Machine Learning with Python

Agenda

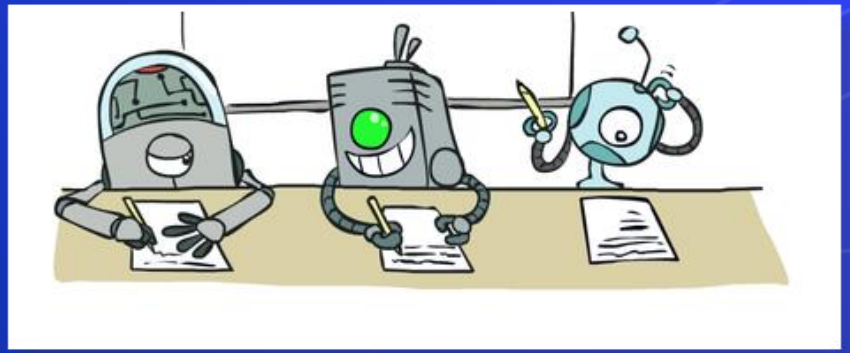
- What is Machine Learning
- Supervised Learning
  - Regression
  - Classification
- Unsupervised Learning
  - Clustering
  - Dimension Reduction
- Model Selection & Evaluation
  - Cross Validation
  - Hyperparameter Tuning
- Intro to Deep Learning

Agenda

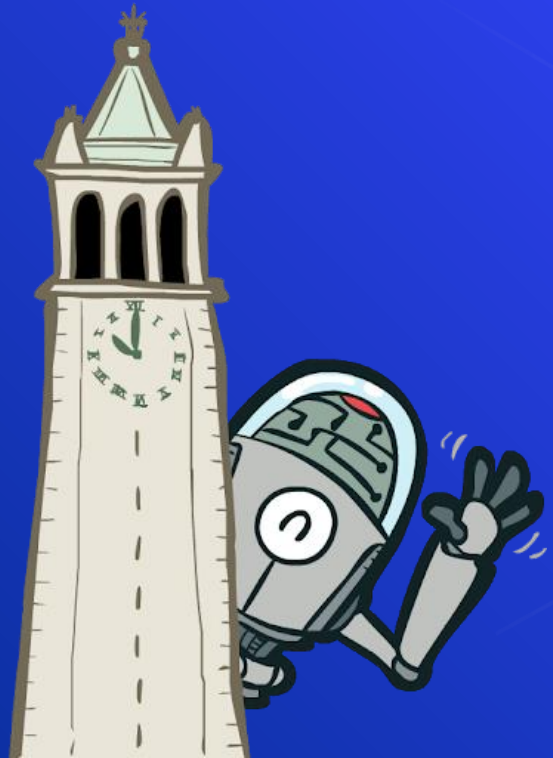
- What is Machine Learning
- Supervised Learning
  - Regression
  - Classification
- Unsupervised Learning
  - Clustering
  - Dimension Reduction
- Model Selection & Evaluation
  - Cross Validation
  - Hyperparameter Tuning
- Intro to Deep Learning

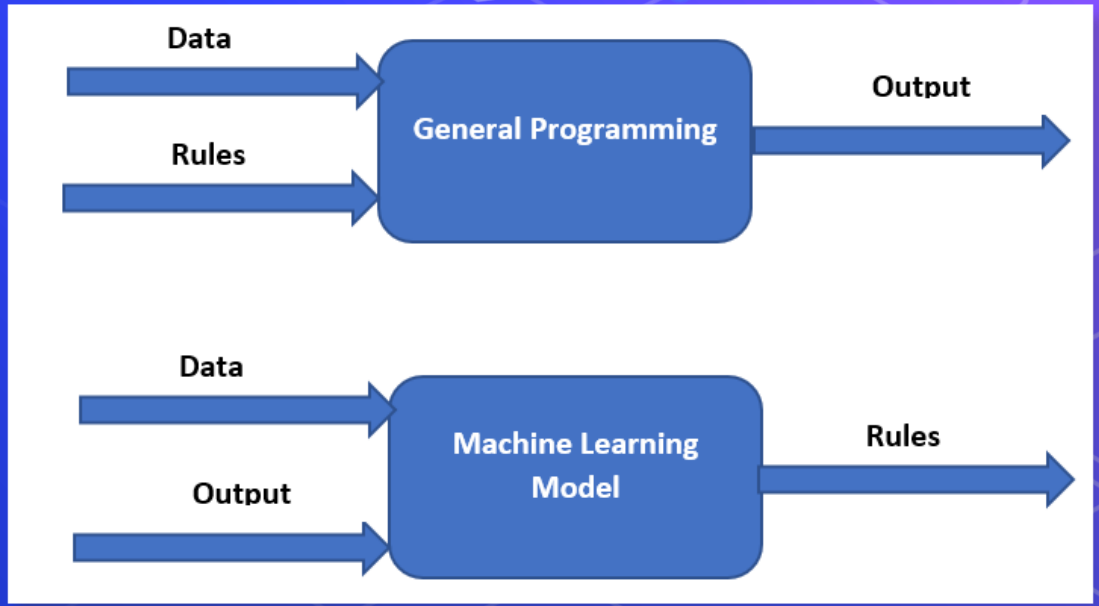
What is Machine Learning

Machine learning involves computers discovering how they can perform tasks without being explicitly programmed to do so. It involves computers learning from data provided so that they carry out certain tasks.

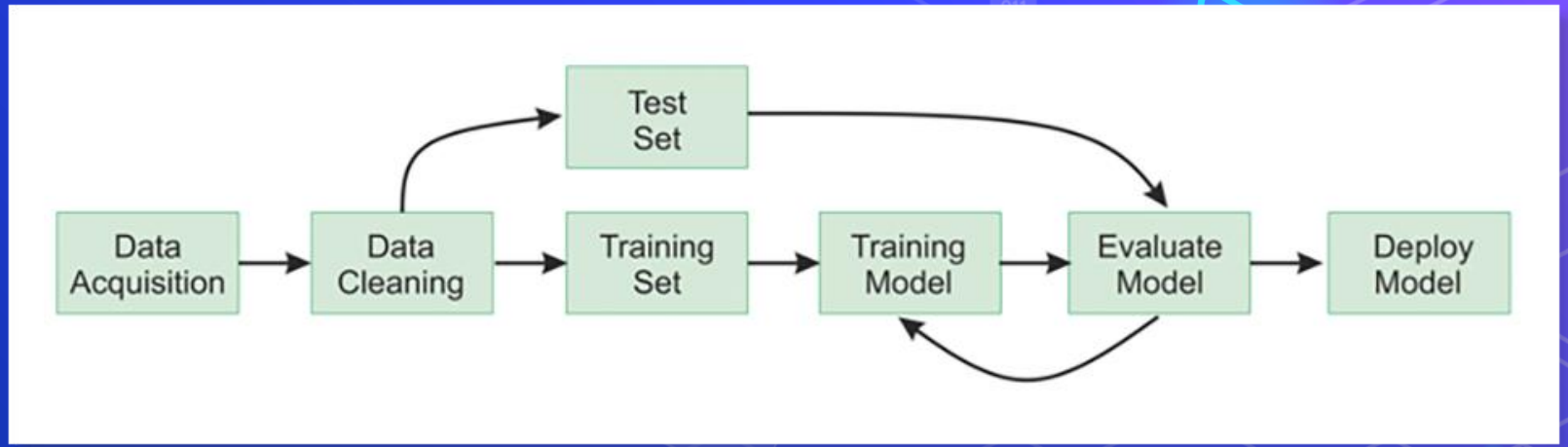


What is Machine Learning

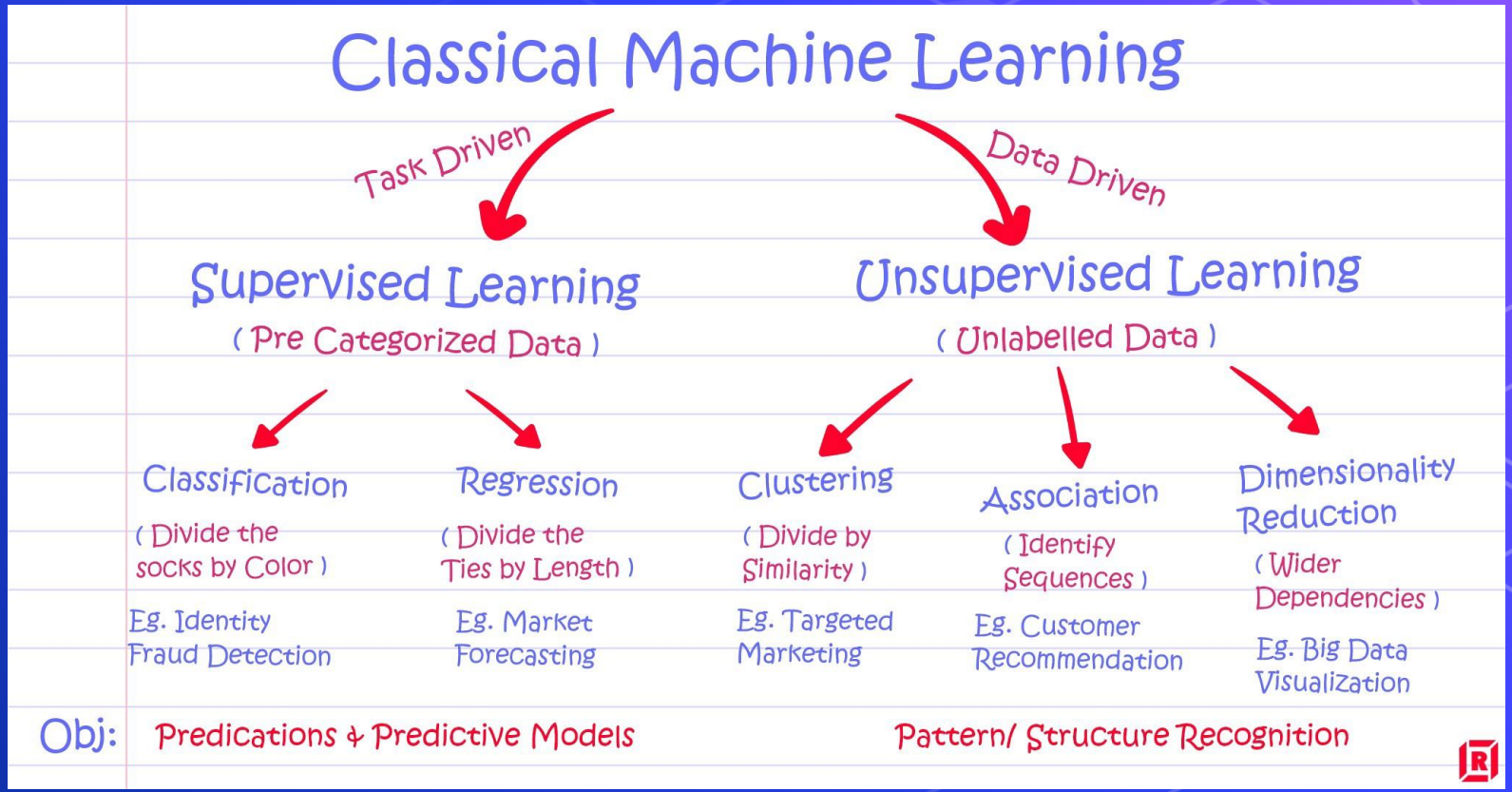




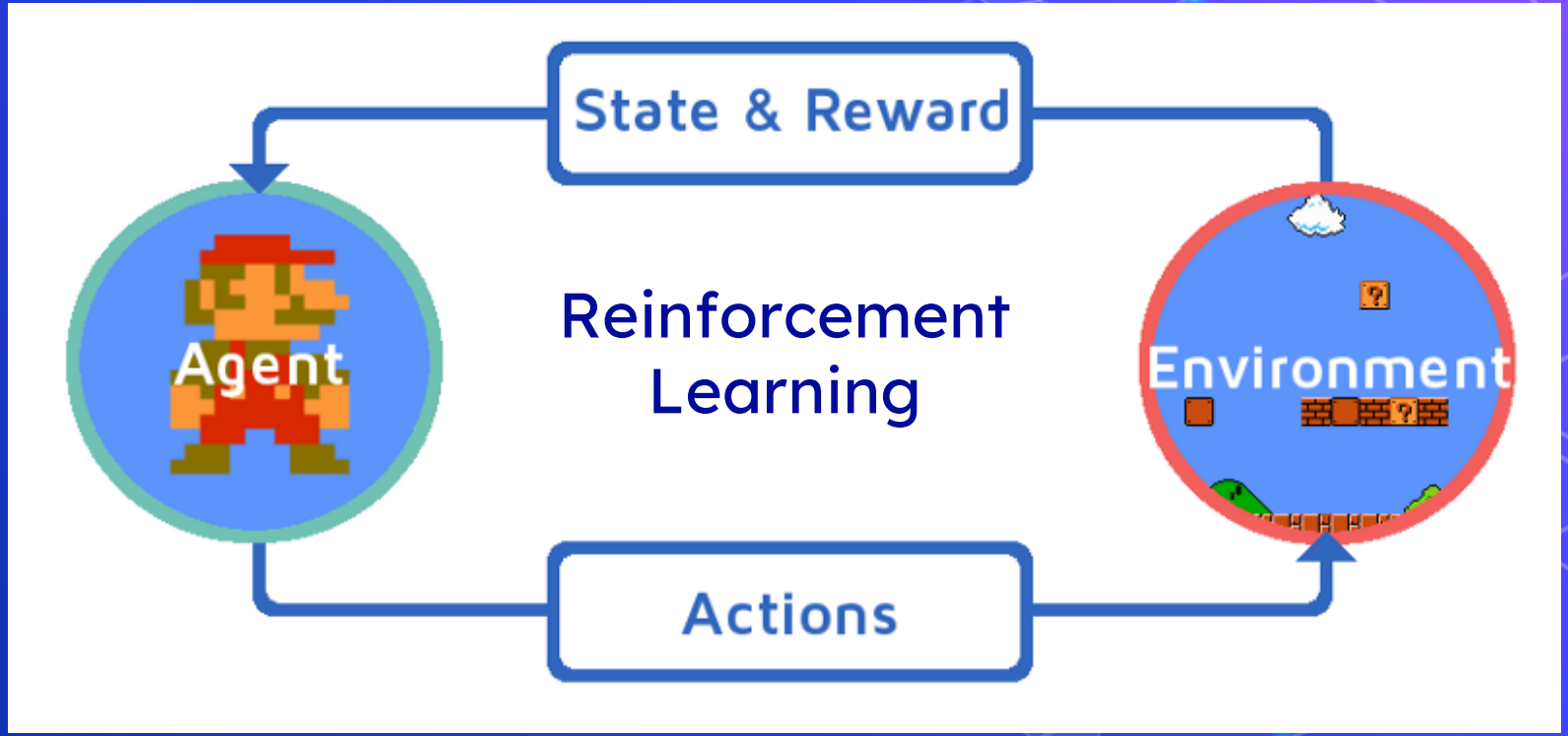
What is Machine Learning



What is Machine Learning



What is Machine Learning



What is Machine Learning

- Supervised learning : Task Driven (Classification, Regression)
- Unsupervised learning : Data Driven (Clustering)
- Reinforcement learning :
  - Close to human learning.
  - Algorithm learns a policy of how to act in a given environment.
  - Every action has some impact in the environment, and the environment provides rewards that guides the learning algorithm.

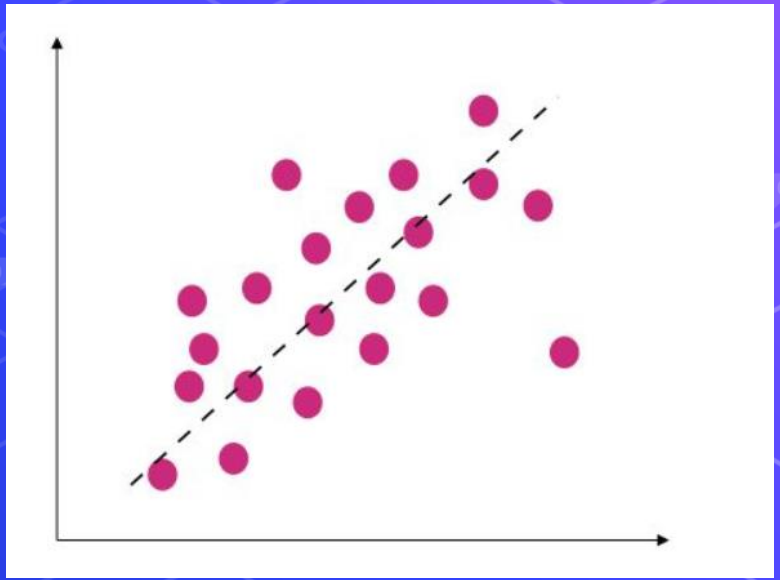
Agenda

- What is Machine Learning
- **Supervised Learning**
  - **Regression**
  - Classification
- Unsupervised Learning
  - Clustering
  - Dimension Reduction
- Model Selection & Evaluation
  - Cross Validation
  - Hyperparameter Tuning
- Intro to Deep Learning

Regression

Regression quantifies the relationship between one or more predictor variable(s) and one outcome variable.

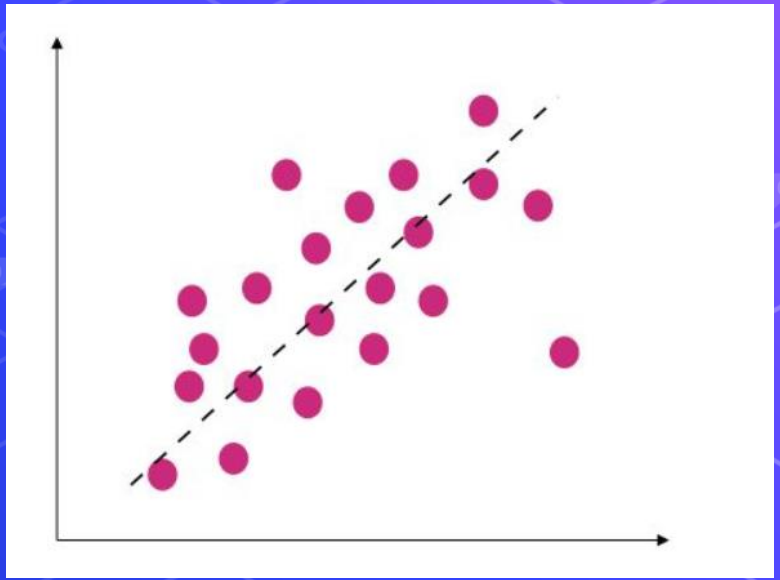
For example,
it can be used to quantify the relative impacts of age, gender, and diet (the predictors) on weight (the output or dependent).



- House prices based on size, locations,...
- Salary prediction based on experience
- Stock Market prediction
- Weather prediction
- ...

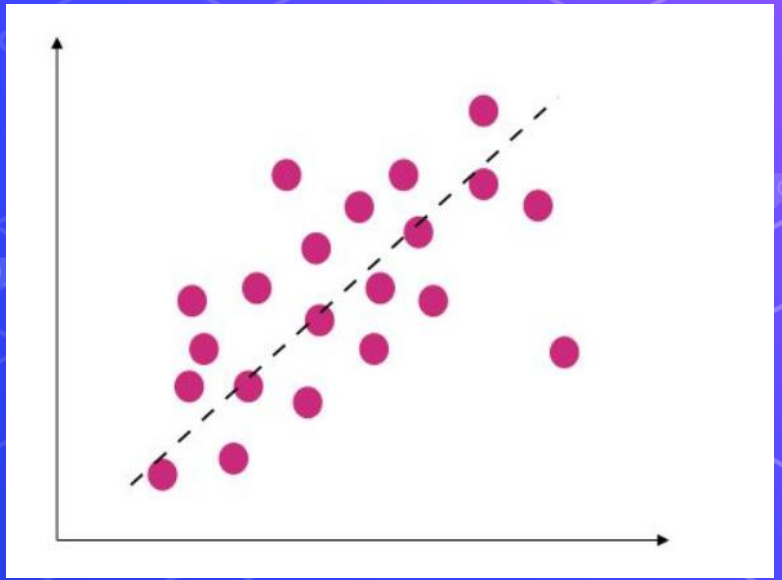
Regression

- Simple & Multiple Linear Regression
- Other Regression Algorithms
- Evaluating Model Performance



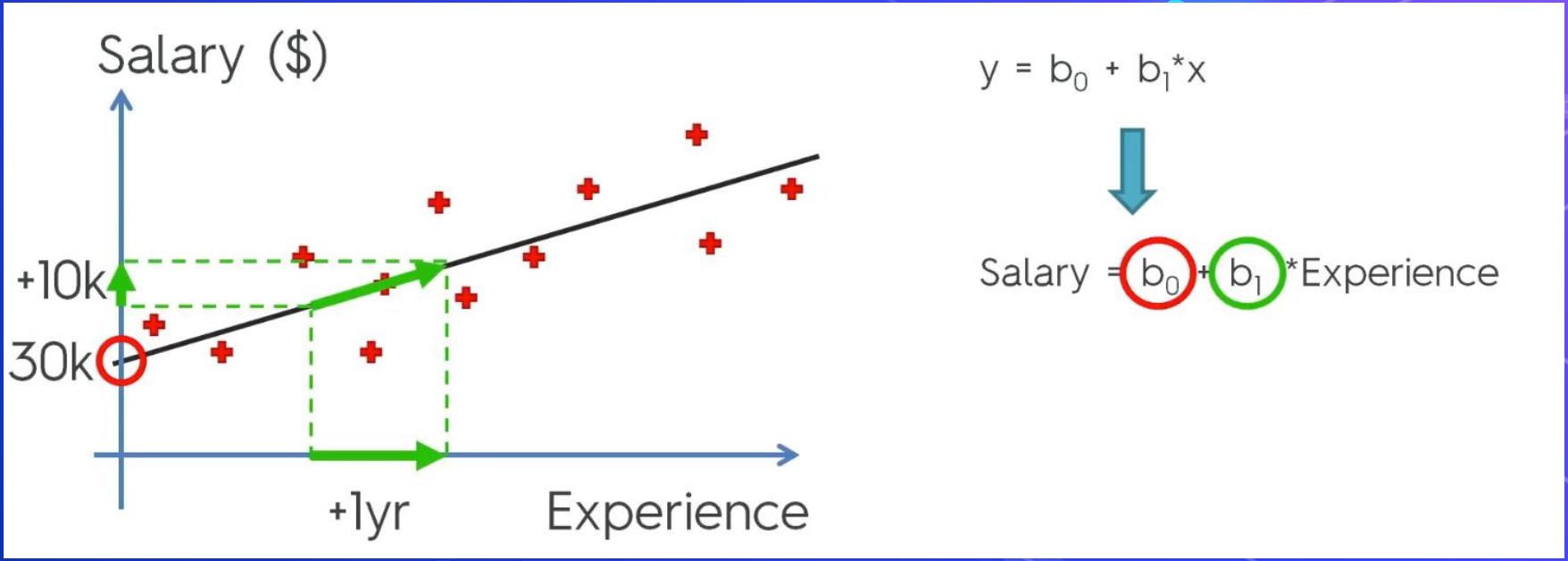
Regression

- Simple & Multiple Linear Regression
- Other Regression Algorithms
- Evaluating Model Performance

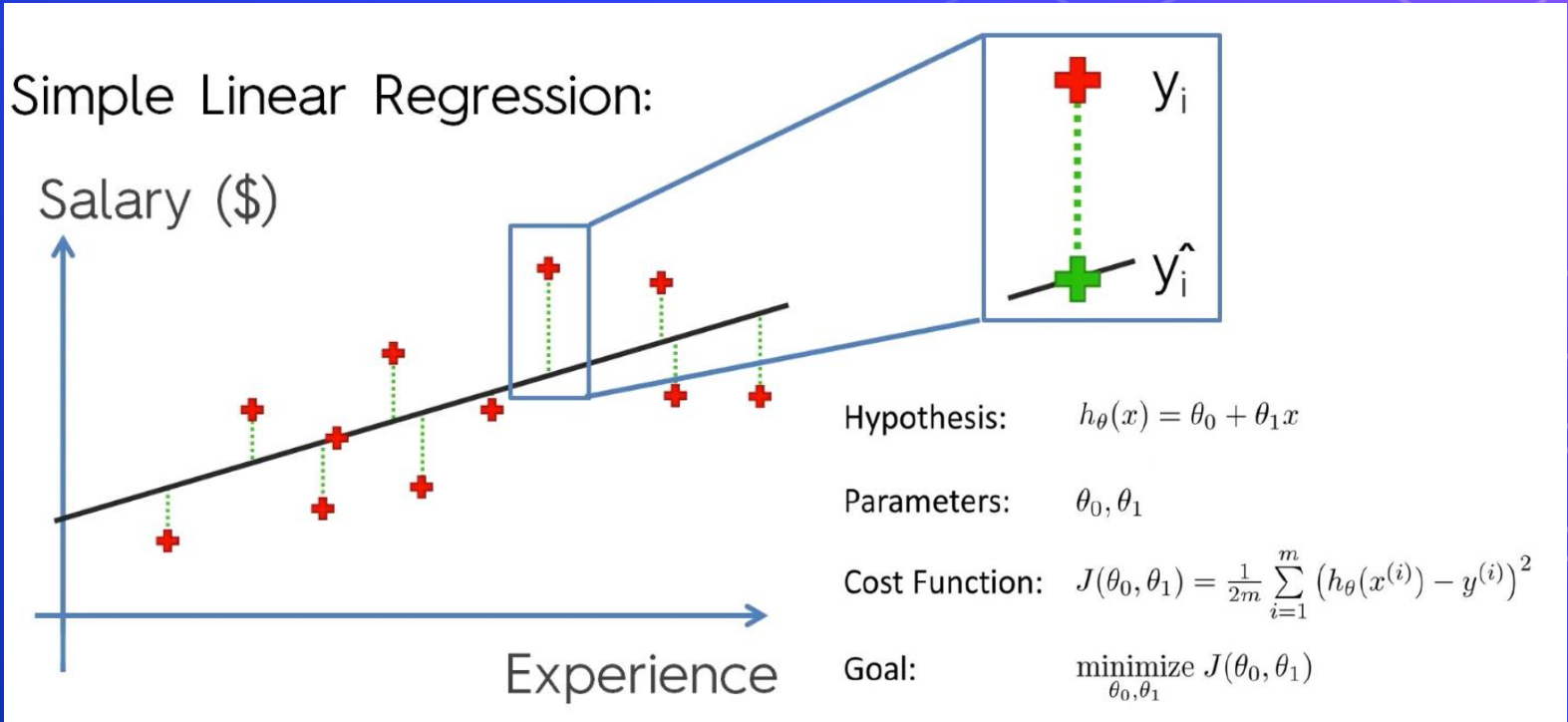


Simple Linear Regression (Equation of a straight line)

It also called ordinary least squares (OLS)



Simple Linear Regression (Calculate Cost Function)



Simple Linear Regression (Calculate Cost Function)

The goal is to minimize the sum of squares of residuals (y-y') is as small as possible.

Hypothesis:	$h_\theta(x) = \theta_0 + \theta_1 x$
Parameters:	θ_0, θ_1
Cost Function:	$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$
Goal:	$\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Simple Linear Regression (Example)

Theta0 = 5 , theta 1 = 2 Equation h(x) = 5 + 2x

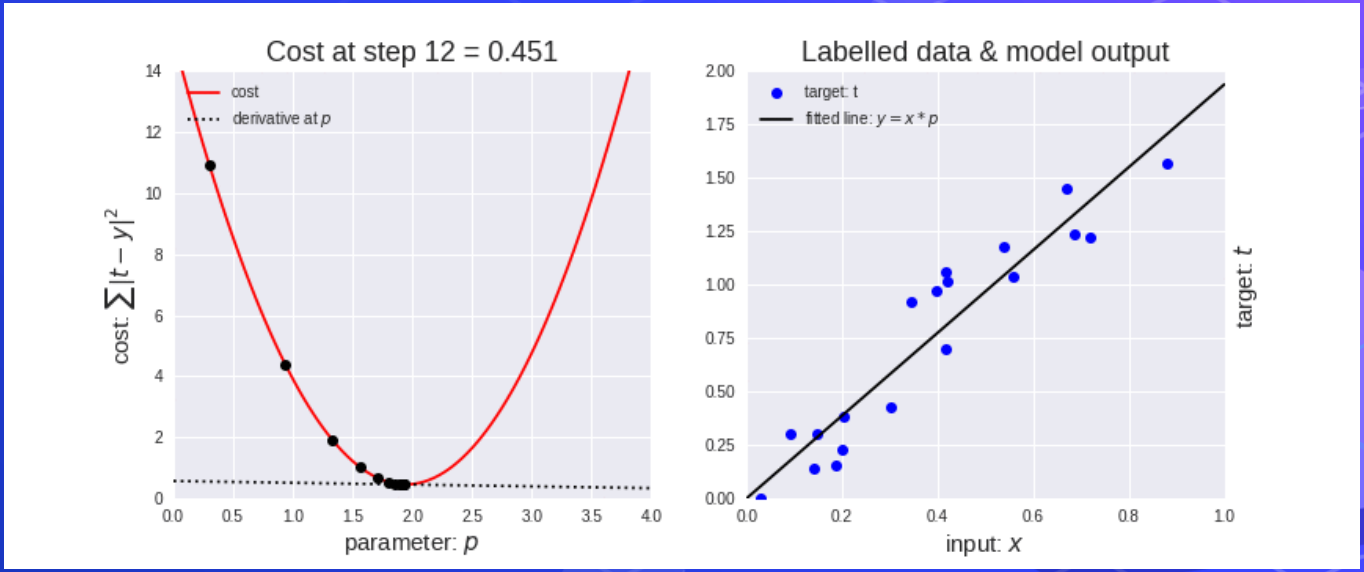
X	Y	h(x)	h(x) - y	$(h(x) - y)^2$
1	7	7	0	0
2	8	9	1	1
2	7	9	2	4
3	9	11	2	4
4	11	13	2	4
5	10	15	5	25
5	12	15	3	9

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

J = 1 / 14 (0+1+4+4+4+25+9)

J = 47/14 = 3.3

Simple Linear Regression (Minimize cost using Gradient Descent)



repeat until convergence:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} ((h_\theta(x_i) - y_i) x_i)$$

Simple Linear Regression (Minimize cost using Gradient Descent)

Cost Function

$$J(\Theta_0, \Theta_1) = \frac{1}{2m}\sum_{i=1}^{m}[h_\Theta(x_i) - y_i]^2$$

$h_\Theta(x_i)$: Predicted Value; y_i: True Value

Gradient Descent

$$\Theta_j = \Theta_j - \alpha\frac{\partial}{\partial\Theta_j}J(\Theta_0, \Theta_1)$$

α: Learning Rate

Now,

$$\frac{\partial}{\partial\Theta}J_\Theta = \frac{\partial}{\partial\Theta}\frac{1}{2m}\sum_{i=1}^{m}[h_\Theta(x_i) - y]^2$$

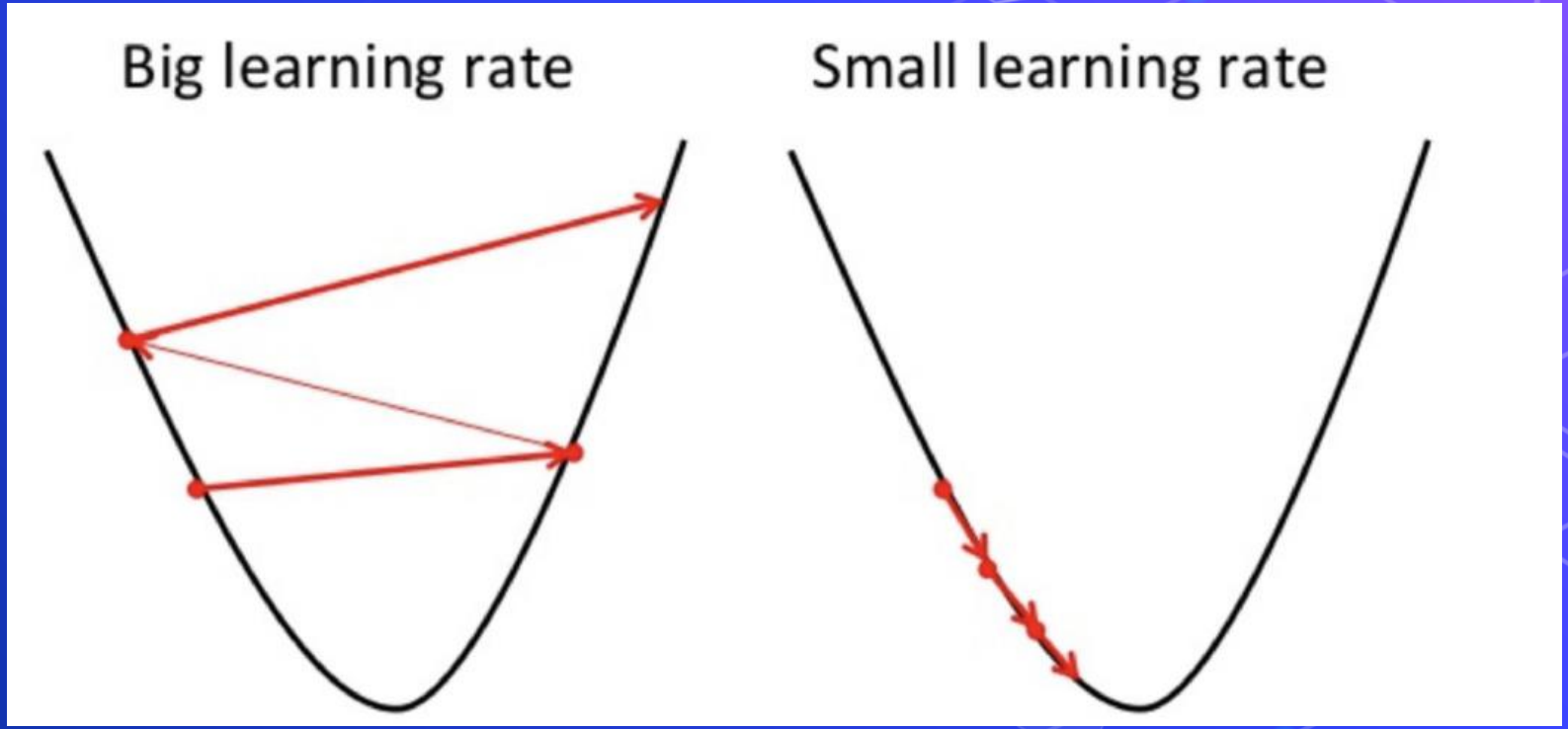
$$= \frac{1}{m}\sum_{i=1}^{m}(h_\Theta(x_i) - y)\frac{\partial}{\partial\Theta_j}(\Theta x_i - y)$$

$$= \frac{1}{m}(h_\Theta(x_i) - y)x_i$$

Therefore,

$$\Theta_j := \Theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}[(h_\Theta(x_i) - y)x_i]$$

Simple Linear Regression (Alpha constant)

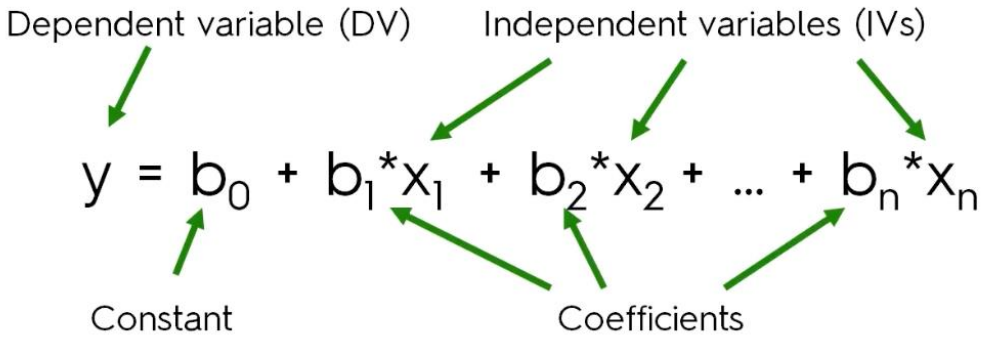


Multiple Linear Regression (Equation)

Simple Linear Regression

$$y = b_0 + b_1 * x_1$$

Multiple Linear Regression



Multiple Linear Regression (Update theta values)

repeat until convergence: {

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

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_2^{(i)}$$

$\dots$

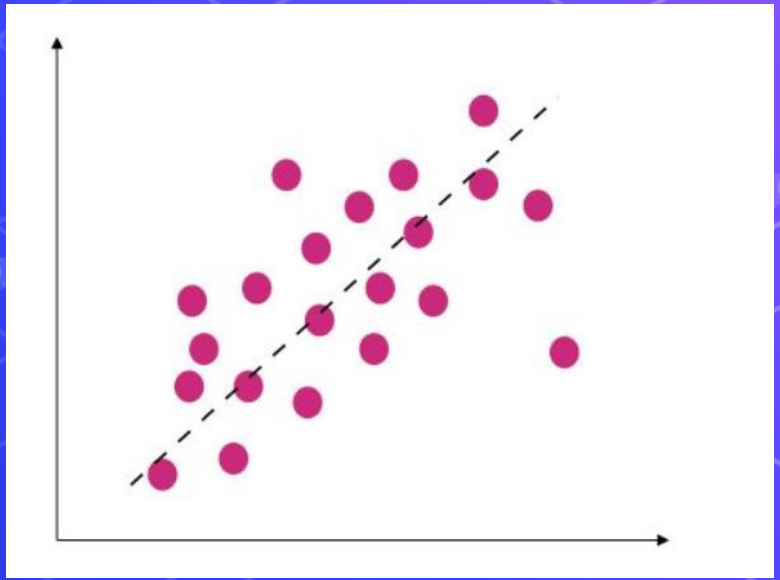
}

Linear Regression (Code)

```python
from sklearn.linear_model import LinearRegression

# make model object
model = LinearRegression()

# train
model.fit(x_train, y_train)

# test
model.predict(x_test)

# calculate R2 score on training data
model.score(x_train, y_train)

# calculate R2 score on testing data
model.score(x_test, y_test)

# get model parameters
model.coef_
model.intercept_
```

Regression

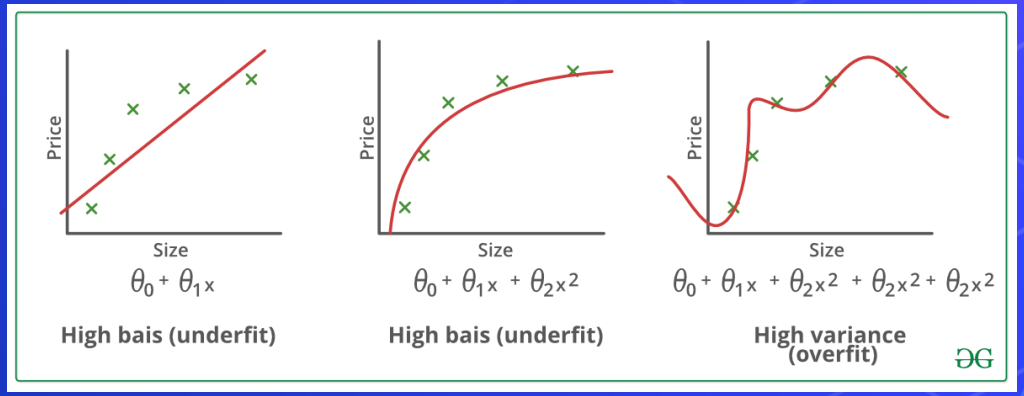
- Simple & Multiple Linear Regression
- **Other Regression Algorithms**
- Evaluating Model Performance

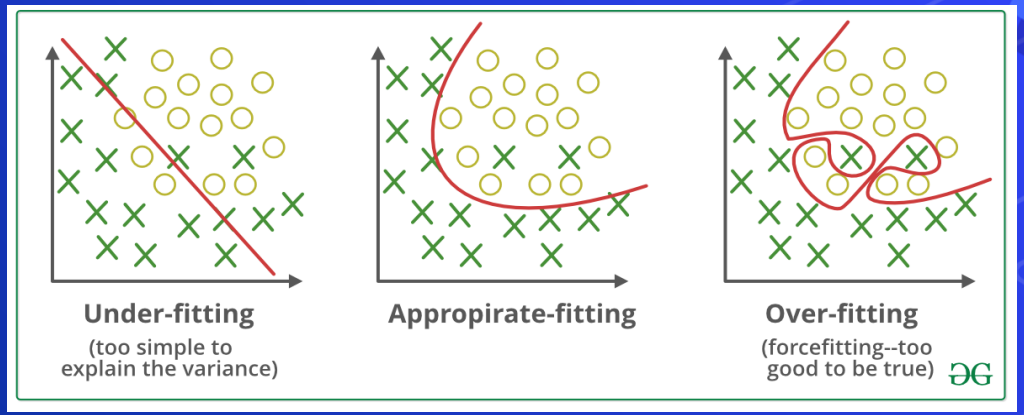


Other Regression Algorithms

- K-Nearest Neighbors (KNN)
- SVM
- Decision Trees
- Random Forests
- XGBoost

Underfitting & Overfitting





Underfitting & Overfitting

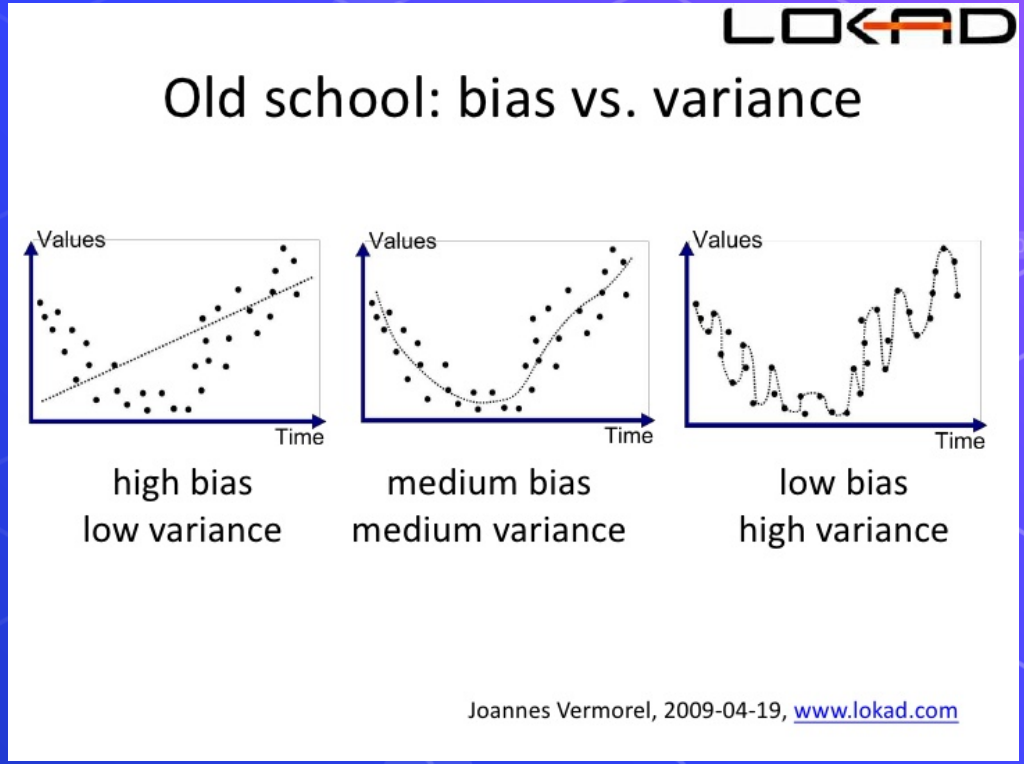
The more features you add the accuracy improves but this can lead to **Overfitting** that means the model memorizes the data too much and not applying **Generalization**.

So we should know about the trade-off between **Bias** and **Variance**.

- A model with a high bias error **underfits** data and makes very simplistic assumptions on it.

- A model with a high variance error **overfits** the data and learns too much from it.

- A good model is where both Bias and Variance errors are balanced

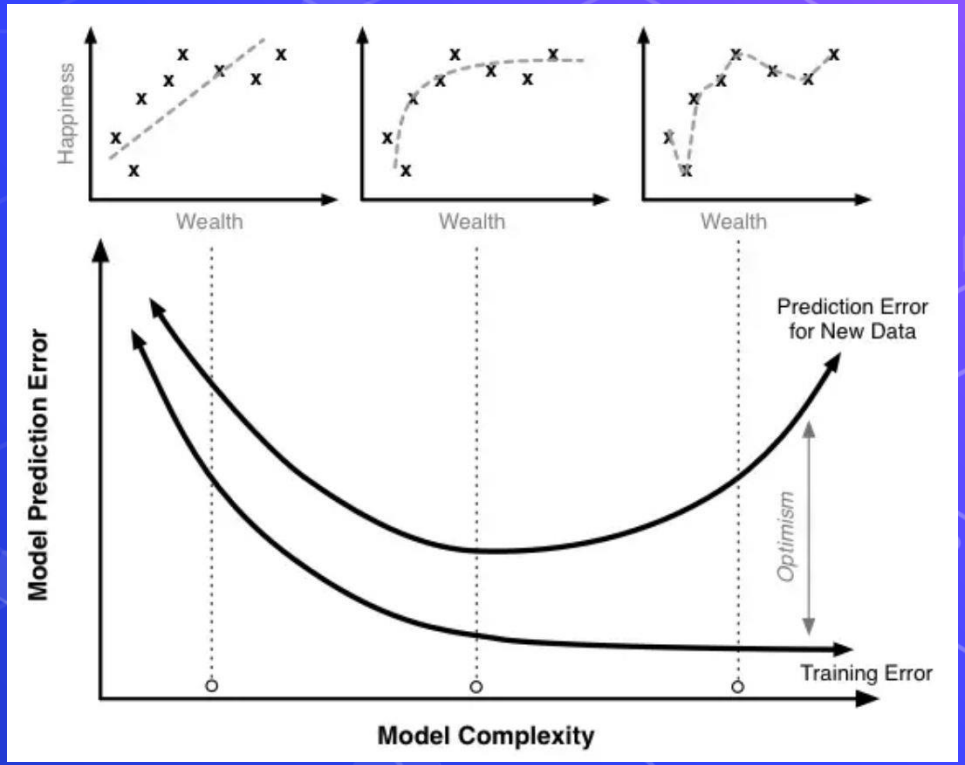


Underfitting & Overfitting

Underfitting (High Bias, Low Variance)
Model is so simple and weak to understand and represent data.
High Train Error + High Test Error

Overfitting (Low Bias, High Variance)
Model is so complex so it memorizes training data and can't apply generalization.
Low Train Error + High Test Error

Optimal (Low Bias, Low Variance)
Model is not simple nor complex and apply generalization well.
Low Train Error + Low Test Error



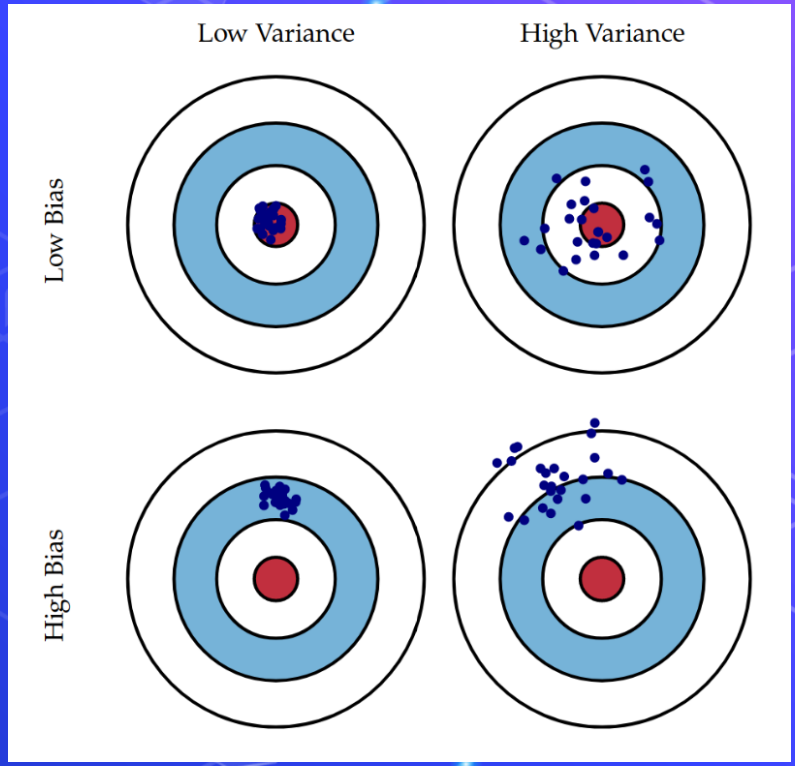
Underfitting & Overfitting

Techniques to reduce underfitting

- Increase model complexity.
- Increase number of features.
- Performing feature engineering.
- Remove noise from the data.

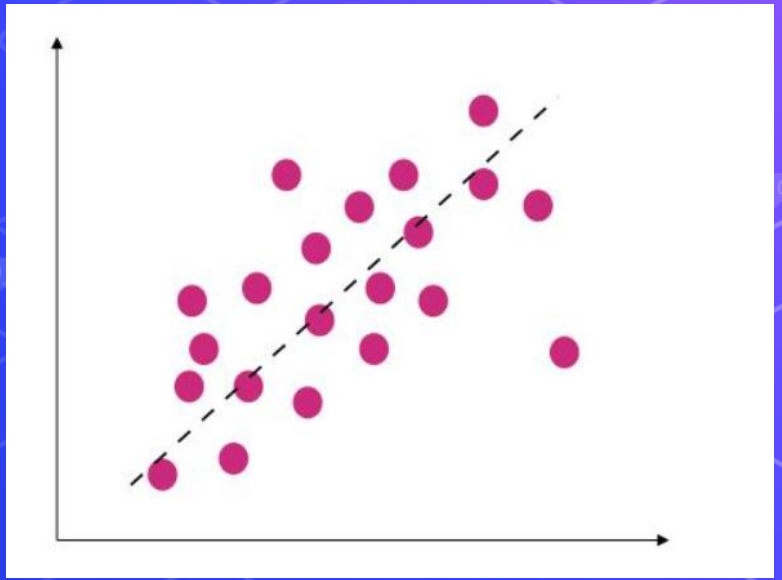
Techniques to reduce overfitting

- Increase training data.
- Remove correlated features.
- Use simpler model.
- Regularization (add penalty bias).



Regression

- Simple & Multiple Linear Regression
- Other Regression Algorithms
- Evaluating Model Performance



Evaluating Model Performance (R2)

Hrs Studied (X)	Marks (Y)
0	40
2	52
3	53
4	55
4	56
5	72
6	71
6	88
7	56
7	74
8	89
9	67
9	89
5.38	66.31
Mean	

$$R^2 = SSR/SST$$

$$= 1844.12/3028.77$$

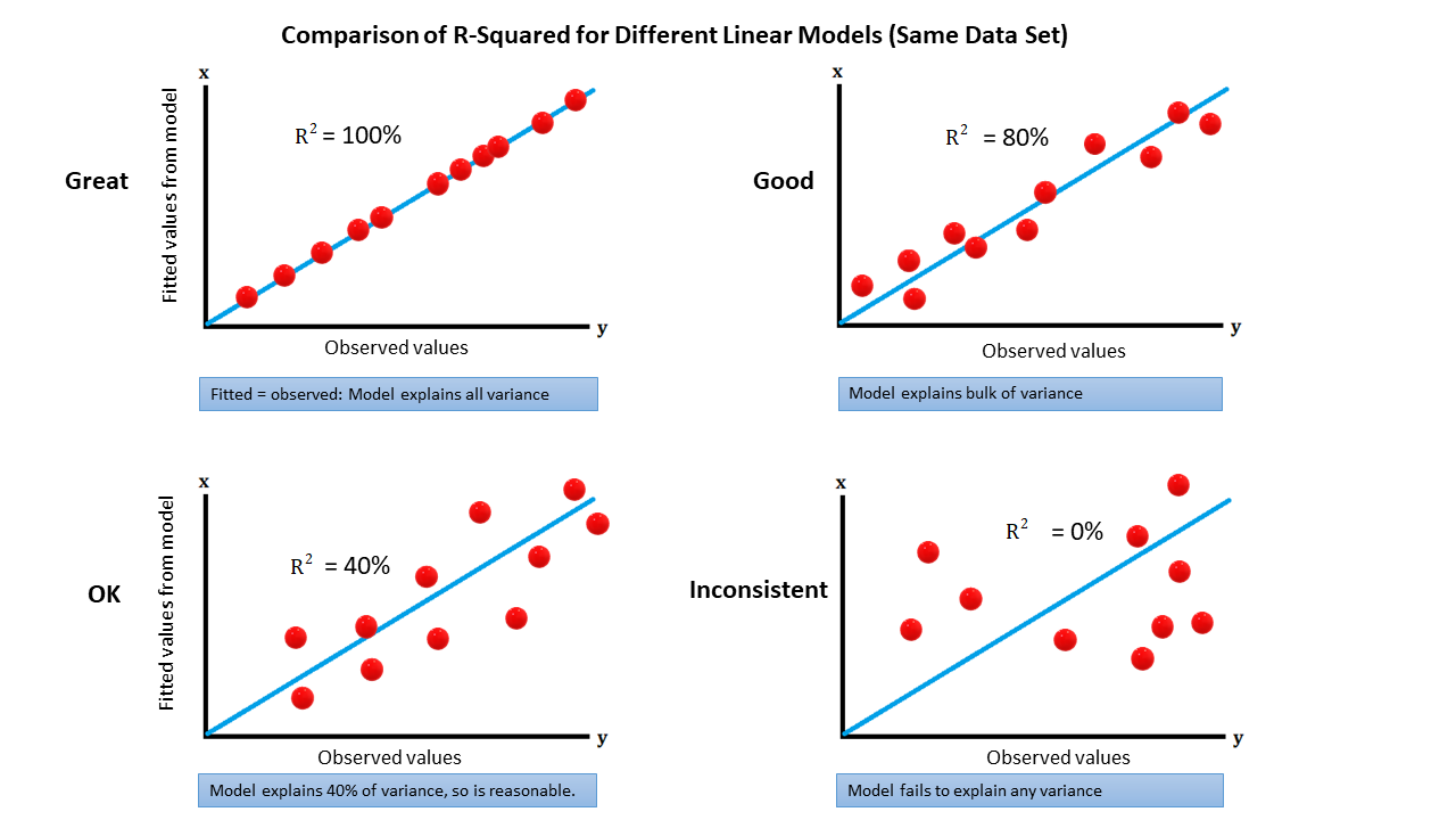
$$= 0.60886$$

Higher the value → Variation in Y is explained by variation in X.

$\hat{y} \rightarrow y$ SSR → SST $R^2 \approx 1$

$(Y - \bar{Y})^2$	$(\hat{Y} - \bar{Y})^2$
692.22	600.74
204.78	237.47
177.16	117.94
127.92	39.82
106.30	39.82
32.38	3.10
22.00	7.78
470.46	7.78
106.30	53.88
59.14	53.88
514.84	141.37
0.48	270.27
514.84	270.27
3028.77	1844.12
SST	SSR

Evaluating Model Performance (R2)



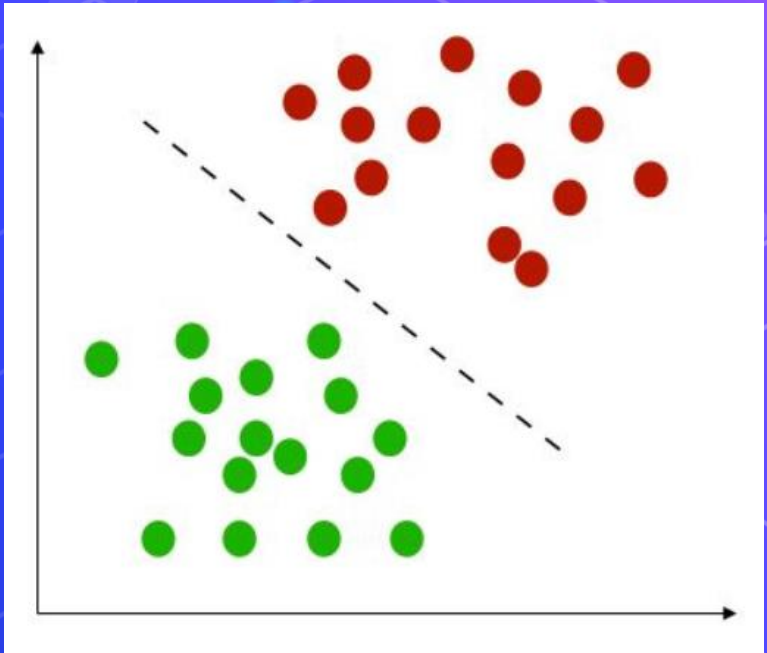
Agenda

- What is Machine Learning
- **Supervised Learning**
  - Regression
  - **Classification**
- Unsupervised Learning
  - Clustering
  - Dimension Reduction
- Model Selection & Evaluation
  - Cross Validation
  - Hyperparameter Tuning
- Intro to Deep Learning

Classification

Classification specifies the class to which data elements belong to and is best used when the output has finite and discrete values. It predicts a class for an input variable as well.

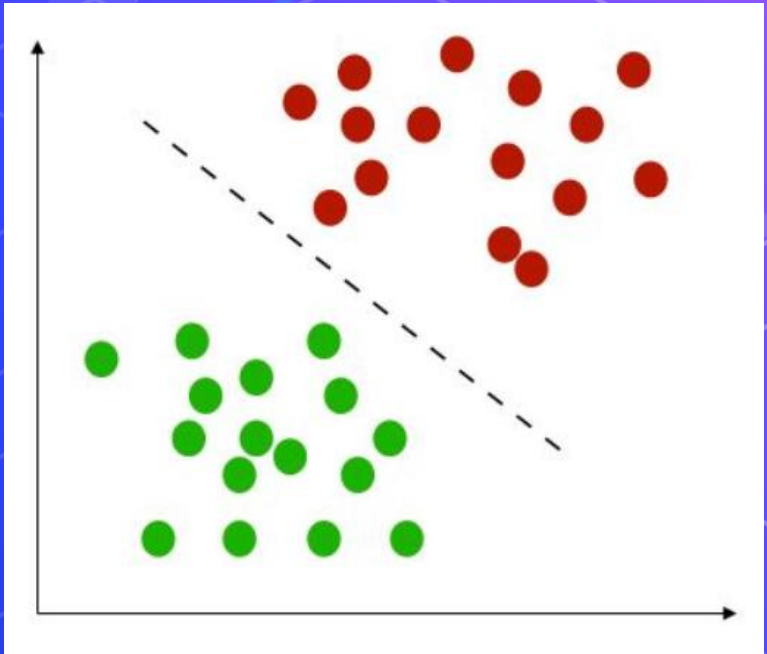
For example,
it can be used to classify the animal in an image, it's either a cat or a dog.



- Email spam detection
- Face classification
- Patient has cancer or not
- Fraud detection
- ...

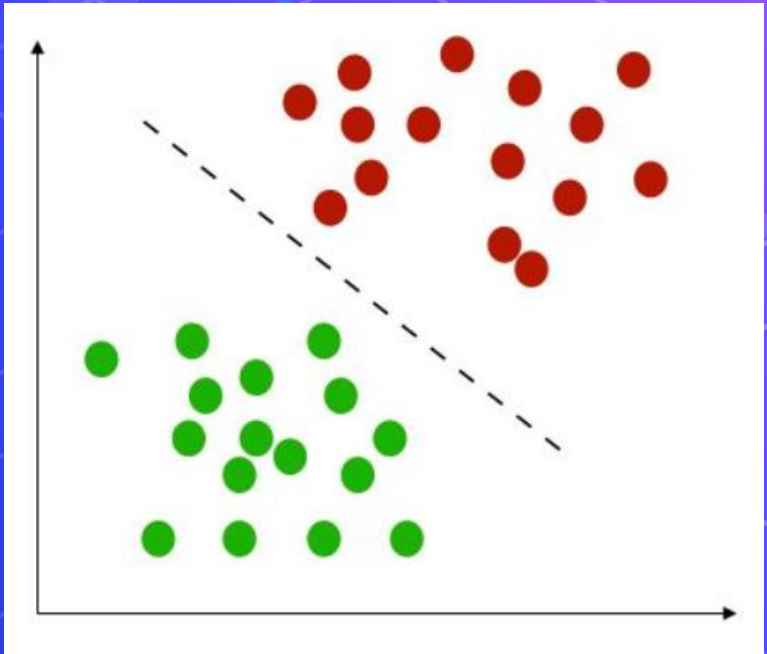
Classification

- Logistic Regression
- K-Nearest Neighbors (KNN)
- Naive Bayes
- SVM
- Decision Trees
- Ensemble Methods
  - What is Bagging & Boosting
  - Random Forests
  - XGBoost
- Evaluating Model Performance



Classification

- Logistic Regression
- K-Nearest Neighbors (KNN)
- Naive Bayes
- SVM
- Decision Trees
- Ensemble Methods
  - What is Bagging & Boosting
  - Random Forests
  - XGBoost
- Evaluating Model Performance



Logistic Regression

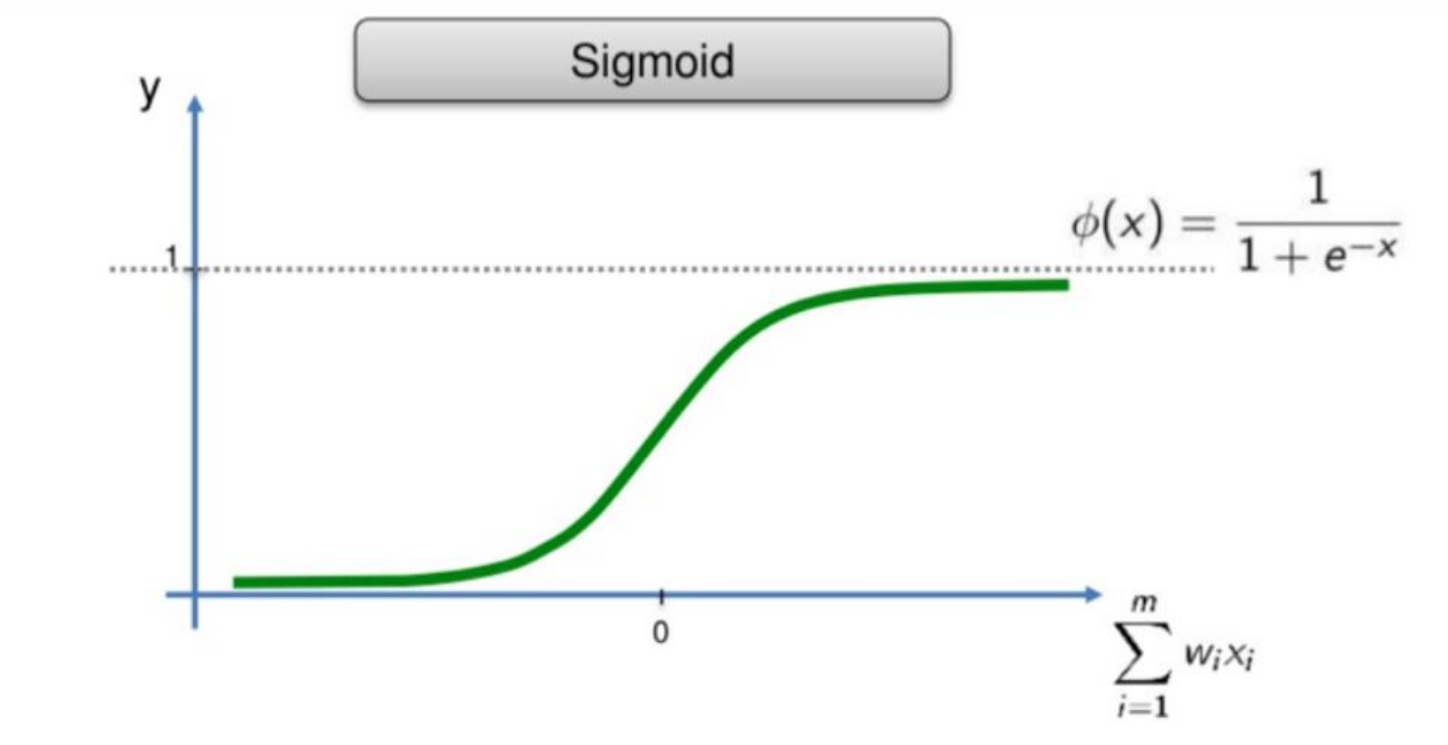
The Logistic Function

$$y = \frac{1}{1 + e^{-f(x_1, x_2, \ldots, x_n)}} \in (0, 1)$$

where

$$f(x_1, x_2, \ldots, x_n) = a_0 + a_1 x_1 + \ldots + a_n x_n \in (-\infty, +\infty)$$

Logistic Regression

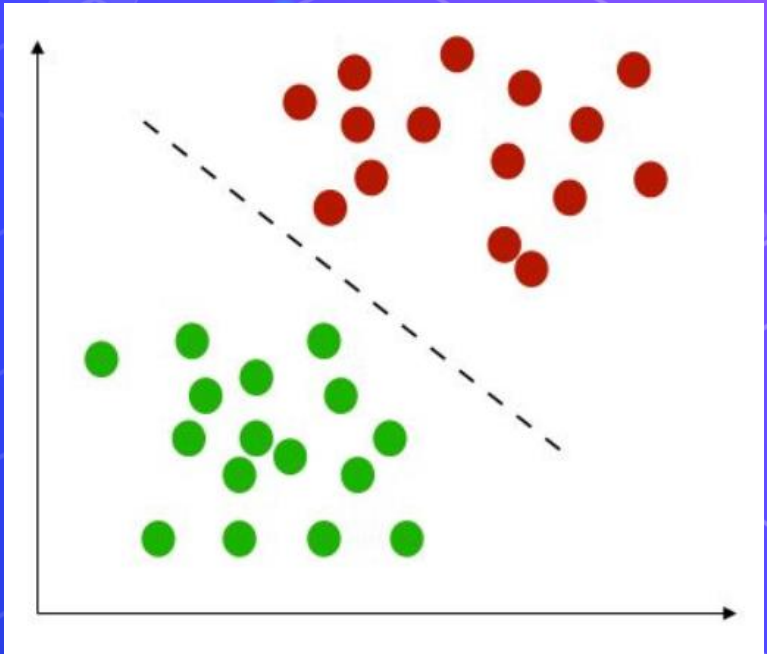


Logistic Regression

```python
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix


model = LogisticRegression()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)

print(confusion_matrix(y_test, y_pred))
print(accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

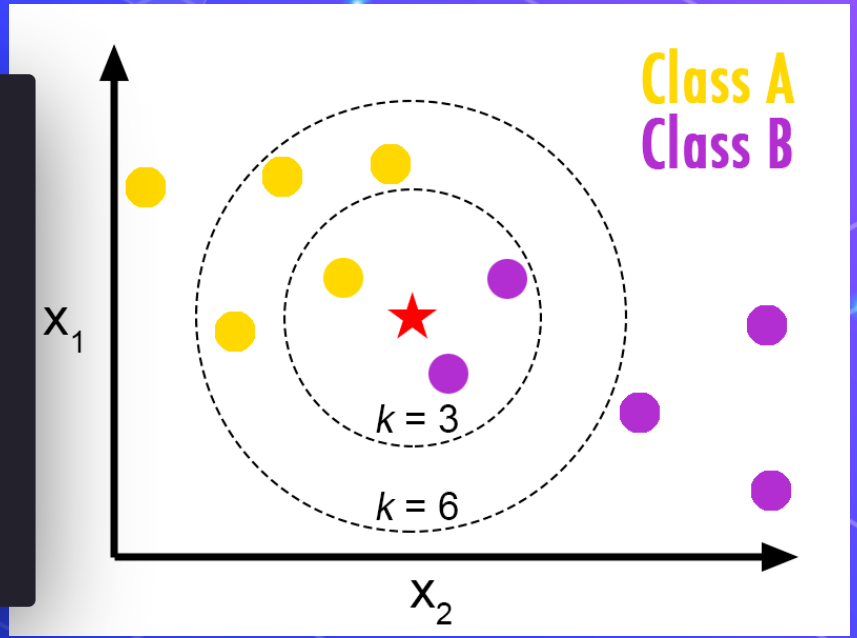
Classification

- Logistic Regression
- K-Nearest Neighbors (KNN)
- Naive Bayes
- SVM
- Decision Trees
- Ensemble Methods
  - What is Bagging & Boosting
  - Random Forests
  - XGBoost
- Evaluating Model Performance



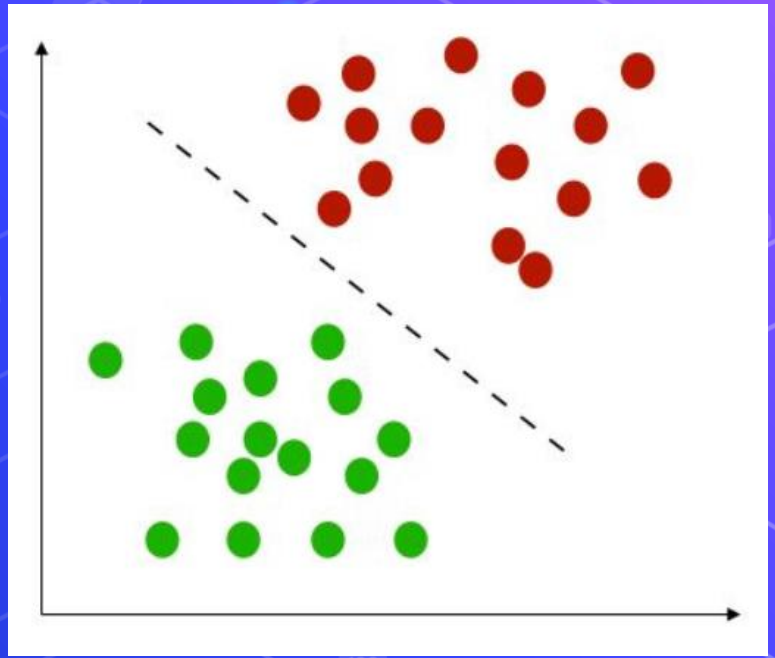
K-Nearest Neighbors (KNN)

```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix


model = KNeighborsClassifier()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)

print(confusion_matrix(y_test, y_pred))
print(accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

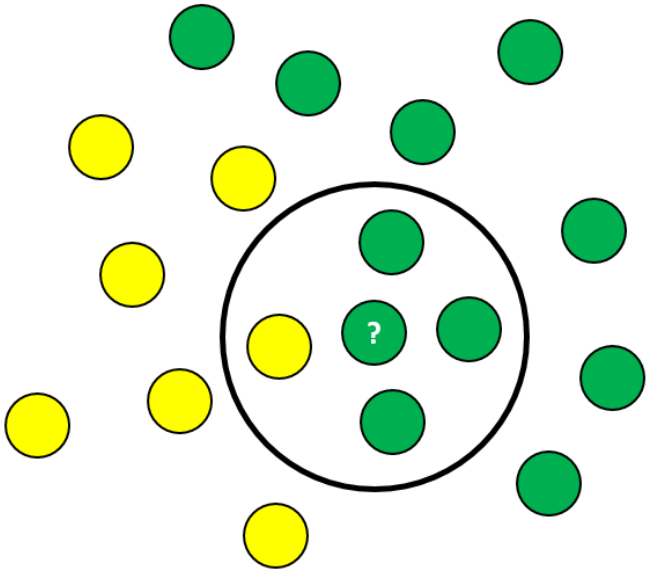


Classification

- Logistic Regression
- K-Nearest Neighbors (KNN)
- **Naive Bayes**
- SVM
- Decision Trees
- Ensemble Methods
  - What is Bagging & Boosting
  - Random Forests
  - XGBoost
- Evaluating Model Performance



Naive Bayes



$P(\text{yellow}) = \frac{7}{17}$

$P(\text{green}) = \frac{10}{17}$

$P'(? \mid \text{green}) = \frac{3}{10}$

$P'(? \mid \text{yellow}) = \frac{1}{7}$

prior probabilities
number of samples in a given class
divided by the total number of samples

we consider just the vicinity
of the new sample we wan to classify

posterior probability $P''(? \text{ is green}) = P(\text{green}) * P'(? \mid \text{green}) = \frac{10}{17} * \frac{3}{10} = \frac{30}{170}$

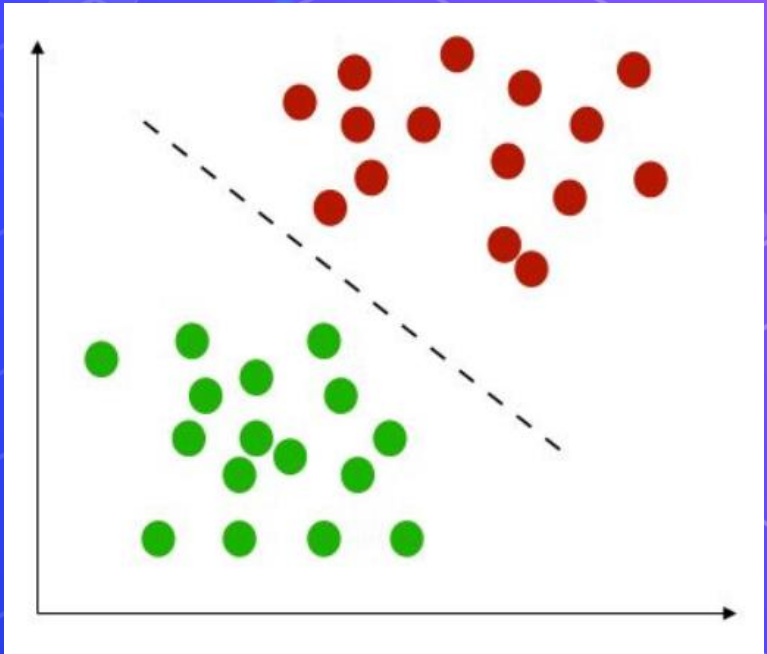
posterior probability $P''(? \text{ is yellow}) = P(\text{yellow}) * P'(? \mid \text{yellow}) = \frac{7}{17} * \frac{1}{7} = \frac{7}{119}$

Naive Bayes

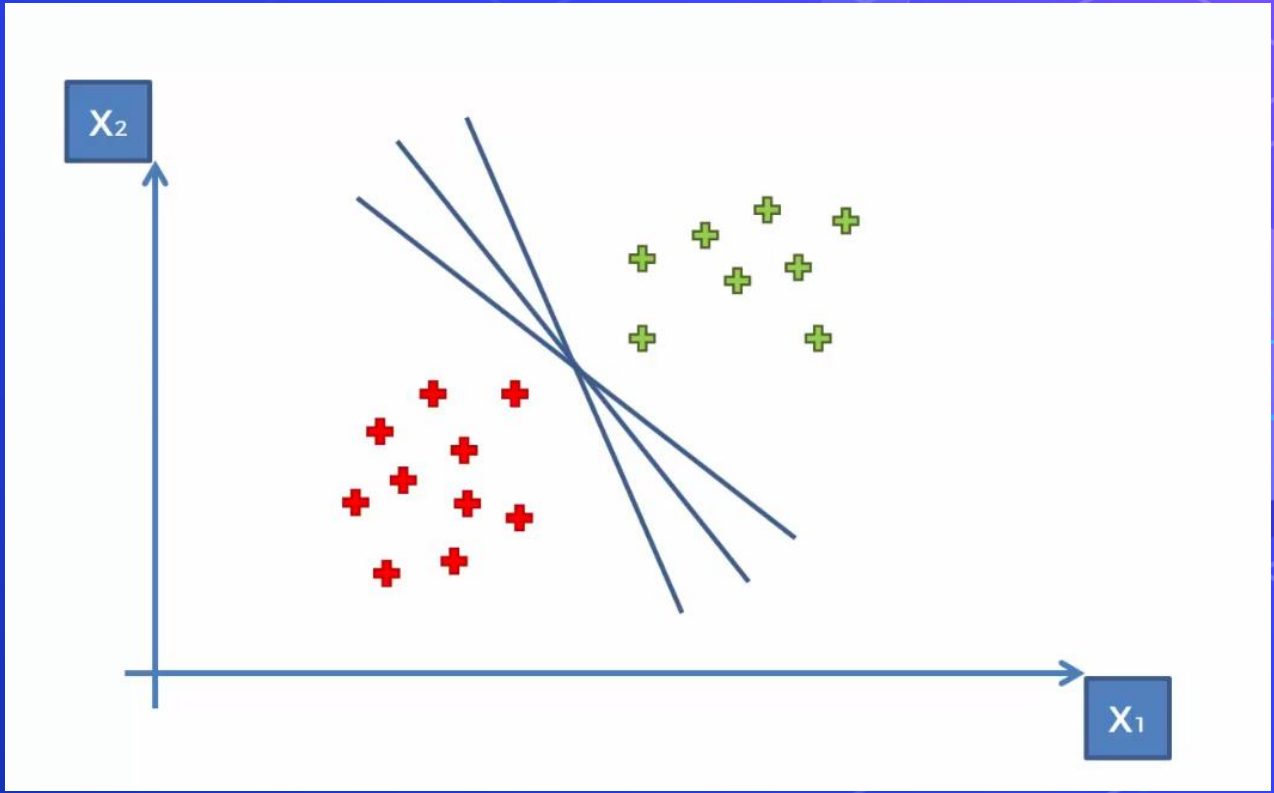
```python
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix


model = MultinomialNB()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)

print(confusion_matrix(y_test, y_pred))
print(accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Classification

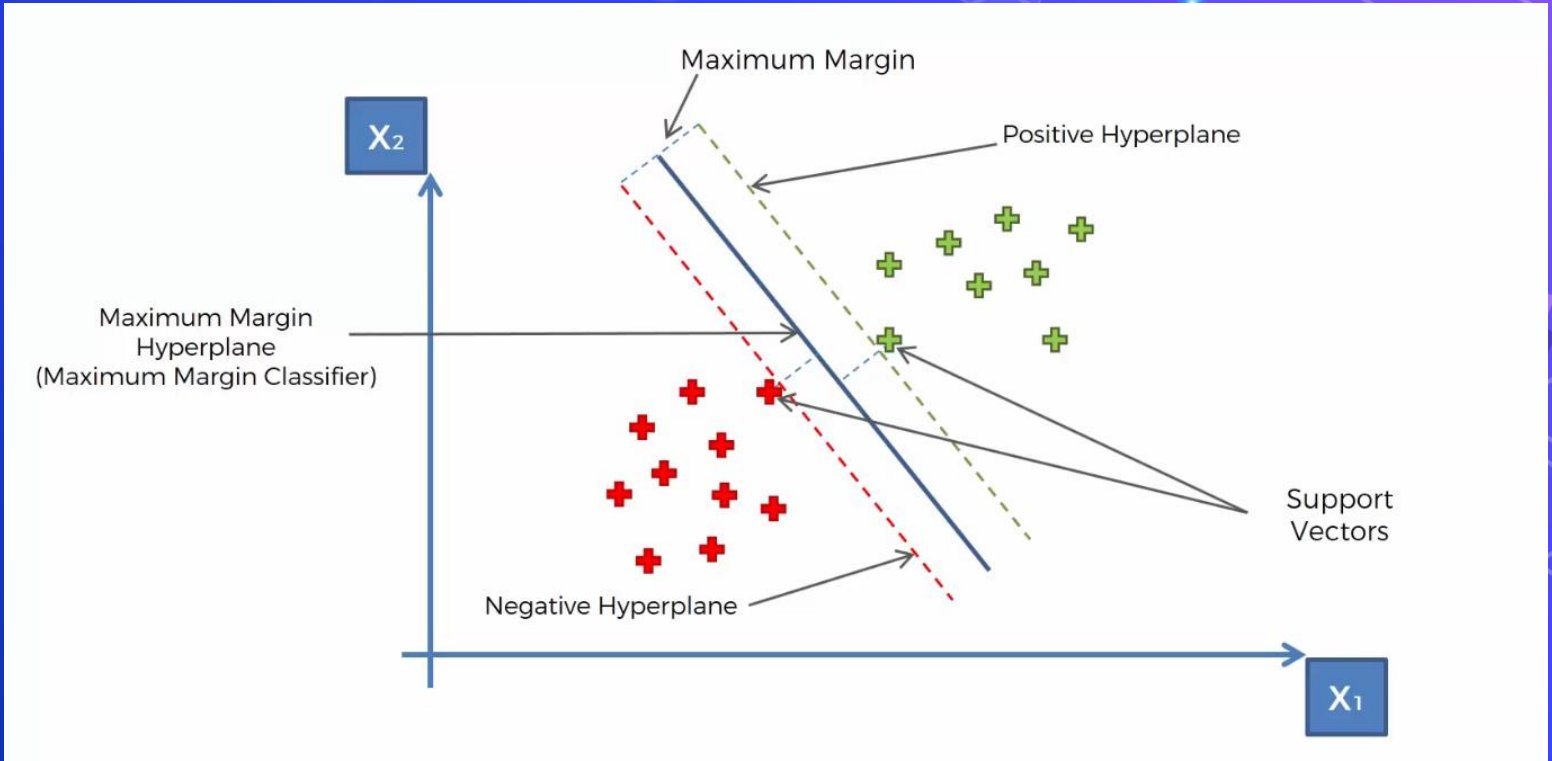
- Logistic Regression
- K-Nearest Neighbors (KNN)
- Naive Bayes
- SVM
- Decision Trees
- Ensemble Methods
  - What is Bagging & Boosting
  - Random Forests
  - XGBoost
- Evaluating Model Performance



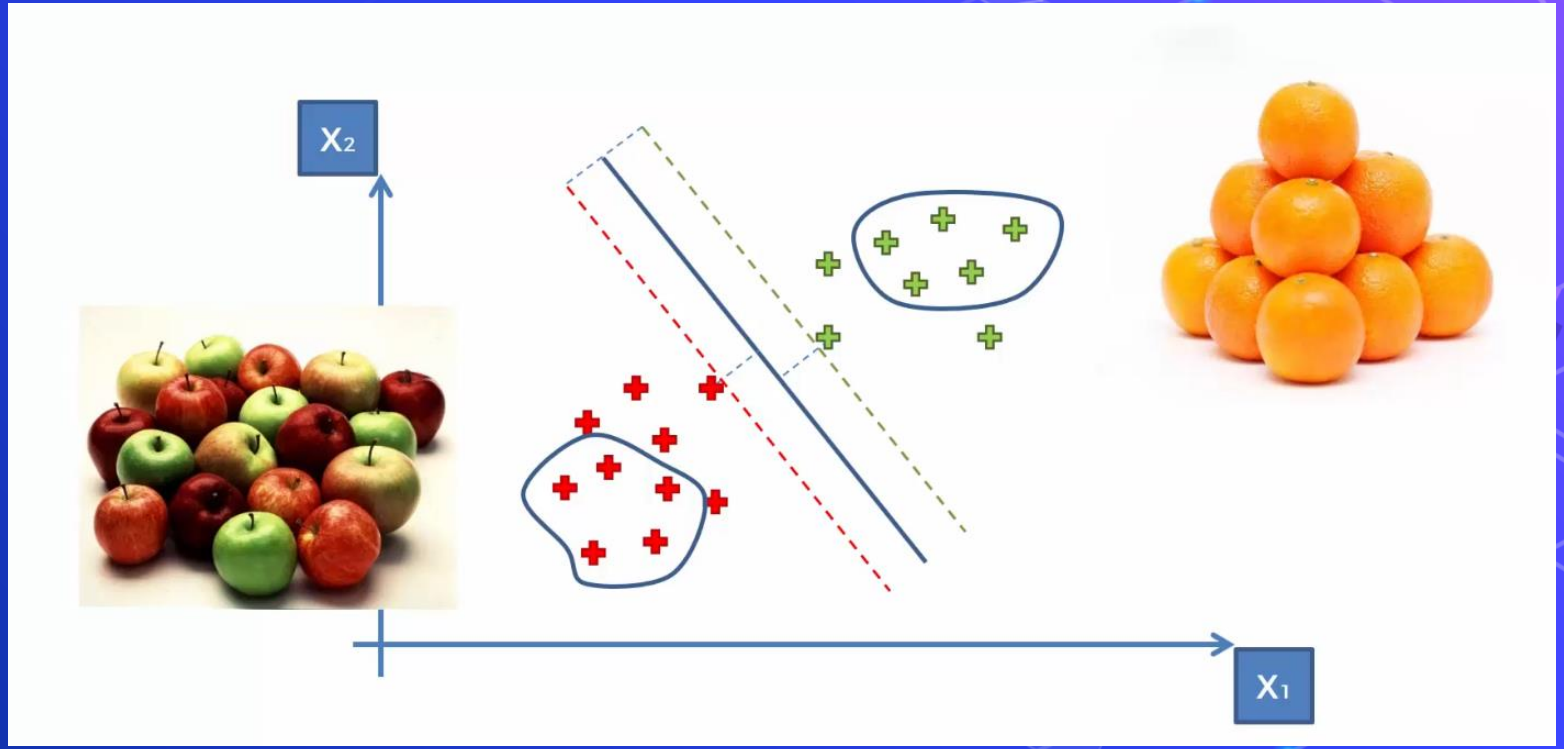
SVM (Support Vector Machine)



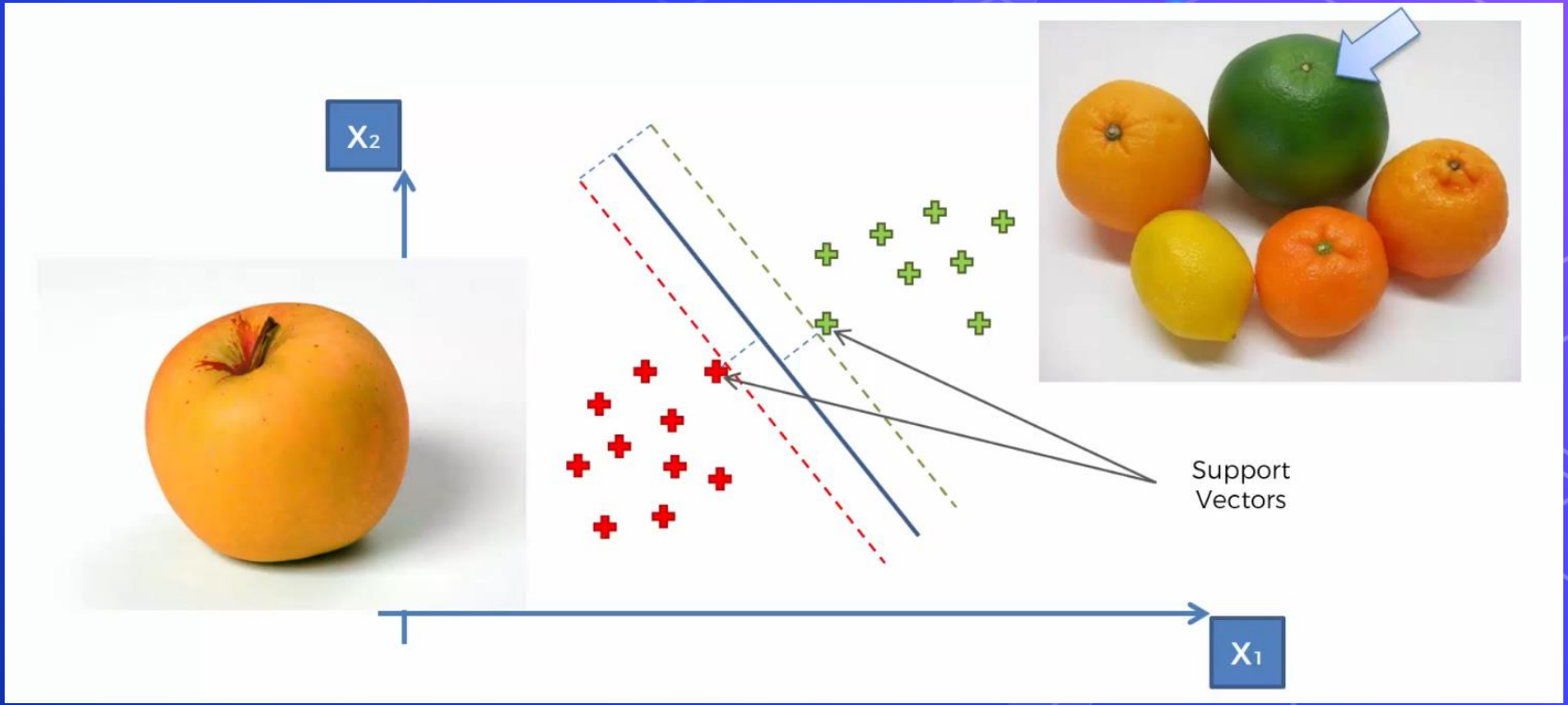
SVM (Support Vector Machine)



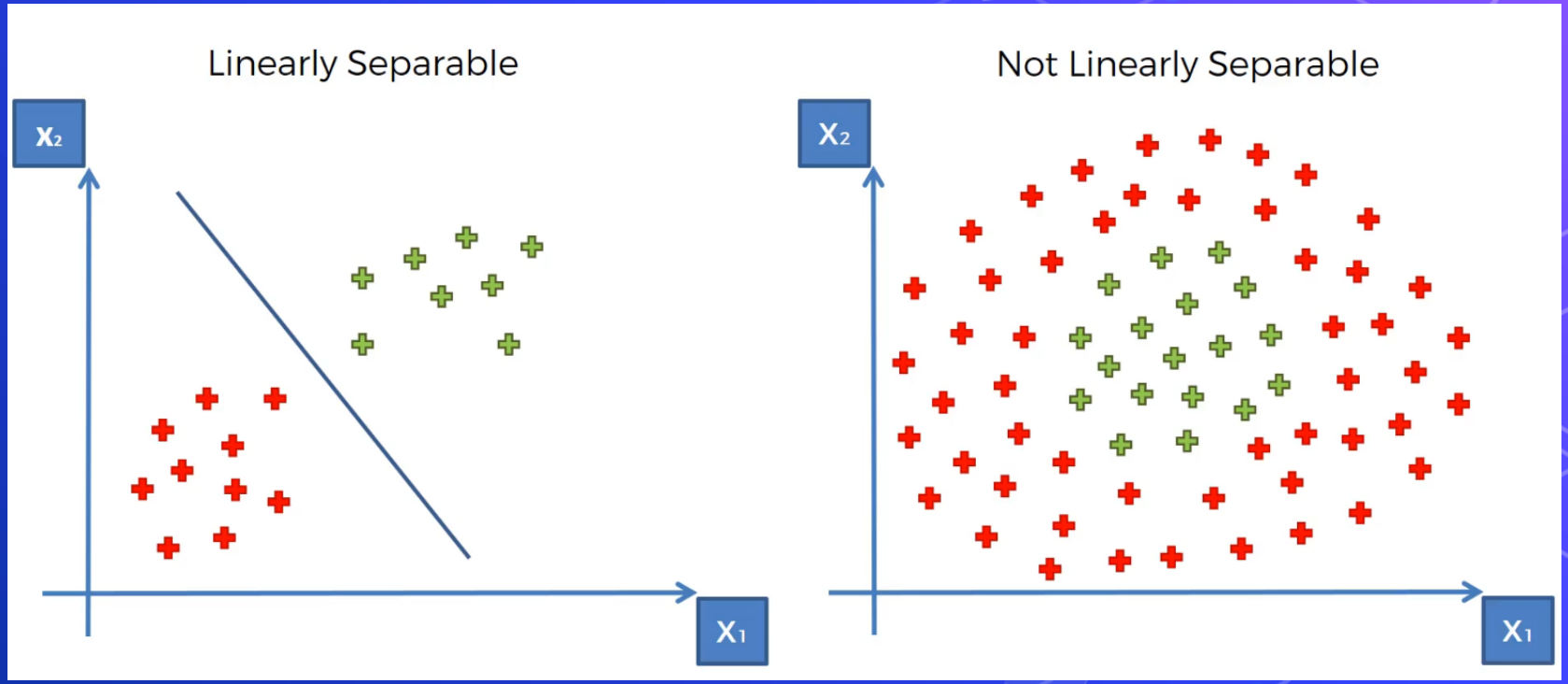
SVM (Support Vector Machine)



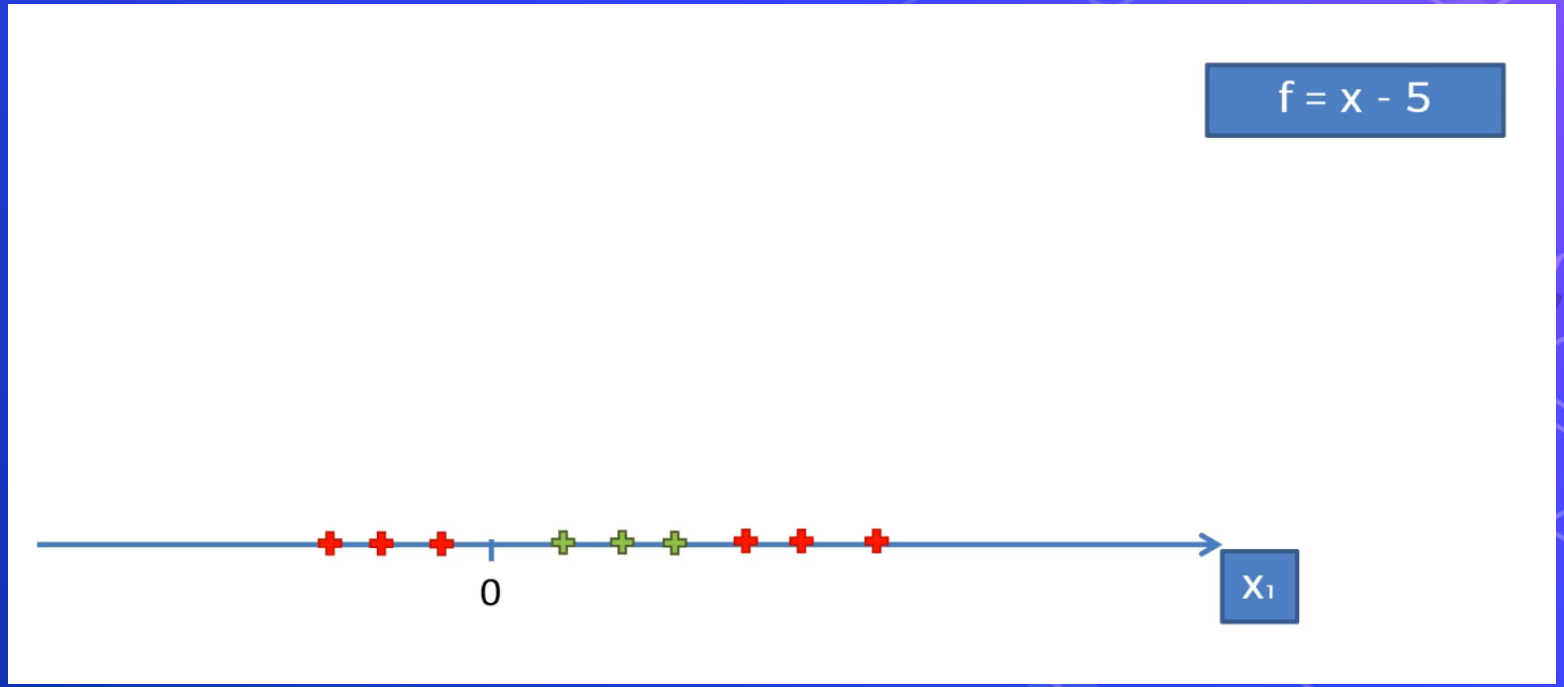
SVM (Support Vector Machine)



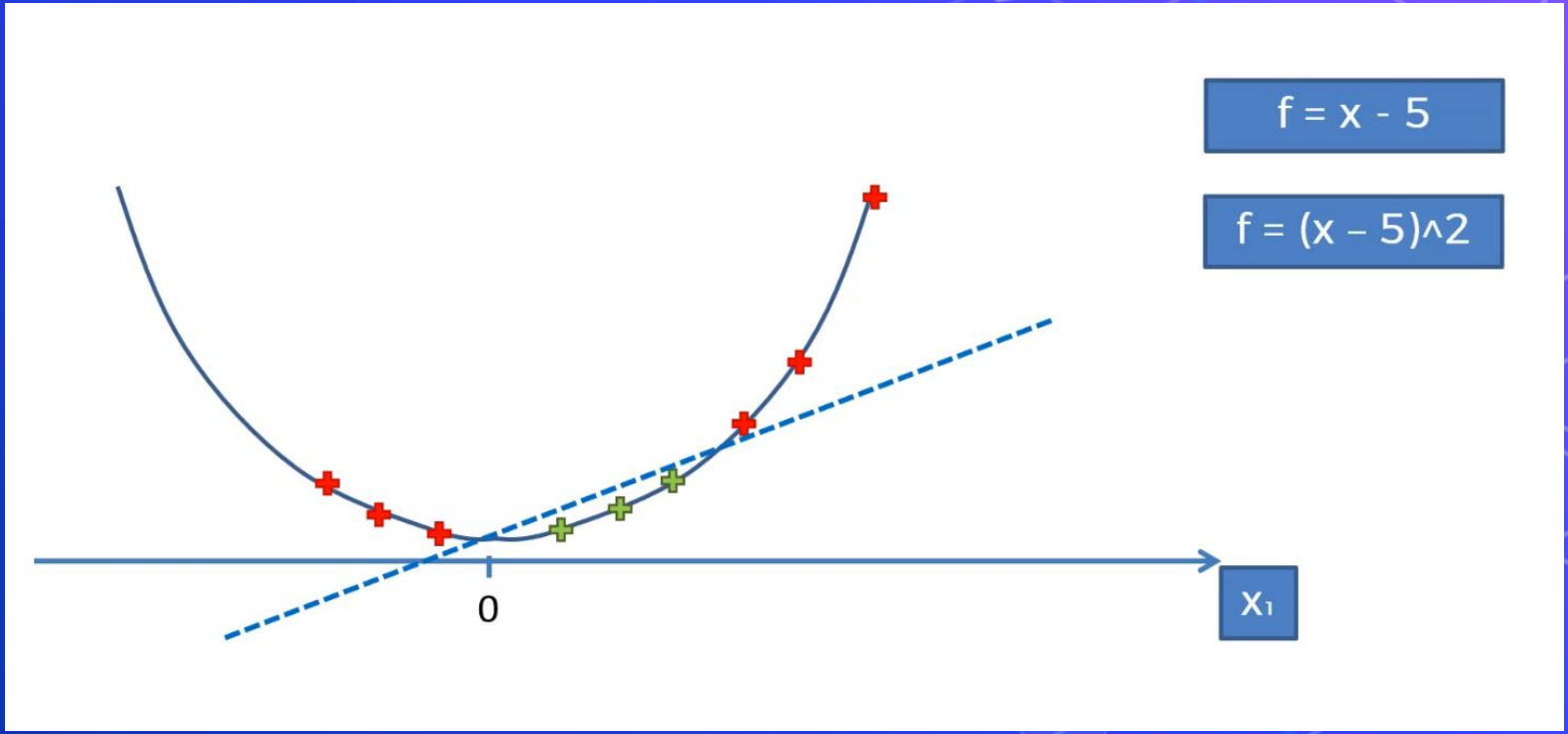
SVM (Support Vector Machine)



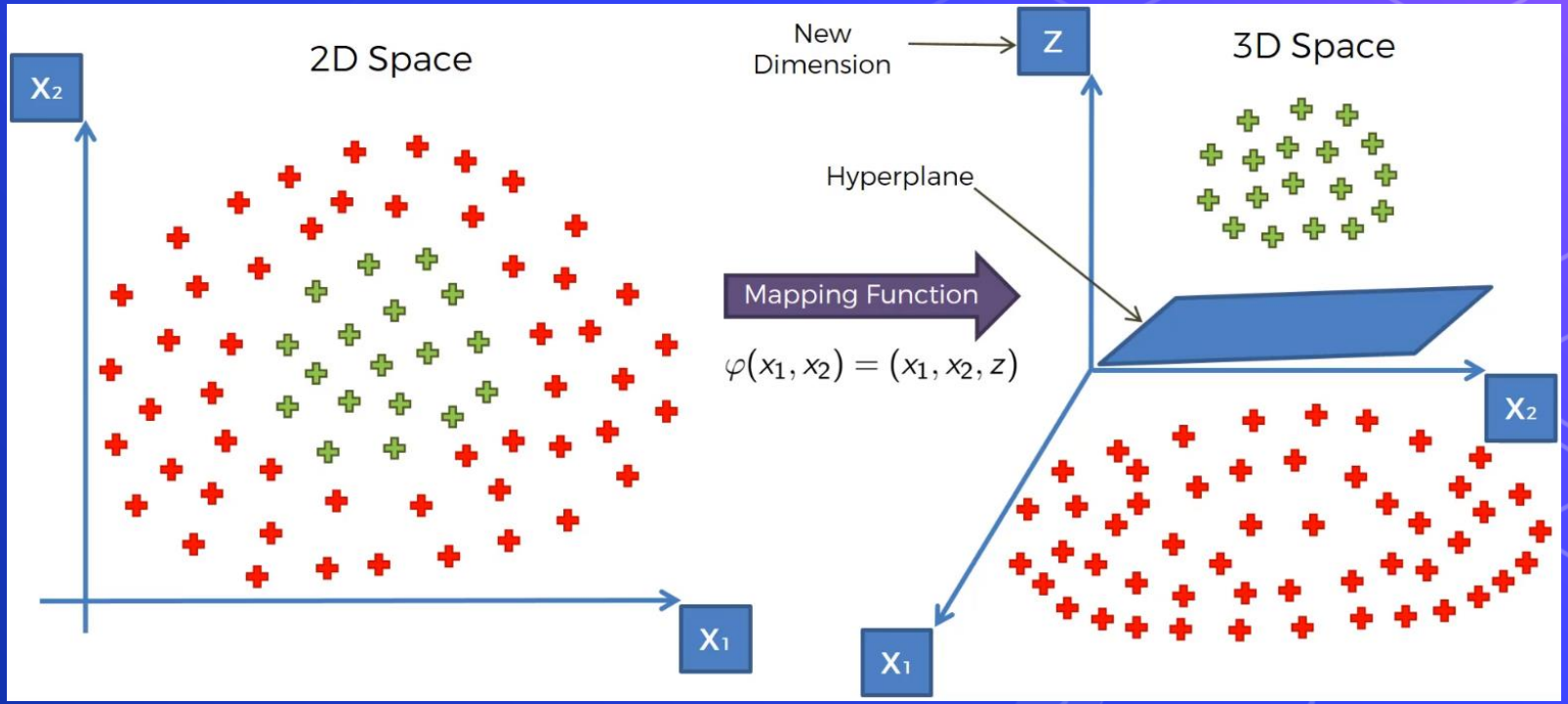
SVM (Support Vector Machine)



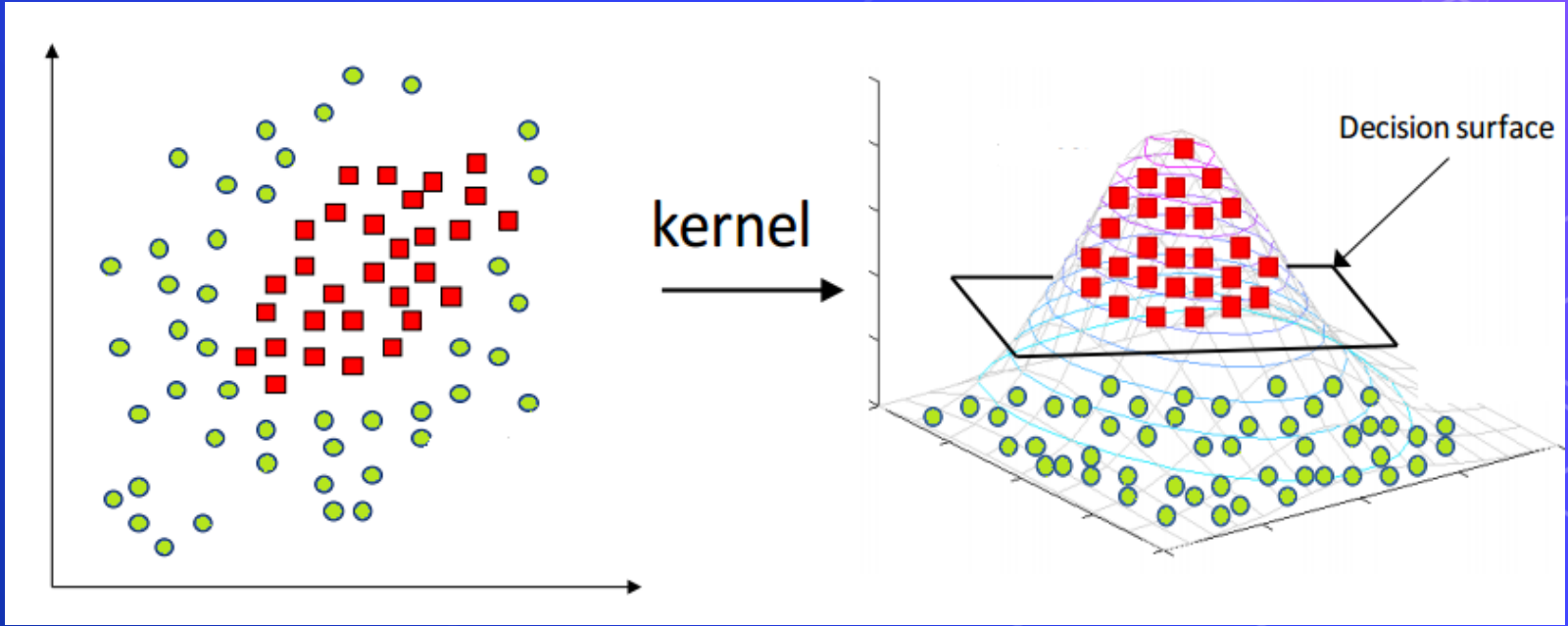
SVM (Support Vector Machine)



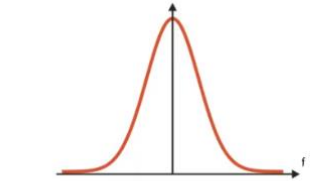
SVM (Support Vector Machine)



SVM (Support Vector Machine)

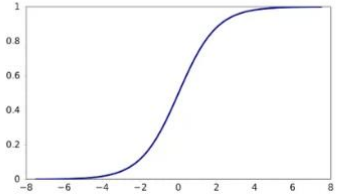


SVM (Support Vector Machine)



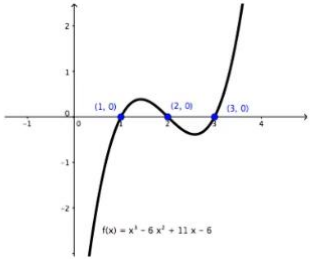
Gaussian RBF Kernel

$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$



Sigmoid Kernel

$$K(X, Y) = \tanh(\gamma \cdot X^T Y + r)$$



Polynomial Kernel

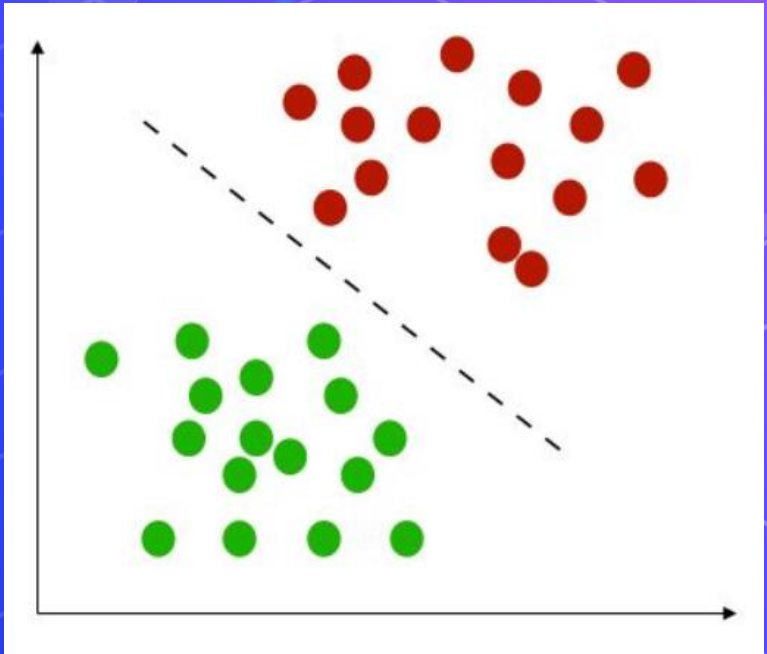
$$K(X, Y) = (\gamma \cdot X^T Y + r)^d, \gamma > 0$$

SVM (Support Vector Machine)

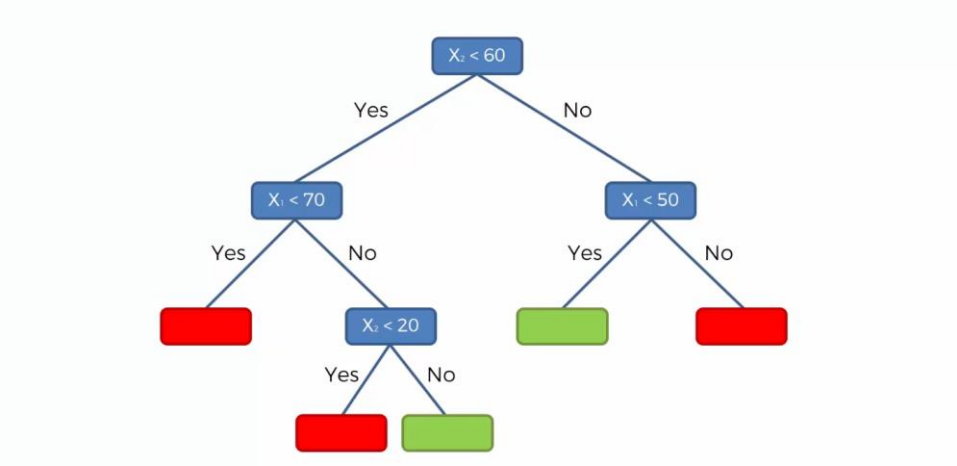
```python
from sklearn.svm import SVC
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix


model = SVC()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)

print(confusion_matrix(y_test, y_pred))
print(accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

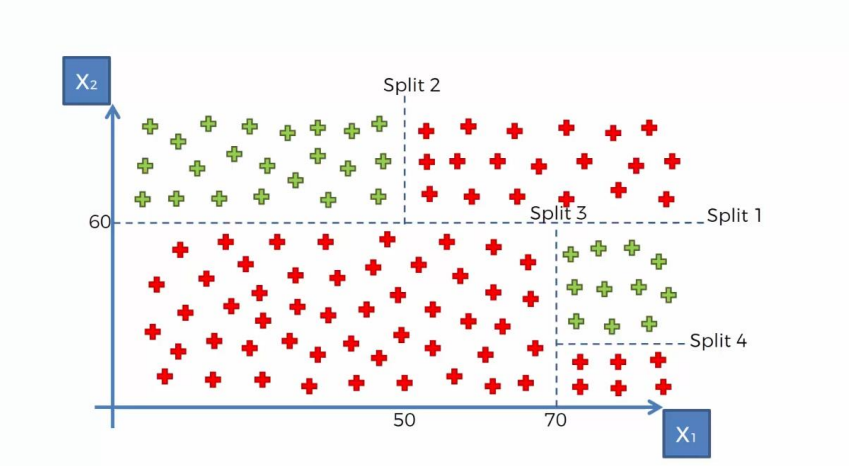
Classification

- Logistic Regression
- K-Nearest Neighbors (KNN)
- Naive Bayes
- SVM
- **Decision Trees**
- Ensemble Methods
  - What is Bagging & Boosting
  - Random Forests
  - XGBoost
- Evaluating Model Performance



Decision Trees



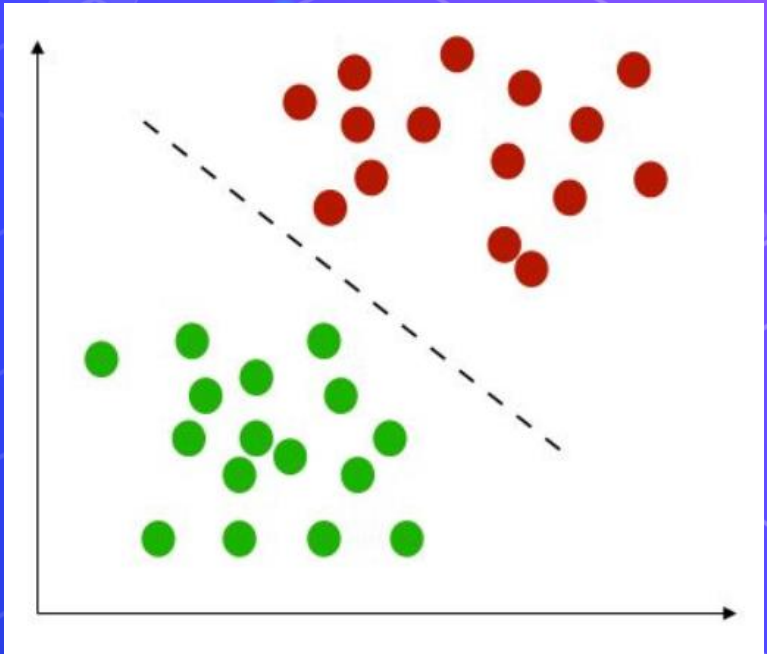


Decision Trees

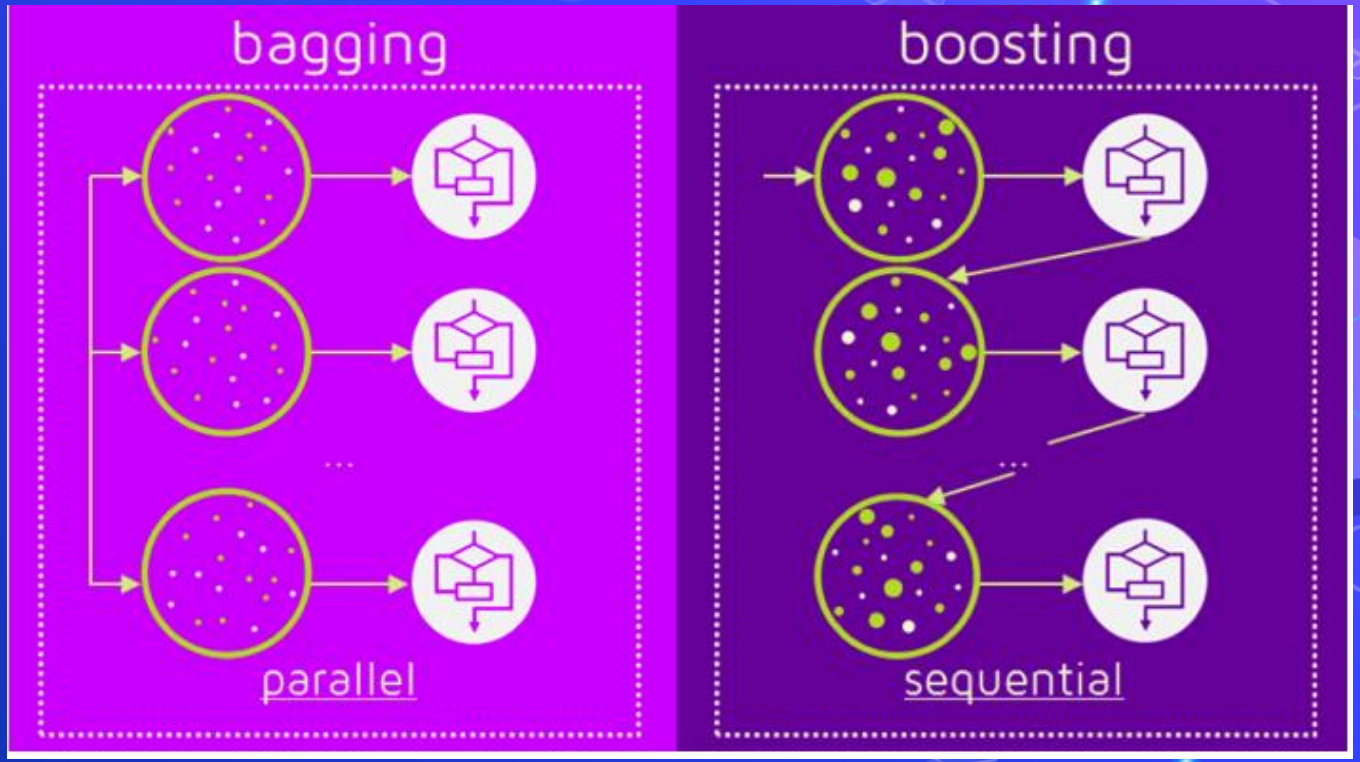
```python
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix


model = DecisionTreeClassifier()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)

print(confusion_matrix(y_test, y_pred))
print(accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Classification

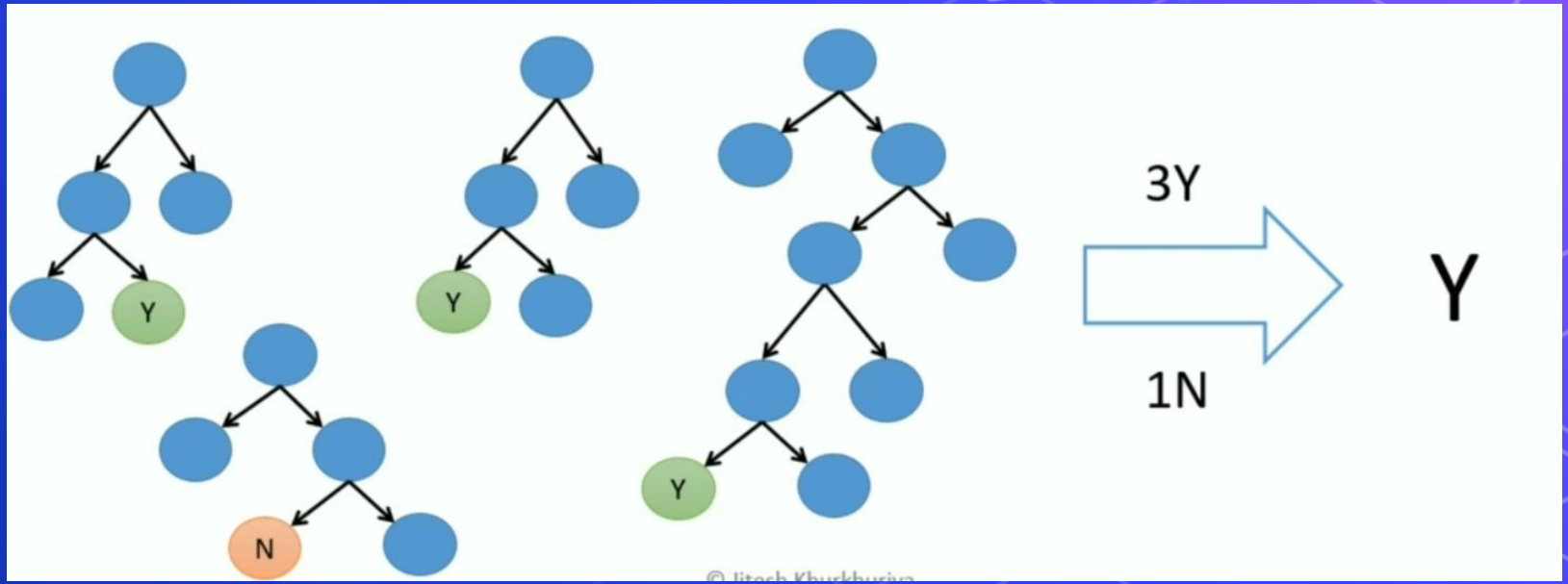
- Logistic Regression
- K-Nearest Neighbors (KNN)
- Naive Bayes
- SVM
- Decision Trees
- **Ensemble Methods**
  - What is Bagging & Boosting
  - Random Forests
  - XGBoost
- Evaluating Model Performance



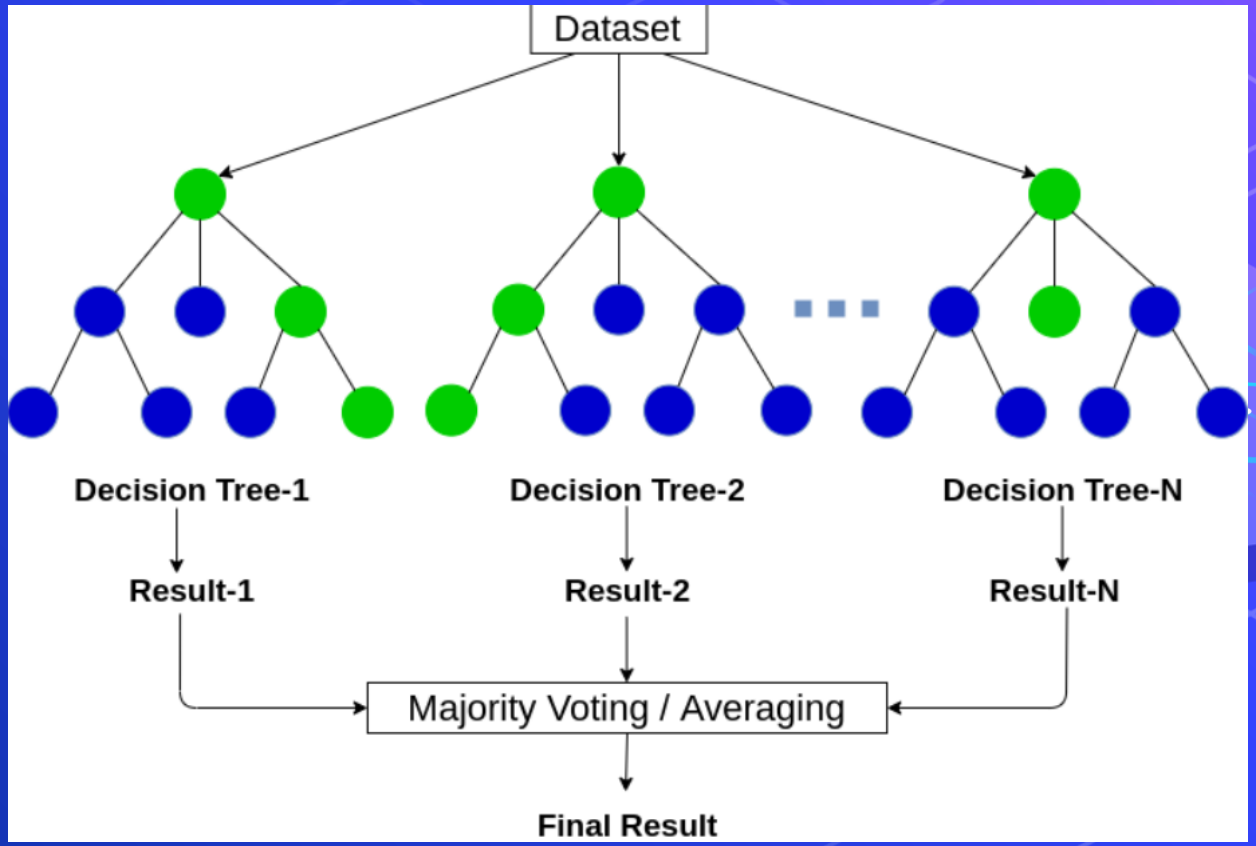
What is Bagging & Boosting



What is **Bagging** & Boosting



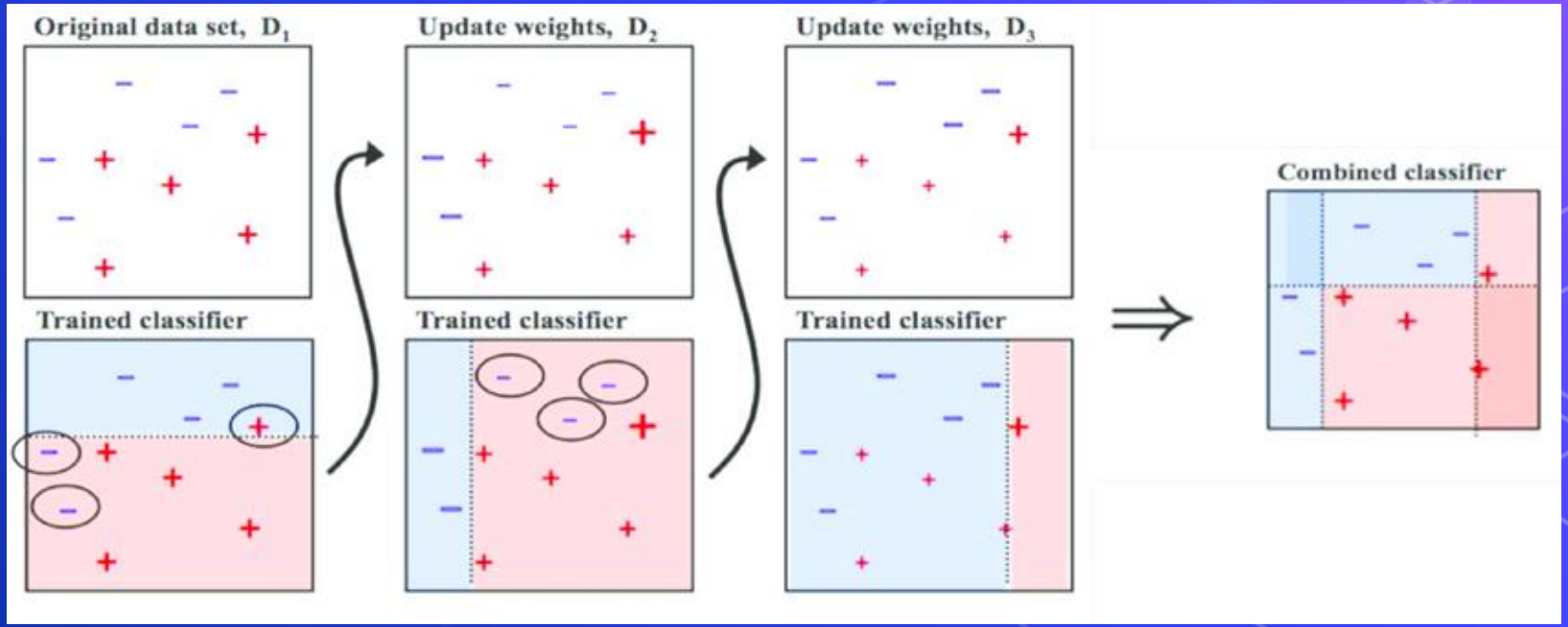
Random Forests



Random Forests

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix


model = RandomForestClassifier()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)

print(confusion_matrix(y_test, y_pred))
print(accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

What is Bagging & **Boosting**

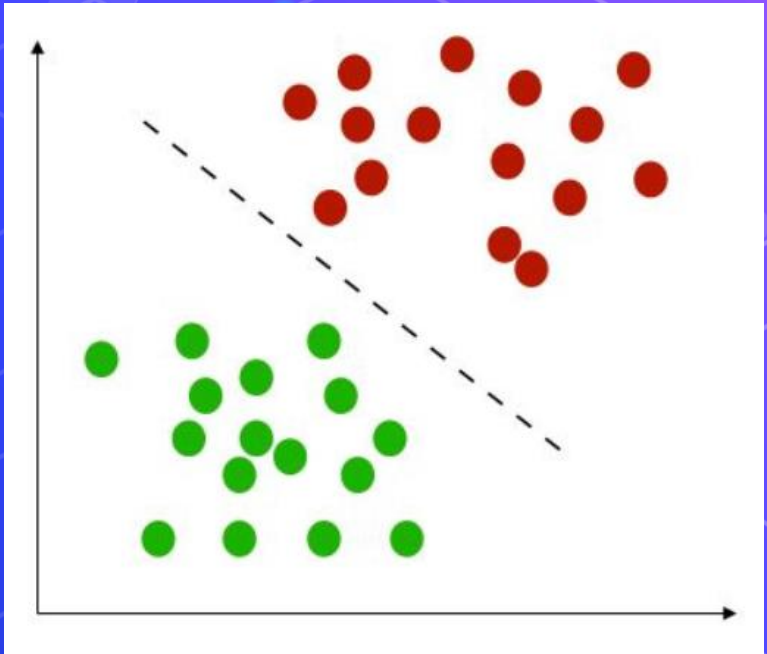


XGBoost

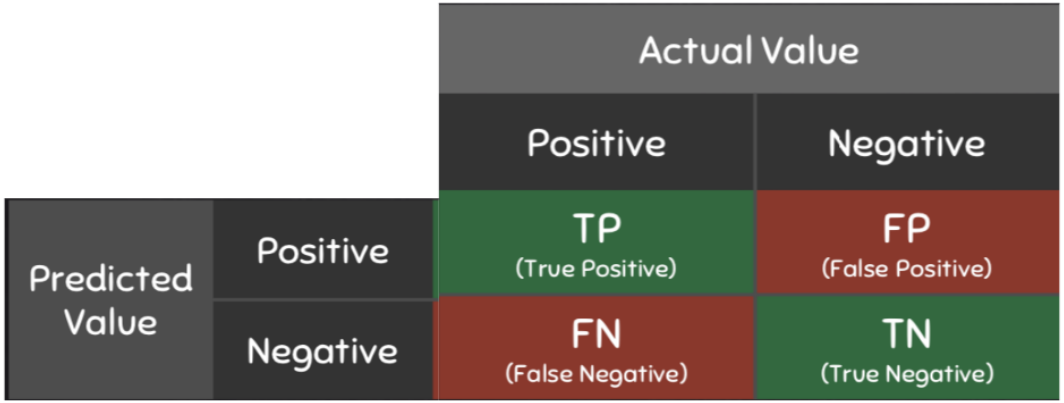
```python
import xgboost as xgb
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix


model = xgb.XGBClassifier()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)

print(confusion_matrix(y_test, y_pred))
print(accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Classification

- Logistic Regression
- K-Nearest Neighbors (KNN)
- Naive Bayes
- SVM
- Decision Trees
- Ensemble Methods
  - What is Bagging & Boosting
  - Random Forests
  - XGBoost
- Evaluating Model Performance

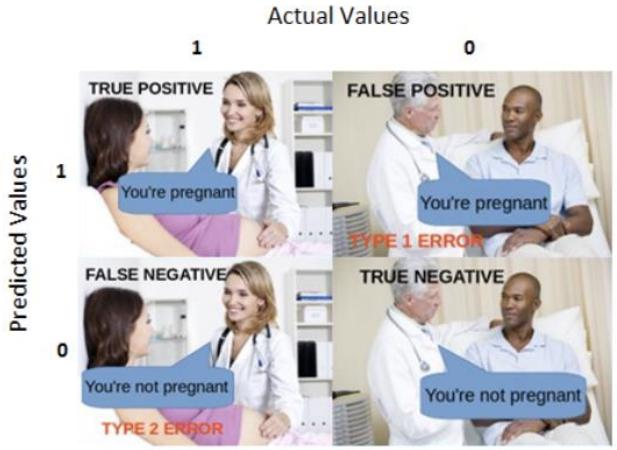


Evaluating Model Performance (Confusion Matrix)



- True Positive (TP) : Observation is positive, and is predicted to be positive.
- False Negative (FN) : Observation is positive, but is predicted negative.
- True Negative (TN) : Observation is negative, and is predicted to be negative.
- False Positive (FP) : Observation is negative, but is predicted positive.

Evaluating Model Performance (Confusion Matrix)



True positives (TP): actuals are positives and are predicted as positives. *You predicted that a woman is pregnant and she actually is.*	**False positives (FP)** - Type 1 Error: actuals are negatives and are predicted as positives. *You predicted that a man is pregnant but he actually is not.*
False negatives (FN) - Type 2 Error: actuals are positives and are predicted as negatives. *You predicted that a woman is not pregnant but she actually is.*	**True negatives (TN):** actuals are negatives and are predicted as positives. *You predicted that a man is not pregnant and he actually is not.*

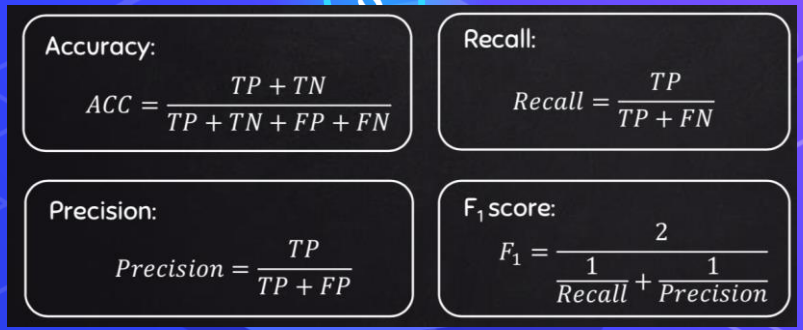
Evaluating Model Performance (Confusion Matrix)

```python
from sklearn.metrics import confusion_matrix


confusion_matrix(y_pred, y_true)

"""
array([[43,  2],
       [11, 87]], dtype=int64)
"""
```

Evaluating Model Performance (Accuracy)

Accuracy

- It’s a measure of how good the model is.
- It’s the ratio of the true predicted values over all the values.



Evaluating Model Performance (Accuracy)

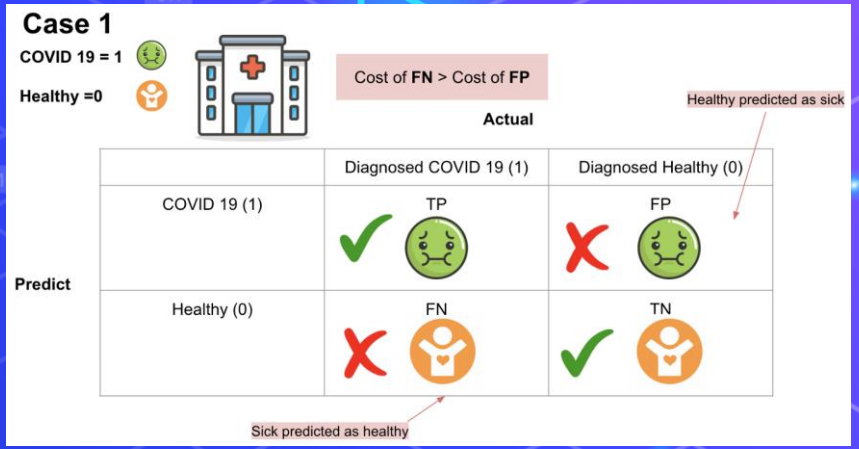
```
1 from sklearn.metrics import accuracy_score
2
3
4 accuracy_score(y_pred, y_true)
5
6 """
7 0.9090909090909091
8 """
```

Evaluating Model Performance (Recall or Sensitivity)

- TP will be that the COVID 19 residents diagnosed with COVID-19.
- TN will be that healthy residents are with good health.
- FP will be that those actually healthy residents are predicted as COVID 19 residents.
- FN will be that those actual COVID 19 residents are predicted as the healthy residents.

which scenario do you think will have the highest cost?

Imagine that if we predict COVID-19 residents as healthy patients and they do not need to quarantine, there would be a massive number of COVID-19 infections. The cost of false negatives is much higher than the cost of false positives.

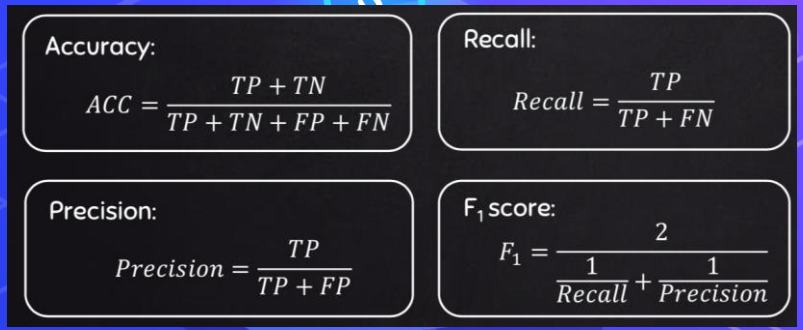


Evaluating Model Performance (Recall or Sensitivity)

Recall (Sensitivity)

- It's a ratio of True Positives to all the positives in your data.
- Low recall: the more False Negatives the model predicts, the lower the recall.

Recall = $\frac{\text{True Positive}}{\text{Predicted Results}}$ or $\frac{\text{True Positive}}{\text{True Positive + False Negative}}$



Evaluating Model Performance (Recall or Sensitivity)

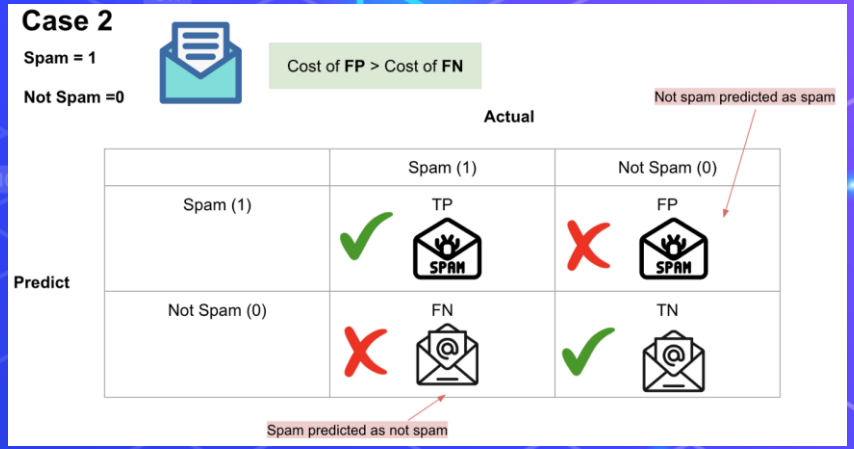
```python
from sklearn.metrics import recall_score


recall_score(y_pred, y_true)

"""
0.9090909090909091
"""
```

Evaluating Model Performance (Precision)

- TP will be that spam emails are placed in the spam folder.
- TN will be that important emails are received.
- FP will be that important emails are placed in the spam folder.
- FN will be that spam emails are received.

which scenario do you think will have the highest cost?

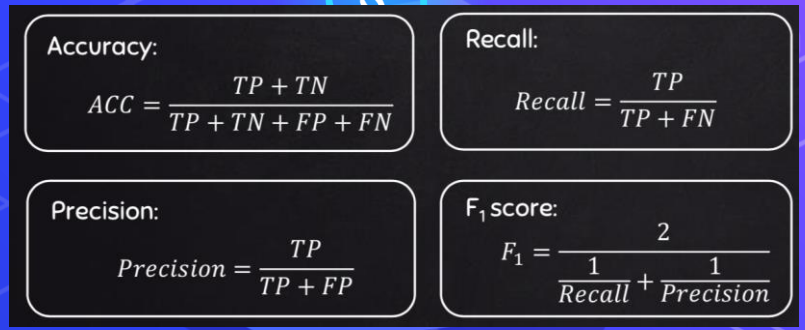
Well, since missing important emails will clearly be more of a problem than receiving spam, we can say that in this case, FP will have a higher cost than FN.



Evaluating Model Performance (Precision)

Precision

- It's a ratio of True Positives to all the positives predicted by the model.
- Low precision: the more False positives the model predicts, the lower the precision.



$$\text{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

Evaluating Model Performance (Precision)

```python
from sklearn.metrics import precision_score


precision_score(y_pred, y_true)

"""
0.9090909090909091
"""
```

Evaluating Model Performance

Case 1

COVID 19/ Healthy

Cost of **FN** > Cost of **FP**

Recall

$$recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

Case 2

Spam/Not Spam

Cost of **FP** > Cost of **FN**

Precision

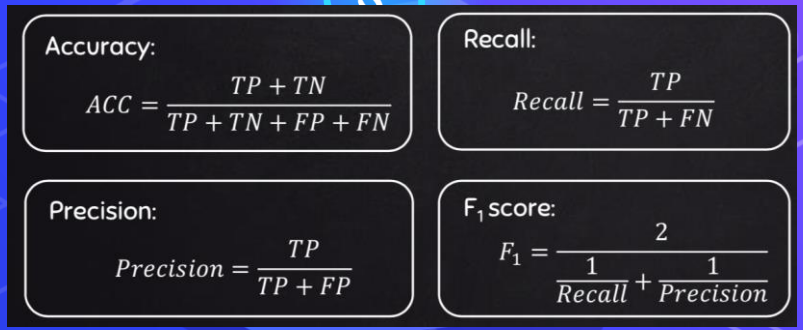
$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

Evaluating Model Performance (F-1 Score)

F-1 Score

- F-Score called the Harmonic mean of precision and recall.
- F-Score provides a single score that balances both the concerns of precision and recall in one number.
- A good F1 score means that you have low false positives and low false negatives.
- An F1 score is considered perfect when it's 1, while the model is a total failure when it's 0.

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$



Evaluating Model Performance (F-1 Score)

```
1 from sklearn.metrics import f1_score
2
3
4 f1_score(y_pred, y_true)
5
6 """
7 0.9090909090909091
8 """
```

Evaluating Model Performance (Classification Report)

```
from sklearn.metrics import classification_report


classification_report(y_true, y_pred)

"""
              precision    recall  f1-score   support
     class 0       0.50      1.00      0.67         1
     class 1       0.00      0.00      0.00         1
     class 2       1.00      0.67      0.80         3
"""
```

Evaluating Model Performance (F-beta Score)

F-beta Score

- Generalization of F-Score where beta is a parameter to weight the result of the precision or recall.
- Beta = 0.5 to focus more on precision, It's called F-0.5 Score.
- Beta = 2 to focus more on recall, It's called F-2 Score.
- Beta value should be used depending on your domain knowledge business case.

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}.$$

Evaluating Model Performance (F-beta Score)

```python
from sklearn.metrics import fbeta_score


fbeta_score(y_true, y_pred, beta=0.5)

"""
0.9090909090909091
"""
```

Agenda

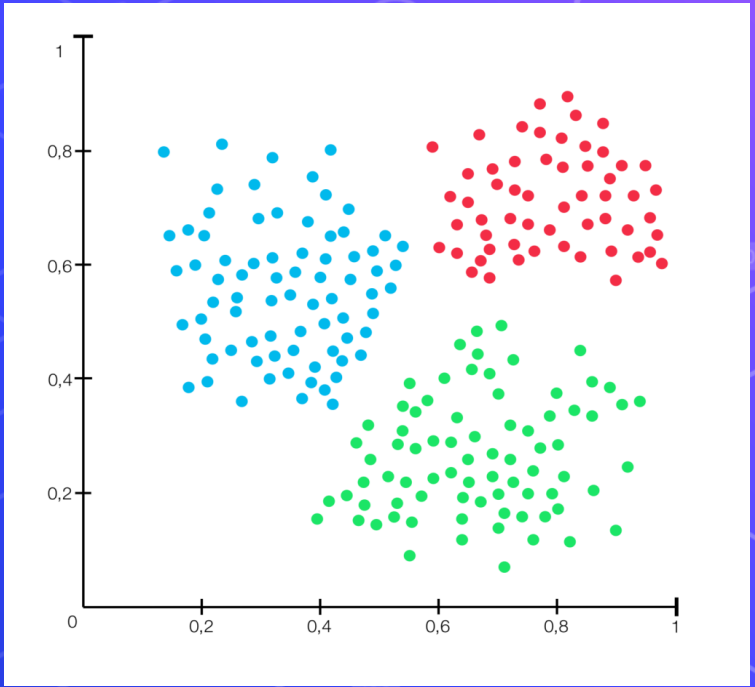
- What is Machine Learning
- Supervised Learning
  - Regression
  - Classification
- **Unsupervised Learning**
  - **Clustering**
  - Dimension Reduction
- Model Selection & Evaluation
  - Cross Validation
  - Hyperparameter Tuning
- Intro to Deep Learning

Clustering

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups.

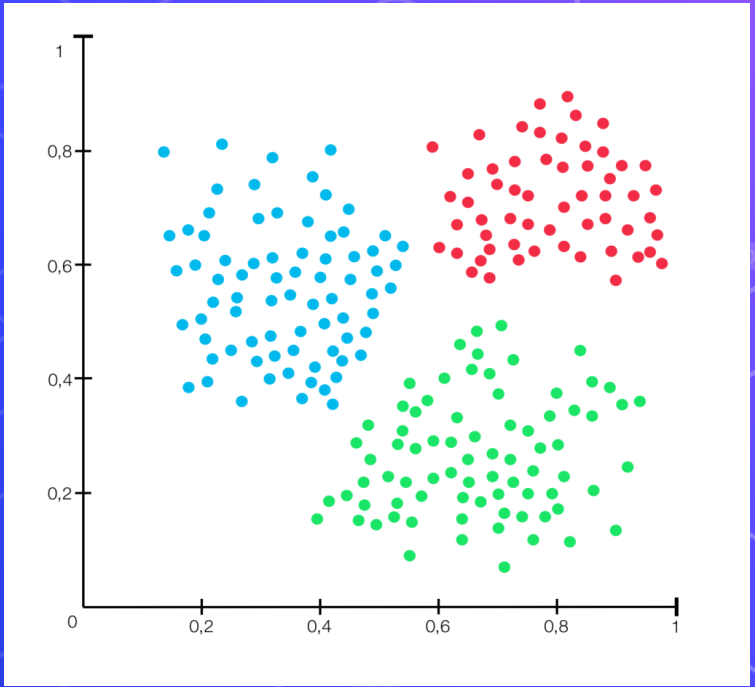
For example,
it can be used to cluster the university students to many segments depending on features like scores, departments, etc.

- University students clustering
- Mall customers segmentation
- Recommendation systems
- ...



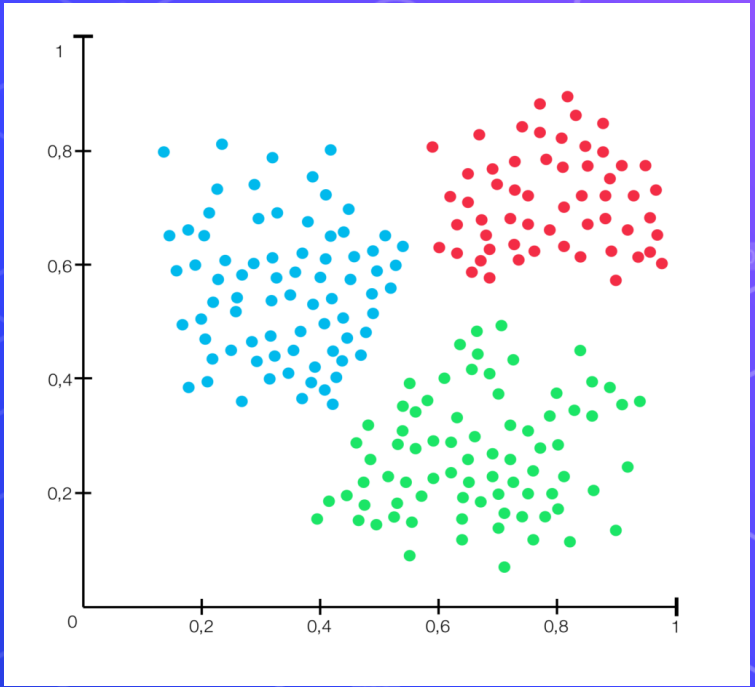
Clustering

- KMeans
- Hierarchical Clustering
- Density Based Clustering – DBSCAN

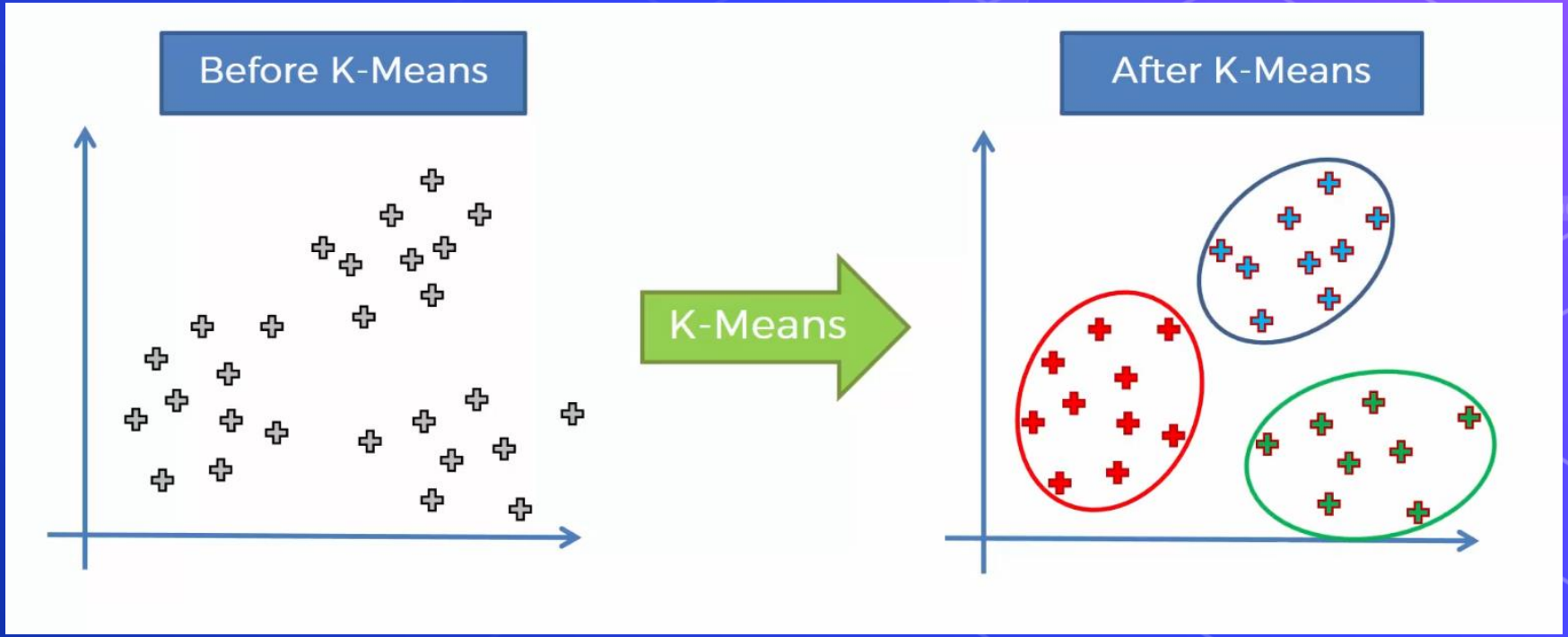


Clustering

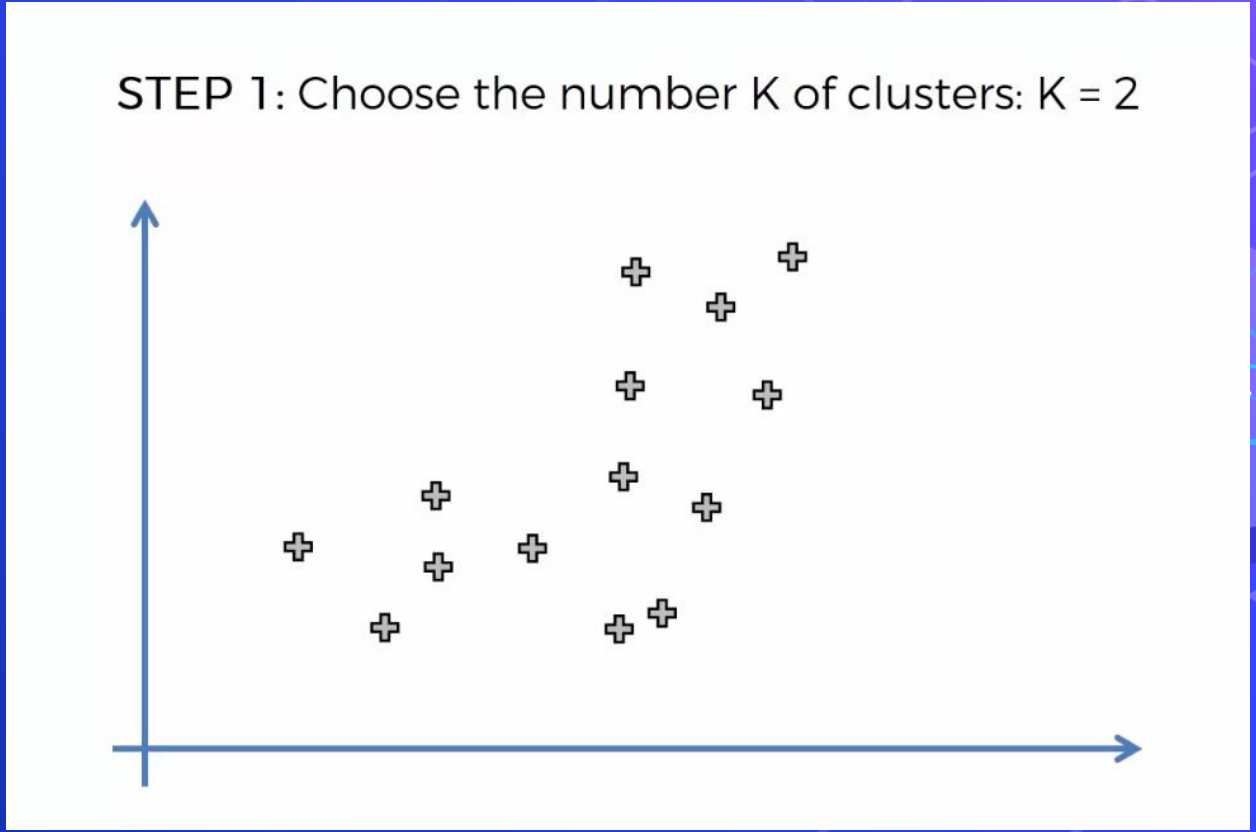
- KMeans
- Hierarchical Clustering
- Density Based Clustering – DBSCAN



KMeans

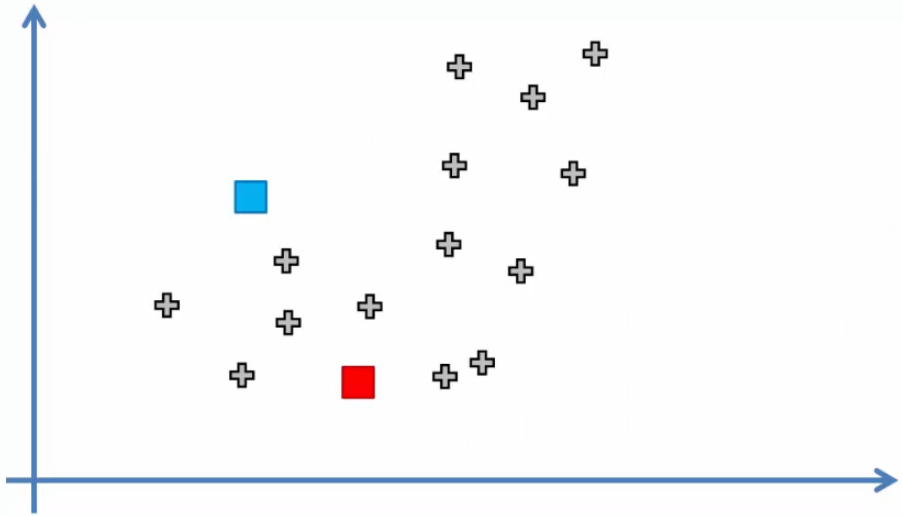


KMeans



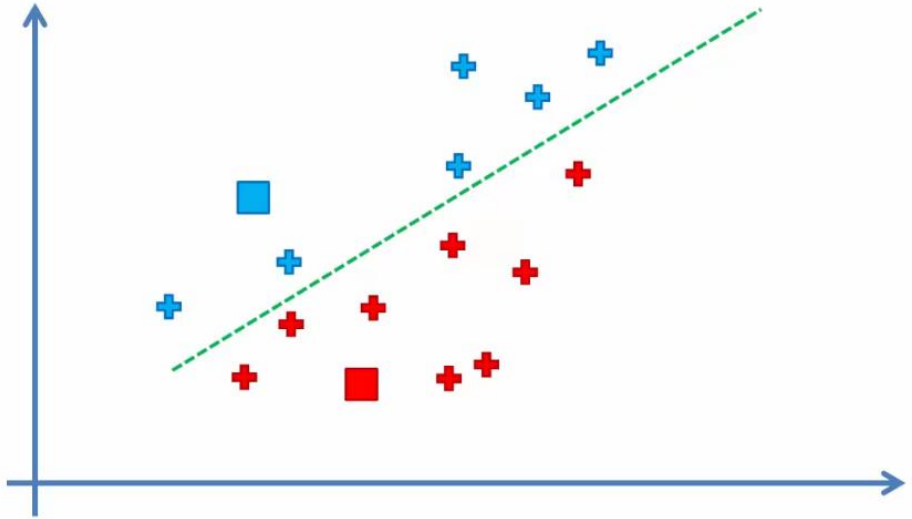
KMeans

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



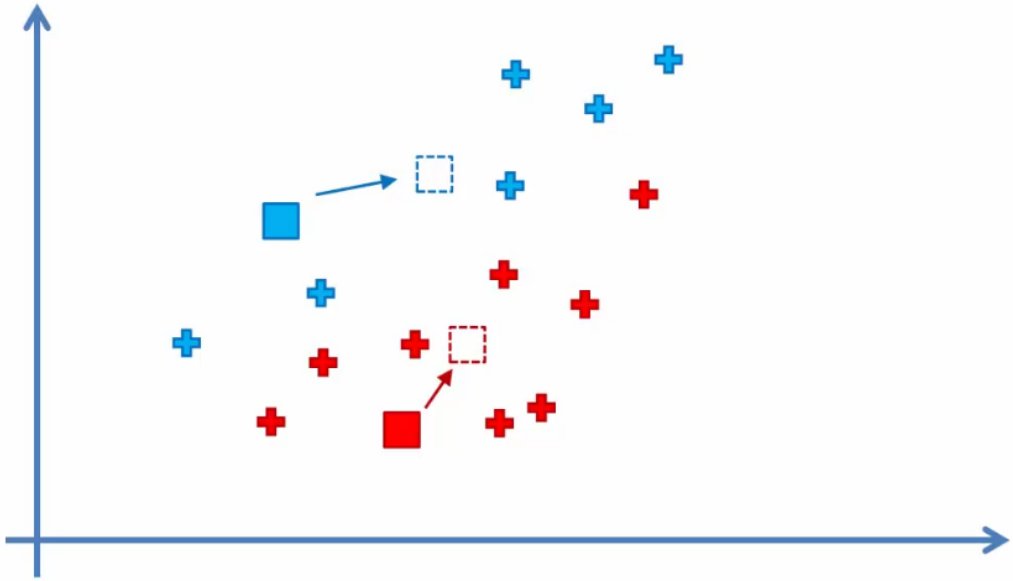
KMeans

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



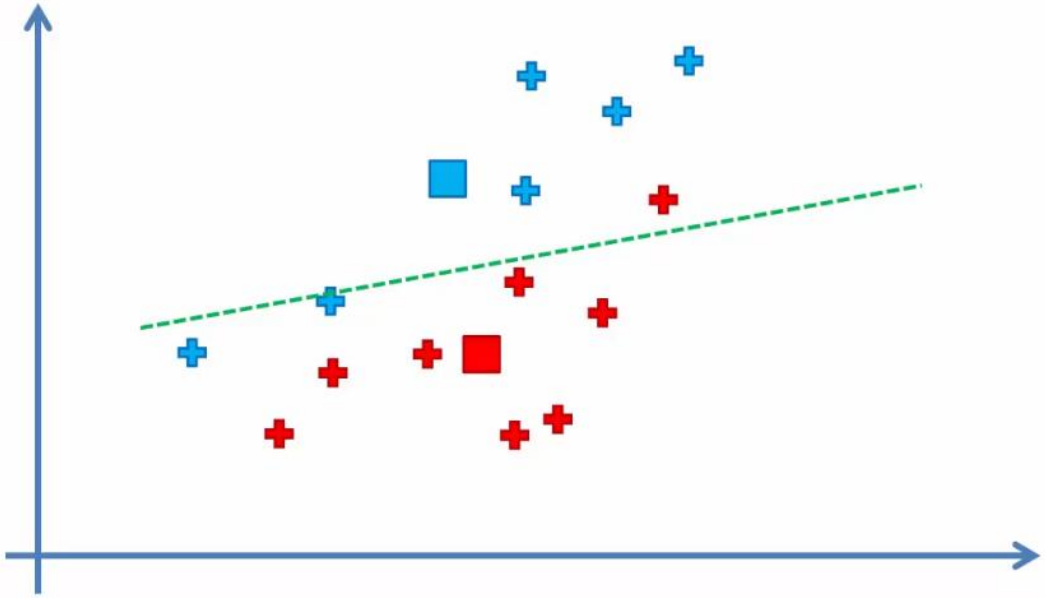
KMeans

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



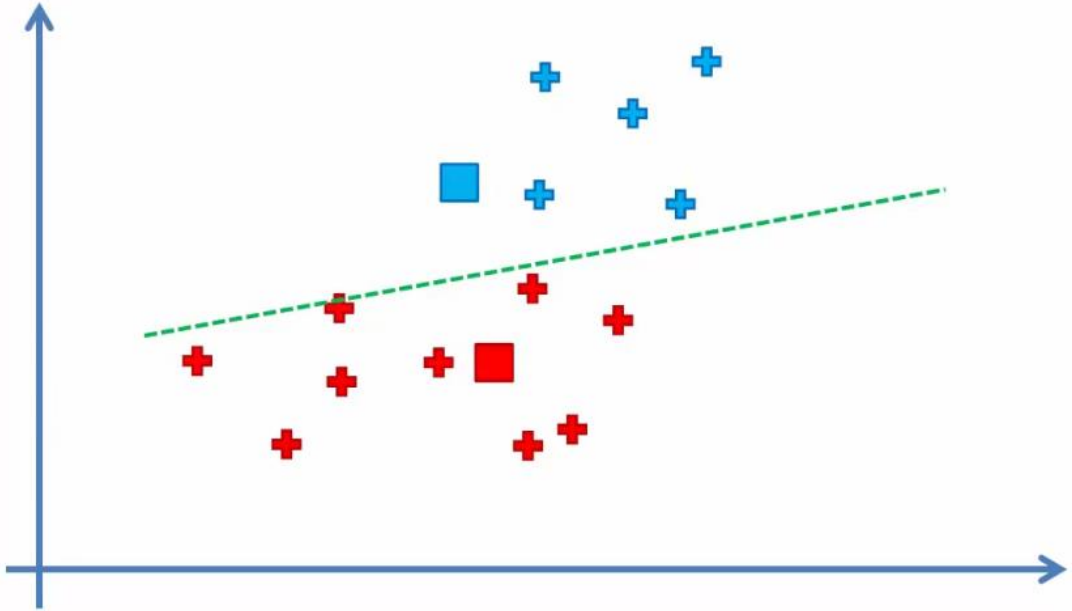
KMeans

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

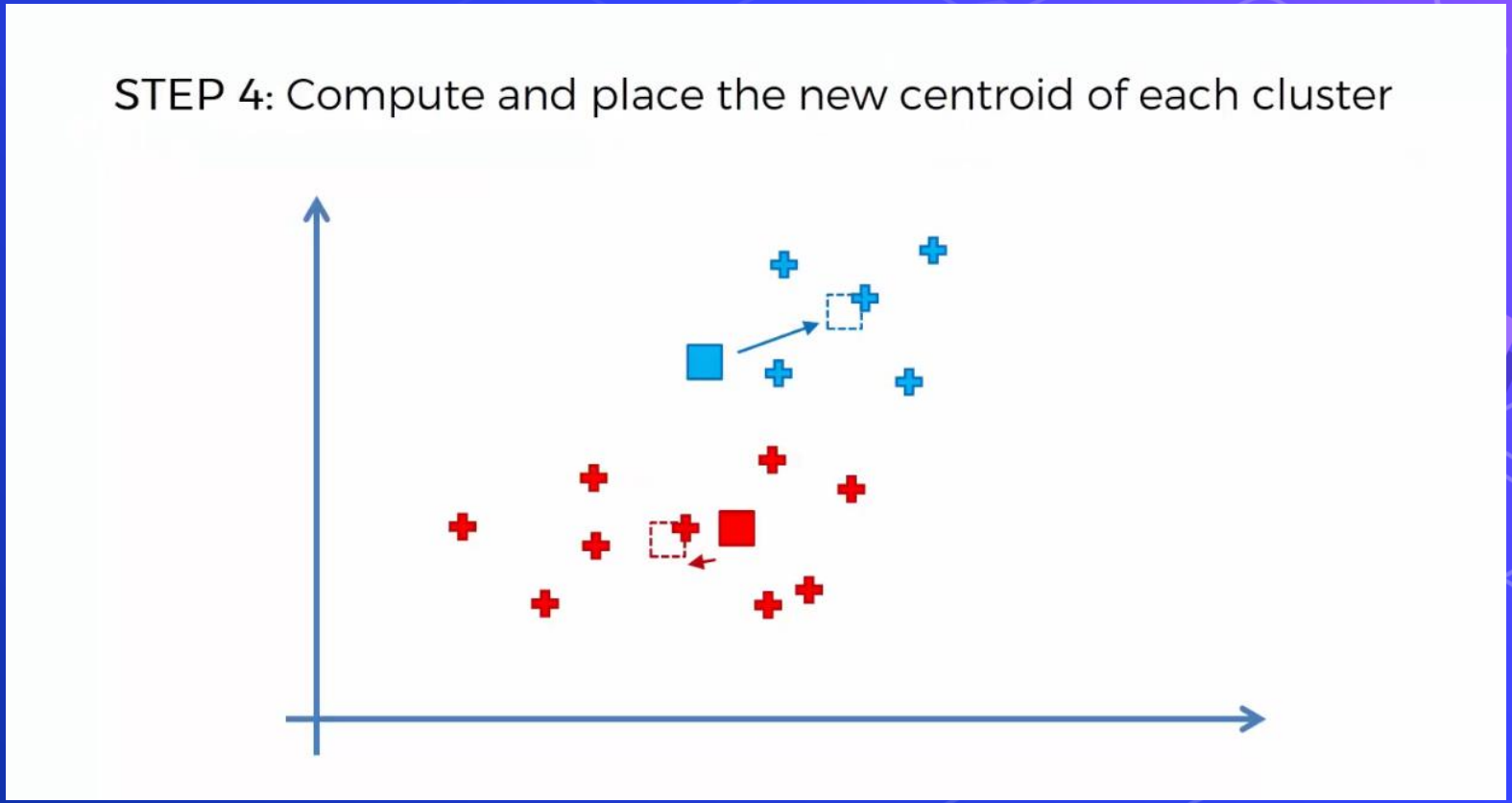


KMeans

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

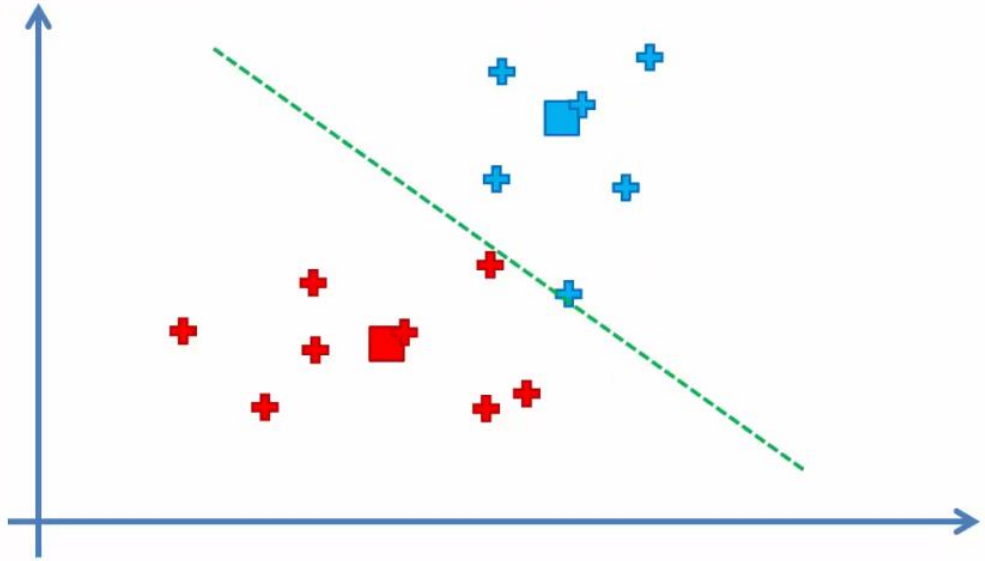


KMeans



KMeans

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

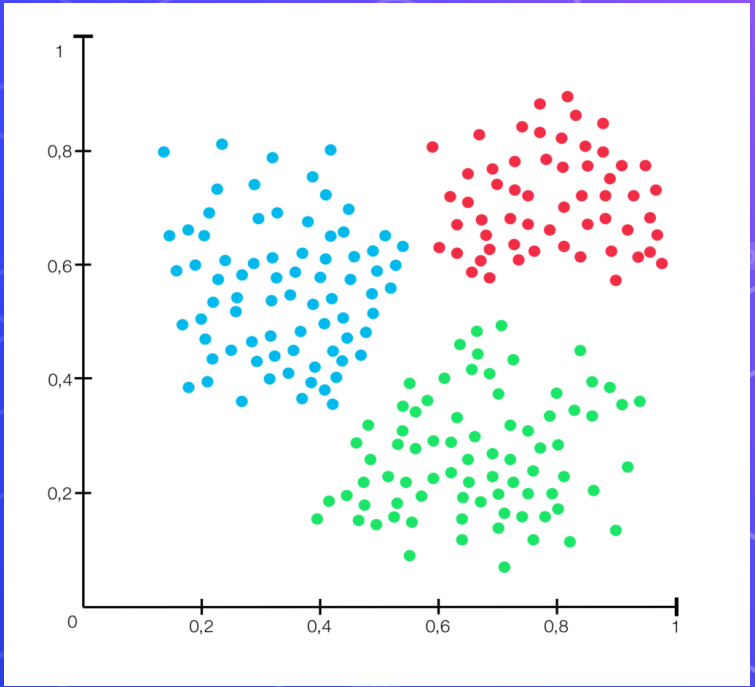


KMeans

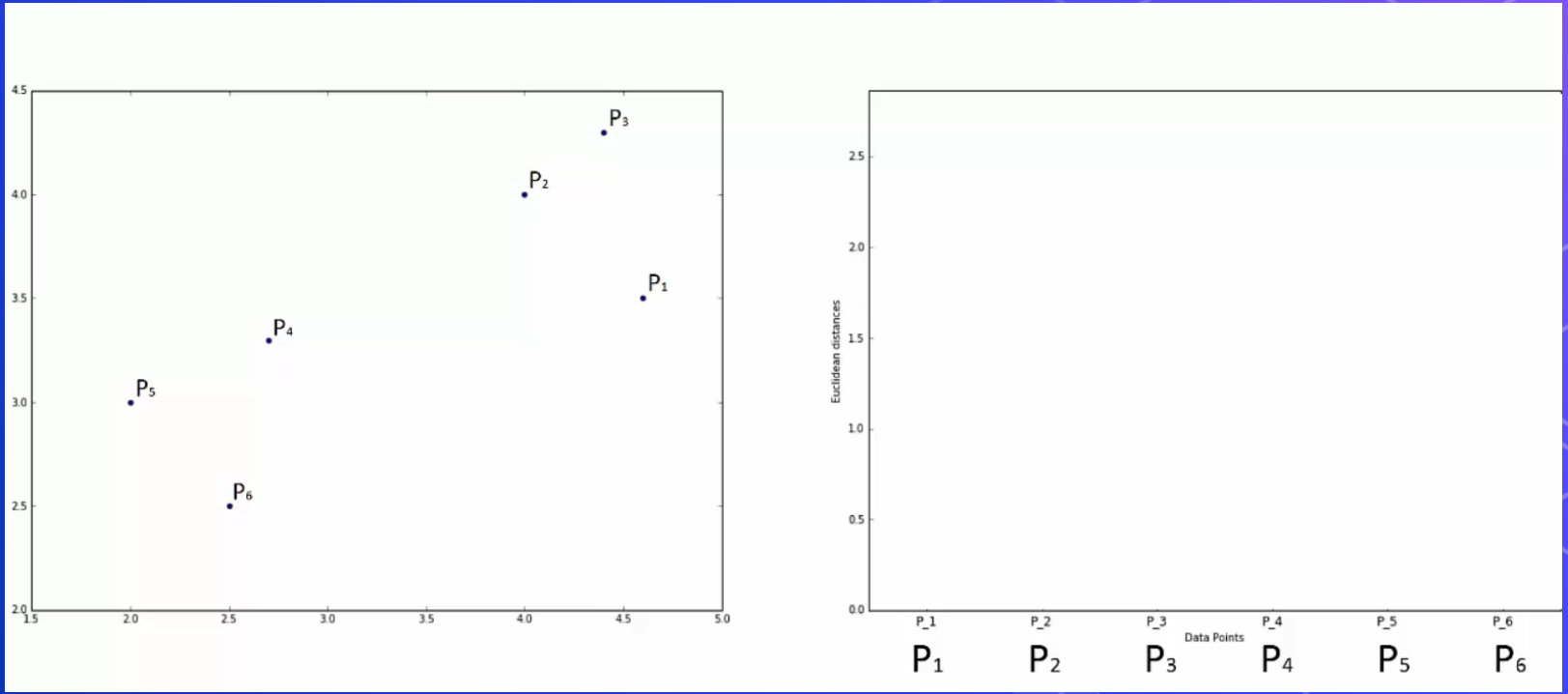
```
1 from sklearn.cluster import KMeans
2
3 kmeans = KMeans(n_clusters=3)
4 kmeans.fit(X)
5 kmeans.predict(X)
```

Clustering

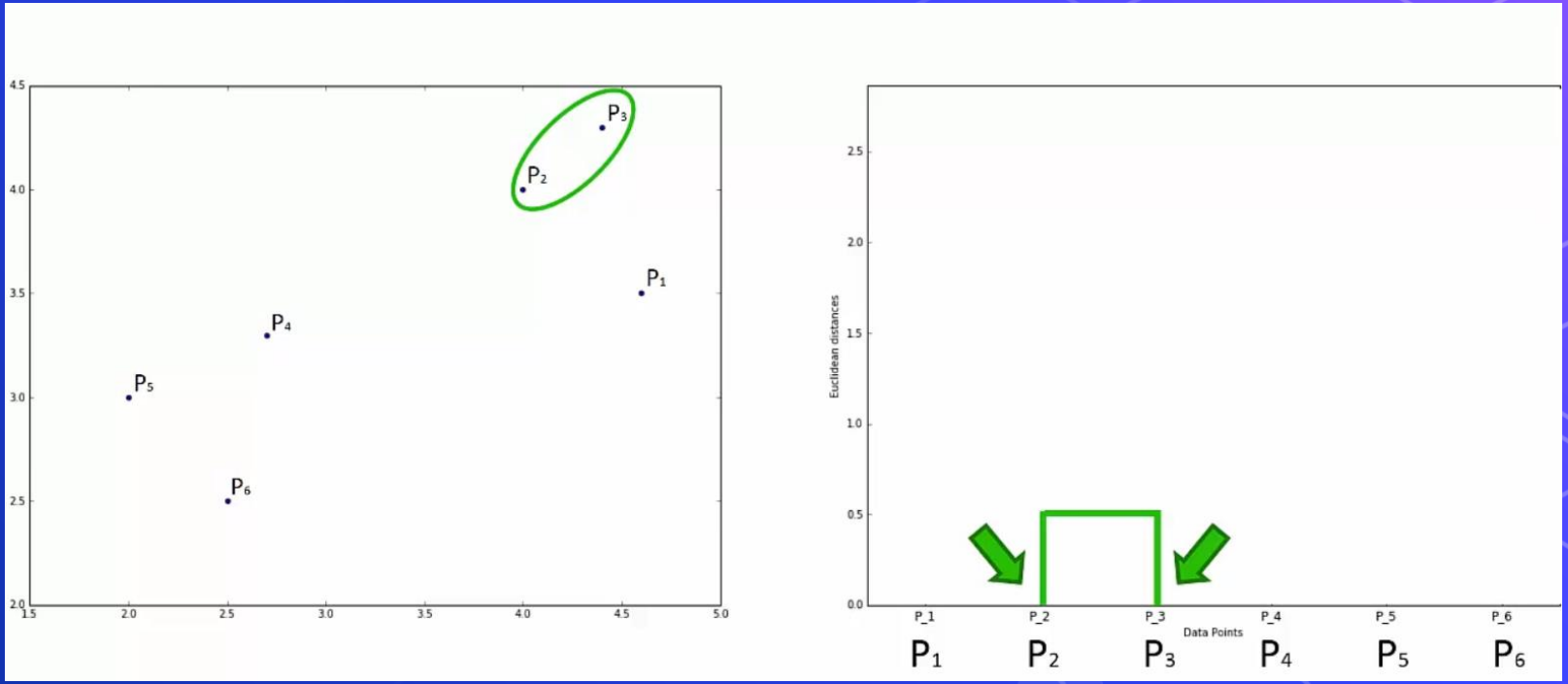
- KMeans
- Hierarchical Clustering
- Density Based Clustering – DBSCAN



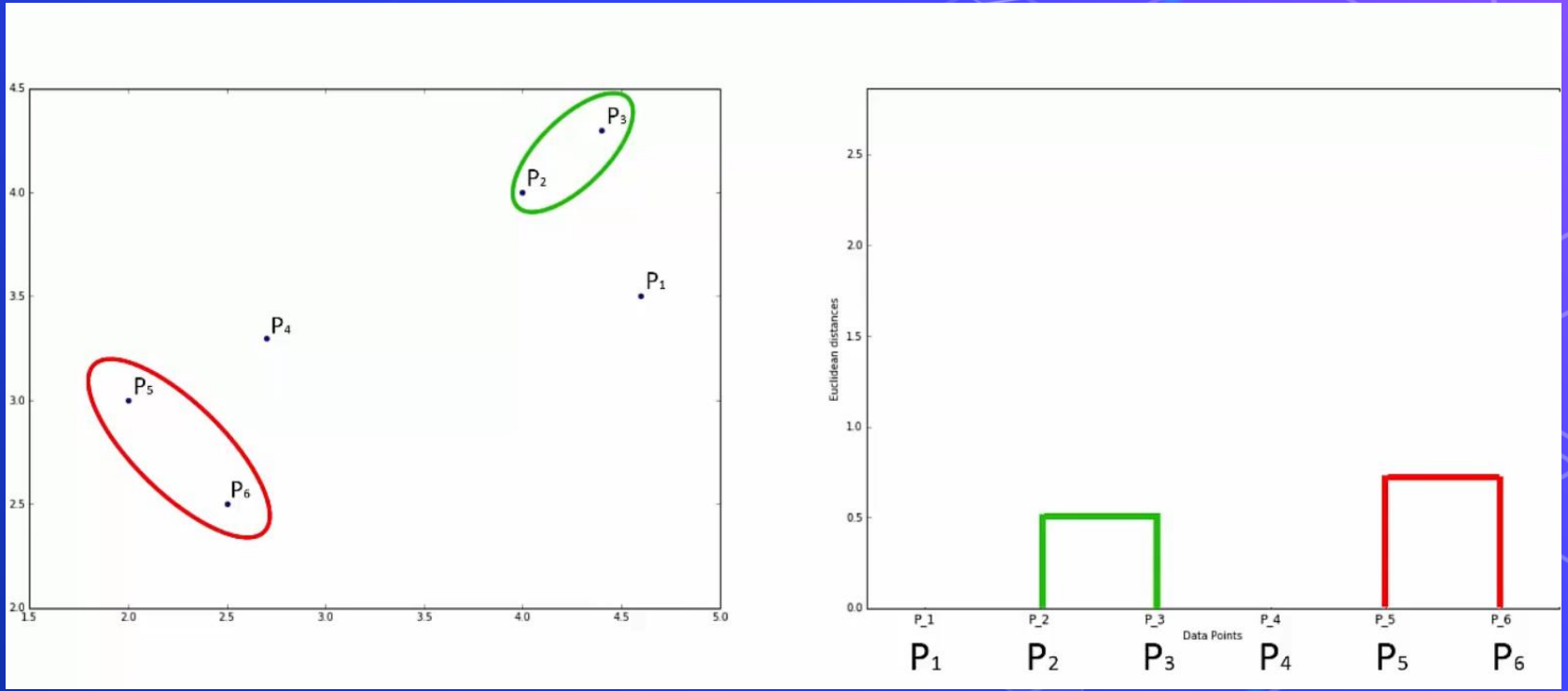
Hierarchical Clustering with dendrogram



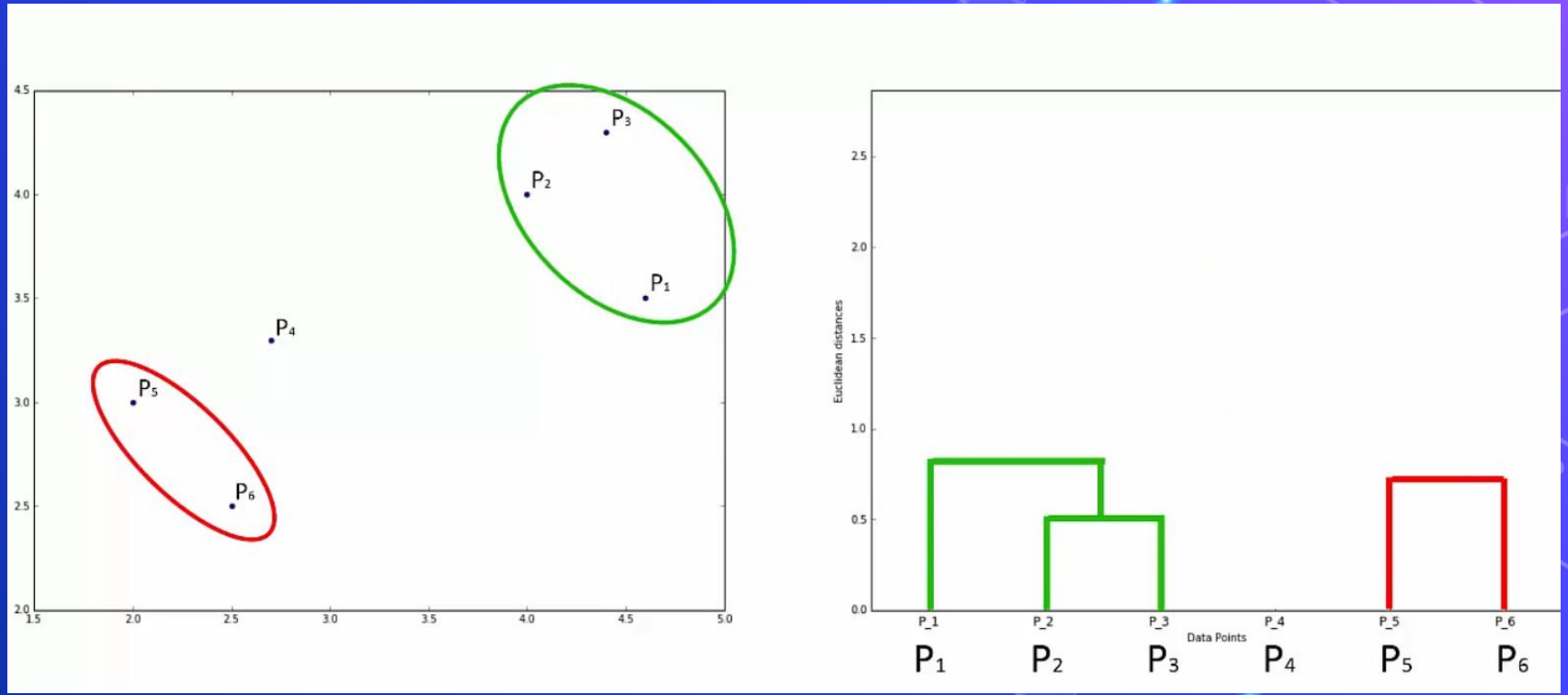
Hierarchical Clustering with dendrogram



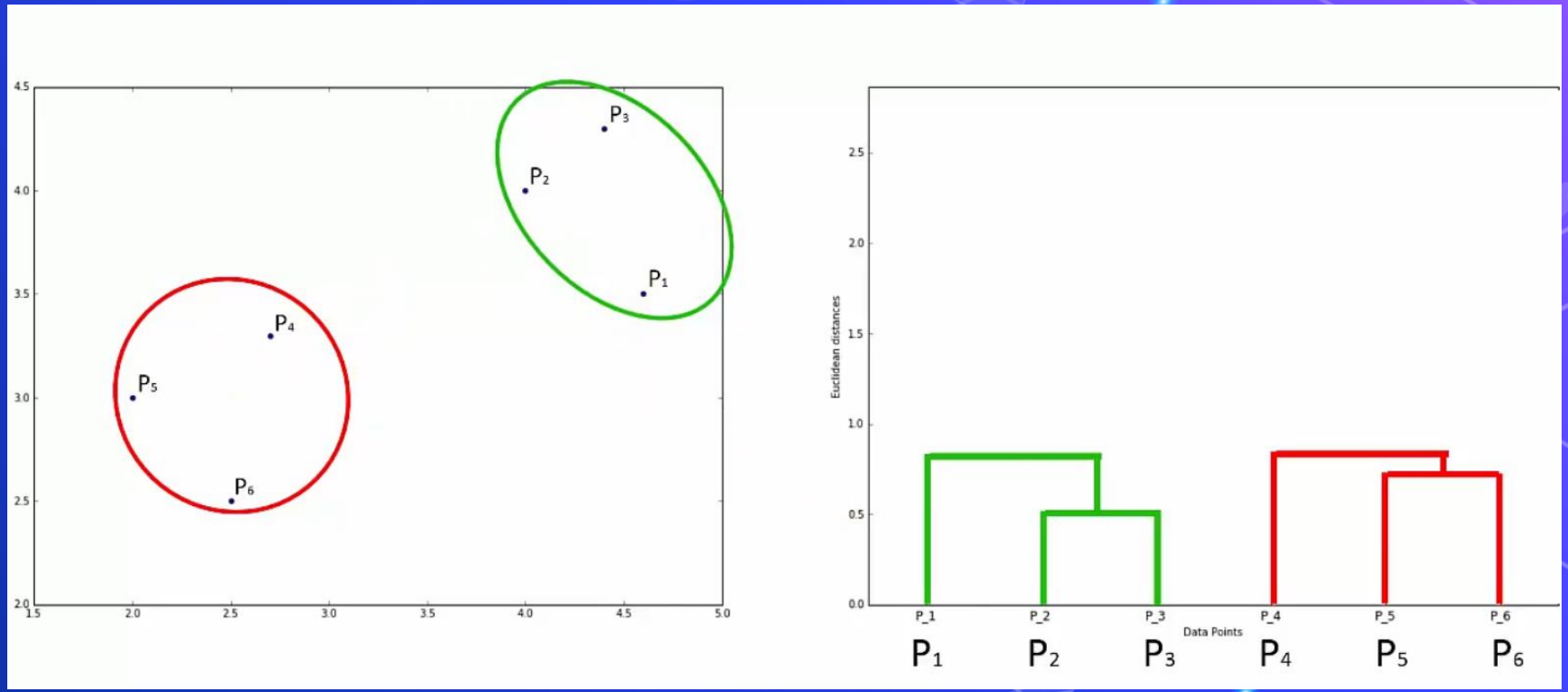
Hierarchical Clustering with dendrogram



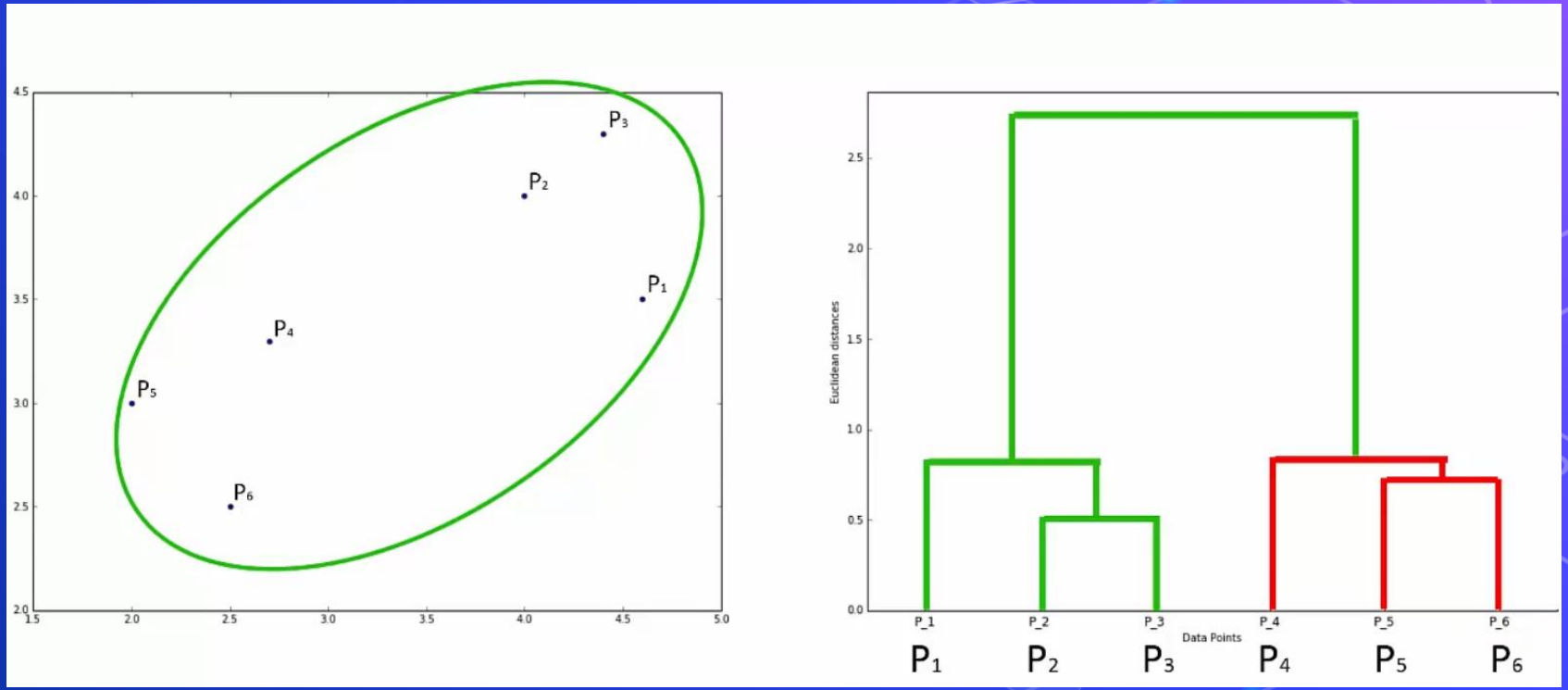
Hierarchical Clustering with dendrogram



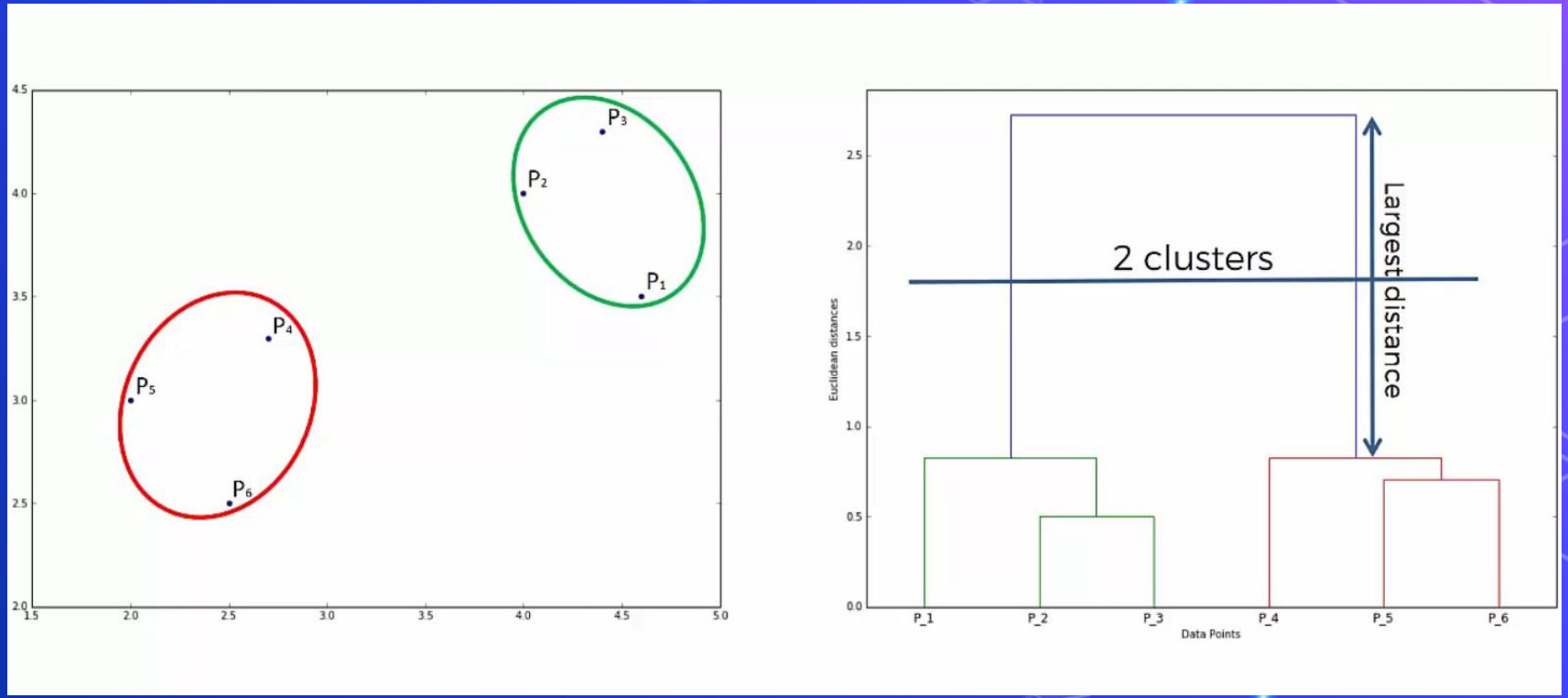
Hierarchical Clustering with dendrogram



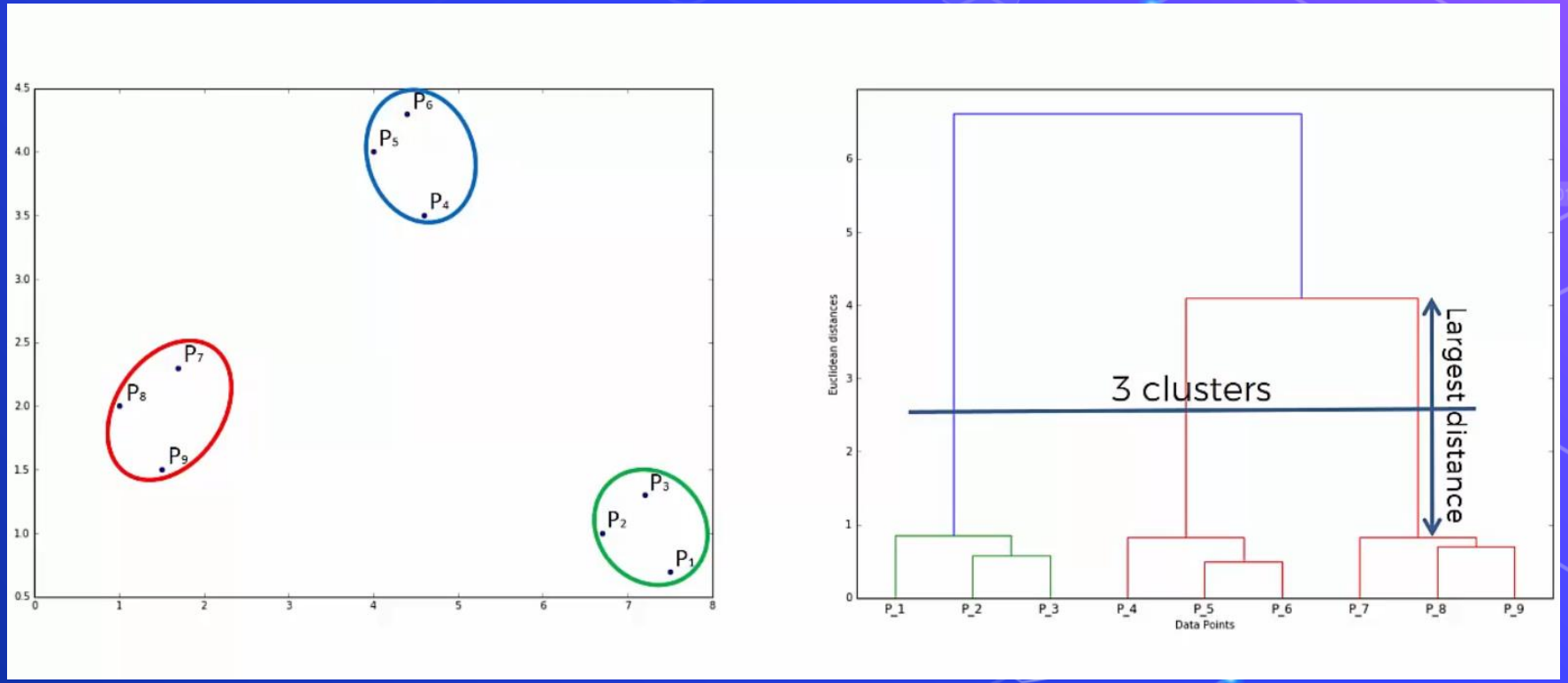
Hierarchical Clustering with dendrogram



Hierarchical Clustering with dendrogram



Hierarchical Clustering with dendrogram

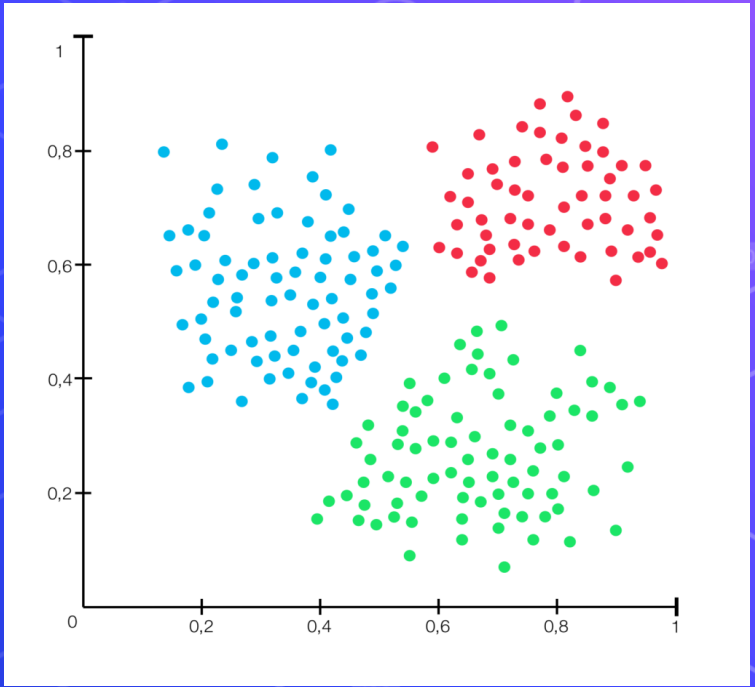


Hierarchical Clustering with dendrogram

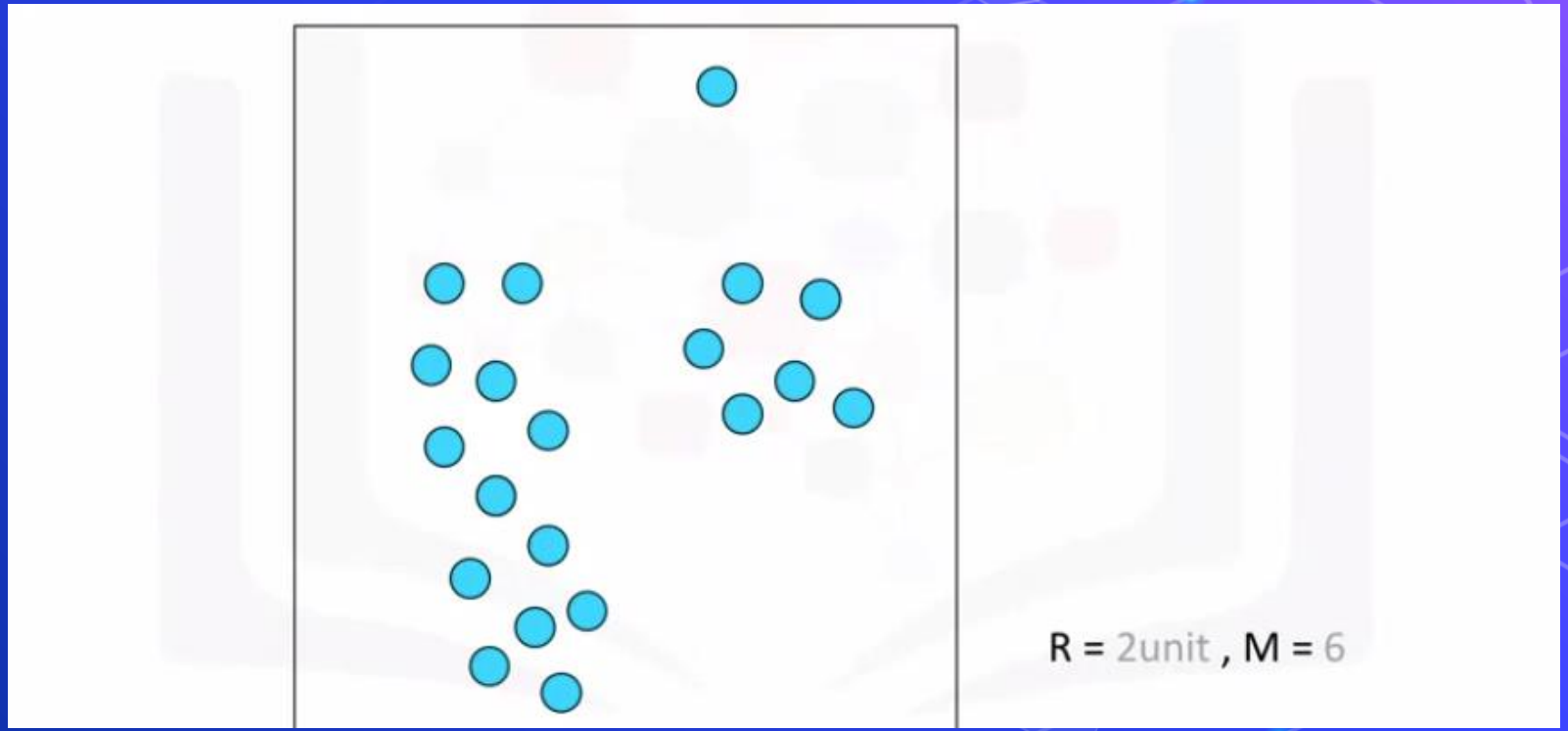
```python
import scipy.cluster.hierarchy as sch
from sklearn.cluster import AgglomerativeClustering

# visualize dendrogram
dendrogram = sch.dendrogram(sch.linkage(x, method='ward'))

# train model
model = AgglomerativeClustering(n_clusters=3)
y_labels = model.fit_predict(x)
```

Clustering

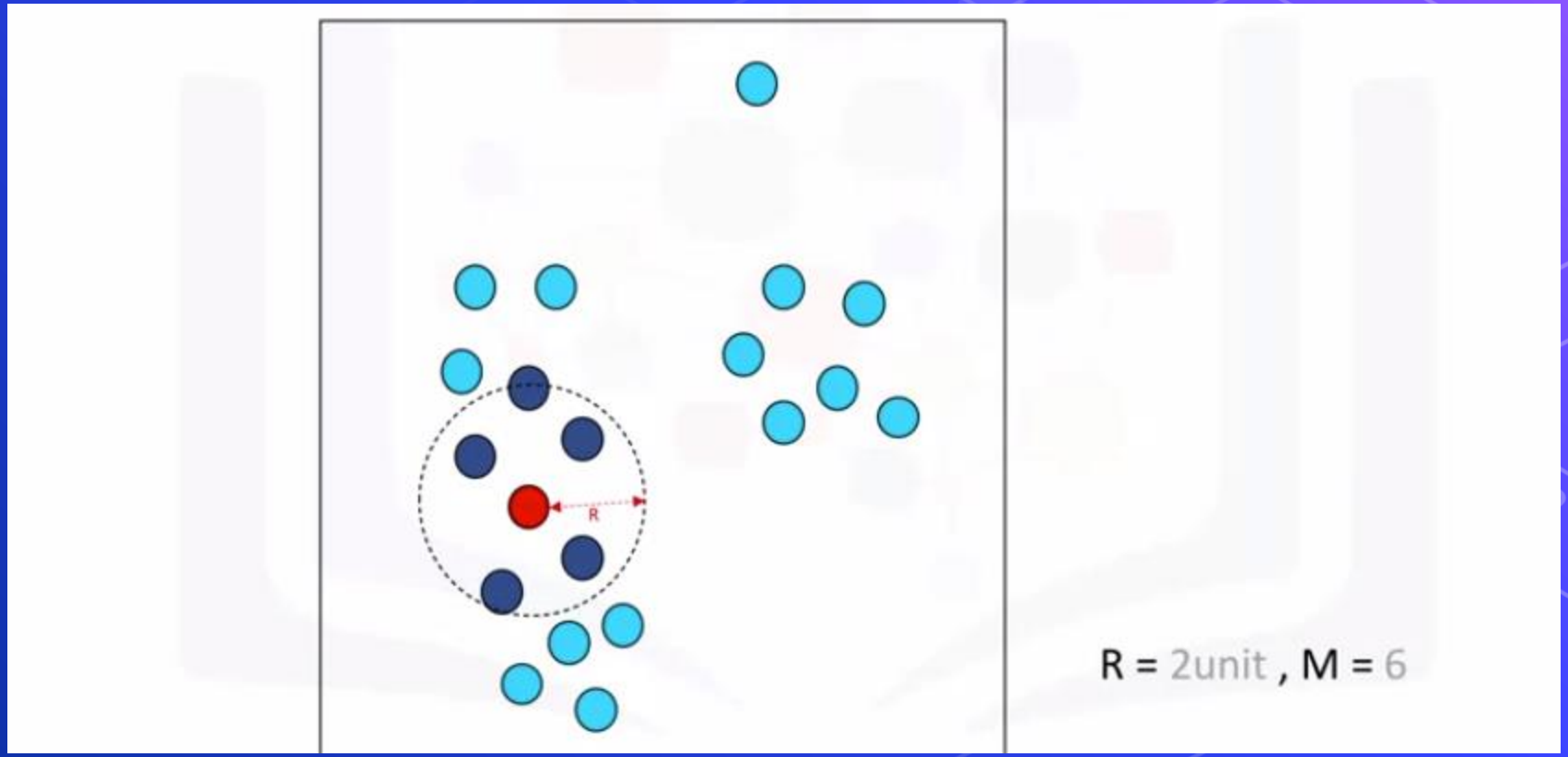
- KMeans
- Hierarchical Clustering
- **Density Based Clustering – DBSCAN**



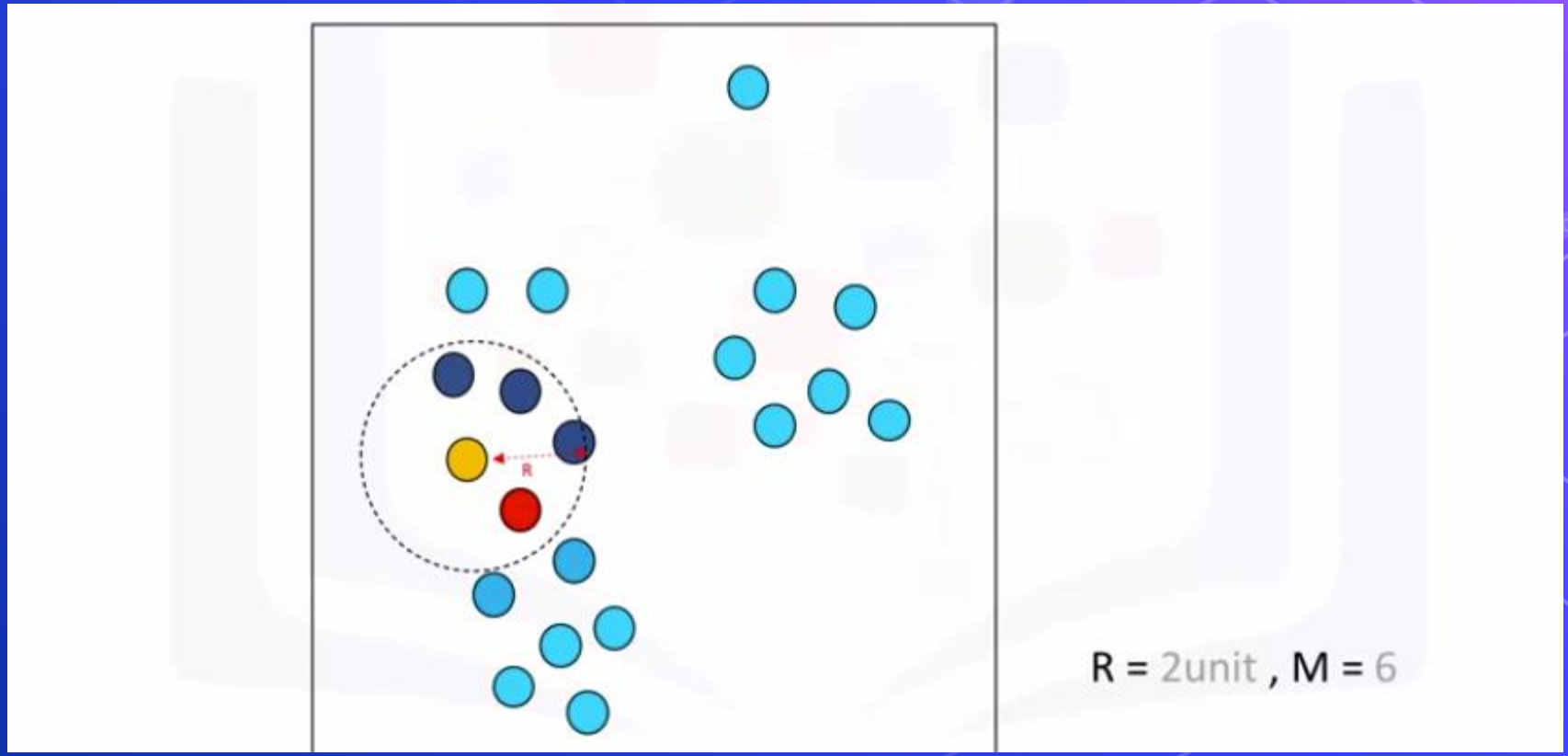
Density Based Clustering - DBSCAN



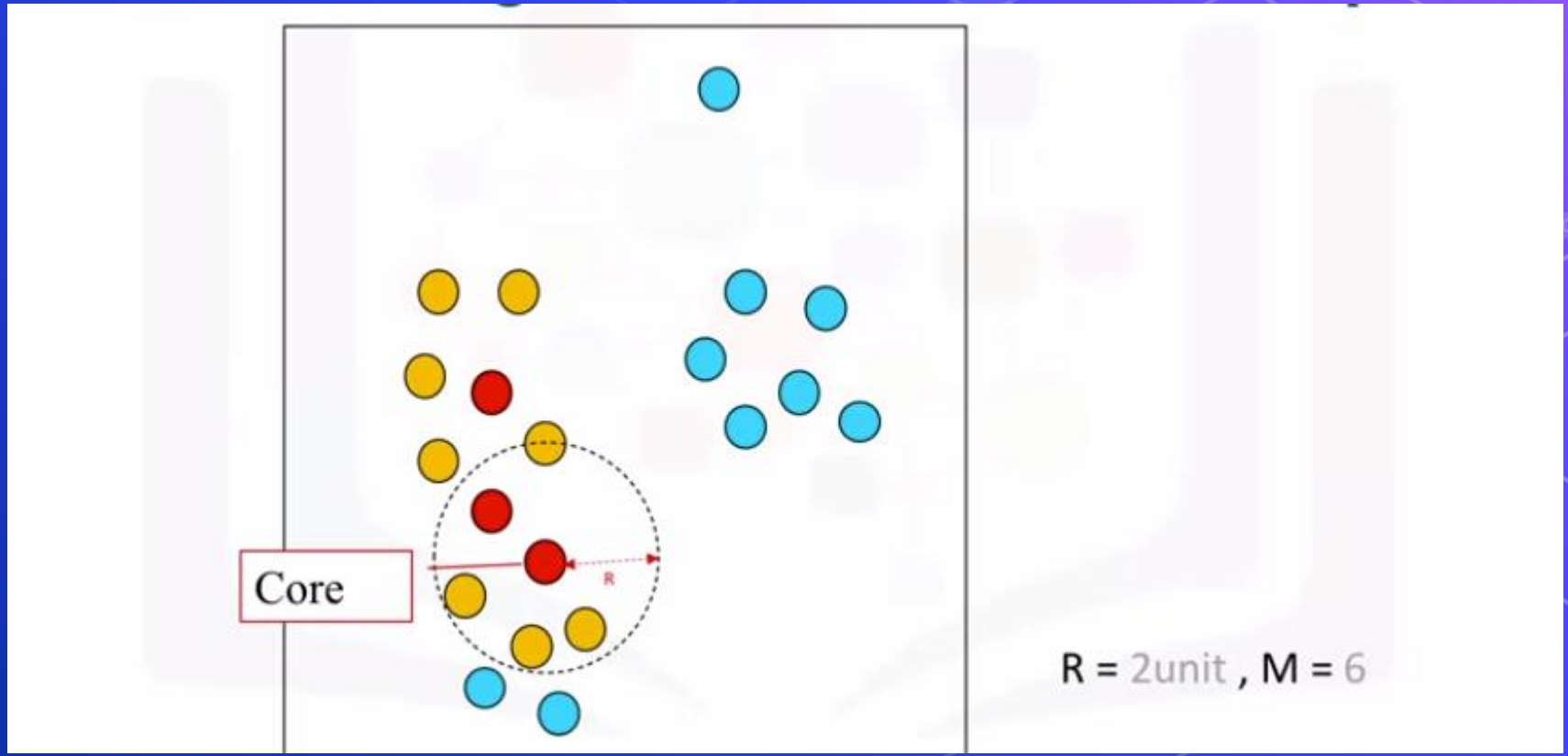
Density Based Clustering – DBSCAN (Core point)



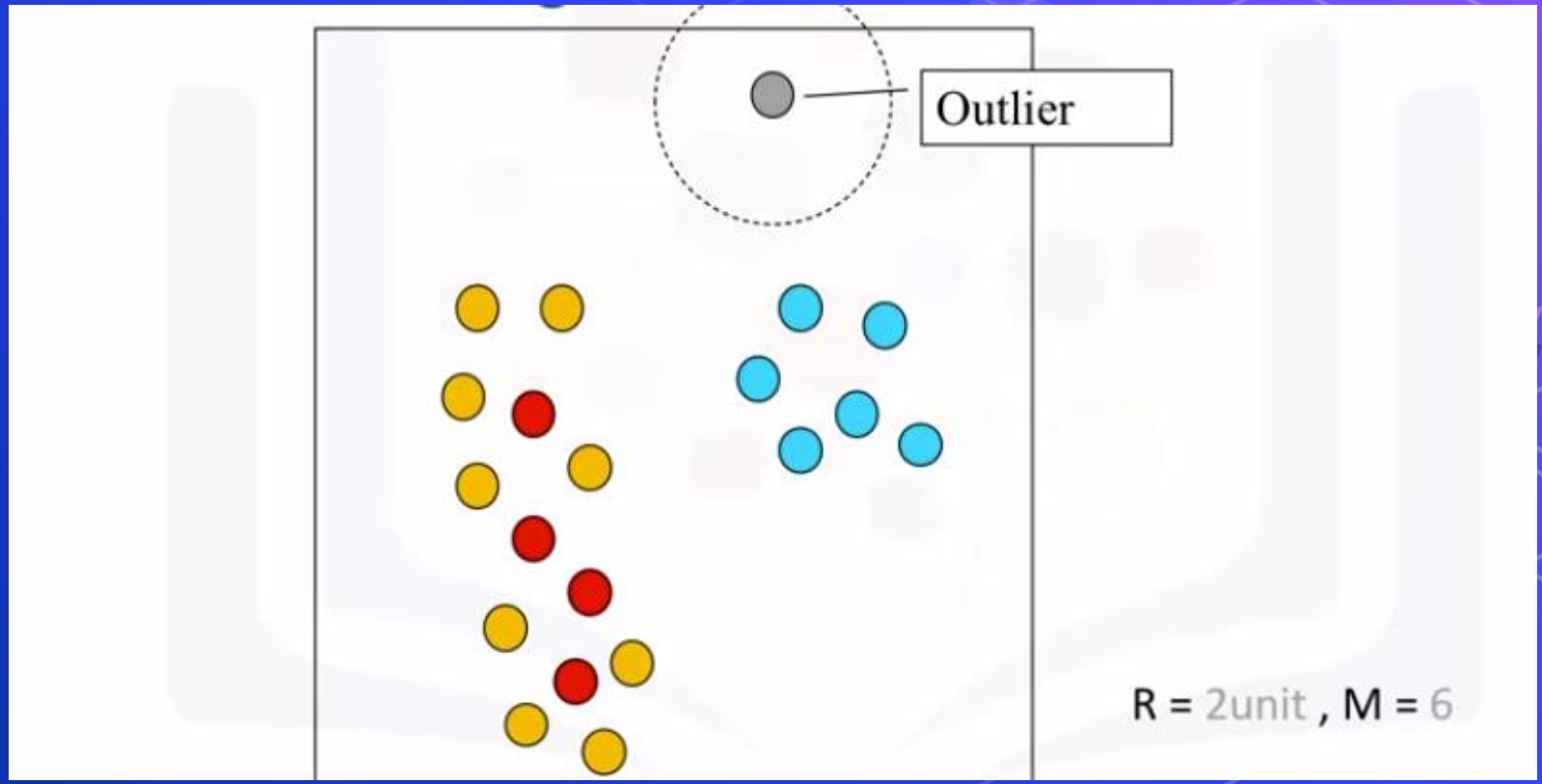
Density Based Clustering – DBSCAN (Border point)



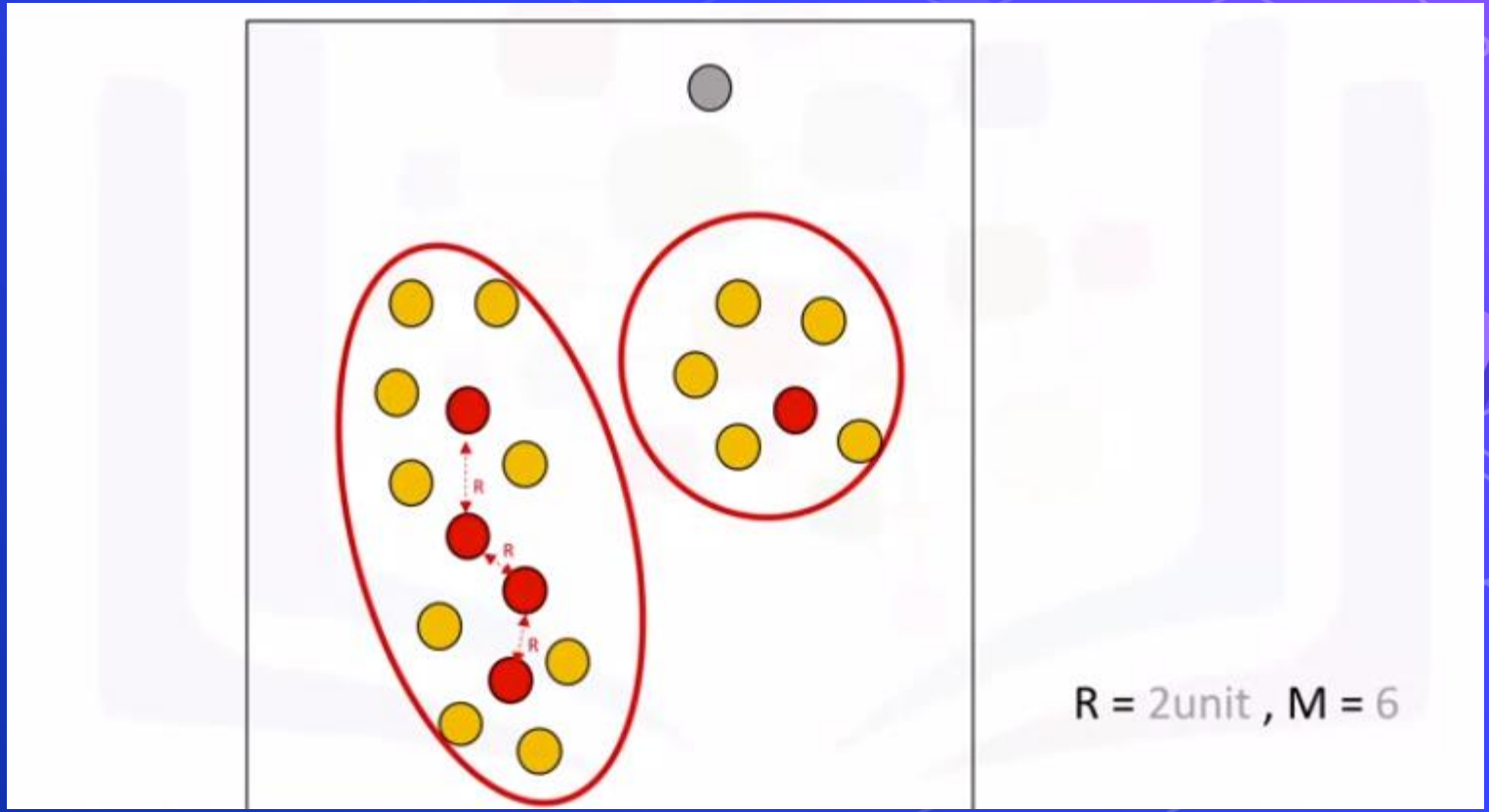
Density Based Clustering - DBSCAN



Density Based Clustering - DBSCAN



Density Based Clustering - DBSCAN



Density Based Clustering - DBSCAN

```
1 from sklearn.cluster import DBSCAN
2
3 model.DBSCAN(eps=0.3, min_samples=10)
4 y_labels = model.fit_predict(x)
```

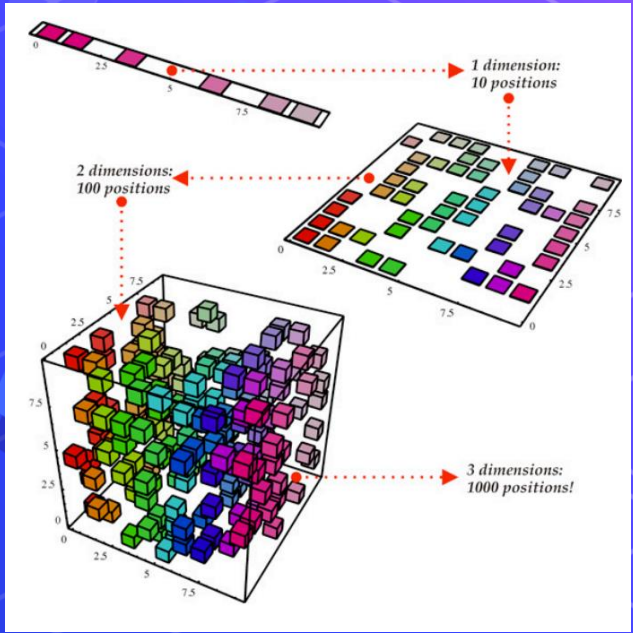
Agenda

- What is Machine Learning
- Supervised Learning
  - Regression
  - Classification
- **Unsupervised Learning**
  - Clustering
  - **Dimension Reduction**
- Model Selection & Evaluation
  - Cross Validation
  - Hyperparameter Tuning
- Intro to Deep Learning

Dimension Reduction

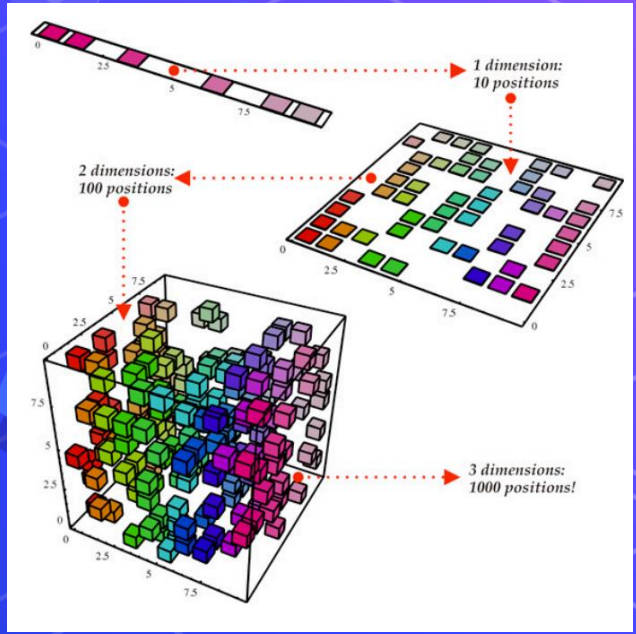
is the transformation of data from a high-dimensional space into a low-dimensional space so that the low-dimensional representation retains some meaningful properties of the original data

For example,
it can be used as data compression to reduce size of data for easy processing.



Dimension Reduction

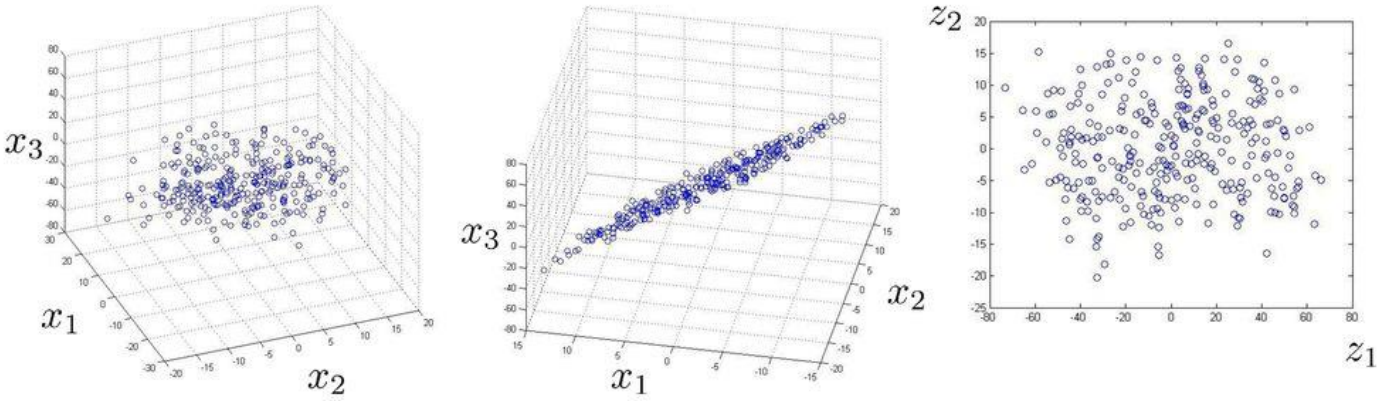
- PCA



PCA

Data Compression

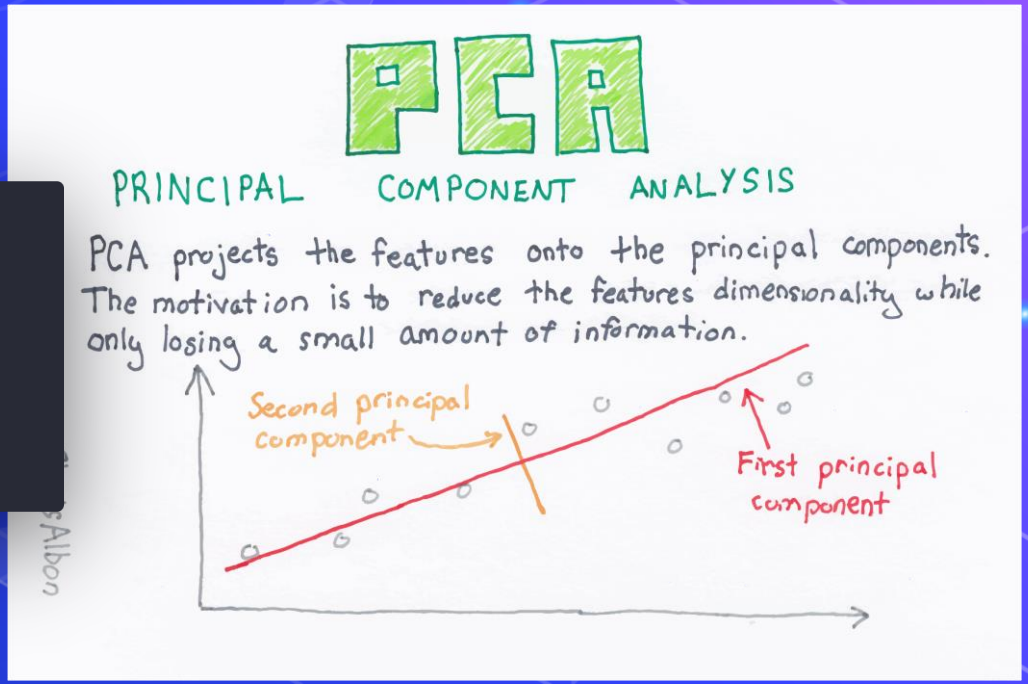
Reduce data from 3D to 2D



Andrew Ng

PCA

```python
from sklearn.decomposition import PCA

X = # feature vector

pca = PCA(0.9)
X = pca.fit_transform(X)
pca.explained_variance_ratio_
```



https://en.wikipedia.org/wiki/Principal_component_analysis

Agenda

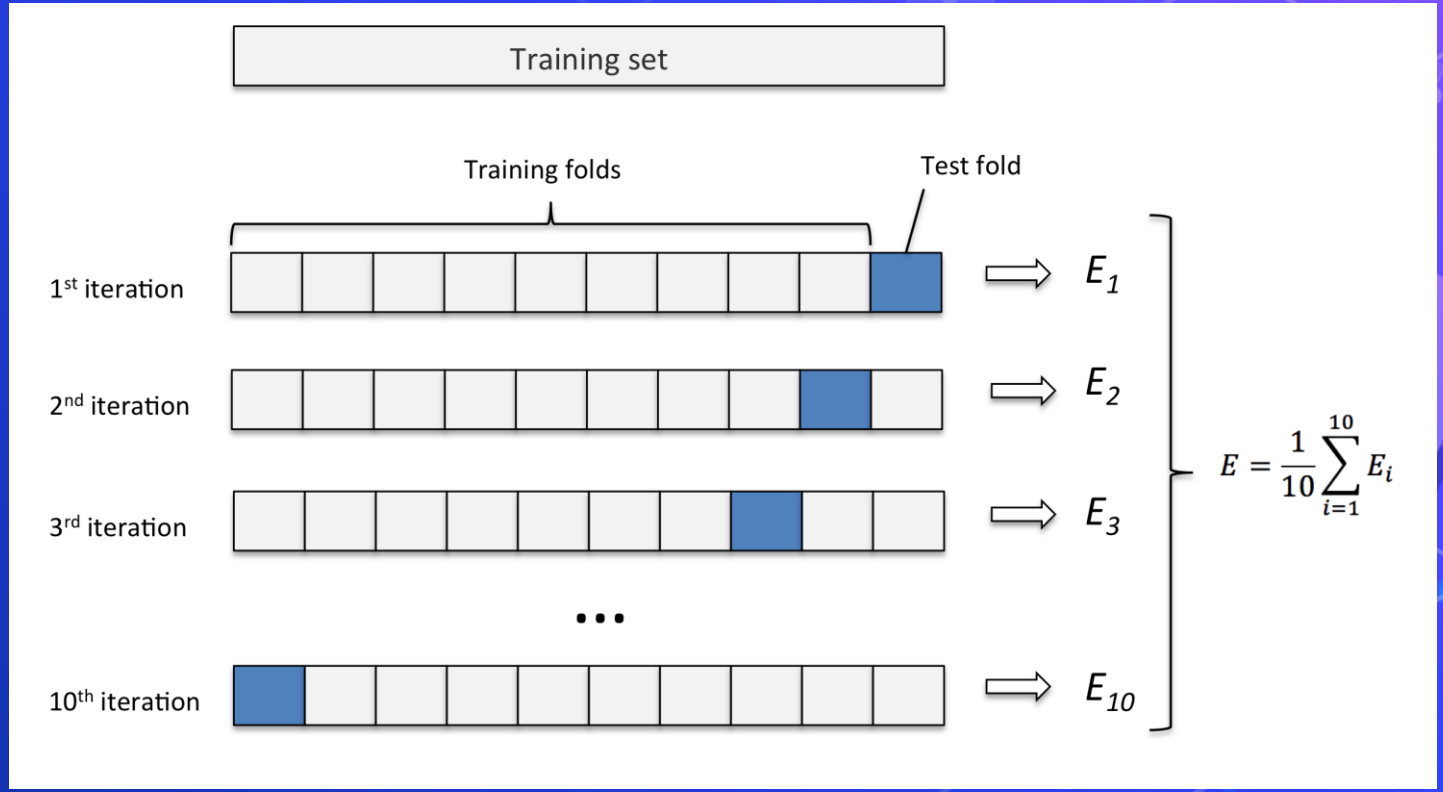
- What is Machine Learning
- Supervised Learning
  - Regression
  - Classification
- Unsupervised Learning
  - Clustering
  - Dimension Reduction
- **Model Selection & Evaluation**
  - **Cross Validation**
  - Hyperparameter Tuning
- Intro to Deep Learning

Cross Validation with K-Fold

Why we use Cross Validation

- Cross Validation uses **Stratified** technique to make sure the distribution of the categorical target is the same in every fold.
- More “efficient” use of data as every observation is used for both training and testing.
- Reduce overfitting and the model achieve the generalization.

Cross Validation with K-Fold



Cross Validation with K-Fold

```
from sklearn.svm import SVC
from sklearn.model_selection import cross_validate

svc = SCV()
cv_results_svc = cross_validate(svc, X, Y, cv=10, return_train_score=True)

cv_results_svc['test_score'].mean()
"""
0.8036085674097743
"""
```

Agenda

- What is Machine Learning
- Supervised Learning
  - Regression
  - Classification
- Unsupervised Learning
  - Clustering
  - Dimension Reduction
- **Model Selection & Evaluation**
  - Cross Validation
  - **Hyperparameter Tuning**
- Intro to Deep Learning

Hyperparameter Tuning

- Grid Search
- Randomized Search

```python
from sklearn.model_selection import GridSearchCV


params = [
        {'C':[1, 10, 100], 'kernel':['linear', 'sigmoid', 'poly']},
        {'C':[1, 10, 100], 'kernel':['rbf'], 'gamma':[0.5, 0.6, 0.7, 0.1, 0.01, 0.01]}
         ]

grid_search = GridSearchCV(estimator=model,
                           param_grid=params,
                           scoring='accuracy',
                           cv=10)
grid_search.fit(x_train, y_train)


print(grid_search.best_params_)
print(grid_search.best_params_)
```

Hyperparameter Tuning

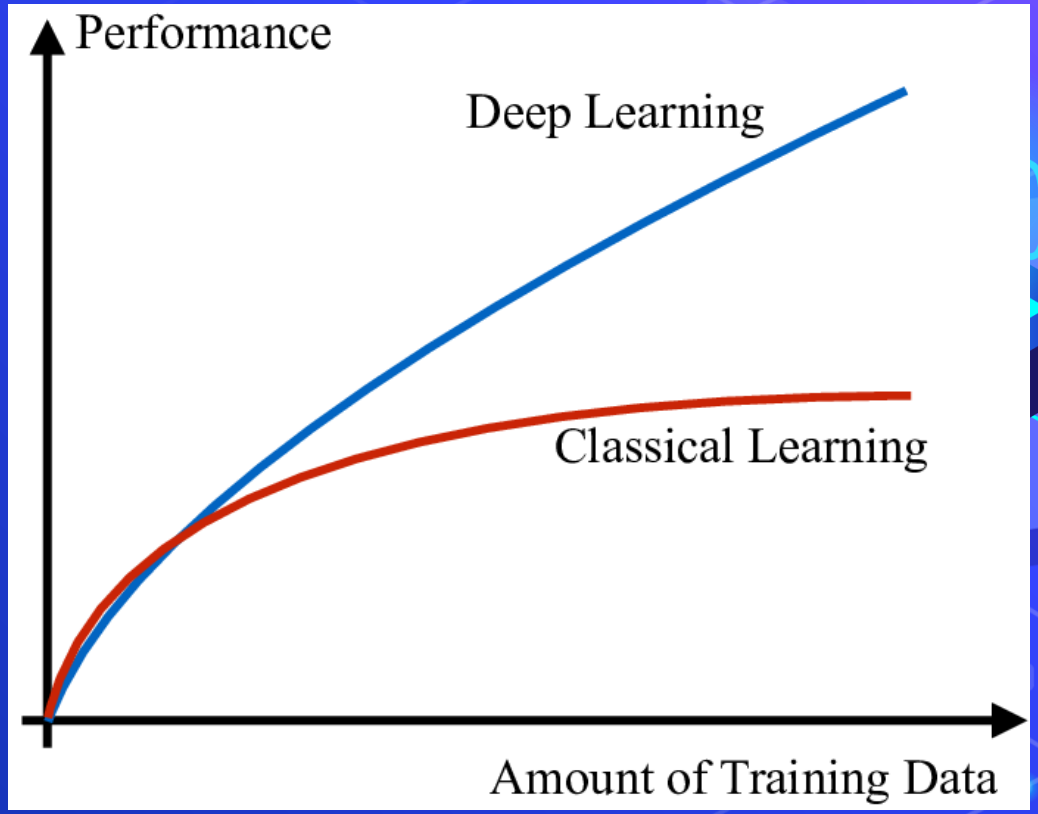
- Grid Search
- Randomized Search

```python
from sklearn.model_selection import RandomizedSearchCV


params = [
        {'C':[1, 10, 100], 'kernel':['linear', 'sigmoid', 'poly']},
        {'C':[1, 10, 100], 'kernel':['rbf'], 'gamma':[0.5, 0.6, 0.7, 0.1, 0.01, 0.01]}
         ]

randomized_search = RandomizedSearchCV(estimator=model,
                           param_grid=params,
                           scoring='accuracy',
                           cv=10)
randomized_search.fit(x_train, y_train)


print(randomized_search.best_params_)
print(randomized_search.best_params_)
```

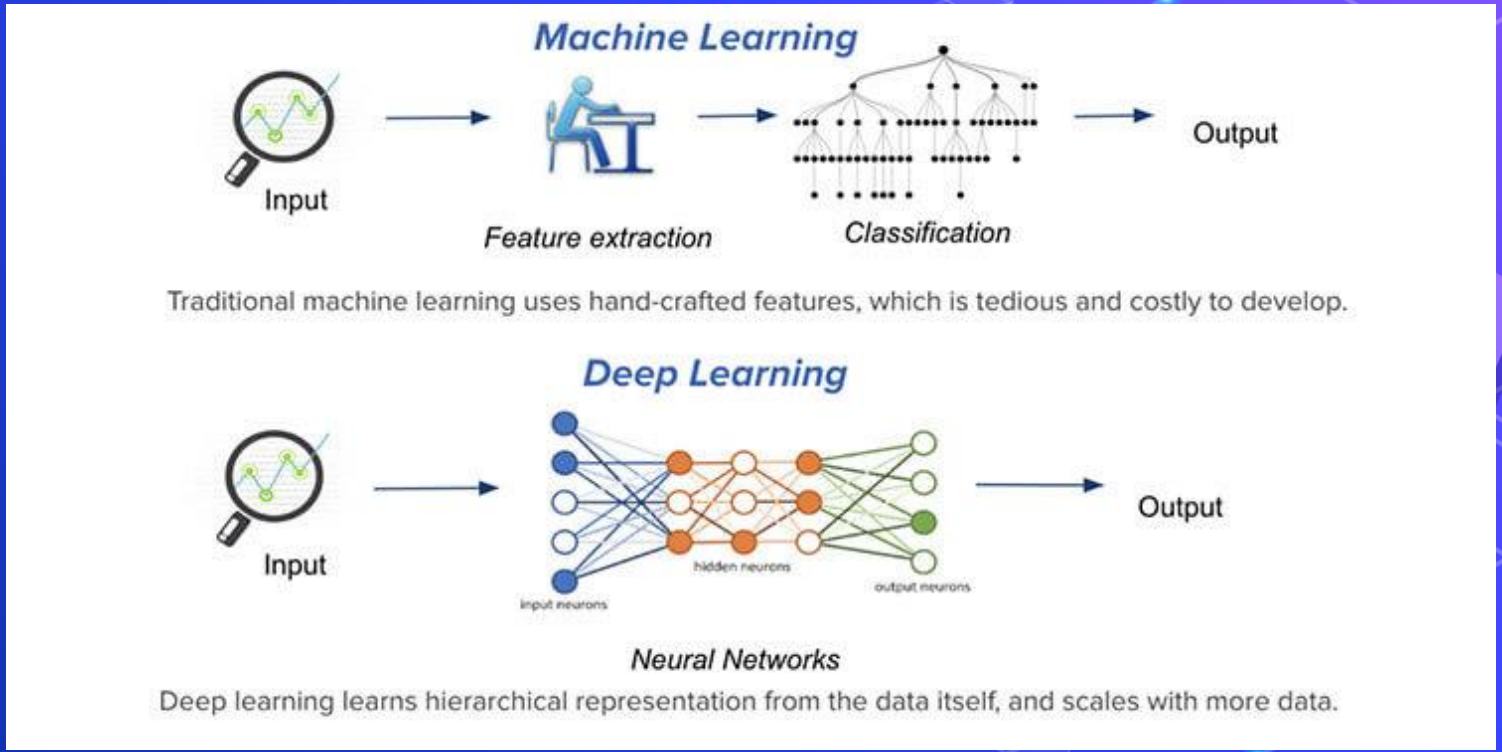
Agenda

- What is Machine Learning
- Supervised Learning
  - Regression
  - Classification
- Unsupervised Learning
  - Clustering
  - Dimension Reduction
- Model Selection & Evaluation
  - Cross Validation
  - Hyperparameter Tuning
- **Intro to Deep Learning**

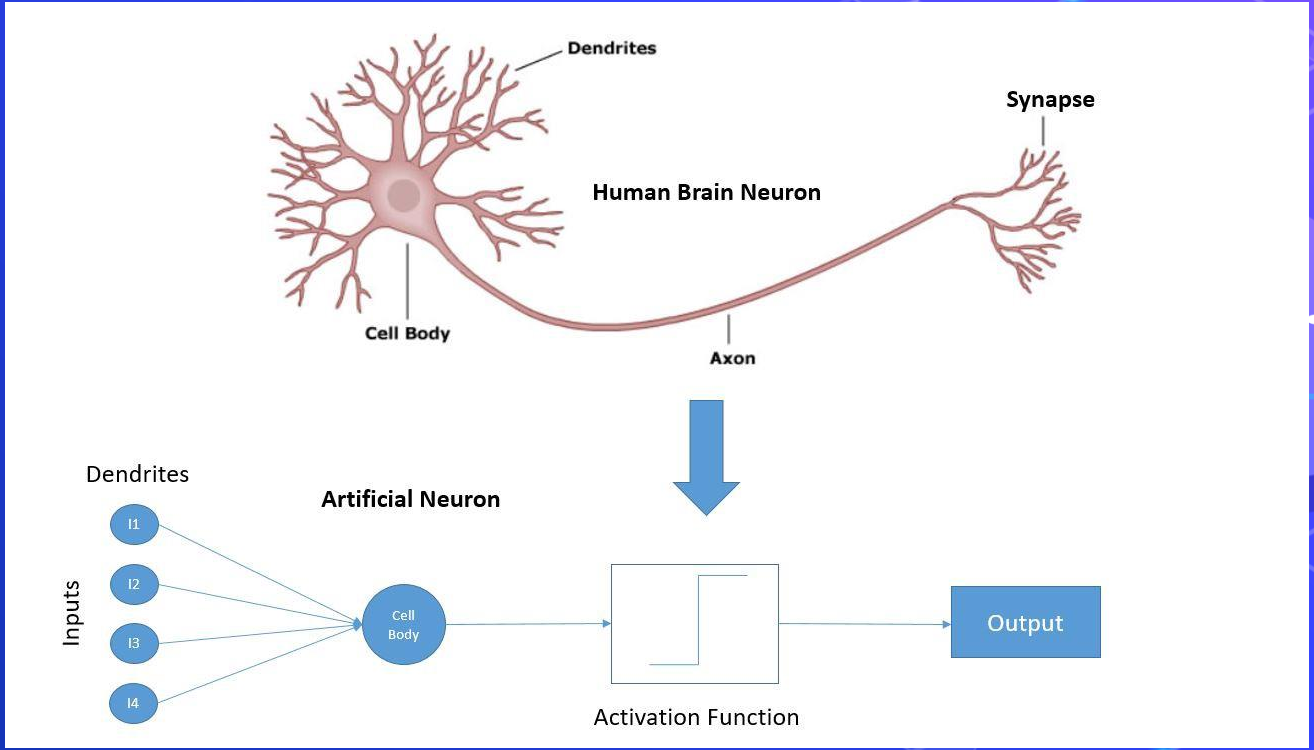
What is Deep Learning



What is Deep Learning



What is Deep Learning



What is Deep Learning

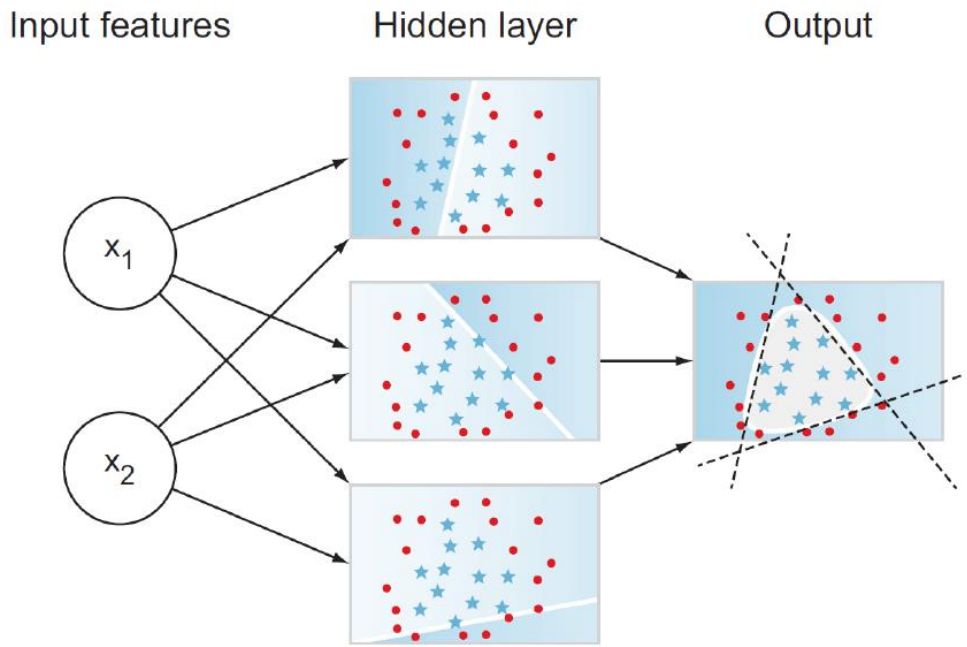


Figure 2.10

What is Deep Learning

Why is Deep Learning Hot Now?

Big Data Availability

facebook 350 millions images uploaded per day

Walmart 2.5 Petabytes of customer data hourly

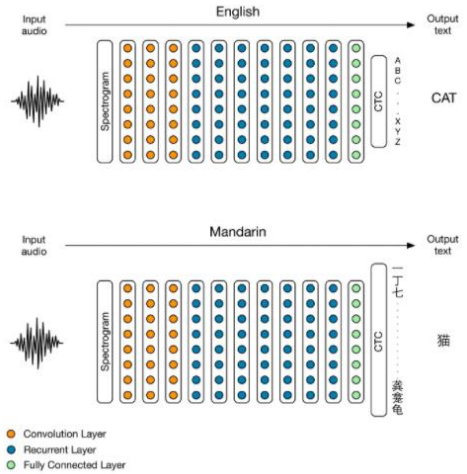
YouTube 300 hours of video uploaded every minute

New ML Techniques

GPU Acceleration

Deep Learning Applications

DL Today: Speech-to-Text



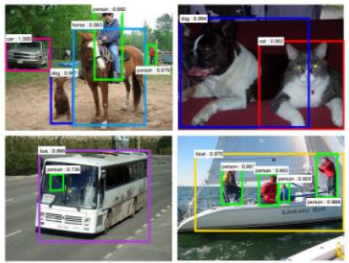
[Baidu 2014]

Deep Learning Applications

DL Today: Vision

[Krizhevsky 2012]

[Ciresan et al. 2013]



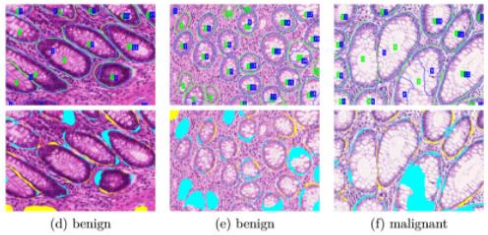
[Faster R-CNN - Ren 2015]

[NVIDIA dev blog]

Deep Learning Applications

DL Today: Vision

[Stanford 2017]



[Nvidia Dev Blog 2017]

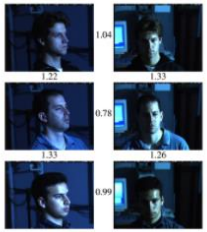


Figure 1. **Illumination and Pose invariance.**

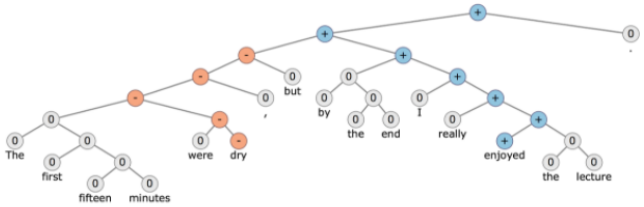
[FaceNet - Google 2015]

[Facial landmark detection CUHK 2014]

Deep Learning Applications

DL Today: NLP

[Google Translate System - 2016]



[Socher 2015]

Deep Learning Applications

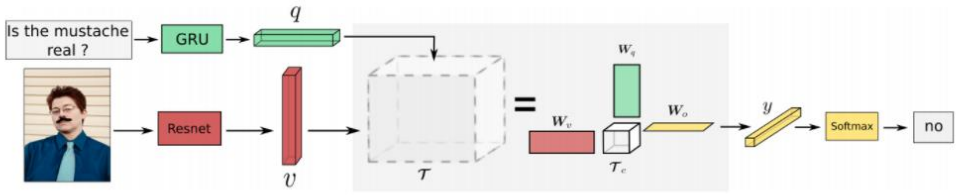
DL Today: NLP

[Google Inbox Smart Reply]

[Amazon Echo / Alexa]

Deep Learning Applications

DL Today: Vision + NLP



[VQA - Mutan 2017]

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"boy is doing backflip on wakeboard."

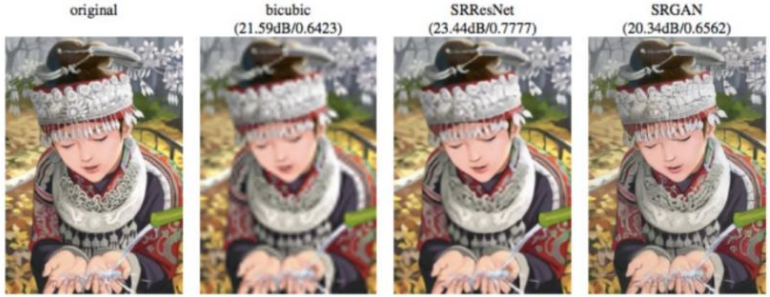
[Karpathy 2015]

Deep Learning Applications

DL Today: Image translation

[DeepDream 2015]

[Gatys 2015]



[Ledig 2016]

Deep Learning Applications

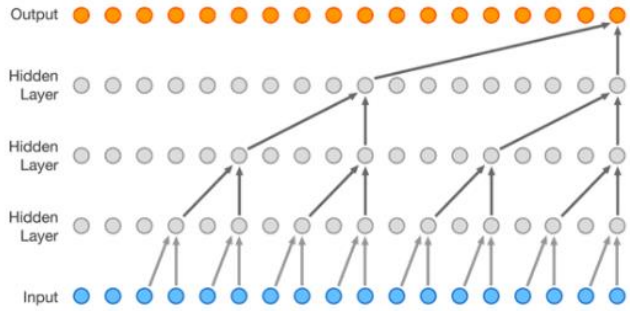
DL Today: Generative models

Sampled celebrities [Nvidia 2017]

StackGAN v2 [Zhang 2017]

Deep Learning Applications

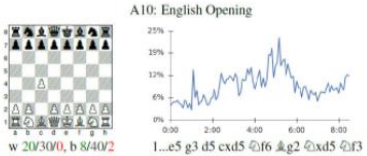
DL Today: Generative models



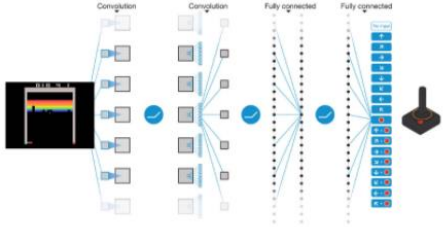
Sound generation with WaveNet [DeepMind 2017]

Deep Learning Applications

DL for AI in games



[Deepmind AlphaGo / Zero 2017]



[Atari Games - DeepMind 2016]

[Starcraft 2 for AI research]

Questions ?!

Thanks!

>_ Live long and prosper