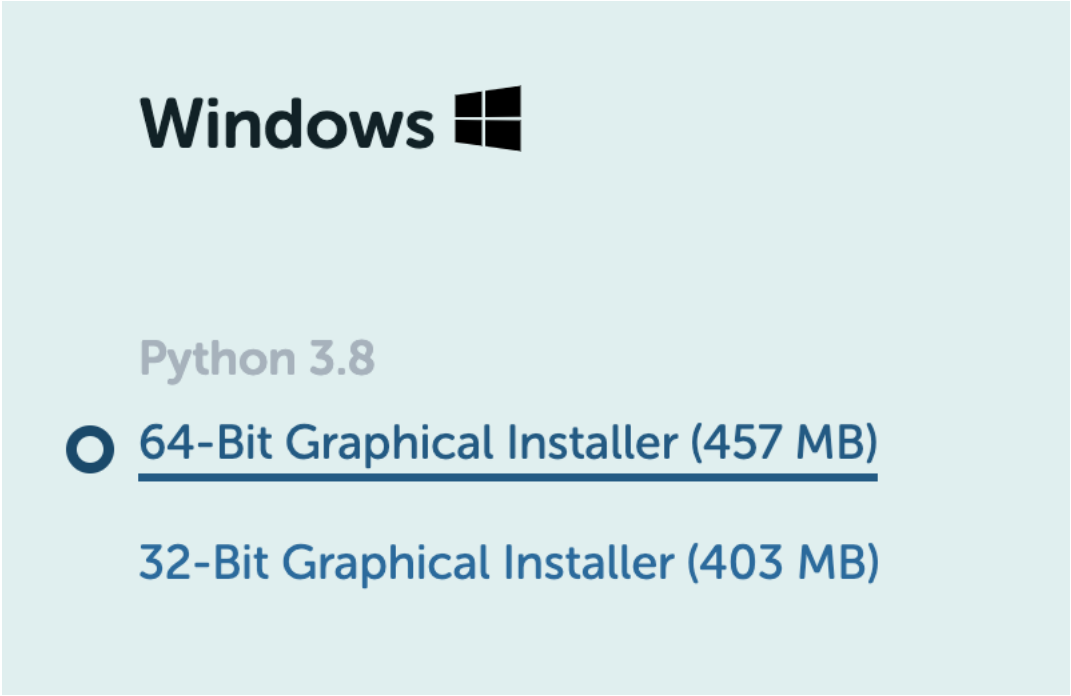
PRIME INTUIT
Finishing School

Getting started with Machine Learning

PRIME INTUIT
Finishing School

Getting started with Anaconda

https://www.anaconda.com/products/individual



PRIME INTUIT
Finishing School

Machine Learning

- ✓ What is Machine Learning
  - ✓ How does it contrast with other means of problem solving
- ✓ When to use Machine Learning
- ✓ How to use Machine Learning (Classification Problems)

PRIME INTUIT
Finishing School

Machine Learning

Working with Real life problem – Spam Detection

- Classify emails as Spam or Ham
- Figure this out before the email is sent to the inbox

Machine Learning

Spam detection – Rule based approach

- Define set of rules to identify spam
- Rules can be intuitive or logical
- Requires domain knowledge
- Needs access to historical data

Machine Learning

Spam detection – Rule based approach (Technique)

- Blacklist
  - Emails from specific IP address
  - Specific words and Pharses
- Whitelist
  - Emails form contacts of contacts of contacts
  - Certain domains

Machine Learning

Spam detection – Rule based approach

From: Spammer@donottrust.com
Subject: You won a lottery !

Nirionline,

Congratulations! You won a 10 million lottery, we need the following details to credit the same to your account.

Machine Learning

Spam detection – Rule based approach

From: Spammer@donottrust.net
Subject: look who won a lucky draw !

Niranjan,

You have won a lucky draw, we need the following details to credit the same to your account.

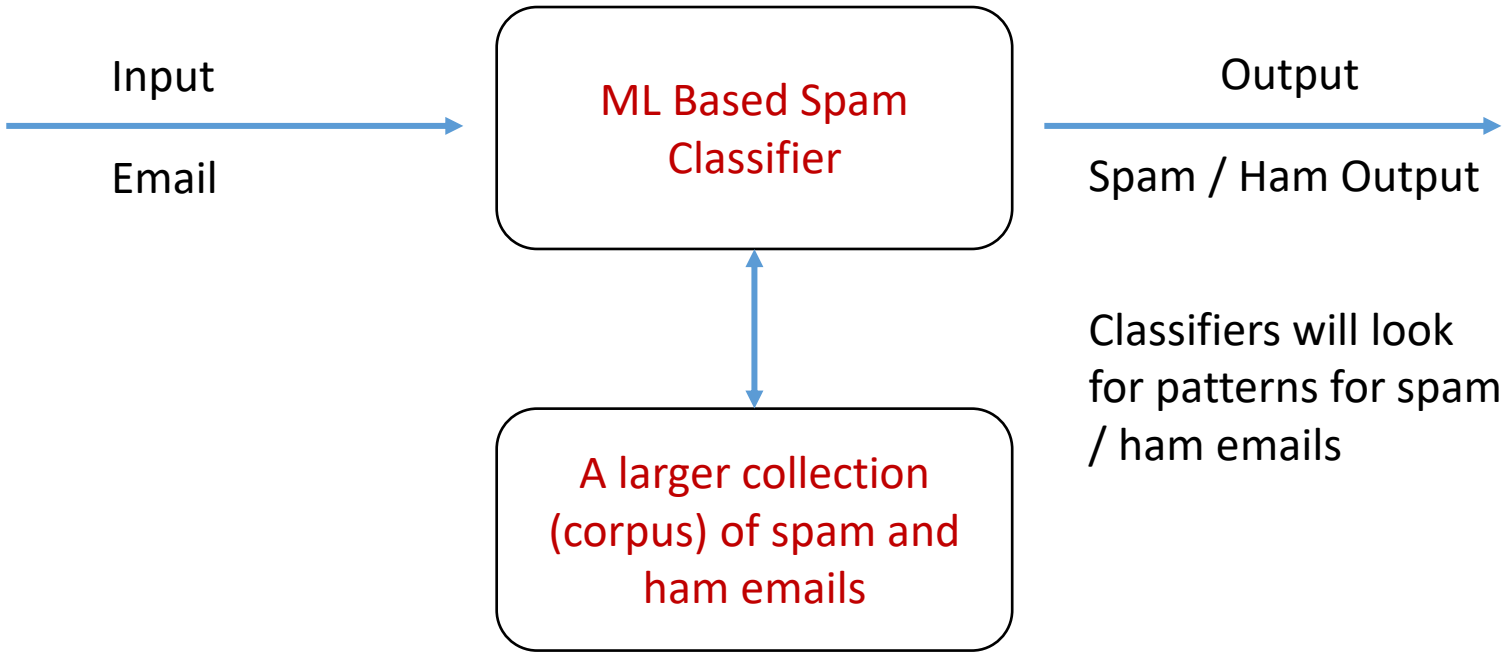
Machine Learning

Spam detection – Rule based approach (limitations)

- Rules get outdated
- Rules are static – Spammers are dynamic
- Hard to maintain many rules
- Rethinking rules is time consuming
- Easier for spammer to get around your rules than the time taken for you to build the rules.

Machine Learning

Spam detection – ML based approach



Machine Learning

Spam detection – ML based approach (Advantages)

- Algorithm can vary approach based on data
- Corpus can be updated based on user feedback
- Can detect patterns not visible to humans
- May be less complex than rule based - approach

PRIME INTUIT
Finishing School

Machine Learning

What is machine learning ?

It’s the ability of a program to read, understand and learn from the data and there by draw inferences about the data.

Machine Learning

When to use machine learning

- The alternative is a complex set of hard to maintain rules
- Dynamic environments with changing data
- Rule based solutions can not solve the problem
- To get insights on to your data

PRIME INTUIT
Finishing School

Machine Learning

ML – based Vs Rule based

ML Based	Rule Based
Dynamic	Static
Experts optional	Experts required
Need corpus	Corpus required
Training required	No training

Machine Learning

Types of Machine Learning

PRIME INTUIT
Finishing School

Machine Learning

Supervised Learning

- Learning under the supervision of a coach / Teacher
- ML algorithm given data along with labels / responses
- Learns from looking at questions and answers

PRIME INTUIT
Finishing School

Machine Learning

Supervised Learning

- Classification
  - Spam or Ham
  - Mammal or fish or reptiles or birds or Amphibian
- Regression (predict numeric values)
  - Income of shopper

PRIME INTUIT
Finishing School

Machine Learning

Supervised Learning

Cause

Effect

X Cause Y

Machine Learning

X Variables

- Attributes which an ML algorithm focuses on are called features
- Each data point is a list (or vector) of such features
- Input to an ML algorithm is a feature vector
- Feature vectors are called x variables
- Also called independent variables or predictors

Machine Learning

Y Variables

- Attributes which an ML algorithm tries to predict are called labels
- Labels can be:
  - Categorical (classification)
  - Continuous (regression)
- Labels are referred to as y variables
- Also called dependent variable

PRIME INTUIT
Finishing School

Machine Learning

Data Set

Target Variables

Attributes

Features

Sender	Subject	IP	Body	Result
Xyz@bank.com	Loan	77.44.5.89	Dear customer your loan application….	Spam
123@builder.com	quotation	101.22.76.81	Dear Niranjan Your quotation……	Ham
lmn@school.com	reciept	301.99.23.45	Hi sir your Sons school fee……	Ham

Feature Vector

Labels

PRIME INTUIT
Finishing School

Machine Learning

Supervised Learning
(Training Stage)

$$y = f(x)$$

Most machine learning algorithms aim to learn the function

f which links the function x to label y

PRIME INTUIT
Finishing School

Machine Learning

Supervised Learning models

- Linear Regression
- Logistic Regression

PRIME INTUIT
Finishing School

Machine Learning

Supervised Learning models

- Linear Regression
- Logistic Regression

PRIME INTUIT
Finishing School

Machine Learning

Linear Regression

$$Y = m\,(x) + C$$

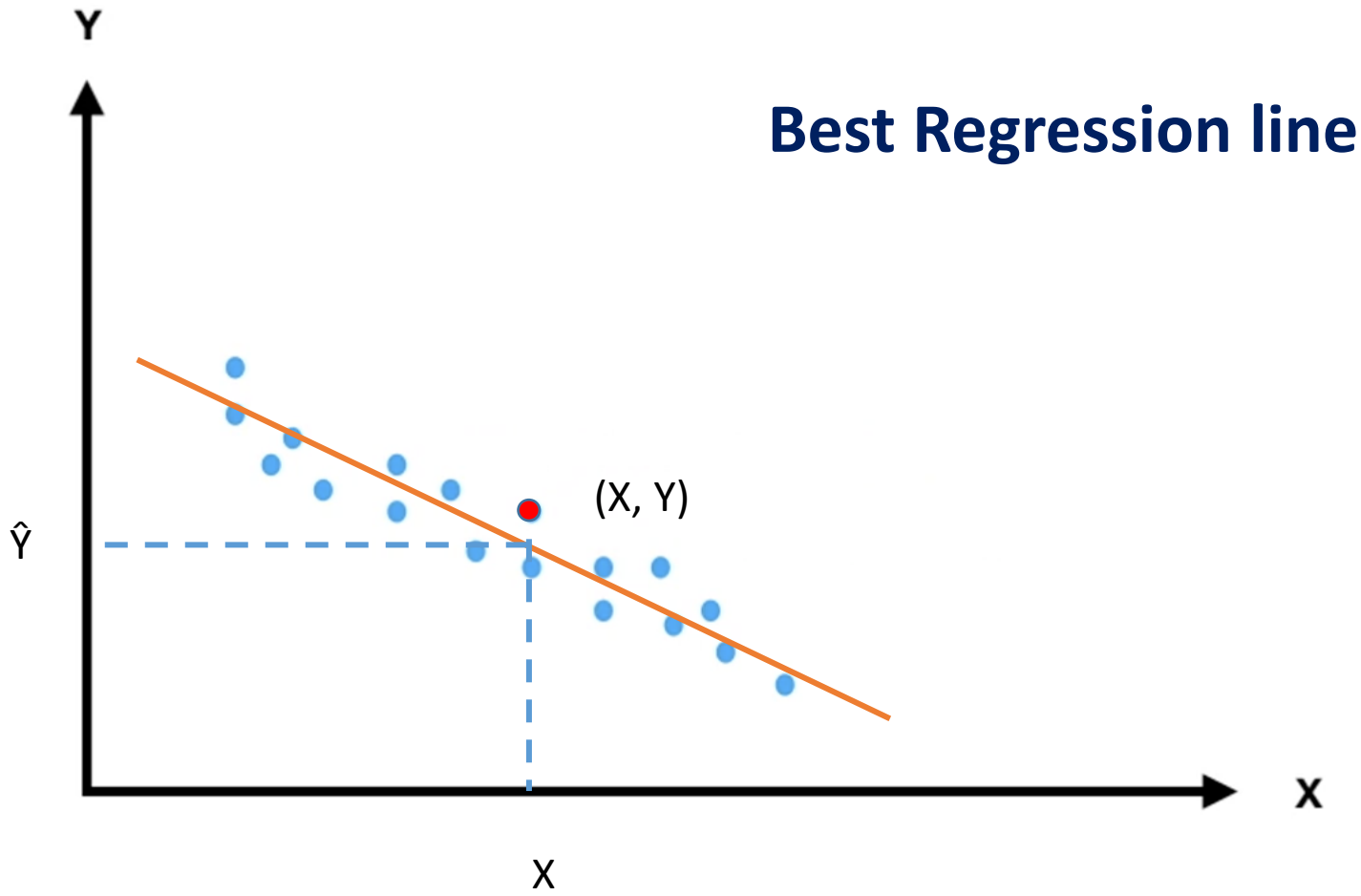
$$f(y) = m\,(x) + C$$

Here Linear regression specifies upfront that the relationship between x & y represented by f is linear

PRIME INTUIT
Finishing School

Machine Learning

Linear Regression



Best Regression line

PRIME INTUIT
Finishing School

Machine Learning

Best regression line

$$\text{Residual} = e = Y - \hat{Y}$$

Data = (x1, y1), (x2, y2), (x3, y3) ---------- (xn, yn)

Residual = (Y1 – $\hat{Y}$1), (Y2 – $\hat{Y}$2), (Y3 – $\hat{Y}$3) ---------- (Yn – $\hat{Y}$n)

To get the best regression line our aim should be to get the minimum error for training data

Eg: least Square method

Minise: (Y1 – $\hat{Y}$1)2 + (Y2 – $\hat{Y}$2)2 + (Y3 – $\hat{Y}$3)2 ---------- (Yn – $\hat{Y}$n)2

PRIME INTUIT
Finishing School

Machine Learning

Cavoites of regression line

- Can be used only when the errors are normally distributed
- Can also be used with multiple independent variables

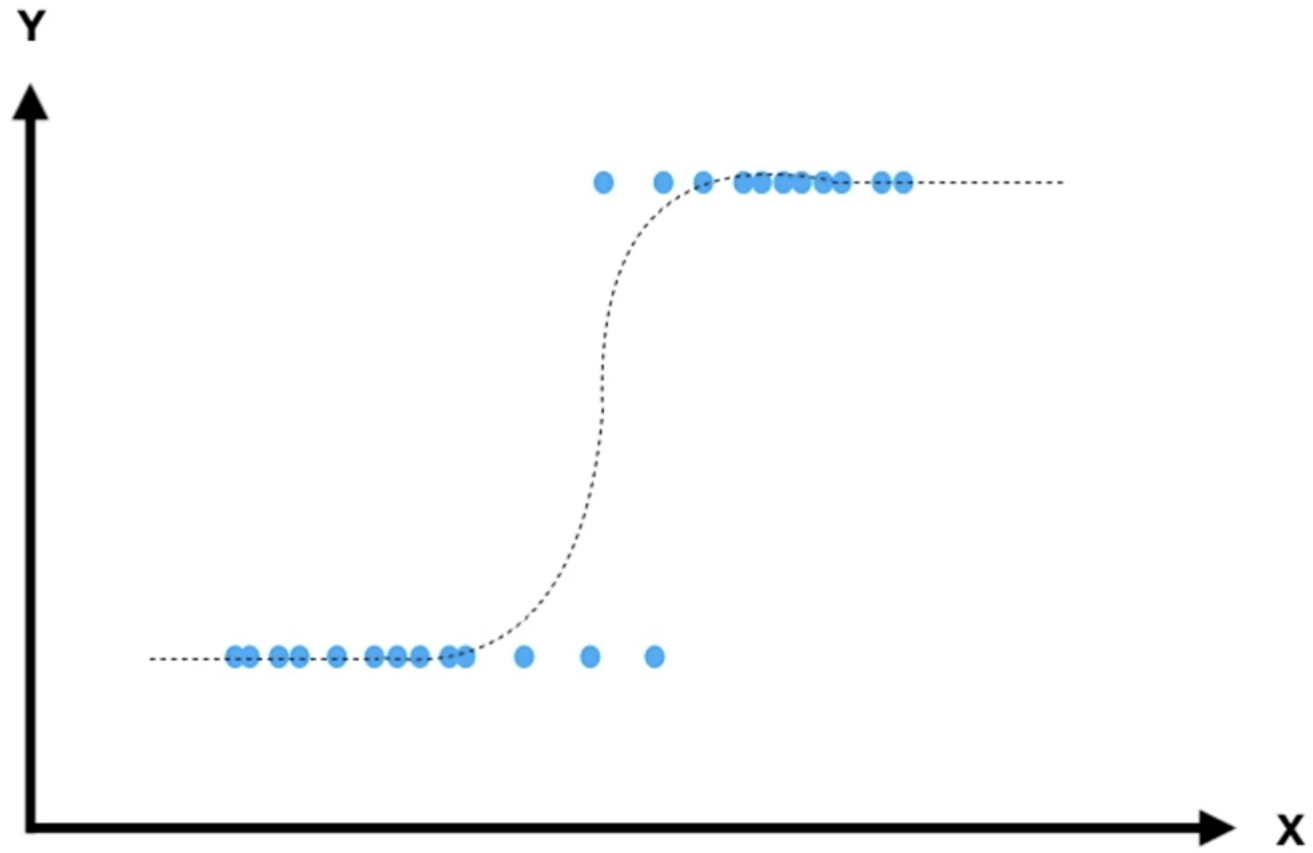
Logistic Regression

- Used when the dependent variable is categorical (y)
- Mutiple Independent variables can be continuous or categorical (x)
- Predict probability of each outcome – assign result to category with highest probability

PRIME INTUIT
Finishing School

Machine Learning

Logistic Regression



Machine Learning

Logistic Regression

Objective function: Minimize cross entropy

Cross-entropy is a measure from the field of information theory, building upon entropy and generally **calculating the difference between two probability distributions.** It is closely related to but is different from KL divergence that calculates the relative entropy between two probability distributions, whereas cross-entropy can be thought to calculate the total entropy between the distributions.

Cross-entropy is also related to and often confused with logistic loss, called log loss. Although the two measures are derived from a different source, when used as loss functions for classification models, both measures calculate the same quantity and can be used interchangeably.

PRIME INTUIT
Finishing School

Machine Learning

Unsupervised Learning

- There is only input data X – no output data
- Model the underlaying structure to learn more about the data
- Algorithms self discover patterns of structure in the data

PRIME INTUIT
Finishing School

Machine Learning

Unsupervised Learning - algorithms

- Autoencoding – Identify latent factors that drive data (eg: PCA)
- Clustering – Identify patterns in data items

Looking within

- Be emotionally self sufficient
- Learn what matters (to you)
- Identify others who share them and those who don’t
- Eliminate what does not matter
- Train yourself to navigate the outside world

PRIME INTUIT
Finishing School

Machine Learning

Why Look with in

In Life

- Be emotionally self sufficient
- Learn what matters (to you)
- Identify others who share them
- and those who don’t
- Eliminate what does not matter
- Train yourself to navigate the outside world

In ML

- Make unlabeled data self sufficient
- Latent factor analysis (fields relevant for a cause)
- Clustering
- Anomaly detection
- Quantisation

PRIME INTUIT
Finishing School

Machine Learning

Why Look with in

ML Technique	In Life
• Make unlabeled data self sufficient	• Identify photo of specific individual
• Latent factor analysis (fields relevant for a cause)	• Find common drivers for 100 Stocks
• Clustering	• Find relevant document in a corpus
• Anomaly detection	• Flag fraudulent credit card transactions
• Quantisation	• Compress 24 bit true color to 8 bit

PRIME INTUIT
Finishing School

Machine Learning

Why Look with in

What

- Make unlabeled data self sufficient
- Latent factor analysis (fields relevant for a cause)
- Clustering
- Anomaly detection
- Quantisation

How

- Autoencoding
- Autoencoding

- Clustering
- Autoencoding
- Clustering

Machine Learning

Support Vector Machines

PRIME INTUIT
Finishing School

Machine Learning

What is an SVM

- SVMs are used to build binary classifiers
- Make Classification decision on basis of a “Linear function” of point’s co-ordinates
- Does not require prior knowledge of probability distribution of the points
- Involves an explicit training stage

PRIME INTUIT
Finishing School

Machine Learning

What is an SVM

- SVMs are supervised machine learning approach used to build linear, non probabilistic and binary classifiers

Machine Learning

What is an SVM

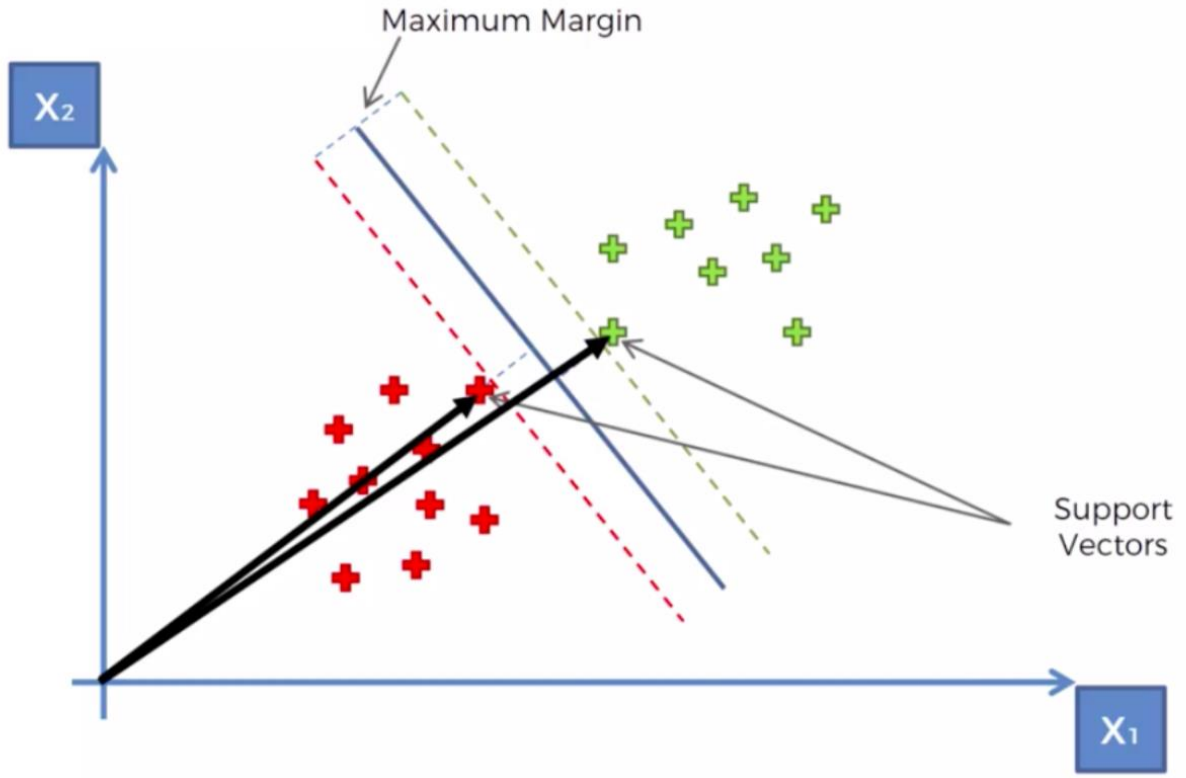
The linear SVM classifier works by drawing a straight line between two classes. All the data points that fall on one side of the line will be labeled as one class and all the points that fall on the other side will be labeled as the second. Sounds simple enough, but there's an infinite amount of lines to choose from. How do we know which line will do the best job of classifying the data? This is where the LSVM algorithm comes in to play.

PRIME INTUIT
Finishing School

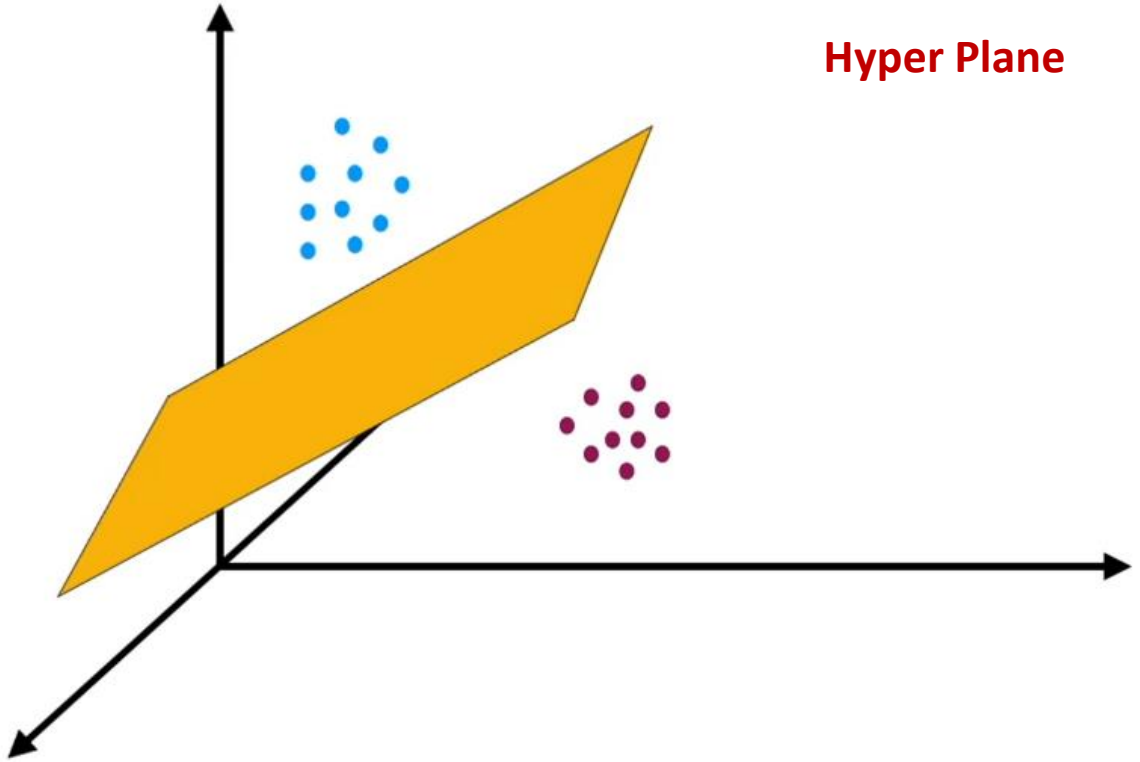
Machine Learning

What is an SVM

The LSVM algorithm will select a line that not only separates the two classes but stays as far away from the closest samples as possible. In fact, the “support vector” in “support vector machine” refers to two position vectors drawn from the origin to the points which dictate the decision boundary.



How SVMs work



In an n dimensional data/ space SVM finds an n-1 hyperplane to separate points into two categories

Machine Learning

Hyper plane

In a vector space of n dimensions, a hyperplane is a geometrical shape with (n-1) dimensions and zero thickness in one dimension.

In our example $Ax + By + Cz = D$

Points on one side of the hyperplane :

$Ax + By + Cz > D$

Points on other side of hyperplane:

$Ax + By + Cz < D$

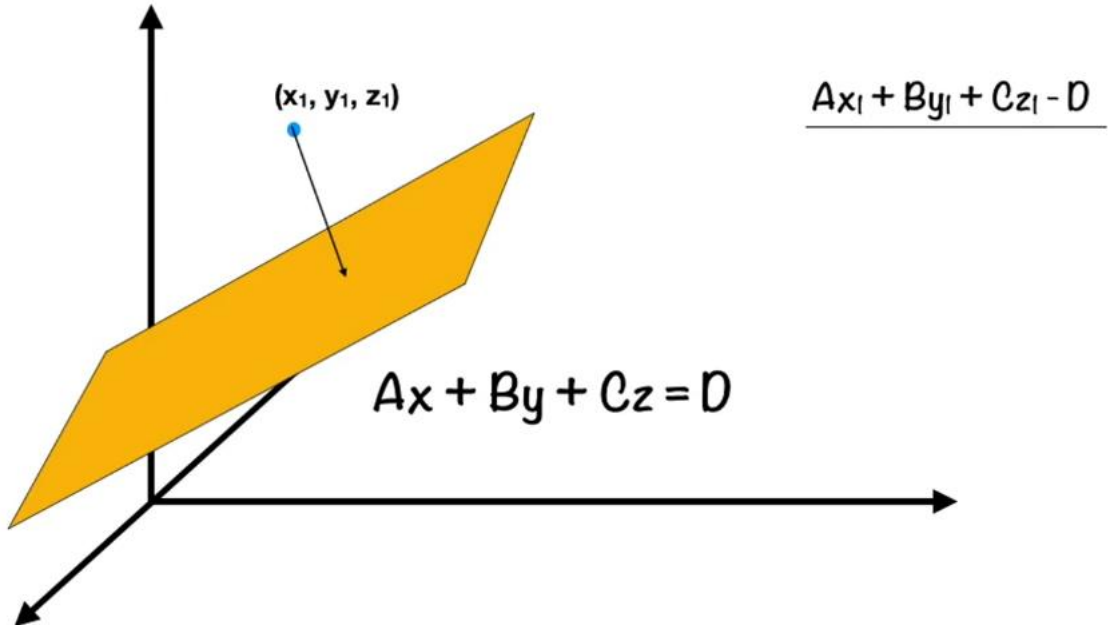
SVM tries to find such a linear equation

PRIME INTUIT
Finishing School

Machine Learning

Hyper plane

In a vector space of n dimensions, a hyperplane is a geometrical shape with (n-1) dimensions and zero thickness in one dimension.



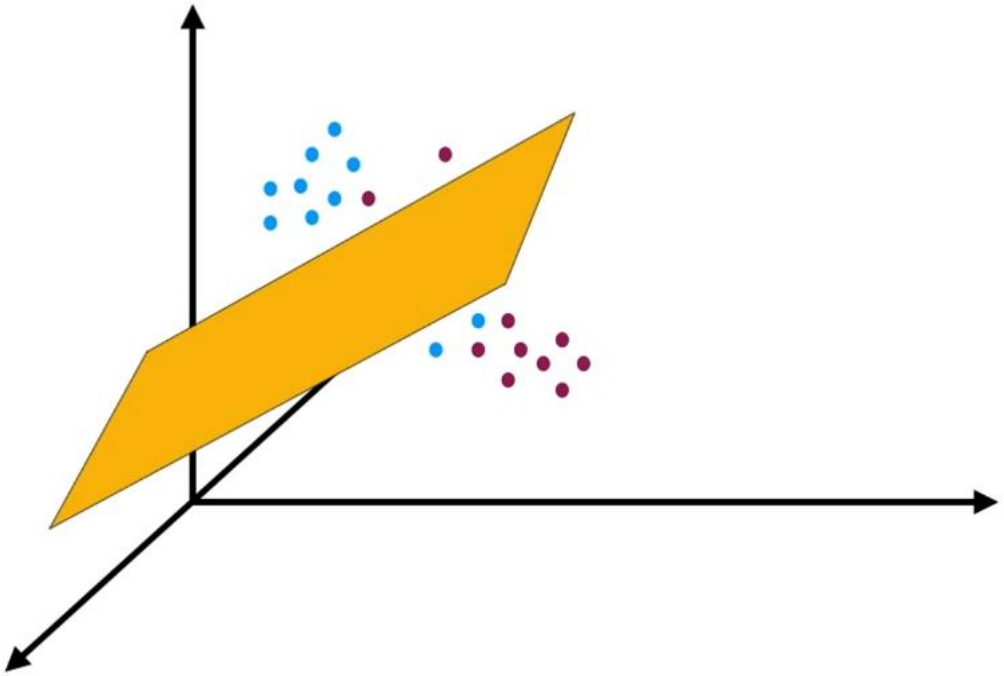
The best hyper plane is the one which maximizes the sum of distances of the nearest points on either side of the plane

PRIME INTUIT
Finishing School

Machine Learning

Hyper plane

What if the points on the hyperplane are not linearly separable



The soft margin method finds a hyper plane which performs as clean a separation of points as possible

PRIME INTUIT
Finishing School

Machine Learning

Non linear separation

SVM is a linear classifier

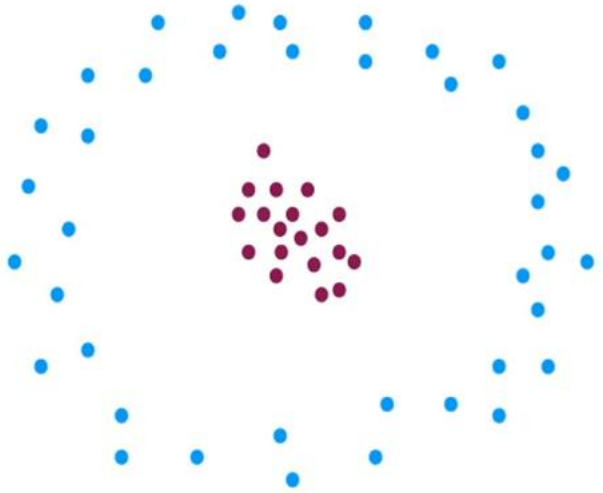
But can be used to perform non linear separation

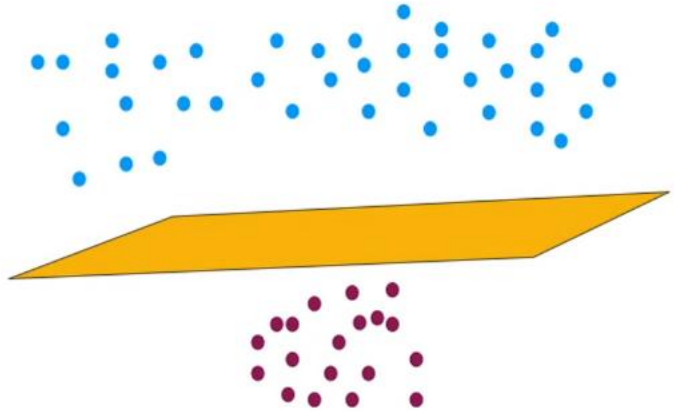
Achieved by using Kernel Trick

PRIME INTUIT
Finishing School

Machine Learning

Kernel Trick





Transformed / modified to be separated by a linear plane

Can be separated only by a circle (quadratic)

PRIME INTUIT
Finishing School

Machine Learning

Linear Classification does a dot product of vectors with many elements

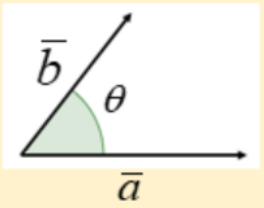
Algebraically, the dot product is **the sum of the products of the corresponding entries of the two sequences of numbers**. Geometrically, it is the product of the Euclidean magnitudes of the two vectors and the cosine of the angle between them. ... In modern geometry, Euclidean spaces are often defined by using vector spaces.

Dot Product

If $\bar{a} =< a_1, a_2, a_3 >$ and $\bar{b} =< b_1, b_2, b_3 >$

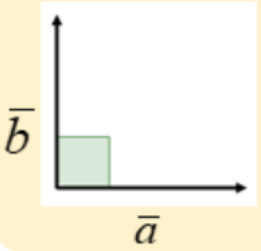
then the dot product is

$$\bar{a} \bullet \bar{b} = a_1 b_1 + a_2 b_2 + a_3 b_3$$



If θ is the angle between $\bar{a}$ and $\bar{b}$ then

$$\bar{a} \bullet \bar{b} = |\bar{a}||\bar{b}|\cos\theta$$



$\bar{a} \bullet \bar{b}$ are orthogonal (perpendicular) if and only if $\bar{a} \bullet \bar{b} = 0$

PRIME INTUIT
Finishing School

Machine Learning

Non linear separation

Kernel function operates in feature space which may have which may have many more dimensions than original feature space

Finds the maximum margin in hyper plane in modified feature space

It’s a linear function in the modified feature space

Kernel trick allows a way to solve problems where the data is not linearly separable by projecting such data into a higher dimensional space

PRIME INTUIT
Finishing School

Machine Learning

SVM – Use case

https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data

Machine Learning

SVM – Use case

```
import numpy as np
import pandas as pd
original_data = pd.read_csv("adult.csv",
                    names=['Age','Workclass', 'fnlwgt', 'Education', 'Education-Num',
'Marital Status',
                    'Occupation', 'Relationship', 'Race', 'Gender', 'Capital Gain',
'Capital Loss',
                    'Hours per week', 'Country', 'Target'], sep =r'\s*,\s*', engine
='python', na_values = "?")
original_data.head()
```

Machine Learning

SVM – Use case

```
import matplotlib.pyplot as plt

import math

%matplotlib inline

fig = plt.figure(figsize = (20,20))

cols = 3

rows = math.ceil(float(original_data.shape[1] / cols))
```

Machine Learning

SVM – Use case

```
for i, column in enumerate(['Age', 'Workclass', 'Education', 'Occupation', 'Race',
'Gender']):
    ax = fig.add_subplot(rows, cols, i+1)
    ax.set_title(column)
    if original_data.dtypes[column] ==np.object:
        original_data[column].value_counts().plot(kind='bar', axes = ax)
        plt.xticks(rotation='vertical')
plt.subplots_adjust(hspace=0.7, wspace=0.2)
plt.show()
```

Machine Learning

SVM – Use case

```
# Use label encoder to convert text to numberic

import sklearn.preprocessing as preprocessing


le = preprocessing.LabelEncoder()

original_data['Occupation'] =

le.fit_transform(original_data['Occupation'].astype(str))

original_data.head()

()
```

PRIME INTUIT
Finishing School

Machine Learning

SVM – Use case

```
original_data['Target'] =
le.fit_transform(original_data['Target'].astype(str))
original_data.tail()
```

```
original_data.Target.unique
```

PRIME INTUIT
Finishing School

Machine Learning

SVM – Use case

https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data

SVM – Use case

https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data

PRIME INTUIT
Finishing School

Machine Learning

SVM – Use case

https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data

PRIME INTUIT
Finishing School

Machine Learning

SVM – Use case

https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data

PRIME INTUIT
Finishing School

Machine Learning

SVM – Use case

https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data

PRIME INTUIT
Finishing School

Machine Learning

SVM – Use case

https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data

PRIME INTUIT
Finishing School

Machine Learning

SVM – Use case

https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data

PRIME INTUIT
Finishing School

Machine Learning

SVM – Use case

https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data

PRIME INTUIT
Finishing School

Machine Learning

SVM – Use case

https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data

PRIME INTUIT
Finishing School

Machine Learning

Decision Tree

PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

Should I go biking on a Saturday morning

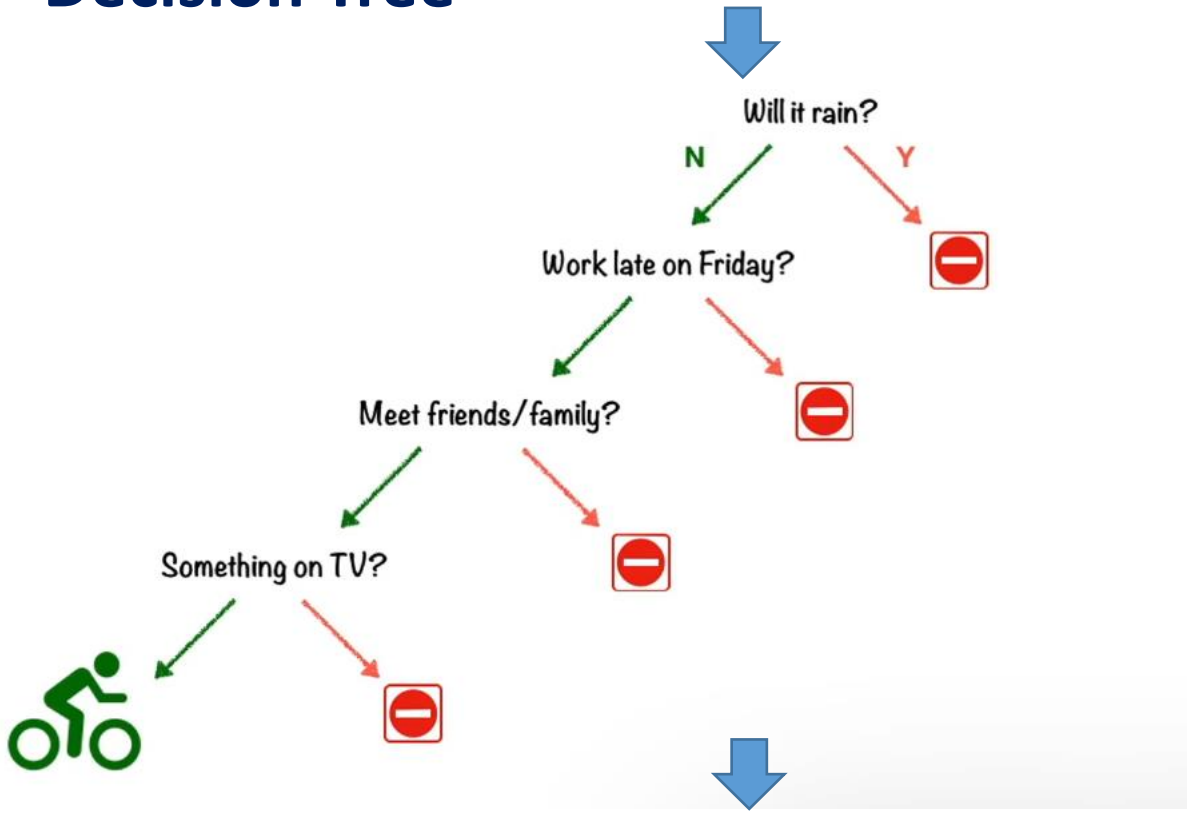
- Will it rain ?
- Will I work late on Friday ?
- Do I need to meet friends / family ?
- Is there something on TV ?

PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

Decision Tree

Input variables / Predictors



Outcome / Output Variables

PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

Decision Tree

- Helps predict the outcome given a set of inputs
- In business, it represents visually how a decision is taken with inputs and consequences of each decision

Machine Learning (Decision Tree)

Decision Tree

- Can be used for classification, just like SVM
- With SVM, can't understand the relationship between input variable and outcome
- Decision Trees are not a black box

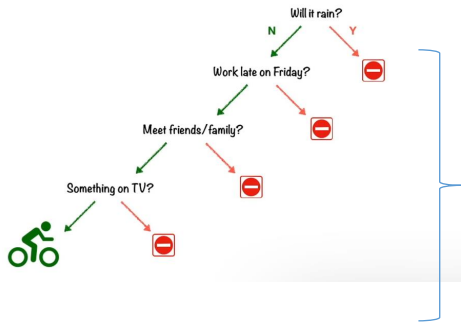
PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

Decision Tree

- Inputs can be categorical or continuous
- Output can also be categorical or continuous
  - Called regression tree when outcome is continous

root



leafs

PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

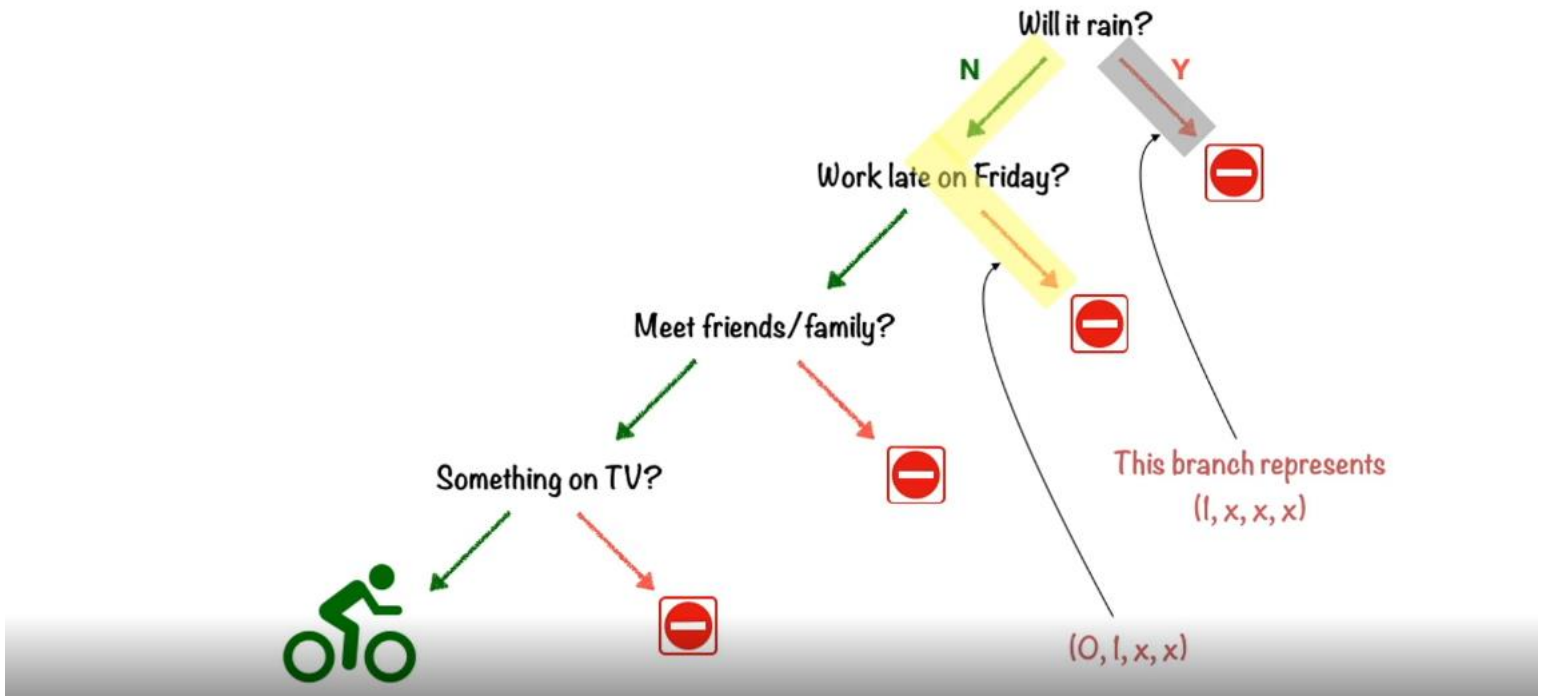
Decision Tree

- The leaves of the tree are the outcomes
- In classification problem there are class labels
- The tree gives the more likely outcome given the values of the variables
- The branches are combinations of predictor values leading to an outcome

PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

Decision Tree



PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

Decision Tree Learning

- The process of creating / learning s decision tree from trining data
- Start with training data in the form of feature vector, label / outcome
  - (1, 0, 1, 1) No biking
  - (0, 0, 0, 0) Go biking
- Obtain a decision tree used to classify / predict a new instance
- A supervised learning approach

PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

Decision Tree Learning

- Recursive partitioning is the most common strategy for decision tree learning
- Decision tree learning algorithms
  - CART
  - ID3
  - C4.5
  - CHAID

PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

Decision Tree Learning

- Algorithm needs to tell us the order in which the predictors are evaluated
- If the predictors is continuous, the tree needs to split the variables into ranges

PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

Greedy Algorithm for Learning a Decision Tree

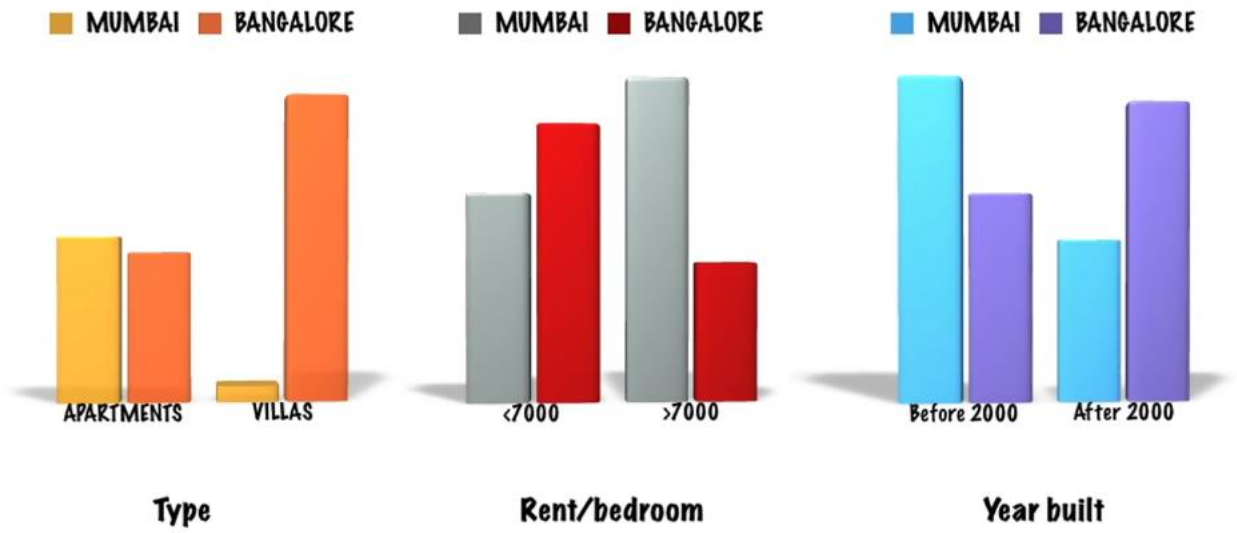
- Given the type of housing, rent /bedroom and the year it was built
- Predict the city to which a residence belongs
  - Mumbai (or) Bangalore

PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

Greedy Algorithm for Learning a Decision Tree

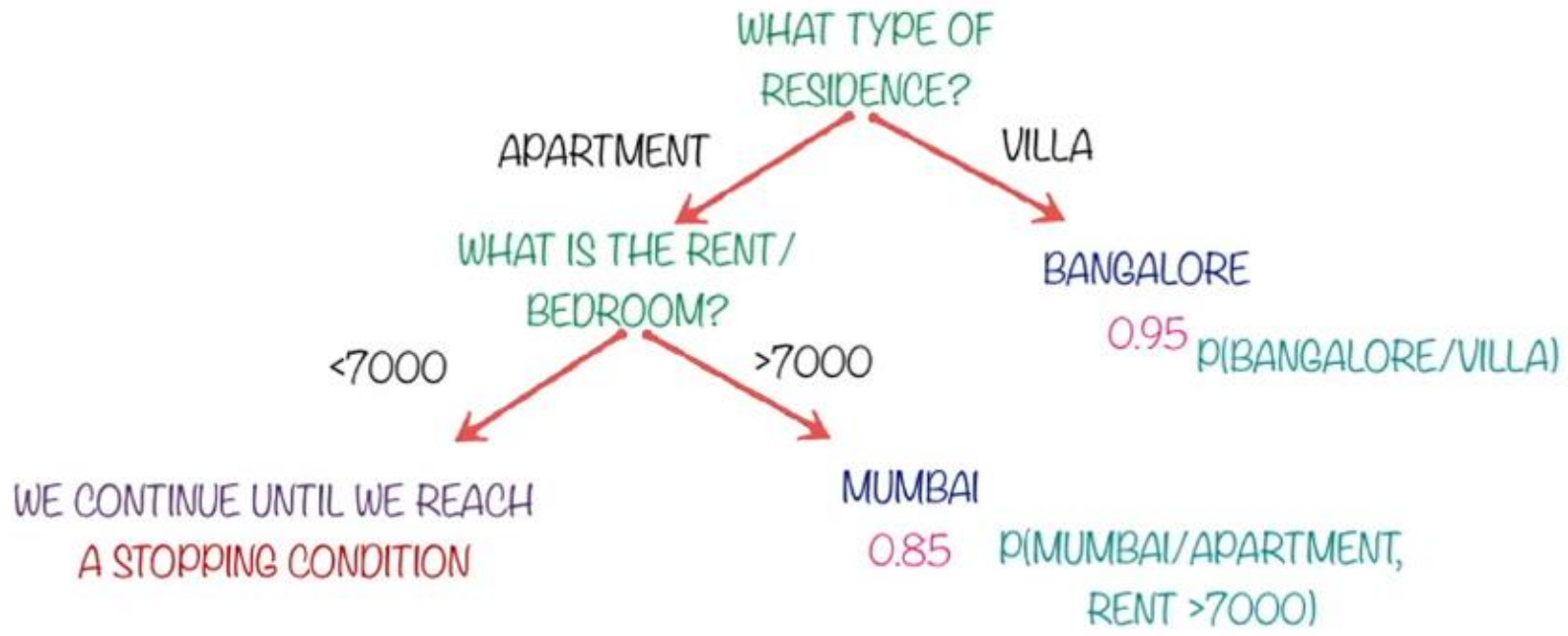
Draw a histogram for each attribute, for residences in each city



PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

Greedy Algorithm for Learning a Decision Tree



This is Recursive Partitioning

PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

The stopping condition could be

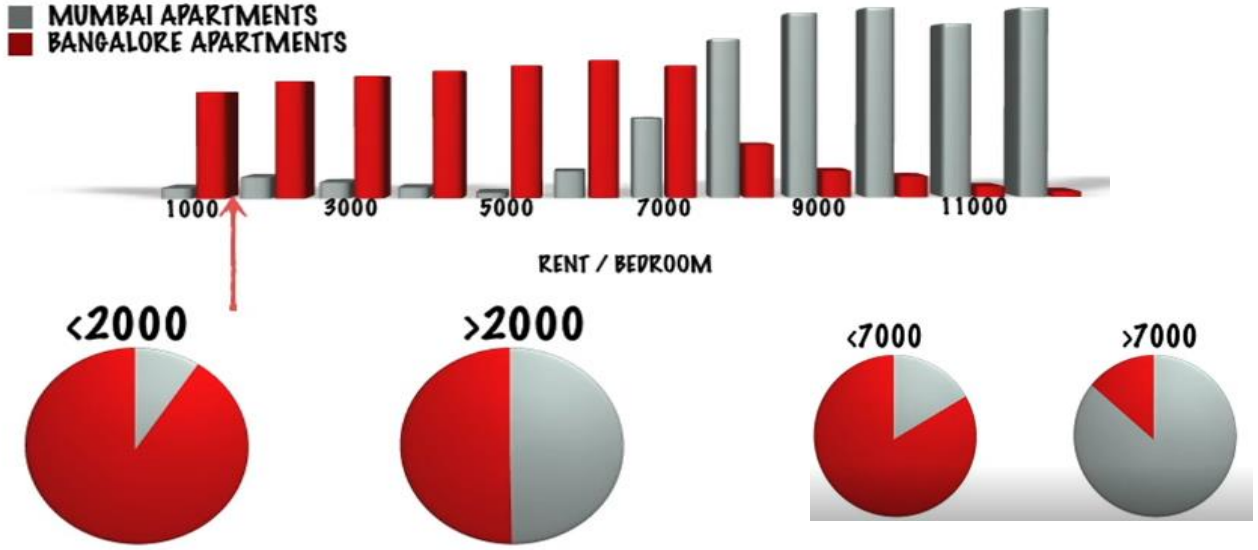
- All our subsets are mostly homogeneous
- We have run out of attributes
- The Tree is too large

PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

The best split for a continuous input variable

- What is the best split for rent in our example ?
- Plot a histogram based on rent per bedroom



The best split is the point where the subsets we get are mostly homogeneous

PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

Information Gain

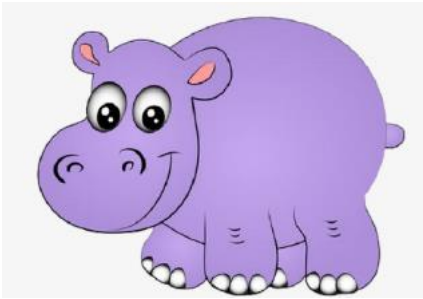
- Any statement, News or Message Contains Information
- Some have more information and some have less
- The Idea of information gain is to reduce entropy and maximize information

PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

Information Gain (closer look)

Let's say you have to classify an animal as a giraffe or a hippo



We are told the animal has 4 legs

This basically useless as both the animals have 4 legs

If we are told that the animal is 10 feet tall

This is a useful information, It tells us the animal is very likely a giraffe

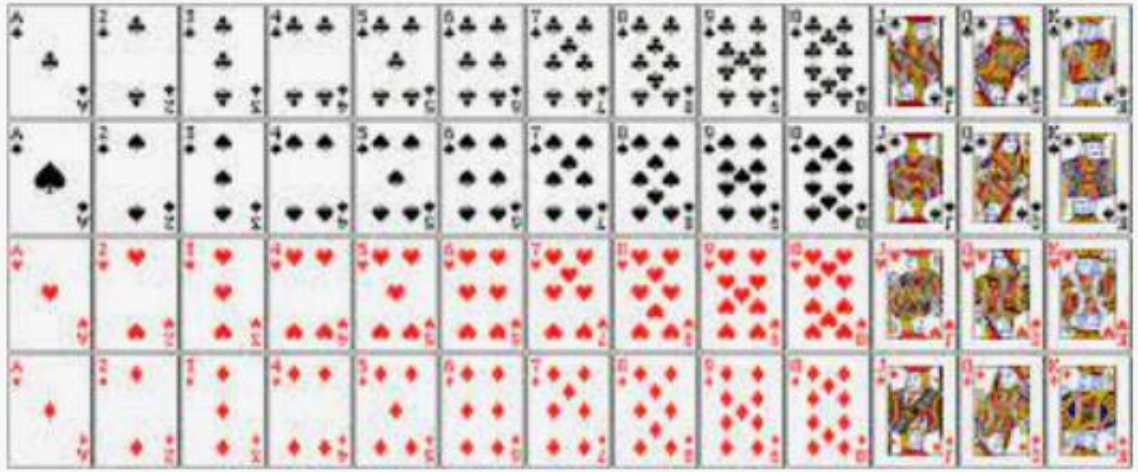
So clearly – The values of some attributes give us more information than others
&
There are mathematical way to measure this information

PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

Information Gain

- **Another example: Game of cards**



Guess the card held by your opponent

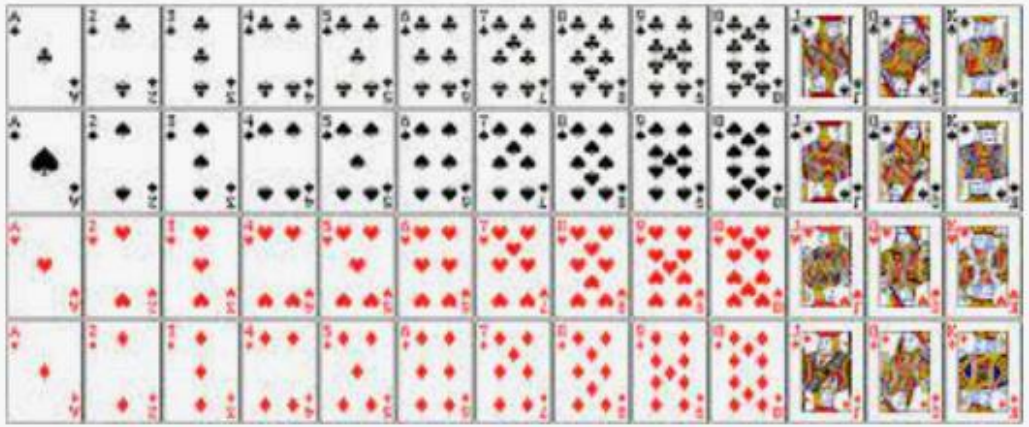
You are allowed to ask Yes or No questions

To begin with you have 52 possible outcomes

PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

Initially there are 52 possible outcomes in all



Your question 1: Is the card an ACE ?

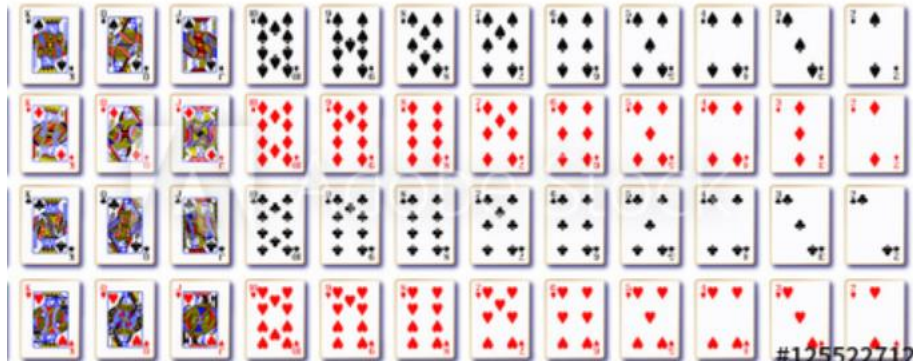
Yes

No

You are left with 4 possible outcomes

The answer Yes gives us more information then answer NO

You are left with 48 possible outcomes



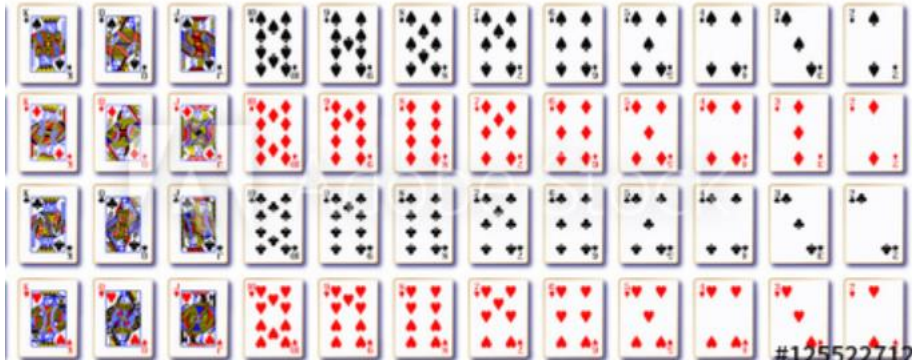
PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

Initially there are 52 possible outcomes in all

Yes

No



The answer "Yes" has a lower probability

P(Yes) = 4/52

The answer "No" has a Higher probability

P(Yes) = 48/52

The lower the probability of the answer, the more information you get

Your question 1: Is the card an ACE ?

The answer Yes gives us more information then answer NO

The probability of any occurrence is b/n 0-1 and the Log of value is always –ve regardless of the base of the log

.

The lower the probability the more –ve the log value will be, however since we are considering the –ve log for info content the lower the probability of the value Higher the information content

If X is the random variable that represents the answer to our question

Information content of (X = Yes) = - Log (P(X = Yes))

Information content of (X = No) = - Log (P(X = No))

Information content of ($X = x$) = - Log (P($X = x$))

PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

Information Gain

- Any statement, News or Message Contains Information
- Some have more information and some have less
- The Idea of information gain is to reduce entropy and maximize information

PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

Information Gain

If X is the random variable that represents the answer to our question

Information content of ($X = x$) = - Log (P($X = x$))

Average value of the information content also called the expected value = ΣP($X = x$) (- Log (P($X = x$)))

Entropy H(X)

Entropy is the amount of uncertainty / Unpredictability there is in the answer

Entropy increases with:

1) Number of possible answers
2) The evenness of probability distribution

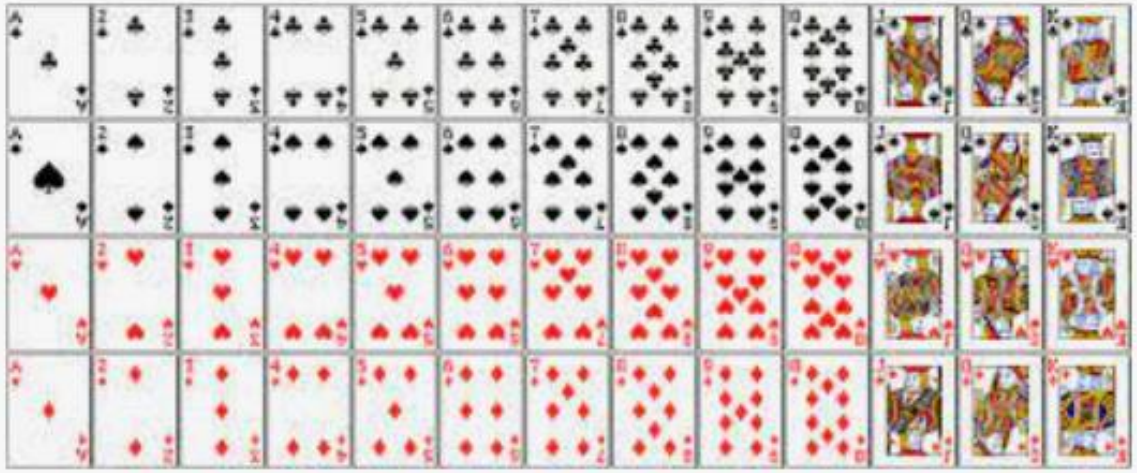
P(Yes) = 0 => There is no uncertainty = > entropy = 0

Yes and No have equal probability = > Very High entropy

PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

Back to our Cards example



Initially there are 52 possible outcomes

Each card has the same probability = 1/52

Information content of ($X = x$) = - Log (P($X = x$))

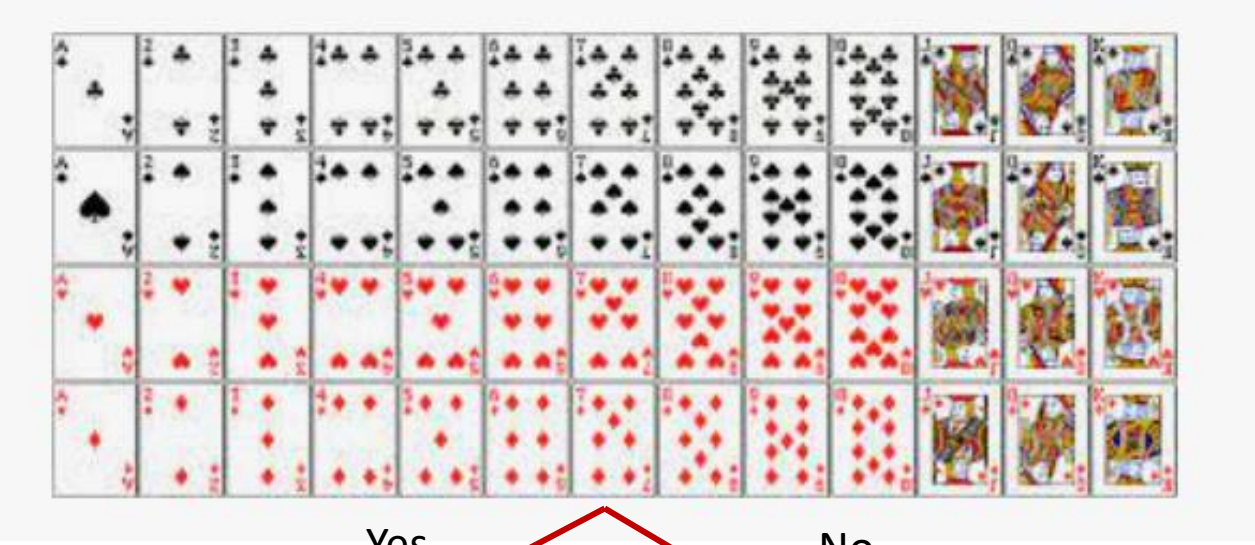
Entropy H(X) = Average value of the information content (Also called as expected value
= ΣP($X = x$) (- Log (P($X = x$)))

Entropy = H(X) = Σ(1/52) (-LOG(1/52))
= LOG(52)

PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

Back to our Cards example



Yes

No

Entropy = H(X/Q1 = Yes)
= Log(4) P(Yes) = 4/52

Entropy = H(X/Q1 = No) =
Log (48) P(No) = 48/52

Within each group the entropy has decreased

The more homogenous each group is the lower the entropy

Entropy H(X) = Average value of the information content (Also called as expected value
= ΣP($X = x$) (- Log (P($X = x$)))

Before we have asked the question the uncertainty (entropy) in our guess is very high

Entropy = H(X) = Σ(1/52) (-LOG(1/52))
= LOG(52)

However, Once we ask the question,
Is the card an ACE ?

PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

Information Gain

Before we have asked any yes/no questions, the uncertainty (Entropy) in our guess is very high

Entropy = H(X)

After we ask the question, Is the card ACE ?

Entropy after Q1 = H(X / Q1)

Information Gain = Reduction in Entropy Overall = H(X) – H (X/Q1)

As we saw whenever we ask a question, subsets are formed

When each of these subsets are homogenous, the information gain maximum

PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

Information Gain

Entropy = H(X)

Entropy after Q1 = H(X / Q1)

Information Gain = Reduction in Entropy Overall = H(X) – H (X/Q1)

As we saw whenever we ask a question, subsets are formed

When each of these subsets are homogenous, the information gain maximum

Is It an ACE ?
H(X/Q1) = 4/52 * Log(4) + 48 / 52 * Log (48)

IQ = H(X) – H(X/Q1) = 0.12

Is It a Black card ?
H(X/Q1) = Log(26)

IQ = H(X) – H(X/Q1) = Log(52) – Log(26) = 0.30

More Homogenous our subset the greater the information gain……

PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

Decision Tree Learning

- Recursive partitioning is the most common strategy for decision tree learning
- Decision tree learning algorithms

• CART	GINI Impurity
• ID3 • C4.5	Information Gain
• CHAID	Chi Square Statistic

Each of these learning algorithms have slightly different way of arriving or measuring the homogeneity of a subset

PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

CART

CART is another decision tree learning method

CART : Classification and Regression Trees

It uses a different way to choose an attribute

The idea is to Minimize the GINI Impurity at each step

Idea behind GINI impurity is to choose an attribute such that if you stop the decision tree with that attribute and go no further, The probability of a false label is minimized

PRIME INTUIT
Finishing School

Machine Learning (Decision Tree Lab)

Data for sample case

https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/

https://www.kaggle.com/uciml/pima-indians-diabetes-database

PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

```
(base) PS C:\Users\LENOVO> conda install python-graphviz
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##
```

PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

Decision Tree Learning

- Recursive partitioning is the most common strategy for decision tree learning
- Decision tree learning algorithms

• CART	GINI Impurity
• ID3 • C4.5	Information Gain
• CHAID	Chi Square Statistic

Each of these learning algorithms have slightly different way of arriving or measuring the homogeneity of a subset

PRIME INTUIT
Finishing School

Machine Learning (Random Forest)

Random Forest

PRIME INTUIT
Finishing School

Machine Learning (Random Forest)

Random Forest

PRIME INTUIT
Finishing School

Machine Learning (Overfitting)

GFC -2007

Some ways of Mitigating this problem:

- **Cross Validation**
- **Regularization**
- **Ensemble Learning**

PRIME INTUIT
Finishing School

Machine Learning (Overfitting)

Example: Restaurant rating

A Model should be simple and generic
Vs
Model is complex and more accurate on training data

PRIME INTUIT
Finishing School

Machine Learning (Overfitting)

Overfitting happens when model picks up random phenomena or noise present in the training set. Instead of the underlaying relationship between the input and output

Why is Over fitting such a common problem:

The training set is only part of the much larger set

We are trying to find models that describe this much larger set

It's like trying to describe a photograph but you are shown only a small zoomed portion of the entire photograph

PRIME INTUIT
Finishing School

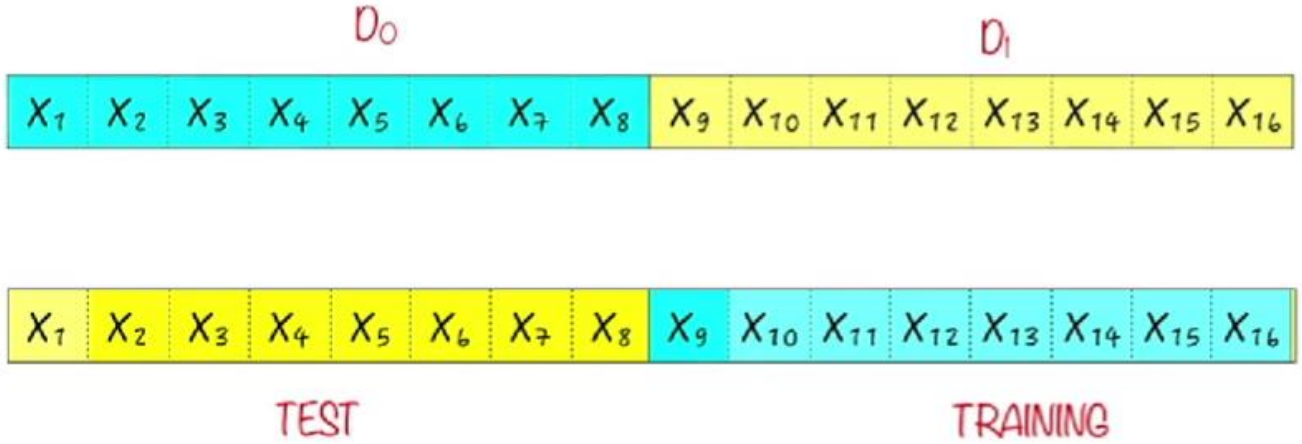
Machine Learning (Overfitting)

Cross Validation

THE BELOW TABLE REPRESENTS THE ENTIRE TRAINING DATA SET

1. DIVIDE THE TRAINING SET RANDOMLY INTO TWO EQUAL PARTS - D_0 AND D_1

2. USE D_0 TO TRAIN THE MODEL AND D_1 TO TEST THE PERFORMANCE



3. THEN, USE D_1 TO TRAIN THE MODEL AND D_0 TO TEST THE PERFORMANCE

THE BEST MODEL IS THE ONE WITH BEST AVERAGE PERFORMANCE

THIS TECHNIQUE IS CALLED

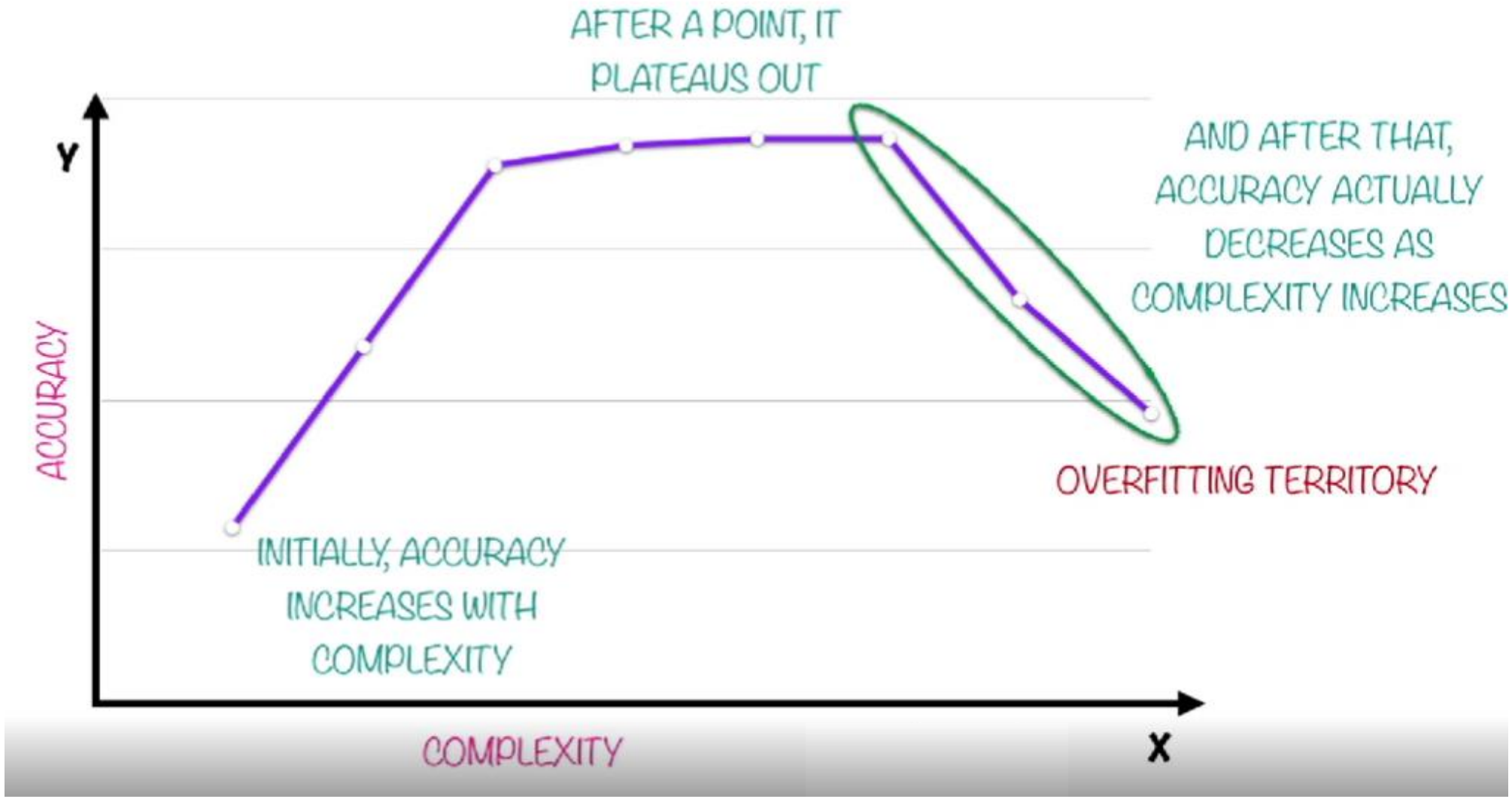
2-FOLD CROSS VALIDATION

PRIME INTUIT
Finishing School

Machine Learning (Overfitting)

Regularization

$E'(f) = E(f) + \lambda R(f)$



PRIME INTUIT
Finishing School

Machine Learning (Overfitting)

Ensemble Technique

Bagging

- Random Forest

Boosting

- ADABoost
- Gradient Boosting
- XGBoosting

PRIME INTUIT
Finishing School

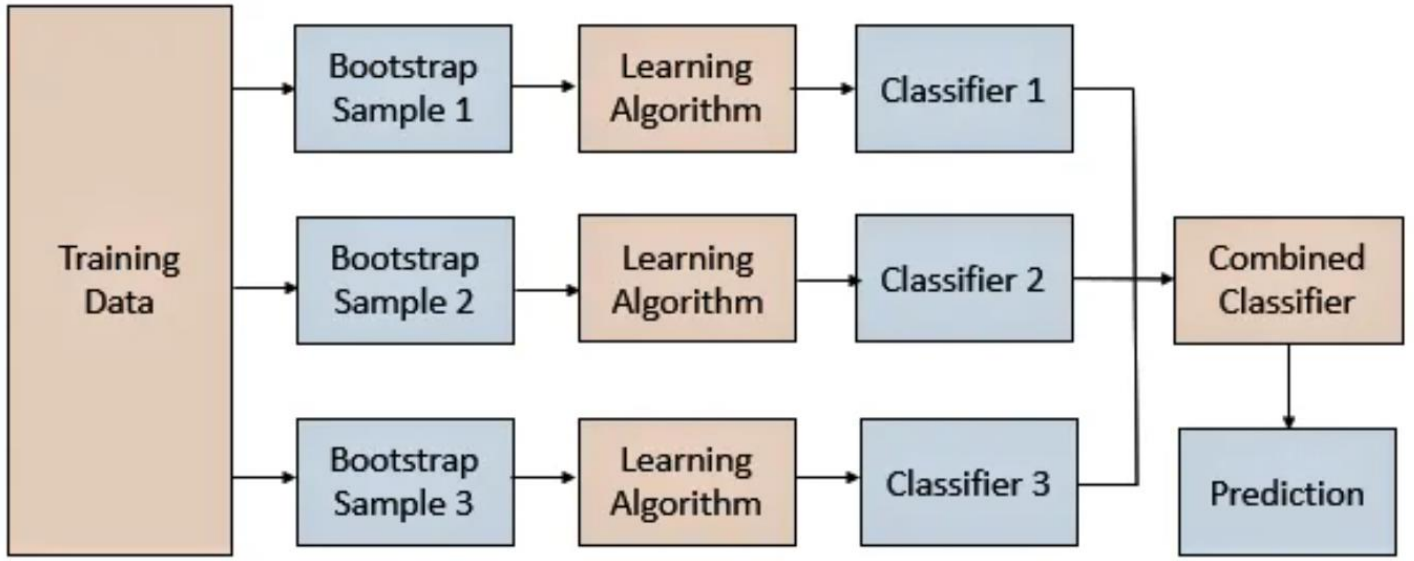
Machine Learning (Decision Tree)

Bagging

- **Bagging is a technique used to reduce the variance of our predictions by combining the result of multiple classifiers models on different sub-samples of the same data set.**

The steps followed in bagging are:

1. **Create Multiple datasets**
2. **Build Multiple Classifiers**
3. **Combine Classifiers**

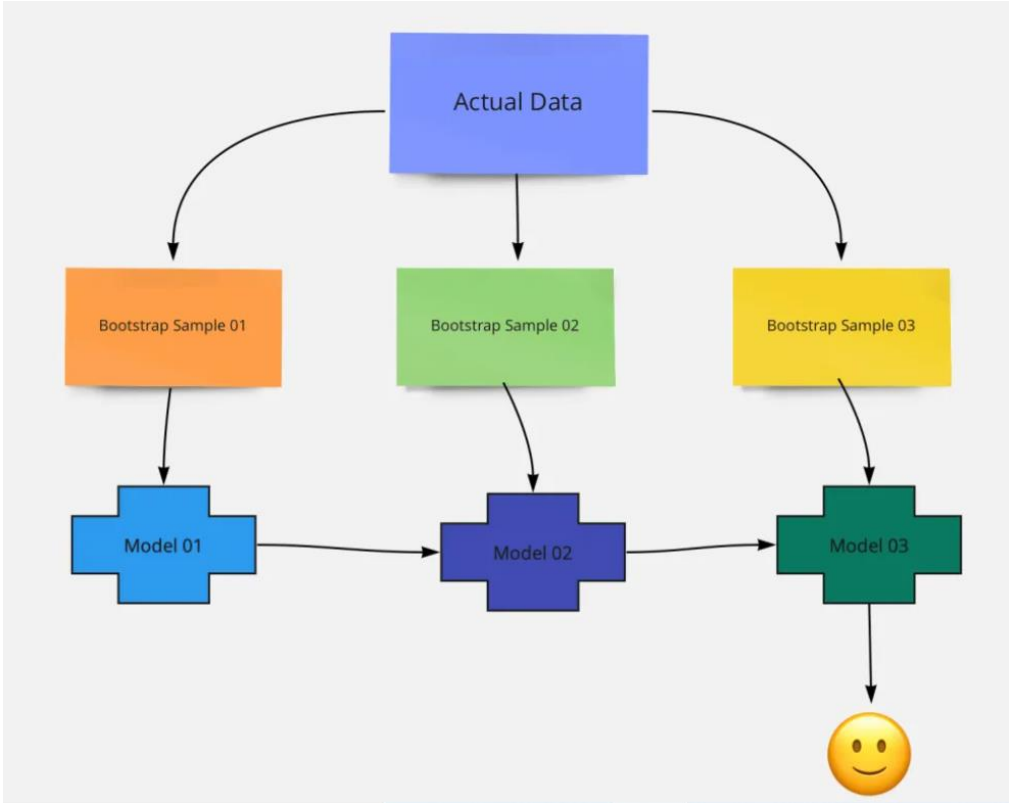


PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

Boosting

- In boosting method, all the individual models are built sequentially. Which means the outcome of the first model passes to the next model and so on.
- In bagging the models are built parallel so we don’t know what the error of each model is. Whereas in boosting once the first model built we know the error of that model. So when we pass this first model to the next model the intention is to reduce the error further. In some boosting algorithm, each model has to reduce a minimum of 50% of error



PRIME INTUIT
Finishing School

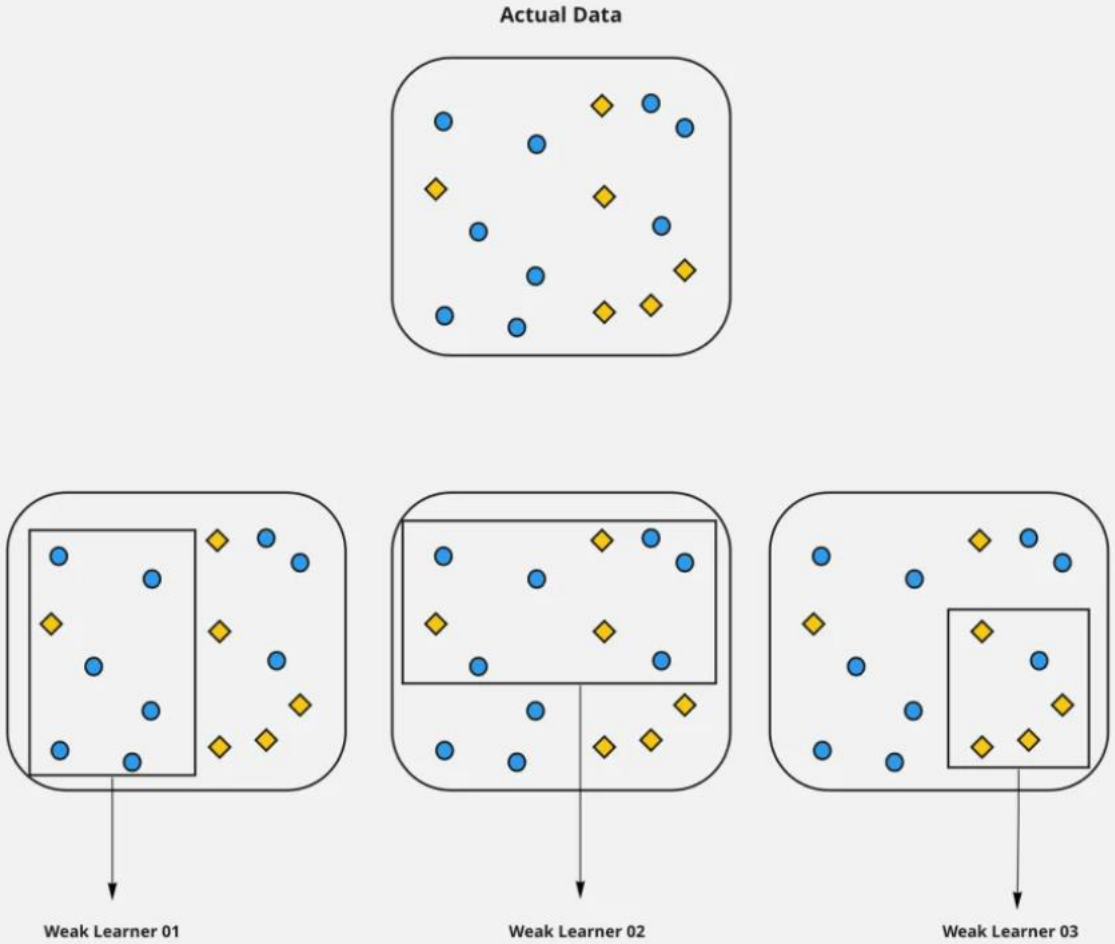
Machine Learning (Decision Tree)

Bootstrapping

For each model, we need to take a sample of data, but we need to be very careful while creating these samples of data. Because if we randomly take the data, in a single sample we will end up with only one target class or the target class distribution won't be the same. This will affect model performance.

To overcome this we need a smart way to create these samples, known as bootstrapping samples.

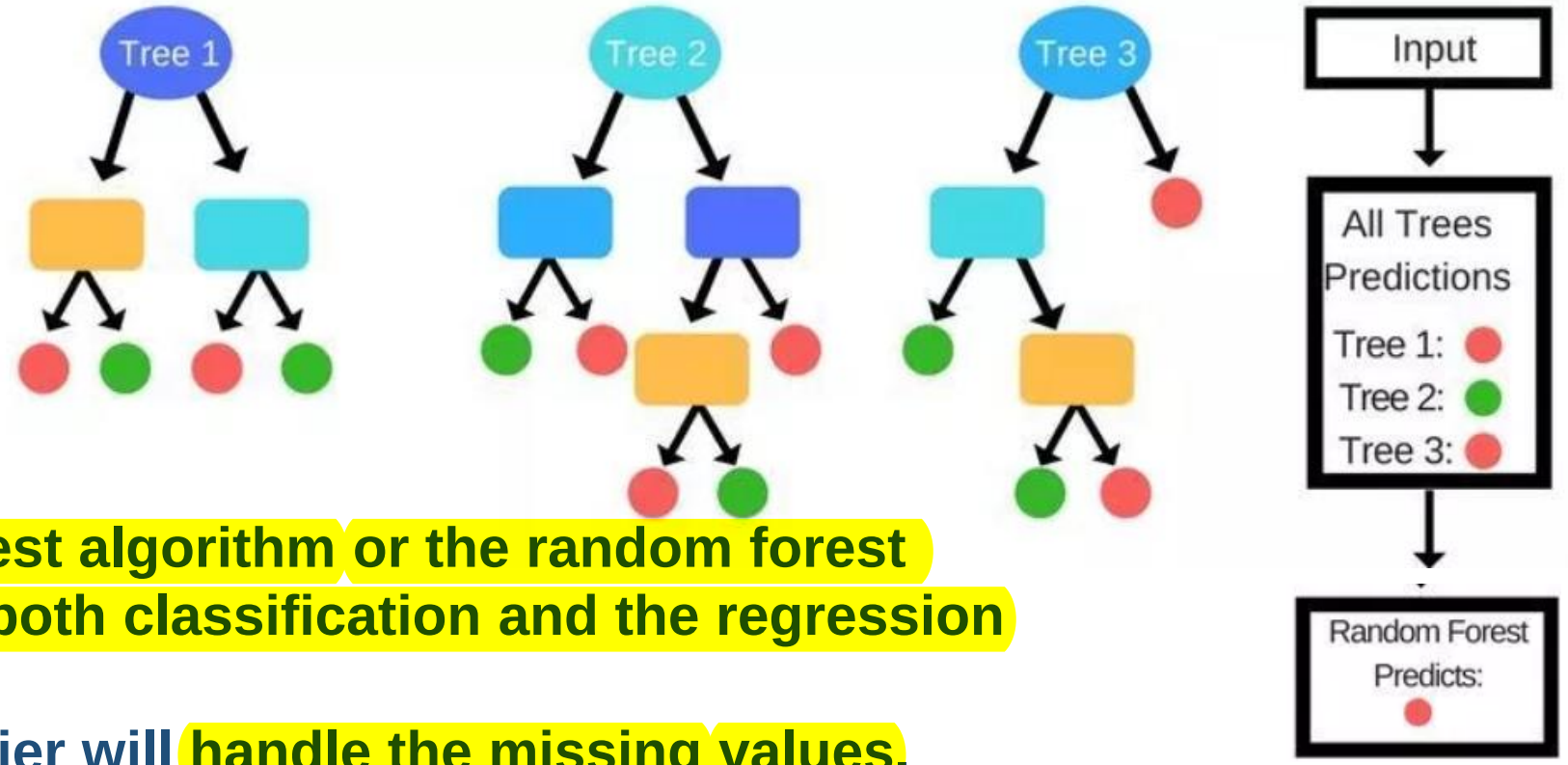
Bootstrapping is a statistical method to create sample data without leaving the properties of the actual dataset. The individual samples of data called bootstrap samples.



PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

Random Forest



- The same random forest algorithm or the random forest classifier can use for both classification and the regression task.
- Random forest classifier will handle the missing values.
- When we have more trees in the forest, a random forest classifier won't overfit the model.
- Can model the random forest classifier for categorical values also.

PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

Random Forest

Random Forest pseudocode:

- Randomly select “k” features from total “m” features.
- Where k << m
- Among the “k” features, calculate the node “d” using the best split point.
- Split the node into daughter nodes using the best split.
- Repeat 1 to 3 steps until “l” number of nodes has been reached.
- Build forest by repeating steps 1 to 4 for “n” number times to create “n” number of trees.

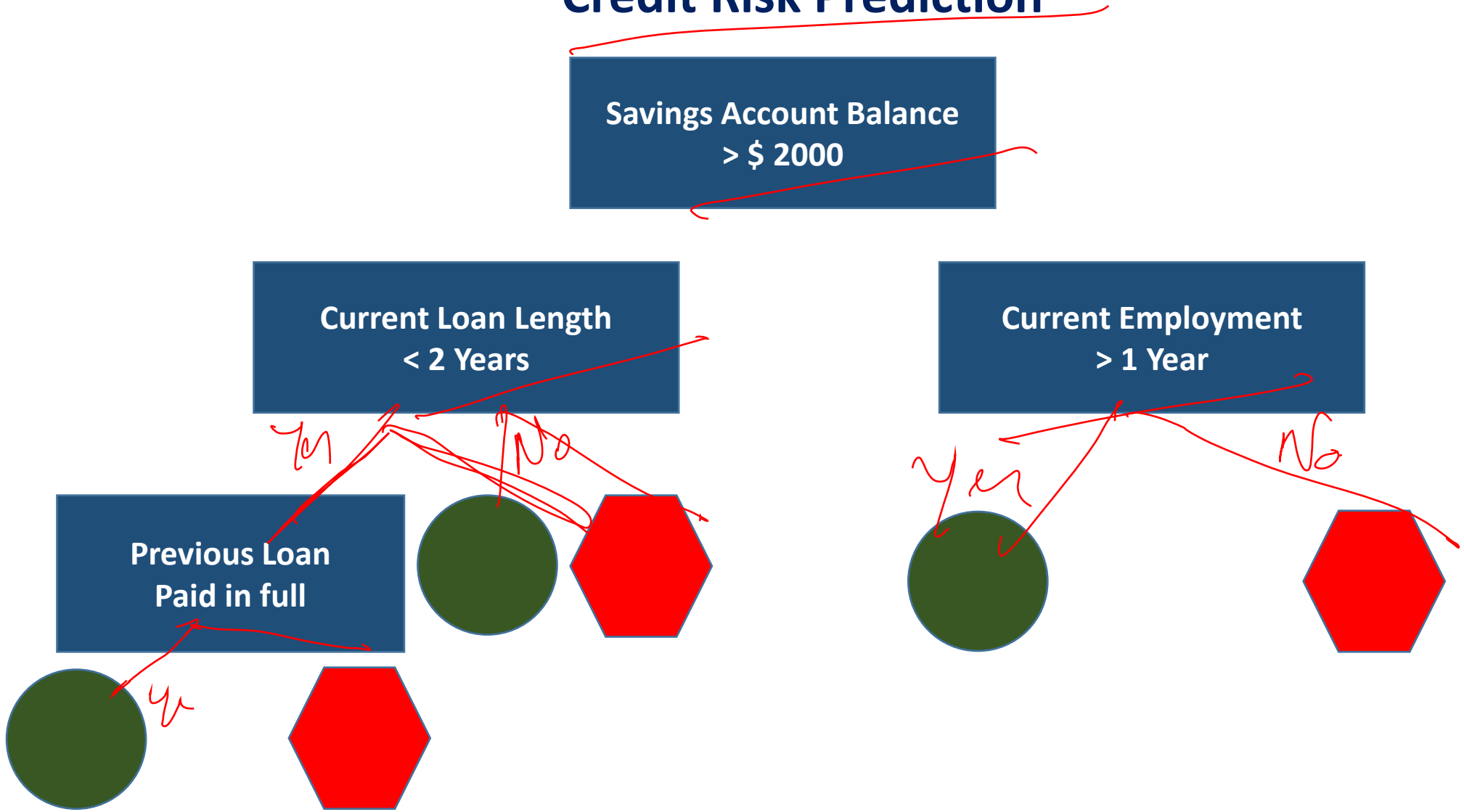
Random forest prediction pseudocode:

- Takes the test features and use the rules of each randomly created decision tree to predict the outcome and stores the predicted outcome (target)
- Calculate the votes for each predicted target.
- Consider the high voted predicted target as the final prediction from the random forest algorithm.

PRIME INTUIT
Finishing School

Machine Learning (Decision Tree)

Credit Risk Prediction



PRIME INTUIT
Finishing School

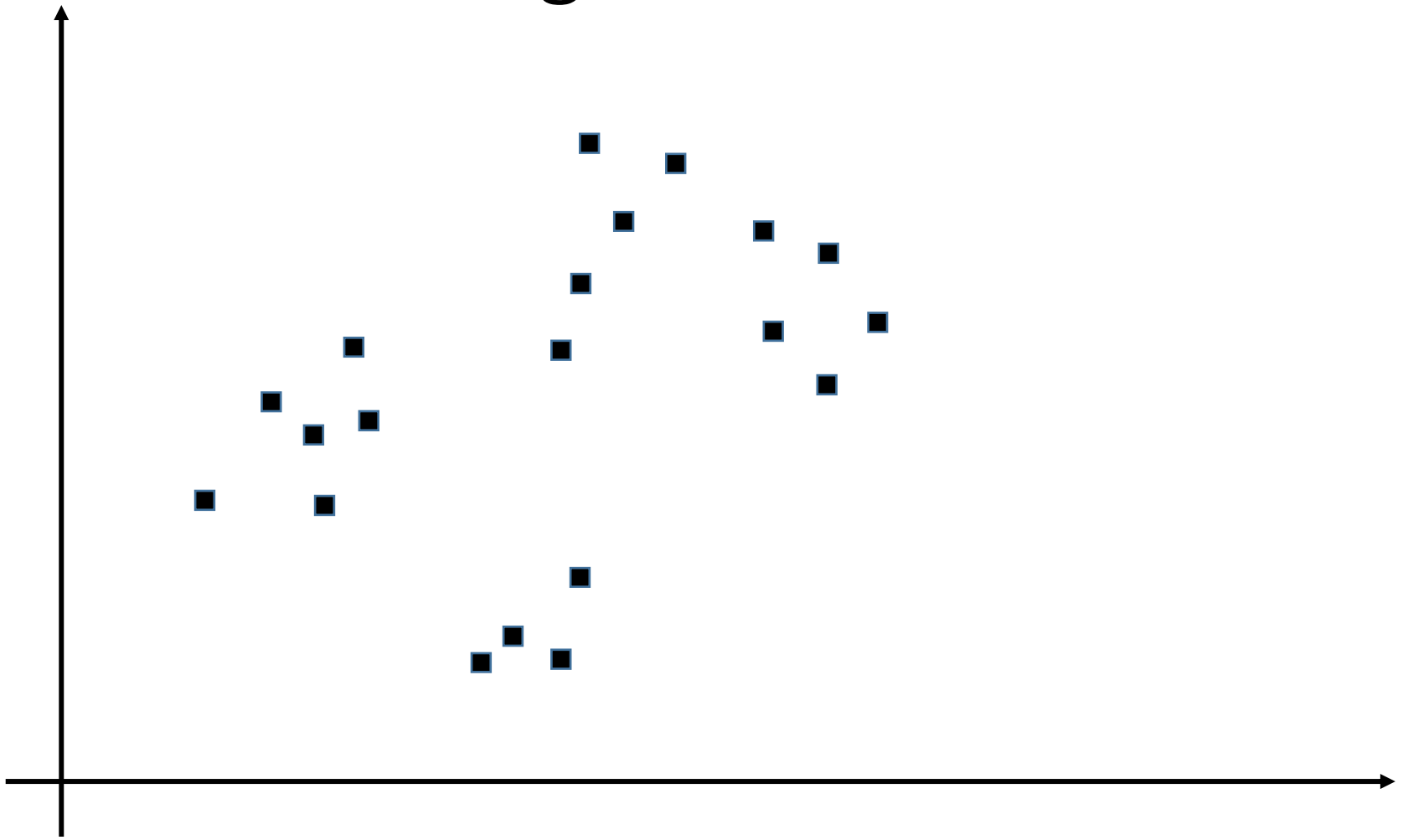
K Means

Machine Learning

- Supervised Machine Learning
- Unsupervised Machine Learning
- Semi Supervised Machine Learning
- Reinforcement Machine Learning

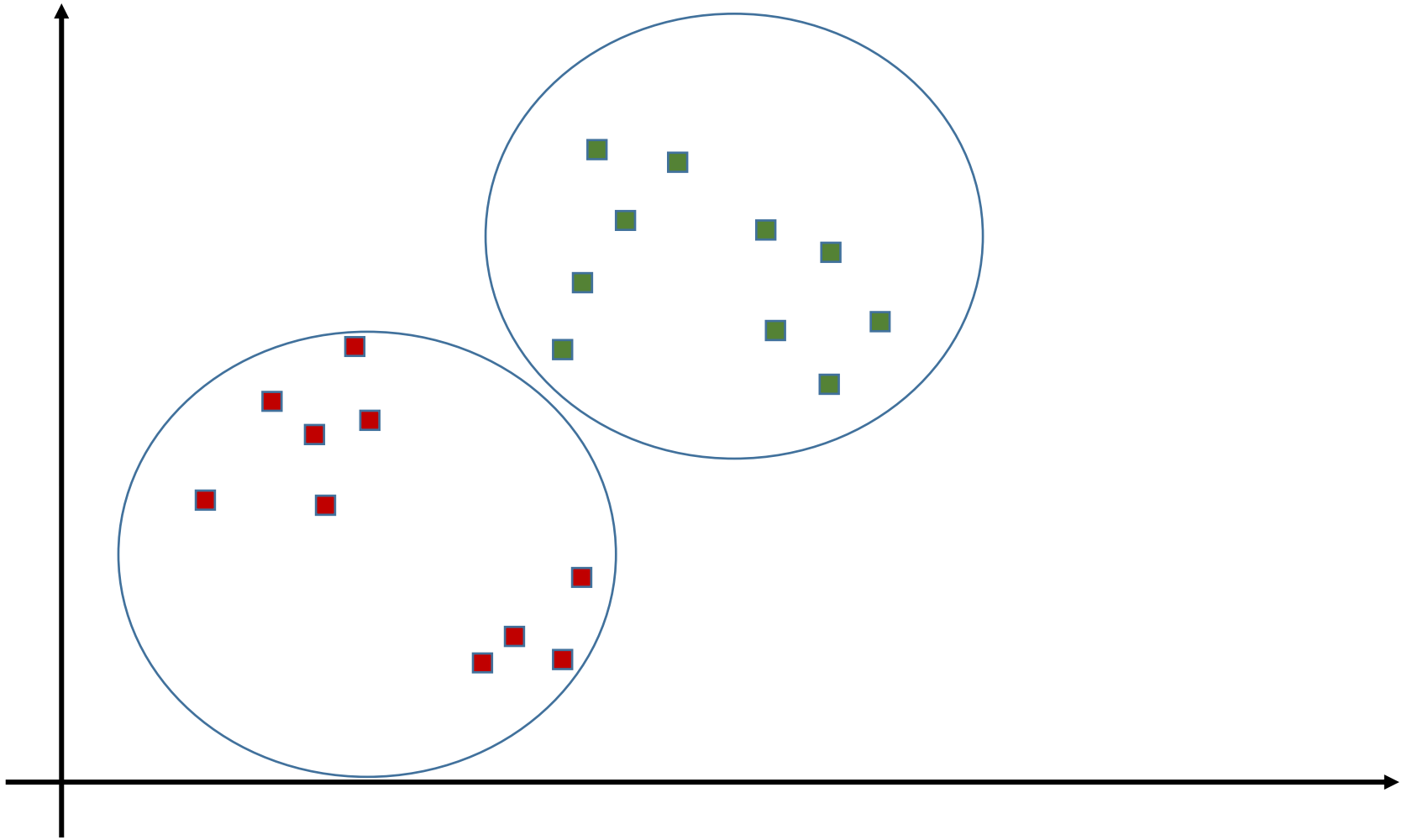
PRIME INTUIT
Finishing School

What is clustering



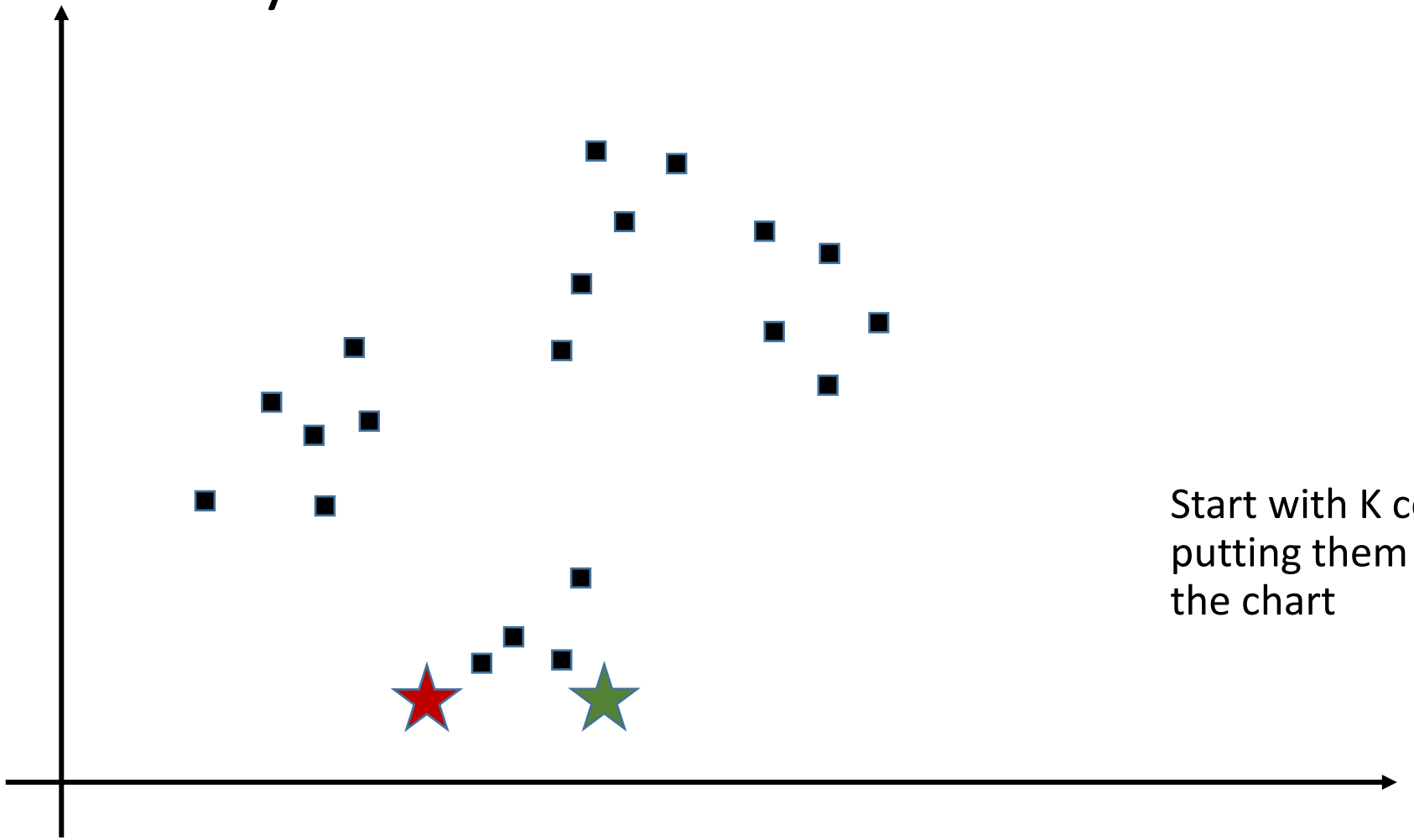
PRIME INTUIT
Finishing School

How many clusters



PRIME INTUIT
Finishing School

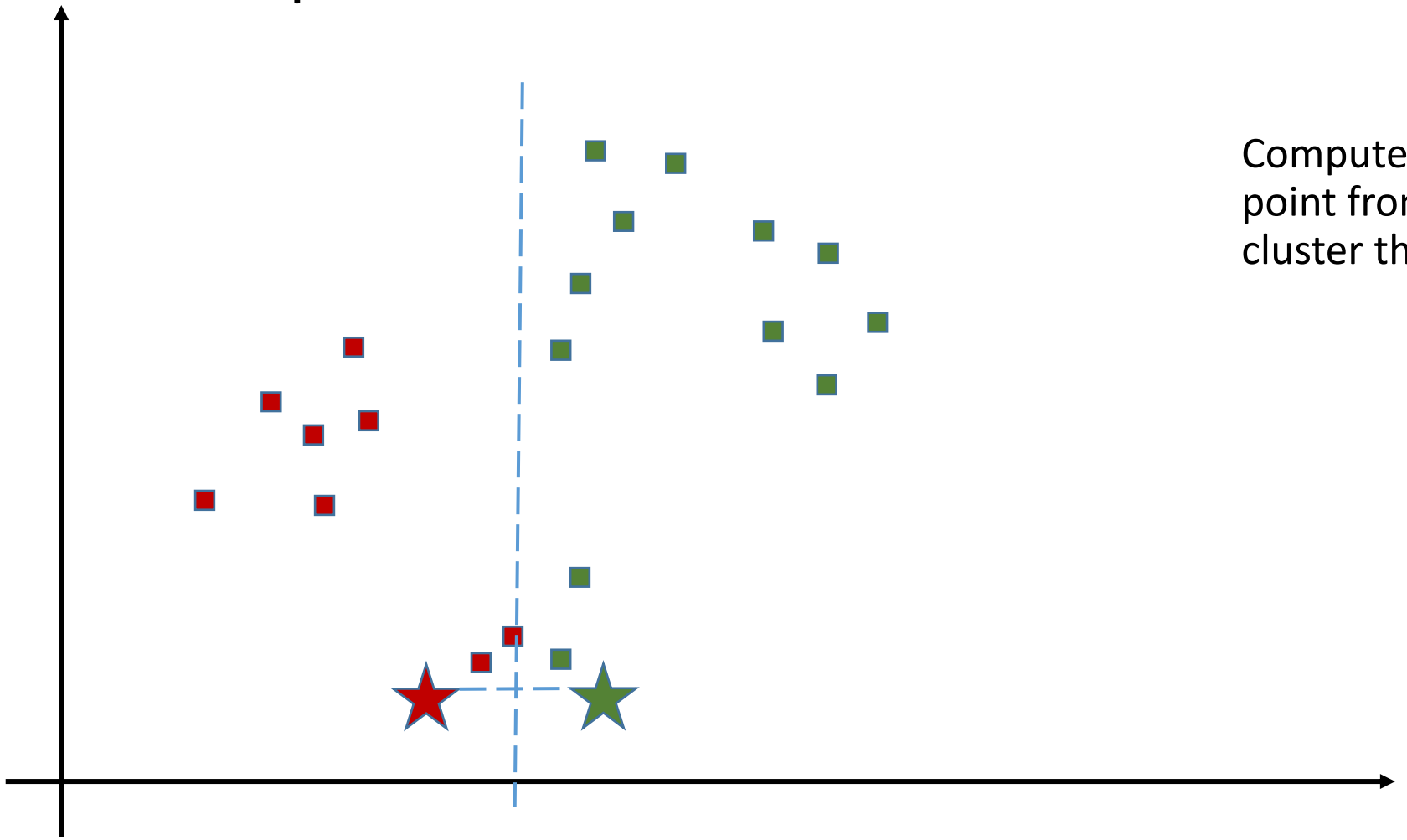
How many centroids



Start with K centroids by putting them at any place on the chart

PRIME INTUIT
Finishing School

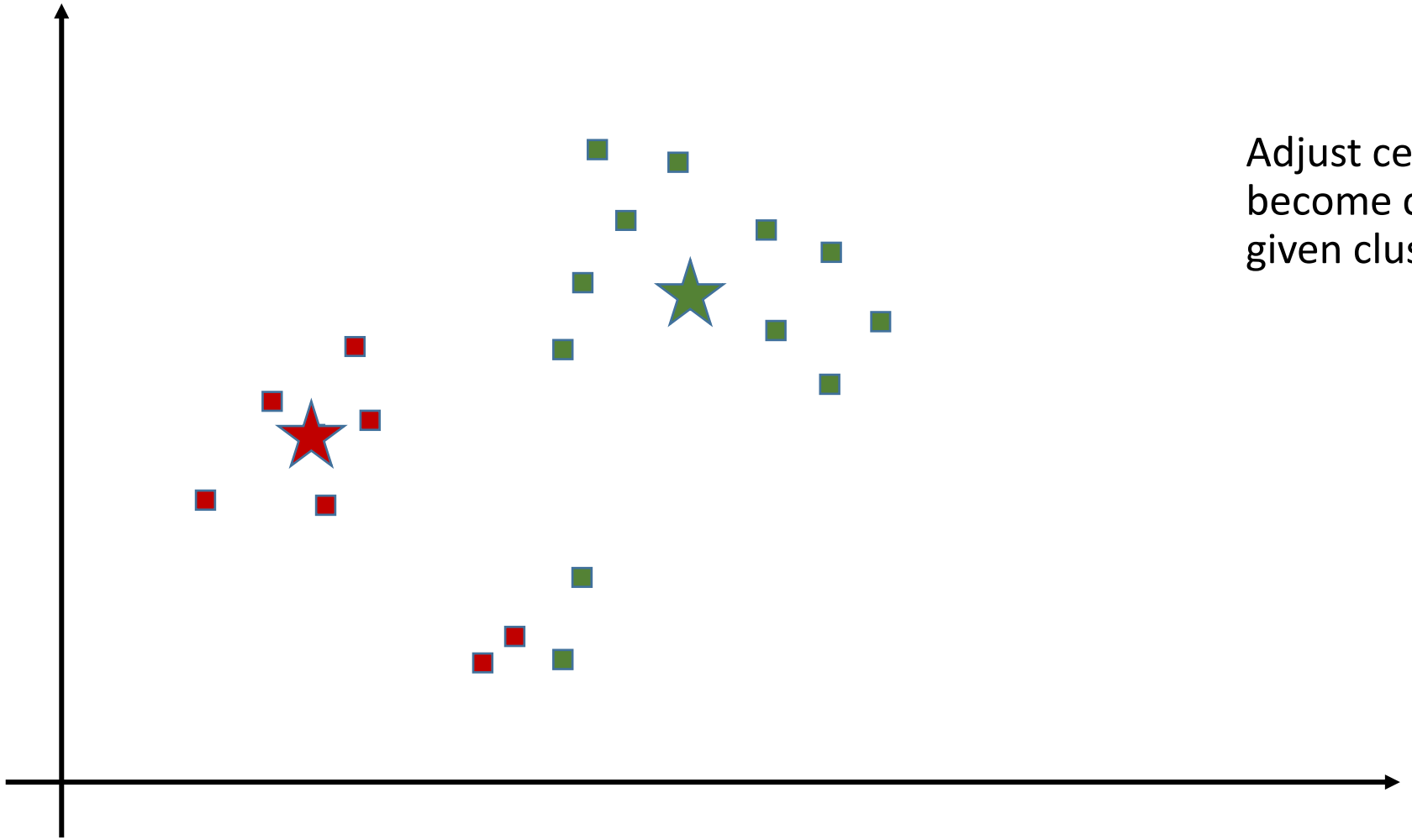
How to improve



Compute distance of every point from centroid and cluster them accordingly

PRIME INTUIT
Finishing School

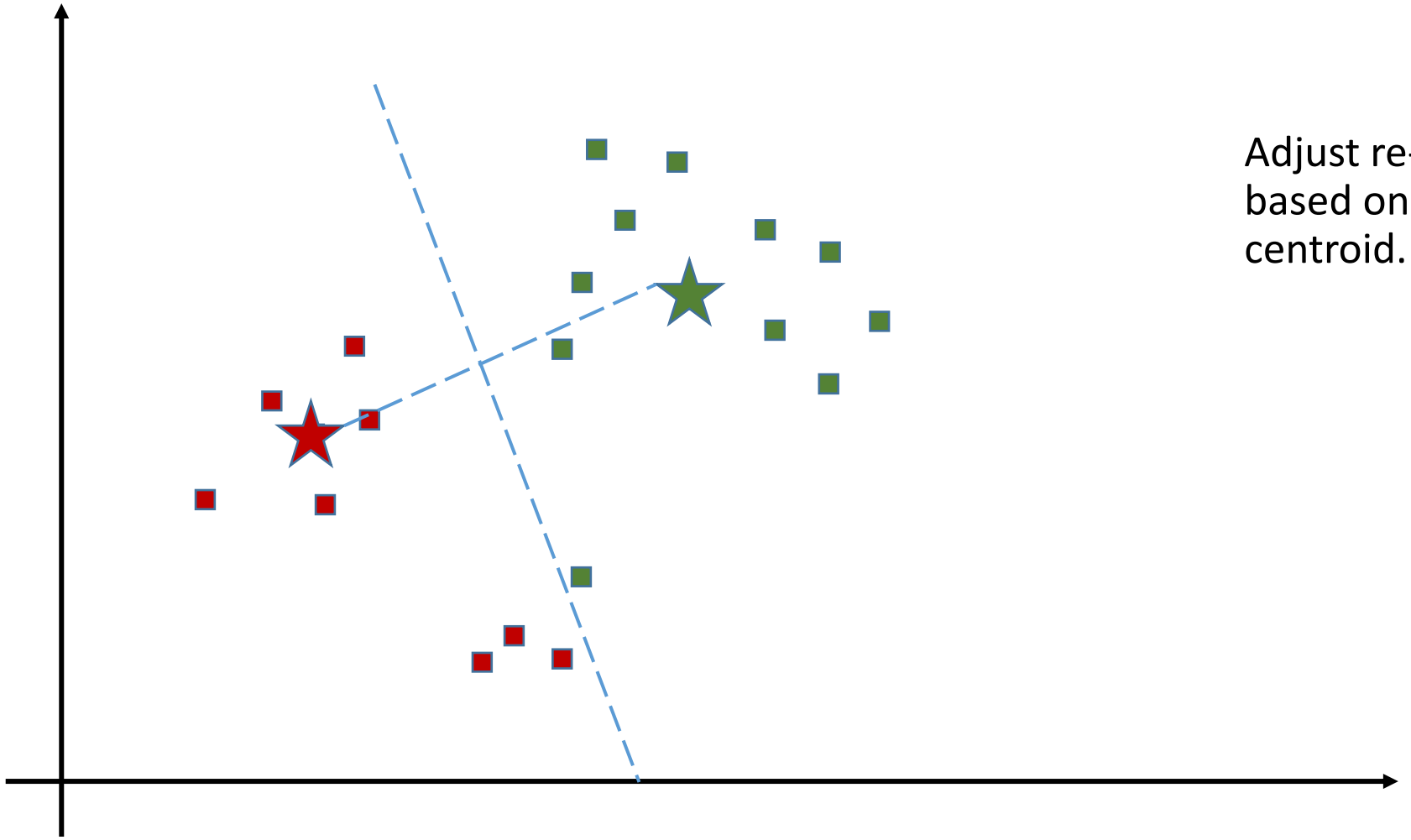
How to improve



Adjust centroids so that they become center of gravity for given cluster

PRIME INTUIT
Finishing School

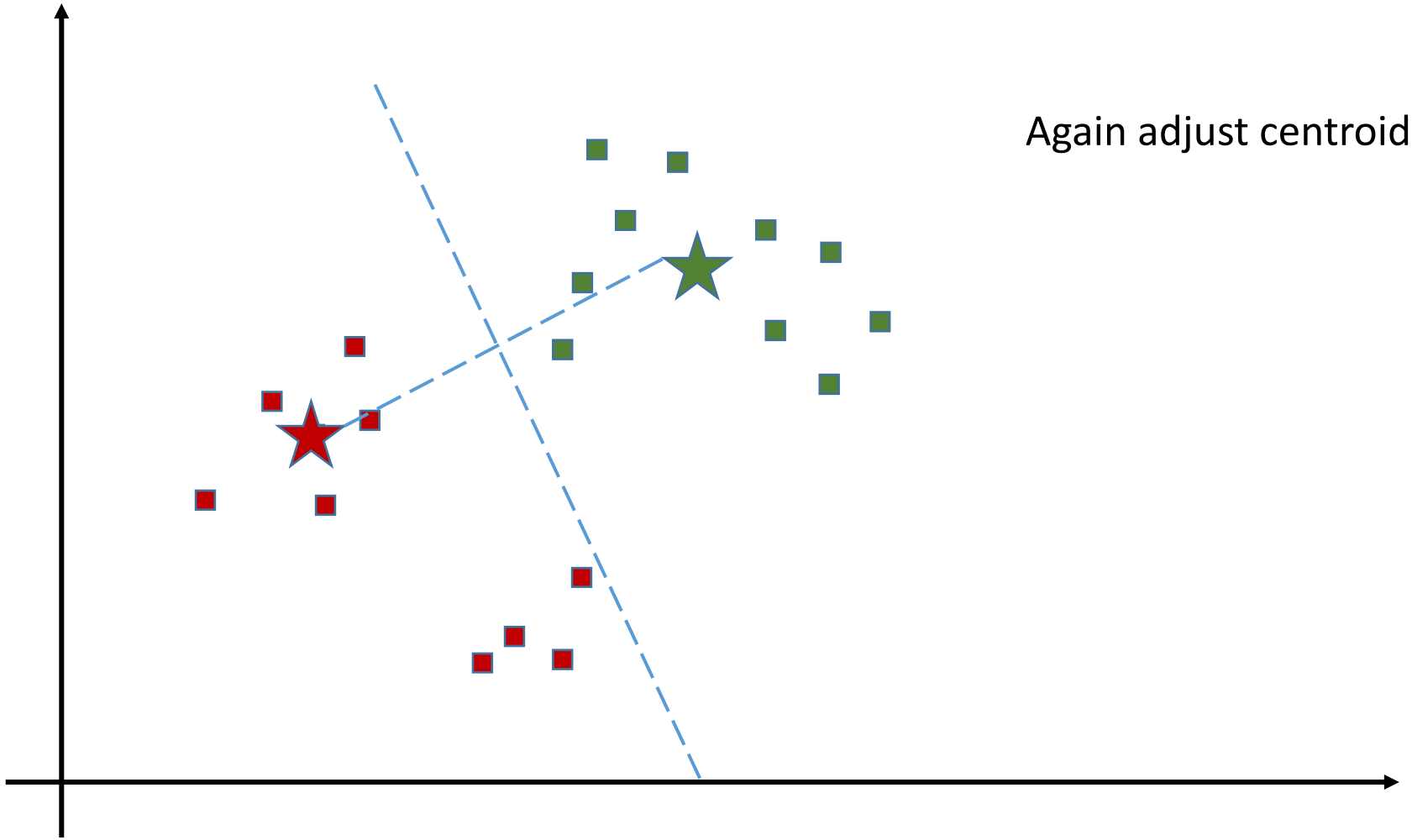
Re-clustering



Adjust re-cluster every point based on their distance with centroid.

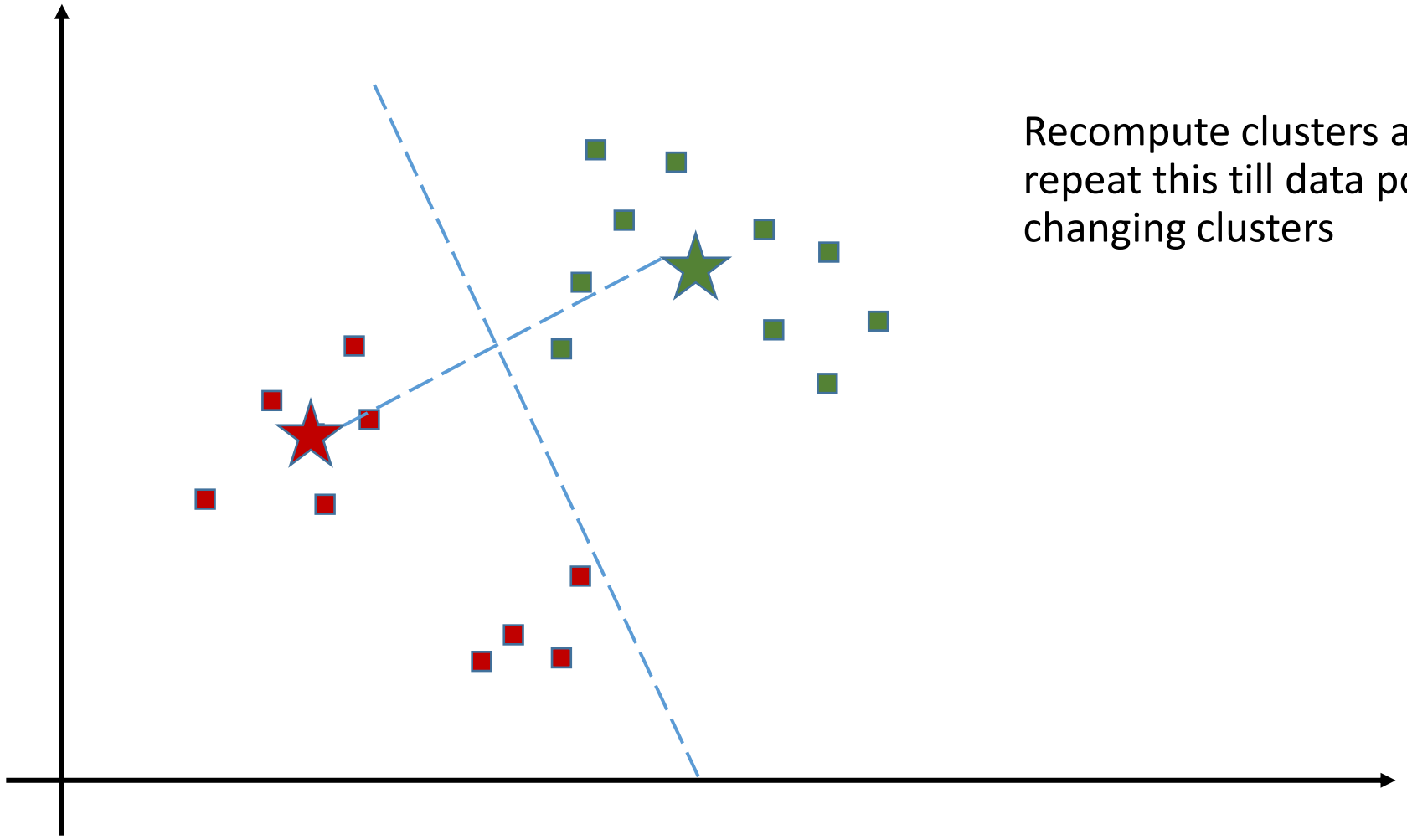
PRIME INTUIT
Finishing School

Re-clustering



Again adjust centroids

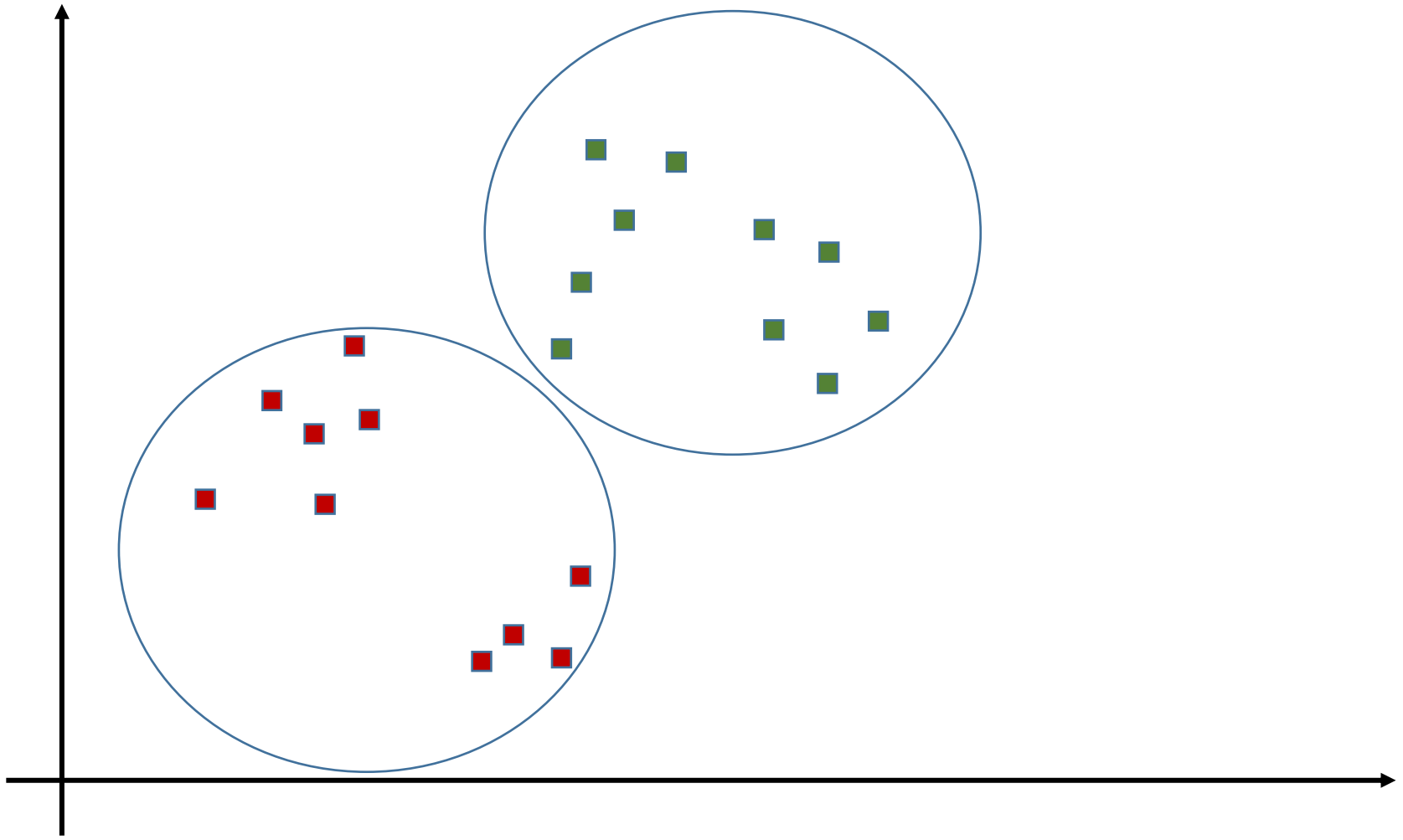
Re-clustering



Recompute clusters and repeat this till data points stop changing clusters

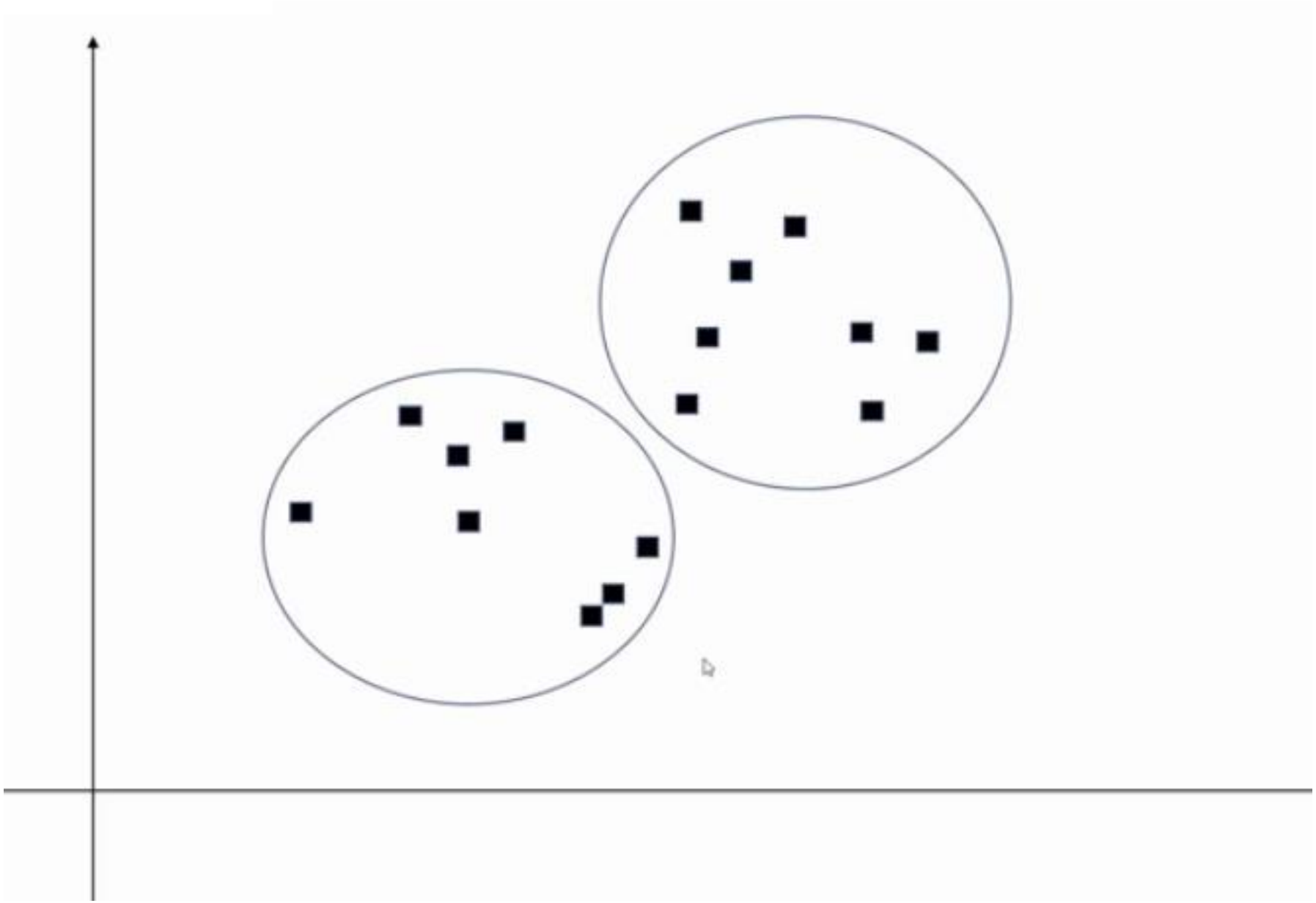
PRIME INTUIT
Finishing School

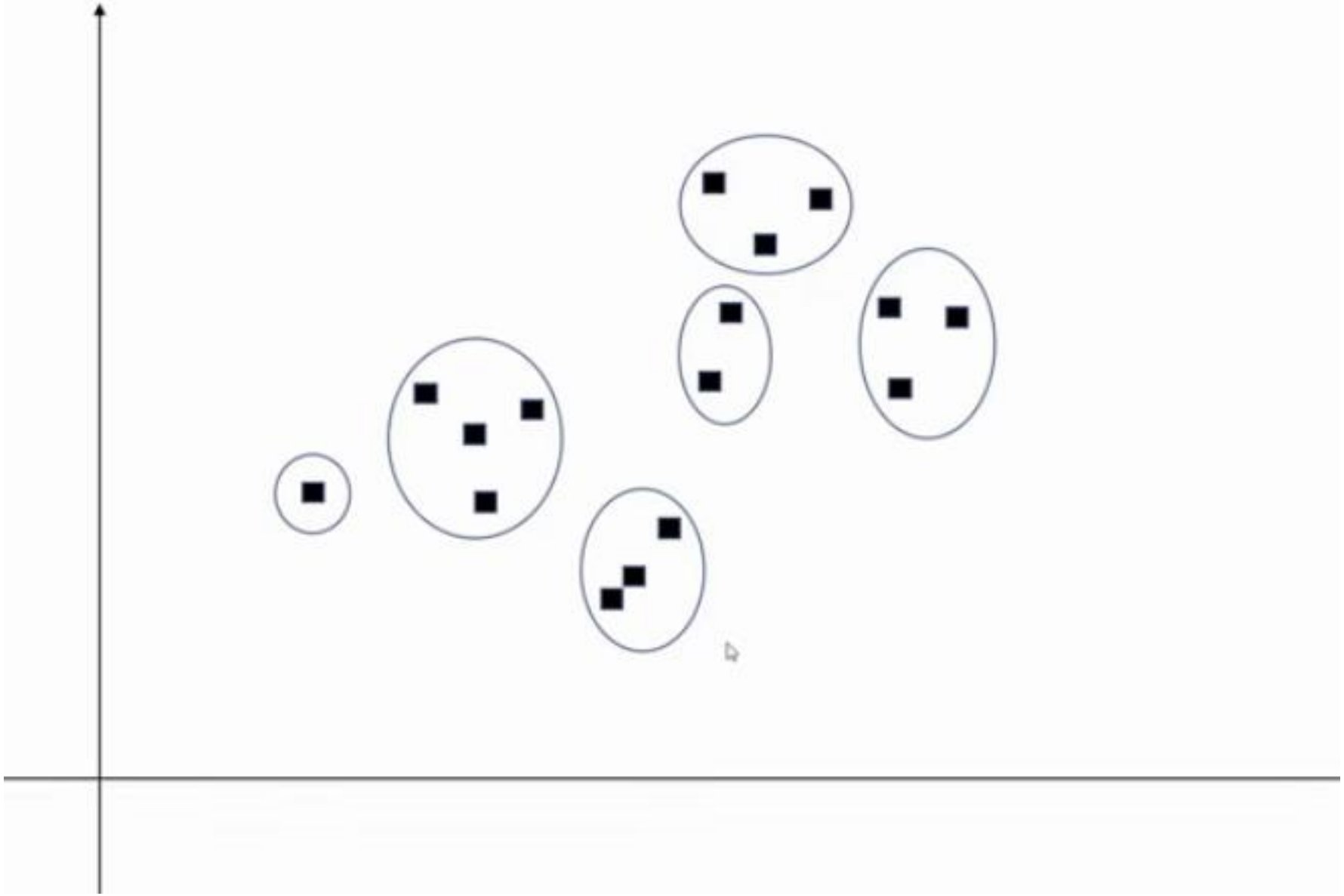
Re-clustering



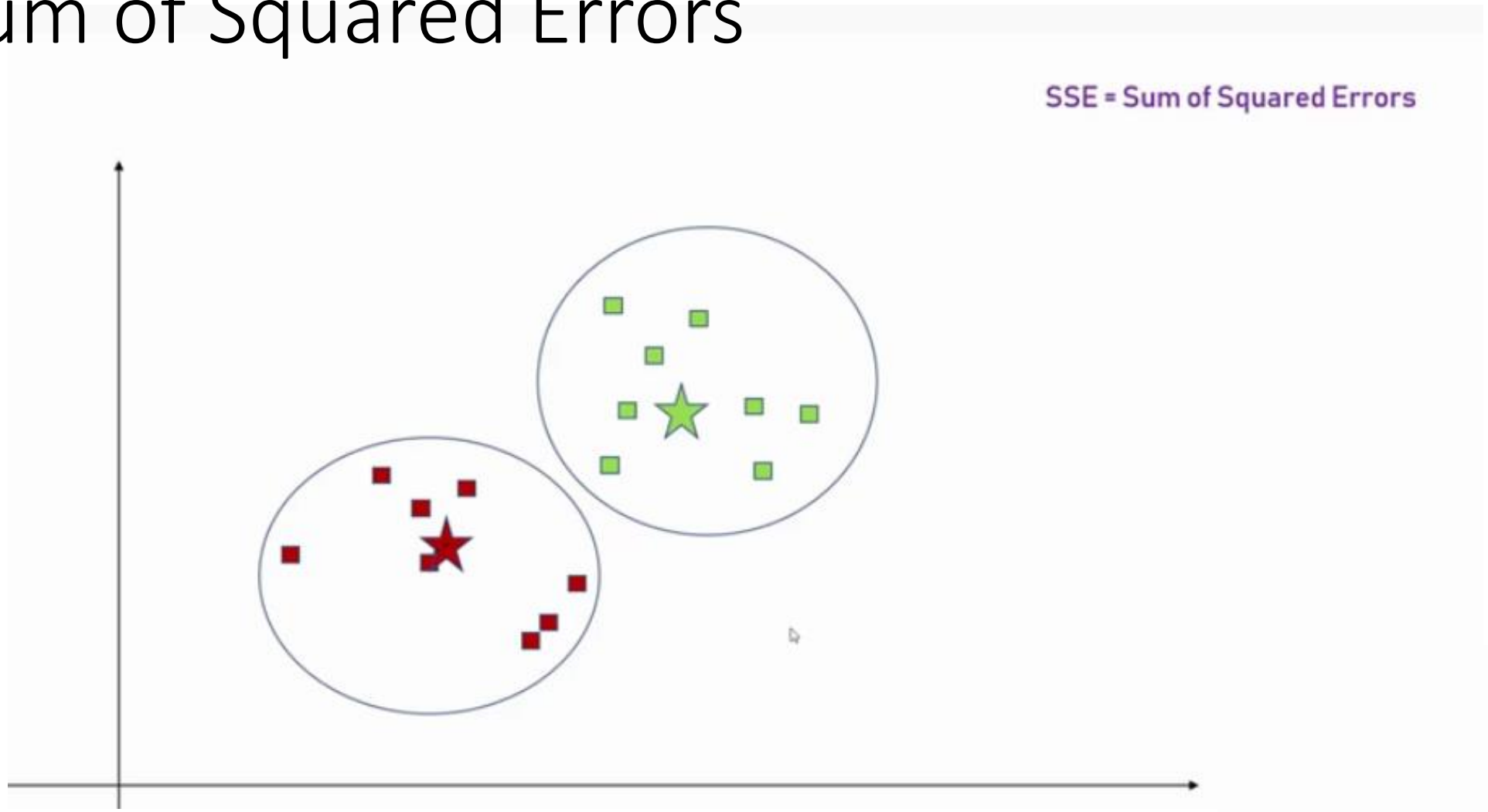
PRIME INTUIT
Finishing School

How to determine correct number of clusters (k)?





Sum of Squared Errors



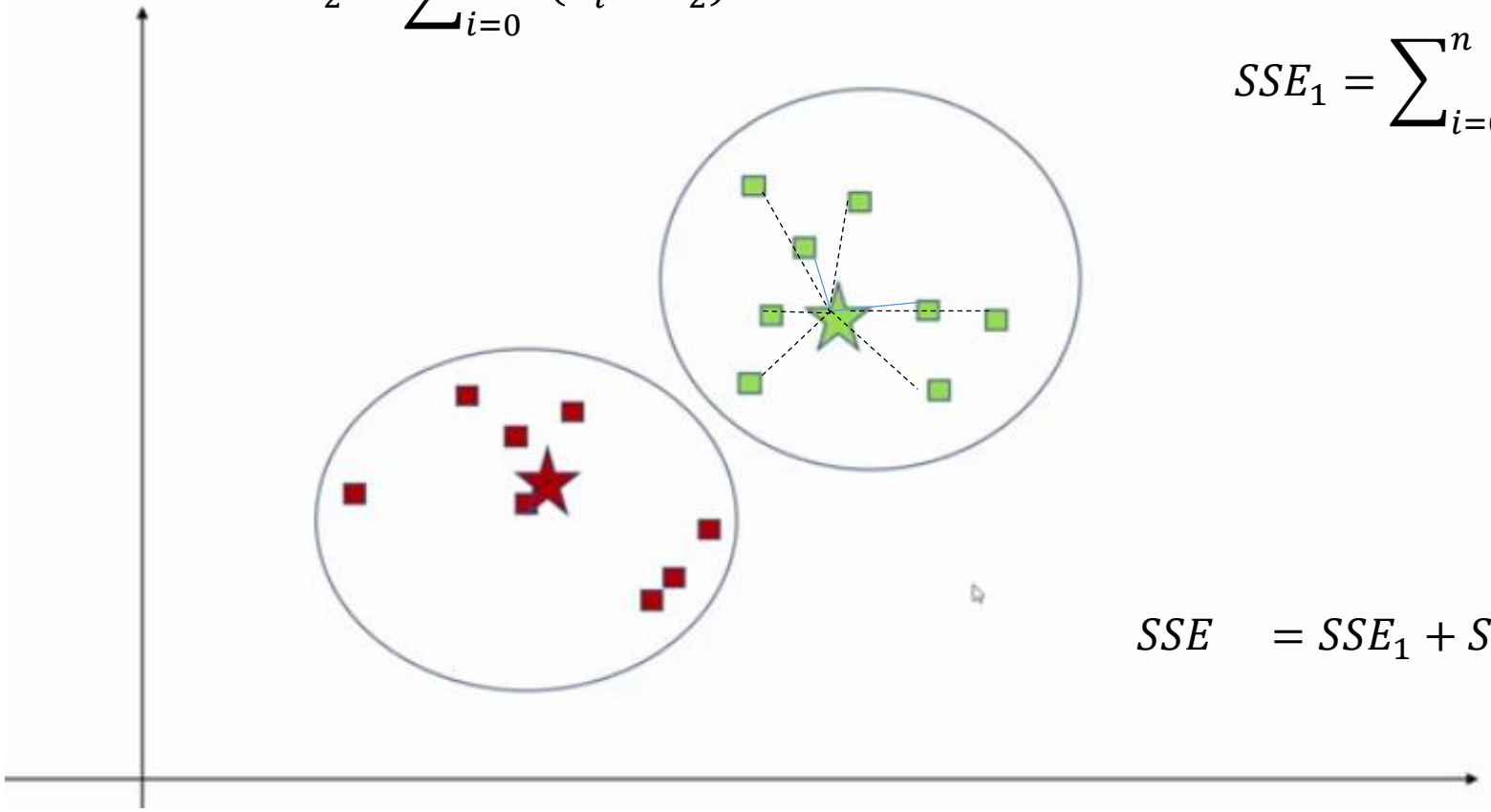
PRIME INTUIT
Finishing School

Sum of Squared Errors

SSE = Sum of Squared Errors

$$SSE_2 = \sum_{i=0}^{m} \mathrm{d}(X_i - C_2)^2$$

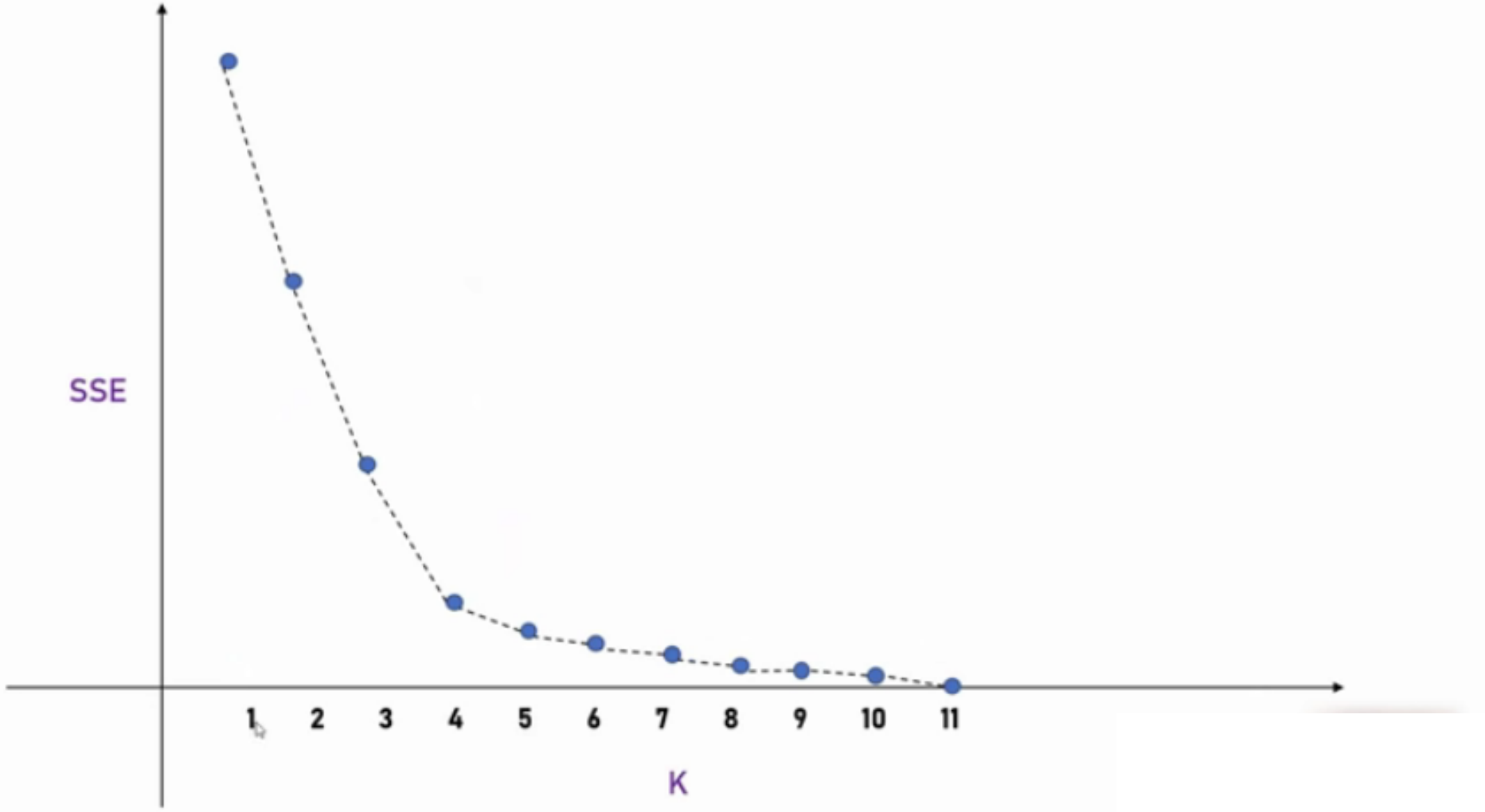
$$SSE_1 = \sum_{i=0}^{n} \mathrm{d}(X_i - C_1)^2$$



$$SSE = SSE_1 + SSE_2 + SSE_3 + \ldots\ldots\ldots + SSE_{\mathrm{K}}$$

PRIME INTUIT
Finishing School

Elbow Technique



Code

```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import MinMaxScaler
%matplotlib inline
```

Code

```
df = pd.read_csv("income.csv")
df.head()

plt.scatter(df['Age'], df['Income($)'])

km = KMeans(n_clusters=3)
Km

y_predict = km.fit_predict(df[['Age', 'Income($)']])
y_predict
```

Code

```
df['cluster'] = y_predict
df.tail()

df1 = df[df.cluster==0]
df2 = df[df.cluster==1]
df3 = df[df.cluster==2]

plt.scatter(df1['Age'], df1['Income($)'], color ='green')
plt.scatter(df2['Age'], df2['Income($)'], color ='red')
plt.scatter(df3['Age'], df3['Income($)'], color ='black')

plt.xlabel('Age')
plt.ylabel('Income($)')
plt.legend()
```

Code

```
scaler = MinMaxScaler()
features = ['Income($)']
scaler.fit(df[features])
df['Income($)'] = scaler.transform(df[features])

age = ['Age']
scaler.fit(df[age])
df.Age = scaler.transform(df[age])
df
```

Code

```
km = KMeans(n_clusters=3)
y_predict = km.fit_predict(df[['Age', 'Income($)']])
y_predict



df['cluster'] = y_predict
Df

km.cluster_centers_  # second run
```

Code

```
df1 = df[df.cluster==0]
df2 = df[df.cluster==1]
df3 = df[df.cluster==2]


plt.scatter(df1['Age'], df1['Income($)'], color ='green')
plt.scatter(df2['Age'], df2['Income($)'], color ='red')
plt.scatter(df3['Age'], df3['Income($)'], color ='black')
plt.scatter(km.cluster_centers_[:,0], km.cluster_centers_[:,1], color = 'purple', marker='*',
label='centroid') # second run
plt.xlabel('Age')
plt.ylabel('Income($)')
plt.legend()
```

Code

```
k_rng = range(1,10)
sse = []
for k in k_rng:
    km = KMeans(n_clusters = k)
    km.fit(df[['Age', 'Income($)']])
    sse.append(km.inertia_)
sse
```

Code

```
plt.xlabel('x')
plt.ylabel('sum of squared errors')
plt.plot(k_rng,sse)
```

https://www.kaggle.com/patelprashant/employee-attrition

PRIME INTUIT
Finishing School

Unsupervised Machine Learning (K-Means)

The k-means clustering method is an unsupervised machine learning technique used to identify clusters of data objects in a dataset. There are many different types of clustering methods, but k-means is one of the oldest and most approachable.

- **What k-means clustering is**
- **When to use k-means clustering to analyze your data**
- **How to implement k-means clustering in Python with scikit-learn**
- **How to select a meaningful number of clusters**

PRIME INTUIT
Finishing School

Unsupervised Machine Learning (K-Means)

What Is Clustering?

Clustering is a set of techniques used to partition data into groups, or clusters. Clusters are loosely defined as groups of data objects that are more similar to other objects in their cluster than they are to data objects in other clusters.

Clustering helps identify two qualities of data:

- Meaningfulness
- Usefulness

Unsupervised Machine Learning (K-Means)

What Is Clustering?

Meaningful clusters expand domain knowledge. For example, in the medical field, researchers applied clustering to gene expression experiments. The clustering results identified groups of patients who respond differently to medical treatments.

Useful clusters, on the other hand, serve as an intermediate step in a data pipeline. For example, businesses use clustering for customer segmentation. The clustering results segment customers into groups with similar purchase histories, which businesses can then use to create targeted advertising campaigns

Unsupervised Machine Learning (K-Means)

Overview of Clustering Techniques ?

Important factors that effects the selection of cluster algorithm include:

- Characteristics of the clusters
- Features of the dataset
- Number of outliers
- Number of data objects, etc

Three popular categories of clustering algorithms:

- Partitional clustering
- Hierarchical clustering
- Density-based clustering

PRIME INTUIT
Finishing School

Unsupervised Machine Learning (K-Means)

Overview of Clustering Techniques ?

Non deterministic process

Partitional Clustering

Partitional clustering divides data objects into nonoverlapping groups. In other words, no object can be a member of more than one cluster, and every cluster must have at least one object.

These techniques require the user to specify the number of clusters, indicated by the variable ***k***. Many partitional clustering algorithms work through an iterative process to assign subsets of data points into *k* clusters. Two examples of partitional clustering algorithms are ***k*-means and *k*-medoids.**

PRIME INTUIT
Finishing School

Unsupervised Machine Learning (K-Means)

Overview of Clustering Techniques ?

Partitional Clustering

Advantages:
They work well when clusters have a **spherical shape**.
They're **scalable** with respect to algorithm complexity.

Disadvantages:
They're not well suited for clusters with **complex shapes** and different sizes.
They break down when used with clusters of different **densities**.

PRIME INTUIT
Finishing School

Unsupervised Machine Learning (K-Means)

Overview of Clustering Techniques ?

deterministic process

Hierarchical Clustering

Hierarchical clustering determines cluster assignments by building a hierarchy. This is implemented by either a bottom-up or a top-down approach:
These methods produce a tree-based hierarchy of points called a **dendrogram**. Similar to partitional clustering, in hierarchical clustering the number of clusters (*k*) is often predetermined by the user. Clusters are assigned by cutting the dendrogram at a specified depth that results in *k* groups of smaller dendrograms.

Agglomerative clustering is the bottom-up approach. It merges the two points that are the most similar until all points have been merged into a single cluster.
Divisive clustering is the top-down approach. It starts with all points as one cluster and splits the least similar clusters at each step until only single data points remain.

PRIME INTUIT
Finishing School

Unsupervised Machine Learning (K-Means)

Overview of Clustering Techniques ?

Hierarchical Clustering

Advantages

They often reveal the finer details about the **relationships** between data objects.
They provide an **interpretable dendrogram**.

Disadvantages

They're **computationally expensive** with respect to algorithm complexity.
They're sensitive to **noise** and **outliers**.

PRIME INTUIT
Finishing School

Unsupervised Machine Learning (K-Means)

Overview of Clustering Techniques ?

deterministic process

Density-based clustering

Density-based clustering determines cluster assignments based on the density of data points in a region. Clusters are assigned where there are high densities of data points separated by low-density regions.

Unlike the other clustering categories, this approach doesn't require the user to specify the number of clusters. Instead, there is a distance-based parameter that acts as a tunable threshold. This threshold determines how close points must be to be considered a cluster member.

Examples:
DBSCAN: Density-Based Spatial Clustering of Applications with Noise
OPTICS: Ordering Points To Identify the Clustering Structure

PRIME INTUIT
Finishing School

Unsupervised Machine Learning (K-Means)

Overview of Clustering Techniques ?

Density-based clustering

Advantages

- They excel at identifying clusters of **nonspherical shapes**.
- They're resistant to **outliers**.

Disadvantages

- They aren't well suited for clustering in **high-dimensional spaces**.
- They have trouble identifying clusters of **varying densities**.

Unsupervised Machine Learning (K-Means)

Understanding the K-Means Algorithm

Non deterministic process

Understanding the K-Means Algorithm

- Randomly select *k* centroids, where *k* is equal to the number of clusters you choose. (**Centroids** are data points representing the center of a cluster.)
- The main element of the algorithm works by a two-step process called **expectation-maximization**.
  - The **expectation** step assigns each data point to its nearest centroid.
  - The **maximization** step computes the mean of all the points for each cluster and sets the new centroid.

PRIME INTUIT
Finishing School

Unsupervised Machine Learning (K-Means)

Understanding the K-Means Algorithm

Non deterministic process

Algorithm 1: K – means algorithm

- Specify the number k of clusters to assign
- Randomly initialize k centroids
- **Repeat:**
  - **Expectation:** Assign each point to its closest centroid.
  - **Maximization:** Compute the new centroid mean for each cluster
- **Untill:** The centroid positions do not change

The quality of the cluster assignments is determined by computing the **sum of the squared error (SSE)** after the centroids **converge**, or match the previous iteration's assignment. The SSE is defined as the sum of the squared Euclidean distances of each point to its closest centroid. Since this is a measure of error, the objective of *k*-means is to try to minimize this value.

file:///D:/Prime%20Intuit/centroids_iterations.webp

PRIME INTUIT
Finishing School

Unsupervised Machine Learning (K-Means)

K-Means Clustering Code in Python

- conda install -c conda-forge kneed

```
(base) PS C:\Users\LENO
Collecting package meta
Solving environment: do
```

```
## Package Plan ##

  environment location: C:\Users\LENOVO\anaconda3

  added / updated specs:
    - kneed


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    conda-4.10.3               |   py38haa244fe_0         3.1 MB  conda-forge
    kneed-0.7.0                |     pyh9f0ad1d_0          12 KB  conda-forge
    python_abi-3.8             |           2_cp38           4 KB  conda-forge
    ------------------------------------------------------------
                                           Total:         3.1 MB

The following NEW packages will be INSTALLED:

  kneed              conda-forge/noarch::kneed-0.7.0-pyh9f0ad1d_0
  python_abi         conda-forge/win-64::python_abi-3.8-2_cp38

The following packages will be UPDATED:

  conda              pkgs/main::conda-4.10.1-py38haa95532_1 --> conda-forge::conda-4.10.3-py38haa244fe_0


Proceed ([y]/n)?
```

PRIME INTUIT
Finishing School

Unsupervised Machine Learning (K-Means)

Writing your K-Means Algorithm code

Import the modules needed for the code

```
import matplotlib.pyplot as plt
from kneed import KneeLocator
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler
```

PRIME INTUIT
Finishing School

Unsupervised Machine Learning (K-Means)

Writing your K-Means Algorithm code

Generate data using make_blobs() function

We can generate the data using make_blobs(), a convenience function in scikit-learn used to generate synthetic clusters. make_blobs() uses these parameters:

n_samples is the total number of samples to generate.
centers is the number of centers to generate.
cluster_std is the standard deviation.
make_blobs() returns a tuple of two values:

A two-dimensional NumPy array with the x- and y-values for each of the samples
A one-dimensional NumPy array containing the cluster labels for each sample

PRIME INTUIT
Finishing School

Unsupervised Machine Learning (K-Means)

Writing your K-Means Algorithm code

PRIME INTUIT
Finishing School

KNN Classifications

PRIME INTUIT
Finishing School

K Nearest Neighbor Classifier

Understanding the K Nearest Neighbour Algorithm

Purpose:

The purpose of the k Nearest Neighbor (KNN) algorithm is to use a database in which the data points are separated into several separate classes to predict the classification of a new sample point.

Algorithm:

The algorithm can be summarized as:

1. A positive integer k is specified, along with a new sample
2. We select the k entries in our database which are closest to the new sample
3. We find the most common classification of these entries
4. This is the classification we give to the new sample

PRIME INTUIT
Finishing School

K Nearest Neighbor Classifier

Understanding the K Nearest Neighbour Algorithm

Algorithm 1: K – means algorithm

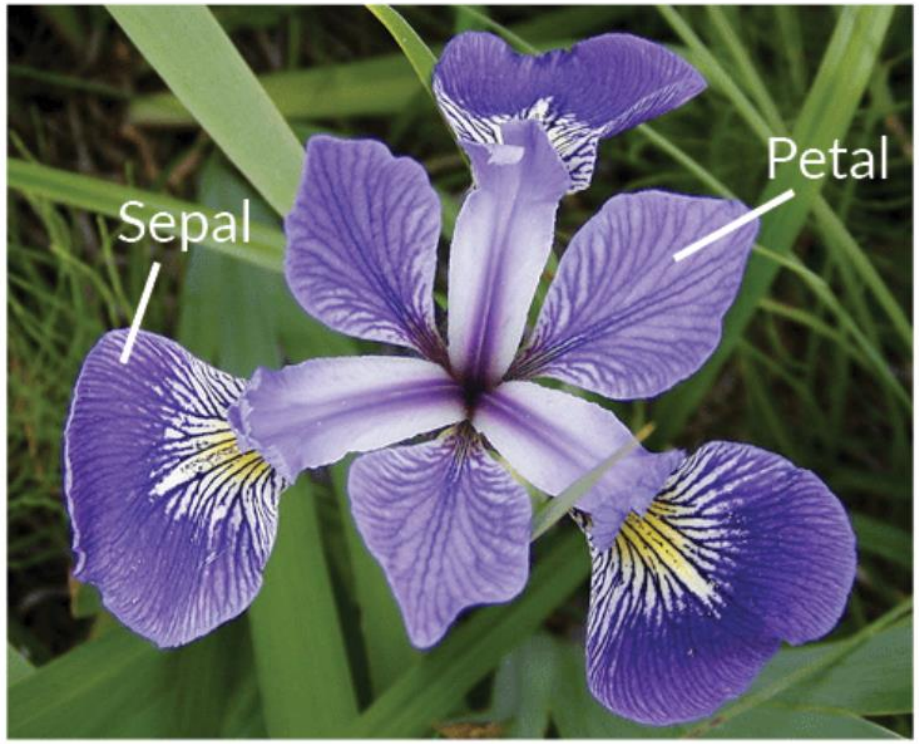
- Specify the number k of clusters to assign
- Randomly initialize k centroids
- **Repeat:**
  - **Expectation:** Assign each point to its closest centroid.
  - **Maximization:** Compute the new centroid mean for each cluster
- **Untill:** The centroid positions do not change

The quality of the cluster assignments is determined by computing the **sum of the squared error (SSE)** after the centroids **converge**, or match the previous iteration's assignment. The SSE is defined as the sum of the squared Euclidean distances of each point to its closest centroid. Since this is a measure of error, the objective of *k*-means is to try to minimize this value.

PRIME INTUIT
Finishing School

K Nearest Neighbor Classifier

Understanding the K Nearest Neighbour Algorithm



Iris Versicolor

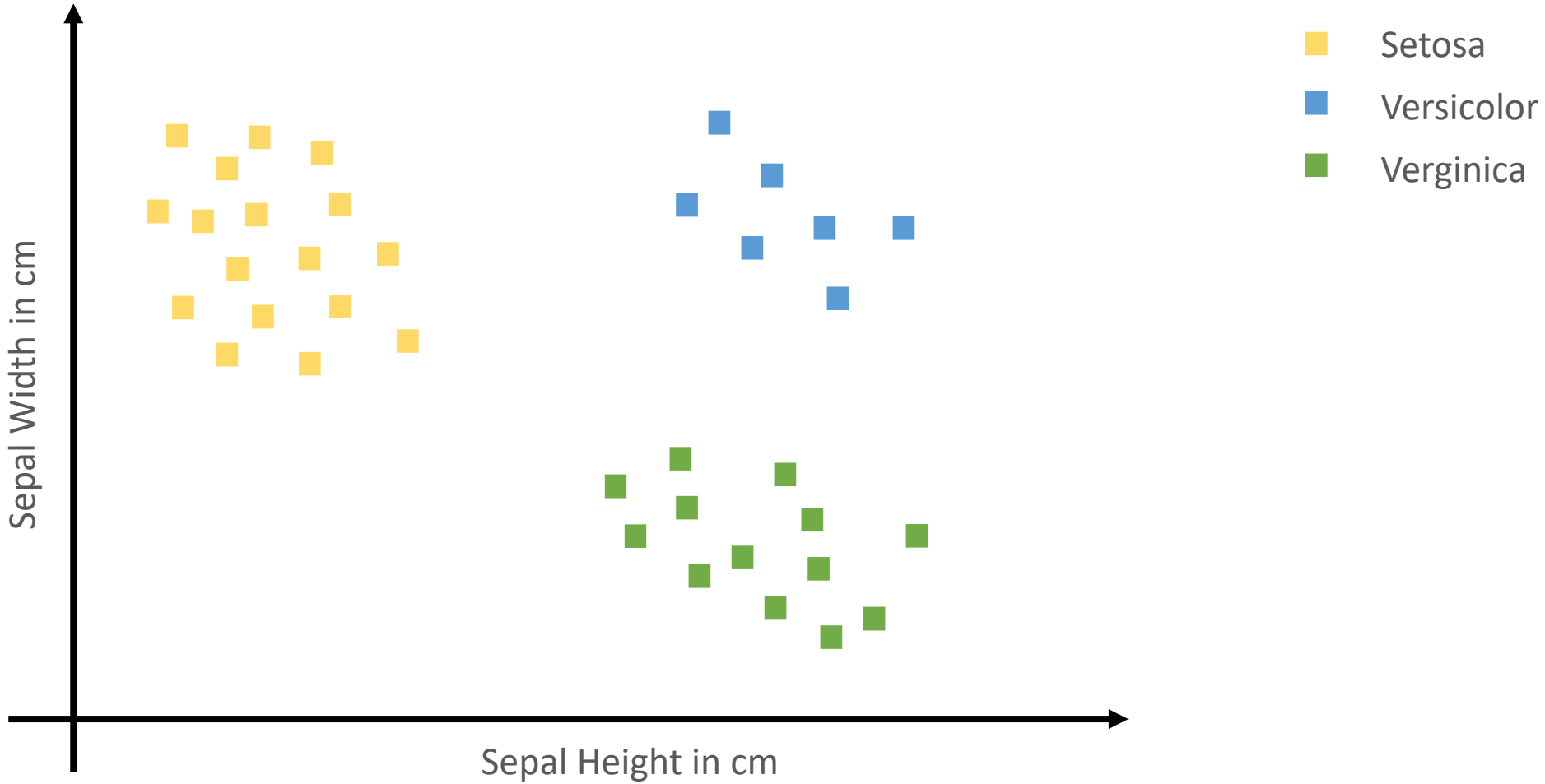
Iris Setosa

Iris Virginica

PRIME INTUIT
Finishing School

K Nearest Neighbor Classifier

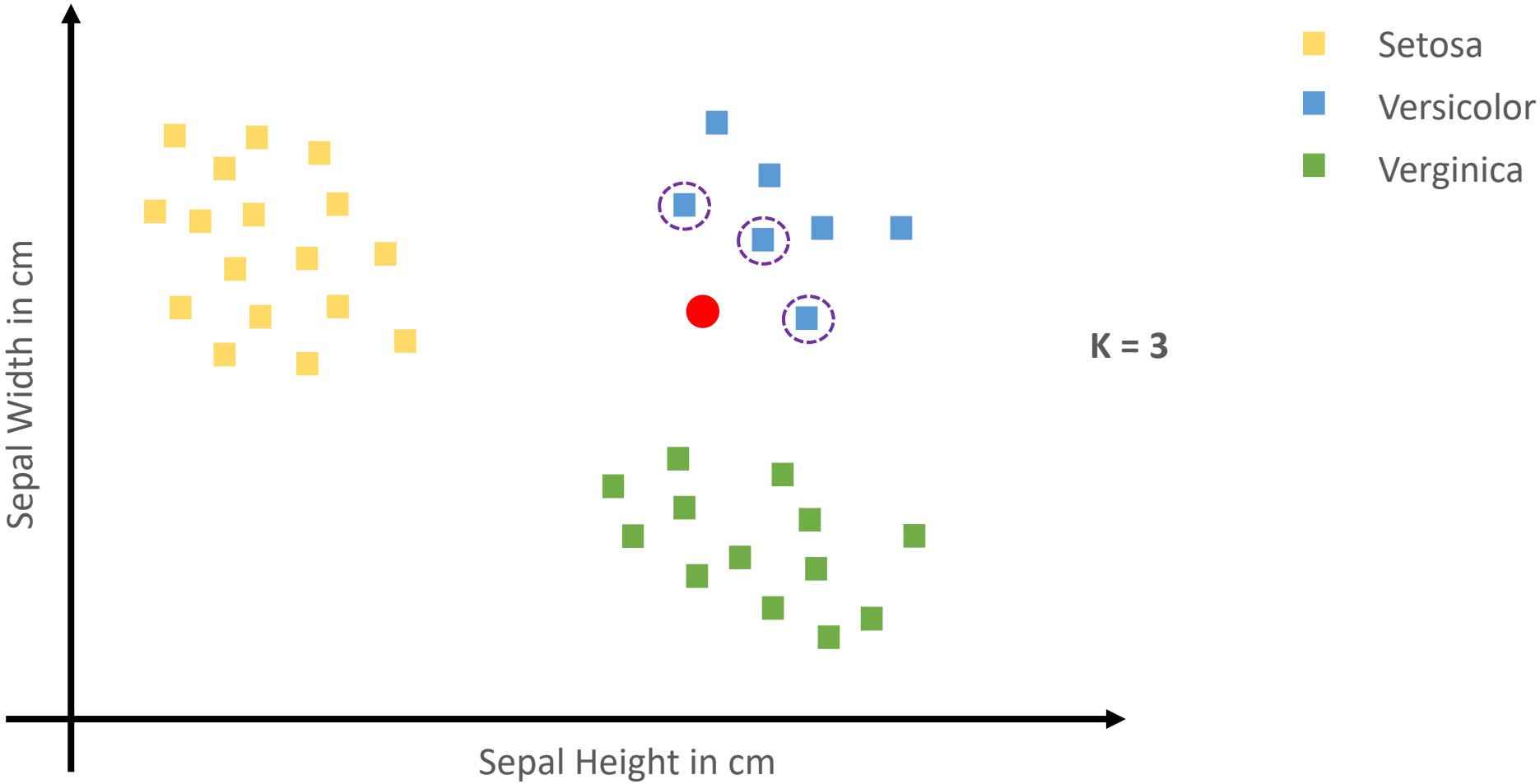
Understanding the K Nearest Neighbour Algorithm



PRIME INTUIT
Finishing School

K Nearest Neighbor Classifier

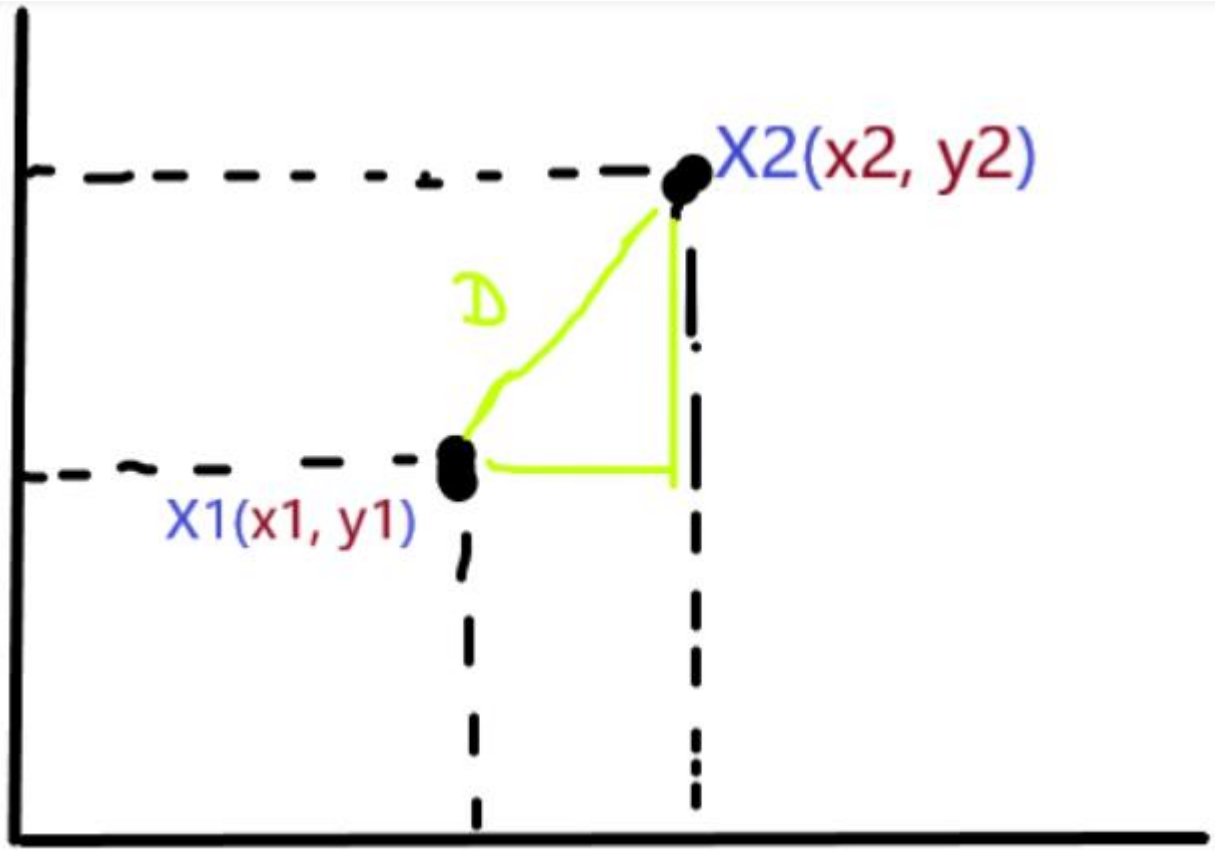
Understanding the K Nearest Neighbour Algorithm



PRIME INTUIT
Finishing School

K Nearest Neighbor Classifier

Understanding the K Nearest Neighbour Algorithm - Euclidean Distance



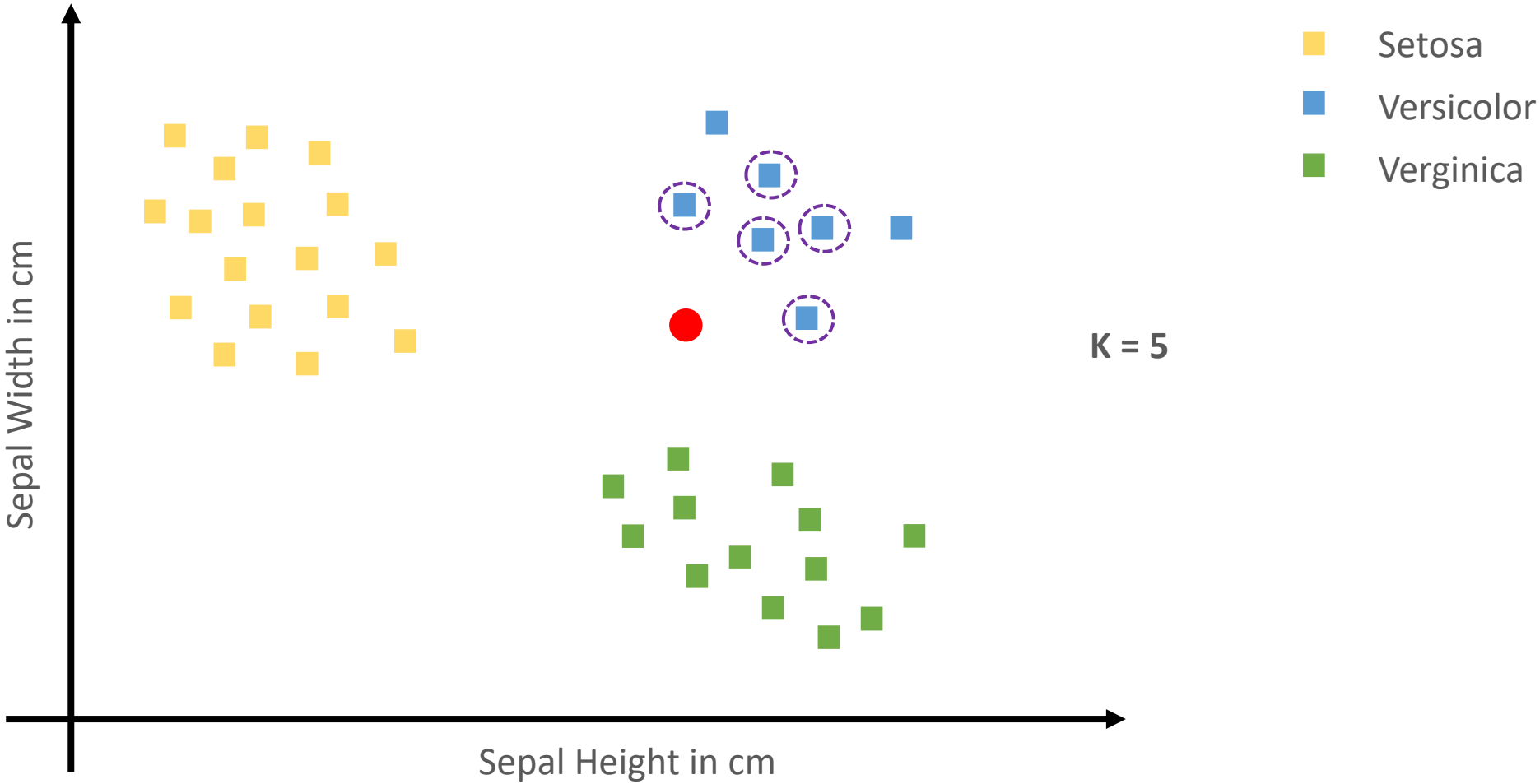
There are 2-Dim data X1 and X2 placed at certain coordinates in 2 dimensions, suppose X1 is at (x1,y1) coordinates and X2 is at (x2,y2) coordinates. We have 2-dim data so we considered F1 and F2 two features and D is considered as the shortest line from X1 and X2, If we find the distance between these data points that can be found by the Pythagoras theorem and can be written as:

d = sqrt{(x_2 - x_1)^2 + (y_2-y_1)^2}

PRIME INTUIT
Finishing School

K Nearest Neighbor Classifier

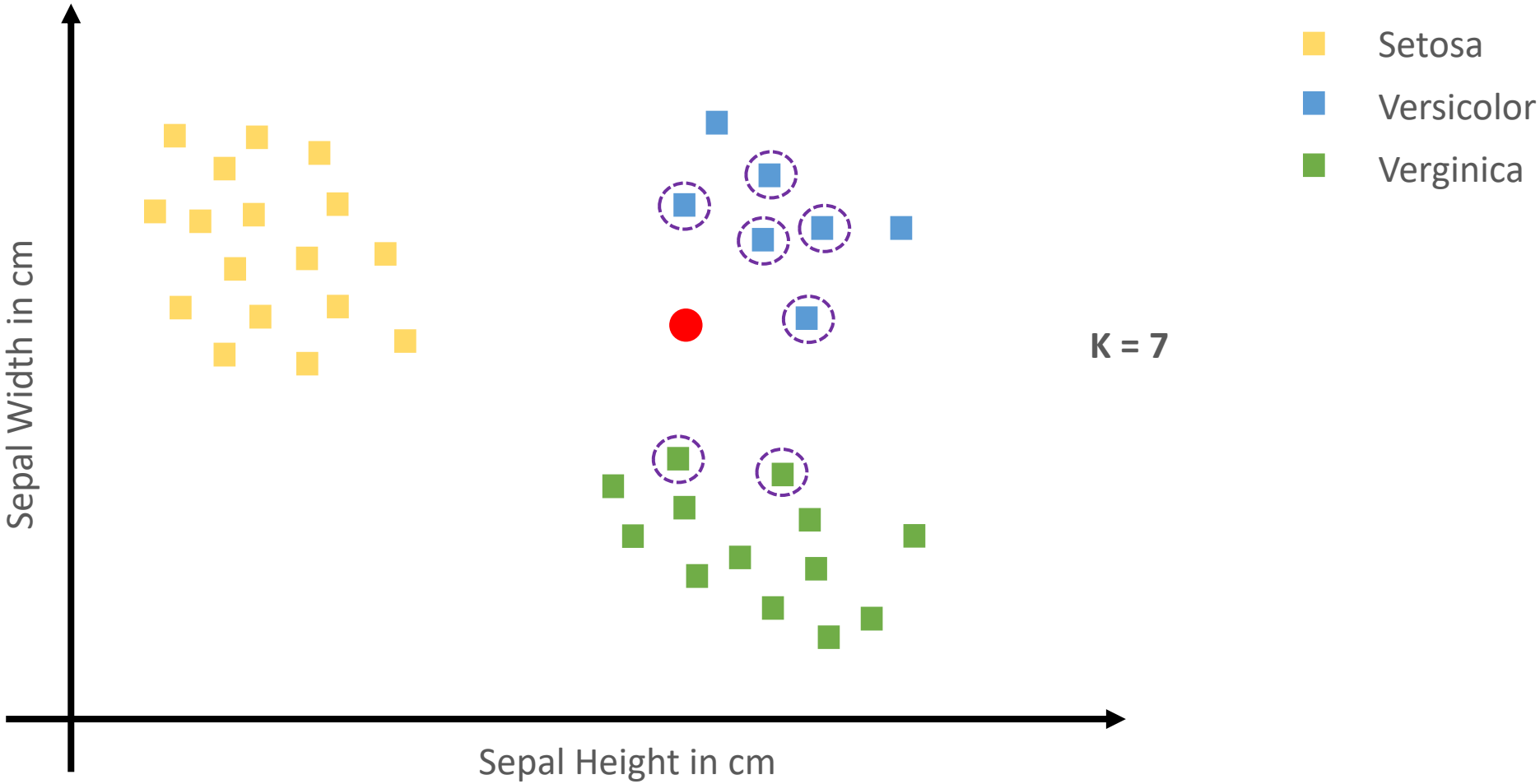
Understanding the K Nearest Neighbour Algorithm



PRIME INTUIT
Finishing School

K Nearest Neighbor Classifier

Understanding the K Nearest Neighbour Algorithm



PRIME INTUIT
Finishing School

K Nearest Neighbor Classifier

Working with Iris data set

```
import numpy as np
import pandas as pd
df = pd.read_csv('Iris.csv')
df.head()

df.shape
df.describe()
df.groupby('Species').size()

# Dividing data into features and labels
feature_columns = ['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm','PetalWidthCm']
X = df[feature_columns].values
y = df['Species'].values
```

PRIME INTUIT
Finishing School

K Nearest Neighbor Classifier

Working with Iris data set

```
# Change Species coloumn to numbers

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)



# Split data into training and test

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

PRIME INTUIT
Finishing School

K Nearest Neighbor Classifier

Working with Iris data set

```
# Data Visualisation

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

# Pair Plot

plt.figure()
sns.pairplot(df.drop("Id", axis=1), hue = "Species", size=3, markers=["o", "s", "D"])
plt.show()
```

PRIME INTUIT
Finishing School

K Nearest Neighbor Classifier

Working with Iris data set

```
# Box Plots


plt.figure()
df.drop("Id", axis=1).boxplot(by="Species", figsize=(15, 10))
plt.show()

# Fitting clasifier to the Training set
# Loading libraries
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.model_selection import cross_val_score
```

PRIME INTUIT
Finishing School

K Nearest Neighbor Classifier

Working with Iris data set

```
# Instantiate learning model (k = 3)
classifier = KNeighborsClassifier(n_neighbors=3)

# Fitting the model
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

print(y_pred)
```

PRIME INTUIT
Finishing School

K Nearest Neighbor Classifier

Working with Iris data set

```
# Define a function module to print results of ML Classifier Score

from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score, f1_score

def print_score(clf, x_train, y_train, x_test, y_test, train = True):
    if train:
        pred = clf.predict(x_train)
        print("Train Result:\ ================================================")
        print(f"accuracy score: {accuracy_score(y_train, pred):.4f}\n")
        print("Classification Data:")
        print(f"Precision: {precision_score(y_train, pred, average=None, zero_division=1)}\n")
        print (f"Recall Score: {recall_score(y_train, pred, average=None, zero_division=1)}\n")
        print(f"Confusion_matrix:\n {confusion_matrix(y_train, clf.predict(x_train))}\n")
    elif train == False:
        pred = clf.predict(x_test)
        print("Test Result:\ ================================================")
        print(f"accuracy score: {accuracy_score(y_test, pred)}\n")
        print("Classification Data:")
        print(f"Precision: {precision_score(y_test, pred, average=None, zero_division=1)}\n")
        print (f"Recall Score: {recall_score(y_test, pred, average=None, zero_division=1)}\n")
        print(f"Confusion_matrix:\n {confusion_matrix(y_test, clf.predict(x_test))}\n")
```

K Nearest Neighbor Classifier

Working with Iris data set

```
# Print results
print_score(classifier, X_train, y_train, X_test, y_test, train = True)
print_score(classifier, X_train, y_train, X_test, y_test, train = False)

# Using cross-validation for parameter tuning:

# creating list of K for KNN
k_list = list(range(1,50,2))

# creating list of cv scores
cv_scores = []

# perform 10-fold cross validation
for k in k_list:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, X_train, y_train, cv=10, scoring='accuracy')
    cv_scores.append(scores.mean())

    print(cv_scores)
```

PRIME INTUIT
Finishing School

K Nearest Neighbor Classifier

Working with Iris data set

```
# changing to misclassification error
MSE = [1 - x for x in cv_scores]

plt.figure()
plt.figure(figsize=(15,10))
plt.title('The optimal number of neighbors', fontsize=20, fontweight='bold')
plt.xlabel('Number of Neighbors K', fontsize=15)
plt.ylabel('Misclassification Error', fontsize=15)
sns.set_style("whitegrid")
plt.plot(k_list, MSE)

plt.show()

# finding best k
best_k = k_list[MSE.index(min(MSE))]
print("The optimal number of neighbors is %d." % best_k)
```

PRIME INTUIT
Finishing School

K Nearest Neighbor Classifier

Working with Iris data set

```
# Instantiate learning model (k = 9)
classifier = KNeighborsClassifier(n_neighbors=9)

# Fitting the model
classifier.fit(X_train, y_train)

print_score(classifier, X_train, y_train, X_test, y_test, train = True)
print_score(classifier, X_train, y_train, X_test, y_test, train = False)
```