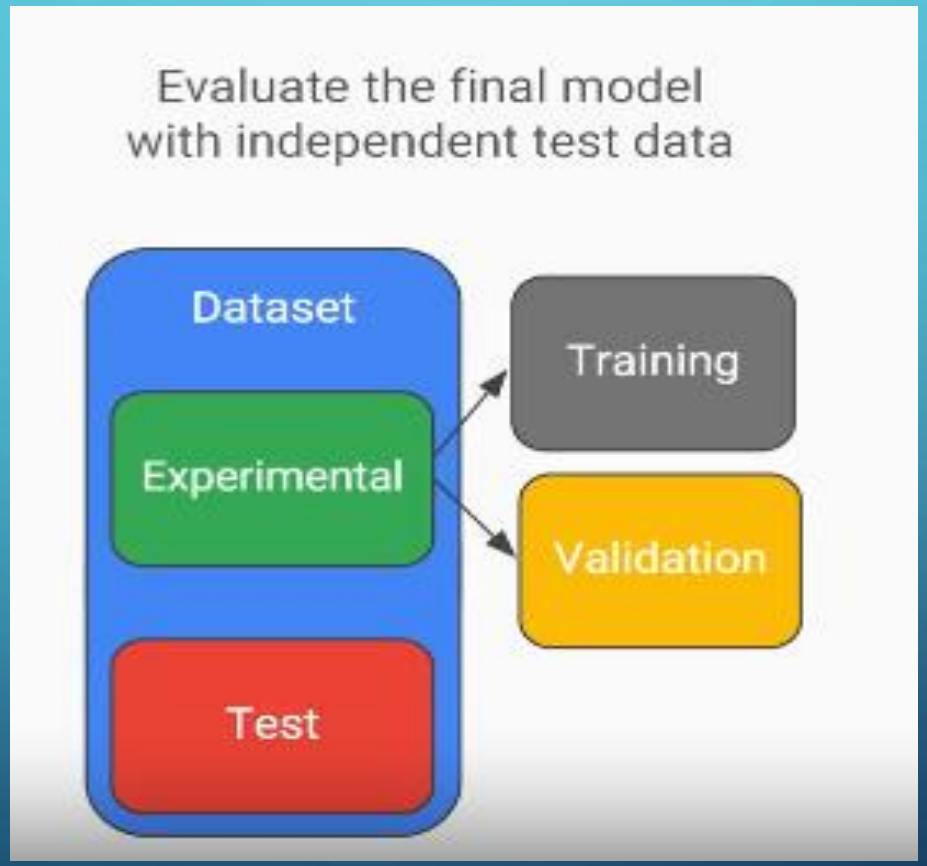
- Suppose you have implemented linear regression to predict housing prices
- But you find out that it makes unacceptably large errors in its predictions. What to try next?
- Options:

1. Get more training examples
2. Try smaller set of features
3. Try getting additional features
4. Try adding polynomial features
5. Increase/Decrease Regularization term

MACHINE LEARNING DIAGNOSTIC

- A test that you can run to gain insight what is/isn't working with a learning algorithm, and gain guidance as to how best to improve its performace
- Diagnostic can take time to implement, but doing so can be a very good use of your time

REVISIT DATA SPLIT



MODEL SELECTION

Model selection

1. $h_\theta(x) = \theta_0 + \theta_1 x$
2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$
3. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_3 x^3$

$\vdots$

10. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_{10} x^{10}$

Train/validation/test error

Training error:

$$J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Cross Validation error:

$$J_{cv}(\theta) = \frac{1}{2m_{cv}} \sum_{i=1}^{m_{cv}} (h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$

Test error:

$$J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} (h_\theta(x_{test}^{(i)}) - y_{test}^{(i)})^2$$

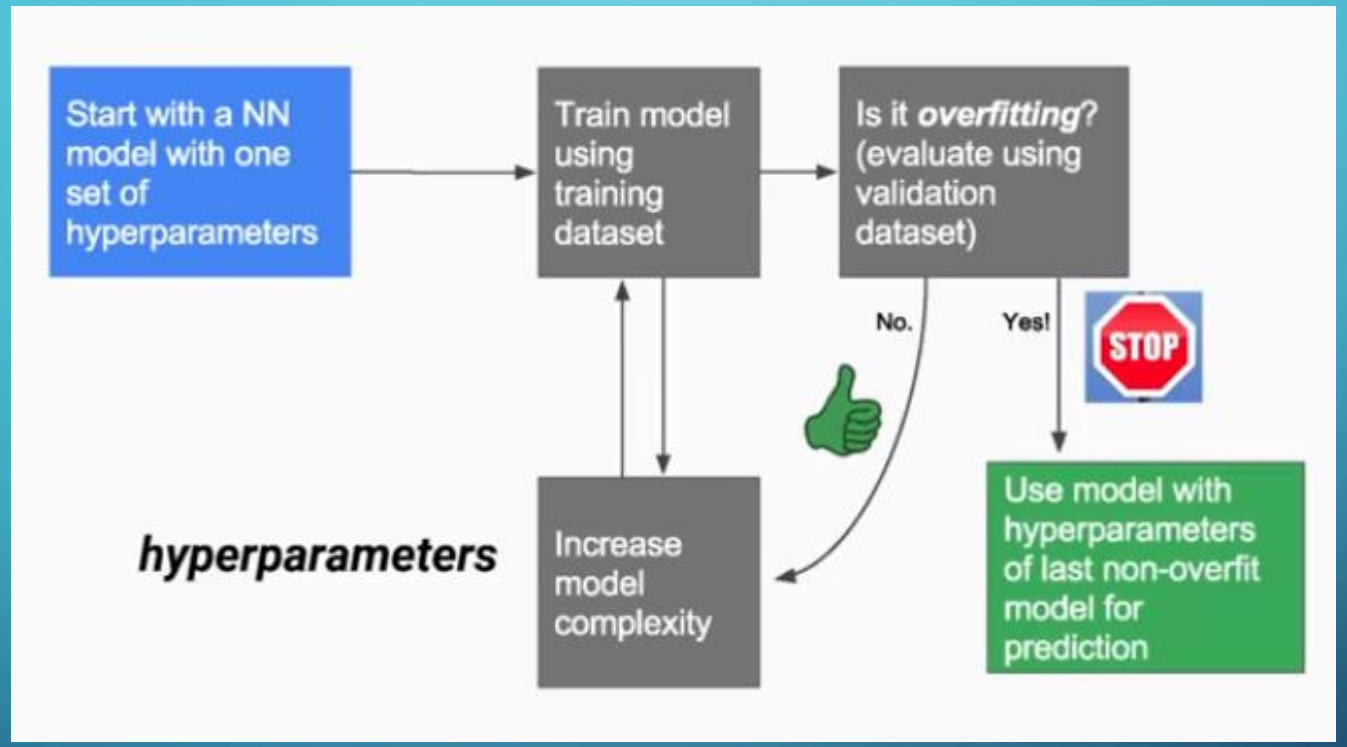
Model selection

1. $h_\theta(x) = \theta_0 + \theta_1 x$ → $\min_\theta J(\theta)$ → $\theta^{(1)}$ → $J_{cv}(\theta^{(1)})$
2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$ → $\theta^{(2)}$ → $J_{cv}(\theta^{(2)})$
3. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_3 x^3$ → $\theta^{(3)}$
   ⋮
10. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_{10} x^{10}$ → $\theta^{(10)}$ → $J_{cv}(\theta^{(10)})$

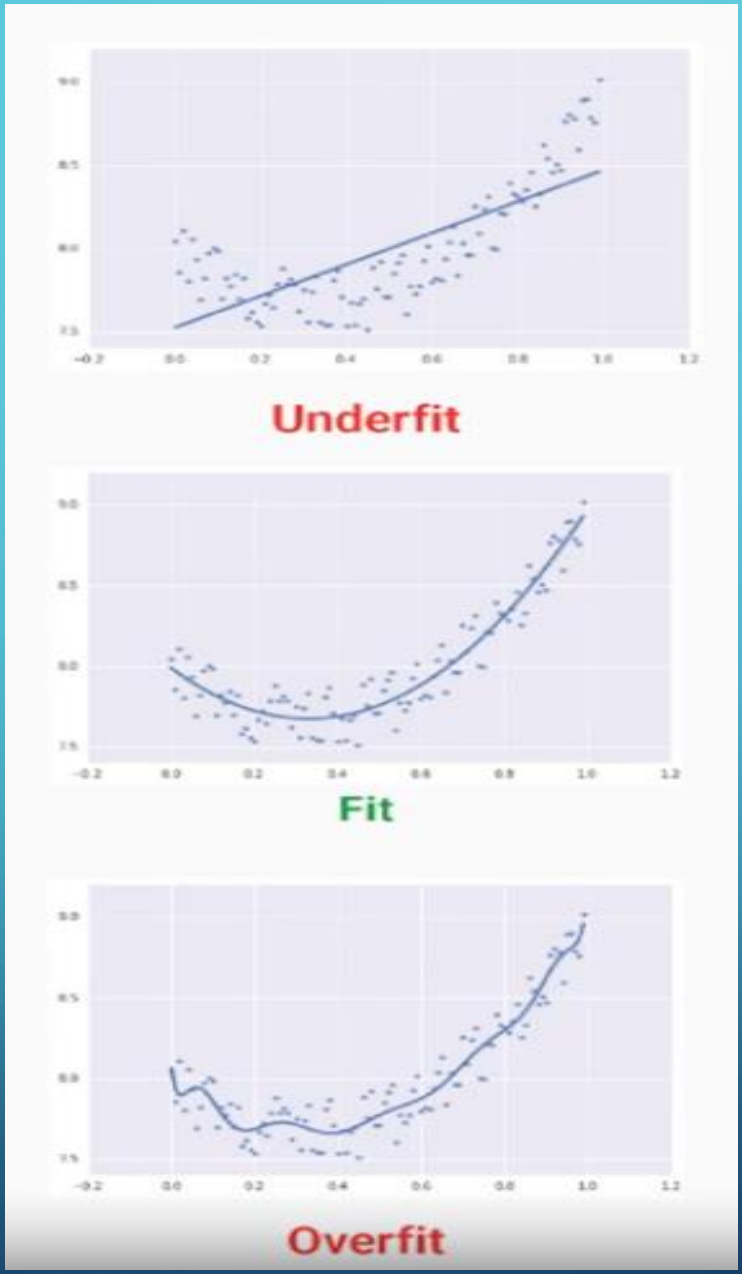
$J_{cv}(\theta^{(4)})$

Pick $\theta_0 + \theta_1 x_1 + \cdots + \theta_4 x^4$ ←

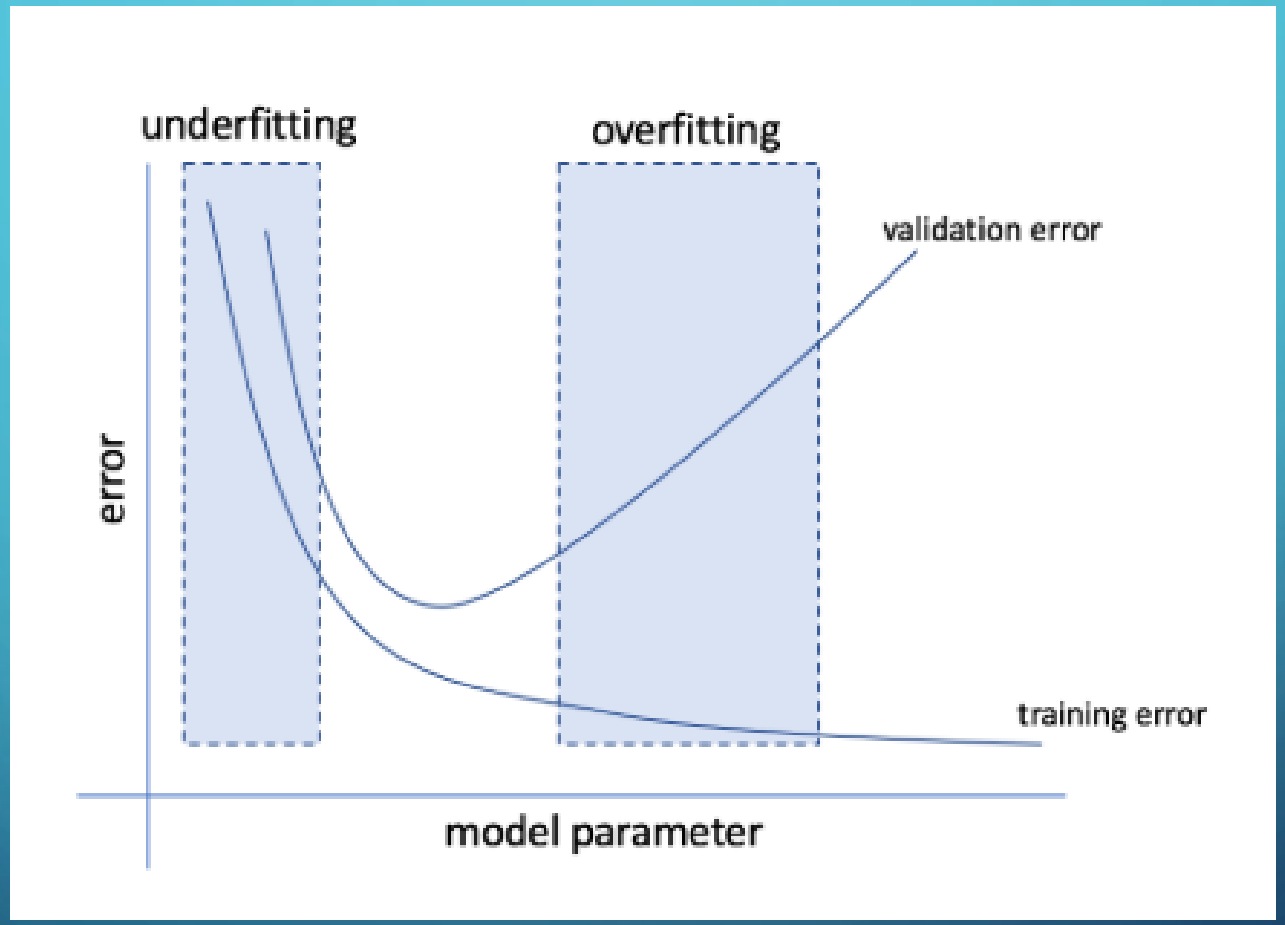
Estimate generalization error for test set $J_{test}(\theta^{(4)})$


Start with a NN model with one set of hyperparameters
Train model using training dataset
Is it *overfitting*? (evaluate using validation dataset)
No.
Yes!
STOP
Use model with hyperparameters of last non-overfit model for prediction
Increase model complexity
hyperparameters

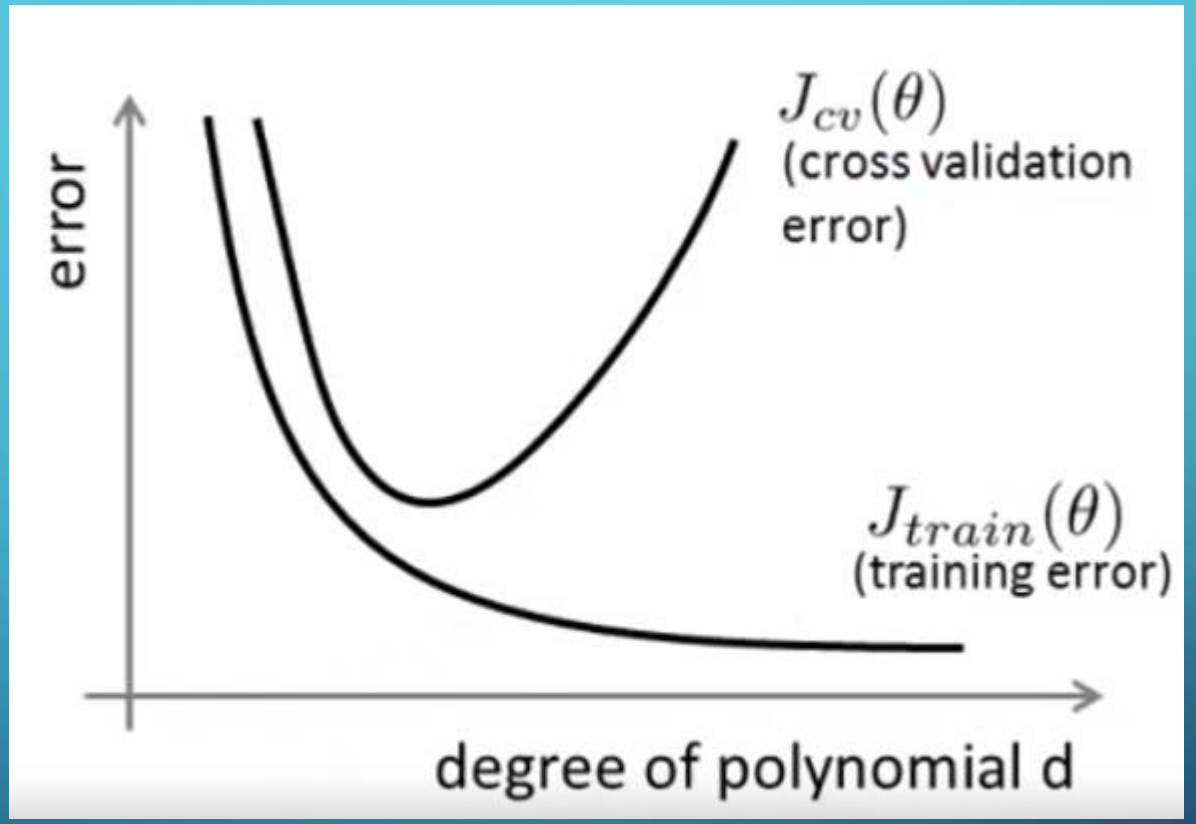
BIAS VS VARIANCE


Underfit
Fit
Overfit

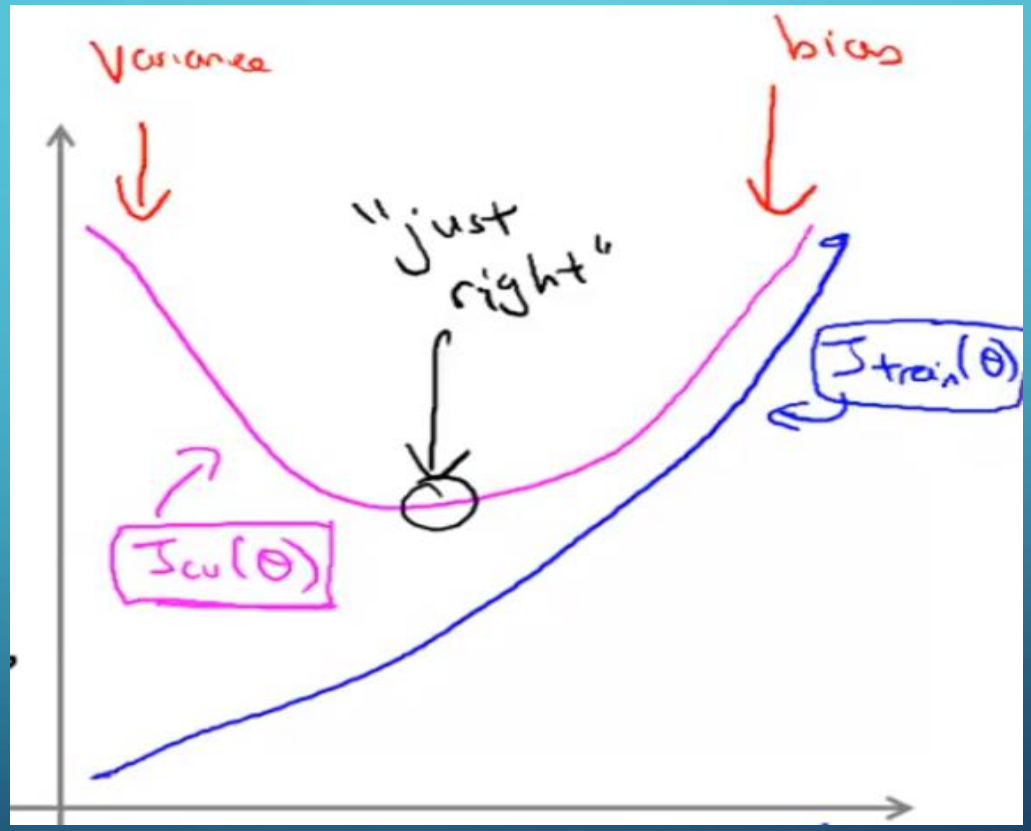
DEGREE POLYNOMIAL



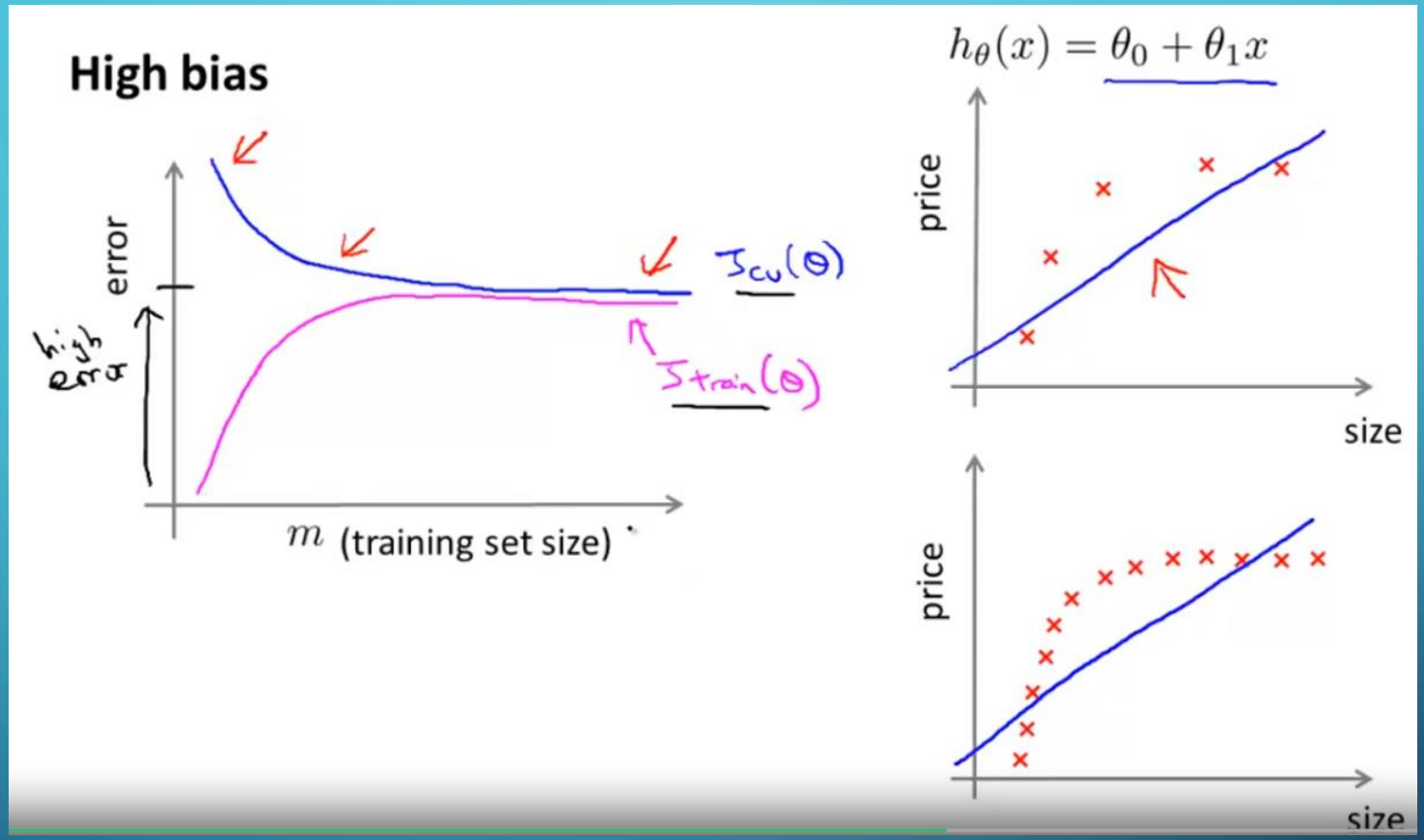
Degree Polynomial can be viewed as Model parameter


$J_{cv}(\theta)$
(cross validation error)
$J_{train}(\theta)$
(training error)
error
degree of polynomial d

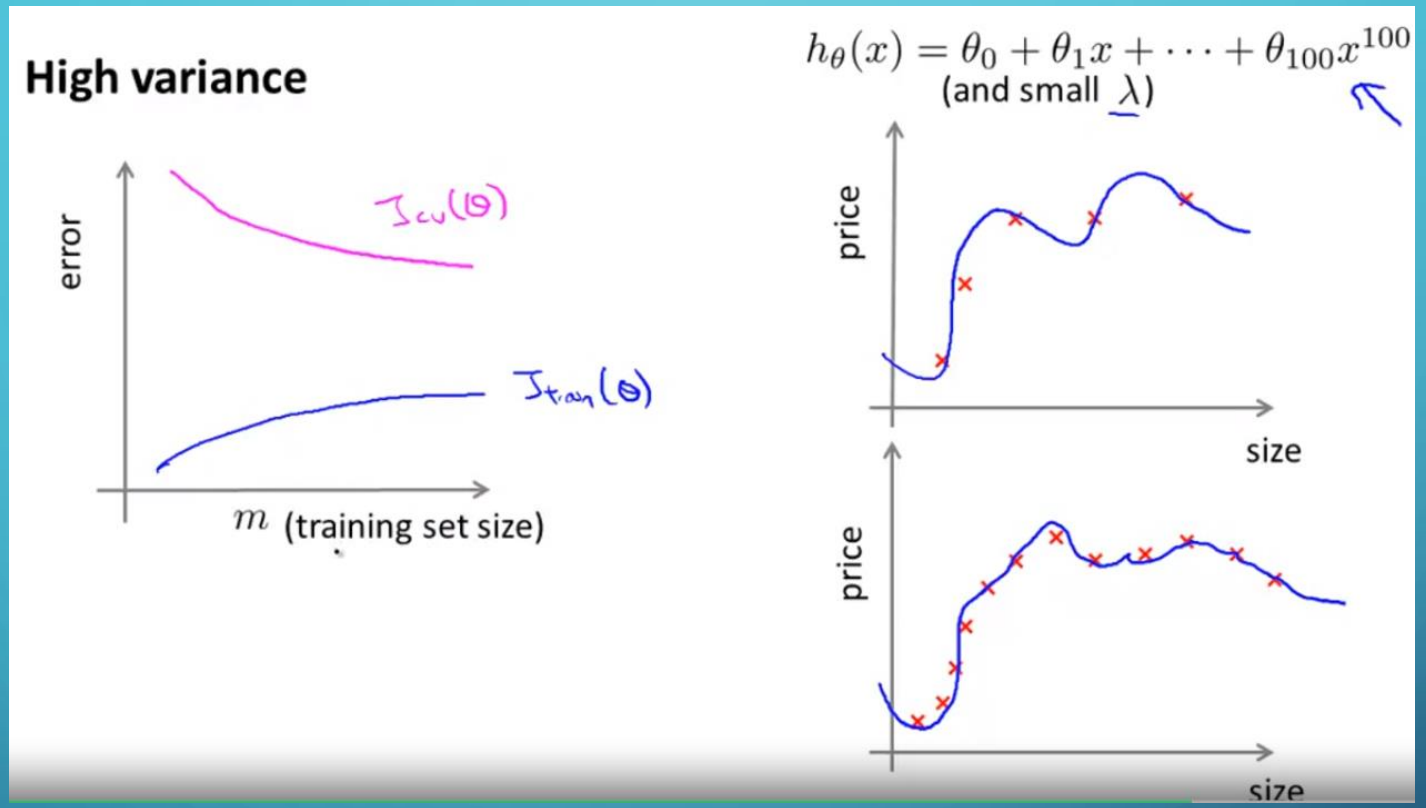
REGULARIZER



TRAINING SET SIZE



No use of getting more data



Getting more data is likely to help

TIPS TO RESOLVE ML PROBLEMS

Underfitting (Highly bias)	Overfitting (High Variance)
Try getting additional features	Get more training examples
Try adding polynomial features	Try smaller set of features
Try decreasing λ	Try increasing λ

PERFORMANCE METRICS

PROBLEM WITH LOSS FUNCTION

1000 parking spaces
990 of them are **taken**
10 are **available**

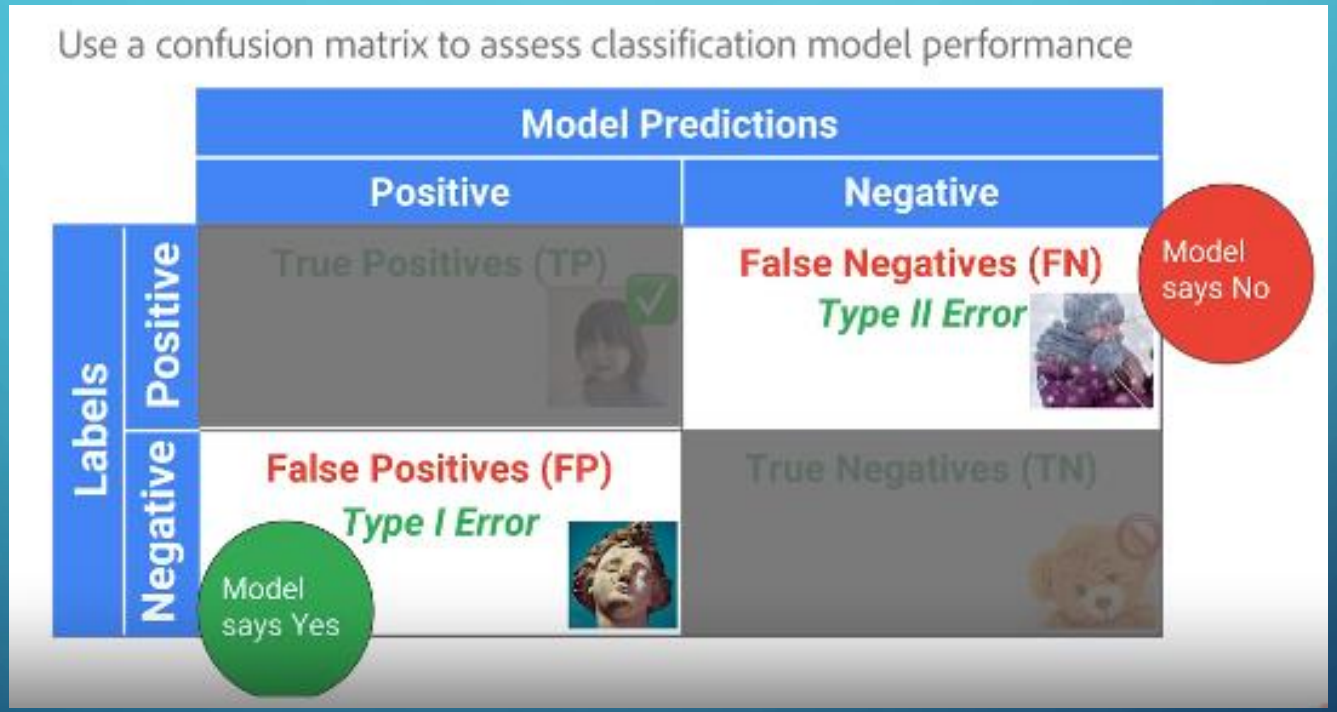
An ML model that always reported that a space was occupied would be right 99/100 times.

PERFORMANCE METRICS > LOSS FUNCTIONS

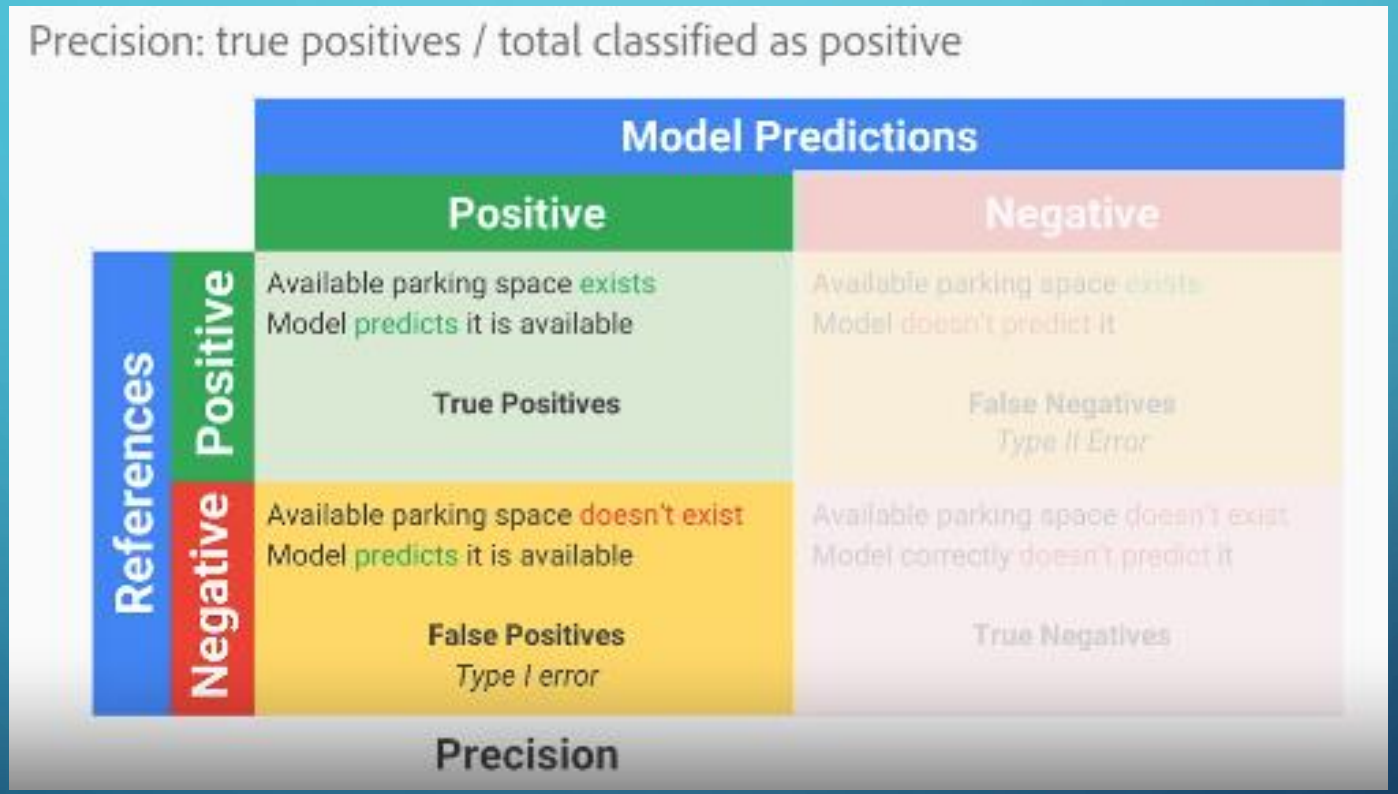
Performance metrics allow us to measure what matters

Loss Functions	Performance Metrics
• During training • Harder to understand • Indirectly connected to business goals	• After training • Easier to understand • Directly connected to business goals

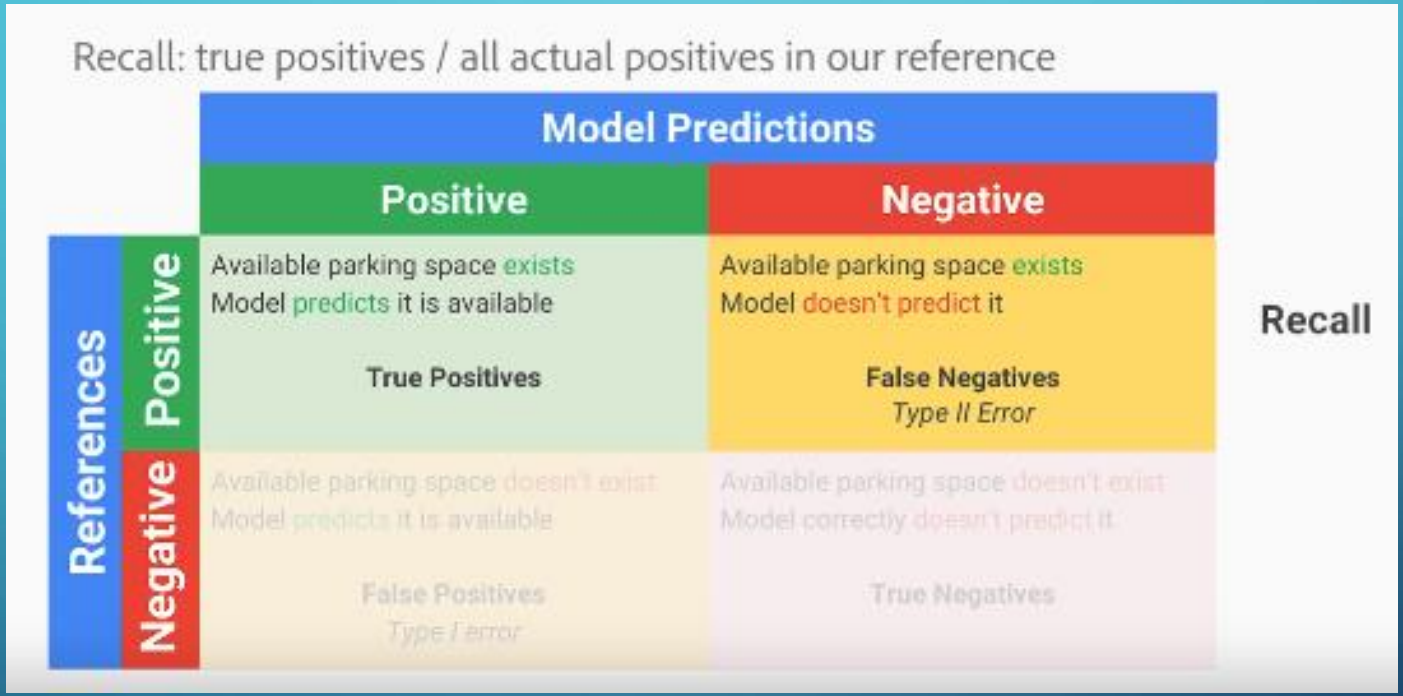
INTUITION OF CONFUSION MATRIX



PRECISION


Precision: true positives / total classified as positive
Model Predictions
Positive
Negative
References
Positive
Negative
Available parking space exists
Model predicts it is available
True Positives
Available parking space exists
Model doesn't predict it
False Negatives
Type II Error
Available parking space doesn't exist
Model predicts it is available
False Positives
Type I error
Available parking space doesn't exist
Model correctly doesn't predict it
True Negatives
Precision

RECALL


Recall: true positives / all actual positives in our reference
Model Predictions
Positive
Negative
References
Positive
Negative
Available parking space exists
Model predicts it is available
True Positives
Available parking space exists
Model doesn't predict it
False Negatives
Type II Error
Available parking space doesn't exist
Model predicts it is available
False Positives
Type I error
Available parking space doesn't exist
Model correctly doesn't predict it
True Negatives
Recall

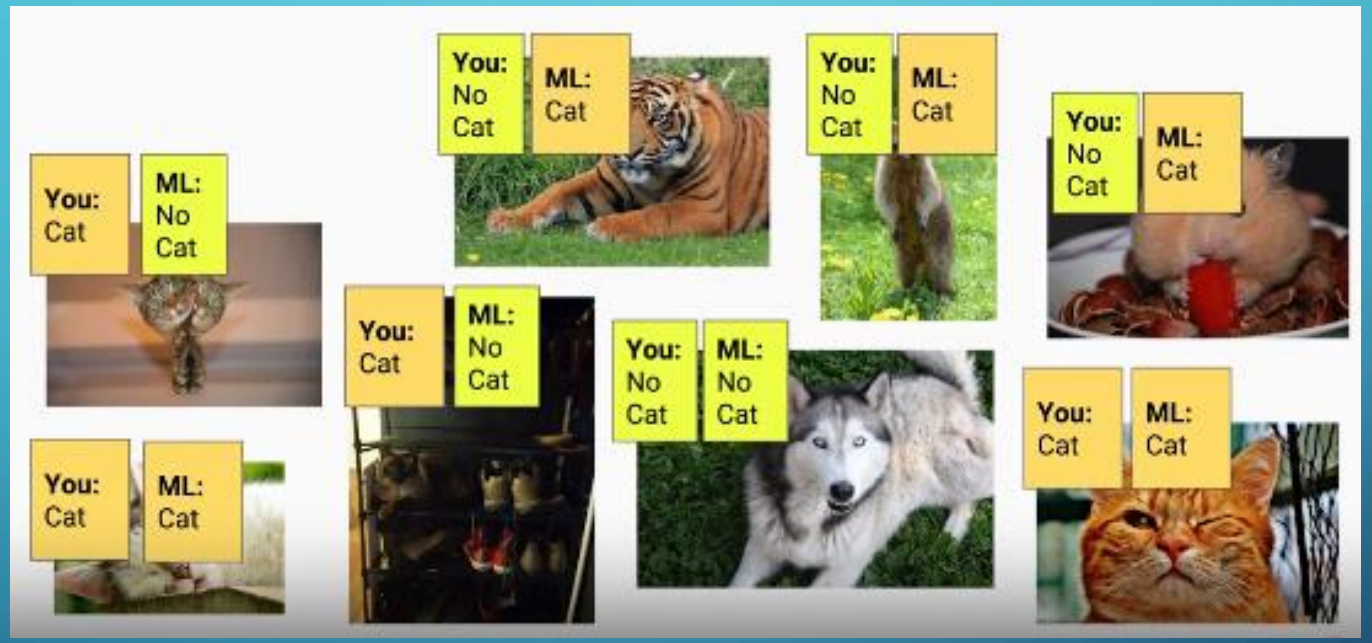
PRACTICE TIME

IDENTIFY CATS IN THE IMAGE AS CAT/ NOT CAT


You:
Cat
You:
No
Cat
You:
No
Cat
You:
No
Cat
You:
Cat
You:
Cat
You:
No
Cat
You:
Cat

HYPOTHETICAL RESULT FROM OUR ML MODEL

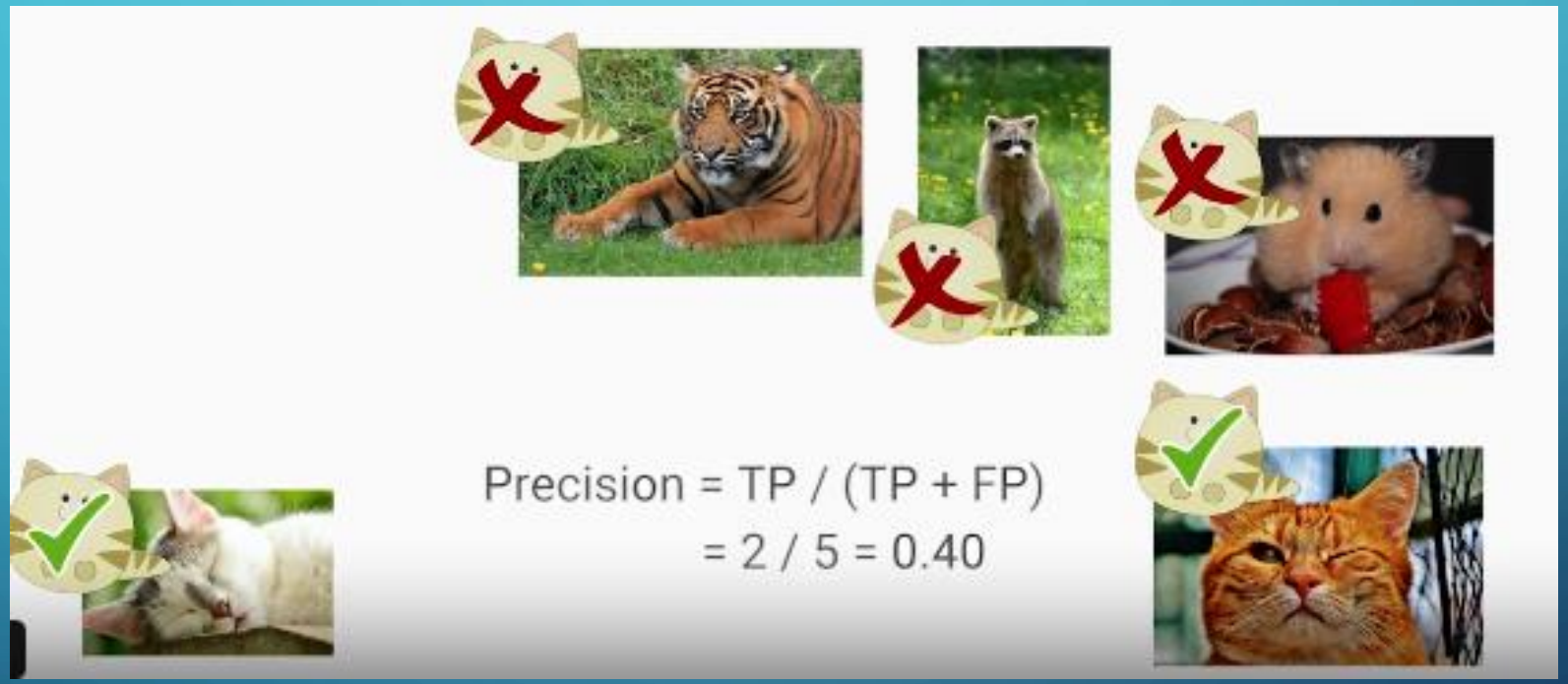
COMPARISON


You: Cat
ML: No Cat
You: No Cat
ML: Cat
You: No Cat
ML: Cat
You: No Cat
ML: Cat
You: Cat
ML: No Cat
You: No Cat
ML: No Cat
You: Cat
ML: Cat
You: Cat
ML: Cat

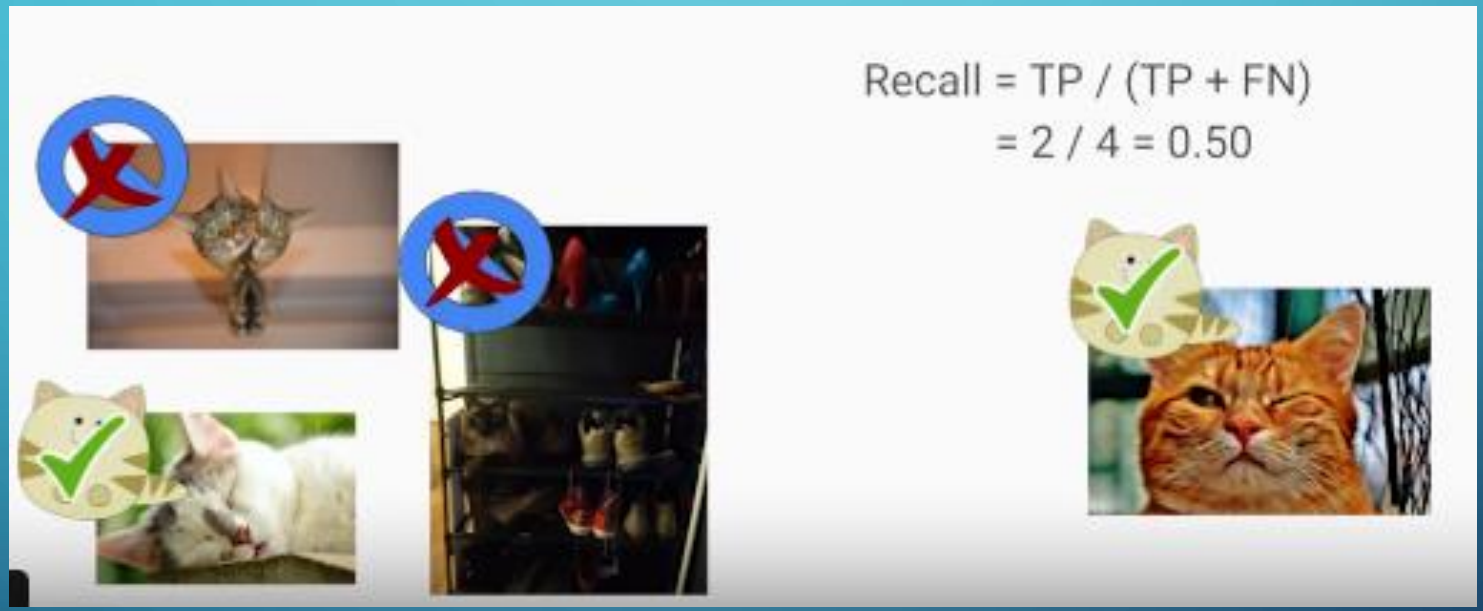
ACCURACY


Accuracy = 3 / 8
= 0.375
You: Cat
ML: No Cat
You: No Cat
ML: Cat
You: No Cat
ML: Cat
You: No Cat
ML: Cat
You: Cat
ML: No Cat
You: No Cat
ML: No Cat
You: Cat
ML: Cat
You: Cat

PRECISION



RECALL



SUMMARY

- **Precision**: Makes sure whatever model predicts is correct, even if that means it misses out on some positives
- Recall: Makes sure that model correctly predicts maximum number of positives, even if that means it has more false positives.

TRADE OFF BETWEEN PRECISION AND RECALL

How to compare precision/recall numbers?

	Precision(P)	Recall (R)
Algorithm 1	0.5	0.4
Algorithm 2	0.7	0.1
Algorithm 3	0.02	1.0

AVERAGE WONT WORK!!

	Precision(P)	Recall (R)	Average
Algorithm 1	0.5	0.4	0.45
Algorithm 2	0.7	0.1	0.4
Algorithm 3	0.02	1.0	0.51

F1 SCORE: WHAT AN IDEA SIRJEE!!!

- F1 score is given by: $\frac{2PR}{P+R}$

….. where P=Precision, R=Recall

	Precision(P)	Recall (R)	Average	F_1 Score
Algorithm 1	0.5	0.4	0.45	0.444
Algorithm 2	0.7	0.1	0.4	0.175
Algorithm 3	0.02	1.0	0.51	0.0392

TYPES OF GRADIENT DESCENT

- Batch Gradient Descent
- Mini-Batch Gradient Descent
- Stochastic Gradient Descent

WHAT WE KNOW?

- We have already seen the Batch Gradient Descent

BATCH GRADIENT DESCENT

Linear regression with gradient descent

$$h_\theta(x) = \sum_{j=0}^{n} \theta_j x_j$$

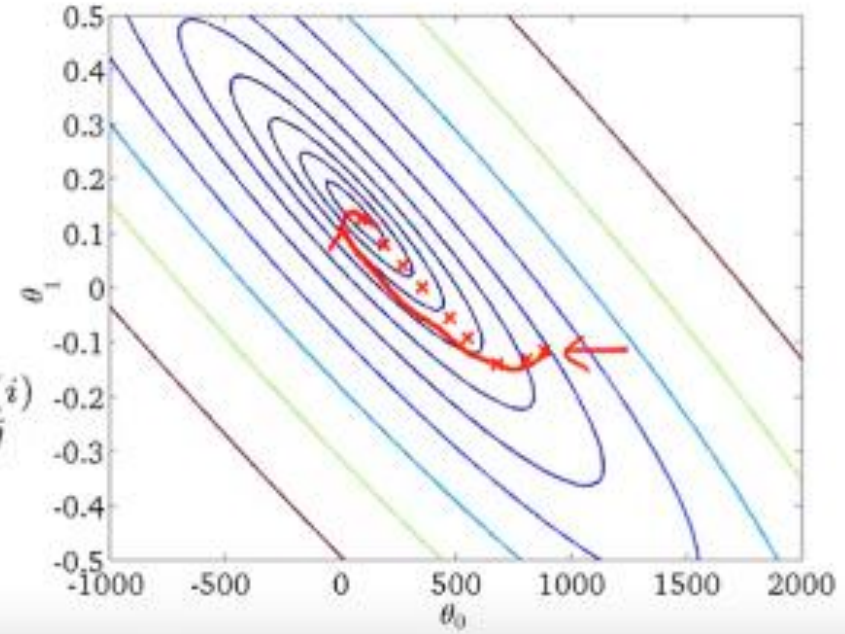
$$J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(for every $j = 0, \ldots, n$)

}

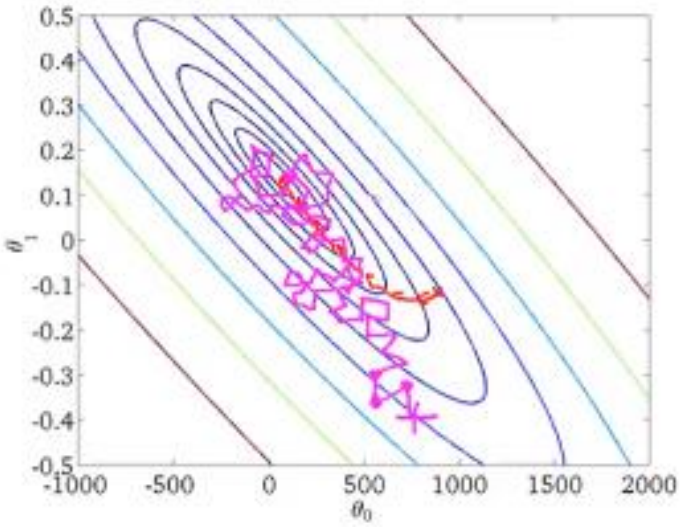


STOCHASTIC GRADIENT DESCENT

Stochastic gradient descent

1. Randomly shuffle (reorder) training examples

2. Repeat {
   for $i := 1, \ldots, m$ {
   $\theta_j := \theta_j - \alpha(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$
   (for every $j = 0, \ldots, n$)
   }
   }



Batch gradient descent: Use all m examples in each iteration

Stochastic gradient descent: Use 1 example in each iteration

MINI-BATCH GRADIENT DESCENT

- Use b examples in each iteration … where b=mini-batch size

Mini-batch gradient descent

Say $b = 10, m = 1000$.

Repeat {

for $i = 1, 11, 21, 31, \ldots, 991$ {

$$\theta_j := \theta_j - \alpha \frac{1}{10} \sum_{k=i}^{i+9} (h_\theta(x^{(k)}) - y^{(k)}) x_j^{(k)}$$

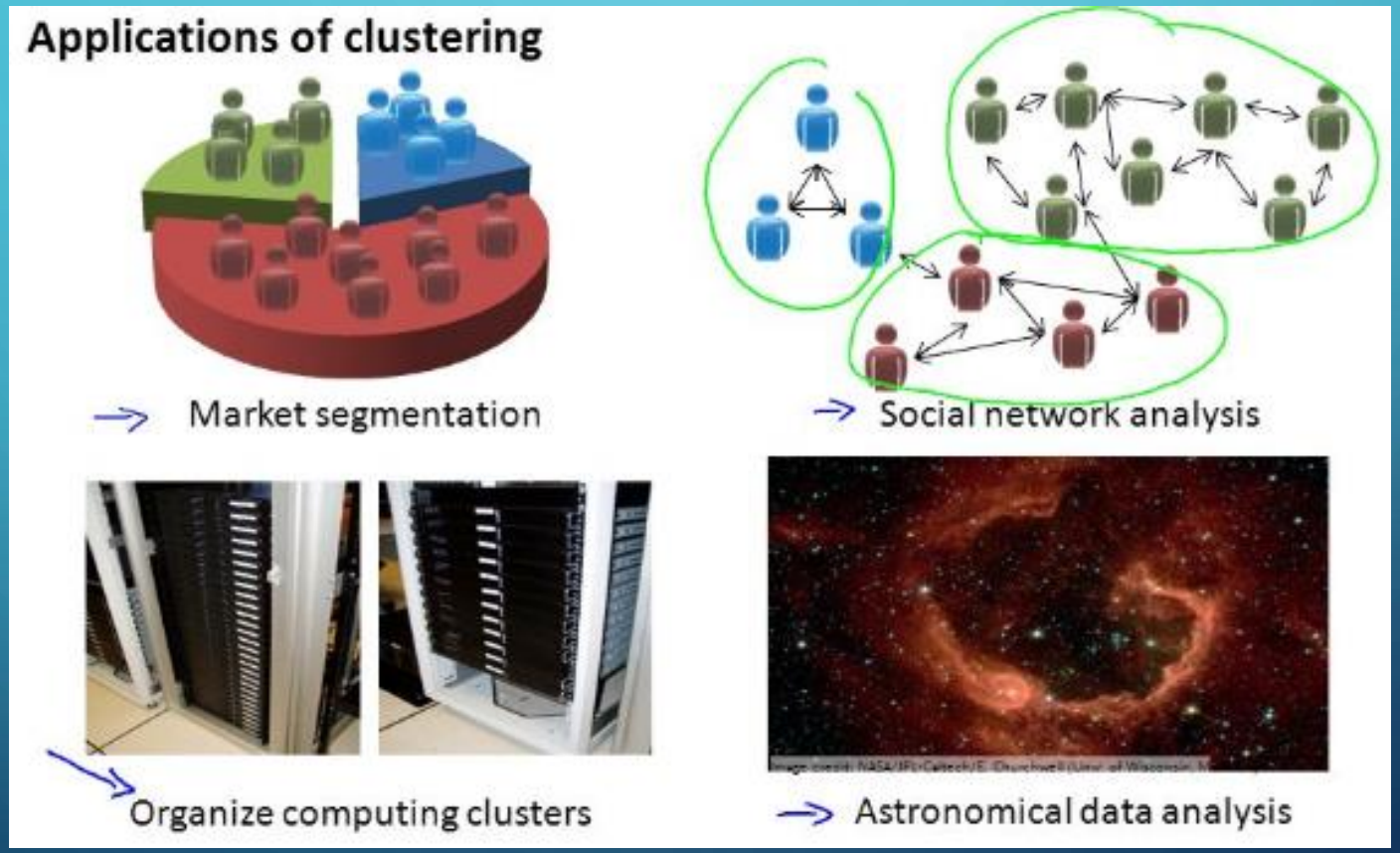
(for every $j = 0, \ldots, n$)

}

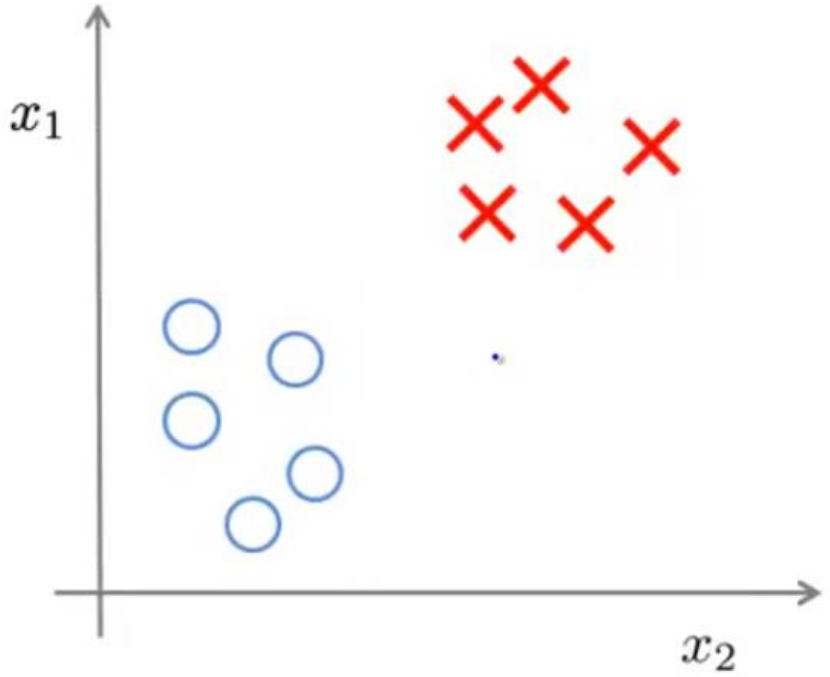
}

UNSUPERVISED LEARNING

APPLICATIONS

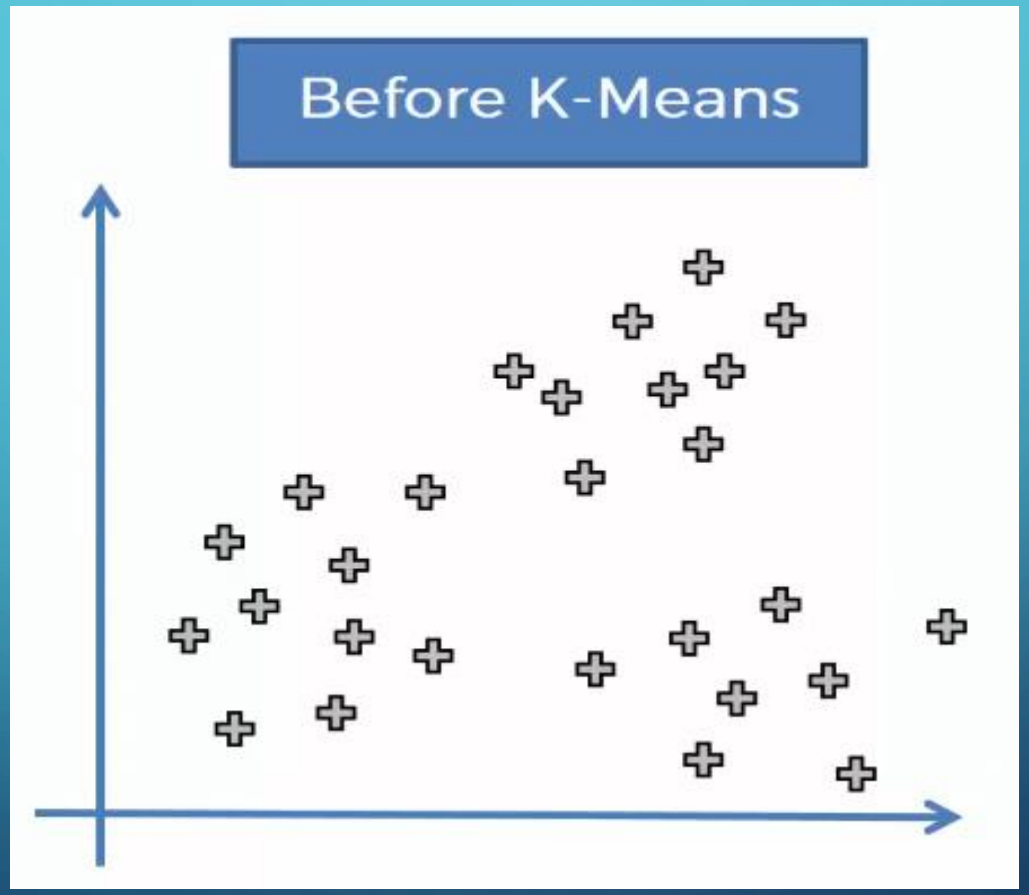


WHAT WE KNOW?

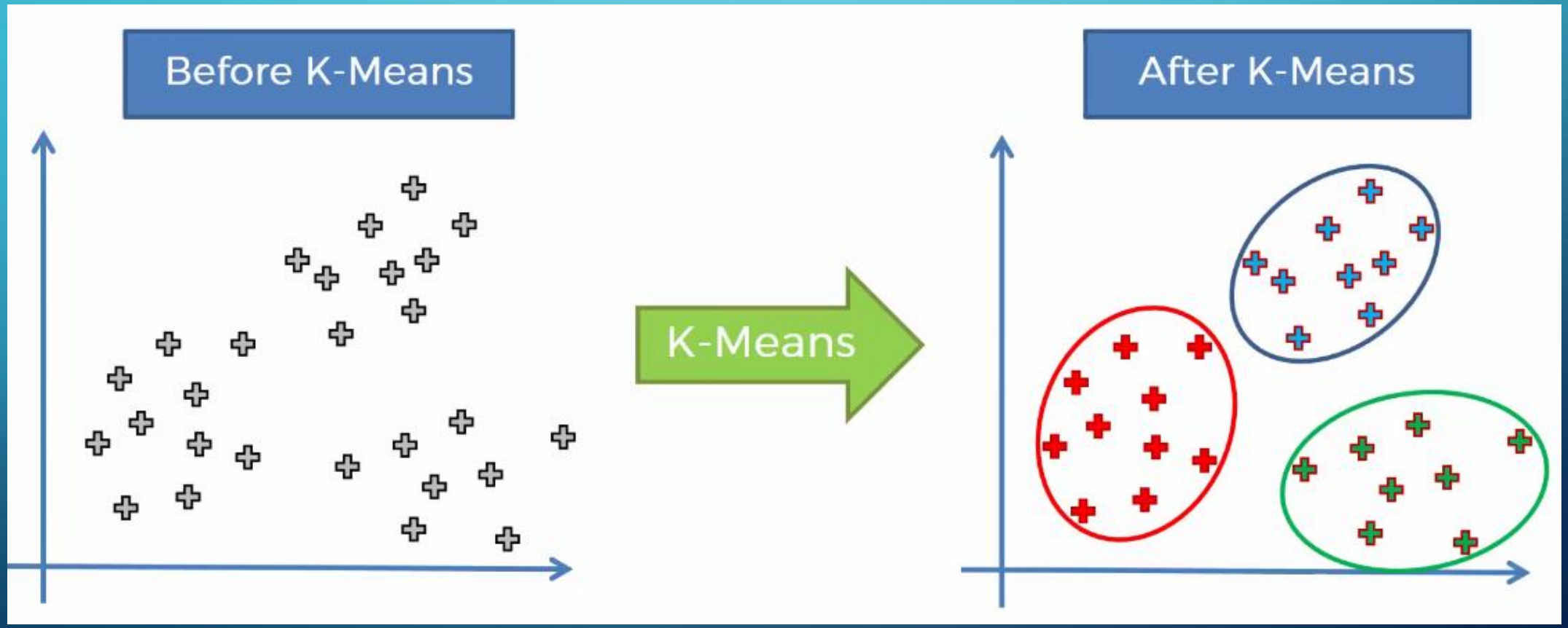


Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), \ldots, (x^{(m)}, y^{(m)})\}$

NO LABELS



CLUSTERS



K-MEANS ALGORITHM

STEP 1: Choose the number K of clusters

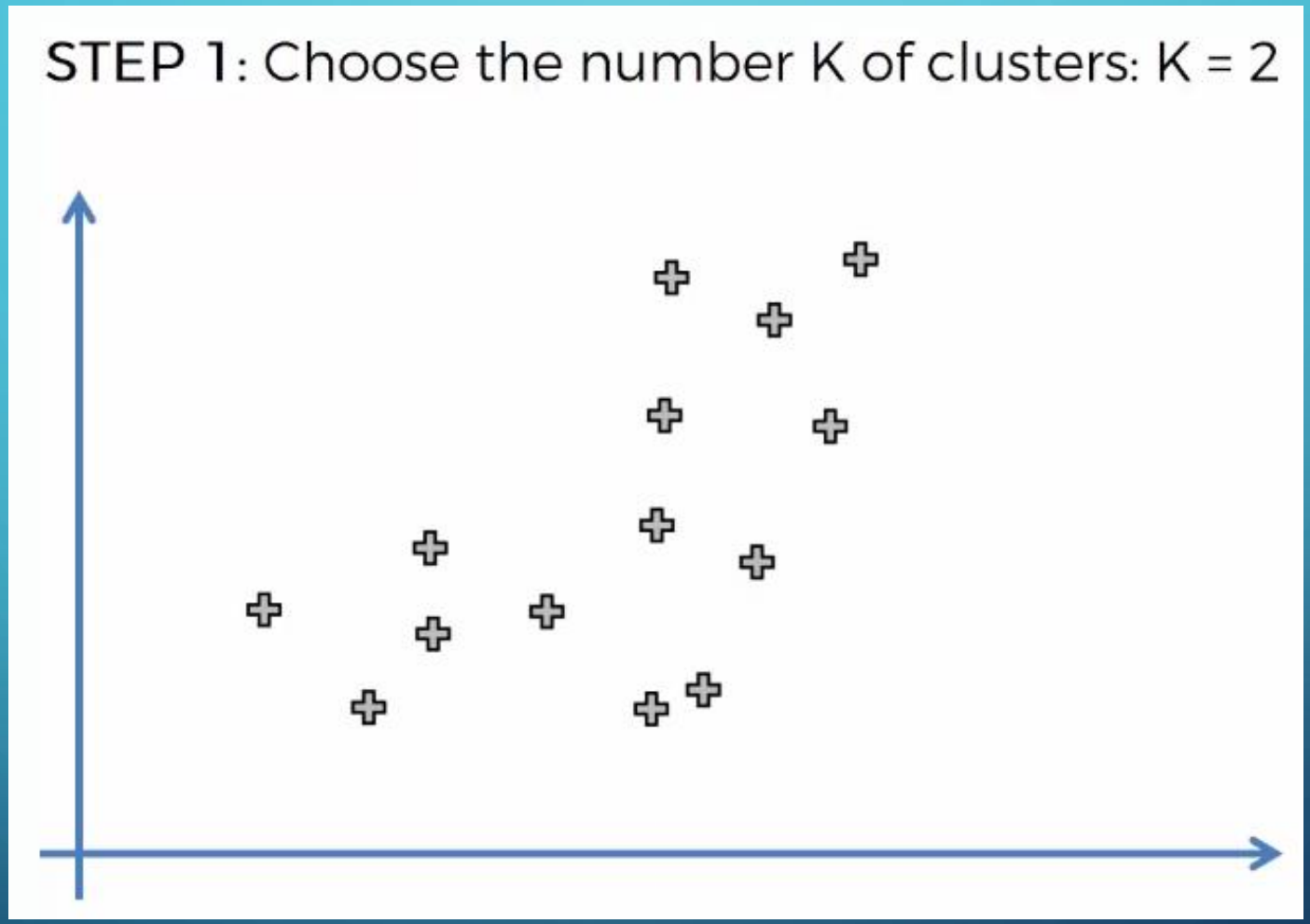
STEP 2: Select at random K points, the centroids (not necessarily from your dataset)

STEP 3: Assign each data point to the closest centroid → That forms K clusters

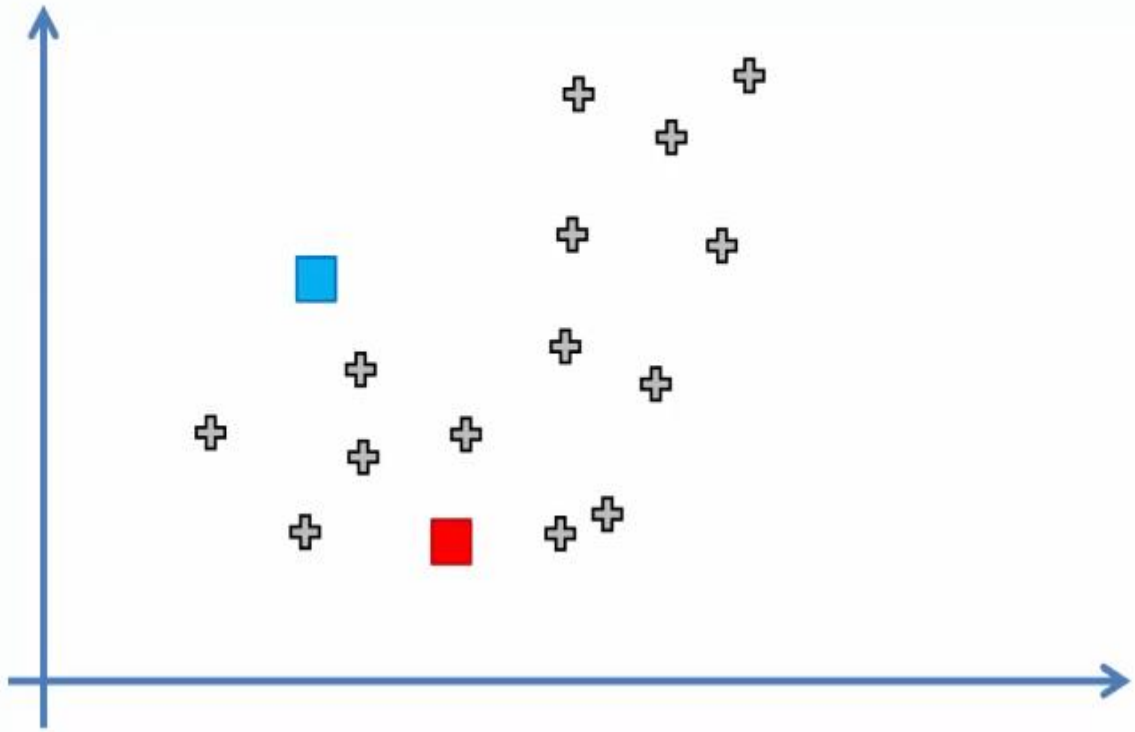
STEP 4: Compute and place the new centroid of each cluster

STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.

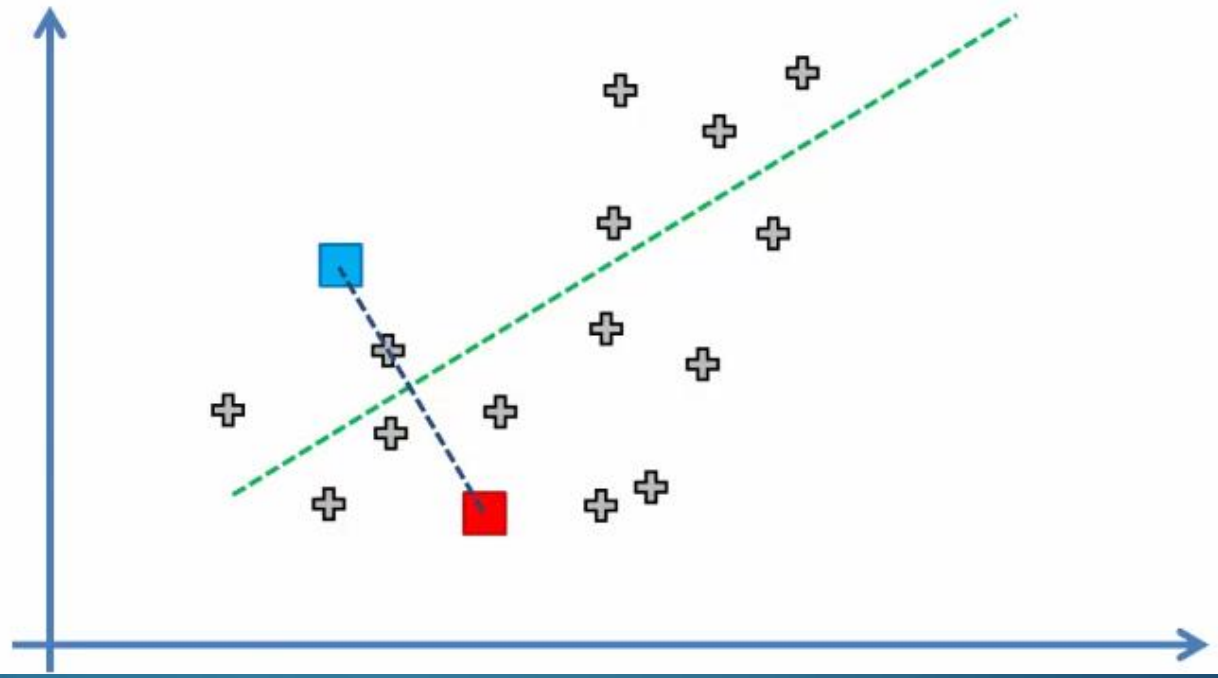
Your Model is Ready


STEP 1: Choose the number K of clusters: K = 2

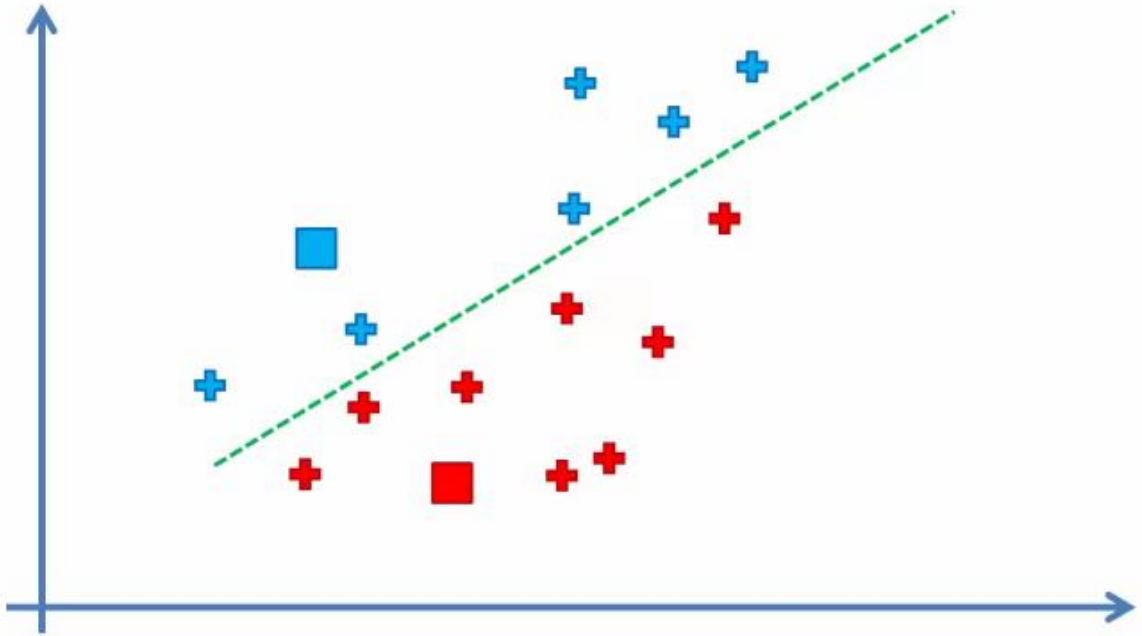
STEP 2: Select at random K points, the centroids (not necessarily from your dataset)



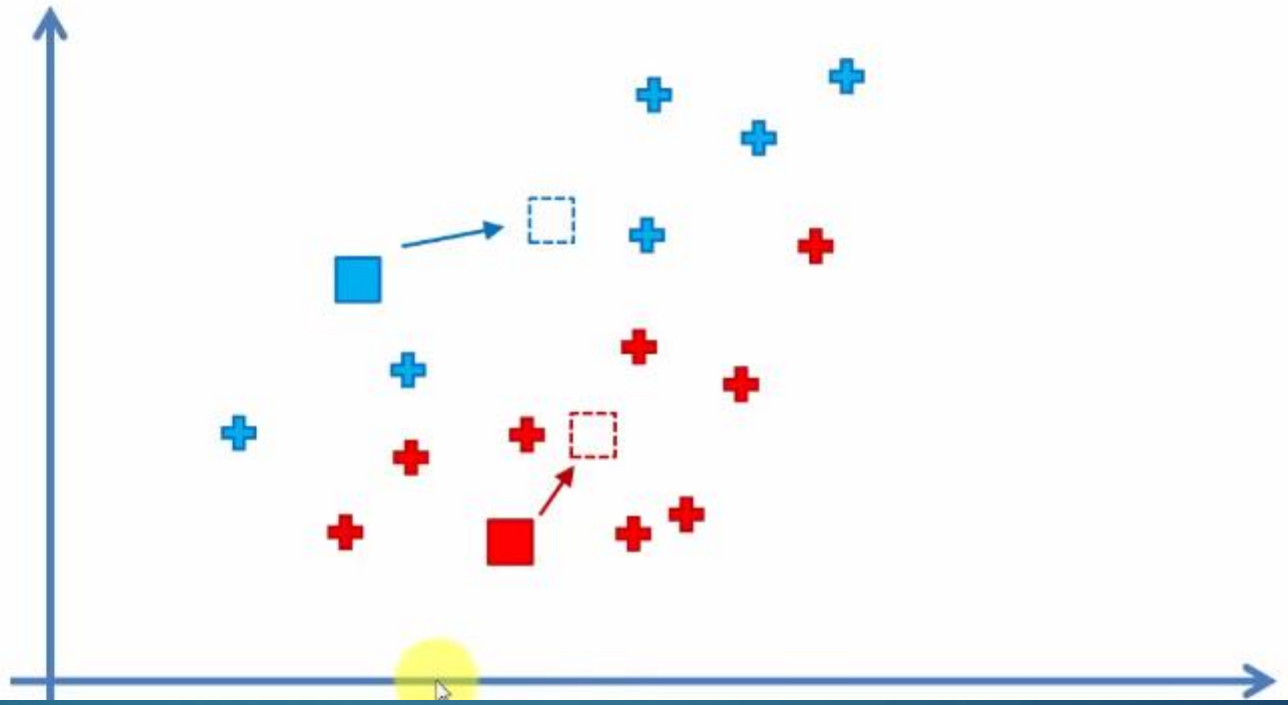
STEP 3: Assign each data point to the closest centroid ➡ That forms K clusters



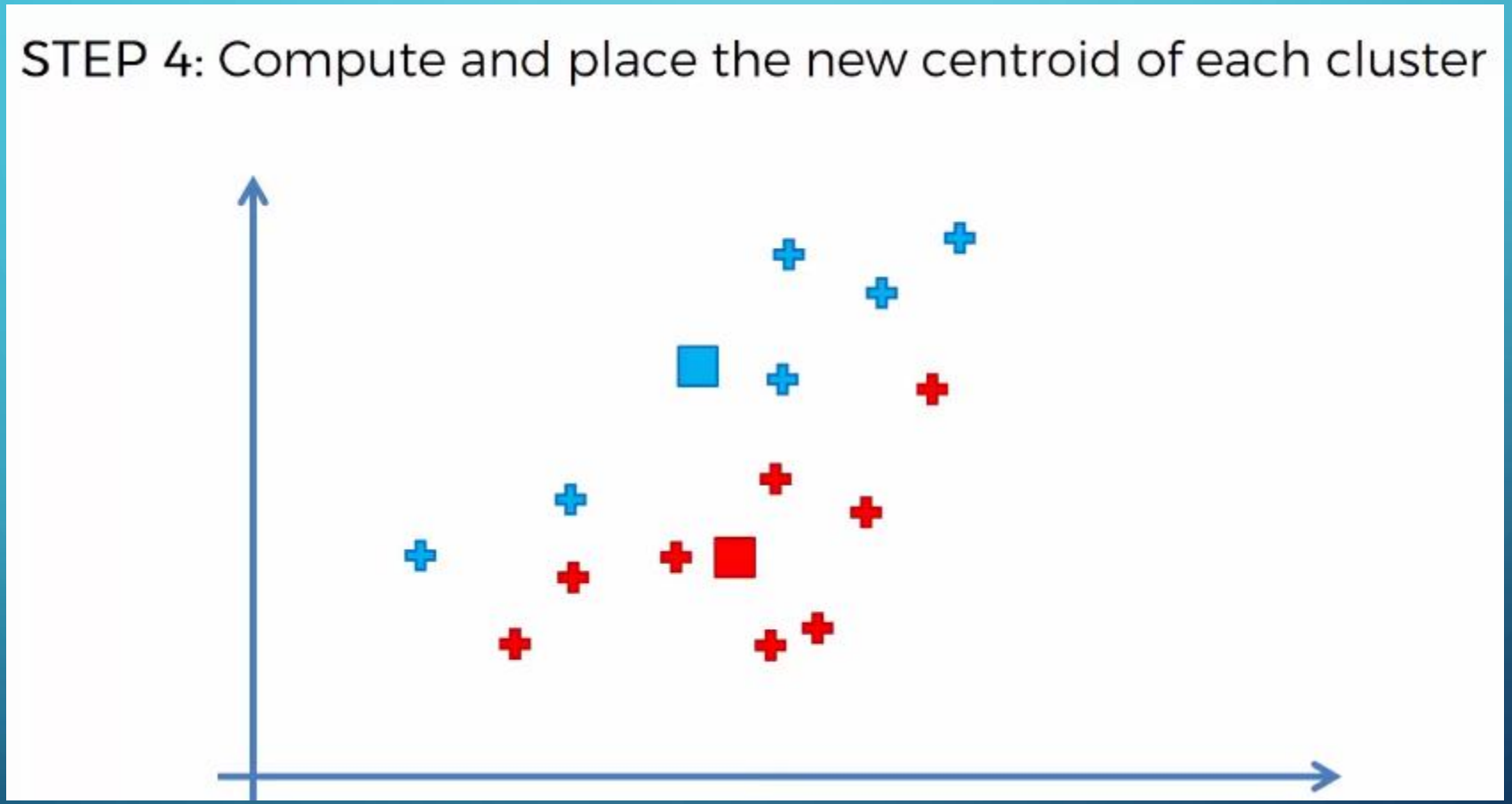
STEP 3: Assign each data point to the closest centroid → That forms K clusters



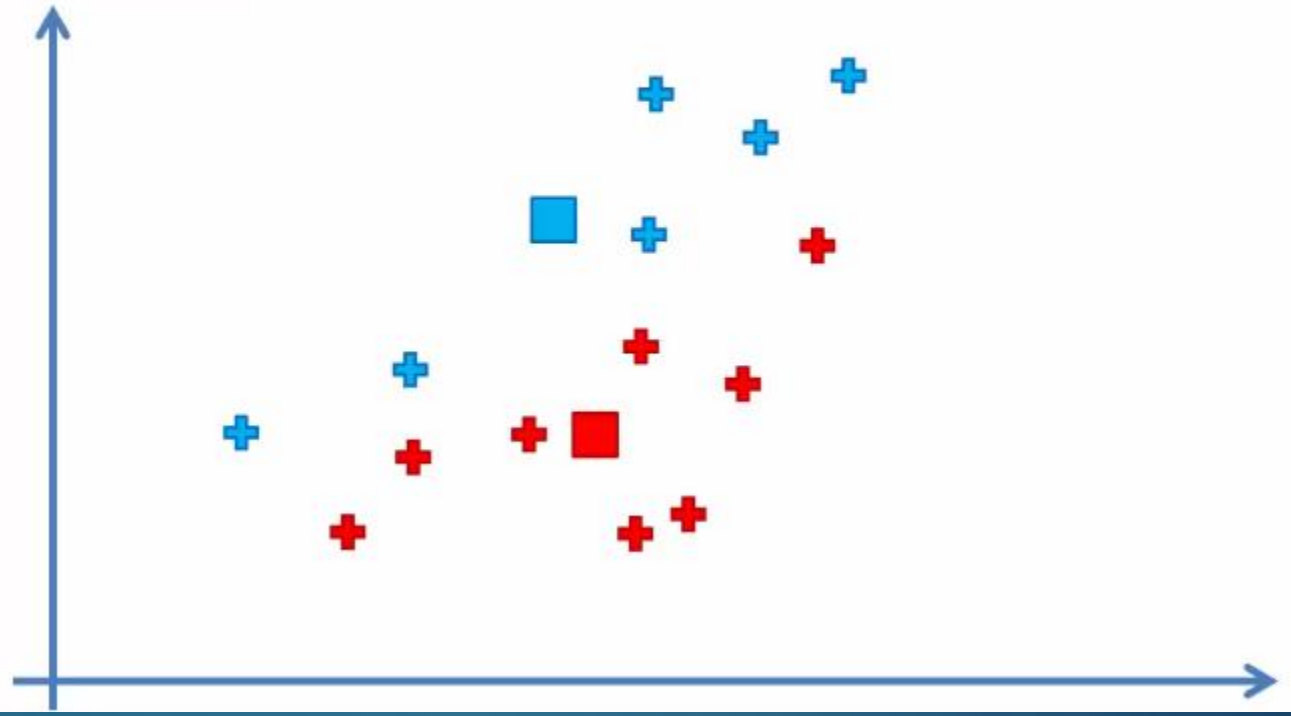
STEP 4: Compute and place the new centroid of each cluster



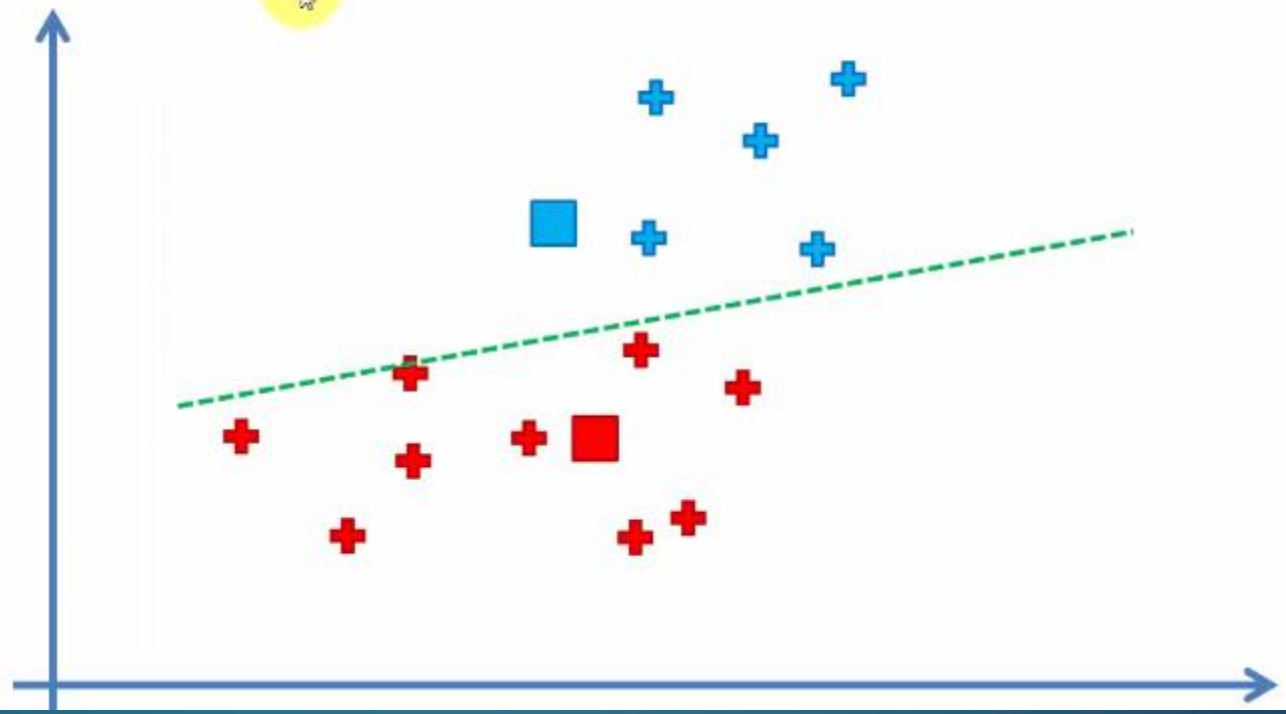
STEP 4: Compute and place the new centroid of each cluster



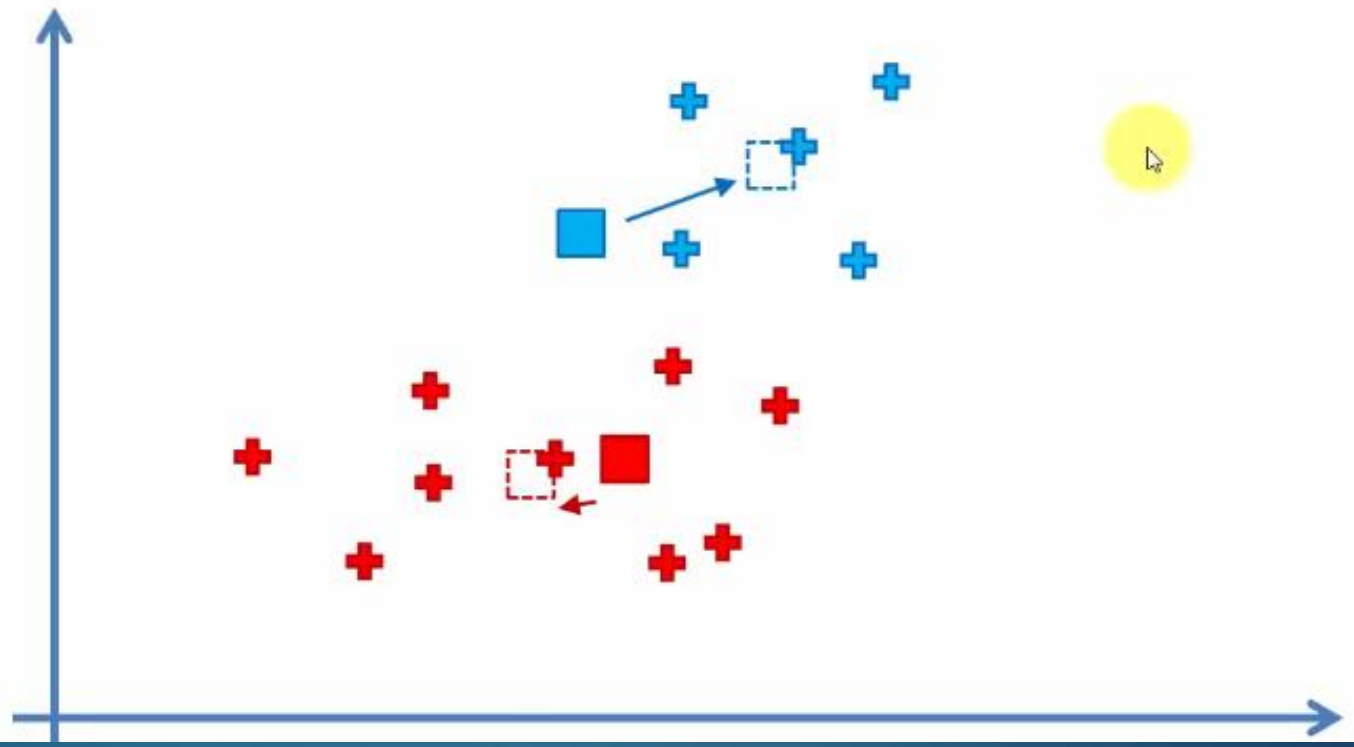
STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.



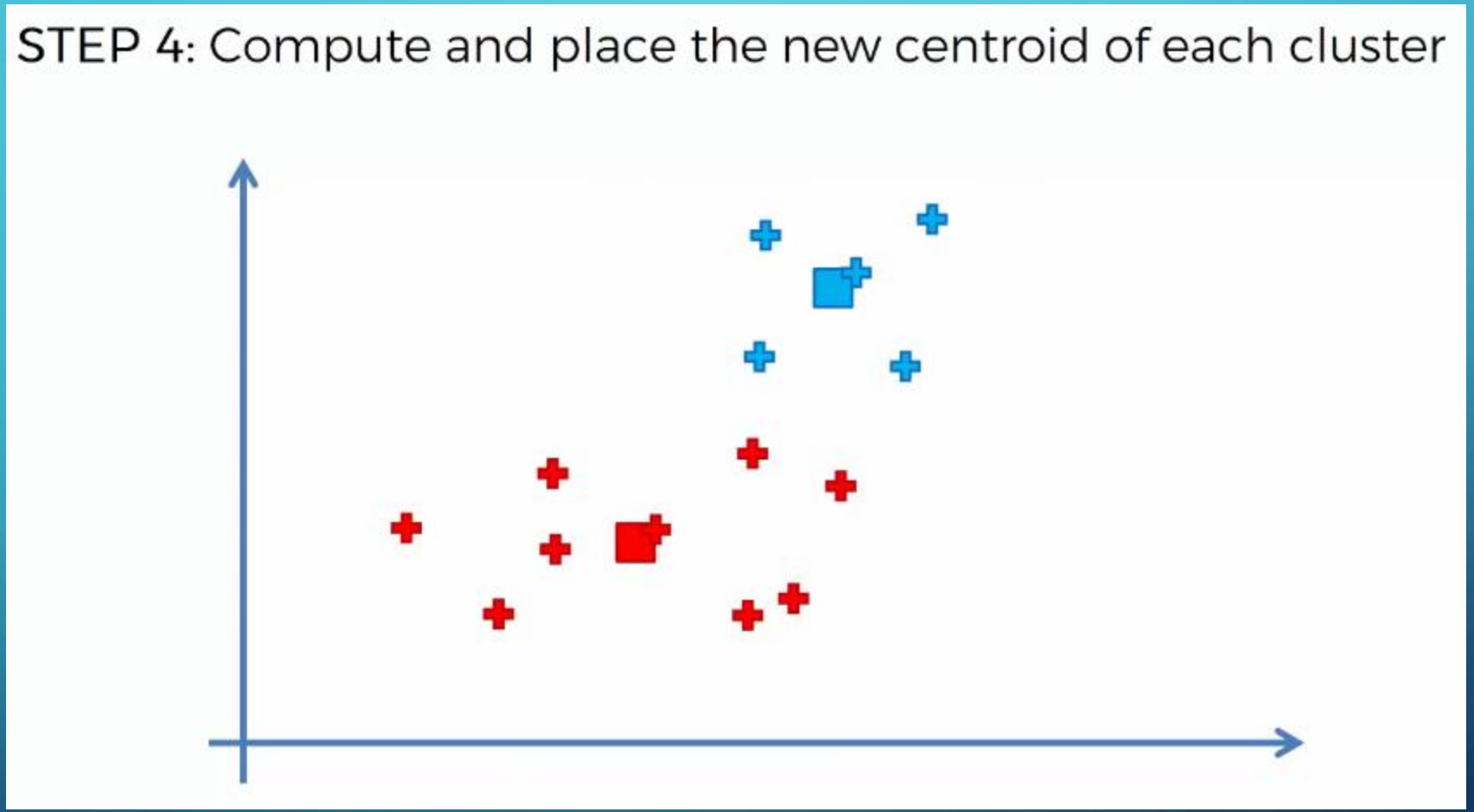
STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.



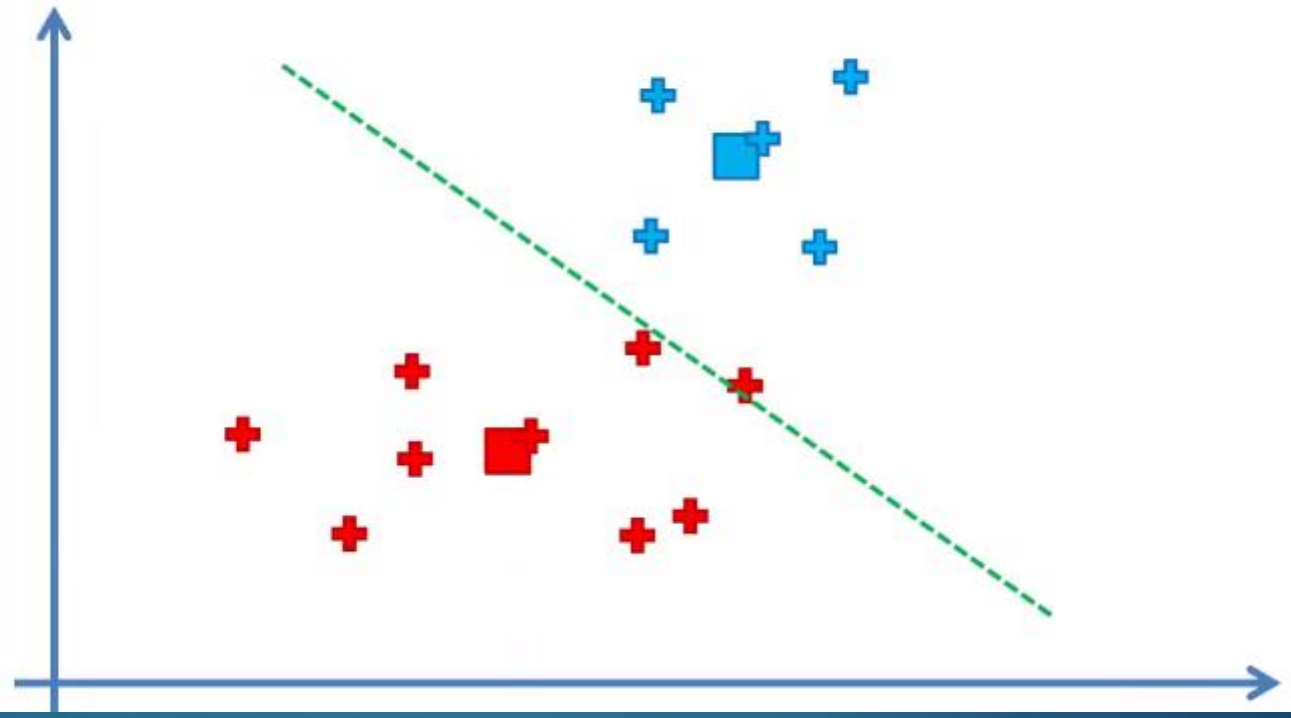
STEP 4: Compute and place the new centroid of each cluster



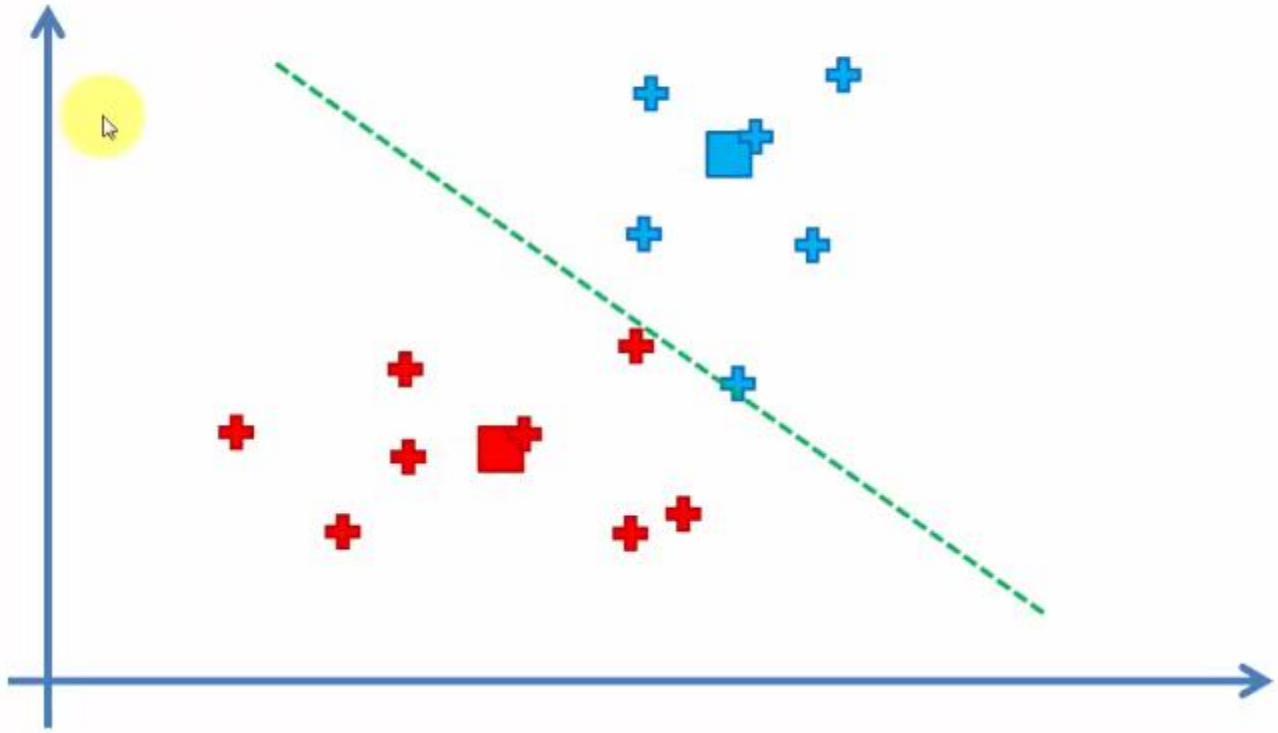
STEP 4: Compute and place the new centroid of each cluster



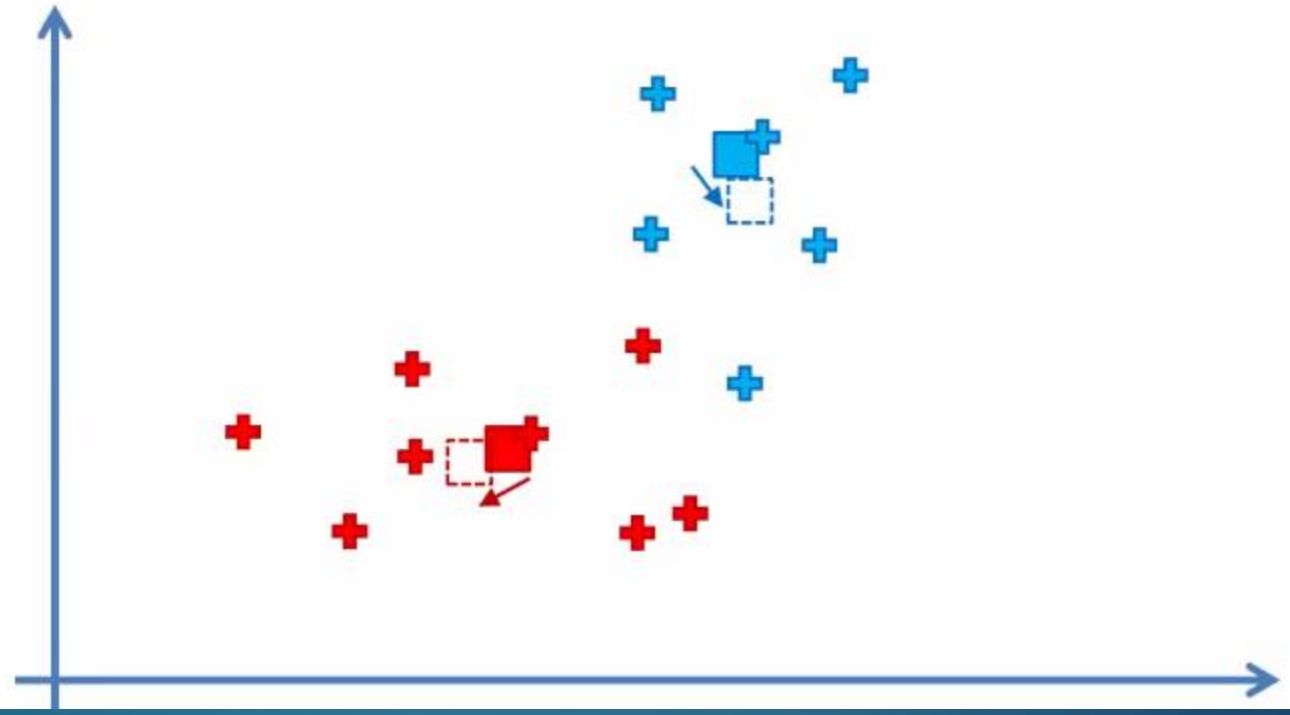
STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.



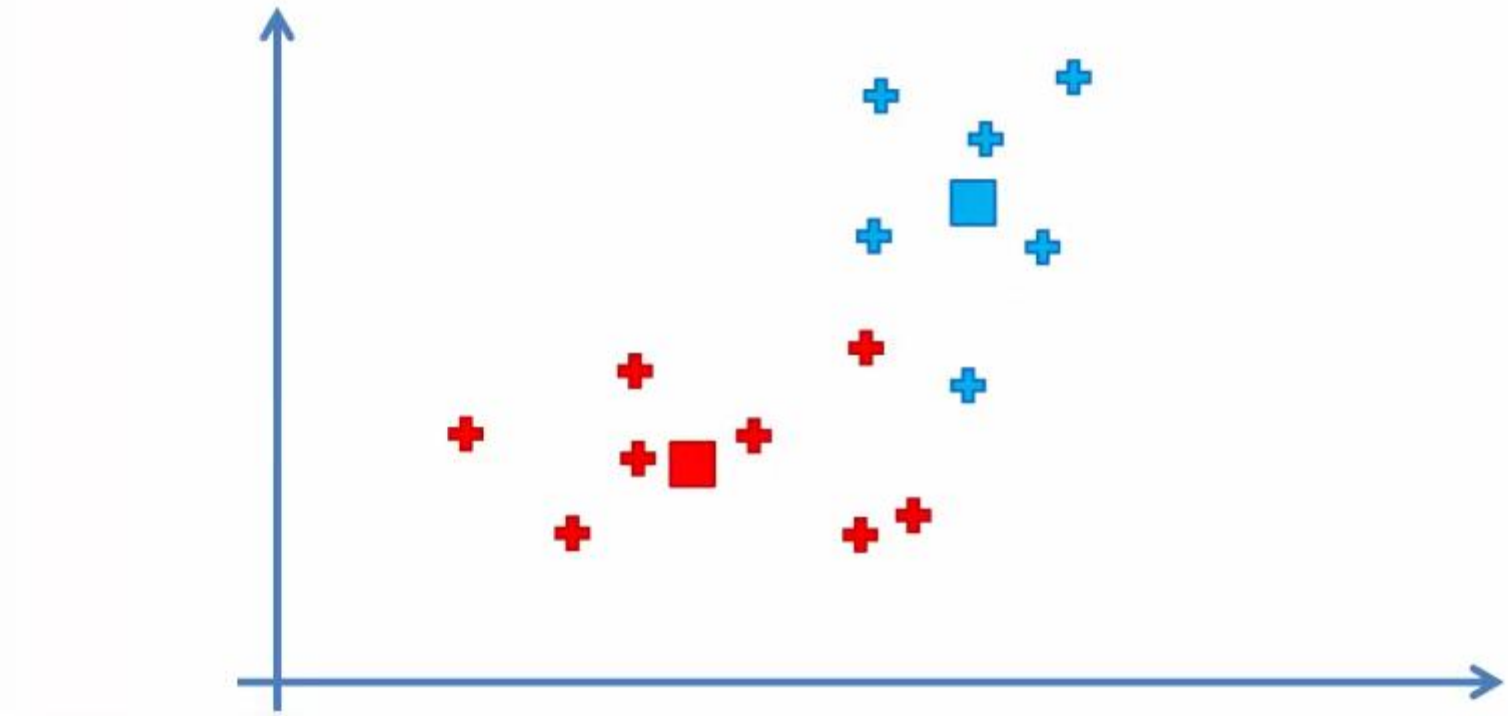
STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.



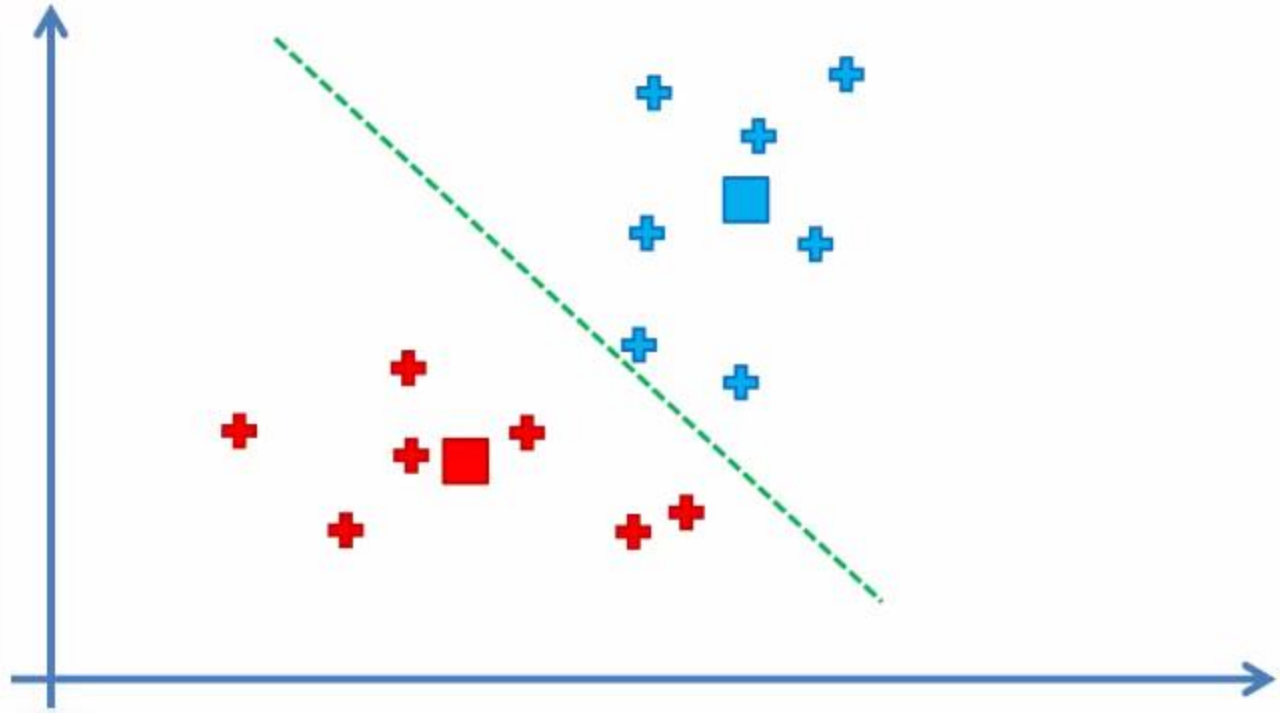
STEP 4: Compute and place the new centroid of each cluster

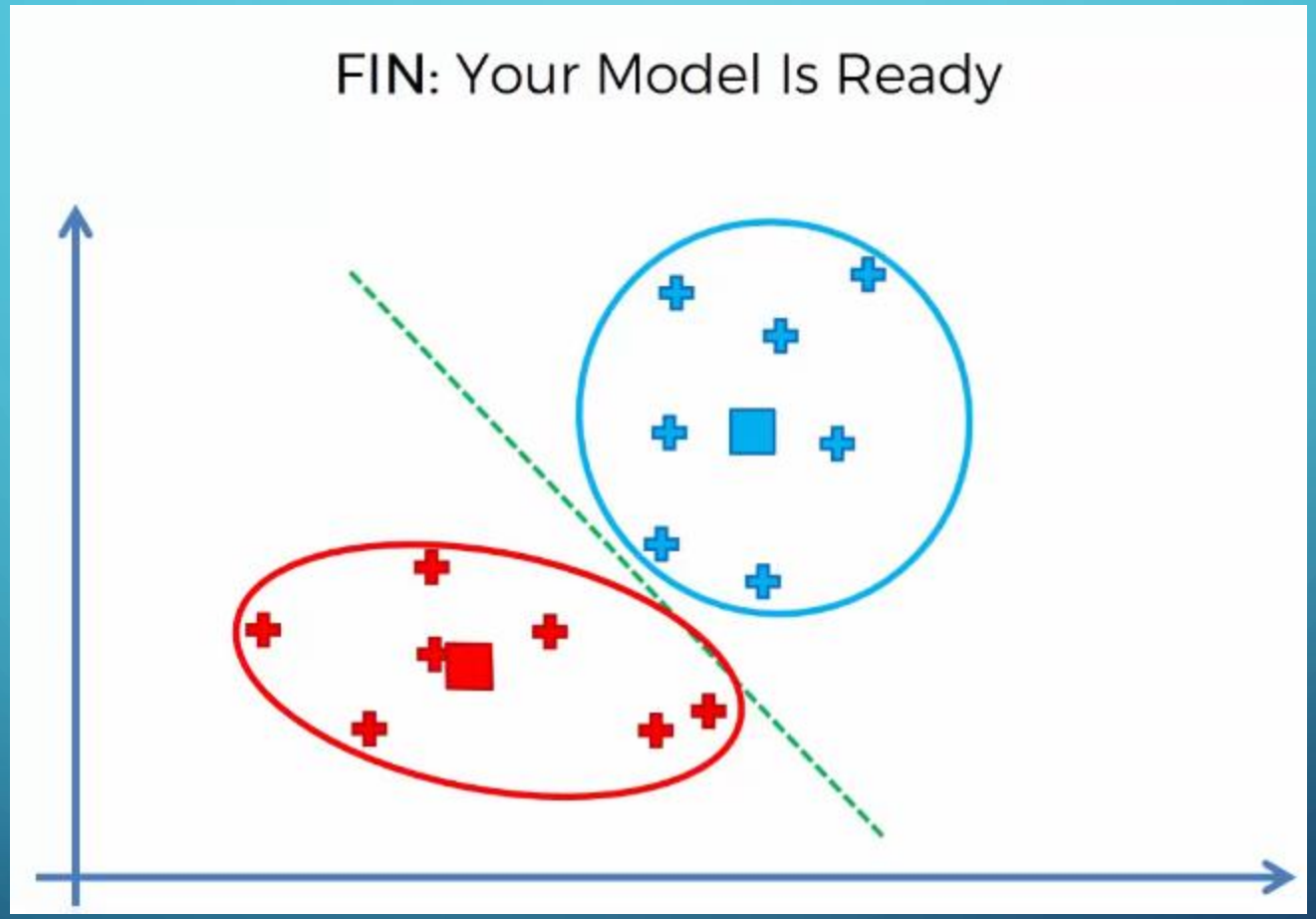


STEP 4: Compute and place the new centroid of each cluster

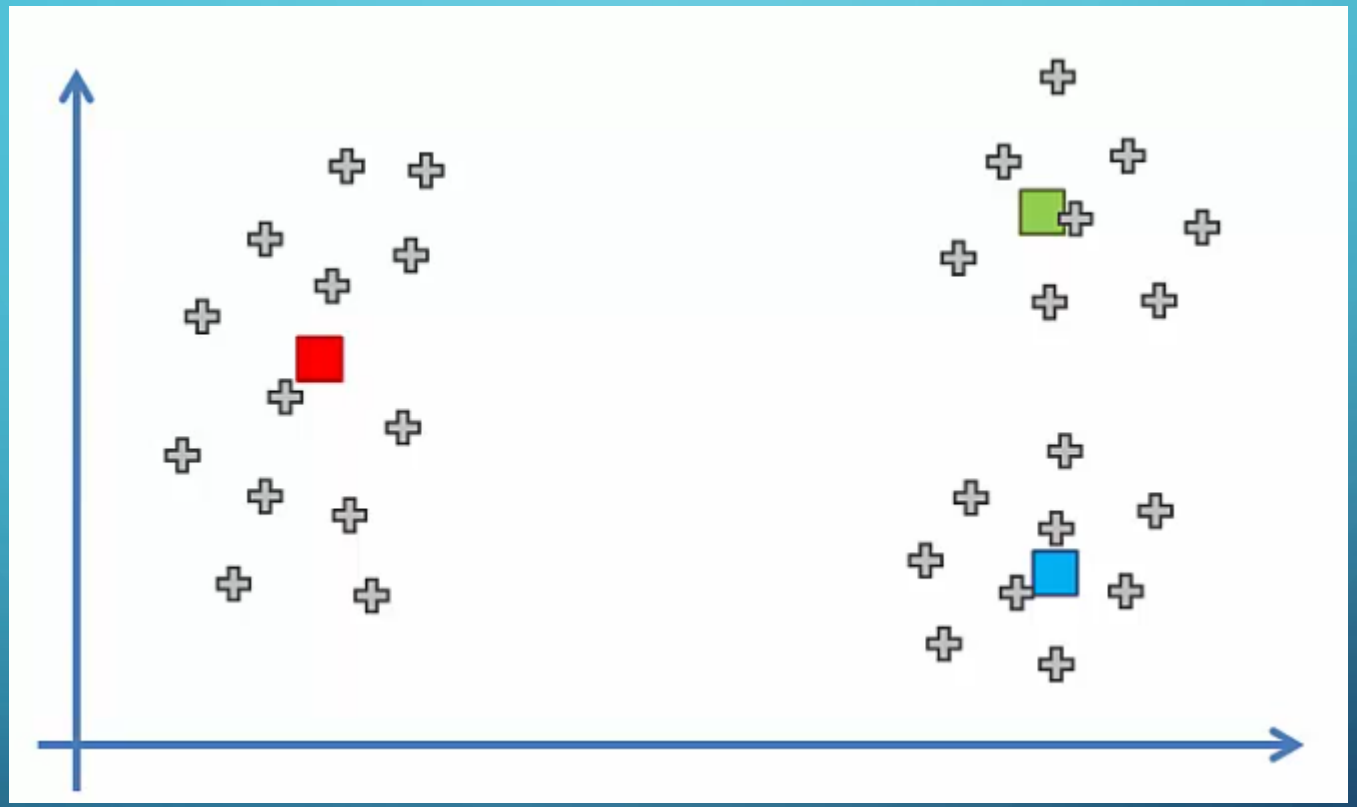


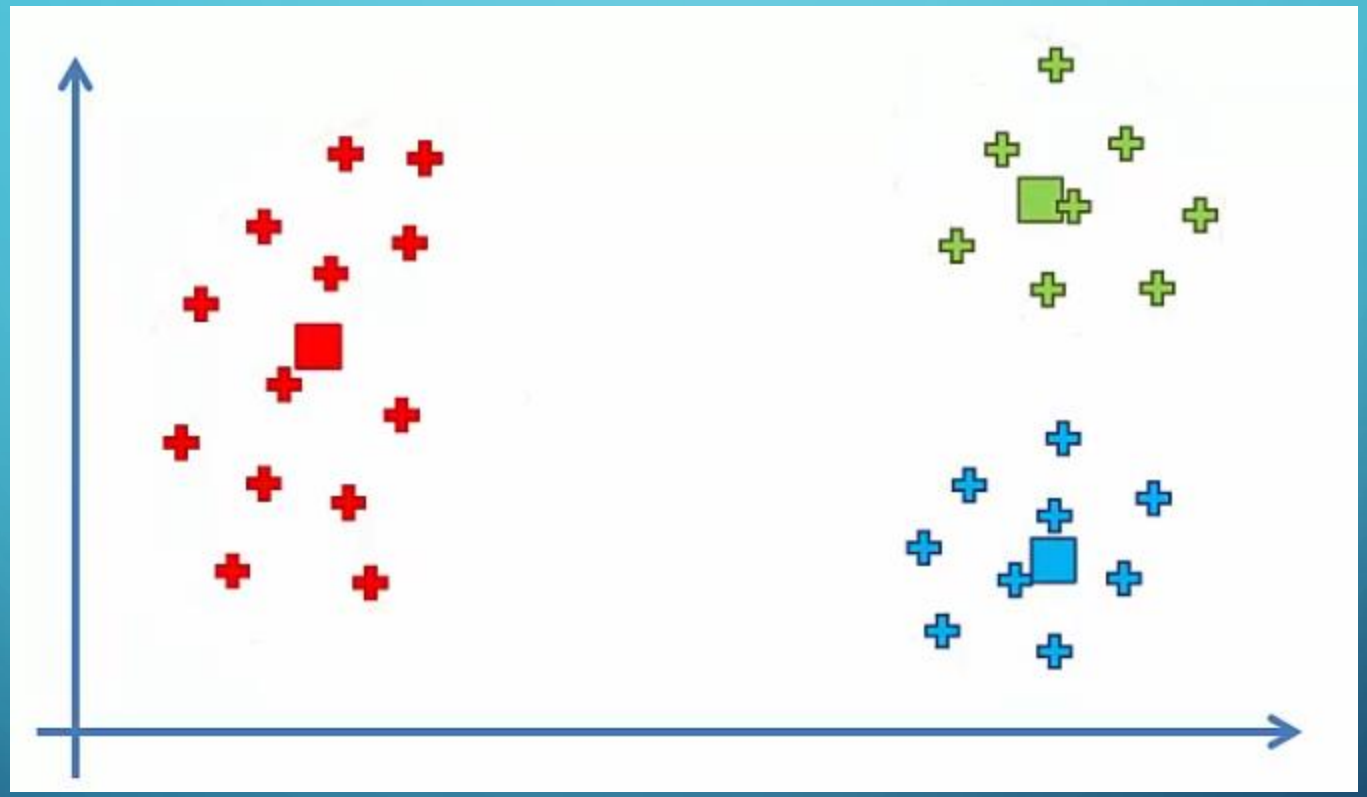
STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.

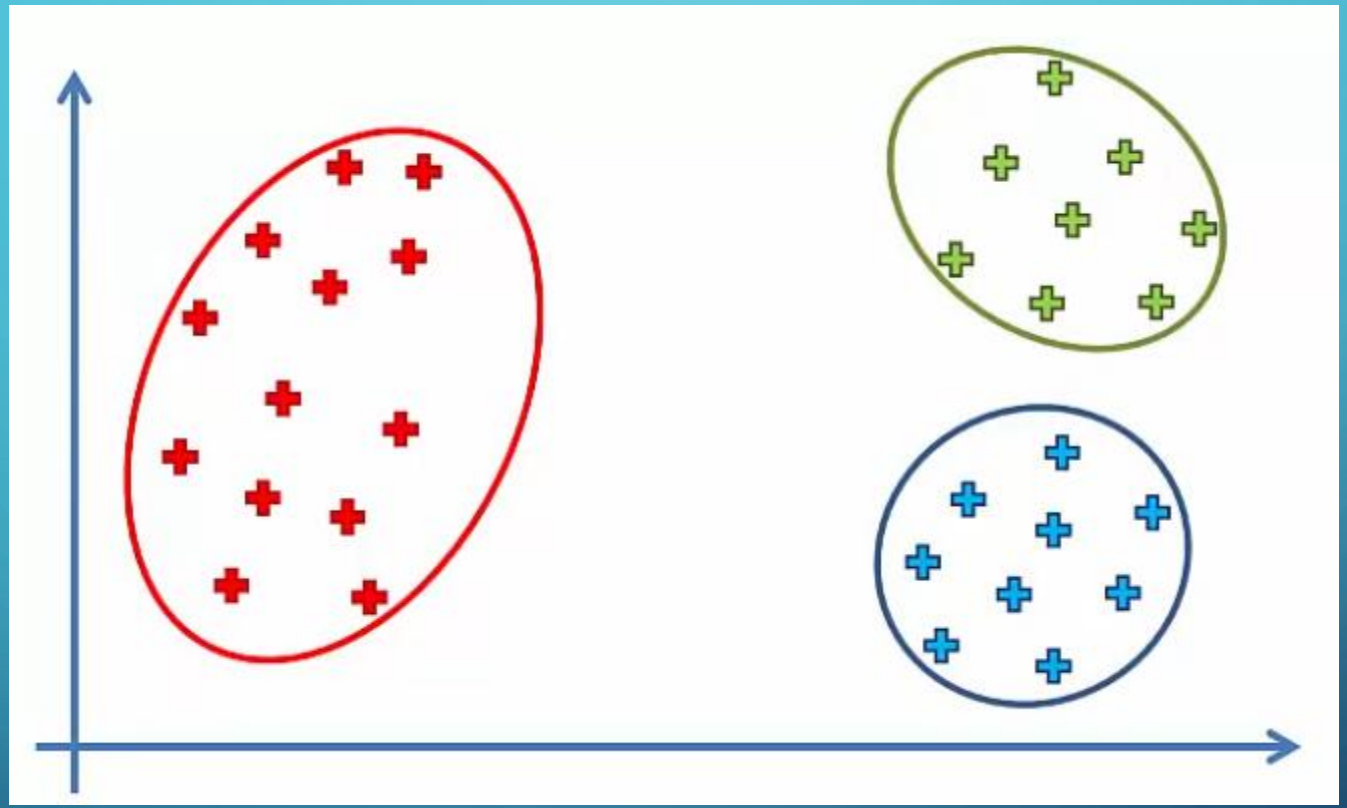



FIN: Your Model Is Ready

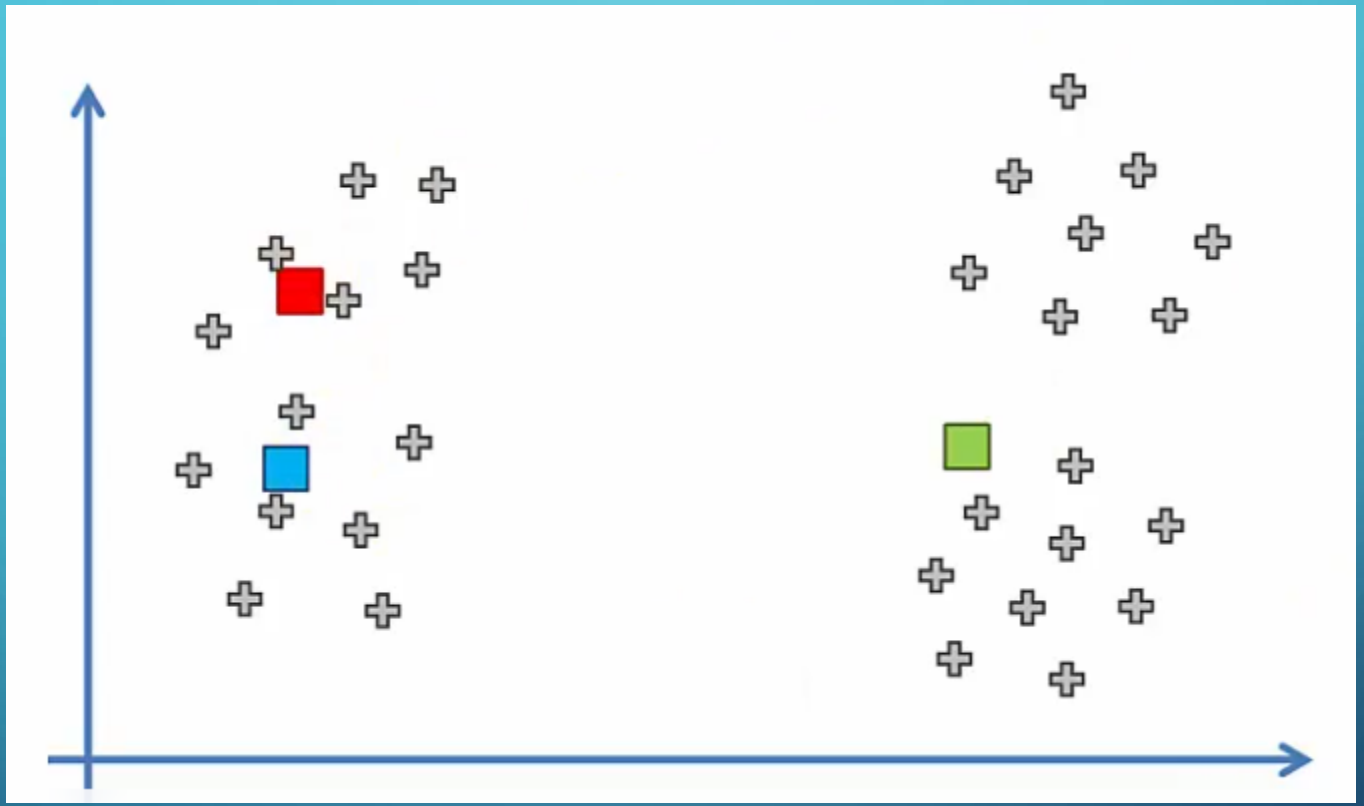
RANDOM INITIALIZATION TRAP

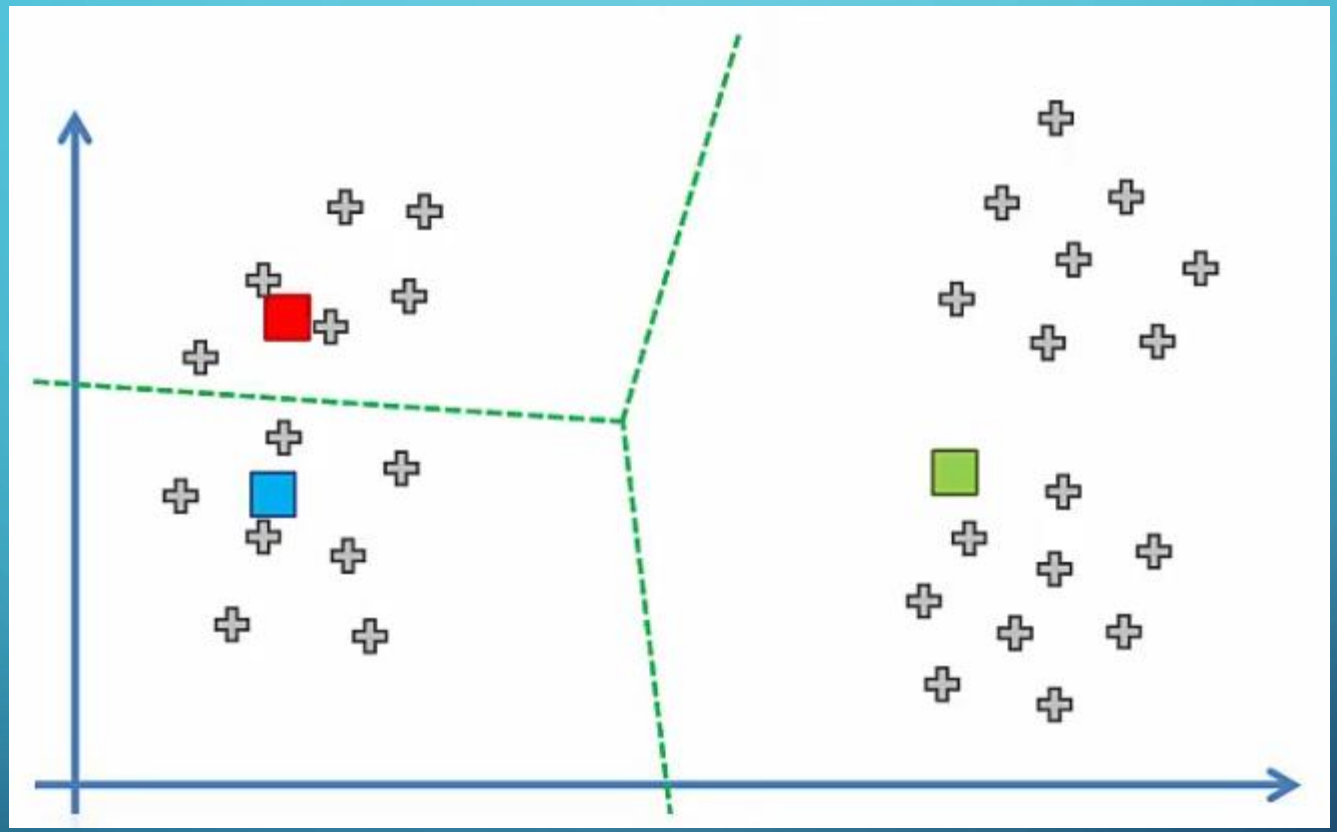


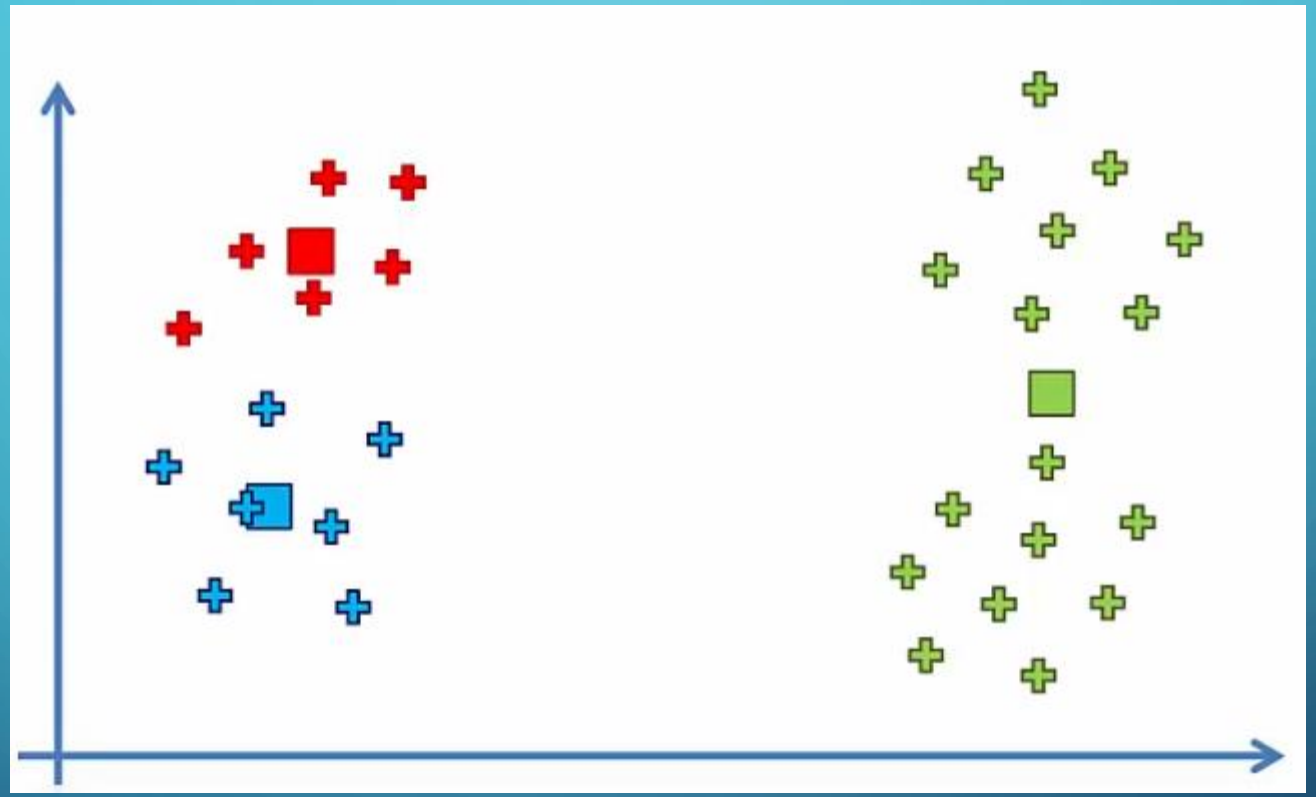


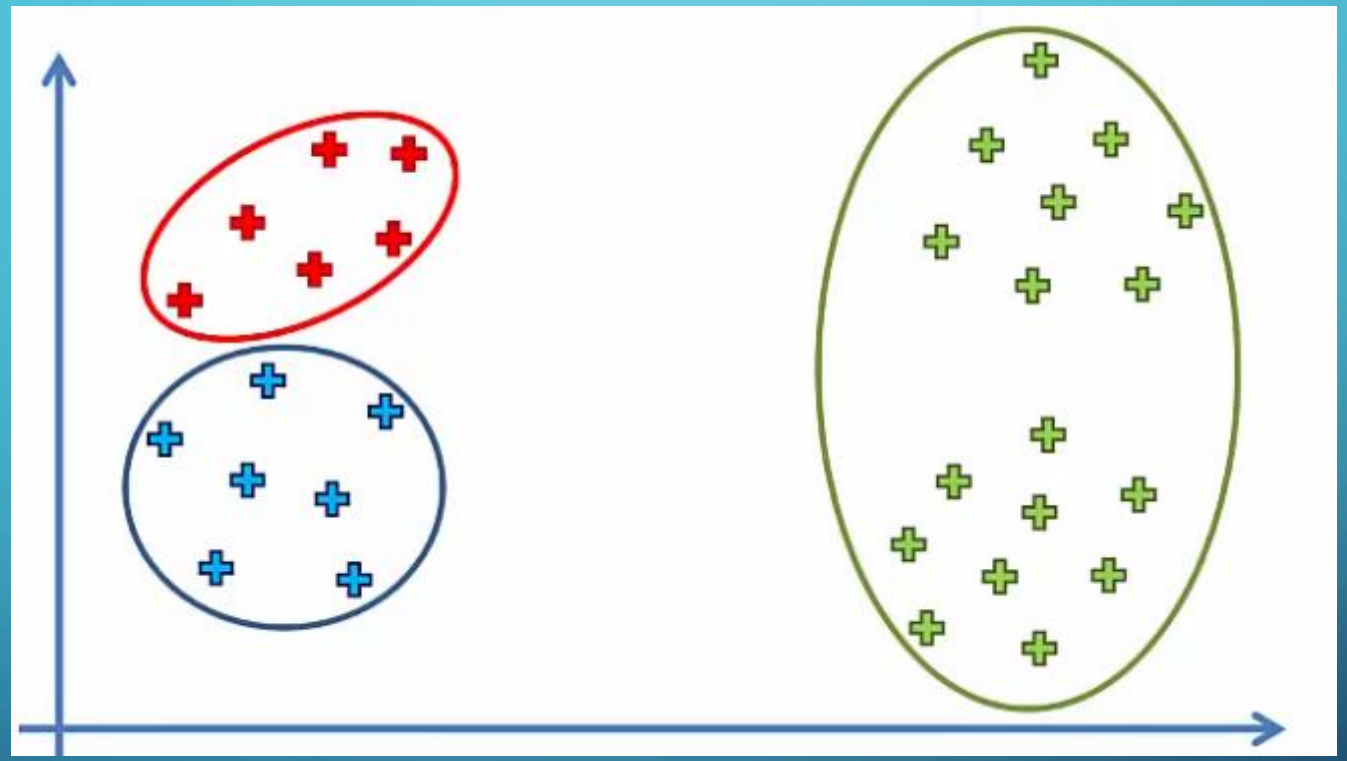


BUT…

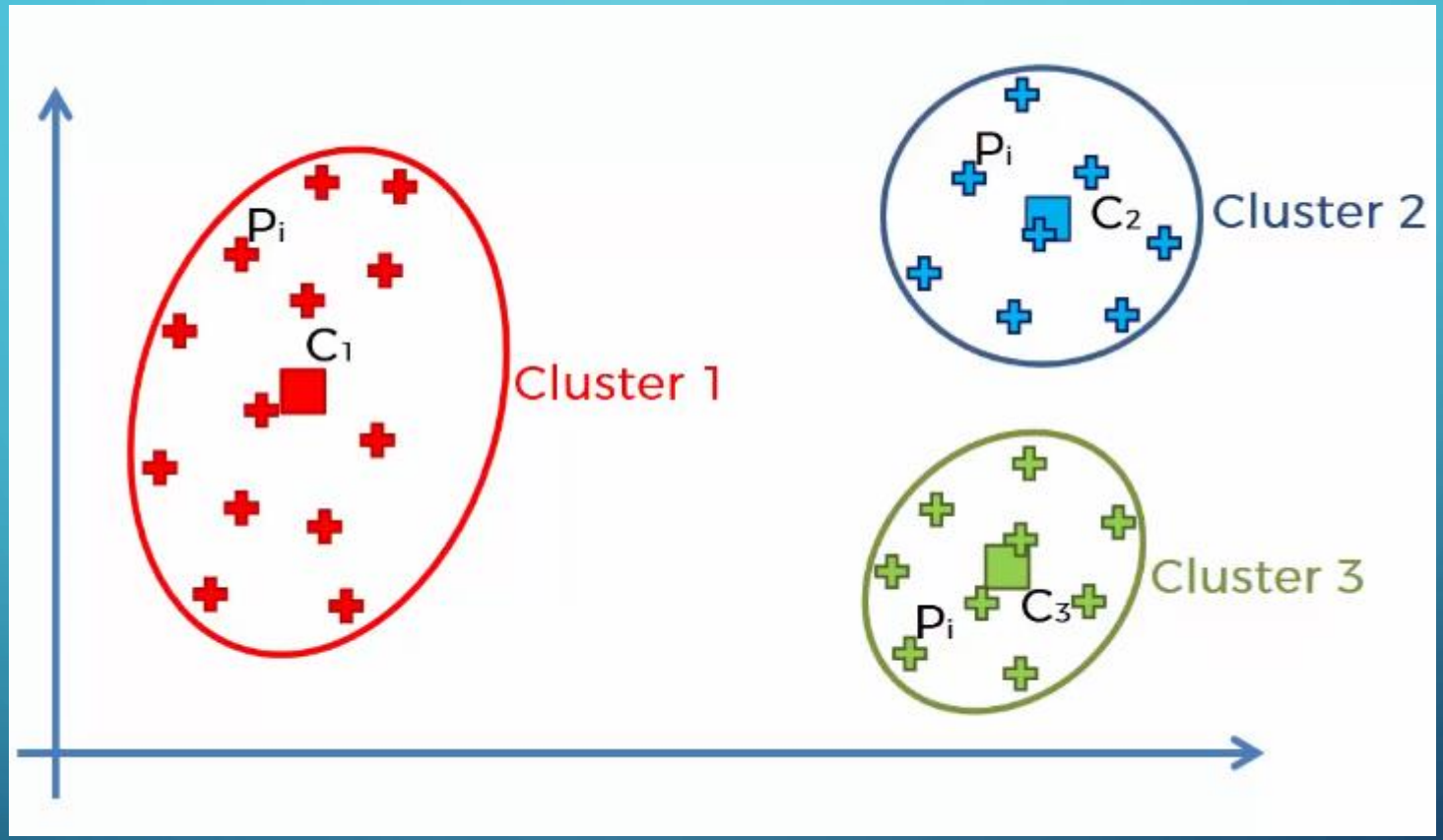






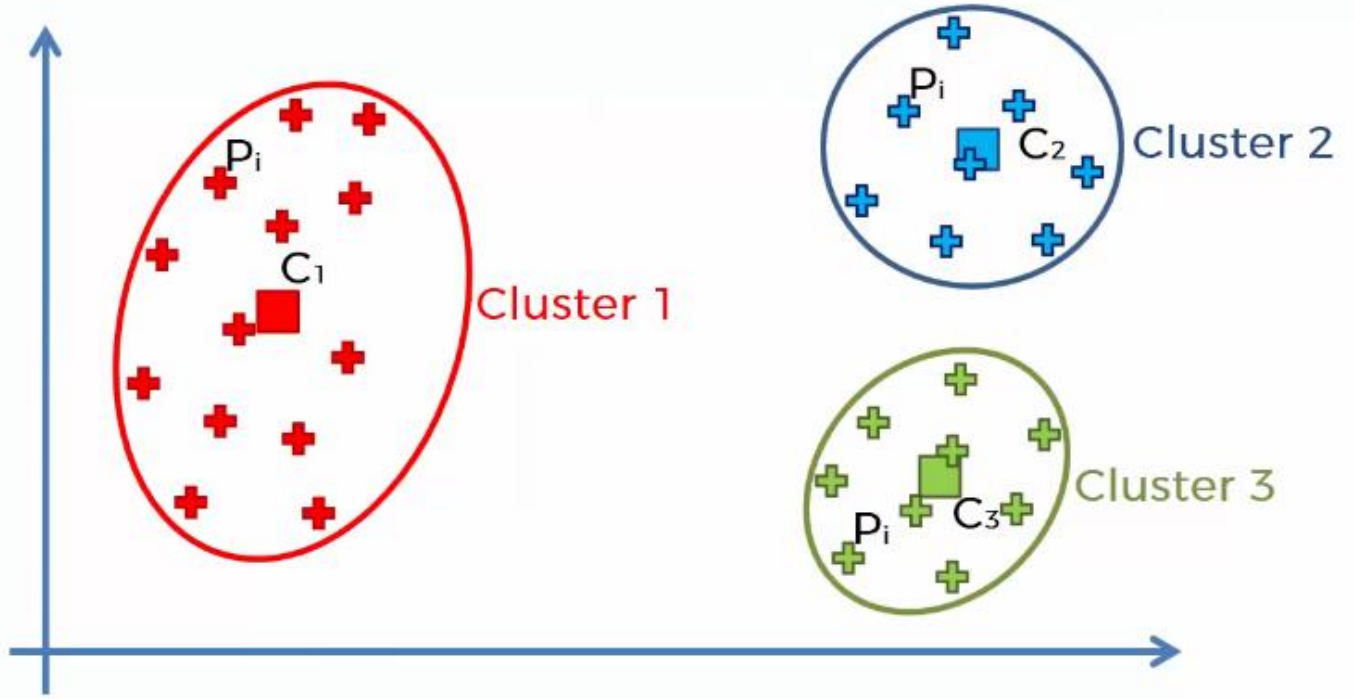


K-MEANS++


Pi
C1
Cluster 1
Pi
C2
Cluster 2
Pi
C3
Cluster 3

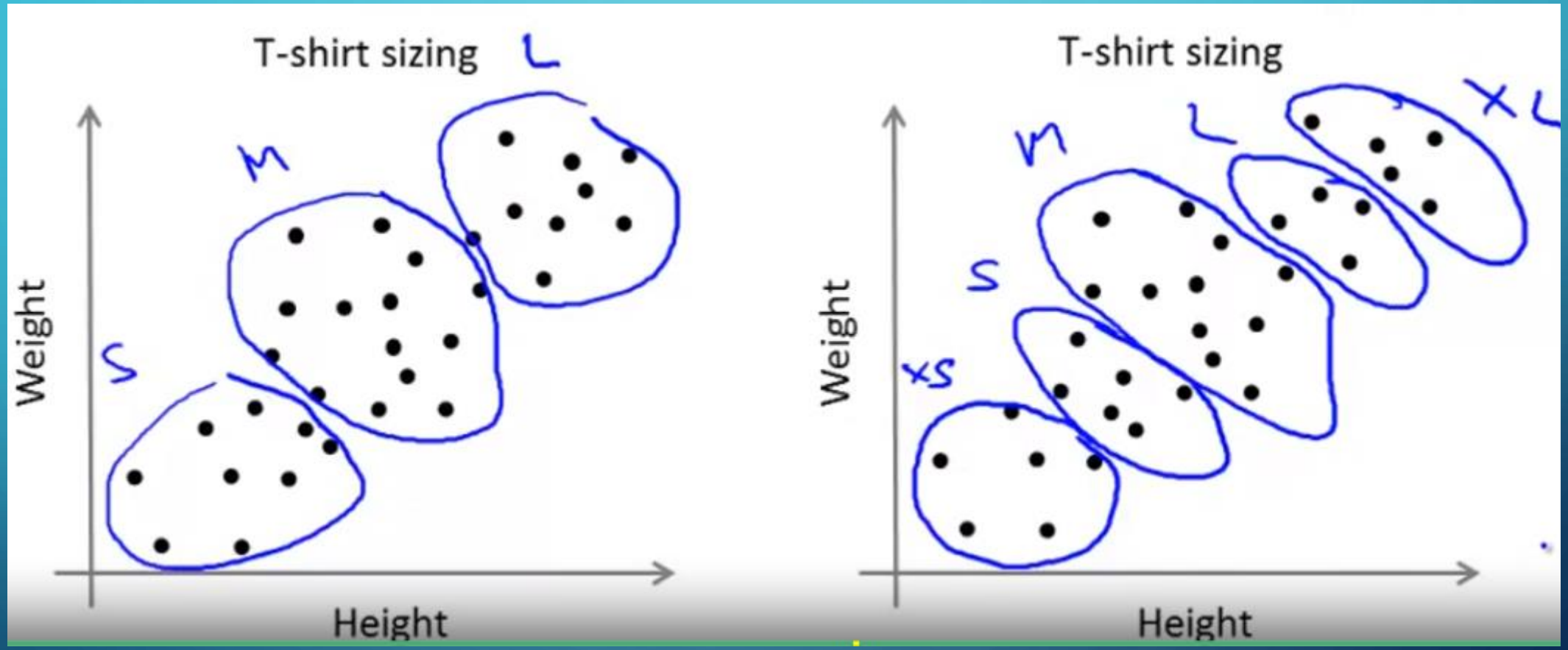
CHOOSING THE RIGHT CLUSTER

$$\text{WCSS} = \sum_{P_i \text{ in Cluster 1}} \text{distance}(P_i, C_1)^2 + \sum_{P_i \text{ in Cluster 2}} \text{distance}(P_i, C_2)^2 + \sum_{P_i \text{ in Cluster 3}} \text{distance}(P_i, C_3)^2$$



$$\text{WCSS} = \sum_{P_i \text{ in Cluster 1}} \text{distance}(P_i, C_1)^2 + \sum_{P_i \text{ in Cluster 2}} \text{distance}(P_i, C_2)^2 + \sum_{P_i \text{ in Cluster 3}} \text{distance}(P_i, C_3)^2$$

HOW TO CHOOSE NUMBER OF CLUSTERS


T-shirt sizing
L
M
S
Weight
Height
T-shirt sizing
XL
L
M
S
XS
Weight
Height

THANK YOU

AGENDA FOR TOMORROW:

- Doubt session
- Practical sessions
- Using Github as repository